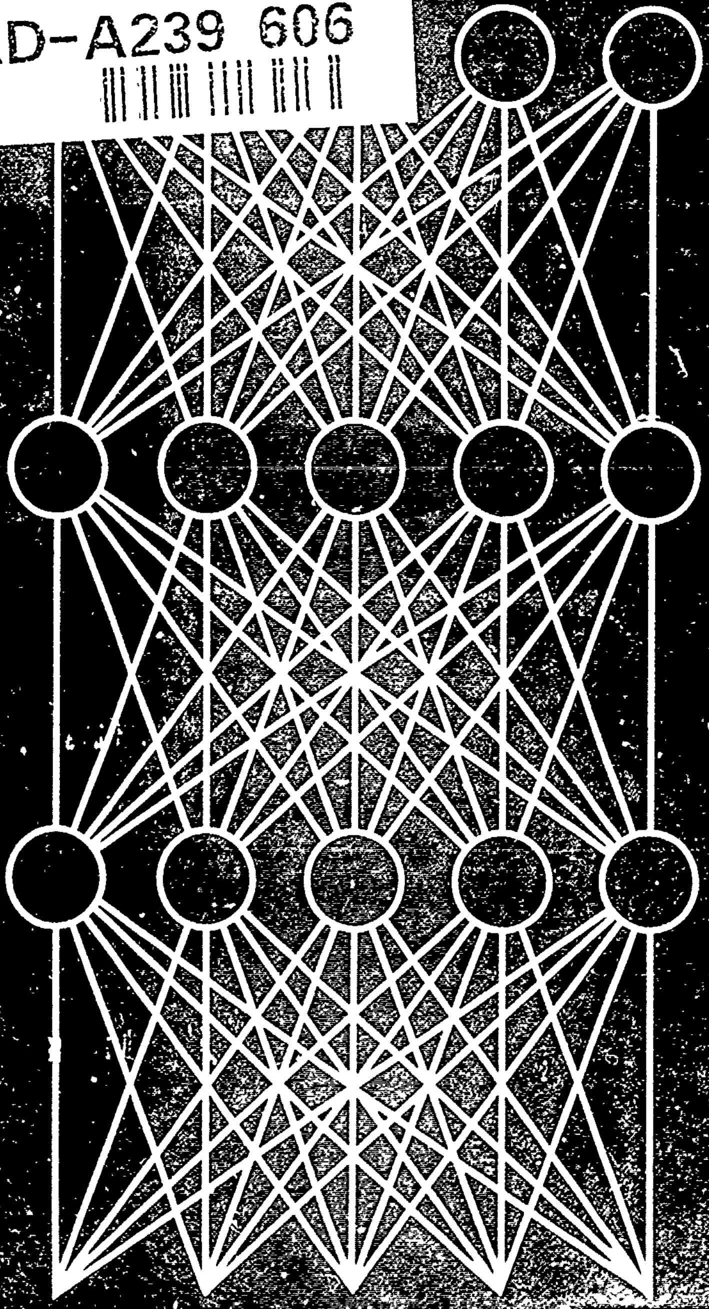
CONNECTIONIST MODELS

Proceedings of the 1990 Summer School



Edited by:
David S. Touretzky
Jeffrey L. Elman
Terrence J. Sejnowski
Geoffrey E. Hinton

CONNECTIONIST MODELS

Proceedings of the 1990 Summer School

N00014-90-J-1843

ONR

Edited by:

David S. Touretzky (Carnegie Mellon University)
Jeffrey L. Elman (University of California, San Diego)
Terrence J. Sejnowski (Salk Institute and University of California, San Diego)
Geoffrey E. Hinton (University of Toronto)

Morgan Kaufmann Publishers, Inc.
2929 Campus Drive, Suite 260
San Mateo, California 94403

Sponsoring Editor Bruce M. Spatz
Production Editor Sharon E. Montooth
Cover Designer Victoria Ann Philp
Typesetter Technically Speaking Publications

Library of Congress Cataloging-in-Publication Data

Connectionist models : proceedings of the 1990 summer school / edited by David S. Touretzky ... [et al.].
p. cm.
Proceedings of the Connectionist Models Summer School held at the University of California at San Diego.
Includes bibliographical references and index.
ISBN 1-55860-156-2
1. Connection machines--Congresses. 2. Neural networks (Computer science)--Congresses. I. Touretzky, David S. II. Connectionist Models Summer School (1990 : University of California, San Diego)
QA76.5.C61938 1991
006.3--dc20 90-21144
CIP

ISBN 0-55860-156-2
MORGAN KAUFMANN PUBLISHERS, INC.
2929 Campus Drive, Suite 260
San Mateo, California 94403

ABCDEFGHIJK-RP-93210

CONTENTS

Part I MEAN FIELD, BOLTZMANN, AND HOPFIELD NETWORKS

Part II REINFORCEMENT LEARNING

Part III GENETIC LEARNING

Part IV TEMPORAL PROCESSING

Part IX BIOLOGY

Part X VLSI IMPLEMENTATION

Foreword

The forty papers in this volume exemplify the tremendous breadth of research under way in the field of connectionist modeling. The interests of the summer school students and faculty range from theoretical analysis of networks to empirical investigations of learning algorithms, from speech and image processing to cognitive psychology, and from computational neuroscience to VLSI design. The papers selected for the proceedings offer an intense, pithy snapshot of the state of the art in 1990.

When the first Connectionist Models Summer School was held at Carnegie Mellon in 1986, one of its goals was to help promising young graduate students at institutions with few faculty working in the neural nets area. The 1986 summer school attendees were in a sense pioneers. At that time neural nets had not yet attained their present status as a "hot" research topic; the Rumelhart and McClelland PDP books weren't even in print yet. Spirits ran high during the event, lasting friendships were formed, and some interesting research was done on site.

By the time of the second summer school in 1988, the field had advanced markedly. That year we saw three U.S. neural net conferences and half a dozen workshops. The first neural net journal appeared, and a second was announced. The raw pioneering spirit of the first summer school was perhaps diminished somewhat, but in compensation, the level of sophistication of the students was clearly increased, and the enthusiasm level was as high as ever. After the event was over, we found that the students had been holding extra sessions in their rooms after dinner, continuing sometimes until well past midnight.

We decided that the next summer school should be held on the west coast. Since Terry Sejnowski was moving to UCSD, La Jolla was the obvious choice of venue. Jeff Elman kindly volunteered to serve as organizer for the 1990 event. The rest of us (Sejnowski, Touretzky, and Hinton) were his advisory committee, and collaborated in the production of this proceedings. Each student was assigned one of us as his or her editor, based either on a draft paper submitted prior to the summer school or our knowledge of the student's interests. The papers they submitted were then revised in response to detailed comments. The quality of the results is extremely high.

At the completion of this year's summer school we decided it had been as successful as its predecessors, but it had more of a workshop flavor and less that of a tutorial. The faculty of past summer schools found the workshop aspects much to their liking, but the change this year was really a reflection of the advanced nature of the students, who are already pursuing interesting research on a wide variety of topics. They are among the best and the brightest in the neural nets game. Pictures of some of the attendees, taken by Timor Ash, appear at the back of the proceedings.

We would like to thank UCSD, the Office of Naval Research, the McDonnell-Pew Foundation, and the MacArthur Foundation for sponsoring this event. Their generous support was important to making the summer school a success. We are also grateful to Marilee Bateman and Sondra Buffett of UCSD for excellent administrative support, and to the grad students at UCSD for providing hospitality to the students and faculty. Kim Plunkett did yeoman duty in helping coordinate the logistical arrangements. We would also like to thank Bruce Spatz and Sharon Montooth at Morgan Kaufmann for helping to make these proceedings possible.

Finally, we express our deep appreciation to the faculty of the summer school. The faculty's dedication and commitment to the scientific training of graduate students was apparent throughout. Without their participation, the CMSS would not exist.

DAVID S. TOURETZKY
Carnegie Mellon

JEFFREY L. ELMAN
University of California, San Diego

TERRENCE J. SEJNOWSKI
The Salk Institute, and
University of California, San Diego

GEOFFREY E. HINTON
University of Toronto

Participants in the 1990 Connectionist Models Summer School

Organizer

Jeffrey L. Elman

Advisory Commitee

Geoffrey E. Hinton
Terrence J. Sejnowski
David S. Touretzky

Faculty

Andrew Barto, University of Massachusetts
Richard Belew, University of California, San Diego
Patricia Churchland, University of California, San Diego
Garrison Cottrell, University of California, San Diego
Jack Cowan, University of Chicago
Richard Durbin, Stanford University
Jeffrey Elman (Organizing Committee), University of California, San Diego
Jerome Feldman, International Computer Science Institute
David Haussler, University of California, Santa Cruz
Geoffrey Hinton (Organizing Committee), University of Toronto
Michael Jordan, Massachusetts Institute of Technology
Shawn Lockery, University of California, San Diego
James McClelland, Carnegie Mellon University
George Lakoff, University of California, Berkeley
Carver Mead, California Institute of Technology
Janet Metcalfe, University of California, San Diego
Randal Nelson, University of Rochester
David Rumelhart, Stanford University
Terrence Sejnowski (Organizing Committee), Salk Institute
Martin Sereno, University of California, San Diego
Al Selverston, University of California, San Diego
Paul Smolensky, University of Colorado
David Touretzky (Organizing Committee), Carnegie Mellon University
Alex Waibel, Carnegie Mellon University
Ron Williams, Northeastern University
David Zipser, University of California, San Diego

List Of Accepted Students

Alston, Michael, University of California, San Diego
Bailly, Gerard, Institut de la Communication Parlee
Barto, Andrew G., University of Massachusetts
Bennett, David, Brown University
Burani, Cristina, Istituto Di Psicologia
Carlin, Michael J., Naval Ocean Systems Center
Camporese, Daniel S., University of British Columbia
Chalmers, David J., Indiana University
Chau, Paul, University of California, San Diego
Chen, James R., University of California, San Diego
Chrisley, Ronald L., New College
Cohn, David, University of Washington
Cottrell, Garrison W., University of California, San Diego
Crowder, R. Scott, Carnegie Mellon University
Day, Shawn P., University of British Columbia
Dayan, Peter, University of Edinburgh
Doutriaux, Antoine, University of California, San Diego
Galland, Conrad C., University of Toronto
Goebel, Rainer, University of Braunschweig
Goodhill, Geoffrey J., University of Sussex
Guest, Clark C., University of California, San Diego
Gullapalli, Vijaykumar, University of Massachusetts
Hampshire, John B., Carnegie Mellon University
Hinton, Geoffrey E., University of Toronto
Heirich, Alan, California Institute of Technology
Holyoak, Keith J., University of California, Los Angeles
Huberman, Bernardo A., Xerox PARC
Intrator, Nathan, Brown University
Jennings, Peggy J., University of Oregon
Keele, Steven W., University of Oregon
Knudsen, Steen L., University of Aarhus
Koch, Christof, California Institute of Technology
Laboissiere, Rafael, Institut de la Communication Parlee
Lange, Trent, University of California, Los Angeles
Levin, Asriel, Yale University
Marchman, Virginia, University of California, San Diego
Melz, Eric R., University of California, Los Angeles
Miller, Geoffrey F., Stanford University
Movellan, Javier R., Carnegie Mellon University
Nolfi, Stefano, Istituto Di Psicologia Del C. N. R.
Ossen, Arnfried, Technical University of Berlin
Parisi, Domenico, Istituto Di Psicologia Del C. N. R.
Pearlmutter, B., Carnegie Mellon Univesity
Pinkas, Gadi, Washington University
Plunkett, Kim, University of Aarhus
Rose, Daniel E., University of California, San Diego
Rumelhart, David E., Stanford University
Sanger, Terence D., Massachusetts Institute of Technology
Schmidhuber, Jurgen, Universitat Munchen
Schwartz, J. L., Institut de la Communication Parlee
Schyns, Philippe G., Brown University
Shimabukuro, Randy L., Naval Ocean Systems Center
Shoemaker, Patrick, Naval Ocean Systems Center

Singh, Satinder Pal, University of Massachusetts
Todd, Peter M., Stanford University
Touretzky, David S., Carnegie Mellon University
Tsung, Fu-Sheng, Univesity of California, San Diego
Vallar, Giuseppe, Universita Di Milano
Wan, Eric A., Stanford University
Watkins, Steven, University of California, San Diego
Weigend, Andreas S., Stanford University
Wieland, Alexis, University of California, Los Angeles
Wharton, Charles M., University of California, Los Angeles
Williams, Christopher K.I., University of Toronto
Zipser, David, University of California, San Diego

Part I

Mean Field, Boltzmann, and Hopfield Networks

Deterministic Boltzmann Learning in Networks with Asymmetric Connectivity

Conrad C. Galland
Physics Dept.
University of Toronto
Toronto, Canada
M5S 1A7

Geoffrey E. Hinton
Computer Science Dept.
University of Toronto
Toronto, Canada
M5S 1A4

Abstract

The simplicity and locality of the "contrastive Hebb synapse" (CHS) used in Boltzmann machine learning makes it an attractive model for real biological synapses. The slow learning exhibited by the stochastic Boltzmann machine can be greatly improved by using a mean field approximation and it has been shown (Hinton, 1989) that the CHS also performs steepest descent in these deterministic mean field networks. A major weakness of the learning procedure, from a biological perspective, is that the derivation assumes detailed symmetry of the connectivity. Using networks with purely asymmetric connectivity, we show that the CHS still works in practice provided the connectivity is grossly symmetrical so that if unit i sends a connection to unit j, there are numerous indirect feedback paths from j to i. So long as the network settles to a stable state, we show that the CHS approximates steepest descent and that the proportional error in the approximation can be expected to decrease as the size of the network increases.

1 INTRODUCTION

The learning procedures used to train connectionist networks are often criticized for having only superficial relevance to real neural systems. The most widely used procedure, back-propagation, requires an elaborate apparatus to correctly determine and distribute the necessary error gradient information to the different weights of the net. When the sort of complex feedback wiring common in real neural systems is incorporated, the learning becomes even more involved (Pineda, 1987) and no attempt is made to justify it biologically.

The CHS, on the other hand, yields the correct error derivatives using only information local to each weight, and it can be applied readily to networks with large degrees of feedback. However, based on the original theory (Hinton and Sejnowski, 1986) symmetrical connectivity appears necessary for the CHS to perform gradient descent, and there is no evidence of such detailed symmetry in the CNS.

2 DETERMINISTIC BOLTZMANN LEARNING IN SYMMETRIC NETWORKS

A mean field approximation (Glauber, 1963; Hopfield, 1984) was used by Peterson and Anderson (1987) to replace the stochastic units of the original Boltzmann machine with deterministic analog units, resulting in a significant improvement in learning efficiency. Details of basic DBM theory can be obtained from Hinton (1989). In brief, a DBM with unit states in the range $[-1, 1]$ has mean field free energy

$$\begin{aligned} F &= E - TH \\ &= -\sum_{i<j} y_i y_j w_{ij} + T \sum_i \Big[\frac{(1+y_i)}{2} \log \frac{(1+y_i)}{2} \\ &\qquad + \frac{(1-y_i)}{2} \log \frac{(1-y_i)}{2} \Big] \end{aligned} \tag{1}$$

where E and H are the mean field energy and entropy. The DBM settles to a low temperature minimum, F^*, where the states of the units are given by

$$y_i = \tanh(\frac{1}{T} \sum_j y_j w_{ij}) \tag{2}$$

Detailed symmetry of the weights is assumed such that $w_{ij} = w_{ji}$.

Treating the weights as independent, the total derivative of F^* with respect to a particular weight is given by

$$\frac{dF^*}{dw_{ij}} = \frac{\partial F^*}{\partial w_{ij}} + \sum_k \frac{\partial F^*}{\partial y_k} \cdot \frac{\partial y_k}{\partial w_{ij}} \qquad (3)$$

The first term represents the effect of a weight change on the free energy with all unit states held constant. The second term represents the effect on F^* of a weight change via the resultant changes in the equilibrium states of the units. However, at the minimum, $\partial F^* / \partial y_k = 0$ for all k, so that the second term vanishes, leaving

$$\frac{dF^*}{dw_{ij}} = -y_i y_j \qquad (4)$$

If the probability of an output vector, O_β, given an input vector, I_α, is defined as

$$P^-(O_\beta|I_\alpha) = \frac{e^{-F^*_{\alpha\beta}/T}}{e^{-F^*_\alpha/T}} \qquad (5)$$

where $F^*_{\alpha\beta}$ is the minimum free energy settled to with I_α and O_β clamped (positive phase) and F^*_α the minimum free energy with just I_α clamped (negative phase), then the CHS can be seen to change each weight in proportion to the gradient of $\log P^-(O_\beta|I_\alpha)$. Assuming $T = 1$,

$$\Delta w_{ij} = \epsilon(y_i^+ y_j^+ - y_i^- y_j^-) = \epsilon \frac{\partial \log P^-(O_\beta|I_\alpha)}{\partial w_{ij}} \qquad (6)$$

where y_i^+ and y_i^- refer to the states of unit i at the minimum settled to in the positive and negative phases. Hence, by making ϵ small enough, the CHS will approximate gradient descent in the objective function G.

$$G = \sum_{\alpha,\beta} P^+(I_\alpha, O_\beta) \log \frac{P^+(O_\beta|I_\alpha)}{P^-(O_\beta|I_\alpha)} \qquad (7)$$

3 ASYMMETRIC NETWORKS

The assumption that $w_{ij} = w_{ji}$ is not necessary in order to define an energy function over the network. The symmetry is required to allow each unit to compute the derivative of a global quadratic energy function from locally available information. Without symmetry, the natural dynamics of the activity levels do not consist of following the gradient of the energy function.

Hopfield (1982) points out that nets with asymmetric weights such that $w_{ij} \neq w_{ji}$ can be described as having a symmetric part, which defines the energy function, plus an anti-symmetric part, which can be considered as noise during settling. However, in nets with completely asymmetric connectivity, where only one of w_{ij} and w_{ji} is allowed to exist, the anti-symmetric and symmetric parts are of equal magnitude. Simulations demonstrate that deterministic networks with randomly asymmetric connectivity typically settle to a stable state, and that the anti-symmetric part of the weight matrix affects which state is settled to.

3.1 THE CHS IN NETWORKS WITH ASYMMETRIC CONNECTIVITY

In networks with detailed symmetry of the *connectivity*, the CHS, coupled with weight decay, will automatically symmetrize initially asymmetric weights due to the inherent symmetry of the learning procedure (Hinton, 1989). It is not equally apparent why the CHS should do anything sensible when applied in asymmetrically connected nets.

Consider the free energy of an asymmetrically connected network with the energy term defined using just the symmetric part of the weight matrix:

$$F = -\frac{1}{2} \sum_{i<j} (y_i y_j w_{ij} + y_i y_j w_{ji}) - TH \qquad (8)$$

In the calculation of F, all asymmetric weights that come from permanently clamped units, like the inputs, can simply be treated as symmetric, so that $w_{ij} = w_{ji}$. For all other connections, only one of w_{ij} and w_{ji} is non-zero, and the contribution to the energy term is thus half that of the effectively symmetric weights. This follows from the fact that an asymmetric connection of weight w_{ij} can be thought of as having a symmetric part of $\frac{1}{2}w_{ij}$, as well as an anti-symmetric part of equal magnitude.

The dynamics of this network will typically consist of settling to a stable state where equation (2) holds for each non-clamped unit, and we designate the free energy of the network at this point as F^*. However, this stable state will no longer be at a minimum of the free energy, so that $\partial F^* / \partial y_i \neq 0$. Expressed another way, at the fixed point (assuming $T = 1$)

$$\frac{\partial E}{\partial y_i} \neq \frac{\partial H}{\partial y_i} \qquad (9)$$

However, we can say that

$$\frac{\widehat{\partial E}}{\partial y_i} = \frac{\partial H}{\partial y_i} \qquad (10)$$

The left-hand side of equation (10) represents the actual total input to unit i, and it can be considered the unit's estimate of the true $\partial E / \partial y_i$ derivative. Figure 1 shows the true values and unit estimates of this derivative for a simple network.

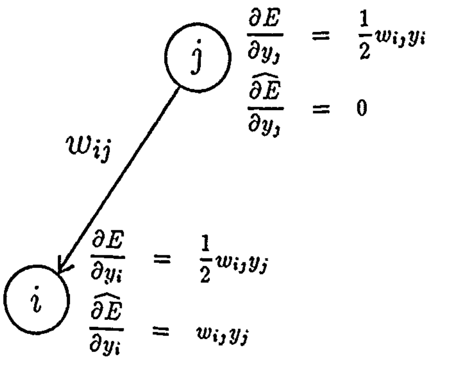

j
$\frac{\partial E}{\partial y_j} = \frac{1}{2} w_{ij} y_i$
$\frac{\widehat{\partial E}}{\partial y_j} = 0$
w_{ij}
i
$\frac{\partial E}{\partial y_i} = \frac{1}{2} w_{ij} y_j$
$\frac{\widehat{\partial E}}{\partial y_i} = w_{ij} y_j$

Figure 1: The true, $\frac{\partial E}{\partial y_i}$, and estimated, $\frac{\widehat{\partial E}}{\partial y_i}$, values of the energy derivative for two asymmetrically connected units.

We will use the F^* of an asymmetrically connected net to define the probability of an input vector given an output vector as in equation (5), and then investigate how close the CHS approaches gradient descent in the objective function with the probabilities so defined. Proceeding as in equation (3), the total derivative of F^* with respect to a particular weight is given by

$$\begin{aligned} \frac{dF^*}{dw_{ij}} &= \frac{\partial F^*}{\partial w_{ij}} + \sum_k \frac{\partial F^*}{\partial y_k} \cdot \frac{\partial y_k}{\partial w_{ij}} \\ &= -s y_i y_j + \sum_k \left(\frac{\partial E}{\partial y_k} - \frac{\widehat{\partial E}}{\partial y_k} \right) \frac{\partial y_k}{\partial w_{ij}} \quad (11) \end{aligned}$$

where s is 1 for a symmetric weight and $\frac{1}{2}$ for an asymmetric weight. The difference between the true and estimated energy derivatives is zero for a symmetric network, and the CHS predicted weight changes perform exact gradient descent. For an asymmetric network, this difference is not zero, and the second term in equation (11) is ignored by the CHS. The larger the proportion of weights that can be considered symmetrical, the smaller this difference will be, and the closer the CHS will come to performing gradient descent. For a given fan-in, f_a, of truly asymmetric connections, the difference between the true and estimated values of the energy derivatives for a particular unit can be expected to scale as

$$\left| \frac{\partial E}{\partial y_k} - \frac{\widehat{\partial E}}{\partial y_k} \right| = k \, \langle |w_{ij}| \rangle \, \sqrt{f_a} \qquad (12)$$

where k is a proportionality constant, and $\langle |w_{ij}| \rangle$ is the expected value of the magnitude of the weights. The $\sqrt{f_a}$ term arises because the various $\frac{1}{2} w_{ij} y_j$ terms can be expected to be random in sign, and hence to sum as in a random walk.

We would like to determine how the error between the true and CHS estimated gradients scales with increasing fan-in. Following all weight changes in either the positive or negative phase, the resultant change in the free energy due to the first term in equation (11) is of order

$$|\Delta F^*_{CHS}| = k_1 \, \langle y_i^2 y_j^2 \rangle \, (\frac{1}{2} f_a + f_s) u \qquad (13)$$

where f_s is the symmetric weight fan-in, u is the total number of unclamped units, and k_1 is some constant. Using equation (12), the change in the free energy due to the random effects of the weight changes via the second term of equation (11) is of order

$$|\Delta F^*_{err}| = k_2 \, \langle |\Delta y_i| \rangle \, \langle |w_{ij}| \rangle \, \sqrt{f_a} \, \sqrt{u} \qquad (14)$$

where $\langle |\Delta y_i| \rangle$ is the expected value of the change in the stable state activities of the units as a result of all the weight changes, and k_2 is some constant.

In order to maintain the same degree of "hardness" in the network as the fan-in increases, it is necessary to keep constant the expected value of the magnitude of the total input to a unit. It would then be expected that $\langle |w_{ij}| \rangle$ would scale as $1/\sqrt{f}$ where f is the total fan-in, and that $\langle y_i^2 y_j^2 \rangle$ would be scale independent. Thus, we obtain

$$\begin{aligned} |\Delta F^*_{CHS}| &= k_{1'} (\frac{1}{2} f_a + f_s) u \\ |\Delta F^*_{err}| &= k_{2'} \, \langle |\Delta y_i| \rangle \, \frac{\sqrt{f_a}}{\sqrt{f}} \sqrt{u} \end{aligned}$$

In symmetric networks, the effects of changes in the stable state values of the units can be neglected because the stable state is a local minimum of the free energy. This is not the case for asymmetric nets, and the difficult question of how $\langle |\Delta y_i| \rangle$ scales must be addressed. There are three plausible regimes:

- If $\langle |\Delta y_i| \rangle$ does not depend on the number of connections, then (assuming $\frac{f_a}{f}$ is constant)

$$r = \frac{|\Delta F^*_{err}|}{|\Delta F^*_{CHS}|} \propto \frac{\sqrt{f_a}}{(\frac{1}{2} f_a + f_s)\sqrt{f}\sqrt{u}} \propto \frac{1}{(\frac{1}{2} f_a + f_s)\sqrt{u}}$$

- If $\langle |\Delta y_i| \rangle$ scales as $\sqrt{f}$, then

$$r \propto \frac{\sqrt{f_a}}{(\frac{1}{2} f_a + f_s)\sqrt{u}} \propto \frac{1}{\sqrt{(\frac{1}{2} f_a + f_s)}\sqrt{u}}$$

- If $\langle |\Delta y_i| \rangle$ scales as f, then

$$r \propto \frac{\sqrt{f_a}}{\sqrt{f}\sqrt{u}} \propto \frac{1}{\sqrt{u}}$$

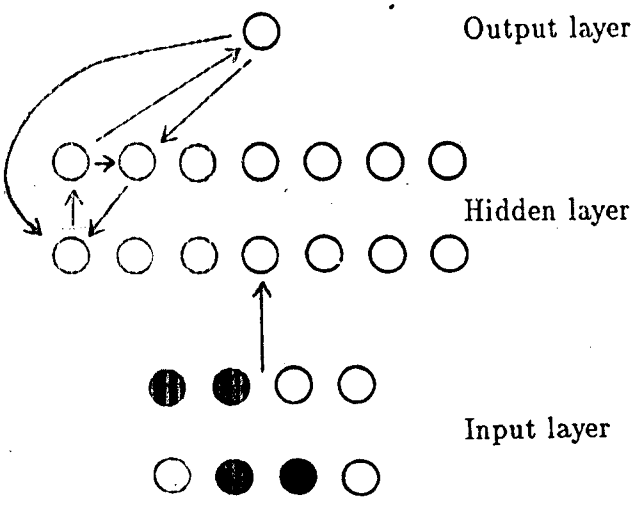


Figure 2: The network architecture for the asymmetric 4-bit shifter is shown including possible hidden-to-hidden, hidden-to-output, and output-to-hidden connections for three of the hidden units. 24 hidden units are used in the 8-bit shifter, and 40 in the 16-bit shif

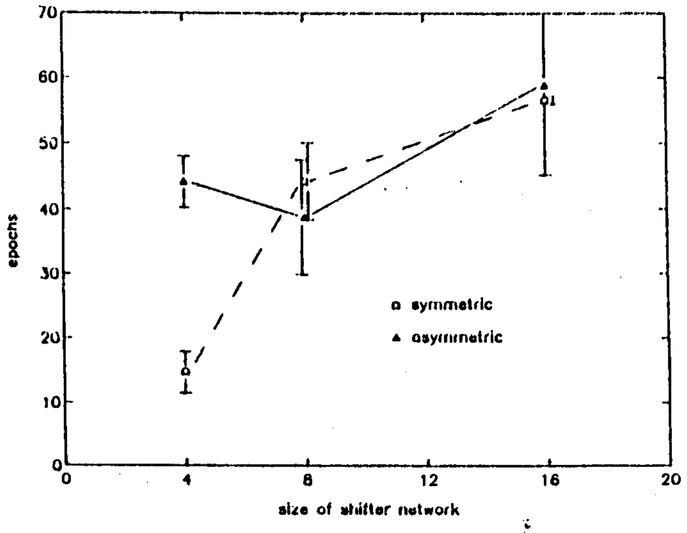


Figure 3: Graph showing the number of training epochs required to perfectly learn the task in various sized shifters. Since the number of possible input vectors increases dramatically with the size of the shifter, the number of presentations per epoch is much higher for the larger networks.

The success of the CHS in simulations of asymmetric networks suggests that, for certain architectures, the erroneous side-effects can indeed be neglected. The second regime seems most likely, and its scaling prediction is consistent with performance results in networks of increasing size. As detailed in the next section, simulations have also shown that, in the networks tested, the CHS consistently chooses a trajectory in weight space that has a strongly positive cosine with the true gradient which can be computed by taking into account the second term of equation (11).

4 SIMULATION RESULTS

The CHS was applied to asymmetrically connected networks to solve the shifter problem. This is a non-trivial second order problem that cannot be solved without a hidden layer. The network architecture is shown in figure 2. There is full connectivity from the input to the hidden layer, and for *every* pair of non-input units, i and j, exactly one of w_{ij} and w_{ji} is chosen at random to exist. This extensive asymmetric connectivity permits many possible closed loops involving the hidden and output layers.

The task is to detect the sense of a shift between two binary vectors where the second vector is generated by shifting the first a single "pixel" to the right or left using wraparound. In the positive phase, a binary vector and its left or right shifted image are clamped on the inputs, with the output unit being correspondingly clamped 'off' for left and 'on' for right. In the negative phase, only the inputs are clamped. This is repeated for each case, with the weights changed after each presentation. This "on-line" training technique was used in order to avoid the biologically less plausible storage of gradients across all the training cases.

As can be seen from figure 3, the CHS successfully trained various sized, asymmetrically connected, shift-detecting networks. The learning performance was compared with that obtained in symmetrical networks of roughly the same complexity in terms of the number of weights and units. Since each pair of units is effectively connected by two weights in the symmetric case, the architecture of the symmetric shifters had the same number of units in the same basic arrangement as the corresponding asymmetric nets, but with half of all the possible hidden-to-hidden and hidden-to-output symmetric connections chosen at random.[1] Figure 3 shows how the difference in learning performance decreases with increasing network size, suggesting a decrease in the proportional error of the gradient approximation as proposed in the previous section. The error bars were obtained by running repeated experiments on networks with different random connectivity patterns. Details of the simulations are given in the Appendix.

[1] It is difficult to decide whether to compare with a symmetric net having the same number of effects, as we have chosen, or the same number of degrees of freedom. In fact, increasing the number of hidden-to-hidden connections in the symmetric net significantly slows learning, and it is fastest with no such connections at all.

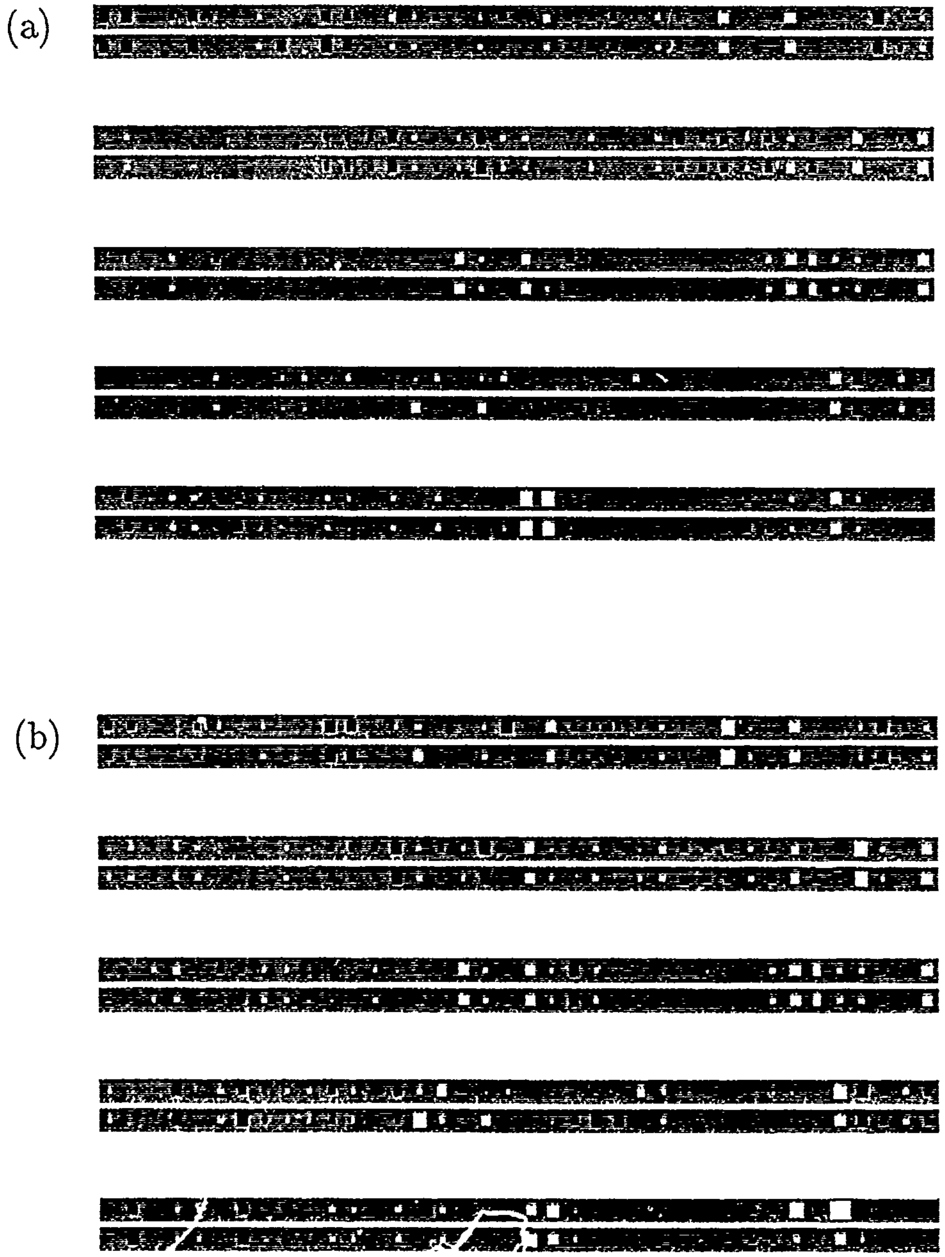


Figure 4. The true and estimated dF^*/dw_{ij} gradients are displayed, in arbitrary order, for an asymmetric 4-bit shifter late in learning, in (a) the positive and (b) the negative phases. The top and bottom rows of each pair represent the true and CHS estimated values respectively. The size and colour of each square represent the magnitude and sign of the gradient for each weight averaged over all cases.

In order to test the accuracy of the estimated gradients determined by the CHS, the true dF^*/dw_{ij} gradients were computed in both phases by measuring the change in the stable point free energy following perturbations to each weight. The true gradient thus incorporates the effects of the changing stable state activities of the units indicated in equation (11). Figure 4 shows the true and estimated gradients averaged over all training cases for an asymmetric 4-bit shifter in both the positive and negative phases. Clearly there is good agreement, and the cosine between the two gradients is typically strongly positive for the majority of the learning process.

Simulations were performed to test whether the antisymmetric part of the weight matrix could simply be ignored. After an asymmetric network had been successfully trained on the shifter problem, each asymmetric weight was replaced by a symmetric weight of half the magnitude, effectively removing the antisymmetric component from each connection. When tested on the training set, the new network gave results little better than chance.

An attempt was also made to train the shifter networks to perform a different task in which one of the two input vectors was treated as the output. That is, with both the reference vector and the shift bit clamped, the net was to produce the shifted vector. This re-arrangement of inputs and outputs markedly decreases the number of connections that can be considered symmetric, and thus increases the error in the CHS predicted gradients. As a result, all attempts to train the networks to learn this task were unsuccessful. Preliminary results suggest that problems with multiple output units in asymmetrically connected nets can be solved using the CHS so long as there are correlations between the output units across cases.

5 DISCUSSION

We have shown that the CHS is able to solve a difficult second order problem in asymmetrically connected networks. By removing the strict requirement of detailed symmetry, the CHS becomes considerably more attractive as a possible model of real biological synapses.

For asymmetrically connected networks, the antisymmetric part of the weight matrix affects the settling of the network and, hence, the required gradients. Although the CHS only performs exact gradient descent for a symmetric net, the effects of the random changes in the stable state activities of the units can be neglected in problems that have a significant number of effectively symmetric weights. Further research is needed to establish the magnitude of this unwanted effect, and precisely how it depends on the architecture of the network and the nature of the problem.

Acknowledgements

This research was supported by grants from the Ontario Information Technology Research Center, the Medical Research Council of Canada and the National Science and Engineering Research Council of Canada. Geoffrey Hinton is a fellow of the Canadian Institute for Advanced Research.

References

Glauber, R. J. (1963) Time-dependent statistics of the Ising Model. *Journal of Mathematical Physics*, 4:2, 294-307.

Hinton, G. E. (1989) Deterministic Boltzmann learning performs steepest descent in weight-space. *Neural Computation*, 1.

Hinton, G. E. and Sejnowski, T. J. (1986) Learning and relearning in Boltzmann machines. In Rumelhart, D. E., McClelland, J. L., and the PDP group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, MIT Press, Cambridge, MA.

Hopfield, J. J. (1982) Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences U.S.A.*, 79, 2554-2558.

Hopfield, J. J. (1984) Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences U.S.A.*, 81, 3088-3092.

Peterson, C. and Anderson, J. R. (1987) A mean field theory learning algorithm for neural networks. *Complex Systems*, 1, 995-1019.

Pineda, F. J. (1987) Generalization of backpropagation to recurrent neural networks. *Phys. Rev. Lett.*, 18, 2229-2232.

APPENDIX

Table 1 shows typical learning schedules used to train the various shifter networks, where ϵ is the learning rate given in equation (6). On-line learning was used exclusively, with no storage of gradients across cases, and all weights were initialized to random values between -0.5 and 0.5.

The 1000 training cases used in the 16-bit shifter were randomly selected from the much larger set of all possible shifted vectors. Since the details of the connectivity were determined randomly, there were occasions when particular networks required slightly different learning schedules, but the values given in the table were generally successful.

A four step annealing schedule was usually employed, where $T = 15, 5, 1$ and 0.5. An initial $T = 25$ step was

Table 1: Learning Schedules for the Shifter Networks

Network type		learning schedule	number of cases
4-bit			24
	symmetric	$\epsilon = 0.01$ to completion	
	asymmetric	$\epsilon = 0.005$ to completion	
8-bit			504
	symmetric	$\epsilon = 0.005$ for 40 epochs, 0.001 to completion	
	asymmetric	$\epsilon = 0.005$ for 40 epochs, 0.001 to completion	
16-bit			1000
	symmetric	$\epsilon = 0.002$ for 50 epochs, 0.001 for 5 epochs 0.0005 to completion	
	asymmetric	$\epsilon = 0.001$ for 60 epochs, 0.0005 to completion	

found helpful in the 16-bit symmetric network. At each temperature, the network was settled to a stable state using a synchronous, discrete time approximation of the set of differential equations

$$\frac{dy_i}{dt} = -y_i + \tanh(\frac{1}{T}\sum_j y_j w_{ij}) \qquad (15)$$

This was accomplished by damping each unit so that, after each synchronous update, the unit activities would be given by

$$\begin{aligned} y_i(t) &= (1-\alpha)\tanh\left(\frac{1}{T}\sum_j y_j(t-1)w_{ij}\right) \\ &\quad + \alpha y_i(t-1) \qquad (16) \end{aligned}$$

for α between 0 and 1. 12 synchronous updates were employed at each temperature in all the 4- and 8-bit shifters, with $\alpha = 0.5$. The 16-bit shifters required 14 synchronous updates with an α of 0.6. The 4-bit symmetric shifters generally required fewer updates and less damping than the comparable asymmetric nets, but this difference largely disappeared in the larger networks.

Contrastive Hebbian Learning in the Continuous Hopfield Model

Javier R. Movellan
Department of Psychology
Carnegie Mellon University
Pittsburgh, Pa 15213
email: jm2z+@andrew.cmu.edu

Abstract

This paper shows that contrastive Hebbian, the algorithm used in mean field learning, can be applied to any continuous Hopfield model. This implies that non-logistic activation functions as well as self connections are allowed. Contrary to previous approaches, the learning algorithm is derived without considering it a mean field approximation to Boltzmann machine learning. The paper includes a discussion of the conditions under which the function that contrastive Hebbian minimizes can be considered a proper error function, and an analysis of five different training regimes. An appendix provides complete demonstrations and specific instructions on how to implement contrastive Hebbian learning in interactive activation and competition models (a convenient version of the continuous Hopfield model).

1 INTRODUCTION

In this paper we refer to interactive activation networks as the class of neural network models which have differentiable, bounded, strictly increasing activation functions, symmetric recurrent connections, and for which we are interested in the equilibrium activation states rather than the trajectories to achieve them. This type of network is also known as the continuous Hopfield model [6]. Some of the benefits of interactive activation networks as opposed to feed-forward networks are their completion properties, flexibility in the treatment of units as inputs or outputs, appropriateness for solving soft-constraint satisfaction problems, suitability for modeling cognitive processes [9], and the fact that they have an associated energy function that may be applied in pattern recognition problems [13]. Contrastive Hebbian Learning[7] (CHL), which is a generalization of the Hebbian rule, updates the weights proportionally to the difference in the crossproducts of activations in a clamped and a free running phase. This modification of the Hebbian learning, first applied by Hopfield to improve the storage capacity of discrete content addressable memories without hidden units [5], appears in the Boltzmann learning algorithm [1] and its mean field approximation [11][3]. This paper shows that Hinton's observation that CHL depends on a performance measure [3] can be generalized to any case of the continuous Hopfield model. Contrary to previous approaches, the derivations do not presume the existence of Boltzmann machines approximated with mean field networks. The paper includes a discussion of the conditions under which the function that CHL minimizes can be considered a proper error function, an analysis of undesirable effects that may occur in CHL learning, and a classification of training regimes that minimize these effects.

The paper is divided in two sections and one appendix. Section 1 describes the dynamics of the activations in interactive networks. Section 2 shows how to modify the weights for the stable states of the network to reproduce desired patterns of activations. The original contribution in this paper are:

- To show that CHL works with any continuous Hopfield network and thus that self-connections as well as non-logistic activation functions are allowed.
- To show that the principles involved in the mean field learning algorithm can be derived independently of the Boltzmann Machine.
- To show that except for the case where there are no hidden units, the function that CHL minimizes is not a proper error function but that in practice there are training regimes that make it work as such.

For completeness we present some of the classical proofs provided in Hopfield [6]. The appendix contains mathematical details and specific comments on how to implement Contrastive Hebbian Learning in interactive networks.

2 STABILITY OF ACTIVATIONS

Since interactive networks have recurrent paths, it is in principle possible that their activations never stabilize. Fortunately, if some simple conditions investigated by Hopfield [6] are met, we can guarantee that the activations will settle, and that at equilibrium they will be at a minimum of an *Energy* function.

Let the activation vector $\mathbf{a}^T = [a_1, ...a_n]$, be represented by bounded, monotonically increasing, differentiable activation functions $f_i(.)$. Let $\mathbf{W} = [\mathbf{w}_1, ...\mathbf{w}_n]$ be the matrix of connections, where the $\mathbf{w}_i^T = [w_{1,i}, ...w_{n,i}]$, are bounded, fan-in weight row vectors. Let $d()/dt$ be derivatives with respect to time, $net_i = \mathbf{a}_i^t\ \mathbf{w}_i$, and $rest = f(0)$. Define a continuous Hopfield Energy function [6]

$$F = E + S \qquad (1)$$

where

$$E = -\frac{1}{2}\,\mathbf{a}^t \mathbf{W}\,\mathbf{a} = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i w_{ij} a_j \qquad (2)$$

and

$$S = \sum_{i=1}^{n} \int_{rest}^{a_i} f_i^{-1}(a)\, da \qquad (3)$$

It can be shown that E, which reflects the constraints imposed by the weights in the network, tends to drive activations to extreme values [1]. On the other hand, S is a penalty function that tends to drive the activations to a central value (the resting point). In principle we are interested in activation states that minimize E for they are maximally harmonious with the information encoded in the weights. As we will study later, varying the relative importance of E vs. S as the activations settle may help achieve maximally harmonious (minimum E) states. In the appendix it is shown that if the activation functions are the standard (0-1) logistic, F becomes the Helmholtz free energy function as defined in [3].

Hopfield [6] showed that if the network is governed by the set of differential equations

$$\frac{d\, f_i^{-1}(a_i)}{dt} = \lambda\,(-f_i^{-1}(a_i) + net_i)\,;\ \ i = 1...n \qquad (4)$$

and the weights are symmetric, the activations stabilize in a minimum of F. For completeness, I present in the appendix a version of Hopfield's proof and show that stability in a global minimum can also be achieved with the following equation, typically used in interactive activation networks [8][9] [10]

$$\frac{d\, a_i}{dt} = \lambda\ ((-a_i + f_i(net_i)) \qquad (5)$$

Notice that if we apply either equation 4 or 5, on equilibrium (when the derivatives are zero),

$$f_k^{-1}(\check{a}_i) = n\check{e}t_i \qquad (6)$$

where (ˇ) represents equilibrium. These properties will be used to derive the learning algorithm in the next section.

3 CONTRASTIVE LEARNING

Learning is viewed as the modification of connections between units so that the stable states of the network reproduce desired patterns of activations. We will see that CHL minimizes a *contrastive function J* [2] and then we will discuss the conditions under which minimization of J guarantees learning.

Define a pattern p as the pair $p = \{\mathbf{a}^{I(+)}, \mathbf{a}^{O(+)}\}$, where I stands for input set, O for output set, and (+) indicates that the activation of these units are fixed by the pattern. In connectionist terms $\mathbf{a}^{O(+)}$ is the teacher vector. We will say that p has been learned if clamping the input units to $\mathbf{a}^{I(+)}$ then

$$\check{\mathbf{a}}^{O(-)} = \mathbf{a}^{O(+)} \qquad (7)$$

where $\check{\mathbf{a}}^{O(-)}$ denotes the activation of the output set when inputs are clamped and outputs are free. Note that in this form of supervised learning we are only interested in the outputs obtained after equilibrium is achieved and not in the trajectories followed to equilibrium.

Define the *contrastive function J* as

$$J = \check{F}^{(+)} - \check{F}^{(-)} \qquad (8)$$

where $\check{F}^{(+)}$ and $\check{F}^{(-)}$ respectively are the values of the energy functions at equilibrium when the inputs and outputs are clamped (+), and when the inputs are clamped and outputs are free (-). Notice that $\check{F}^{(-)}$ has the same free parameters that $F^{(+)}$, the activations of the hidden units, plus some additional free parameters, the activations of the output units. Assume that, over the working region of activation states, F has a unique minimum[3]. Thus, since the minimum is unique $\check{F}^{(+)} \geq \check{F}^{(-)}$, and if $J = 0$, then

[1] If self connections are allowed, minima in E may also occur for intermediate activation values.

[2] Based on different arguments, Hinton [3] showed that CHL minimizes the equivalent of J when the activation is logistic.

[3] This assumption, which is shared with Boltzmann learning, Mean Field Learning, and the Almeida-Pineda algorithm is discussed in section 4.

$\breve{a}^{O(-)} = a^{O(+)}$. This makes J a potential candidate for learning by gradient descent on weight space. In particular, following the derivations detailed in the appendix, we have

$$\frac{\partial \breve{E}}{\partial w_{ij}} = -\breve{a}_i\breve{a}_j - \sum_{k=1}^{n} n\breve{e}t_k(\frac{\partial \breve{a}_k}{\partial w_{ij}}) \; ; \; i \neq j \quad (9)$$

$$\frac{\partial \breve{E}}{\partial w_{ij}} = -\frac{1}{2}\breve{a}_i^2 - \sum_{k=1}^{n} n\breve{e}t_k(\frac{\partial \breve{a}_k}{\partial w_{ij}}) \; ; \; i = j \quad (10)$$

$$\frac{\partial \breve{S}}{\partial w_{ij}} = \sum_{k=1}^{n} f_k^{-1}(\breve{a}_k)\frac{\partial \breve{a}_k}{\partial w_{ij}} \quad (11)$$

And following equation 6 it is easy to see that

$$\frac{\partial \breve{F}}{\partial w_{ij}} = -\breve{a}_i\breve{a}_j \; ; \; i \neq j \quad (12)$$

$$\frac{\partial \breve{F}}{\partial w_{ij}} = -\frac{1}{2}\breve{a}_i\breve{a}_j \; ; \; i = j \quad (13)$$

$$(14)$$

making

$$\frac{\partial \breve{J}}{\partial w_{ij}} \propto \breve{a}_i^{(-)}\breve{a}_j^{(-)} - \breve{a}_i^{(+)}\breve{a}_j^{(+)} \quad (15)$$

which shows that the contrastive Hebbian learning rule

$$\Delta w_{ij} \propto \breve{a}_i^{(+)}\breve{a}_j^{(+)} - \breve{a}_i^{(-)}\breve{a}_j^{(-)} \quad (16)$$

descends in the J function.

4 DISCUSSION

We have seen that CHL minimizes the contrastive function

$$J = \breve{F}^{(+)} - \breve{F}^{(-)} \quad (17)$$

so that after each learning step, the difference in energy at equilibrium between the clamped and free states becomes smaller. At this point we will study the conditions under which J can be considered a proper error function.

An important property of error functions is that they decrease as the difference between the obtained and the desired states decreases. CHL guarantees that the difference between energies in the clamped and free phases will become smaller but as we will see this does not always guarantee that the difference between the clamped and free state activations will decrease. If both $F^{(+)}$ and $F^{(-)}$ have a unique minimum (e.g. when there are no hidden units) and since in the (-) phase there are more free parameters to minimize F than in the (+) phase it follows that J cannot be smaller than zero and that when $J = 0$ the activations in the free and clamped phase are equal. In this case, J is a proper error function. In the appendix it is shown that if there are no hidden units CHL is in fact equivalent to backpropagation learning.

However, if there are multiple minima in F, we can no longer guarantee that a local minima in the free phase will have lower energy than a local minima in the clamped phase. One way to avoid this problem is to use training regimes that maximize the probability that the activations in both the free-running and the clamped phase equilibrate in the same region of attraction of $F^{(-)}$ space. A way to visualize how CHL works is to imagine the energy surface over activation space as a membrane with several minima. CHL pushes down the minimum corresponding to the clamped case and pulls up the stable states for the free phases. If both stable states are in the same attractor, after several learning trials, both minima converge. What follows is a brief analysis of five different training regimes:

- Case 1: *Activations are reset to random numbers after each learning phase.* In this case, the starting points for the clamped and free phase are different and, very likely the stable states will also be apart. Under this condition, CHL learning does not work well.

- Case 2: *First settle for the clamped phase, and then, without resetting activations, free the output units and settle again.* This procedure guarantees that $\breve{F}^{(+)} \geq \breve{F}^{(-)}$ and that when $\breve{F}^{(+)} = \breve{F}^{(-)}$ the activations on the free and clamped phases are the same. Gradient descent on J assures that when the minima for the clamped phase is achieved, if the output unit activations are free, they will not change. This form of learning, which can be used for recognition of familiar patterns, is very rapidly achieved with CHL (it just takes about 3 trials to "recognize" the XOR or the 4-3-4 encoder patterns). Unfortunately, if we just clamp the input units without information about the teachers, in general the activations will not converge to the desired minima.

- Case 3: *Settle during the free phase and then, without resetting activations, clamp the output units and resettle again.*

  This scheme minimizes the probability that the clamped and free phases end up in different regions of attraction. In general this procedure works well and achieves learning speeds comparable to backpropagation. There are two phenomena though that sometimes occur [3]. Occasionally the network may settle in a different attractor than the one to which it had converged in previous trials. This may result in a sudden change in activations and and a temporary "unlearning" of the weights. In figure 1 it can be seen a typical

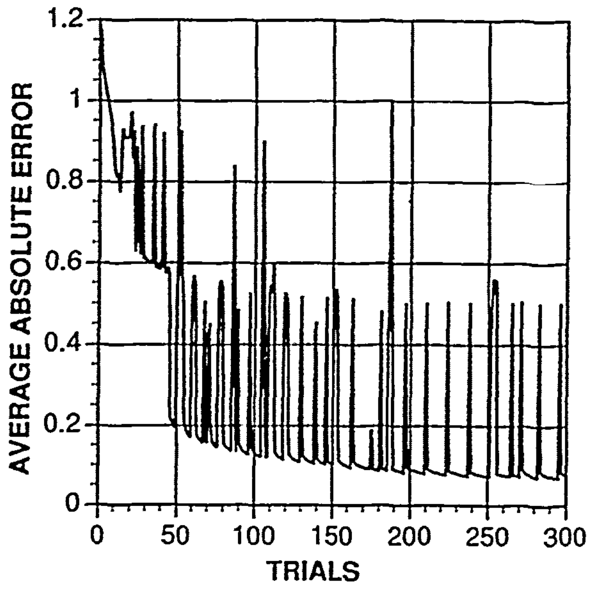


Figure 1. *Typical learning curve for the XOR problem.*

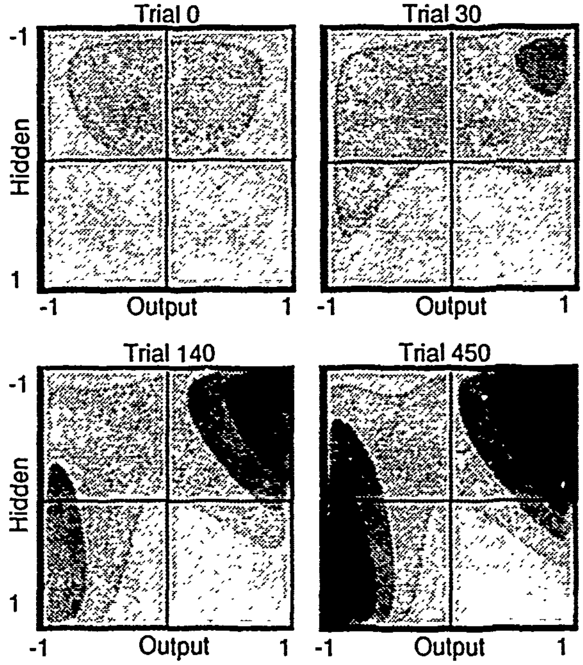


Figure 2. *Contour curves of the free energy function as learning progresses. Note that although initially there is a unique minimum, eventually an spurious local minimum is generated.*

learning curve with these temporary unlearning of the patterns. It is our experience that as learning progresses these sudden jumps to other minima tend to diminish. Another problem is that if the the clamped and free phases stabilize in different regions, the energy of the clamped phase may become lower than the energy in the free phase. If this happens, learning usually deteriorates, making it advisable to start with a different set of weights. Figure 2 shows the energy functions generated after training a simple network, with an input unit, a hidden unit and an output unit, to learn the 1-1-1 encoder problem. It can be seen that as learning progresses a minimum is created at the desired state (output unit activation =1) but that a spurious local minimum is also created. If in some trial the activations equilibrate in that local minimum abrupt distortions in the learning curve and temporary unlearning of the desired activations may occur [4].

- Case 4: *Sharpening Schedules* [5].

As we previously mentioned if there is a unique global minimum of $\check{F}^{(-)}$ and of $\check{F}^{(+)}$, and the activations settle into this minimum then the contrastive function J can be considered a proper error measure. One way of decreasing the likelihood of settling into spurious local minima is the use of *sharpening schedules*. Sharpening schedules modify the *gain* or *sharpness* of the activation functions as the settling of the activations proceeds. Usually we start with low gain (flat activation functions) and progressively increase it. The rational for this procedure is as follows: Decreasing the sharpness of the activation functions is equivalent to weighting more heavily the S part of the free energy as defined in equation 1 [6]. Figure 3 shows the effect of sharpening on the error function learned for the 1-1-1 encoder problem mentioned in Case 3. For large decay values (low gain) the spurious local minima dissapear. Ideally, we start settling the activations with appropriately large decay values that get rid of the spurious local minima but allow the global minimum to survive. Then we let the activations settle towards this global minimum and slowly guide them away from rest values by progressively decreasing decay (increasing gain). Peterson and Anderson [11] and Peterson and Hartman[12] call this procedure *annealing* for it is a mean field approximation to annealing schedules of discrete Boltzmann machines. I have decided to use the term *sharpening schedules* as defined in [2] to clarify the fact that sharpening does not have anything to do with increasing the randomness of the network as one would expect of annealing schedules

[4] Since the networks we are considered so far are deterministic, the settling state is determined by the final state. However, slight weight modifications made by the learning algorithm may be sufficient to make the activations settle in completely different attractors

[5] I thank Conrad Galland for showing me the role that sharpening schedules play in contrastive Hebbian learning.

[6] In the interactive activation and competition model [10] this is controlled by a the decay parameter. If logistic activations are used, sharpening is achieved by controlling the gain of the activation functions.

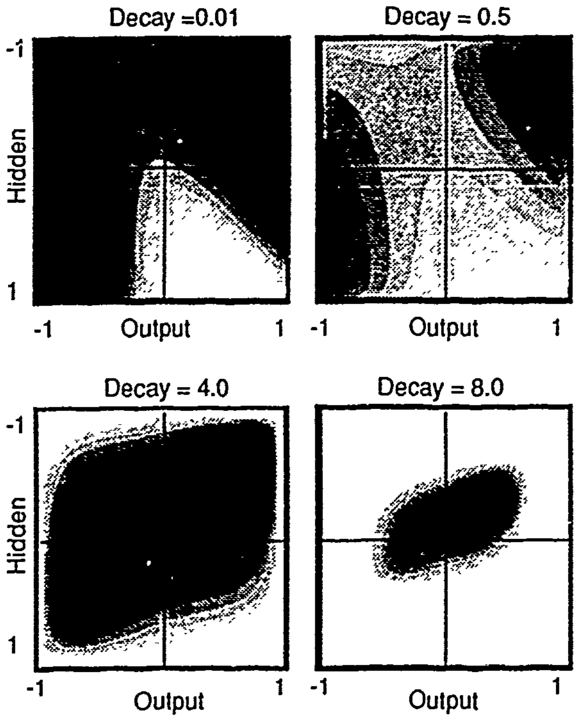


Figure 3: *The energy function for different degrees of sharpening. In this case sharpening is controlled by a decay parameter. Note how for low sharpening levels spurious minima disappear*

[7]. Although sharpening schedules are not strictly necessary for CHL to work, there are reasons to believe that they improve learning performance.

- Case 5: *Annealing schedules: In search of the continuous Boltzmann machine.* Annealing schedules are another method for avoiding spurious minima. Annealing schedules progressively decrease the randomness of the activations as settling progresses. This may be achieved in interactive networks by injecting some form of noise (e.g. logistic noise) to the net input of each unit. The standard deviation of the noise distribution plays a similar effect to the temperature parameter in discrete Boltzmann machines. Case 1 in this discussion can be viewed as a particular case of annealing schedule in which the standard deviation of noise is very large for the initial cycle (producing a random starting point) and then goes to zero on the next cycle (making the network deterministic). Using slowly decreasing annealing schedules may improve the likelihood of settling in the best minimum and avoiding spurious ones. It can be shown that this settling method defines a continuous state hidden Markov model (a diffussion process). The activation settling algorithm can be seen as an approximation to gradient descent of the expected value of the Hopfield energy function F. The appropriate learning algorith should use the difference in the equilibrium expected values of the activation crossproducts in the clamped and free running phase.

A nice property shown in the appendix is that if logistic noise is added to the net input, the limiting behavior of interactive networks as the sharpness of the activation functions increases is given by the discrete Boltzmann machine. This property may be used to implement both Boltzmann machines and continuous interactive networks with the same program.

In spite of its problems and the fact that more research is needed before using it for large scale problems, CHL is a promising, simple to implement learning method that works for a wide variety of interactive architectures. In Table 1 appear the results of simulations using CHL with the activation update rule of the Interactive Activation and Competition model. [10].

5 APPENDIX

5.1 S AND THE MEAN FIELD ENTROPY TERM

The entropy term of the energy function proposed by Hinton [3] for the Mean Field Algorithm is

$$-\sum_{i=1}^{n}(a_i\, log a_i + (1-a_1)\, log(1-a_i)) . \qquad (18)$$

If we use the standard logistic (0-1) activation function,

$$y = \frac{1}{1+e^{-x/T}} \qquad (19)$$

whose inverse is

$$x = T\, log\left(\frac{y}{1-y}\right) \qquad (20)$$

then

$$\sum_{i=1}^{n}\int_{0.5}^{y} T\, log\left(\frac{y}{1-y}\right)\, dy \quad (21)$$

$$= T\sum_{i=1}^{n}((y\, log\, y + (1-y)\, log(1-y)) - log 0.5) \quad (22)$$

5.2 S AND THE INTERACTION ACTIVATION AND COMPETITION MODEL

The activation update rule of the interactive activation and competition model (IAC) as defined in [9] is

$$\Delta a_i = \lambda((max - a_i)\, net_i - (a_i - rest)\, decay) \;,\; net \geq 0 \qquad (23)$$

[7]The term "sharpening" is not without problems either for it conceals the fact that sharpening is a mean field approximation to true annealing. Another possible term is "mean field annealing" but this may be confused with the true annealing method as discussed in case 5.

$$\Delta a_i = \lambda\left((a_i - min)\,net_i - (a_i - rest)\,decay\right)\ ;\ net \leq 0 \tag{24}$$

which can be derived from equation 5 applied to the following activation function

$$a_i = \frac{max\,net_i + rest\,decay}{net_i + decay}\ ;\ net \geq 0 \tag{25}$$

$$a_i = \frac{min\,net_i - rest\,decay}{net_i - decay}\ ;\ net \leq 0 \tag{26}$$

where max is the maximum value of the activation, rest the activation when the net input is zero, min the minimum value of the activation, and decay a positive constant. And applying equation 3, it is easy to see that

$$S = \sum_{i=1}^{n} S_i \tag{27}$$

with

$$S_i = decay((max - rest)\ log\left(\frac{max - rest}{max - a_i}\right) \tag{28}$$

$$-\,(a_i - rest));\ \ a_i \geq rest \tag{29}$$

$$S_i = decay((min - rest)\ log\left(\frac{a_i - min}{rest - min}\right) \tag{30}$$

$$+\,(a_i - rest));\ \ a_i \leq rest \tag{31}$$

with the decay parameter assuming the same function than gain in the logistic model.

5.3 HOPFIELD'S PROOF OF THE STABILITY OF ACTIVATIONS

Hopfield showed that if the network is regulated by equation 4 it will stabilize. This is done by showing that the Hopfield model has an associated Lyapunov function. Here, a similar argument is used to show that using equation 5 the network also stabilizes. To facilitate the calculation of the derivatives of E, we collapse into Q the part of F that does not depend on a_i,

$$E = -\frac{1}{2}\sum_{k=1}^{n}\sum_{l=1}^{n} a_k w_{kl} a_l \tag{32}$$

$$= -\frac{1}{2}(a_i w_{ii} a_i + a_i(\sum_{\substack{k=1\\k\neq i}}^{n} a_k w_{ik} \tag{33}$$

$$+\sum_{\substack{l=1\\l\neq i}}^{n} a_l w_{li}) + Q) \tag{34}$$

and considering that the weights are symmetric,

$$\frac{\partial E}{\partial a_i} = -\frac{1}{2}\left(2a_i w_{ii} + 2\sum_{\substack{k=1\\k\neq i}}^{n} a_k w_{ik}\right) = -net_i \tag{35}$$

Regarding S, since the derivative of the integral of a function is the function itself

$$\frac{\partial S}{\partial a_i} = f_i^{-1}(a_i) \tag{36}$$

and

$$\frac{\partial F}{\partial a_i} = \frac{\partial E}{\partial a_i} + \frac{\partial S}{\partial a_i} = -net_i + f_i^{-1}(a_i) \tag{37}$$

If we make

$$\frac{da_i}{dt} = \lambda\left(-a_i + f(net_k)\right) \tag{38}$$

then, applying the chain rule,

$$\frac{dF}{dt} = \sum_{k=1}^{n} \frac{\partial F}{\partial a_k}\frac{da_k}{dt} \tag{39}$$

$$= \sum_{k=1}^{n} \lambda\left(-net + f_k^{-1}(a_k)\right)\left(-a_k + f(net_k)\right) \tag{40}$$

but since f_k is monotonic, $\left(-net + f_k^{-1}(a_k)\right)$ has the same sign as $(-f(net_k) + a_k)$, making

$$\frac{dF}{dt} \leq 0 \tag{41}$$

Since F is bounded and on each time step F decreases then

$$\lim_{t\to\infty} \frac{dF}{dt} = 0 \tag{42}$$

and since equation 39 shows that

$$\frac{dF}{dt} = 0 \Longleftrightarrow \frac{da_k}{dt} = 0 \Longleftrightarrow \frac{\partial F}{\partial a_k} = 0\ \ ;k = 1, ..., n \tag{43}$$

then

$$\lim_{t\to\infty} \frac{da_i}{dt} = 0\ \ ;k = 1, ..., n \tag{44}$$

which tells us that the activations will tend to equilibrium as time progresses and that on equilibrium they are in a minima of F.

5.4 SMOOTHING NET INPUTS VS. ACTIVATIONS

Equation 4 can be discretized as

$$\Delta f_i^{-1}(a_{i(t)}) = \lambda\left(-f_i^{-1}(a_{i(t)}) + net_{i(t)}\right) \tag{45}$$

or

$$f_i^{-1}(a_{i(t+1)}) = (1 - \lambda) f_i^{-1}(a_{i(t)}) + \lambda net_{i(t)} \tag{46}$$

equivalently, equation 5 would be discretized as

$$a_{i(t+1)} = (1 - \lambda) a_{i(t)} + \lambda f(net_{i(t)}) \tag{47}$$

Equation 46 is an exponential smoothing of the net input. The activation function is then applied to this smoothed net. Another possibility, represented in equation 47, is to apply the activation function to the instantaneous net input and then to exponentially smooth these activations.

5.5 THE DERIVATIVES OF THE EQUILIBRIUM POINTS OF F

First consider the weights that connect different units. Extracting the cross products with a w_{ij} term. We have

$$\breve{E} = -\frac{1}{2}\left(2\breve{a}_i w_{ij}\breve{a}_j + \sum_{k=1}^{n}\sum_{\substack{l=1\\ k,l\neq i,j;k,l\neq j,i}}^{n} \breve{a}_k w_{kl}\breve{a}_l\right) \quad (48)$$

and considering that w_{ij} is the only weight depending on w_{ij}.

$$\frac{\partial \breve{E}}{\partial w_{ij}} = -\frac{1}{2}(2\breve{a}_i\breve{a}_j + 2w_{ij}\breve{a}_i\frac{\partial \breve{a}_j}{\partial w_{ij}} + 2w_{ij}\breve{a}_j\frac{\partial \breve{a}_i}{\partial w_{ij}} \quad (49)$$

$$+\sum_{k=1}^{n}\sum_{\substack{l=1\\ k,l\neq i,j;k,l\neq j,i}}^{n} w_{kl}(\breve{a}_k\frac{\partial \breve{a}_l}{\partial w_{ij}} + \breve{a}_l\frac{\partial \breve{a}_k}{\partial w_{ij}})) \quad (50)$$

Reorganizing terms considering that the weights are symmetric,

$$\frac{\partial \breve{E}}{\partial w_{ij}} = -\frac{1}{2}\left(2\breve{a}_i\breve{a}_j + 2\sum_{k=1}^{n}\frac{\partial \breve{a}_k}{\partial w_{ij}}\sum_{l=1}^{n} w_{kl}\breve{a}_l\right) \quad (51)$$

which easily leads to equation 9. Similar arguments can be applied to derive equation 10. Eq:.ation 11 is easy to obtain by applying the chain rule and the fact that the derivative is the inverse of the integral.

5.6 CONTRASTIVE HEBBIAN AND BACKPROPAGATION LEARNING ARE EQUIVALENT WHEN THERE ARE NO HIDDEN UNITS

The backpropagation weight update equation is

$$\Delta w_{ij} \propto I_i f'(a_j)(t_j - a_j) \quad (52)$$

where w_{ij} is the weight connecting input unit i with output unit j, f' the derivative of the activation function, and t_j the teacher for output unit j. The contrastive Hebbian weight update is

$$\Delta w_{ij} \propto \breve{a}_i^{(+)}\breve{a}_j^{(+)} - \breve{a}_i^{(-)}\breve{a}_j^{(-)} \quad (53)$$

Since the input units are clamped in both phases, they are not influenced by the output units and the equilibrium point of the activations would be the same as in ɔackpropagation. Taking this into consideration and reorganizing terms we have

$$\Delta w_{ij} \propto \breve{a}_i^{(+)}(\breve{a}_j^{(+)} - \breve{a}_i^{(-)}) \quad (54`$$

where $\breve{a}_j^{(+)}$ is the same as the teacher, and $\breve{a}_i^{(+)}$ the clamped input. Since the derivative of the activation function is always positive (for strictly increasing activation functions), the cosine of the angle between the update vectors in backpropagation and contrastive Hebbian is positive and therefore they both minimize the same error function. Since there are no hidden units, the error function has a unique minima and thus the final learned solutions will be equivalent in both backpropagation and contrastive Hebbian.

5.7 INTERACTIVE NETWORKS WITH LOGISTIC NOISE APPROXIMATE BOLTZMANN MACHINES AS THE SHARPNESS OF THE ACTIVATION FUNCTIONS INCREASES

As sharpening increases, the activation function converges to a threshold function

$$a_i = max; net_i > 0 \quad (55)$$

$$a_i = min; net_i \leq 0 \quad (56)$$

where *max* is the upper bound of the activation, *min* the lower bound, and *net*$_i$ has an added logistic noise component (σ) with variance $\frac{T^2\pi^2}{2}$.

$$net_i = \mathrm{a}_i^t\,\mathrm{w}_i + \sigma \quad (57)$$

It follows that

$$Prob(a_i = max) = Prob(net_i > 0) \quad (58)$$

$$= 1 - Prob(\sigma \leq -\mathrm{a}_i^T\,\mathrm{w}_i) \quad (59)$$

$$= 1 - \frac{1}{1 + e^{(\mathrm{a}_i^T\,\mathrm{w}_i)/T}} \quad (60)$$

$$= \frac{1}{1 + e^{-(\mathrm{a}_i^T\,\mathrm{w}_i)/T}} \quad (61)$$

which defines a Boltzmann machine.

5.8 SKETCH OF THE MAIN ROUTINES OF A CONTRASTIVE HEBBIAN PROGRAM

1. Get a training pattern.
2. Reset activations to rest and net inputs to zero.
3. Clamp inputs to desired pattern.
4. Settle activations according to equations 23 and 24. The program may provide some facility for sharpening schedules (changing the decay or gain parameter through settling), and annealing schedules (changing the standard deviation of noise added to the net inputs).
5. Collect cross products of activations multiplied by a negative constant.
6. Clamp also the output units to the desired pattern.
7. Settle activations according to equations 23 and 24.
8. Collect cross products of activations multiplied by a positive constant.

Termination of settling may be done after a fixed number of iterations (let us say 30), or after the changes in activations are smaller than a certain criteria (e.g biggest activation change is smaller than 0 01) Bellow is an example of a settling cycle using the update function of the IAC model [9].

```
for(i=1; i≤ number_of_ units; i++){
   for(j=1; j≤ number_of_units; j++){
      net[i] = net[i] + w[j][i]*activation[j] ;
   }
   if(net[i] ≥ 0) act[i] = acti[i] + lambda*((actimax
-act[i]) *net[i] - decay*(acti[i]-rest));
else acti[i] =acti[i] + lambda*((acti[i] - actimin)*net[i]
- decay*(acti[i]-rest));
if(acti[i] > max) acti[i]=max;
if(acti[i] < min) acti[i]=min;
}
```

Where max is the maximum value of the activations, min the minimum value, rest the activation when net is zero, lambda the stepsize of the activation change (smoothing constant), and decay a positive constant. We have obtained good results with actimax =1.0, actimin = -1.0, rest =0.0, decay =0.1, lambda = 0.2.

This is an example of a weight change routine

```
for(i=1; ≤ number_of_units; i++)

   for(j=1; ≤ number_of_units;  j++){

      weight_change[i][j] = weight_change[i][j] + phase
*epsilon* a[i]*a[j] ;
   }
}
```

Where *phase* is (+1) for the clamped case and (-1) for the free case. Epsilon is the stepsize of the weight change. Weights may be updated after each phase, each pattern or after a batch of patterns. Table 1 may be used to standardize new implementations of the algorithm.

	stepsize			
hidden u.	1.0	0.1	0.01	0.001
1	19 (0)	30 (0)	246 (5)	≥300 (10)
2	13 (0)	15 (0)	72 (1)	≥300 (7)
4	55 (3)	20.5 (0)	40.5 (0)	≥300 (6)
8	47 (1)	23 (0)	21 (0)	245 (4)
16	≥300 (10)	98.5 (0)	24.5 (0)	142(1)

Table 1: Results for the XOR problem. Networks were fully connected (including self connections). There were 10 simulations per cell with different starting random weights from a uniform (-0.5 to 0.5) distribution. Outside parenthesis are median number of epochs until maximum absolute error was smaller than 0.2. In parenthesis are number of simulations in which learning took longer than 300 epochs. Learning was on batch mode. Update of activation was done according to the IAC equations with the following parameter values: max=1.0, min=-1.0, rest=0.0, decay=0.1, lambda= 0.2. Update of activations was stopped after the change in all the activations was smaller than 0.0001 or the number of cycles bigger than 100. No annealing or sharpening was used

Acknowledgements

This research was founded by the NSF grants: BNS 88-12048 and BNS 86-09729. This paper would not have been possible without the support and motivating ideas of James McClelland and his research group at Carnegie Mellon University. I also thank Geoffrey Hinton, Conrad Galland and Chris Williams for their advice and helpful comments.

References

[1] Ackley D, Hinton G and Sejnowski T (1985) A learning algorithm for Boltzmann machines. *Cognitive Science*, 9, 147-169.

[2] Akiyama Y, Yamashita A, Kajiura M, Aiso H (1989) Combinatorial Optimization with Gaussian Machines, *Proceedings of the International Joint Conference on Neural Networks* 1, 533-540.

[3] Hinton G E (1989) Deterministic Boltzmann Learning Performs Steepest Descent in Weight-Space, *Neural Computation*, 1, 143-150.

[4] Hopfield J (1982) Neural Networks and Physical Systems with emergent collective computational abilities. *Proceedings of the National Academy of Science* USA, 79, 2254-2558.

[5] Hopfield J, Feinstein D, Palmer R (1983) Unlearning has a stabilizing effect in collective memories. *Nature* 304, 158-159.

[6] Hopfield J (1984) Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences* U.S.A., 81, 3088-3092.

[7] Galland C, Hinton G (1989) Deterministic Boltzmann Learning in Networks with Asymmetric Connectivity. University of Toronto. Department of Computer Science Technical Report. CRG-TR-89-6.

[8] Grossberg S A (1978) A theory of visual coding, memory, and development. in E Leeuwenberg and H Buffart (Eds.) *Formal Theories of visual perception* New York, Willey.

[9] McClelland J, Rumelhart D (1981) An Interactive Activation Model of Context Effects in Letter Perception: Part 1. An account of Basic Findings. *Psychological Review*, 88, 5.

[10] McClelland J Rumelhart D (1989) Interactive activation and Competition. Chapter II of *Explorations in Parallel Distributed Processing: A Handbook of Models, Programs, and Exercises* Cambridge, MIT Press.

[11] Peterson C, Anderson J R (1987) A mean field theory learning algorithm for neural networks. *Complex Systems*, 1, 995-1019.

[12] Peterson C, Hartman E (1989) Explorations of the Mean Field Theory Learning Algorithm. *Neural Networks*, 2, 475-494.

[13] Williams C, Hinton G (1990) Mean field networks that learn to discriminate temporally distorted strings. *Pre-Proceedings of the 1990 Connectionist Models Summer School*.

Mean field networks that learn to discriminate temporally distorted strings

Christopher K. I. Williams
Department of Computer Science
University of Toronto
10 King's College Road
Toronto M5S 1A4, Canada

Geoffrey E. Hinton
Department of Computer Science
University of Toronto
10 King's College Road
Toronto M5S 1A4, Canada

Abstract

Neural networks can be used to discriminate between very similar phonemes and they can handle the variability in time of occurrence by using a time-delay architecture followed by a temporal integration (Lang, Hinton and Waibel, 1990). So far, however, neural networks have been less successful at handling longer duration events that require something equivalent to "time warping" in order to match stored knowledge to the data. We present a type of mean field network (MFN) with tied weights that is capable of approximating the recognizer for a hidden markov model (HMM). In the process of settling to a stable state, the MFN finds a blend of likely ways of generating the input string given its internal model of the probabilities of transitions between hidden states and the probabilities of input symbols given a hidden state. This blend is a heuristic approximation to the full set of path probabilities that is implicitly represented by an HMM recognizer. The learning algorithm for the MFN is less efficient than for an HMM of the same size. However, the MFN is capable of using distributed representations of the hidden state, and this can make it exponentially more efficient than an HMM when modelling strings produced by a generator that itself has componential states. We view this type of MFN as a way of allowing more powerful representations without abandoning the automatic parameter estimation procedures that have allowed relatively simple models like HMM's to outperform complex AI representations on real tasks.

INTRODUCTION

Neural networks have been successful at phoneme discrimination tasks, but many researchers currently feel that the best way to deal with longer duration events is to use a neural network as a front-end to a hidden markov model. The difficult task of matching models to data across temporal distortions is then handled by the hidden markov recognizer which efficiently considers all possible matches.

Despite their powerful matching and learning procedures, HMM's have a serious drawback: They implicitly assume that the ensemble of input strings is generated by a stochastic finite-state automaton and this strongly limits the types of structure that they can efficiently represent. Suppose, for example, that 20 independent binary constraints operate between the first and second half of a string and that as a result of all these separate constraints the mutual information between the two halves is 20 bits.[1] To model these 20 constraints, a hidden markov generator would need at least 2^{20} hidden states because the only way that the first half of a string can constrain the second half is via the hidden state of the generator as it finishes generating the first half.

In a hidden markov generator, 2^{20} hidden states implies 2^{20} hidden nodes. In a neural net that uses distributed representations, 2^{20} hidden state vectors only requires 20 binary hidden units. So if the mutual information between the first and second half of each string is genuinely componential, a neural network can be exponentially more efficient in representing the constraints. In effect, the only way that a HMM can deal with a set of independent constraints is to use the cross-product of the HMM's that would be needed to capture each constraint separately, so it is unable to take advantage of the fact that the constraint structure

[1] As a concrete example, we might suppose that the first half of a sentence is singular or plural, active or passive, past or present tense, abstract or concrete, etc. etc. and that the second half "agrees" with the first half along all these dimensions.

can be factorized.

Our aim is to develop a neural network alternative to HMM's that can take advantage of constraints which can be factorized by using separate hidden units to enforce separate constraints. To achieve this we have been forced to use a matching procedure that is only a heuristic approximation to the recognition procedure used in an HMM, and a learning procedure that is considerable slower than the HMM learning procedure for a comparably sized network. Eventually, we hope to show that these disadvantages are more than offset by the ability of the neural network to use a *small* representation that captures the componential structure of the constraints efficiently. In this paper, however, we only show that the neural network *can* use distributed representations to capture the componential structure and can generalize at least as well as a comparably sized HMM.

PREVIOUS APPROACHES USING NEURAL NETS

Bridle's alpha-net (Bridle, 1989) is a translation of a HMM into a recurrent neural net framework. He shows how to implement the forward pass calculations of a HMM-based recogniser (using "sigma-pi" units with a linear output) and he presents a gradient-based back-propagation training method.

As Bridle points out, this implementation of a HMM in a neural network points the way forward to other methods of constructing and training networks which offer more general non-linear structures going beyond HMM methods. Watrous (Watrous *et al.*, 1990) and Kuhn (Kuhn *et al.*, 1989) have investigated training recurrent networks for some problems in speech recognition such as phoneme discrimination, using the back-propagation learning rule. Others such as Williams and Zipser (1988) and Cleeremans *et al.* (1989) have looked at tasks which involve learning finite state automata with recurrent nets, but from the viewpoint of predicting the next symbol given left context. However, to date the issues of "time-warping" have not been directly addressed by this work.

These recurrent nets allow a "frame-by-frame" processing of the incoming data. An alternative to this is to "spatialize" time by laying out the data in an input buffer - the method used in this paper. This has a number of disadvantages, but does mean that all of the input data is readily available, whereas in recurrent nets the information on the important properties of the input must be extracted and stored in the states of the hidden units.

THE LEARNING PROCEDURE FOR THE MEAN FIELD MODULES

We assume familiarity with mean field networks and just give a brief overview here. Hopfield (1984) or Hinton (1989) give more detailed descriptions. Mean field networks use real-valued analog units with a logistic activation function that can be viewed as a deterministic approximation to the stochastic binary units used in Boltzmann machines. Mean field networks use a parallel updating algorithm to settle to a local minimum of the mean-field free energy (Hopfield, 1984). The mean field equivalent of simulated annealing is to increase the gain of each unit as the network settles. With low gains, the network will typically settle to a state in which the units have intermediate activity levels and, using an independence assumption, this state can be viewed as representing a high entropy blend of many possible binary states. In the model we describe, each such binary state in the blend represents a possible way of aligning the model of a word, stored in the tied weights, with the input data.

The input/output learning rule for MFN's (Peterson and Anderson, 1987) is based on an approximation of the Boltzmann machine learning procedure (Ackley, Hinton and Sejnowski, 1985). The replacement of stochastic binary units by deterministic real values units permits much faster learning. Below we present a different learning rule for use when the task is the classification of temporally distorted strings (which might correspond to vector-quantised time-slices of speech data) into one of N possible "word" models (word is in quotes as the entity may actually be a phoneme or some other unit – word is used for convenience only).

Each word has its own mean field module which computes a score indicating how likely it was that that model could have generated the particular string presented. Each module has weight constraints as shown in Fig. 1 that permit dynamic time warping in a similar manner to HMMs. Then the word is classified as belonging to the model that produces the highest score. The score $q_i(y)$ for module i when presented with string y is

$$q_i(y) = e^{-F_i^*(y)} \tag{1}$$

where F^* represents the free energy of the module at a minimum of free energy. This minimum is attained by performing the mean field equivalent of simulated annealing from a high temperature to $T = 1$.

With activities of the units in the range $[0, 1]$, a mean field module has free energy

$$F = -\frac{1}{2}\sum_{i,j\neq i} p_i p_j w_{ij} + T\sum_i [p_i \ln p_i + (1-p_i)\ln(1-p_i)] \tag{2}$$

where p_i is the activation of the unit i. At a minimum

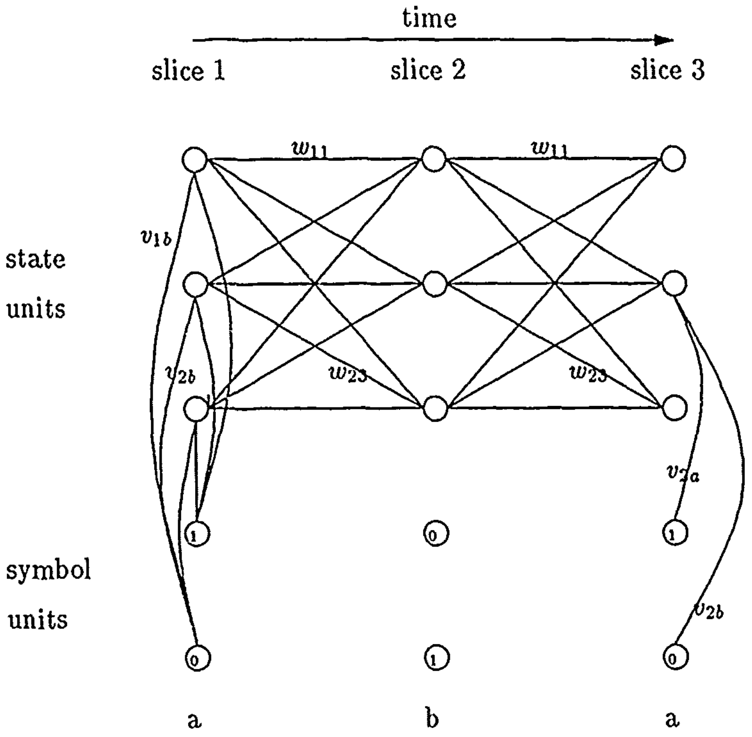


Figure 1: Part of a MFN showing some of the weight constraints (not all weights to the symbol units are shown)

of F the activities obey

$$p_i = \sigma(\frac{1}{T}\sum_j p_j w_{ij}) \tag{3}$$

where $\sigma(x)$ is the logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$.

At the minimum, the derivative of F^*_m with respect to a particular weight at $T = 1$ is given by (Hinton, 1989)

$$\frac{\partial F^*_m}{\partial w^m_{jk}} = - < p_j p_k >^m \tag{4}$$

where $< >^m$ indicates that we are considering module m. The learning rule for each module is based on increasing the normalized score for those occasions when the module is the correct generator of the string, and decreasing it otherwise. Define the normalized score to be

$$r_i(y) = \frac{q_i(y)}{\sum_{j=1}^{N} q_j(y)} \tag{5}$$

Then the objective function to be maximized by the training is

$$B = \sum_{y \in examples} \ln r_i(y) \tag{6}$$

where i indexes the correct word class. Differentiating with respect to weight w_{jk} in module m, we get

$$\frac{\partial B}{\partial w^m_{jk}} = \sum_{examples} < p_j p_k >^m [\delta_{im} - r_m] \tag{7}$$

where δ_{im} is the Kronecker delta.

This gradient can then be used by steepest ascent techniques or more sophisticated line-search/conjugate-gradient methods to maximize B. This learning procedure has the flavour of a rule that maximizes the mutual information between the input vectors and the classification, as it maximizes the $r_i(y)$'s which take into account the scores of the other modules, rather than just maximizing the score of the correct class. In fact, if the $q_i(y)$'s represent the probability of module i producing string y, then the algorithm exactly maximizes the mutual information.

THE TASK USED IN THE SIMULATIONS

To test the learning algorithm shown above, two word models were set up using HMMs with componential structure. HMMs were used because it is easy to generate and test data with HMMs, and because the Baum-Welch training algorithm (Baum *et al.*, 1970) is optimal as a *discriminant* training procedure for data generated by HMMs (Brown, 1987). This means that we can fairly compare learning by HMMs and the neural networks.

Good datasets for discriminant tasks must have similar zero order statistics (symbol frequencies), otherwise good discrimination can be achieved by HMMs with just one unit which simply detects the symbol frequencies, even though the generating HMMs had many more states. The data we used was generated by the "cross-product" of two three state HMMs. The state transition diagram for the three state HMMs is shown in Fig. 2

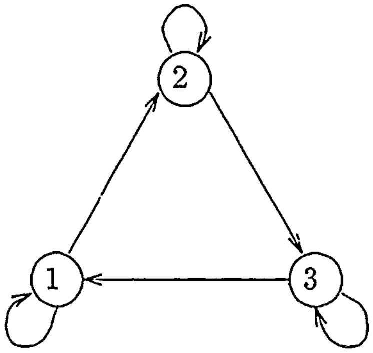


Figure 2: State transition diagram for the three state HMMs

One three state HMM called the ABC model produced symbols "a", "b" and "c" with high probability in states 1, 2 and 3 respectively. The ACB model produced "a", "c" and "b" in states 1, 2 and 3 respectively. The probability of producing the correct symbol was 0.94, and the probability of producing the other symbols was 0.03 . The transition from the start state to each of the generating states was equiprobable, so that strings generated were equally likely to begin with "a", "b" or "c".

To make componential data, one dataset was produced from the cross-product of two ABC models, and the other dataset was produced from two ACB generators. The symbols output by the cross-product HMMs are related to the two component HMMs by the following code

gen1 output : a a a b b bc c c
gen2 output : a b c a b c a b c

combined output : 1 2 3 4 5 6 7 8 9

The symbol frequencies for each of the nine symbols were similar between the data generated by the two cross-product generators. Datasets of 1000 examples of strings six symbols long generated by each cross-product HMM were used as a training set, with test and cross-validation sets also of 1000 examples each.

First order Markov models gave 28 errors on the test data. Nine state HMMs trained by the Baum-Welch algorithm gave an average of 18.25 errors and six-state HMMs gave an average of 64.68 errors with a standard deviation of 8.70 on 16 runs, with a best performance of 52 errors.

RESULTS AND DISCUSSION

Networks with six units in each hidden slice were trained on the task. Two slightly different architectures were tried. In one, the hidden units were partitioned into two fully connected groups of three units each, and the symbol units were fully connected to all units. In the other, there was no such split, all six hidden units being fully connected. The training was carried out with a conjugate-gradient with restarts algorithm, stopping when the number of errors on the cross-validation set began to increase.

The results of the simulations were 29 and 41 errors on the test data for two runs with the split weights, and 31 errors for a run with fully connected weights. These are significantly better than the results of the six state HMMs, indicating that componential structure is being discovered by the networks. Further proof of this was found by analysing the unit activities. In some of the split networks each group of three units was found to develop a distributed coding of one of the component generators' state. For the network without a split, the activities of the hidden units show that the state of one of the components is represented by the activities of all six units, the other generator's state being indicated by small modulations in these activities.

It is hard for the MFN learning rule to compete with the Baum-Welch algorithm when learning to discriminate data generated by non-componential HMMs. This is partly because the MFN learning rule is a step-size dependent method of gradient ascent, rather than a re-estimation algorithm.

With sequences generated by two three-state HMMs, the potential benefits of MFNs are small because $3+3$ is not much smaller than 3×3. We are working on further simulations with data generated by pairs of four-state HMMs, which should show the advantages of the MFNs more clearly. It is also possible to design stochastic networks with architectures similar to that of Fig. 1 that will produce data which cannot be generated by HMMs of polynomially related size.

Acknowledgements

Thanks go to Tony Plate for help with the Xerion connectionist simulator, and to Radford Neal for his HMM programs. Steve Nowlan, Evan Steeg, Tony Plate and Prof. Evangelos Milios provided helpful comments on the M.Sc thesis version of this work.

References

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains. *The Annals of Mathematical Statistics*, 41(1):164-171, 1970.

[3] J. S. Bridle. Alpha-Nets: A recurrent neural network architecture with a Hidden Markov Model interpretation. SP4 Research Note 104, Royal Signals and Radar Establishment, Great Malvern, UK, 1989. To appear in *Speech Communication* special *Neurospeech* issue, 1990.

[4] Peter Brown. *The acoustic modelling problem in Automatic Speech Recognition.* PhD thesis, Carnegie-Mellon University, 1987. Also published as IBM Research Division Technical Report RC 12750.

[5] A. Cleeremans, D. Servan-Schreiber, and J.L. McClelland. Finite state automata and simple recurrent networks. *Neural Computation*, 1(3):372–381, 1989.

[6] G. E. Hinton. Deterministic boltzmann learning performs steepest descent in weight-space. *Neural Computation, 1*:143–150, 1989.

[7] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences U.S.A.*, 81:3088–3092, May 1984.

[8] G. M. Kuhn, R. L. Watrous, and B. Ladendorf. Connected recognition with a recurrent network. In *Proceedings Neurospeech*, 1989. To appear in *Speech Communication* special *Neurospeech* issue, 1990.

[9] K. J. Lang, A. H. Waibel, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43, 1990.

[10] C. Peterson and J.R. Anderson. A mean field theory learning algorithm for neural networks. *Complex Systems*, 1:995–1019, 1987.

[11] R. L. Watrous, B. Ladendorf, and G. Kuhn. Complete gradient optimization of a recurrent network applied to /b/, /d/, /g/ discrimination. *J. Acoust. Soc. Am.*, 87(3):1301–1309, 1990.

[12] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. Technical Report ICS Report 8805, University of California at San Diego, 1988.

Energy Minimization and the Satisfiability of Propositional Logic

Gadi Pinkas
Department of Computer Science
Washington University
St. Louis, MO 63130

Abstract

Interactive connectionist networks with symmetric weights (like Hopfield nets and Boltzman Machines), use gradient descent to find a minimum for quadratic binary functions called energy functions. We show an equivalence between the problem of satisfiability in propositional calculus and the problem of minimizing those energy functions. The equivalence is in the sense that for any satisfiable Well Formed Formula (WFF) we can find a quadratic function that describes it, such that if "true" and "false" are mapped to "one" and "zero", respectively, then the set of solutions that minimize the function is equal to the set of models (truth assignments) that satisfy the WFF. We also show that in the same sense every quadratic energy function describes some satisfiable WFF. Algorithms are given to transform any propositional Well Formed Formula into an energy function that describes it and vice versa. Using Sigma-Pi units we define high order energy minimization models that are stable, and we show equivalence between those high order models and quadratic ones. In fact we show that Sigma-Pi units are not needed if additional simple hidden units are added into the network. An algorithm to convert high order energy functions into low order functions is used as a powerful tool to implement a satisfiability problem solver on a connectionist network. The results give better understanding of the role of hidden units and of the limitations and the capabilities of these interactive connectionist models. The techniques developed here for the satisfiability problem may be applied as well to a wide range of other problems.

1 Introduction

The problem of satisfiability in propositional calculus is to decide whether there exists a truth assignment (a model) for the variables of a given propositional well formed formula (WFF), such that the formula is evaluated to be true. In many cases it is not enough just to decide whether a WFF is satisfiable or not. A truth assignment that satisfies it is also desired. It is well-known that any of the problems in $\mathcal{NP}$ can be reduced to the satisfiability problem and that satisfiability is $\mathcal{NP}-$ complete (Garey, Johnson 79).

Apart from its theoretical importance, satisfiability has its direct applications. In the area of AI for example, logic is used to represent knowledge, and inference mechanisms are used to draw conclusions from this knowledge. Inferring what *must* be the truth values of the atomic propositions for a knowledge base to be consistent lets one further decide whether novel, compound WFFs logically follow from the knowledge base or contradict it. Thus, a connectionist system that solves satisfiability can be used as the engine of a theorem prover.

In this paper we prove an equivalence between the satisfiability problem and the problem of energy minimization. The equivalence means that in order to decide whether a WFF is satisfiable and to find a truth assignment that satisfies it, we can find a global minimum to some "quadratic binary" energy function such that the values of the variables of this function when the minimum is reached can be translated into a model that satisfies the original WFF. Also, any connectionist quadratic energy minimization problem may be described as a satisfiable WFF that is satisfied for the same models that cause the function to reach global minima. Algorithms will be given for converting any satisfiability problem into an energy minimization problem and vice versa.

Finding minima for a "quadratic binary" function is the essence of some connectionist models used for parallel constraint satisfaction (Hopfield 82),(Hopfield

84), (Hinton, Sejnowski 86). They are characterized by recurrent network architecture, symmetric weight matrix (with zero diagonal) and a quadratic energy function that should be minimized (Lyapunov function). Each unit asynchronously computes the gradient of the function and adjusts its activation value, so that energy decreases monotonically. The network eventually reaches equilibrium, settling on either a local or a global minimum. We call this family of models "energy minimization" models. It has been demonstrated by Hopfield and Tank (Hopfield, Tank 85), that certain complex optimization problems can be approximated by this kind of network, and some of the work done in connectionist reasoning and knowledge representation has used these energy minimization models. (For examples see: (Ballard 86), (Touretzky, Hinton 88), (Derthick 87)).

There is a direct mapping between these models and quadratic energy functions, and most of the time we will not distinguish between the function and the network that minimizes it. Thus, the equivalence between energy minimization and satisfiability means that everything that can be stated as satisfiability of some WFF *and nothing more* can also be expressed in these connectionist models. The techniques described are used in this paper for the direct implementation of a satisfiability problem solver on connectionist networks, however they may also be used for the application of logic reasoning, construction of arbitrary associative memories, and other applications.

2 Satisfiability and models of propositional formulas

A Well Formed Formula (WFF) is an expression that combines atomic propositions (variables) and connectives $(\vee, \wedge, \neg, \rightarrow, (,))$. A model (or a truth assignment) is a vector of binary values that assigns "one" ("true") or "zero" ("false") to each of the variables. A WFF φ is satisfied by a model $\bar{x}$ if its characteristic function H_φ evaluates to "one" given the vector $\bar{x}$.

The characteristic function is defined to be: $H_\varphi : 2^n \rightarrow \{0,1\}$ such that:

- $H_{x_i}(x_1, \ldots, x_n) = x_i$

- $H_{(\neg\varphi)}(x_1, \ldots, x_n) = 1 - H_\varphi(x_1, \ldots, x_n)$

- $H_{(\varphi_1 \vee \varphi_2)}(x_1, \ldots, x_n) = H_{\varphi_1}(x_1, \ldots, x_n) + H_{\varphi_2}(x_1, \ldots, x_n) - H_{\varphi_1}(x_1, \ldots, x_n) \times H_{\varphi_2}(x_1, \ldots, x_n)$

- $H_{(\varphi_1 \wedge \varphi_2)}(x_1, \ldots, x_n) = H_{\varphi_1}(x_1, \ldots, x_n) \times H_{\varphi_2}(x_1, \ldots, x_n)$

- $H_{(\varphi_1 \rightarrow \varphi_2)}(x_1, \ldots, x_n) = H_{(\neg\varphi_1 \vee \varphi_2)}(x_1, \ldots, x_n)$

We denote that a model $\bar{x}$ satisfies φ by $\varphi(\bar{x}) = 1$. The problem of satisfiability of a WFF φ is to find an $\bar{x}$ (if one exists) such that $\varphi(\bar{x}) = 1$.

3 Equivalence between WFFs

We call the atomic propositions that are of interest for a certain application "visible variables". We can add additional atomic propositions called "hidden variables" without changing the set of relevant models that satisfy the WFF. The set of models that satisfy φ projected onto the visible variables is then called "The set of visible satisfying models" $(\bar{x} \mid (\exists \bar{t})\varphi(\bar{x}, \bar{t}) = 1)$.

Two WFFs each with n visible variables are equivalent if the set of visible satisfying models of one is equal to the set of visible satisfying models of the other.
Formally:
$\varphi_1(x_1 \ldots x_n, t_1 \ldots t_k) \approx \varphi_2(x_1 \ldots x_n, t'_1 \ldots t'_{k'})$ if for every model $\bar{x}$ of $x_1, \ldots, x_n, t_1, \ldots, t_k$ that satisfies φ_1, there exists some truth assignment $\bar{t}'$ of $t'_1, \ldots, t'_{k'}$ such that $\varphi_2(x_1 \ldots x_n, t'_1 \ldots t'_{k'})$ is satisfied and; for every model $\bar{x}$ of $x_1, \ldots, x_n, t'_1, \ldots, t_{k'}$ that satisfies φ_2, there exists a truth assignment $\bar{t}$ of the hidden variables of $t_1, \ldots, t_k$ such that $\varphi_1(x_1, \ldots, x_n, t_1, \ldots, t_k)$ is satisfied.

Next we will see that every WFF φ is equivalent to a WFF ψ in a conjunction of triples form.

4 Conversion of a WFF into Conjunction of Triples Form (CTF)

A WFF φ is in CTF if $\varphi = \bigwedge_{i=1}^m \varphi_i$ and every φ_i is a sub-formula of at most three variables.

EXAMPLE 4.1 $(A \vee (B \vee (\neg C))) \wedge (T \vee ((\neg E) \wedge (\neg B))) \wedge ((\neg T) \rightarrow C)$ is in CTF.

Every WFF can be converted into an equivalent WFF in CTF. Intuitively, if a sub-formula has more then 3 variables, we generate a new hidden variable for every binary logical connective (eg: $\vee, \wedge, \rightarrow$). We "name" the binary logical operation with the new hidden variable using the logical connective "if and only if" $(\leftrightarrow)$. We perform this operation on the parse tree of the sub-WFF bottom up, except from the top most binary connective, each time we generate a new sub-formula of only two binary connectives (the original and the $\leftrightarrow$). The new hidden variables that we create are used to refer branches of the parse tree that were converted before. The sub-formulas are connected with boolean AND $(\wedge)$. Thus each sub-formula has three variables at most and the conjunction of all sub-formulas is in CTF. A formal algorithm and proof is given in (Pinkas 90).

EXAMPLE 4.2 Converting $\varphi = (((\neg(\neg A)) \wedge B) \rightarrow ((\neg C) \rightarrow D))$ into conjunction of triples generates:

From $((\neg(\neg A))\wedge B)$ we generate. $(((\neg(\neg A))\wedge B)\leftrightarrow T_1)$.
From $((\neg C) \rightarrow D)$ we generate: $(((\neg C) \rightarrow D)\leftrightarrow T_2)$.
For the top most connective $(\rightarrow)$ we generate: $(T_1 \rightarrow T_2)$.
The conjunction of these sub-formulas is :
$(((\neg(\neg A))\wedge B)\leftrightarrow T_1)\wedge(((\neg C) \rightarrow D)\leftrightarrow T_2)\wedge(T_1 \rightarrow T_2)$.
It is in CTF and is equivalent to φ.

5 Energy functions

A k-order energy function is a function $E : \{0,1\}^n \rightarrow \mathcal{R}$ that can be expressed in a sum of products form with product terms of up to k variables[1]. We denote the sum of products form for a k-order energy function by:

$E^k(x_1,\ldots,x_n)$
$= \sum_{1\leq i_1<i_2<\cdots<i_k\leq n} w_{i_1,\ldots,i_k}x_{i_1}\cdots x_{i_k}$
$+ \sum_{1\leq i_1<\cdots<i_{k-1}\leq n} w_{i_1,\ldots,i_{k-1}}x_{i_1}\cdots x_{i_{k-1}} + \cdots$
$+ \sum_{1\leq i\leq n}^{1} w_i x_i + w$

Quadratic energy functions are special cases of energy functions in the form :

$$\sum_{1\leq i<j\leq n} w_{ij}x_ix_j + \sum_{i\leq n} w_ix_i + w.$$

We can arbitrarily divide the variables of an energy function into two sets (like we did to the WFFs in section 3):

1. Visible variables are usually of interest to an observer. An instantiation of these "external" variables is considered to be an answer to the problem.
2. Hidden variables are usually not of interest to an external observer.

We call the set of minimizing vectors projected onto the visible variables, "the set of visible solutions" of the minimization problem.
$(\{\bar{x} \mid (\exists\bar{t})E(\bar{x},\bar{t}) = min_{\bar{y},\bar{t}'}\{E(\bar{y},\bar{t}')\}\})$.

There is a direct mapping between the quadratic energy functions described above and connectionist networks with symmetric weights that minimize them, like Hopfield nets and Boltzman machines. Given a function we can create the network and given a network we can generate the appropriate function. The variables of the function map into nodes in a graph. Each node is assigned a neuron unit and is connected by symmetric arcs to other units. Unit i is connected to unit j by a weight w iff the energy function includes a term of the form: $-wx_ix_j$. A unit i has a non zero threshold w iff the energy function includes a term of the form: wx_i.

EXAMPLE 5.1 The energy function $E = -2NT - 2ST - 2WT + 5T + 2RN - WN + W + S - R - N$ is represented by the symmetric network that appears in figure 1.

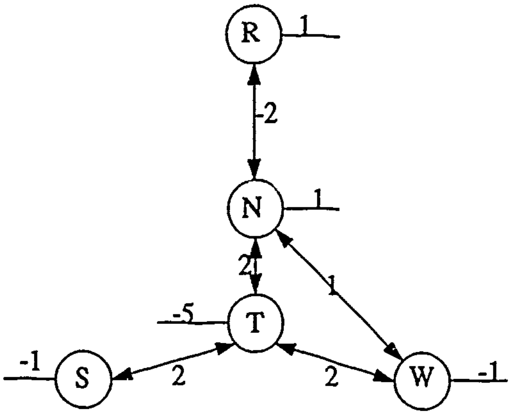


Figure 1: A quadratic network that represents the energy function $E = -2NT - 2ST - 2WT + 5T + 2RN - WN + W + S - R - N$. T is a hidden unit

Further, we can extend these interactive connectionist models to minimize also high order functions (Sejnowski 86). In the extended model the variables become nodes in a hyper graph and a weighted term $w_{i_1\cdots i_k}x_{i_1}\cdots x_{i_k}$ becomes a weight $-w_{i_1,\ldots,i_k}$ of a symmetric hyper arc that connects the nodes: $x_{i_1},\ldots,x_{i_k}$. Each node is assigned a Sigma-Pi processing unit (Rumelhart, Hinton, McClelland 86) that updates its activation value using:

$$a_i = F(net_i)$$

where F is the activation rule (threshold, sigmoid, stochastic...) that is used in the model and

$$net_i = -\sum_{i_1\cdots i\cdots i_k} w_{i_1,\ldots,i,\ldots,i_k} \prod_{1\leq j\leq k, i_j\neq i} x_{i_j}$$

For example , the binary Hopfield model will set the activation to one if $net_i > 0$, to zero if $net_i < 0$ and will not change it if $net_i = 0$.

A Sigma-Pi unit of order k multiplies up to k inputs in each term it sums and considers the weight only if all these inputs are ones. We can show that in the extended model, like in the quadratic one, energy decreases monotonically and equilibrium is reached at either global or local minimum. (The high order energy function is the Lyapunov). Note that the continuous case (when values are allowed to be between zero and one), is also captured in the extended model. The lack of terms with powers of variables (like wXY^2Z or uY^2) causes the minimizing solutions to be at the corners of the unit hyper-cube.

[1] The name "multi linear functions" is sometimes used.

Example 5.2 The cubic energy function $E = -NSW + 2RN - WN + W + S - R - N$ is represented by the hyper graph of figure 2. The units N, S, W are connected by a hyper-arc and therefore they are Sigma-Pi units.

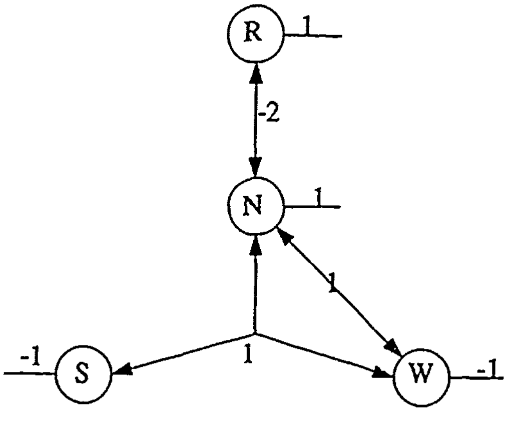


Figure 2: A cubic network that represents $E = -NSW + 2RN - WN + W + S - R - N$ using Sigma-Pi units.

We'll see now that high order models are equivalent to quadratic models and that there is a tradeoff between the order and the number of hidden units.

6 The equivalence between high order models and low order models

Infinitely many energy functions (networks) seem to solve only a single minimization problem.
We call energy functions that have the same set of visible solutions, *equivalent.*
Formally: E_1 is *equivalent* to E_2 ($E_1 \approx E_2$) iff $\{\bar{x} \mid (\exists \bar{t}_1) E_1(\bar{x}, \bar{t}_1) = min_{\bar{y}\bar{t}}\{E_1(\bar{y}, \bar{t})\}\} = \{\bar{x} \mid (\exists \bar{t}_2) E_2(\bar{x}, \bar{t}_2) = min_{\bar{y}\bar{t}}\{E_2(\bar{y}, \bar{t})\}\}$

Example 6.1 $aXY + bYZ - XYZ \approx aXY + bYZ - 2XT - 2YT - 2ZT + 5T$ for any a,b .

We will show now that any high order energy function is equivalent to a low order one with additional hidden variables. An algorithm is given for the conversion of a high order energy function to a low order one. In addition, another algorithm is given for transforming a low order energy function into a (possibly) higher one by elimination of some or all of the hidden variables. (Formal proofs are given in (Pinkas 90)). This algorithms allow us to trade the computational power of Sigma-Pi units for additional simple units, and vice versa.

Note that we could have used a stronger definition for equivalence of energy functions. The functions that are generated by the transformations not only have the same set of minima, but have the same energy as the original function (up to a constant difference) for all instantiations of the visible variables

6.1 Converting a high order energy function into a lower order function

Every k order energy function E can be transformed into an equivalent $(k-1)$ order energy function by adding extra hidden variables. Transformation is done by replacing each of the k-order terms in E with a sum of (k-1) order expressions:

- Any k-order term $(\alpha \prod_{i=1}^{k} x_i)$, with a NEGATIVE coefficient α, can be replaced by a quadratic term of the form : $\sum_{i=1}^{k} 2\alpha X_i T - (2k-1)\alpha T$ generating an equivalent energy function with one additional hidden variable, T.
- Any k-order term $(\alpha \prod_{i=1}^{k} x_i)$, with a POSITIVE coefficient α, can be replaced by a term of order $(k-1)$ of the form : $\alpha \prod_{i=1}^{k-1} x_i - (\sum_{i=1}^{k-1} 2\alpha X_i T) + 2\alpha X_k T + (2k-3)\alpha T$, generating an equivalent energy function with one additional hidden variable[2].

Example 6.2 When term is negative: $-3XYZU \approx -6XT - 6YT - 6ZT - 6UT + 21T$.

Example 6.3 When term is positive:

$$\begin{aligned} XYZU &\approx XYZ - 2XT - 2YT - 2ZT + 2UT + 5T \\ &\approx XY - 2XT' - 2YT' + 2ZT' + 3T' \\ &\quad -2XT - 2YT - 2ZT + 2UT + 5T \end{aligned}$$

Example 6.4 The cubic energy function $E = -NSW + 2RN - WN + W + S - R - N$ is equivalent to $-2NT - 2ST - 2WT + 5T + 2RN - WN + W + S - R - N$ by using additional hidden variable T. The corresponding networks appear in figure 2 and figure 1.

6.2 Eliminating hidden variables

The symmetric transformation, from low order into high order energy, is also possible: Every k- order energy function with at least one hidden variable T, can be transformed into an equivalent (possibly) higher order energy function that does not include T, using the following method:

[2] We conjecture that the asymmetry between the positive and the negative transformations is a result of the asymmetric encoding we used for true (1) and false (0).

Assume T is a hidden variable to be eliminated.
Let $\alpha_j T \prod_{i=1}^{l_j} X_{j_i}$ be a $l_j + 1$-order term that shares an arc with T. We combine all these terms to form: $(\sum_{j=1}^{k} \alpha_j \prod_{i=1}^{l_j} X_{j_i})T$.
We replace this term with a new term that is generated using the following procedure:

Consider all instantiations $\bar{x} = (x_{i_1}, \ldots, x_{i_l})$, of the variables $X_{i_1}, \ldots X_{i_l}$ such that
$\beta_S = \sum_{j=1}^{k} \alpha_j \prod_{i=1}^{l_j} x_{j_i} < 0$, where S is an assignment for the l variables $x_{i_1}, \ldots, x_{i_l}$ (called negative assignment).
If we clamp the visible variables using a negative assignment S, T will settle on the value one (generating negative energy). All other instantiations will cause T to settle on zero (generating zero energy). We will create now a new term (without T) that will generate the same energy (β_S) for the negative assignments, and zero energy for all other assignments.

For every instantiation $(x_{i_1}, \ldots, x_{i_l})$ obtained by assignment S, let the function L_S^j be:
$L_S^j(\hat{X}) = \begin{cases} X_{i_j} & \text{if } S(X_{i_j}) = 1 \\ 1 - X_{i_j} & \text{if } S(X_{i_j}) = 0 \end{cases}$
Then, generate the term:

$$newterm = \sum_{\text{S such that } \beta_S < 0} \beta_S \prod_{j=1}^{l} L_S^j(\hat{X})$$

Replace the old term: $(\sum_{j=1}^{k} \alpha_j \prod_{i=1}^{l_k} X_{j_i})T$ with the newterm.
The term $\prod_{j=1}^{l} L_S^j(\hat{X})$ is one for assignment S and zero for all other assignments.

The new term generates the same energy as the original term for all assignments of the visible variables (not including T).

Example 6.5 Let T be the hidden variable to be eliminated, then:
$AB+TAC-TA+2TB-T = AB+T(AC-A+2B-1)$
The following assignments for (A, B, C) cause β to be less then zero:
$\beta_{(0,0,0)} = -1$; $\beta_{(0,0,1)} = -1$;
$\beta_{(1,0,0)} = -2$; $\beta_{(1,0,1)} = -1$;
The new term equals:
$-(1-A)(1-B)(1-C) - (1-A)(1-B)C - 2A(1-B)(1-C) - A(1-B)C$
$= -ABC + AB + AC - A + B - 1$
Therefore: $AB + TAC - TA + 2TB - T$
$\approx -ABC + 2AB + AC - A + B$.

Note that (1,0,0) uniquely minimizes both functions, but the energy of the two functions is equal for all assignments of the visible variables.

7 Describing WFFs by energy functions

An energy function E describes a WFF φ if the set of visible satisfying models of φ is equal to the set of visible solutions of the minimization of E.
Formally: E describes φ if $(\forall \bar{x})(((\exists \bar{t})(\varphi(\bar{x}, \bar{t}) = 1) \Longleftrightarrow ((\exists \bar{t}')E(\bar{x}, \bar{t}')) = min_{\bar{y}}\{E(\bar{y})\}))$.

Example 7.1 $((\neg A) \vee (\neg B) \vee C)$ is described by $AB - 2AT - 2BT - 2CT + 5T$
The minimizing set is:
{(0000,0001,0010,0011,0100,0101,1111)}
and after projection onto the visible variables ABC, we get {(000,001,010, 011,100,101,111)} which is the set of all the models that satisfy the WFF.

Next we show that if $\varphi(x_1, \ldots, x_n)$ is a satisfiable WFF of n variables then φ is described by some n-order energy function with no hidden variables (contradictions can not be described by any energy function).

8 The penalty function

The penalty E_φ of a WFF φ is a function $P_\varphi : V^n \rightarrow \mathcal{N}$, that gives a penalty to every subexpression of the WFF that is not satisfied. It looks at the conjunctive terms in the upper level of the WFFs structure and for every term φ_i in $\varphi_1 \wedge \varphi_2 \wedge \ldots \varphi_l$, it computes the characteristic of the negation of φ_i:

- $E_{x_i}(x_1, \ldots, x_n) = 1 - H_{x_i}(x_1, \ldots, x_n)$
- $E_{(\neg\varphi_1)}(x_1, \ldots, x_n) = H_{\varphi_1}(x_1, \ldots, x_n)$
- $E_{(\varphi_1 \vee \varphi_2)}(x_1, \ldots, x_n)$
  $= H_{((\neg\varphi_1)\wedge(\neg\varphi_2))}(x_1, \ldots, x_n)$
- $E_{(\varphi_1 \wedge \varphi_2)}(x_1, \ldots, x_n)$
  $= E_{\varphi_1}(x_1, \ldots, x_n) + E_{\varphi_2}(x_1, \ldots, x_n)$
- $E_{(\varphi_1 \rightarrow \varphi_2)}(x_1, \ldots, x_n) = E_{((\neg\varphi_1)\vee\varphi_2)}(x_1, \ldots, x_n)$

A more intuitive way to look at the penalty function is to observe that if $\varphi = \bigwedge_{i=1}^{m} \varphi_i$ then:

$$P_\varphi = \sum_{i=1}^{m} (1 - H_{\varphi_i})$$

If all terms are satisfied, P_φ gets the value zero, otherwise, the function computes the number of unsatisfied terms. The function generated may be simplified to a sum of products form of an energy function that has a maximum order of n.

It is easy to see that φ is satisfied by $\bar{x}$ iff E_φ is minimized by $\bar{x}$ and the global minimum is zero. Therefore, every satisfiable WFF φ has a function E_φ such that E_φ describes φ.

The penalty function of a contradiction does not describe it and is always greater then zero. The set of minimizing solutions is never equal to the set of satisfying models of a contradiction (the empty set). However it is equal to the set of models that satisfy a maximal consistent subset (of triples).

We can also notice that if φ is a conjunction of WFFs each of maximum k variables, then φ is described by a k-order energy function.

EXAMPLE 8.1 $E_{A\wedge(B\vee(\neg C)))}$
$= E_A + E_{(B\vee(\neg C))}$
$= (1-A) + H_{((\neg B)\wedge C)}$
$= (1-A) + (1-B)C = 1 - A + C - BC$

Because a WFF in CTF is a conjunction of subexpressions each containing at most three variables, it may be described by a cubic energy function. Since every WFF is equivalent to a WFF in CTF, we can conclude that every WFF is described by a cubic energy function.

We show now that all WFFs are described by quadratic energy functions, and that all energy functions describe some WFFs.

9 Mapping from a satisfiability problem to a minimization problem and vice versa.

9.1 Every WFF is described by some quadratic energy function.

The following algorithm transforms a WFF into a quadratic energy function that describes it by adding $O(length(\varphi))$ hidden variables.

- Convert into conjunction of triples (adding $O(length)\varphi))$ hidden variables) using the method of section 4. The variables added here reduce both the order and the fan-out of the connectionist network that we generate.
- Convert a conjunction of triples into a cubic energy function and simplify it to a sum of products form (using the penalty function of section 8).
- Convert cubic terms into quadratic terms. (adding $O(length(\varphi))$ hidden variables) using the construction of section 6.1.

EXAMPLE 9.1 Converting
$\varphi = (((\neg(\neg A)) \wedge B) \rightarrow ((\neg C) \rightarrow D))$ to conjunction of triples generates:
$(((\neg(\neg A))\wedge B)\leftrightarrow T_1)\wedge(((\neg C) \rightarrow D)\leftrightarrow T_2)\wedge(T_1 \rightarrow T_2)$.
Eliminating ↔ and → generates:
$(T_1 \vee (\neg A) \vee (\neg B)) \wedge ((\neg T_1) \vee (A \wedge B)) \wedge (T_2 \vee (\neg C)) \wedge (T_2 \vee (\neg D)) \wedge ((\neg T_2) \vee C \vee D) \wedge ((\neg T_1) \vee T_2)$
Generating the cubic energy function.
$(1-T_1)AB + T_1(1-A) + T_1(1-B) + (1-T_2)C + (1-T_2)D + T_2(1-C)(1-D) + T_1(1-T_2)$
$= -ABT_1 + CDT_2 + AB - AT_1 - BT_1 - 2CT_2 - 2DT_2 - T_1T_2 + 3T_1 + C + D + T_2$
Converting into a quadratic function.
$-2AT_3 - 2BT_3 - 2T_1T_3 + CD - 2CT_4 - 2DT_4 + 2T_2T_4 + 5T_3 + 3T_4 + AB - AT_1 - BT_1 - 2CT_2 - 2DT_2 - T_1T_2 + 3T_1 + C + D + T_2$

EXAMPLE 9.2 Our WFF is composed of a conjunction of the following sub-formulas:
$N \wedge S \rightarrow W$
$R \rightarrow (\neg N)$
$N \vee (\neg W)$
$S \rightarrow N$
$N \vee R$
We can compute the penalty function for each of the sub-formulas and then add them all:
$NS - NSW +$
$RN +$
$W - WN +$
$S - NS +$
$1 - R - N + RN$
$= -NSW + 2RN - WN + W + S - R - N + 1$.
This cubic energy function is represented by the network of figure 2. Its transformation to quadratic function generates: $-2NT - 2ST - 2WT + 5T + 2RN - WN + W + S - R - N + 1$. This quadratic function describes the same knowledge base and is represented by the network of figure 1.

The algorithm features incremental updating of the knowledge base: If we wish to add another piece of knowledge, delete a WFF or update one, we make only local changes to the network and we do not have to recompute it all over again.

EXAMPLE 9.3 To add the WFF: $N \rightarrow R$ to the knowledge base of the previous example, we need only to compute the penalty function for this WFF and add it to the previous energy function. In the case of $N \rightarrow R$ we add $N - NR$ to the previous function and the result is: $-2NT - 2ST - 2WT + 5T + RN - WN + W + S - R + 1$. If we now wish to delete $R \rightarrow (\neg N)$, we subtract RN and we get the function. $-2NT - 2ST - 2WT + 5T - WN + W + S - R + 1$. The new knowledge base is represented by the network of figure 3.

9.2 Every energy function describes some satisfiable WFF.

To complete the proof that satisfiability problems are equivalent to energy minimization problems, we need to show that for any energy function E with n visible variables and j hidden variables there exists a satisfiable WFF φ, such that E describes φ. We have shown that any energy function φ with hidden variables is equivalent to some other energy function (possibly of higher order) with no hidden variables at all. We show now that for any k-order energy function E with

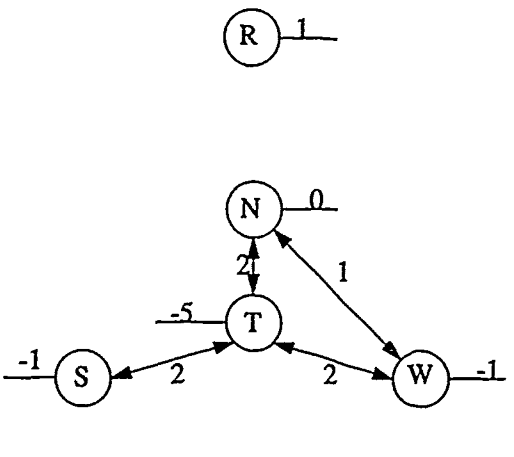


Figure 3: The energy function $-2NT-2ST-2WT+5T-WN+W+S-R+1$ captures the knowledge base after adding the WFF: $N \rightarrow R$ and deleting $R \rightarrow (\neg N)$.

no hidden variables there exists a satisfiable WFF φ such that E describes φ. A method for constructing φ is given below:

Method: Given E and n variables $\hat{X} = (x_1, \ldots, x_n)$, there are $| \{0,1\}^n | = 2^n$ possible instantiations. Compute $E(\bar{x})$ for all instantiations $S(\hat{X}) = \bar{x}\epsilon\{0,1\}^n$ and find $min_{\bar{x}}\{E(\bar{x})\} = min_E$

Construct a boolean function:

$$H_E(\hat{X}) = \begin{cases} 1 & \text{if } E(\hat{X}) = min_E \\ 0 & \text{otherwise} \end{cases}$$

Build a WFF: $\varphi = \bigvee_{H_E(S(\hat{X}))=1}(\bigwedge_{i=1}^n L_S^i)$ that is characterized by H_E.

Where $L_S^i = \begin{cases} X_i & \text{if } S(X_i) = 1 \\ (\neg X_i) & \text{if } S(X_i) = 0 \end{cases}$

φ is a disjunction of terms. Each term is of the form. $t_S = (\bigwedge_{i=1}^n L_S^i)$ (This method proves existence of such WFF although the algorithm is very inefficient). Note, that it is only the minima that cause a function to describe a WFF. The exact energy surface is irrelevant for this purpose.

Example 9.4 $E(X,Y) = -XY + 1.5X$
Trying all instantiations:
$E(0,0) = 0; E(0,1) = 0;$
$E(1,0) = 1.5; E(1,1) = 0.5$
The characteristic function will be:
$H(0,0) = 1; H(0,1) = 1; H(1,0) = 0; H(1,1) = 0$
The WFF that is described by E is therefore: $(((\neg X)\wedge(\neg Y)) \vee ((\neg X) \wedge Y))$

9.3 Complexity

The algorithm to convert a WFF into a network of simple units is an efficient one and generates a network with economical size and fan-out.

We define the size of the generated network as the number of hidden units, and we define the fan-out of a variable to be the number of different multi-variable terms shared by a variable.

The conversion to CTF generates new variables proportional to the number of connectives in the WFF. The number of hidden variables that are generated by the conversion of a cubic function into quadratic function is also on the order of the number of binary connectives. The reason for this is that a sub-formula of 3 variables generates one cubic term that generates in turn only one hidden unit. As a result we conclude that the number of hidden units that are generated is linear in the length of the original WFF.

The fan-out of all these hidden variables is bounded by six for those generated for CTF and bounded by three for those generated by conversion into a quadratic function.

10 Summary, applications and conclusions

We have shown an equivalence between the problem of satisfiability of propositional calculus and the problem of minimizing connectionist energy functions.

Any propositional satisfiable WFF can be described by an order n energy function with no hidden variables, or by a quadratic energy function with additional hidden variables. Any quadratic (or higher order) energy function with some hidden variables is equivalent to a possibly higher order energy function with no hidden variables, and every energy function describes some satisfiable WFF. We have presented algorithms to convert a given WFF into a quadratic energy function that produces hidden variables with fan-out bounded by a constant. The number of hidden units that are generated is linear in the length of the WFF. Using this algorithm we can determine the topology and the weights of a connectionist network that represents and "solves" [3] a given satisfiability problem. What the energy landscape looks like, and how easy is it for the network to escape from local minima are areas for future research.

High order, stable connectionist models with Sigma-Pi units that minimize high order functions can be constructed. We have also identified procedures to transform high order energy functions into quadratic energy functions and vice versa, (by adding or eliminating hid-

[3] We assume that the model is capable of escaping from local minima.

den variables).
As a result, we may conclude that these Sigma-Pi units are not needed, if we agree to add more hidden units into the network. We can also eliminate hidden units by adding Sigma-Pi capab. ities to some of the other units.

As a result of the equivalence relations defined both on energy functions and on WFFs, we can show that the space of connectionist energy functions is divided into equivalence classes and so is the space of propositional WFFs. There is a one to one mapping (bijection) between the set of equivalence classes of energy functions and the set of classes of satisfiable WFFs. The cardinality of these two sets is $2^{(2^n)} - 1$. There is also a bijection between the set of non-zero boolean functions and the set of energy classes since a non-zero boolean function characterizes a satisfiable WFF which can be described by a class of energy functions.

We may also conclude a limitation to the kind of problems energy minimization connectionist networks *can* solve. Only those problems that can be stated as satisfiability problems of propositional calculus (and every such problem) can also be stated as energy minimization problems.

Several applications may benefit from the techniques developed here. We'll mention briefly two of them. associative memory and finding a maximal consistent subset.

We can construct a network implementing any boolean function in a size that is proportional to the size of the boolean implementation of the function (using logical gates). Assuming $\varphi(x_1, \ldots, x_n)$ is the boolean implementation of the given function then applying the methods described in this paper to the WFF: $(out \leftrightarrow \varphi)$ will produce a connectionist implementation of the function. Clamping the variables $x_1, \ldots, x_n$ will cause the unit "out" to get the value of the function when the network reaches equilibrium (assuming it can escape from local minima). By clamping $out = 1$ and supplying only a portion of the inputs, this mechanism features also pattern completion and thus can be used as an associative memory. For any set of binary vectors we wish to store in the associative memory, we construct the boolean implementation that outputs one for any member of the set and zero otherwise. If we can minimize this boolean implementation to polynomial size, we can also build an associative memory of polynomial size.

As a last example consider a set of possibly contradicting logic beliefs (Derthick 87), we can construct a network that will search for a maximal consistent subset and thus be used for reasoning from a conflicting set of beliefs. In the case where no priorities are assigned to the beliefs, the maximal consistent subset is chosen to be a consistent subset of WFFs with maximal cardinality. However, in the general case each of the beliefs may be assigned a penalty (representing degree of belief for example) and a total order of preference between beliefs is determined. Contradicting WFFs compete among themselves and the network defeats some of .he beliefs in favor of others. At equilibrium, providing a gloLal minimum was found, the network finds a consistent subset of beliefs such that the total penalty is minimized.

Acknowledgments

I thank William Ball, Dan Kimura, Arun Kumar, Stan Kwasny, Ron Loui and Dave Touretzky for helpful comments and suggestions. I thank also McDonnell Douglas Co., Southwestern Bell Co., and Mitsubishi Electronics America, Inc. the sponsors of the Center for Intelligent Computing Systems.

References

D. H. Ballard "Parallel Logical Inference and Energy Minimization" *Proceedings of the 5th National conference on Artificial Intelligence* Philadelphia, Pa., August 1986, pp. 203-208

M. Derthick "A Connectionist Architecture for representing and Reasoning about Structured Knowledge" *Proceedings of the Ninth Cognitive Science Society*,1987, Seattle, WA

M.R Garey, D.S Johnson *"Computers and Intractability - A Guide to the theory of NP- Completeness"* (W.H. Freeman and Company San Francisco) 1979

G.E Hinton and T.J. Sejnowski "Learning and Relearning in Boltzman Machines" in J. L. McClelland, D. E. Rumelhart *"Parallel Distributed Processing:Explorations in the Microstructure of Cognition"* Vol I pp. 282 - 317 MIT Press 1986

J.J. Hopfield "Neural networks and physical system with emergent collective computational abilities," *Proceedings of the National Academy of Sciences* USA, 1982,79,2554-2558.

J.J. Hopfield "Neurons with graded response have collective computational properties like those of two-state neurons". *Proceedings of the National Academy of Sciences* USA, 1984,Vol 81, pp. 3088-3092.

J.J. Hopfield, D.W. Tank "Neural Computation of Decisions in Optimization Problems" *Biological Cybernetics*, Vol 52, pp. 144-152.

G. Pinkas, "Energy Minimization and the Satisfiability of Propositional Calculus " *Technical Report. WUCS-90-03, Department of Computer Science, Washington University* 1990

D.E. Rumelhart,G.E Hinton and J. L. McClelland "A General Framework for Parallel Distributed Processing" in J. L. McClelland, D. E. Rumelhart *"Parallel*

Distributed Processing:Explorations in the Microstructure of Cognition" Vol I pp. 282 - 317 MIT Press 1986

D.S. Touretzky, G.E. Hinton "A distributed connectionist production system" *Cognitive Science* 12(3):423-466.

T.J. Sejnowski "High-order Boltzman Machines", Neural Networks for Computing, *Proceedings of the American Institute of Physics* 151, Snowbird (Utah) pp 3984.

Part II

Reinforcement Learning

On the Computational Economics of Reinforcement Learning

Andrew G. Barto
Dept. of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Satinder Pal Singh
Dept. of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Abstract

Following terminology used in adaptive control, we distinguish between *indirect* learning methods, which learn explicit models of the dynamic structure of the system to be controlled, and *direct* learning methods, which do not. We compare an existing indirect method, which uses a conventional dynamic programming algorithm, with a closely related direct reinforcement learning method by applying both methods to an infinite horizon Markov decision problem with unknown state-transition probabilities. The simulations show that although the direct method requires much less space and dramatically less computation per control action, its learning ability in this task is superior to, or compares favorably with, that of the more complex indirect method. Although these results do not address how the methods' performances compare as problems become more difficult, they suggest that given a fixed amount of computational power available per control action, it may be better to use a direct reinforcement learning method *augmented* with indirect techniques than to devote all available resources to a computationally costly indirect method. Comprehensive answers to the questions raised by this study depend on many factors making up the economic context of the computation.

1 INTRODUCTION

In its simplest form, reinforcement learning is based on the commonsense idea that if an action is followed by a satisfactory state of affairs, or an improvement in the state of affairs (as determined in some clearly defined way), then the tendency to produce that action is strengthened, i.e., reinforced. This idea plays a fundamental role in theories of animal learning and is elaborated mathematically in the theory of learning automata (Narendra and Thathachar, 1989). Embedding this idea within a framework for associative learning includes a role for stimulus patterns in eliciting actions (Barto, Sutton, and Brouwer, 1981; Barto and Anandan, 1985; Klopf, 1982). Extending reinforcement learning further, it is possible to specify the idea of being "followed by a satisfactory state of affairs" in terms of the long-term consequences of an action, or of a policy for performing actions, instead of simply short-term consequences. By combining methods for adjusting action-selection rules with methods for estimating the long-term consequences of actions, reinforcement learning methods can be devised that are applicable to control problems involving temporally extended behavior (e.g., Anderson, 1987; Barto, Sutton, and Anderson, 1983; Barto, Sutton and Watkins, 1990, to appear; Hampson, 1989; Jordan and Jacobs, 1990; Sutton, 1984; Watkins, 1989; Werbos, 1987; Witten, 1977a, b).

Although control architectures based on reinforcement learning can be quite complex, including components permitting off-line, look-ahead planning (Sutton, 1990), reinforcement learning is usually regarded as a very simple and direct method for adjusting behavior. The utility of simple, direct learning methods as compared to the utility of more complex methods depends upon the particular algorithms in question, specific characteristics of the ensemble of tasks of interest, as well as a host of other factors influencing the outcome of possible cost-benefit analyses. Are the performance improvements expected of a "sophisticated" learning method going to be worth its additional computational cost? What would happen if available computational power were used to implement many different simple learning methods instead of a few complex methods? Do conditions, in fact, favor learning at all as opposed to a hand-crafted solution? Questions such as these, which we regard as involving the computational economics of learning, cannot be answered independently of the relevant context, but they address factors that play major roles in shaping biological systems and that should play major roles in the design of artificial systems.

The simulations described in this paper were motivated by a simple, but nevertheless unanswered, question about the relative efficiency of two approaches to learning how to solve a particular type of stochastic control problem. Following terminology used in adaptive control (e.g., Goodwin and Sin, 1984), we distinguish between *indirect* learning methods, which learn explicit models of the dynamic structure of the system to be controlled, and *direct* learning methods, which do not.[1] Indirect methods estimate unknown parameters describing the system to be controlled and define a control rule in terms of these estimates; that is, they employ a system identification procedure to form a model of the system together with a control design procedure that is executed on-line to compute the current control rule from the current system model.[2] The need for the repeated execution of this design procedure is what justifies the term indirect. Direct methods, on the other hand, estimate parameters that directly specify the control rule instead of the system to be controlled. Although we knew that direct methods based on reinforcement learning require less computation for each control action than indirect methods, we did not know, even for small artificial control problems, how the performance of such a direct method would compare with that of a more conventional indirect method in terms of the number of control actions required for learning.

We compared two learning methods that are as similar as possible except that one is indirect and the other is a direct method utilizing reinforcement learning. We applied them to an infinite horizon Markov decision problem with unknown state-transition probabilities. The simulation results show that although the direct method requires much less space and dramatically less computation per control action, its learning ability in this task is superior to, or compares favorably with, that of the more complex indirect method. Because our simulation results comparing these methods were obtained on a single small example of a Markov decision problem, they do not address how the methods' performances compare as problems become larger and/or more difficult. However these results demonstrate that direct reinforcement learning methods are not necessarily less capable than much more complex indirect methods, and they raise questions, which we discuss below, about the computational economics of learning.

The indirect method we implemented is that Sato, Abe, and Takeda (1988), which performs system identification and uses dynamic programming (DP) to estimate optimal actions from the system model. The direct method we implemented replaces the DP component of the Sato et al. method with Watkins' Q-Learning algorithm for incrementally approximating the results of DP (Watkins, 1989). We call the resulting direct reinforcement learning algorithm the Exploratory Q-Learning, or EQ, algorithm. We selected the method of Sato et al. for this study because its action-selection component is readily adaptable to direct methods. However, in fairness, we note that the contribution of Sato et al. (1988) is a convergence theorem for their algorithm rather than a demonstration of its efficiency, and here we do not prove a comparable convergence result for the EQ algorithm (although such a result can be proved, as we will report in a forthcoming article).

In this paper we do not address all of the issues that are critical in more elaborate applications of the learning methods that we discuss. We assume that the states of the Markov chain underlying the decision problem are completely and unambiguously observable, thereby eliminating from consideration the important issues for control raised by incomplete state information. We also assume that representation and storage of information is accomplished in simple lookup table form. More general representation and storage schemes involve the kinds of parameterized models and distributed representations that may make artificial neural networks useful for these types of control problems. We assume the reader can extrapolate from what we present here to relate our observations about the economic context of reinforcement learning to methods implemented by artificial neural networks.

2 INDIRECT AND DIRECT ADAPTIVE CONTROL

For some approaches to adaptive control, the distinction between indirect and direct methods amounts to little more than the difference between expressing the control rule in terms of the parameters of the system model on-line during learning (the indirect case) or off-line before the start of learning (the direct case). If the computation required by the control design procedure is relatively simple, as it is in many adaptive control methods, the distinction between direct and indirect methods has minor impact on computational cost.

Here, however, we are interested in control tasks in which this computation can be extremely costly. This occurs when the control objective is not to make the controlled system closely follow a specified reference trajectory the kind of task which is most widely studied in adaptive control but to control the system to maximize a measure of long-term performance not in-

[1] This distinction parallels that between parametric and non-parametric approaches to pattern classification (e.g., Duda and Hart, 1973). Watkins (1989) made a similar distinction between *model-based* and *primitive* learning methods, terminology we adopted in Barto and Singh (1990).

[2] A control design procedure is any method for determining a control rule based on a system model and performance specifications.

volving a prespecified trajectory. For nonlinear systems, solving these optimal control problems requires extensive computation even if the system to be controlled is completely known. In the general case, a search has to be conducted in the space of all possible trajectories, which grows explosively as a function of the number of control actions, system states, and the time-horizon of the task. For the problems in which we are interested, here illustrated by Markov decision problems, this complexity can be dramatically reduced by applying DP methods, but the computational complexity (both space and time complexity) still remains a critical limitation.

If the system to be controlled is not completely known and the control design procedure is costly for all members of the class of system models under consideration, then the computational requirements become especially severe. One strategy for such problems is to abandon the goal of performing learning on-line while the system is being controlled. A separate system identification phase can be completed to a satisfactory degree of accuracy, and then the control design procedure can be executed *once* based on the resulting system model. This is essentially the traditional non-adaptive approach in which system modeling and control are considered as separate tasks.

For learning on-line during control, an indirect method requires the repeated application of the costly design procedure (such as DP) during learning as the system model is updated. In such cases, therefore, direct methods can have significant advantages by eliminating the need for the repeated application of the design procedure. Unfortunately, for the optimal control problems in which we are interested, in the absence of restrictive assumptions, there is no known way to precompute an optimal control rule in terms of a system model to form an easy-to-evaluate function of parameter estimates. Stated differently, except in special cases, there is no known analytical way to circumvent the required search in the space of trajectories.

It is possible, however, to reorganize this search by distributing it differently over system states, control actions, and time. This is the basis of direct approaches to adaptive optimal control, such as the EQ algorithm described below, which use incremental DP methods. The intuition underlying these approaches is that it is not worth the computational effort to perform extensive long-term planning based on highly uncertain information.

3 MARKOV DECISION PROBLEMS

A Markov decision problem is defined in terms of a discrete-time stochastic dynamical system with finite state set $\{1, \ldots, N\}$. At each time step a controller observes the system's state and selects an action from the action set $A = \{1, \ldots, K\}$ (where we simplify slightly by not letting this set depend on the observed state). If i is the observed state and action k is selected, the state at the next time step will be j with probability p_{ij}^k. We further assume that under action k, a transition from state i to state j produces a payoff r_{ij}^k, where $|r_{ij}^k| < \infty$ for each i, j, and k.[3] The controller can implement a state feedback control law, called a policy, to provide a control action at each time step as a function of the observed state. A stationary policy, denoted $U = (u_1, \ldots, u_N) \in A^N$, specifies that the controller performs action u_i when state i is observed. The stochastic system together with a stationary policy U define a stationary finite state Markov chain with probability $p_{ij}^{u_i}$ of making a transition from state i to state j.

For any stationary policy U and state i, let v_i^U denote the expected *infinite-horizon discounted return*, which we simply call the *return*, for state i given policy U. Letting $r(t)$ denote the payoff at time t, this is defined as follows:

$$v_i^U = E_U \left[\sum_{t=0}^{\infty} \gamma^t r(t) | i(0) = i\right], \qquad (1)$$

where $i(0)$ is the system's initial state, γ, $0 \leq \gamma < 1$ is a factor used to discount future payoffs, and E_U is the expectation assuming the controller always uses policy U. It is usual to call v_i^U the *value* of i under policy U. The function assigning values to states is called the *value function* corresponding to the given policy. The objective of the type of Markov decision problem considered here is to find a policy that maximizes the value of each state i defined by (1). A policy that achieves this objective is an optimal policy which, although not always unique, is denoted $U^* = (u_1^*, \ldots, u_N^*)$. It can be shown that for the formulation given here, all optimal policies are stationary (e.g., Bertsekas, 1987).

Given a complete and accurate model of a Markov decision problem in the form of knowledge of the transition probabilities, p_{ij}^k, and the payoff array, r_{ij}^k, for all states i and j and actions k, it is possible to solve the decision problem by applying one of various DP methods as described, for example, by Bertsekas (1987). Indeed, in the absence of assumptions other than those described above about the structure of the decision problem, DP methods are the only exact methods applicable short of exhaustive searches through the space of all policies. The method of Sato et al. makes use of the DP algorithm called policy iteration, which computes a sequence of improving policies. At each iteration, the value function for the current policy must be computed either by a successive approximation method or by inverting an $N \times N$ ma-

[3] In more general formulations, each payoff r_{ij}^k is generated by a random process that depends on i, j, and k, but we follow Sato et al. (1988) and restrict attention to the case in which the payoff process is deterministic.

trix. The process converges to an optimal policy after a finite number of iterations.

4 INDIRECT AND DIRECT LEARNING FOR MARKOV DECISION PROBLEMS

When a complete model of the decision problem is unavailable, it is necessary to learn about the problem while interacting with the system defining it. Indirect learning methods are the most widely studied. They rely on state-transition models formed by estimating the state-transition probabilities for each control action. These estimates can be formed while the controller is interacting with the system by keeping track of the frequencies with which the various state transitions occur for the various control actions. Indirect methods also require estimating the payoffs r_{ij}^k, for each combination of current state, i, next state, j, and action, k.[4] Indirect methods are based on the certainty equivalence principle of computing and using policies that would be optimal if the current transition probability estimates were correct (Bertsekas, 1987). Most of the methods for the adaptive control of Markov processes described in the engineering literature are indirect (e.g., Borkar and Varaiya, 1979; Kumar and Lin, 1982, Mandl, 1974, Riordon, 1969, Sato-Abe-Takeda, 1982, 1985, 1988).

Although reinforcement learning methods can utilize models in a variety of ways, most such methods are classified as direct because they do not use state-transition models. In a direct learning method, there is no possibility for performing any computation that explicitly requires "thinking about" state transitions without actually causing the controlled system to execute them. Ruled out, therefore, are any methods using conventional DP or heuristic search algorithms. Most examples of direct methods for learning how to solve Markov decision problems make use of stochastic learning automata (e.g., Lakshmivarahan, 1981, Narendra and Thathachar, 1989; Wheeler and Narendra ,1986; Witten, 1977a, b).

Although direct reinforcement learning methods do not use state-transition models, they can use value function models, which we call *value models* in what follows. The simplest methods use the value model's output to evaluate and reinforce control actions as they are performed. The pole-balancing system of Barto, Sutton, and Anderson (1983) illustrates this approach. After each control action, the value model is updated based on the immediate payoff and the value estimate of the next state using an "adaptive critic method." This class of value-estimation methods, developed by Sutton (1984, 1988), who also calls them "temporal difference methods," are related to methods proposed earlier by Klopf (1972), Witten (1977a, b), and Werbos (1977). Werbos has discussed these methods in terms of DP and calls them "heuristic dynamic programming" methods. Similar connections to DP were recently described by Watkins (1989), who uses the term "incremental dynamic programming." This general class of methods is discussed in terms of DP by Barto, Sutton, and Watkins (1990, to appear) and Werbos (1987, 1988, 1989), who also provide references to related research by others.

Another way of using a value model is illustrated by Watkins' (1989) Q-Learning method, described below, which forms a different kind of value model to provide a return estimate for each state/action pair. The output of this value model for a state/action pair is an estimate of the expected return assuming that the given action is performed for the given state, and that an optimal policy is used thereafter. Control decisions can be made according to how control actions are ranked by this value model given the current state. This method is related to the "action-dependent adaptive critic" mentioned by Werbos (1989) and to the classifier systems described by Holland (1986).

When payoff values and control actions are continuous quantities, a value model can be constructed in a form that permits the computation of the gradient of the estimated value with respect to control variables. The policy can then be adjusted via gradient ascent. Using an artificial neural network to represent the value model makes this approach attractive because the value model's gradient can be computed efficiently by error back-propagation. This approach was discussed by Werbos (1977) in relation to the differential dynamic programming method of Jacobson and Mayne (1970), and Jordan and Jacobs (1990) illustrated it using a version of the pole-balancing task with continuous control actions.

Reinforcement learning methods have also been studied that use both state-transition and value models (e.g., El-Fattah, 1981; Sutton, 1990). Werbos (1987, 1988, 1989) discusses gradient methods that make use of system models.

5 AN INDIRECT ALGORITHM

We describe the algorithm proposed by Sato, Abe, and Takeda (1988) as an example of an indirect method for learning to solve Markov decision problems with unknown transition probabilities. This algorithm is an extension of previous research by the same authors (Sato, Abe, and Takeda, 1982, 1985). In Section 6, we combine a component of this algorithm with Q-Learning to produce a comparable direct method.

The method of Sato et al. explicitly estimates the unknown state-transition probabilities by keeping counts

[4] More generally, if the payoff process is stochastic, the expected payoff values must be estimated.

of state transitions observed while controlling the system. Let $n_{ij}^k(t)$ be the number of times action k was taken on a transition from state i to state j before time t. Then $n_i^k(t) = \sum_j n_{ij}^k(t)$ is the number of times action k was taken in state i, and $n_i(t) = \sum_k n_i^k(t)$ is the number of times state i occurred. The estimates at time t of the unknown transition probabilities, which constitute the state-transition model at time t, are $\hat{p}_{ij}^k(t) = n_{ij}^k(t)/n_i^k(t)$. Sato et al. (1988) show that if all state-transition probabilities are positive, then in the limit these estimates converge, almost surely, to the actual transition probabilities. They assume that the payoff array is known.

At each time t, an estimated optimal policy, $\hat{U}^*(t)$, is computed using the policy iteration method of DP based on the current state-transition model and the payoff array. The control action specified by this policy for the current state i, $\hat{u}_i^*$, is used to bias the control decision in favor of the estimated optimal action in a manner described below. Because the state-transition model only changes by a small amount at each time step, the policy iteration method converges after few iterations if it starts with the estimated optimal policy computed on the previous time step.

An explicit mechanism is used to cause sufficient exploratory behavior for the system identification process to converge to the correct state-transition model. This mechanism works by sometimes forcing the controller to take an action that has not been taken for a long time instead of the action currently estimated to be optimal. This explicit tradeoff between estimation and control is implemented in the following way. At each time step t, a quantity, $c_i^k(t)$, is maintained for each state i and action k to reflect the number of times action k has *not been performed* in state i since $t = 0$. These quantities are computed iteratively by letting $c_i^k(0) = 0$ for all i and k and using the following update rule at each time step:

$$c_i^k(t+1) = \begin{cases} c_i^k(t) + \Theta(n_i(t+1) - n_i^k(t+1)) & \text{if } k \neq u(t) \\ c_i^k(t) \quad \text{otherwise,} \end{cases}$$

where $u(t)$ is the action performed at time t. Θ is a positive function that is constant or satisfies the conditions

$$\lim_{n\to\infty} \Theta(n) = 0, \quad \text{and} \quad \sum_{n=1}^{\infty} \Theta(n) = \infty. \qquad (2)$$

The values $c_i^k(t)$ are used to determine the controller's action at time t as follows. If i is the state at time t and $\hat{u}_i^*$ is the action estimated (via policy iteration) to be optimal for state i, then the action actually performed by the controller is the action k which maximizes

$$\begin{array}{ll} c_i^k(t)/n_i^k(t) + \alpha & \text{if } k = \hat{u}_i^* \\ c_i^k(t)/n_i^k(t) & \text{otherwise,} \end{array} \qquad (3)$$

where α is a positive constant. The fractions in (3) cause the controller to sometimes prefer over $\hat{u}_i^*$ an action that has not been been performed for a long time.

Sato et al. (1988) show that if Θ satisfies the conditions given by (2), then in the limit all actions are performed infinitely often for each state, as needed for convergence of the state-transition model, and that the policy converges to an optimal policy. Specifically, they define the *relative frequency coefficient* to be

$$f^*(t) \;=\; \frac{1}{t-1}\sum_i n_i^{u_i^*}(t), \qquad (4)$$

which gives the average number of optimal decisions made before time t. Sato et al. (1988) prove that if all transition probabilities are positive and $\Theta(n) = \Theta_0$ for all n, then

$$\lim_{\Theta_0\to 0}\lim_{t\to\infty} f^*(t) = 1, \quad \text{almost surely,}$$

whereas if $\Theta(n)$ satisfies (2), then

$$\lim_{t\to\infty} f^*(t) = 1, \quad \text{almost surely.}$$

6 Q–LEARNING

Q–Learning is a method proposed by Watkins (1989) that can form the basis of a variety of direct reinforcement learning methods. It is an asynchronous Monte Carlo form of DP that does not require knowledge of the state-transition probabilities or the payoff array. Q–Learning estimates what Watkins calls *state-action values*: the state-action value of state i and action k, denoted Q_{ik}, is the expected infinite-horizon discounted return if action k is performed in initial state i and an optimal policy is followed thereafter. An optimal action for state i is therefore any action k that maximizes Q_{ik}.

Each time the controller takes an action, say action k from state i at time t, the current state-action value estimate for i and k, denoted $\hat{Q}_{ik}(t)$, is updated as follows:

$$\hat{Q}_{ik}(t+1) = (1-\beta_t)\hat{Q}_{ik}(t) + \beta_t[r_{ij}^k + \gamma \max_{l\in A} \hat{Q}_{jl}(t)], \quad (5)$$

where j is the actual next state, γ is the discount factor, and $\{\beta_t\}$ is a sequence of step-size parameters. The state-action value estimates for states other than i and actions other than k remain unchanged. Watkins (1989) shows that these estimates converge to the true state-action values if each action is eventually performed infinitely often from each state and the sequence $\{\beta_t\}$ converges to zero in an appropriate manner.

Given estimated state-action values $\hat{Q}_{ik}(t)$, the policy that is optimal with respect to these estimates (i.e., a kind of certainty equivalence policy) is the policy that selects for each state i the action

$$\hat{u}_i^* = \arg\max_{k \in A} \hat{Q}_{ik}(t), \tag{6}$$

where ties among actions are resolved in some arbitrary way.

If the estimation error of the state-action values is zero, then the policy specifying $\hat{u}_i^*$ for each state i as defined by (6) is an optimal policy, but Q-Learning does not require this policy to be followed during learning. For performing Q-Learning, the policy actually followed by the controller is not important except that it must allow sufficient exploration to permit convergence of the state-action value estimates. However, for the controller to improve its performance while performing Q-Learning, it must bias its policy toward the estimated optimal actions. Because this must be accomplished while permitting sufficient exploration, the issues that arise are identical to those considered by Sato et al. (1988), and it is possible to combine Q-Learning with their method for determining control choices to produce a new direct reinforcement learning algorithm.

This algorithm, which we call the Exploratory Q-Learning, or EQ, algorithm, combines the exploration strategy of Sato et al. with Q-Learning. Instead of using policy interation at each step to estimate the current optimal action, the EQ algorithm uses the action, $\hat{u}_i^*$, computed from the current state-action value estimates according to (6). This action is then used in (3) to determine the control decision, where $c_i^k(t)$ and $n_i^k(t)$ are computed exactly as in the method of Sato et al. Using Q-Learning instead of policy interation leads to great savings in both the space and time complexity of each control step (detailed below). Although it is possible to reduce the space complexity further by replacing the exploration method of Sato et al. with one having less demanding space requirements, we retain their method to facilitate comparison of the indirect and direct aspects of the algorithms.

7 SIMULATION RESULTS

Sato et al. (1988) present simulation results for their method applied to a simple five state, three action, Markov decision problem. The arrays of transition probabilities and payoffs are reproduced here in Tables 1 and 2. Recognizing that this problem is too small to allow strong conclusions to be drawn, and that it was used by Sato et al. merely to illustrate their convergence result, we compared the performances of the Sato et al. and EQ methods on this problem to obtain a preliminary indication of the relative efficiency of directly comparable indirect and direct and learning methods.

Figure 1 shows the evolution of the relative frequency coefficient, $f^*(t)$, defined by (4), as a function of the number control actions for both learning methods and

Table 1: Transition Probabilities

$p_{11}^1 = 0.1$	$p_{12}^1 = 0.2$	$p_{13}^1 = 0.2$	$p_{14}^1 = 0.2$	$p_{15}^1 = 0.3$
$p_{11}^2 = 0.2$	$p_{12}^2 = 0.2$	$p_{13}^2 = 0.2$	$p_{14}^2 = 0.2$	$p_{15}^2 = 0.2$
$p_{11}^3 = 0.1$	$p_{12}^3 = 0.1$	$p_{13}^3 = 0.1$	$p_{14}^3 = 0.3$	$p_{15}^3 = 0.4$
$p_{21}^1 = 0.1$	$p_{22}^1 = 0.1$	$p_{23}^1 = 0.2$	$p_{24}^1 = 0.3$	$p_{25}^1 = 0.3$
$p_{21}^2 = 0.1$	$p_{22}^2 = 0.1$	$p_{23}^2 = 0.6$	$p_{24}^2 = 0.1$	$p_{25}^2 = 0.1$
$p_{21}^3 = 0.1$	$p_{22}^3 = 0.5$	$p_{23}^3 = 0.2$	$p_{24}^3 = 0.1$	$p_{25}^3 = 0.1$
$p_{31}^1 = 0.1$	$p_{32}^1 = 0.4$	$p_{33}^1 = 0.2$	$p_{34}^1 = 0.2$	$p_{35}^1 = 0.1$
$p_{31}^2 = 0.3$	$p_{32}^2 = 0.2$	$p_{33}^2 = 0.2$	$p_{34}^2 = 0.2$	$p_{35}^3 = 0.1$
$p_{31}^3 = 0.3$	$p_{32}^3 = 0.2$	$p_{33}^3 = 0.1$	$p_{34}^3 = 0.1$	$p_{35}^3 = 0.3$
$p_{41}^1 = 0.2$	$p_{42}^1 = 0.5$	$p_{43}^1 = 0.1$	$p_{44}^1 = 0.1$	$p_{45}^1 = 0.1$
$p_{41}^2 = 0.3$	$p_{42}^2 = 0.1$	$p_{43}^2 = 0.1$	$p_{44}^2 = 0.4$	$p_{45}^2 = 0.1$
$p_{41}^3 = 0.2$	$p_{42}^3 = 0.3$	$p_{43}^3 = 0.3$	$p_{44}^3 = 0.1$	$p_{45}^3 = 0.1$
$p_{51}^1 = 0.2$	$p_{52}^1 = 0.2$	$p_{53}^1 = 0.2$	$p_{54}^1 = 0.2$	$p_{55}^1 = 0.2$
$p_{51}^2 = 0.2$	$p_{52}^2 = 0.3$	$p_{53}^2 = 0.2$	$p_{54}^2 = 0.1$	$p_{55}^2 = 0.2$
$p_{51}^3 = 0.1$	$p_{52}^3 = 0.4$	$p_{53}^3 = 0.2$	$p_{54}^3 = 0.1$	$p_{55}^3 = 0.2.$

Table 2: Payoffs

$r_{11}^1 = 1$	$r_{12}^1 = -2$	$r_{13}^1 = -4$	$r_{14}^1 = -1$	$r_{15}^1 = 3$
$r_{11}^2 = -2$	$r_{12}^2 = -7$	$r_{13}^2 = 2$	$r_{14}^2 = 8$	$r_{15}^2 = -1$
$r_{11}^3 = -3$	$r_{12}^3 = 6$	$r_{13}^3 = -1$	$r_{14}^3 = -2$	$r_{15}^3 = 4$
$r_{21}^1 = 7$	$r_{22}^1 = -4$	$r_{23}^1 = 1$	$r_{24}^1 = -2$	$r_{25}^1 = -8$
$r_{21}^2 = -5$	$r_{22}^2 = -2$	$r_{23}^2 = 1$	$r_{24}^2 = -2$	$r_{25}^2 = 5$
$r_{21}^3 = 5$	$r_{22}^3 = -1$	$r_{23}^3 = -3$	$r_{24}^3 = -7$	$r_{25}^3 = 0$
$r_{31}^1 = 3$	$r_{32}^1 = 1$	$r_{33}^1 = 2$	$r_{34}^1 = 4$	$r_{35}^1 = -4$
$r_{31}^2 = 3$	$r_{32}^2 = 0$	$r_{33}^2 = -1$	$r_{34}^2 = -3$	$r_{35}^3 = 7$
$r_{31}^3 = -6$	$r_{32}^3 = 1$	$r_{33}^3 = -2$	$r_{34}^3 = 4$	$r_{35}^3 = 0$
$r_{41}^1 = 5$	$r_{42}^1 = -4$	$r_{43}^1 = 3$	$r_{44}^1 = 1$	$r_{45}^1 = -6$
$r_{41}^2 = 3$	$r_{42}^2 = 2$	$r_{43}^2 = -1$	$r_{44}^2 = -3$	$r_{45}^2 = -5$
$r_{41}^3 = 4$	$r_{42}^3 = 1$	$r_{43}^3 = -6$	$r_{44}^3 = 6$	$r_{45}^3 = 2$
$r_{51}^1 = 2$	$r_{52}^1 = 6$	$r_{53}^1 = 2$	$r_{54}^1 = -1$	$r_{55}^1 = 3$
$r_{51}^2 = -5$	$r_{52}^2 = 1$	$r_{53}^2 = -3$	$r_{54}^2 = 4$	$r_{55}^2 = -4$
$r_{51}^3 = -3$	$r_{52}^3 = 5$	$r_{53}^3 = 2$	$r_{54}^3 = -1$	$r_{55}^3 = -5,$

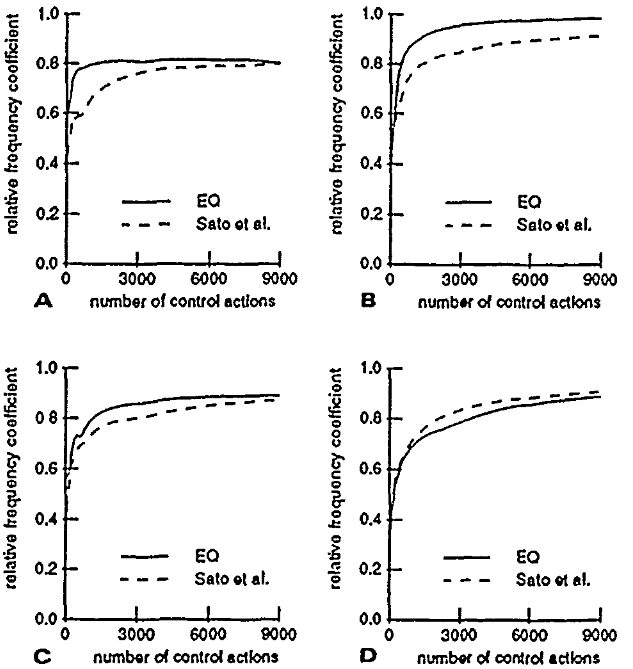


Figure 1: Graphs of the relative frequency coefficient, $f^*(t)$, as a function of the number of control actions performed for the algorithm of Sato et al. (dashed line) and the EQ algorithm (solid line) for four choices for $\Theta(n)$. Each graph is the average of five simulation experiments made with different random number seeds. For all graphs, $\alpha = 1.0$, $\beta = 0.05$, and $\gamma = 0.8$. Panel A: $\Theta(n) = 0.1$, Panel B: $\Theta(n) = 0.05$, Panel C: $\Theta(n) = 1/n$, Panel D: $\Theta(n) = 1/\sqrt{n}$.

for the four choices of the function Θ (indicated in the figure caption) used by Sato et al. (1988). Each graph is the average of five simulation experiments made with different random number seeds. In all cases, the action-selection strategies of the two methods were parameterized identically, so that the only difference between the methods was the manner in which the estimated optimal action, $\hat{u}_t^*$, was computed at each time step The sequence $\{\beta_t\}$ for the Q-Learning algorithm was held constant at 0.05 throughout the simulations, a value not explicitly optimized for this problem.

With the exception of the graphs in Panel D, the graphs in Fig. 1 show that, in this learning task with the indicated parameter values, the EQ algorithm achieves a higher level of performance after any given number of control actions than does the algorithm of Sato et al. Panel D shows somewhat better performance for the method of Sato et al. Note that the EQ algorithm achieves this performance level using only the actual payoff at each time step instead of knowledge of the entire payoff array required by the algorithm of Sato et al.

Not shown in the figure is the relative amount of computation per control action for the two learning methods. Because the method of Sato et al. performs policy iteration after taking each action, whereas the EQ method performs a single Q-Learning step, the EQ method requires much less computation per step. Although policy interation can be approximated without explicit matrix inversion at each iteration (e.g., Riordon, 1969), we assume that each application of policy iteration requires at least one matrix inversion. Assuming that any practical matrix inversion algorithm requires $O(N^3)$ operations for an $N \times N$ matrix, the time taken by policy iteration is $O(N^3+N^2K)$, where N is the number of states and K is the number of actions. The time required for a Q-Learning step (i.e., to apply (5))is just $O(K)$. Hence, the savings for each control action using the EQ method is dramatic. For example, for each action performed in the test problem, the Sato et al. method requires a minimum of about 200 basic computational steps, whereas the EQ method requires essentially 3, the number of actions (not counting the few computations required by each method to implement their common action-selection process).

Additionally, the EQ method is more space efficient than the method of Sato et al.: The latter method requires $O(KN^2)$ storage locations because it has to store the state-transition model and the payoff array, whereas Q-Learning requires $O(KN)$ storage locations for the state-action value estimates. In fact, most of the space used by the EQ method is used to implement the action-selection process it shares with the method of Sato et al. Preliminary simulations using Q-Learning with less complex action-selection processes have produced performance better than that of the EQ method on this problem.

8 DISCUSSION

We were initially surprised by the results shown in Figure 1. Even for the small test problem, we expected the simplicity of the EQ algorithm on a per-control-action basis to extract a higher price in terms of the number of control actions required for achieving a given level of performance. Ignoring the per-control-action cost, how can *any* method perform better than one that performs complete DP at each control step? The answer lies in the consequences that each learning method has for the exploratory behavior of the controller. Both algorithms use the same mechanism for selecting actions on the basis of the current estimate for the optimal action ($\hat{u}_t^*$), but differences in these estimates imply the selection of different actions. Because each Q-Learning step depends on a very small sample from a random process, the behavior produced by the EQ algorithm is more variable than that produced by the algorithm of Sato et al. in the initial stages of learning. This variability seems to produce more effective exploration for the test problem in question, consistent with

Witten's (1977b) observations on exploration in discrete deterministic environments. Under conditions of high uncertainty, therefore, it might be better to avoid complex long-term planning not only to save fruitless computational effort, but also to foster more effective exploratory behavior.

Clearly, as control problems become more difficult due to increases in the number of states and control actions, and increases in "depth" (i.e., increases in the degree to which the long-term consequences of control decisions influence performance), one would expect increases in the utility of performing conventional DP based on a state-transition model. But because the computational cost of this approach increases rapidly as problems become larger and/or deeper, the straightforward extension of such an indirect method to more difficult problems is not necessarily the best approach. As problems become more difficult, the effectiveness of various methods, and combinations of methods, will depend on details of the problems and the conditions under which they must be solved, i.e, on a wide set of issues making up the economic context of the computation. For example, in applying the EQ algorithm and the algorithm of Sato et al. to several problems larger than the test problem described here (problems with 7 and 8 states), sometimes one algorithm and then the other would perform better. We could discern no clear relationship between the task and which algorithm would reach a higher level of performance after a given number of control actions, except that in all cases the EQ algorithm required much less overall computation due to the small number of computational steps it required per control action.

Experience does indicate, however, that neither the indirect nor the direct methods described in this paper efficiently scale up to large nonlinear problems without additional mechanisms. It seems clear that many types of models must be employed in a variety of different ways to achieve effective learning performance on complex tasks. Hence, we emphatically do not interpret the results reported here as suggesting that state-transition models should be *replaced* by value models. These results do raise questions about the most commonly studied methods for using state-transition models in learning to solve Markov decision problems, but when scaling issues are considered, they suggest that combinations of direct and indirect methods may be most useful. Given a fixed amount of computational power available per control action, it may be better to use a direct reinforcement learning method *augmented* with indirect techniques than to devote all available resources to a computationally costly indirect method. One way of combining direct and indirect methods that retains many of the advantages of each approach is illustrated by Sutton's DYNA architecture (Sutton, 1990).

9 CONCLUSION

The simulation results described in this paper show that although the direct EQ algorithm requires less space and much less computation per control action than the indirect method of Sato et al., its learning ability when applied to a test problem is superior to, or compares favorably with, that of the more complex indirect method. Using a certainty equivalence approach, indirect methods for learning to solve Markov decision problems perform costly "pseudo-optimization" on the basis of uncertain information. Direct reinforcement learning methods, on the other hand, keep closer touch to reality by directly using experience with the system itself instead of with a system model.

However, because the comparative study presented in this paper involves only a single very small Markov decision problem and a single pair of learning algorithms, the results merely provide one data point in the study of the relative advantages of direct and indirect learning methods. Although we know how the relative number of computations *per control action* increases with increasing problem size, we do not know what happens to the relative performance of these methods as the task size increases. The utility of performing conventional DP based on a state-transition model surely increases with increasing problem size and difficulty, but is it worth the greatly increasing computational cost?

A comprehensive answer to this question depends on many factors making up the economic context of the computation, but our results suggest that it can be advantageous to distribute the required learning and planning processes over system states, control actions, and time in ways differing from that of conventional indirect learning methods. The theory of reinforcement learning using incremental dynamic programming methods needs to be extended with these issues in mind.

Acknowledgements

This paper is based on material which appeared in Barto and Singh (1990). This research was supported by the Air Force Office of Scientific Research, Bolling AFB, under Grant AFOSR-89-0526, and by the National Science Foundation under Grant ECS-8912623.

References

C. W. Anderson. Strategy learning with multilayer connectionist representations. Technical report TR87-509.3, GTE Laboratories, Incorporated, Waltham, MA, 1987. (This is a corrected version of the report published in *Proceedings of the Fourth International Workshop on Machine Learning*,103–114, 1987, San Mateo, CA: Morgan Kaufmann.)

A. G. Barto and P. Anandan. Pattern recognizing stochastic learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 15.360-375, 1985.

A. G. Barto and S. P. Singh. Reinforcement learning and dynamic programming. In *Proceedings of the Sixth Yale Workshop on Adaptive and Learning Systems*, New Haven, CT, Aug 1990.

A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835-846, 1983. Reprinted in J. A. Anderson and E. Rosenfeld, *Neurocomputing: Foundations of Research*, MIT Press, Cambridge, MA, 1988.

A. G. Barto, R. S. Sutton, and P. S. Brouwer. Associative search network: A reinforcement learning associative memory. *IEEE Transactions on Systems, Man, and Cybernetics*, 40:201-211, 1981.

A. G. Barto, R. S. Sutton, and C. Watkins. Learning and sequential decision making. In M. Gabriel and J. W. Moore, editors, *Learning and Computational Neuroscience.* MIT Press, Cambridge, MA. To appear.

A. G. Barto, R. S. Sutton, and C. Watkins. Sequential decision problems and neural networks. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, San Mateo, CA, 1990. Morgan Kaufmann.

D. P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ, 1987.

V. Borkar and P. Varaiya. Adaptive control of markov chains I: Finite parameter set. *IEEE Transactions on Automatic Control*, 24:953-957, 1979.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley, New York, 1973.

Y El-Fattah. Recursive algorithms for adaptive control of finite markov chains. *IEEE Transactions on Systems, Man, and Cybernetics*, 11:135-144, 1981.

G C. Goodwin and K. S. Sin. *Adaptive Filtering Prediction and Control.* Prentice-Hall, Englewood Cliffs, N.J., 1984.

S. E. Hampson. *Connectionist Problem Solving. Computational Aspects of Biological Learning.* Birkhauser, Boston, 1989.

J. H. Holland. Escaping brittleness. The possibility of general-purpose learning algorithms applied to rule-based systems. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach, Volume II*, pages 593-623. Morgan Kaufmann, San Mateo, CA, 1986.

D. H. Jacobson and D. Q. Mayne. *Differential Dynamic Programming.* Elsevier, New York, 1970.

M. I. Jordan and R. A. Jacobs. Learning to control an unstable system with forward modeling. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 2*, San Mateo, CA, 1990. Morgan Kaufmann.

A. H. Klopf. Brain function and adaptive systems—A heterostatic theory. Technical Report AFCRL-72-0164, Air Force Cambridge Research Laboratories, Bedford, MA, 1972. (A summary appears in *Proceedings of the International Conference on Systems, Man, and Cybernetics*, 1974, IEEE Systems, Man, and Cybernetics Society, Dallas, TX.)

A. H. Klopf. *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence.* Hemishere, Washington, D.C., 1982.

P. R. Kumar and Woei Lin. Optimal adaptive controllers for unknown markov chains. *IEEE Transactions on Automatic Control*, 25:765-774, 1982.

S. Lakshmivarahan. *Learning Algorithms and Applications.* Springer-Verlag, New York, 1981.

P. Mandl. Estimation and control in markov chains. *Advances in Applied Probability*, 6:40-60, 1974.

K. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction.* Prentice Hall, Englewood Cliffs, NJ, 1989.

J. S. Riordon. An adaptive automaton controller for discrete-time markov processes. *Automatica*, 5:721-730, 1969.

M. Sato, K. Abe, and H. Takeda. Learning control of finite markov chains with unknown transition probabilities. *IEEE Transactions on Automatic Control*, 27:502-505, 1982.

M. Sato, K. Abe, and H. Takeda. An asymptotically optimal learning controller for finite markov chains with unknown transition probabilities. *IEEE Transactions on Automatic Control*, 30.1147-1149, 1985.

M. Sato, K. Abe, and H. Takeda. Learning control of finite markov chains with explicit trade-off between estimation and control. *IEEE Transactions on Systems, Man, and Cybernetics*, 18:677-684, 1988.

R. S. Sutton. *Temporal Credit Assignment in Reinforcement Learning.* PhD thesis, University of Massachusetts, Amherst, MA, 1984.

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3.9-44, 1988.

R. S. Sutton. Integrating architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 216-224, San Mateo, CA, 1990. Morgan Kaufmann.

C. J. C. H. Watkins. *Learning from Delayed Rewards.*

PhD thesis, Cambridge University, Cambridge, England, 1989.

P. J. Werbos. Advanced forecasting methods for global crisis warning and models of intelligence. *General Systems Yearbook*, 22:25–38, 1977.

P. J. Werbos. Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man, and Cybernetics*, 1987.

P. J. Werbos. Generalization of back propagation with applications to a recurrent gas market model. *Neural Networks*, 1:339–356, 1988.

P. J. Werbos. Neural networks for control and system identification. In *Proceedings of the 28th Conference on Decision and Control*, pages 260–265, Tampa, Florida, 1989.

R. M. Wheeler and K. S. Narendra. Decentralized learning in finite markov chains. *IEEE Transactions on Automatic Control*, 31:519–526, 1986.

I. H. Witten. An adaptive optimal controller for discrete-time Markov environments. *Information and Control*, 34:286–295, 1977a.

I. H. Witten. Exploring, modelling and controlling discrete sequential environments. *International Journal of Man-Machine Studies*, 9:715–735, 1977b.

Reinforcement Comparison

Peter Dayan
Centre for Cognitive Science &
Department of Physics
University of Edinburgh
2 Buccleuch Place
Edinburgh EH8 3LW
Scotland

Abstract

Sutton [2] introduced a reinforcement comparison term into the equations governing certain stochastic learning automata, arguing that it should speed up learning, particularly for unbalanced reinforcement tasks. Williams' subsequent extensions [3] to the class of algorithms demonstrated that they were all performing approximate stochastic gradient ascent, but that, in terms of expectations, the comparison term has no first order effect.

This paper analyses the second order contribution, and uses the criterion that its modulus should be minimised to determine an optimal value for the comparison term. This value turns out to be different from the one Sutton used, and simulations suggest at its efficacy.

1 INTRODUCTION

Sutton [2] introduced the notion of reinforcement prediction as a way of speeding up the learning of a class of stochastic learning automata. Most previous methods made assumptions about the independence of the learning of the automata from all aspects of their reinforcement history that were not 'compiled' into their current action probabilities. Sutton reasoned that comparing the current reinforcement with some function of its frequency of delivery in the past might be helpful for determining whether or not their actions were making things worse or better. He expected particular utility for such comparisons in the difficult cases in which reinforcement delivery is unbalanced - for instance when all actions tend to be rewarded or punished.

Williams [3] analysed a related set of algorithms, which includes Sutton's, and demonstrated that they were all performing on-line, stochastic gradient ascent in the expected amount of reinforcement. This is reassuring, since it implies that the algorithms are moving in the correct direction, statistically at least. The surprising part of his analysis was that, for the particular case Sutton considered, the comparison term may be eliminated from the analysis at an early stage. The result on stochastic gradient ascent is unaffected by its value. Sutton's simulations, however, demonstrated that different comparison terms perform very differently.

Williams essentially looked at the first order term in the Taylor expansion of the function that relates the expected reinforcement to the weights determining the probability of performing the actions. Although the comparison term vanishes from this, it would not be expected to vanish from the second and higher order terms. Second order analysis should reveal for it both a rôle, and, potentially, an optimal value.

2 THEORY

2.1 WILLIAMS' ANALYSIS

Williams treats a very general problem. At any time, each of n units receives an input $\mathbf{x}^i \in \Re^p, 1 \leq i \leq n$ from some environment, and uses its weight vector $\mathbf{w}^i \in \Re^p$ to determine whether to fire or not, $y_i = 1$ or $y_i = 0$ respectively. Before it chooses its action, and before the environment evaluates the combined set of actions, each unit also chooses a reinforcement comparison value $b_{ij}, 1 \leq i \leq n, 1 \leq j \leq p$ for each component of each weight. The environment returns a global reinforcement value r that is stochastically related to the quality of the actions of the units, and each unit then updates its weight vector according to the reinforcement, its chosen action, and its reinforcement comparison values.

A simple example of such a reinforcement learning system is the two armed bandit problem, which we shall return to later. For this, the automaton has no inputs, but chooses, stochastically on the basis of a stored weight, to pull either the left arm ($y = 0$) of the bandit or the right arm ($y = 1$). The machine delivers reinforcement of $r = \pm 1$ with different probabilities for the two arms, and the automaton has to learn, by changing the weight, which arm it is best to pull.

More formally, Williams proves that if:

$$\Delta w_{ij} = \alpha_{ij}(r - b_{ij})e_{ij}, \qquad (1)$$

where,

- r is the reinforcement,
- b_{ij} are reinforcement baselines, which are conditionally independent of the actions y_i given the weights $\mathbf{W}$ and the inputs $\mathbf{x}^i$,
- α_{ij} is the learning rate parameter for w_{ij},
- $e_{ij} = \dfrac{\delta \ln g_i}{\delta w_{ij}}$ is the so-called eligibility of the weight w_{ij}, a measure of how influential it was in choosing the action,
- $g_i(\xi, \mathbf{w}^i, \mathbf{x}^i) = \mathcal{P}[y_i = \xi | \mathbf{w}^i, \mathbf{x}^i]$ is the probability the i^{th} unit emits action ξ given its weights $\mathbf{w}^i$ and its input $\mathbf{x}^i$.

then:

$$\mathcal{E}[\Delta w_{ij}|\mathbf{W}] = \alpha_{ij}\frac{\delta\mathcal{E}[r|\mathbf{W}]}{\delta w_{ij}}. \qquad (2)$$

where $\mathbf{W}$ is the matrix of all $\mathbf{w}^i$.

Equation 2 implies that these algorithms are all performing stochastic gradient ascent in an averaged sense. The dependence on the values of the b_{ij} drops out at an early stage, since:

$$\mathcal{E}[e_{ij}|\mathbf{W}, \mathbf{x}^i] = 0.$$

However, looking at equation 1, it is apparent that changing the b_{ij} is likely to affect at very least the stability of the algorithm. Sutton [2] investigated this empirically for his algorithm, and found faster convergence across a range of problems for b_{ij} being estimators of the average amount of reinforcement received than for $b_{ij} = 0$.

2.2 THE SECOND ORDER TERM

Unfortunately, treating higher order terms at the same level of generality as Williams is not fruitful. Consider instead the simplest of the cases that Sutton takes. Here there is just one unit, weights w_i, inputs x_i, reinforcement r, and with:

$$\Delta w_i = \alpha(r - b)(y - \pi)x_i$$

where $\pi = \mathcal{E}[y|\mathbf{x}, \mathbf{w}]$. b can depend on $\mathbf{x}$ and $\mathbf{w}$, but not on the output y.

A different way of looking at Williams' result is through the Taylor expansion of $\mathcal{E}[r'|\mathbf{w}, \mathbf{x}]$, using the prime $'$ to indicate that it is the expected value of the reinforcement that will be received at the next time step:

$$\begin{aligned}&\mathcal{E}[r'|\mathbf{w}, \mathbf{x}] = \\ &\quad\mathcal{E}[r|\mathbf{w}] + \sum \mathcal{E}[\Delta w_i|\mathbf{w}, \mathbf{x}]\frac{\delta\mathcal{E}[r|\mathbf{w}]}{\delta w_i} + \\ &\quad\tfrac{1}{2}\sum \mathcal{E}[\Delta w_i \Delta w_j|\mathbf{w}, \mathbf{x}]\frac{\delta^2\mathcal{E}[r|\mathbf{w}]}{\delta w_i \delta w_j} + \ldots\end{aligned}$$

Williams deals with the first order term, showing that the first term in the product is proportional to the second, and that b makes no contribution whatsoever. Setting $\mathcal{F}(\mathbf{z}) = \mathcal{E}[r|\mathbf{z}]$, the second order term is:

$$\begin{aligned}&\sum_{i,j}\frac{\delta^2\mathcal{F}}{\delta w_i \delta w_j}x_i x_j \alpha^2 \times \\ &\quad\sum_{\xi,\rho}\mathcal{P}[y = \xi|\mathbf{w}, \mathbf{x}]\mathcal{P}[r = \rho|\xi, \mathbf{x}] \times \\ &\qquad(\rho - b)^2(\xi - \pi)^2. \qquad (3)\end{aligned}$$

Note that the inner sum does not depend on the value of i or j. Extracting the value $\hat{b}$ that minimises it, gives:

$$\hat{b} = \frac{\mathcal{E}[r(y - \pi)^2|\mathbf{w}, \mathbf{x}]}{\mathcal{V}[y|\mathbf{w}, \mathbf{x}]}$$

However, y can only take on two values: 0 or 1. Let:

$$\begin{aligned}p &= \mathcal{P}[y = 1|\mathbf{w}, \mathbf{x}]\\ r_0 &= \mathcal{E}[r|y = 0, \mathbf{w}, \mathbf{x}]\\ r_1 &= \mathcal{E}[r|y = 1, \mathbf{w}, \mathbf{x}]\end{aligned}$$

then $\pi = p$, and

$$\begin{aligned}\hat{b} &= \frac{p(1-p)^2 r_1 + (1-p)p^2 r_0}{p(1-p)}\\ &= (1-p)r_1 + p r_0.\end{aligned}$$

in which, counterintuitively, the expected reward for emitting action 1 is paired with the probability of emitting action 0, and *vice-versa*. Williams (personal communication) derived the same expression for $\hat{b}$ on the grounds of minimising the variance of the Δw_i.[1] He ultimately considered this an inappropriate reason for choosing the value of b.

The reinforcement comparison algorithm favoured by Sutton involves teaching an extra unit to predict the future reinforcement level. He defines $s = \sum_i v_i x_i$, where $\mathbf{v}$ are the prediction weights. These are changed according to:

$$\Delta v_i = \beta(r - s)x_i.$$

This tends to make s an estimator of sorts of $\mathcal{E}[r|\mathbf{w}, \mathbf{x}]$, or b', where:

$$b' = p r_1 + (1-p)r_0.$$

which, *a priori*, is the more natural pairing.

[1] Minimising the variance leads to the same expression since $\mathcal{E}[\Delta w_i]$ is independent of b, and x_i factors out.

2.3 CHOOSING b

Although $\hat{b}$ minimises the inner sum in the second order term of equation 3, it is not yet clear that this is appropriate. In the one dimensional case, since the reinforcement is bounded above and below, the second derivative $\delta^2\mathcal{F}/\delta w^2$ will be positive for some values of w and negative for others. This means that it is bound both to speed and hinder the learning. Setting $b = \hat{b}$ minimises this effect.

As an example, consider the first task Sutton investigated, which is a two-armed bandit problem. Here, there are two possible actions $y = 0, 1$, and:

$$\begin{array}{lll} y = 0 \Rightarrow & \mathcal{P}[r = 1] = 0.8 & \mathcal{P}[r = -1] = 0.2 \\ y = 1 \Rightarrow & \mathcal{P}[r = 1] = 0.9 & \mathcal{P}[r = -1] = 0.1 \end{array}$$

so the optimal action is $y = 1$.

Choose $\mathcal{P}[y = 1|w] \equiv f(w) = 1/(1 + e^{-w})$, then:

$$\begin{aligned} \mathcal{E}[r|w] &= 0.6 + 0.2f(w) \\ \frac{\delta}{\delta w}\mathcal{E}[r|w] &= 0.2f(w)(1 - f(w)) \\ \frac{\delta^2}{\delta w^2}\mathcal{E}[r|w] &= 0.2f(w)(1 - f(w))(1 - 2f(w)) \end{aligned}$$

So, with $\Delta w = \alpha(r - b)(y - \pi)$, the changes are:

y	r	$\mathcal{P}$	$\Delta w/\alpha$
0	-1	$0.2(1 - f(w))$	$(1 + b)f(w)$
0	1	$0.8(1 - f(w))$	$-(1 - b)f(w)$
1	-1	$0.1f(w)$	$-(1 + b)(1 - f(w))$
1	1	$0.9f(w)$	$(1 - b)(1 - f(w))$

Then $\mathcal{E}[\Delta w] = \alpha \times 0.2f(w)(1 - f(w))$, which, as expected, is independent of b.

However, let $g(w) = \mathcal{E}[r|w] = 0.6 + 0.2f(w)$, then:

$$\begin{aligned} \mathcal{E}[\bar{r}|w] = \; & 0.2(1 - f(w))g(w + \alpha f(w)(1 + b)) + \\ & 0.8(1 - f(w))g(w - \alpha f(w)(1 - b)) + \\ & 0.1f(w)g(w - \alpha(1 - f(w))(1 + b)) + \\ & 0.9f(w)g(w + \alpha(1 - f(w))(1 - b)), \end{aligned}$$

where $\bar{r}$ is the reinforcement received after the automaton's next choice. b will not drop out of this. It is apparent from a graph of the second order term that it helps learning for $w < 0$ and hinders it for $w > 0$. Setting $b = \hat{b}$ minimises both these effects.

The same will be true in higher dimensions, i.. that the second order term will be alternately a hindrance and a help. Minimising its modulus should therefore increase the overall efficacy of gradient ascent, which operates perfectly on linear functions.

There are two types of imbalance that can afflict problems like the two-armed bandit.

- Imbalance in the probabilities - in which both the better and the worse action usually lead to the same value of reinforcement, the only different being in the precise frequency.
- Imbalance in the reinforcement values - in which the actual reinforcement values received are not centred around 0. This can make learning substantially more difficult by making the sign of the changes in the weights on any one occasion independent of the reinforcement received.

Reinforcement comparison only deals with the second of these types of imbalance. Williams (personal communication) has pointed out that the term $r - b_{ij}$ in the formula for the weight change, equation 1, will take both positive and negative values if the b_{ij} lie between the maximum and minimum reinforcement values. Barto [1] provides some reasons why the term $y - \pi$ in the learning rule helps mitigate the effects of the first type of imbalance.

3 RESULTS

Calculating the 'optimal' $\hat{b}$ is more difficult than calculating Sutton's b', because of the cross-pairing of the average reinforcement for action 1 with the probability of doing action 0. It is possible to develop an estimator $p^t(\mathbf{x}) = \sum v_i^t x$ with weights $\mathbf{v}$, as in Sutton's algorithm, and to change them according to:

$$\Delta v_i^t = \beta \left[r^t \left\{ y^t \frac{1 - \pi^t}{\pi^t} + (1 - y^t)\frac{\pi^t}{1 - \pi^t} \right\} - p^t \right] x_i.$$

where π^t is an approximation to $\mathcal{E}[y^t|\mathbf{w}^t]$. p^t then estimates $\hat{b} = (1 - p)r_1 + pr_0$. Since y is never 1 if $\pi^t = 0$, the first term is never infinite. However, this iterative scheme would not be expected to converge.

The alternative way, suggested by, but not discussed in, Sutton's thesis, is to develop separate predictions of r_0 and r_1, using two sets of weights. These would then be combined with π^t as $(1 - \pi^t)r_1 + \pi^t r_0$. Both methods were simulated.

For the sake of comparison, I used the problems that Sutton developed for his thesis [2]. The set chosen are the non-associative ones in Chapter II, although the new comparison term will work for associative tasks too. Table 1, copied from P18, shows the problems. The binary tasks produce reinforcement of ±1, with the probability that it is 1 given in the last two columns of the table. The continuous tasks produce reinforcement spread uniformly within $\pm$ 0.1 of the means given in the last two columns.

Formal descriptions of the algorithms compared are given in table 2, using Sutton's notation. Algorithms $\mathcal{A}$ and $\mathcal{A}'$ are Sutton's algorithms 8 and 9, which he found to be the best. $\mathcal{B}$, $\mathcal{B}'$, $\mathcal{C}$ and $\mathcal{C}'$ all make p^t estimate the quantity recommended by the analysis above. $\mathcal{B}$ and $\mathcal{B}'$ do this through a single term, whereas $\mathcal{C}$ and $\mathcal{C}'$ also employ u_1^t and u_2^t, which are designed to predict r_1 and r_0 respectively.

Table 1: The Tasks (From Sutton).

Task #	Reinforcement Type	r range	r mean	
			Act 1	Act 0
1	Binary	$\{1,-1\}$	0.90	0.80
2	Binary	$\{1,-1\}$	0.20	0.10
3	Binary	$\{1,-1\}$	0.55	0.45
4	Continuous	$\Re$	0.90	0.80
5	Continuous	$\Re$	-0.80	-0.90
6	Continuous	$\Re$	0.05	-0.05

Table 2: The Algorithms (After Sutton).

Algorithm	Update Rule
$\mathcal{A}$	$\Delta w[t] = \alpha(r[t+1] - p[t])(y[t] - \frac{1}{2})$
$\mathcal{A}'$	$\Delta w[t] = \alpha(r[t+1] - p[t])(y[t] - \pi[t])$
$\mathcal{B}$	$\Delta w[t] = \alpha(r[t+1] - q[t])(y[t] - \frac{1}{2})$
$\mathcal{B}'$	$\Delta w[t] = \alpha(r[t+1] - q[t])(y[t] - \pi[t])$
$\mathcal{C}$	$\Delta w[t] = \alpha(r[t+1] - s[t])(y[t] - \frac{1}{2})$
$\mathcal{C}'$	$\Delta w[t] = \alpha(r[t+1] - s[t])(y[t] - \pi[t])$

Where: $\Delta w[t] \equiv w[t+1] - w[t]$, and

$$w[0] = 0, \pi[0] = \tfrac{1}{2}, y[t] \in \{1, 0\}, \alpha > 0,$$

and $\pi[t]$ is the probability that $y[t] = 1$.

For all algorithms, $y[t] = \begin{cases} 1, & \text{if } w[t] + \eta[t] > 0; \\ 0, & \text{otherwise,} \end{cases}$

where $\eta[t]$ is normally distributed $\mathcal{N}[\mu = 0, \sigma = 0.3]$

For $\mathcal{A}$ and $\mathcal{A}'$,

$$\Delta p[t] = \beta(r[t+1] - p[t]), \qquad p[0] = r[1],$$

For $\mathcal{B}$ and $\mathcal{B}'$,

$$\Delta q[t] = \beta\left\{ r[t+1]\left(\frac{(1-y[t])\pi[t]}{1-\pi[t]} + \frac{y(t)(1-\pi[t])}{\pi[t]}\right) - q[t]\right\}, \qquad q[0] = r[1],$$

For $\mathcal{C}$ and $\mathcal{C}'$,

$$\Delta s[t] = \beta\left(u_1[t+1](1-\pi[t]) + u_0[t+1]\pi[t] - s[t]\right), \qquad s[0] = r[1],$$

$$\Delta u_1[t] = \beta(r[t+1] - u_1[t])y[t], \qquad u_1[0] = r[1],$$

$$\Delta u_0[t] = \beta(r[t+1] - u_0[t])(1 - y[t]), \; u_0[0] = r[1],$$

and $\beta = 0.2$.

All algorithms are run for 25 iterations (Sutton used 200), and each mark on the graphs in figures 1-7 is the average over 500 runs.

Figures 1-6 show how the algorithms performed on each of the various tasks, for differing values of α. Figure 7 shows how the algorithms performed across the entire range of tasks, choosing for each its best result. The y-axis shows the terminal probability of choosing action 1, which is the better action for all of the tasks. It is apparent that $\mathcal{C}$ which uses the new estimator, does indeed perform better than $\mathcal{A}$ and $\mathcal{A}'$ which use the original one, although not by much. $\mathcal{B}$ and $\mathcal{B}'$ are particularly bad on the two tasks for which reinforcement is generally negative whichever action is taken. It is unclear why this only happens for these particular tasks, although dividing by π^t or $(1 - \pi^t)$ does build in an instability. The obvious way to cure this - multiplying the rule by $\pi^t(1 - \pi^t)$ does not improve matters substantially.

In a further experiment, the standard deviation σ of the distribution of $\eta[t]$ was set to 0.5. This value determines the balance between the exploitation of the current weight $w[t]$, and the exploration for a better one. Figure 8 is the equivalent of figure 7 for this case, showing the best performance of the algorithms, and again algorithm $\mathcal{C}$ can be seen to be somewhat superior. Indeed, it affords more improvement in this case. It is also unclear why $\mathcal{C}$ should outperform $\mathcal{C}'$, since Sutton generally found algorithms with eligibility terms of the form $y - \mathcal{E}[y]$ were preferable to those employing $y - 1/2$.

A further alternative is to develop explicit estimators of $(1-p)r_1$ and pr_0, and to use their sum. In the non-associative case the resulting algorithms would not differ greatly from $\mathcal{C}$ and $\mathcal{C}'$. They would differ in the associative case, however, since the learning rule for these estimators would not change, whereas the equivalents of $\mathcal{C}$ and $\mathcal{C}'$ would involve estimators of $r_1(\mathrm{x})$ and $r_0(\mathrm{x})$, which do depend on the input x.

4 CONCLUSIONS

At least one of the ways in which reinforcement comparison works is by reducing the effects of the nonlinearity of the function which relates the weights of a stochastic learning automaton to the expected reinforcement. This is not apparent from the first order term, from which one can only conclude that the reinforcement comparison algorithms are performing stochastic gradient ascent, independent of the actual comparison adopted. The second order term also reveals an optimum value for this comparison, and simulations have confirmed that the new term speeds learning, although it does not make for a dramatic improvement.

This analysis, like Williams', says nothing about the convergence of the algorithms. However, Sutton's simulations do provide some grounds for optimism.

Acknowledgements

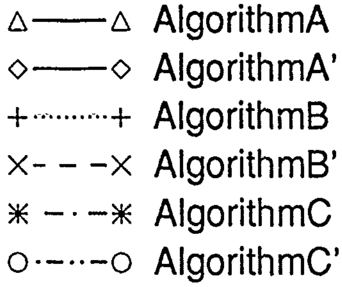
I am very grateful to Andy Barto, Geoff Hinton, Rich Sutton, Ron Williams, and David Willshaw for their helpful comments, to the students and faculty of the Summer School for the ambience, and to the SERC for their money. Part of this work was done at the University of Massachusetts at Amherst, and I particularly thank Andy Barto for his hospitality.

References

[1] Barto, AG (1985). Learning by statistical cooperation of self-interested neuron-like computing elements. *Human Neurobiology,* 4:229-256.

[2] Sutton, RS (1984). *Temporal Credit Assignment in Reinforcement Learning.* PhD Thesis. University of Massachusetts, Amherst, MA.

[3] Williams, RJ (1988). *Toward a theory of reinforcement - learning connectionist systems.* Technical Report NU-CCS-88-3, College of Computer Science, Northeastern University, 360 Huntingdon Avenue, Boston, MA.



Key for the following figures

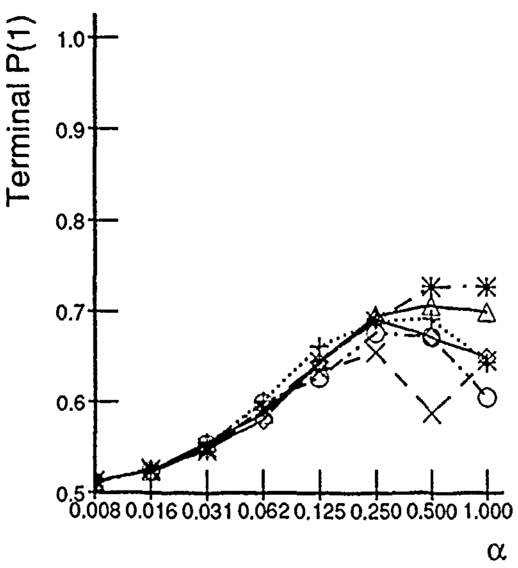


Figure 1: Task 1

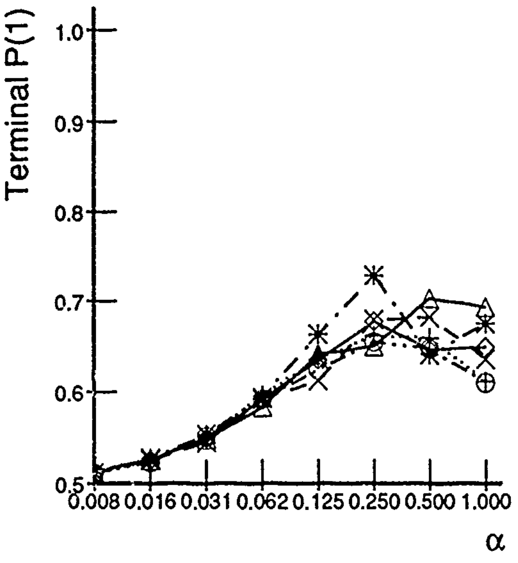


Figure 2: Task 2

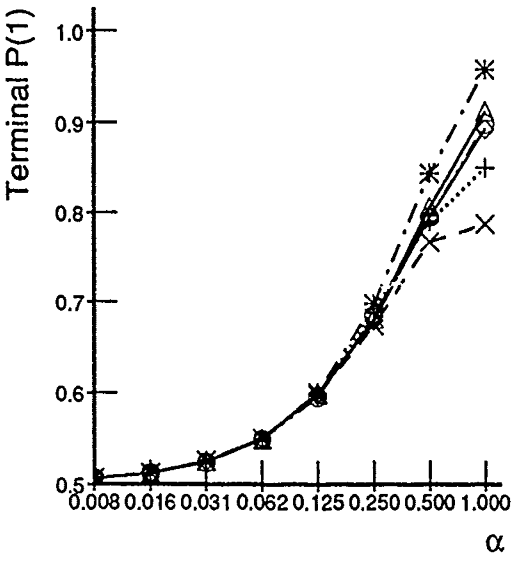


Figure 4: Task 4

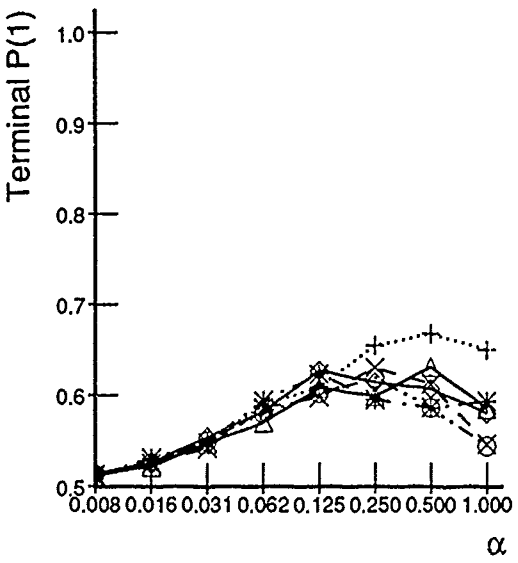


Figure 3: Task 3

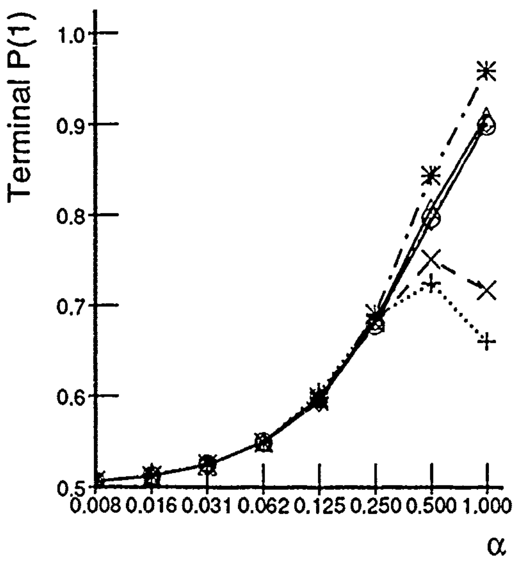


Figure 5: Task 5

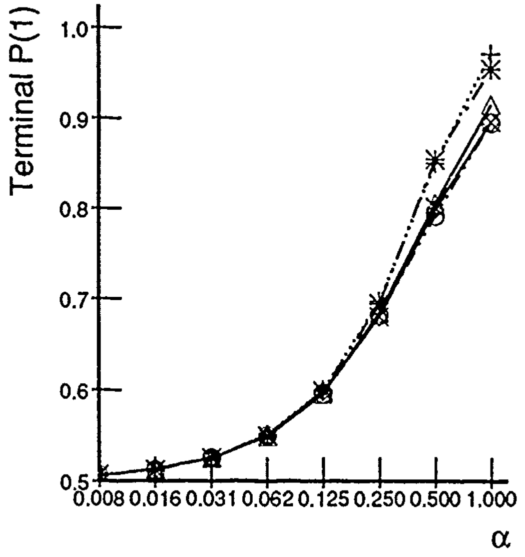


Figure 6: Task 6

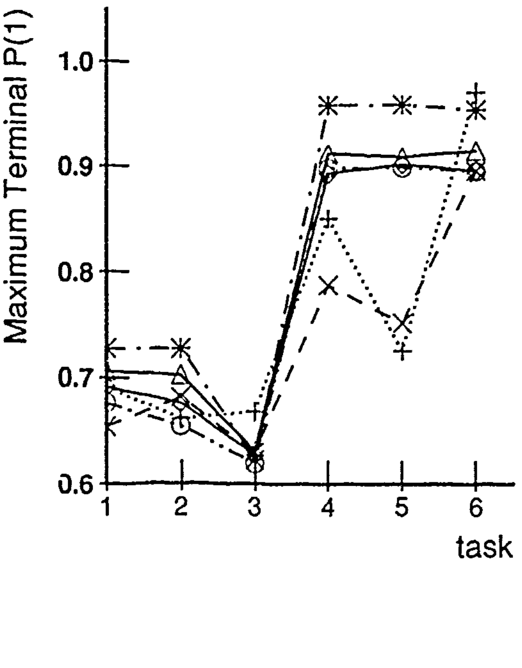


Figure 7: All tasks - best performing α

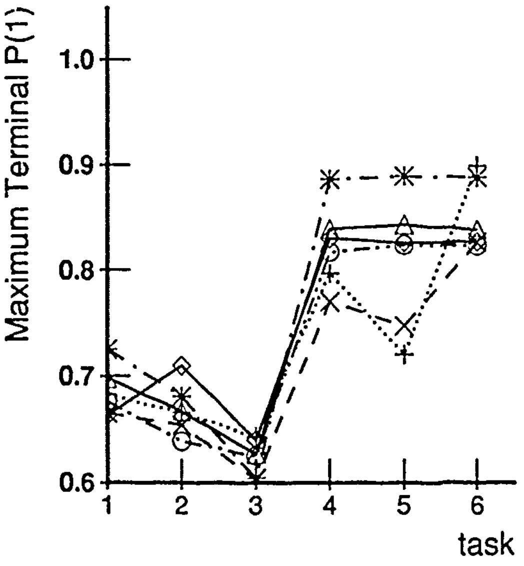


Figure 8: All tasks - best performing α, $\eta[t] \sim \mathcal{N}[0, 0.5]$

Learning Algorithms for Networks with Internal and External Feedback

Jürgen Schmidhuber*
Institut für Informatik
Technische Universität München
Arcisstr. 21, 8000 München 2, Germany
schmidhu@tumult.informatik.tu-muenchen.de

Abstract

This paper gives an overview of some novel algorithms for reinforcement learning in non-stationary possibly reactive environments. I have decided to describe many ideas briefly rather than going into great detail on any one idea. The paper is structured as follows: In the first section some terminology is introduced. Then there follow five sections, each headed by a short abstract. The second section describes the entirely local 'neural bucket brigade algorithm'. The third section applies Sutton's TD-methods to fully recurrent continually running probabilistic networks. The fourth section describes an algorithm based on system identification *and on two interacting fully recurrent 'self-supervised' learning networks. The fifth section describes an application of adaptive control techniques to adaptive attentive vision: It demonstrates how 'selective attention' can be learned. Finally, the sixth section critisizes methods based on system identification and adaptive critics, and describes an adaptive subgoal generator.*

1 Terminology

External feedback. Consider a neural network receiving inputs from a non-stationary environment and being able to produce actions that may have an influence on the environmental state. Since the new state may cause new inputs for the network we say that there is *external feedback.*

Internal feedback. If the network topology is cyclic, then input activations from a given time may alter the way that inputs from later times are processed. In this case there is a potential for the 'representation of state', or 'short term memory', and we speak of *internal feedback.*

Dynamic Learning Algorithms and Networks. A problem that requires credit assignment to past activation states is called a *dynamic problem.* Learning algorithms for handling dynamic problems are called *dynamic learning algorithms.* Learning algorithms that are not dynamic algorithms are called *static algorithms.* For instance, all algorithms that require settling into equilibria while the inputs have to remain stationary are considered to be static algorithms, although the settling process is a dynamic one based on internal feedback.

If a given network type can be employed for dynamic problems, and if there exists a corresponding learning algorithm, then we sometimes speak of a *dynamic network.*

The credit assignment problem. If a neural network is supposed to learn externally posed tasks then it faces Minsky's *fundamental credit assignment problem*: If performance is not sufficient, then which component of the network at which time did in which way contribute to the failure? How should critical components change behavior to increase future performance?

Supervised Learning. A learning task is a *supervised* learning task if there are externally defined desired outputs at certain times, but the network never needs to discover output actions on its own. Supervised learners have to consider only the internal feedback for performing credit assignment.

Reinforcement Learning. A learning task is a *reinforcement* learning task if the teacher only indicates once in a while whether the system is in a desirable state or not, without giving information about how to reach desirable states. Usually an evaluative (non-instructive) teaching mechanism sometimes provides a scalar signal, the reinforcement. whose value indicates success or failure. During training the network is supposed to discover on its own outputs that eventually lead to desirable states. In contrast to supervised learning,

*This work was supported by a scholarship from SIEMENS AG

there can be something like *undesired inputs* caused by former output actions. In general the external *unknown* dynamics have to be taken into consideration to perform credit assignment.

Reinforcement learning is strongly related to *control tasks*. With many control tasks more information is available about goal states than just a simple reinforcement signal. However, just as with reinforcement learning, the (sequential) outputs necessary to achieve the goal states in general are not known.

In the sequel we will concentrate on discrete time versions of dynamic learning algorithms for neural networks. We assume that there are 'time steps', and that state changes only take place from one time step to the next one, not within a time step.

A weak definition of 'locality in space and time' (there also is a stronger definition). A learning algorithm for dynamic neural networks is *local in time* if for given network sizes (measured in number of connections) during on-line learning the peak computation complexity at every time step is $O(1)$, for *arbitrary* durations of sequences to be learned.

A learning algorithm for dynamic neural networks is *local in space* if during on-line learning for limited durations of sequences to be learned and for *arbitrary* network sizes (measured in number of connections) and for *arbitrary* network topologies the peak computation complexity per connection at every time step is $O(1)$.

A learning algorithm for dynamic neural networks is *local* if during on-line learning for *arbitrary* durations of sequences to be learned and for *arbitrary* network sizes (measured in number of connections) and *arbitrary* network topologies the peak computation complexity per connection at every time step is $O(1)$.

These definitions do not imply that a local algorithm is unable to consider actions that have taken place any time before.

In the sequel some novel learning algorithms designed for networks with internal and external feedback will be described. Due to limited space I will describe many ideas briefly rather than going into great detail on any one idea.

2 The Neural Bucket Brigade Algorithm

Abstract. Competitive Learning 'shifts weight substance' from certain incoming connections of a winner-take-all-unit to other incoming connections. A novel algorithm for goal directed learning *with hidden units shifts weight substance from* outgoing *connections to incoming connections. An evaluative critic sometimes provides weight-substance for connections leading to output units. The algorithm's most significant advantage over other goal directed learning algorithms like back-propagation (Werbos, 1974)(Parker, 1985)(LeCun, 1985)(Rumelhart et al., 1986) is. It is biologically more plausible, because* it solely depends on computations which are entirely local in space and time. *It has been successfully applied to some classical non-linear problems involving both feedforward and recurrent networks.*

Competitive Learning (heavily employed in work on unsupervised learning (Kohonen, 1988) (Grossberg, 1976)) may be interpreted as 'shifting weight substance' from certain incoming connections of a winner-take-all-unit to other incoming connections (Rumelhart and Zipser, 1986). A novel algorithm for goal directed learning with hidden units emerges if weight substance is shifted from *outgoing* connections to incoming connections in a certain fashion.

Consider the general reinforcement learning situation where an evaluative critic in the environment sometimes provides 'payoff' in response to successful behavior of a learning feedforward or recurrent network. We translate reinforcement or payoff into weight-substance for connections leading to output units that were active in the moment of payoff. All such connections are immediately strengthened proportional to their last contributions. (A contribution is the product of a weight and an activation.)

However, even in the absence of payoff there are weight changes for all weights, including the weights of connections leading to hidden units. Any connection transporting activation information from an active unit i to another active unit j has to give up a part of its weight substance, which is shifted to those weights that were setting the stage by contributing to the activation of unit i at the last time step. Thus recursive dependencies 'through time' are established between strengths of connections transporting contributions during successive time steps. The environmental critic terminates the recursion. The algorithm shares certain conceptual similarities with the bucket brigade learning algorithm' for rule-based systems (Holland, 1985) and is called the 'Neural Bucket Brigade Algorithm'. One of the many differences is that competition works locally instead of globally.

The algorithm's most significant advantage over other goal directed learning algorithms like back-propagation is: *It solely depends on computations which are entirely local in space and time. This means that during on-line learning the peak computation per connection is not affected by network size or by network topology or by the length of input sequences. It is always O(1). This makes it biologically more plausible than other algorithms* (Schmidhuber, 1989) (Schmidhuber, 1990a).

The basic network structure is an arbitrary (possibly cyclic) graph which is partitioned into input units and small predefined winner-take-all-subsets, each having at least two members.

Notation: $x_j(t)$ is the activation of the jth unit at time t, $w_{ij}(t)$ is the weight on the directed connection from unit i to unit j at time t. $c_{ij}(t) = x_i(t-1)w_{ij}(t-1)$ denotes the 'contribution' of some connection between i and j at time t.

In the beginning weights are initialized with a positive value. The system is continuously receiving inputs, and continuously producing outputs, which again may have an influence on subsequent inputs (external feedback). Activations spread according to the following rules: At time t the input-units are clamped to values determined by the environment. Each non-input unit computes $net_j(t) = \sum_i c_{ij}(t)$. The winner-take-all-subsets ensure that only a fraction of the non-input units can be active simultaneously: $x_j(t)$ equals 1 if the non-input unit j is active, and 0 otherwise.

If unit j is active then its positive modifiable weights change according to

$$\Delta w_{ij}(t) = -\lambda c_{ij}(t) + \frac{c_{ij}(t-1)}{net_j(t-1)} \sum_{k\,active} \lambda c_{jk}(t) + \eta c_{ij}(t)$$

where $0 < \lambda < 1$ determines how much of its weight some particular connection has to pay to those connections that were responsible for *setting the stage* at the previous time step. η is a small constant if unit j is an output unit and if there is external payoff, and η is 0 otherwise. We get a dissipative system: 'Weight substance' enters the system in the case of payoff, flows through 'bucket brigade chains', and leaves the system through connections coming from input units.

Note again that the algorithm is entirely local. This makes a parallel implementation trivial. No teacher has to define something like beginnings and ends of back-propagation phases. No storage is required for past activations or contributions except for the most recent ones. The units do not care whether they are part of a feedforward or of a recurrent network. They do not care for concepts like 'layer structure' or network topology. *Each unit and each connection is performing the same simple operation at every time step.*

The algorithm has been successfully applied to some classical non-linear problems involving both feedforward and recurrent networks. Networks employing that algorithm learned to solve XOR-problems, encoding-problems, and sequence recognition (motion on a one-dimensional 'retina') as well as sequence generation (an oscillation task). To address the question of learning speed: The number of training cycles necessary to find some (not necessarily stable) solution for the XOR-problem is of the same order of magnitude as with conventional back-propagation. However, with the more complex encoding problems back-propagation seems to be faster by about an order of magnitude.

3 A Reinforcement Comparison Algorithm for Continually Running Fully Recurrent Probabilistic Networks

Abstract. The principle of reinforcement comparison (employed for learning to play checkers (Samuel, 1959) and learning to balance a pole (Barto et al., 1983)) says: Let the temporal derivative of the expectation of future reinforcement be the effective reinforcement. This principle is applied to fully recurrent continually running networks of probabilistic binary units. A main advantage of the resulting novel algorithm is its applicability to networks with internal (and possibly external) feedback and its locality in both space and time *(the absence of back-propagation-like operations makes it biologically more plausible than other algorithms).*

In addition to a fully recurrent continually running network with probabilistic binary output units the algorithm described in this section employs a second *linear* static network, called the critic, which learns to judge successive states of the recurrent network by learning to predict the final reinforcement to be received at the end of the current 'episode'. Differences of successive predictions serve to adjust both the critic and the recurrent network. Hereby the weights of the critic are updated according to the principles of Temporal Difference Methods (Sutton, 1988):

First all weights are randomly initialized with real values.

For all episodes:

In the beginning of each episode, at the first time step, the activations of input units of the recurrent network are initialized with values determined by sensory perceptions from the environment, and the activations of hidden and output units are initialized with 0. *For all following time steps, until there is external real-valued reinforcement R indicating failure or success:*

At any given time step t:

1. The critic's output $r = x^T(t-1)v(t)$ *is interpreted as a prediction of the final reinforcement to be received in the future.* ($v(t)$ *is the the critics current weight vector,* $x(t)$ *is the activation vector of all units of the recurrent network).*

2. Each probabilistic non-input unit i of the recurrent net sums its weighted inputs, this sum is passed to the logistic function $l(x) = \frac{1}{1+e^{-x}}$ *which gives the probability that the activation* $x_i(t)$ *becomes* 1, *or* 0, *respectively. Each unit i also stores its last activation $x_i(t-1)$. Output units may cause an action in the environment, and this may lead to new activations for the input units. So besides the internal feedback there may exist external feedback through the environment.*

3. If there is external reinforcement R (this means the end of the current episode) then the variable r′ is defined to be equal to R.

Otherwise r′ is defined to be a new estimation of final discounted reinforcement: $r' = \gamma x^T(t)v(t)$. ($0 < \gamma < 1$ is the discount rate).

The critic associates the last activation vector $x(t-1)$ of the recurrent network with r′, thus 'transporting expectation back in time' for one time step. So the critic's error is given by $r' - r$. Its weight vector is immediately updated according to the Widrow-Hoff rule, the result is a new weight vector $v(t+1)$.

4. Each directed weight $w_{ij}(t)$ from unit i to unit j of the recurrent network is immediately altered according to $\Delta w_{ij}(t) = \lambda(r'-r)x_i(t-1)(x_j(t) - P(x_j = 1 \mid x(t-1), w(t-1))$, where $w(t-1)$ is the last weight vector, and λ is a positive constant. Thus the last transition gets encouraged (or discouraged, respectively).

The algorithm applies the principle of reinforcement comparison to dynamic recurrent neural networks (Schmidhuber, 1990d). Informally, this principle also can be formulated as follows:

If a system is in a state which it assumes to be a bad state, but there is a transition which leads to a state assumed to be a good state, then this transition should be encouraged. Furthermore, from now on the 'bad' state also can be considered to be a good state. Transitions from good states to bad states have to be treated in an analogue fashion.

Note again that *unlike with back-propagation-like algorithms for recurrent networks the algorithm above is local in both space and time.* This means that during on-line learning *the peak computation per connection is not affected by network size or input duration. It is always O(1).*

It is worth mentioning a counterintuitive fact. The critic may be linear, however, the task of the recurrent network may be of the non-linearily separable type. This has been shown by successfully applying the algorithm to a 'delayed XOR-problem'. A reinforcement signal given in the end of each training episode (involving a small number of time steps) indicated whether the recurrent network correctly computed the *delayed* response to one of the four XOR patterns. The critic may be linear, because the final mapping to be implemented by the critic in general is simpler than the final mapping to be implemented by the main network.

The algorithm shares certain conceptual similarities with the 'neural bucket brigade algorithm' (Schmidhuber, 1990g). In (Schmidhuber, 1990d) it also has been described how a *recurrent* critic can interact with the recurrent primary network.

4 Two Interacting Fully Recurrent Self-Supervised Learning Networks for Reinforcement Learning

Abstract. An extension of system identification approaches for adaptive control by Werbos, Jordan, Munro, Widrow, and Robinson and Fallside is described. The algorithm is based on two interacting fully recurrent continually running networks which may learn in parallel. The algorithm has a potential for on-line learning and locality in time, it does not care for 'epoch-boundaries', it needs only reinforcement information for learning, it allows different kinds of reinforcement (or pain), it allows both internal and external feedback with theoretically arbitrary time lags, and it includes a full environmental model thus providing complete 'credit assignment paths' into the past.

An extension of system identification approaches for adaptive control ((Werbos, 1977), (Jordan, 1988), (Munro, 1987), (Nguyen and Widrow, 1989), (Robinson and Fallside, 1989)) is described.

The algorithm attempts to be a very general one. It attacks the fundamental spatio-temporal credit assignment problem as far as it is attackable at all by pure gradient descent methods (Schmidhuber, 1990b).

The output units of a dynamic recurrent *control network* may influence the state of a reactive non-stationary environment, thus influencing subsequent inputs of the control network. The input of a dynamic fully recurrent *model network* at every time is given by the input and the output of the control network. The model network is trained to predict future activations of the input units of the control network. *Among the control network's input units there are 'reinforcement units' whose desired activations are fixed for all times.* For instance, the desired activations of so-called *'pain-units'* are zero for all times. At a given time the quantity to be minimized by the controller is $\sum_{t,i}(c_i - y_i(t))^2$, where $y_i(t)$ is the activation of the ith reinforcement input unit at time t and c_i is its desired activation for all times. (t ranges over all (discrete) time steps that are still to come.)

Following the approach of system identification, the model network helps to define desired output activations for the control network. Errors for the controller's weights are computed by measuring the partial derivatives of cumulative pain predictions of the model network with respect to controller weights. Hereby the *frozen* model network is taken to be an emulator of the environmental dynamics.

The algorithm can be run in two different modes: There is the sequential version and the parallel version. With the sequential version, first the model network is trained by providing it with randomly chosen examples of sequences of interactions between controller and environment. Then the model weights are fixed to their current values, and the controller begins to learn.

With the parallel version both the controller and the model learn concurrently. The advantage of the parallel version is that the model network focusses only on those parts of the environmental dynamics which the controller typically is confronted with. Particularily with complex environments this represents an enormous potential for gaining efficiency. The disadvantage of the parallel version is that the controller sometimes receives wrong error gradients caused by an inperfect model. This should not be serious, as long as the model continues to improve. However, the controller might enter a local minimum relative to the current state of the model network's weights. This in turn may cause the controller to perform the same silly actions all the time, thus preventing the model network from improving (learning about the effects of alternative actions). Then the whole system might be caught in a state from which it cannot escape any more. The sequential version represents a safer way, but it lacks the flavor of real on-line learning and locality in time.

Below we describe the parallel version. The sequential version can be obtained in a straight-forward manner. An on-line version of the Infinite Input Duration (IID) learning algorithm for fully recurrent networks (Robinson and Fallside, 1987) is employed for training both the model network and the control network. (The IID algorithm was first experimentally tested by (Williams and Zipser, 1989).)

At every time step, the parallel version of the algorithm is performing essentially the same operations.

In step 1 of the main loop of the algorithm actions in the external world are computed. Due to the internal feedback, these actions are based on previous inputs and outputs. For all new activations, the corresponding derivatives with respect to all controller weights are updated.

In step 2 actions are executed in the external world, and the effects of the current action and/or previous actions may become visible.

In step 3 the model network tries to predict these effects without seeing the new input. Again the relevant gradient information is computed.

In step 4 the model network is updated in order to better predict the input (including reinforcement and pain) for the controller. Finally, the weights of the control network are updated in order to minimize the cumulative differences between desired and actual activations of the pain and reinforcement units. Since the control network continues activation spreading based on the actual inputs instead of using the predictions of the model network, 'teacher forcing' (Williams and Zipser, 1989) is used in the model network (although there is no teacher besides the environment).

One can find various improvements of the systems described in (Schmidhuber, 1990b) and (Schmidhuber, 1990e). For instance, the partial derivatives of the controller's inputs with respect to the controller's weights are approximated by the partial derivatives of the corresponding predictions generated by the model network. Furthermore, the model sees the last input and current output of the controller at the same time.

Notation (the reader may find it convenient to compare with (Williams and Zipser, 1989)):

C is the set of all non-input units of the control network, A is the set of its output units, I is the set of its 'normal' input units, P is the set of its pain and reinforcement units, M is the set of all units of the model network, O is the set of its output units, $O_P \subset O$ is the set of all units that predict pain or reinforcement, W_M is the set of variables for the weights of the model network, W_C is the set of variables for the weights of the control network, $y_{k_{new}}$ is the variable for the updated activation of the kth unit from $M \cup C \cup I \cup P$, $y_{k_{old}}$ is the variable for the last value of $y_{k_{new}}$, w_{ij} is the variable for the weight of the directed connection from unit j to unit i, $p^k_{ij_{new}}$ is the variable which gives the current (approximated) value of $\frac{\partial y_{k_{new}}}{\partial w_{ij}}$, $p^k_{ij_{old}}$ is the variable which gives the last value of $p^k_{ij_{new}}$, if $k \in P$ then c_k is k's desired activation for all times, α_C is the learning rate for the control network, α_M is the learning rate for the model network.

$| I \cup P |=| O |$, $| O_P |=| P |$. If $k \in I \cup P$, then kpred is the unit from O which predicts k. Each unit from $I \cup P \cup A$ has one forward connection to each unit from $M \cup C$. Each unit from M is connected to each other unit from M. Each unit from C is connected to each other unit from C. Each weight of a connection leading to a unit in M is said to belong to W_M. Each weight of a connection leading to a unit in C is said to belong to W_C. Each weight $w_{ij} \in W_M$ needs p^k_{ij}-values for all $k \in M$. Each weight $w_{ij} \in W_C$ needs p^k_{ij}-values for all $k \in M \cup C \cup I \cup P$.

The parallel version of the algorithm works as follows:

```
INITIALIZATION:
For all $w_{ij} \in W_M \cup W_C$:
  begin $w_{ij} \leftarrow$ random,
  for all possible k: $p^k_{ij_{old}} \leftarrow 0, p^k_{ij_{new}} \leftarrow 0$ end.
For all $k \in M \cup C : y_{k_{old}} \leftarrow 0, y_{k_{new}} \leftarrow 0$.
For all $k \in I \cup P$ :
  Set $y_{k_{old}}$ by environmental perception, $y_{k_{new}} \leftarrow 0$.

FOREVER REPEAT:
1. For all $i \in C : y_{i_{new}} \leftarrow \frac{1}{1+e^{-\sum_j w_{ij}y_{j_{old}}}}$.
   For all $w_{ij} \in W_C, k \in C$:
     $p^k_{ij_{new}} \leftarrow y_{k_{new}}(1-y_{k_{new}})(\sum_l w_{kl}p^l_{ij_{old}} + \delta_{ik}y_{j_{old}})$.
   For all $k \in C$:
     begin $y_{k_{old}} \leftarrow y_{k_{new}}$,
     for all $w_{ij} \in W_C$ : $p^k_{ij_{old}} \leftarrow p^k_{ij_{new}}$ end .
2. Execute all motoric actions based on activations of
   units in A. Update the environment.
   For all $i \in I \cup P$:
     Set $y_{i_{new}}$ by environmental perception.
3. For all $i \in M : y_{i_{new}} \leftarrow \frac{1}{1+e^{-\sum_j w_{ij}y_{j_{old}}}}$.
   For all $w_{ij} \in W_M \cup W_C, k \in M$:
     $p^k_{ij_{new}} \leftarrow y_{k_{new}}(1-y_{k_{new}})(\sum_l w_{kl}p^l_{ij_{old}} + \delta_{ik}y_{j_{old}})$
   For all $k \in M$:
     begin $y_{k_{old}} \leftarrow y_{k_{new}}$,
     for all $w_{ij} \in W_C \cup W_M$ : $p^k_{ij_{old}} \leftarrow p^k_{ij_{new}}$ end.
4. For all $w_{ij} \in W_M$:
     $w_{ij} \leftarrow w_{ij} + \alpha_M \sum_{k \in I \cup P}(y_{k_{new}} - y_{kpred_{old}})p^{kpred}_{ij_{old}}$.
   For all $w_{ij} \in W_C$:
     $w_{ij} \leftarrow w_{ij} + \alpha_C \sum_{k \in P}(c_k - y_{k_{new}})p^{kpred}_{ij_{old}}$.
   For all $k \in I \cup P$:
     begin $y_{k_{old}} \leftarrow y_{k_{new}}$, $y_{kpred_{old}} \leftarrow y_{k_{new}}$,
     for all $w_{ij} \in W_M$ : $p^{kpred}_{ij_{old}} \leftarrow 0$,
     for all $w_{ij} \in W_C$ : $p^k_{ij_{old}} \leftarrow p^{kpred}_{ij_{old}}$ end.
```

To attack the above-mentioned problem with the parallel version of the algorithm we can introduce a probabilistic element for the controller actions. By employing probabilistic output units for C and by using 'gradient descent through random number generators' (Williams, 1988) we can introduce explicit explorative random search capabilities into the otherwise deterministic algorithm. In the context of the IID algorithm, this works as follows: A probabilistic output unit k consists of a conventional unit $k\mu$ which acts as a mean generator and a conventional unit $k\sigma$ which acts as a variance generator. At a given time, the probabilistic output $y_{k_{new}}$ is computed by

$$y_{k_{new}} = y_{k\mu_{new}} + z y_{k\sigma_{new}},$$

where z is distributed e.g. according to the normal distribution. The corresponding $p^k_{ij_{new}}$ have to be updated according to the following rule:

$$p^k_{ij_{new}} \leftarrow p^{k\mu}_{ij_{new}} + \frac{y_{k_{new}} - y_{k\mu_{new}}}{y_{k\sigma_{new}}} p^{k\sigma}_{ij_{new}}.$$

By performing more than one iteration of step 1 and step 3 at each time tick, one can adjust the algorithm to environments that change in a manner which is not predictable by semilinear operations (theoretically three additional iterations are sufficient for any environment).

The parallel version of the algorithm is local in time, but not in space. See (Schmidhuber, 1990b) for a justification of certain deviations from 'pure gradient descent through time', and for a description of how the algorithm can be used for planning action sequences.

Variants of the algorithm are currently tested on certain non-Markovian reinforcement learning tasks. For instance, a controller was able to learn to be a flip-flop similar to the one described in (Williams and Zipser, 1989). Of course, the important difference was that no teacher provided the desired outputs!

Other experiments are currently conducted with a non-Markovian pole balancing task. Unlike with tasks described in (Barto et al., 1983) and (Anderson, 1986), no information about temporal derivatives of the system's state variables (cart position, pole angle with the vertical) is provided. The recurrency of the model network provides a potential for extracting this kind of information, and to represent the state of the environment in a form that allows credit assignment for the controller.

In (Schmidhuber, 1990b) it is described how the algorithm can be employed for *planning* action sequences. It should be noted that the algorithm also could be used as a submodule in an *adaptive critic system* consisting of *three* networks (Schmidhuber, 1990f), where the adaptive critic computes *vector-valued* predictions of future events. This contrasts previous adaptive critics, whose output is just a *scalar* evaluation of the current state.

The parallel version of the algorithm described above has properties which allow to implement something like *the desire to improve the model network's knowledge about the world.* This is related to *curiosity.* In (Schmidhuber, 1990c) it is described how the algorithm can be augmented by dynamic *curiosity* and *boredom* in a natural manner. This can be done by introducing (delayed) reinforcement for actions that increase the model network's knowledge about the world. This in turn requires the model network *to model its own ignorance,* thus showing a rudimentary form of *self-introspective* behavior.

5 An Example for Learning Dynamic Selective Attention: Adaptive Focus Trajectories for Attentive Vision

Abstract. It is shown how certain cases of selective attention can be learned. *'Static' neural approaches to certain pattern recognition tasks can be replaced by a more efficient sequential approach. A system is described which* learns *to generate focus trajectories such that the final position of a moving focus corresponds to a target in a visual scene.* No teacher *provides the desired activations of 'eye-muscles' at various times. The only goal information is the* desired final input *corresponding to the target. The task involves a complex temporal credit assignment problem and an attention shifting problem. The system also learns to track moving targets.*

There is little doubt that selective attention is essential for large scale dynamic control systems. In this section we study the problem of *learning* selective attention in the context of attentive vision with dynamic neural networks. The problem, which in its general form has not been explored before, is the control of sequential physical focus-movements. Hereby we concentrate on the question: How can an attentive vision system *learn without a teacher* to generate focus trajectories such that the final visual input always looks like a desireable input corresponding to a target?

A visual scene is given by an object (with internal details) placed on a 512 x 512 pixel field. The object covers only a small part of the scene and may be rotated or translated in an arbitrary manner. Instead of using tenthousands of input units (as in a straight-forward static approach) only about 40 input units are employed. However, these units are sitting on a focus (a two-dimensional artificial retina) which can be moved across the pixel plane. The focus has high resolution in its center and low resolution in its periphery.

In our approach there is a neural *control network C* that controls sequential focus movements. Motoric actions like 'move focus left', 'rotate focus' are based on the activations of the C's output units at a given time. Thus output actions may cause new activations for the input units, and we say that there is *external feedback* (through the environment). The final desired input is an activation pattern corresponding to the target in a static visual scene. The task is to sequentially generate a focus trajectory such that the final input matches the target input. C's error at the end of a sequential recognition process is given by the *difference between the desired final input and the actual final input.* (Control theory calls this a 'terminal control problem'.)

Pure supervised learning techniques for neural networks work only if there is a teacher who provides target *outputs* at every time step of a trajectory (which in our case usually involves about 30 time steps). In our case, however, there never are externally given desired outputs. There only is one final desired *input.*

In order to allow credit assignment to past output actions of C, we employ a supervised learning *model network M* which separately learns to represent a model of the visible environmental dynamics. This is done by training M at a given time to predict C's next input. This prediction is based on previous inputs and outputs of the controller. M serves to 'make the world differentiable'. It serves to bridge the gap between output units and input units of the controller.

A learning algorithm for dynamic recurrent networks is employed to propagate gradient information for C's weights back through M down into C and back through M etc... M's weights remain fixed during this procedure. In different contexts and with different degrees of generality, this basic principle for credit assignment based on *system identification* has been previously described in (Werbos, 1977), (Jordan, 1988), (Munro, 1987), (Robinson and Fallside, 1989), (Nguyen and Widrow, 1989), and (Schmidhuber, 1990b).

Note that in most cases the model network will not be perfect. For instance, if objects in a visual scene may occupy random positions then it will be impossible for M to exactly predict future focus inputs from previous ones. However, it is not intended to make the model a perfect predictor whose output could replace the input from the environment (in that case not much would be gained compared to the static approach: There would be no need for dynamic attention). It suffices if the inner products of the approximated gradients (based on an inaccurate model) for the control network and the true gradients (according to a perfect model) tend to be positive.

M's main task is to help the controller to move the focus into regions of the plane which *allow to continue with more informed moves.* (Although one can not exactly predict what one will see after moving one's eyes to the door, one is setting the stage for additional eye-movements that help to recognize an entering person.)

One goal of this work is to demonstrate that imperfect models can contribute to perfect solutions. Our experiments show *that the system described above is able to learn (without a teacher) correct sequences of focus movements involving translations and rotations*, although M often makes erroneous predictions. At the end of a trajectory, the focus has moved towards a certain target part of the object and is rotated such that the final input corresponds to the desired input (Schmidhuber and Huber, 1990) (Huber, 1990).

Further experiments showed that the system is well-suited for *target tracking*. The desired detail of the moving object soon is focussed and tracked, as long as the objects velocity does not excess the maximal focus velocity.

Further experiments were conducted where C and M learned concurrently. It was found that two interacting conventional *deterministic* networks were *not* appropriate. So each of C's output units was replaced by a little network consisting of two units, one giving the mean and the other one giving the variance for a random number generator which produced random numbers according to a continuous distribution. (We approximated a Gauss distribution by a Bernoulli distribution.) Weight gradients were computed by applying William's concept of 'back-propagation through random number generators' (Williams, 1988).

It was found that such an on-line learning system can be able to learn appropriate focus trajectories. As it was expected, after training M was a good predictor *only* for those situations which the controller typically was confronted with.

6 An Adaptive Subgoal Generator for Planning Action Sequences

Abstract. None of the existing learning algorithms for sequentially working neural networks with internal and/or external feedback addresses the problem of learning 'to divide and conquer'. It is argued that algorithms based on pure gradient descent or on adaptive critic methods are not suitable for large scale dynamic control problems, and that there is a need for algorithms that perform 'compositional learning'. A system is described which solves at least one problem associated with compositional learning. The system learns *to generate sub-goals. This is done with the help of 'time-bridging' adaptive models that predict the effects of the system's sub-programs.*

The algorithms for attacking the fundamental credit assignment problem with dynamic learning algorithms in non-stationary environments can be classified into two major categories.

First, there is the approach of back-propagation through time'. This approach has been pursued by (Robinson and Fallside. 1987), (Werbos, 1988), (Pearlmutter, 1989), (Rumelhart et al., 1986), (Williams and Zipser, 1989), (Gherrity, 1989) and others in the case where there is only internal feedback. It has been pursued by (Nguyen and Widrow, 1989), (Robinson and Fallside, 1989), (Werbos, 1977), (Jordan, 1988), and (Schmidhuber, 1990b) in the case where there also is external feedback through a reactive environment.

Second, there is the 'Adaptive Critic' approach, which is of primary interest in the case of external feedback. This approach has been pursued by (Samuel, 1959), (Barto et al., 1983), (Werbos, 1990), and (Schmidhuber, 1990d).

Both the algorithms based on pure gradient descent as well as the 'Adaptive Critic' algorithms have at least one thing in common. They show significant drawbacks when the credit assignment process has to bridge long time gaps between past actions and later consequences.

Both approaches show awkward performance in the case where the learning system already has learned a lot of action sequences in the past. Both approaches tend to modify 'sub-programs', instead of modifying the trigger conditions for sub-programs. They do not have an explicit concept of something like a sub-program. Pure gradient descent methods *always* consider *all* past states for credit assignment. Adaptive critics based on Sutton's 'Temporal Differences' (reinforcement comparison methods) or on Werbos' 'Heuristic Dynamic Programming' consider only the most recent states for 'handing expectations back into time'. Both methods in general tend to consider the wrong states. This is a major reason for their slow performance.

In the next section we will isolate one problem associated with *'compositional learning'*, namely, the problem of learning to generate sub-goals when there already exist a number of working sub-programs (Schmidhuber, 1990h).

6.1 Learning to Generate Sub-Goals

The sub-goal generating system to be described in this section consists of three modules. The heart of the system is a neural network with internal and external feedback, called the control network C. C serves as a program executer. It receives as input a start state, a desired goal state, and time-varying inputs from the environment. The start and goal states serve as 'program names'. We assume that C already has learned to solve a number of tasks. This means that there already are various working programs that actually lead from the start states to the goal states by which the programs are indexed. These programs may have been learned by an algorithm for dynamic networks (as described by the authors mentioned above), *or by a recursive application of the principle outlined below.*

A second important module is a static evaluator network E which receives as input a start state and a goal state, and produces an output that indicates whether there is a program that leads from the start state to or 'close' to the goal state. An output of 1 means that there *is* an appropriate sub-program, an output of 0 means that there is no appropriate sub-program. An output between 0 and 1 means that there is a sub-program that leads from the start state to a state that comes close to the goal, in a certain sense. This measure of closeness has to be given by some evaluative process that may be adaptive or not, and which will not be specified in detail in this paper. (It may be based on TD-methods, for instance.) E represents the system's current model of its own capabilities. We assume that E has learned to correctly predict that each of the already existing sub-programs works. We also assume that E is able to predict the closeness of an end state of a sub-program to a goal state, given a start state. E can be trained in an exploratory phase during which various combinations of start and goal states are given to the program executer.

Finally, the system contains a static network which serves as a sub-goal generator. The sub-goal generator receives as input the external start-input to C, and the desired input (the goal) for C at the end of the task.

The output of the sub-goal generator is a sub-goal, of course. Like the goal, the sub-goal is an activation pattern describing the desired external input at the end of some sub-program, which also is the start input for another sub-program. We concentrate on the most simple case, namely, the case where solutions for given tasks can be found by generating only one sub-goal. The sub-goal generator should output a sub-goal for which there exists a sub-program leading from the start state to the sub-goal, and furthermore a sub-program leading from the sub-goal to the goal state.

How does the sub-goal generator, which initially is a *tabula rasa*, learn to generate appropriate sub-goals? We take two copies of E. The first copy sees the description of a start state and the description of the sub-goal generated by the sub-goal generator. The second copy sees the description of the same sub-goal and the description of the goal. The desired output of each of the copies is 1. Whenever one of the outputs of the copies is below 1, an error gradient is propagated through E's copies *down into the sub-goal generator.* E (as well as its copies, of course) remain unchanged during this procedure. Only the weights of the sub-goal generator change. For a given problem the procedure is iterated until the complete error is zero (corresponding to a solution obtained by combining the two sub-programs), or until a local minimum is reached (no solution found). The gradient descent procedure is used for a search in sub-goal space.

In some experiments with a simple environment a robot was taught to solve certain sequential tasks, like moving from one point to another one. Then more complicated tasks were posed that did not have an associated sub-program.

The sub-goal generator soon learned to generate appropriate sub-goals for the robot.

It should be noted that there also is a different slightly more complex architecture which allows *vector-valued* evaluations of the expected effects of sub-programs.

References

Anderson, C. W. (1986). *Learning and Problem Solving with Multilayer Connectionist Systems.* PhD thesis, University of Massachusetts, Dept. of Comp. and Inf. Sci.

Barto, A. G., Sutton, R. S., and Anderson, C. W. (1983). Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics,* SMC-13:834–846.

Gherrity, M. (1989). A learning algorithm for analog fully recurrent neural networks. In *IEEE/INNS International Joint Conference on Neural Networks, San Diego,* volume 1, pages 643–644.

Grossberg, S (1976). Adaptive pattern classification and universal recoding, 1: Parallel development and coding of neural feature detectors. *Biological Cybernetics,* 23:187–202.

Holland, J. H. (1985). Properties of the bucket brigade. In *Proceedings of an International Conference on Genetic Algorithms.* Hillsdale, NJ.

Huber, R. (1990). Selektive visuelle Aufmerksamkeit: Untersuchungen zum Erlernen von Fokustrajektorien durch neuronale Netze. Diplomarbeit. Institut für Informatik, Technische Universität München.

Jordan, M. I. (1988). Supervised learning and systems with excess degrees of freedom. Technical Report COINS TR 88-27, Massachusetts Institute of Technology.

Kohonen, T. (1988). *Self-Organization and Associative Memory.* Springer, second edition.

LeCun, Y. (1985). Une procédure d'apprentissage pour réseau à seuil asymétrique. *Proceedings of Cognitiva 85, Paris,* pages 599–604.

Munro, P. W. (1987). A dual back-propagation scheme for scalar reinforcement learning. *Proceedings of the Ninth Annual Conference of the Cognitive Science Society, Seattle,* WA, pages 165–176.

Nguyen and Widrow, B. (1989). The truck backer-upper: An example of self learning in neural networks. In *IEEE/INNS International Joint Conference on Neural Networks, Washington, D.C.,* volume 1, pages 357–364.

Parker, D. B. (1985). Learning-logic. Technical Report TR-47, Center for Comp. Research in Economics and Management Sci., MIT.

Pearlmutter, B. A. (1989). Learning state space trajectories in recurrent neural networks. *Neural Computation*, 1:263–269.

Robinson, A. J. and Fallside, F. (1987). The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department

Robinson, T. and Fallside, F. (1989). Dynamic reinforcement driven error propagation networks with application to game playing. In *Proceedings of the 11th Conference of the Cognitive Science Society, Ann Arbor*, pages 836–843.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In (Rumelhart and McClelland, 1986), pages 318–362.

Rumelhart, D. E. and McClelland, J. L., editors (1986). *Parallel Distributed Processing*, volume 1. MIT Press.

Rumelhart, D. E. and Zipser, D. (1986). Feature discovery by competitive learning. In (Rumelhart and McClelland, 1986), pages 151–193.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development*, 3:210–229.

Schmidhuber, J. H. (1989). The neural bucket brigade. In Pfeifer, R., Schreter, Z., Fogelman, Z., and Steels, L., editors, *Connectionism in Perspective*, pages 439–446. Amsterdam: Elsevier, North-Holland.

Schmidhuber, J. H. (1990a). A local learning algorithm for dynamic feedforward and recurrent networks. *Connection Science*, 1(4):403–412.

Schmidhuber, J. H. (1990b). An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *Proc. IEEE/INNS International Joint Conference on Neural Networks, San Diego*, volume 2, pages 253–258.

Schmidhuber, J. H. (1990c). A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats, in press*. MIT Press/Bradford Books.

Schmidhuber, J. H. (1990d). Recurrent networks adjusted by adaptive critics. In *Proc. IEEE/INNS International Joint Conference on Neural Networks, Washington, D. C.*, volume 1, pages 719–722.

Schmidhuber, J. H. (1990e). Reinforcement learning with interacting continually running fully recurrent networks. In *Proc. INNC International Neural Network Conference, Paris*, volume 2, pages 817–820.

Schmidhuber, J. H. (1990f). Response to G. Lukes' review of 'Recurrent networks adjusted by adaptive critics'. *Neural Network Review, in press*.

Schmidhuber, J. H. (1990g). Temporal-difference-driven learning in recurrent networks. In Eckmiller, R., Hartmann, G., and Hauske, G., editors, *Parallel Processing in Neural Systems and Computers*, pages 209–212. North-Holland.

Schmidhuber, J. H. (1990h). Towards compositional learning with dynamic neural networks. Technical Report FKI-129-90, Institut für Informatik, Technische Universität München.

Schmidhuber, J. H. and Huber, R. (1990). Learning to generate focus trajectories for attentive vision. Technical Report FKI-128-90, Institut für Informatik, Technische Universität München.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44.

Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University.

Werbos, P. J. (1977). Advanced forecasting methods for global crisis warning and models of intelligence. In *General Systems*, volume XXII, pages 25–38.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1.

Werbos, P. J. (1990). Consistency of HDP applied to a simple reinforcement learning problem. *Neural Networks*, 2:179–189.

Williams, R. J. (1988). On the use of backpropagation in associative reinforcement learning. In *IEEE International Conference on Neural Networks, San Diego*, volume 2, pages 263–270.

Williams, R. J. and Zipser, D. (1989). Experimental analysis of the real-time recurrent learning algorithm. *Connection Science*, 1(1):87–111.

Part III

Genetic Learning

Exploring Adaptive Agency I: Theory and Methods for Simulating the Evolution of Learning

Geoffrey F. Miller
Department of Psychology
Jordan Hall, Building 420
Stanford University
Stanford, CA 94305
geoffrey@psych.stanford.edu

Peter M. Todd
Department of Psychology
Jordan Hall, Building 420
Stanford University
Stanford, CA 94305
todd@psych.stanford.edu

Abstract

Psychology construed as the scientific study of adaptive agency can include not only modelling of specific psychological adaptations in particular species, but general exploration of the adaptive processes (including evolution, learning, and computation) that build, modify, and instantiate those adaptations. Connectionist theory has concentrated on understanding the adaptive processes of learning and computation, and has assumed general-purpose learning principles as the prime constructors of psychological adaptations. But connectionism has thereby ignored the central lesson of a century of learning theory in psychology: learning mechanisms must be understood in terms of their specific adaptive functions, just like other psychological adaptations. This paper introduces the notion of psychology as the study of adaptive agency, outlines a hierarchy of adaptive processes underlying adaptive agency, and reviews the history of learning theory and the emergence of ecological and evolutionary approaches to learning. We then develop a taxonomy of adaptive functions that learning mechanisms might serve, and outline a general simulation framework for exploring those adaptive functions. Finally, we present empirical results concerning the simulated evolution of associative learning.

1 INTRODUCTION

Natural selection has constructed animals' minds and behavioral repertoires for adaptive fit to the environmental problems they must face. As the study of such minds and behavioral repertoires, a properly evolutionarily-informed psychology should focus on the notion of *adaptive agency* -- the generation of action in the world in response to challenges to fitness. This framework encompasses many approaches, including (A) analysis and modelling of complex species-typical psychological adaptations (as in human and animal experimental psychology and cognitive ethology), (B) comparison of psychological adaptations across species and consideration of their phylogenetic origins (as in comparative psychology), and (C) general exploration of the adaptive processes themselves that yield adaptive agency (e.g. by simulation methods, including those in the field of artificial life -- see Langton, 1989). Most connectionists doing psychological modelling have contributed primarily to the first of these three enterprises. This paper concentrates on the third.

One of the central applications of connectionist theory has been to develop parallel distributed processing models of psychological mechanisms in humans and, less frequently, other animals (e.g. Rumelhart & McClelland, 1986, Vol. II; Sutton & Barto, 1987; Gluck, Bower, & Hee, 1989). Although most connectionists would deny the charge that they are the 'neo-Behaviorists' of psychological modelling, many seem to adhere to one of the most central, and most problematic, assumptions of Behaviorist psychology: that learning mechanisms can be studied without regard to their specific adaptive functions for the particular species investigated. The very notion of using an 'animal model' (e.g. a rat or pigeon) to investigate human learning assumes the existence of cross-species universal learning principles which, on evolutionary grounds, we may have little reason to expect. We would argue, to the contrary, that in pursuing the first approach to the study of adaptive agency outlined above,

(1) connectionists interested in modelling portions of human or animal minds should attempt to model real domain-specific psychological adaptations (the natural building blocks of minds -- see Cosmides & Tooby, 1987) rather than other units of analysis; (2) connectionists interested in modelling domain-specific psychologi cal adaptations should explore whether, when, and how specific learning mechanisms might be used by those adaptations, rather than simply assuming the adaptive utility of some hypothesized general-purpose learning mechanism. On the other hand, if one is interested in pursuing the third approach to the study of adaptive agency, we would suggest that (3) connectionists *can* usefully explore the general features of adaptive agency by studying how adaptive processes themselves work and interact, without reference to specific psychological adaptations in a particular species. Point 3 seems to conflict with points 1 and 2 because it suggests a less domain-specific, species-specific methodological focus. But just as one can do theoretical astrophysics without constructing specific astronomical models of particular observable celestial bodies, one can explore the dynamics of adaptive processes in general without modelling their specific structural outcomes in the minds and behavioral repertoires of particular species.

Our empirical research concentrates on point 3, because we are currently more interested in how adaptive processes interact than in the particular adaptations those processes happen to have produced on this plane., at this time. This paper lays out the theoretical framework and methodological principles guiding our work in exploring adaptive agency. Elsewhere (Todd & Miller, 1990), we describe in more detail the specific methods and results obtained in simulating the evolution of associative learning as a mechanism for 'imprinting' on certain features of one's evolutionary niche; and in a forthcoming paper (Todd & Miller, in press), we extend this simulation method to understanding habituation and sensitization mechanisms as evolutionary adaptations for behaviorally tracking short-term environmental dynamics.

2 NATURAL SELECTION AND THE EVOLUTION OF SUBSIDIARY ADAPTIVE PROCESSES

Evolution as an adaptive process has itself undergone changes: "survival of the stable" probably preceded "survival of the fittest" (Dawkins, 1976). Evolution in the earth's early environment is likely to have selected for physical systems (e.g. autocatalytic sets) with relative stability in the simmering primordial soup. After stability came replication and metabolism. the ability to turn external material into copies and extensions of oneself. Simple physical systems thus evolved into replicating systems. Larger, more complex phenotypes probably evolved to protect the replicators against biochemical breakdown and facilitate their intake of materials for replication and metabolism. These larger phenotypes set the stage for the evolution of behavior-generating systems that could produce innately programmed sequences of activity and movement. (In the land of the sessile, the blind, roving, pre-programmed predator may be king.) Sensory systems could then evolve to guide these behavior-generators more adaptively, based on sensitivity to particular environmental cues. (In the land of the blind, the optic-spotted paramecium may be king.) Thus, blind activity may have preceded reactivity -- behavioral adaptation to the current changing environment on a moment-by-moment basis.

Only after these first two adaptive processes (natural selection with a genotype/phenotype distinction, and behavioral reactivity) had emerged could a third evolve -- 'learning,' defined as the ability to make long(ish)-term adaptive changes in evolved behavior-generators in response to particular environmental conditions and dynamics.[1] (See Shepard, 1987a, 1989, for discussion of these adaptive processes in relation to psychology and connectionist modelling). In this hierarchy, learning emerges not as the primary adaptive force that some theorists (e.g. Behaviorists) have assumed it to be, but rather as a tertiary one, following long-term genotypic evolution and short-term environmental reactivity. Once we reconceptualize 'learning' as merely one process among several that help generate and support adaptive agency, the questions we might ask about this process begin to change as well.

Traditionally, researchers using simulation to explore adaptive agency have started by assuming learning as a primary adaptive process, and then asked how evolution might shape and be shaped by learning. For example, Hinton and Nowlan (1987) and Belew (1990) explicate the Baldwin effect in which learning "guides" evolution; Stork and Keesing (1990) and Belew, McInerney, and Schraudolph (1990) investigate how evolution and learning can combine to affect the initial structure of neural networks. But considering the hierarchy of adaptive processes spelled out above, the question *we* want to ask is, rather, given the already-powerful adaptive processes of genotypic evolution and environmental reactivity, under what conditions would the tertiary adaptive process of learning ever prove useful in terms of increasing individual fitness? Assuming *no* learning, when should learning evolve? We consider learning a mystery to be explained rather than a commonsense explanation for other phenomena.

Fortunately, connectionists are not the first group to grapple with these thorny issues. The history of learning theory in comparative psychology has considered such problems for over a century, and is slowly arriving at a

[1] By this definition, learning includes such processes as experience-guided development not commonly included in this category -- see, e g , Singer, 1988 We discuss this issue further in section 5.

theoretical consensus that we can take, almost off the shelf, to guide our modelling and simulation efforts. Ignorance of this intellectual heritage may not preclude success, but knowledge of it should help. Connectionists can, at least, hope to avoid recapitulating the pitfalls of certain historical ways of thinking about learning.

3 A BRIEF HISTORY OF LEARNING THEORY IN (COMPARATIVE) PSYCHOLOGY

Although the earliest comparative psychologists recognized learning mechanisms as evolutionary adaptations, the rise of general process learning theory in Behaviorism effectively obliterated consideration of the adaptive functions of learning. With the accumulation of empirical results indicating biological 'constraints' on learning and theoretical arguments for the necessity of innate structure for learning, general process learning theory was gradually abandoned. Yet only recently has an alternative model, ecological learning theory, emerged, to restore emphasis on the adaptive functions of learning.

3.1 EVOLUTIONARY COMPARATIVE PSYCHOLOGY BEFORE BEHAVIORISM

Few modem learning theorists are aware of how evolutionary-minded the earliest comparative psychology was (see Boakes, 1984). Darwin was well-versed in associationist learning theory; Pavlov concerned himself with the adaptive functions of learning when to salivate (Garci y Robertson & Garcıa, 1988). Klıne (1898), Small (1900), and even the early Thorndıke (1911) and Watson (1919) recognized the importance of animals' intrinsic organization for learning. However, these early attempts to understand learning as an evolutionary adaptation were derailed by the commitment of Romanes and other comparative psychologists to Lamarckian inheritance of learned abilities and to a phylogenetic continuity of mind from simpler to more complex organisms that was taken proximally from Spencer's (1855) *Principles of Psychology* but derived ultimately from Aristotle's 'Great Chain of Being'. Thorndike (1898) correctly rejected both of these doctrines, and "viewed the work of his predecessors, studying learning within an evolutionary framework, as unmitigated failure" (Galef, 1988, p. 55). Unfortunately, Thorndike appears to have been unaware of attempts by Baldwin, Osbom, and Morgan to propose alternative conceptions of evolution, inheritance, and learning that might have served as a better foundation for an evolutionary comparative psychology of learning. In rejecting evolutionary thinking, Behaviorists believed they no longer had to consider the specific niches and adaptive problems facing each species, and their doctrines of equipotentiality (equality of all stimuli and responses with respect to their associability) and of the cross-species universality of learning principles gained temporary hegemony.

3.2 BEHAVIORIST LEARNING THEORY

The emerging Behaviorist paradigm viewed learning as a general mechanism for crafting behavioral order out of initial neurological chaos. Morgan (1896) wrote of learning as sculpting directed activity out of initially random movements; Hull (1943) viewed infants as bundles of reflexes generating random movements that slowly get tuned by conditioning to yield adaptive behavior; Skinner (1953) viewed learning as the differential reinforcement of initially undifferentiated behavior. (Parallels to connectionist learning theory should be obvious.) Empiricism, which was initially a theory of knowledge, became first a theory of perception, and then, only with the rise of Behaviorism, a theory of behavior (Bolles, 1988). No one actually believed animals' minds started as *tabula rasae* in Aristotle's strict empiricist sense, but most Behaviorists held that a few basic reflexes and motivations, in conjunction with the ability to form conditioned stimulus-response associations, sufficed to explain all behavior. Species differences in niche and lifestyle were ignored (e.g. Thomdike, 1911). The simple, almost 'atomic' stimulus-response bond was taken as the basic unit of behavioral analysis in an attempt to emulate the theoretical style of physics and make psychology into a "purely objective experimental branch of natural science" (Watson, 1914/1967, p. 1) Behaviorism's central features (as compiled by Davey, 1989) were the principle of instrumental reinforcement (with 'reinforcement' defined circularly -- see Meehl, 1950), the principle cf stimulus generalization (theoretically impossible without innate structure -- see Shepard, 1987a,b), temporal contiguity as the prime determinate of association strength (demonstrably false -- see Rescorla & Wagner, 1972), and the equipotentiality of stimuli and responses (also demonstrably false -- see the next section).

3.3 EMPIRICAL RECOGNITION OF 'BIOLOGICAL CONSTRAINTS' ON LEARNING

Some disenchantment with Behaviorist learning theory came from psycholinguistic studies of human language acquisition (e.g. Chomsky, 1957), and from ethological studies of learning in natural settings, which focused on life-cycle patterns of learning, the plasticity of natural behavior, and the dynamics of natural patterns in real environments (e.g. Lorenz, 1937; Tinbergen, 1951), However, the strongest challenges to general process learning theory came from within the Behaviorist tradition of animal learning studies itself. In the 1960's and 1970's, a growing number of studies demonstrated biological 'predispositions' and 'constraints' in learning, which challenged the Behaviorist doctrines of equipotentiality and the universality of learning principles across species. Seligman & Hager (1972) and Hinde (1973) review this literature, which includes demonstrations of conditioned food aversion that violates the principle of contiguity (e.g. Garcıa & Koellıng, 1966), animal 'misbehavior' and 'instinctive drift', where instinctive behaviors eventually dominate operantly-conditioned behaviors (e.g. Breland & Breland, 1961); autoshaping, where animals perfect skills without reinforcement (e.g. Williams & Williams, 1969); spontaneous maze learning in rats (Brant & Kavanau, 1965); reward-specific association biases (Jenkins, 1984); species-specific defense reactions (Bolles, 1970, Bolles & Fanselow, 1980); animal's superiority at learning natural over artificial concepts (Hernstein, 1979), preparedness in human phobia learning (Ohman, Dimberg, Ost, 1985); and imprinting to parents (Lorenz, 1952), enemies (Curio & Vieth, 1978), and potential mates (Marler, 1984). In addition, Behaviorists were beginning to recognize that, although they explicitly defined learning in terms of experimental paradigms, they too often were defining it implicitly in terms of highly artificial lab equipment tailored to the constraints and predispositions of each 'animal model' species (Timberlake, 1989) There were 'tricks' to conditioning and shaping (i.e. in selectively avoiding species-typical behaviors) and to setting up lab situations to overcome or circumvent the mechanisms actually used in complex natural learning (Shettleworth, 1984).

Although the empirical proof of biological 'constraints' on learning was overwhelming by the mid-1970's, animal learning theorists simply didn't know what to make of these 'anomalies'. The burden of proof was always against species-specific, domain-specific adaptive learning mechanisms, and for completely general learning principles (Revusky, 1977). But Shettleworth (1984) observed that, from an evolutionary point of view, this bias was absurd, since selective association makes learning more adaptive than it otherwise would be. The burden of proof should, evolutionarily, be against domain-generality and against cross-species universality.

Consternation and confusion accumulated in the animal learning literature, threatening the hegemony of Behaviorist doctrine. Cognitive psychology (which had rejected Behaviorism's philosophy of science and methods of research even as it retained Behaviorism's doctrine of equipotentiality and anti-evolutionary bias) responded by walling itself off from animal learning theory. Where cognitive psychology could not ignore the infiltration of biological 'constraints' into animal learning research, at least it could (and did) contain the damage by maintaining human memory and learning as a separate, pristine field: "the prevailing view of human learning is [still] that it is almost wholly general-purpose in character and can be understood without reference to biological or ecological considerations" (Estes, 1984, p. 626).

3.4 THEORETICAL RECOGNITION OF THE NEED FOR 'BIOLOGICAL CONSTRAINTS'

In fairness, there was some recognition in human cognitive psychology of the need for intrinsic structure to guide learning. Following on Kant's attempt to solve Hume's problem of inferring causality from temporal succession, Michotte (1954) explored the mind's inherent tendencies to organize perceptual experiences in terms of causal relations. Chomsky (1957) argued that intrinsic structure (e.g. a 'Language Acquisition Device') was necessary for learning grammar given the supposed paucity of linguistic input to children. Shepard (1987a,b) maintained that any cognitive system that lacks innate structure in the 'psychological spaces' in which it organizes perceptual experiences would have no basis for generalizing adaptively to new situations. In general, "nontrivial self-programming can take place only if sufficient knowledge about the world in which the system is to learn is already built into the system" (Shepard, 1989, p. 106). Further, learning principles do not emerge directly from the dynamics of replicating systems in the way the principles of natural selection do. Learning is not a self-organizing adaptive process as evolution is: "the principles that govern learning cannot themselves be learned" (Shepard, 1989, p. 106). Of course, arguments for innate constraints were never very surprising to evolutionary biologists. Emlen (1973), for instance, argued that postulating learning without genetic guidance means postulating the evolution of a mechanism that would allow an animal to arbitrarily change its phenotype without regard to its fitness consequences -- an evolutionary implausibility.

3.5 FROM BIOLOGICAL CONSTRAINTS TO ECOLOGICAL LEARNING THEORY

The accumulation of theoretical arguments and experimental evidence for biological 'constraints' in animal learning did not result in the rapid overthrow of general process learning theories, largely because there was not, until recently, an alternative theoretical framework for understanding learning. Describing selective associations and species-specific or domain-specific learning mechanisms in terms of 'predispositions' and 'constraints' in itself reveals the continuation of a general process view of learning. Constraints could be seen as anomalous biological intrusions on an otherwise normative *tabula rasa* (Shettleworth, 1984): "the implications of the term ['constraints'] ... is that animals would be smart if their genes did not constrain their general ability to learn and thereby make them selectively stupid" (Gould & Marler, 1984, p. 254). Timberlake (1989) suggests that predispositions and constraints are evolutionary outcomes to be explained, not explanations in themselves for failures of general process learning theory, and Revusky (1977) holds that viewing learning in terms of constraints is misleading in that it has fostered blindly empirical investigations of limits on general process learning theory in ignorance of relevant ethological information about animals' niches and behavior.

Research on biological constraints became a simple catalog of anomalies and puzzles without integrative or predictive power, and did not lead to development of a contemporary integrated theory of learning (Davey, 1989). As Cosmides and Tooby (1987) point out, almost any complex adaptation can be viewed in terms of 'constraints' (e.g. a bird's wings 'constrain' its ability to swim), but most adaptations are better understood as 'enablers' (e.g. a bird's wings *do* enable it to fly). Moreover, the attempt to view constraints as biological boundary conditions on general laws (e.g. Logue, 1979) has not proved fruitful, nor can selective association be accommodated in general process learning theory by adding more parameters, e.g. a scalar 'preparedness' value for every possible stimulus-response association (see Seligman, 1970).

Biological constraints research eventually crippled general process learning theory, but left a huge theoretical gap. Only recently has ecological learning theory (Davey, 1989) risen to take its place. Ecological learning theory's strategy is to start, like all adaptationist accounts in evolutionary biology and behavioral ecology, with consideration of what adaptive functions might be served by the biological structure or process in question -- in this case, the processes of learning. In this view, learning theory must be linked to a consideration of adaptive pressures bearing on the evolution of learning mechanisms, and of the phylogenetic resources (e.g. 'preadaptations') available for constructing learning mechanisms (Timberlake, 1989). Just as evolution does not 'build down' bodies or organs from more general-purpose designs, evolution would not be expected to somehow 'constrain' general-purpose learning principles. We might rather expect evolution to generate specific learning mechanisms attuned to particular ecological problems. Among the central tenets of ecological learning theory are: recognizing learning mechanisms as evolutionary adaptations (Dawkins, 1983; Shettleworth, 1983); recognizing the possible biological utility of learning (Kamil & Roitblat, 1985; Lea, 1984); considering the ecological problems facing organisms, (Plotkin & Odling-Smee, 1979); attending to relevant ethological information (Johnston, 1981a); taking an evolutionary view of reinforcement (Vaccarino & Glickman, 1989); appreciating that animals are often rather specifically adapted to their niche (Slobodkin & Rapoport, 1974); and understanding the evolution of learning in the context of already-functioning behavioral systems (Mayr, 1974).

3.6 TOWARDS UNDERSTANDING THE ADAPTIVE FUNCTIONS OF LEARNING

An evolutionary and ecological perspective on learning gives rise to very different questions, not only about the proximate mechanisms of learning (what the Behaviorists investigated almost exclusively), but about the ultimate adaptive functions of different learning mechanisms. For example, Davey (1989, p. xiv) asks "What is the biological function of learning? How does it contribute to inclusive fitness? What selection pressures bear on the evolution of learning processes? Could generalized learning processes ever be selected for? Have basic learning processes evolved separately in different species?" Questions of adaptive functions ('why?') have logical priority over analyses of proximate mechanisms ('what?' and 'how?') for any evolved biological system (Davey, 1989), but adaptive mechanisms and adaptive functions do illuminate each other. Analyses of proximate learning mechanisms alone cannot constitute a complete psychological learning theory: "The common belief that 'learning' is an alternative to an evolutionary theory of adaptive function is a category error. Learning is a cognitive process. An adaptive function is not a cognitive process. It is a problem that is solved by a cognitive process" (Cosmides & Tooby, 1987, p. 292). The guiding question in ecological learning theory thus becomes "what kind of learning mechanisms would natural selection have produced?" (ibid.).

Ecological learning theory suggests that we answer this question first by considering what specific kinds of ecological problems might be solved by the evolution of learning mechanisms. Among the central problems facing terrestrial organisms are finding food, finding mates, allocating reproductive effort, caring for offspring and kin, avoiding predators and parasites, and navigating through the environment. We might expect learning mechanisms to be organized around these adaptive problems in the context of *behavioral systems* (Timberlake, 1989), *Darwinian algorithms* (Cosmides & Tooby, 1987), or *psychological adaptations* (our preferred term), each containing cognitive, motivational, emotional, volitional, learning, and memory components. For example, learning has been investigated specifically as an aid to foraging behavior (e.g. Lea, 1984; Staddon, 1980). These considerations support points (1) and (2) advocated in section 1.

4 HOW ECOLOGICAL LEARNING THEORY CAN INFORM CONNECTIONIST LEARNING THEORY

Behaviorists attempted to see how much of real human and animal behavior they could explain just by reference to general principles of learning interacting with environmental contingencies and conditioning paradigms. Connectionists all too often attempt to see how much of human mental life can be explained just by reference to general principles of learning interacting with the 'statistical regularities of the environment'. Arguments from parsimony can be dangerous. The history of learning theory in comparative psychology indicates that in both Behaviorism and Connectionism the burden of proof against evolved psychological adaptations has been misplaced. Evolutionary considerations suggest that we should reverse this traditional burden of proof and assume that most psychological adaptations will include either no learning mechanisms, or very finely-tuned learning mechanisms with quite specific functions, e.g. to promote experience-sensitive development of behavior-generators, to track changes in body shape and size, to allow spatio-temporal integration of certain kinds of information (i.e. 'memory'), or possibly to track certain environmental dynamics. Learning is a subsidiary rather than an autonomous adaptive process, because learning mechanisms evolve to serve particular adaptive functions defined in ecological and evolutionary terms. Ecological learning theory suggests, then, that connectionist *learning* theory per se cannot serve as the core theoretical framework for connectionist modelling of psychological adaptations. Only evolutionary psychology (Cosmides & Tooby, 1987), appropriately extended and modified, can fill that role.

4.1 ECOLOGICAL LEARNING THEORY, CONNECTIONISM, AND EVOLUTIONARY PSYCHOLOGY: A HAPPY RECONCILIATION?

Evolution, learning, and computation can all be construed as *adaptive processes* (Holland, 1975). Connectionism has concentrated almost exclusively on the adaptive processes of learning and computation, but perhaps evolution could be added in as just another process at a longer time scale. This would result in a tidy kind of 'evolutionary connectionism' where every connectionist model of a human psychological adaptation would consider three adaptive processes at different time scales (as outlined by Shepard, 1989). First, at the shortest time scale, computation would allow the connectionist system to adjust its internal representations and overt responses to the requirements of the current environment, e.g. by perceptual completion, interpretation, categorization, prediction, and inference. An activation dynamics equation governs the network's relaxation in state space to fulfill the hard and soft constraints set by the current environmental input. Second, at an intermediate time scale, learning processes would adjust connection weights and biases, perhaps by gradient descent in weight space according to some connection dynamics equation, e.g. error back-propagation. Thirc' at the longest time scale, a simulated process of natural selection (e.g. a genetic algorithm) could evolve network designs by performing a stochastic search through an architecture space (e.g. Miller, Todd, & Hegde, 1989; Belew, 1990; Belew, McInerney, & Schraudolph, 1990). In the limit, we could simply 'evolve' connectionist models of psychological mechanisms in an abstract 'econiche' composed of experimental results and theoretical heuristics which the models should fit. Alternatively, consideration of the adaptive tasks that certain psychological adaptations might have been designed to solve evolutionarily could help guide modelling of those adaptations by human psychologists. According to this view, if we make our computational tasks just a little more ecologically valid, and our learning processes a little more biologically plausible, we'll have a flexible, powerful paradigm that updates connectionist modelling to fit better with ecological learning theory (Davey, 1989) and with the emerging field of *evolutionary psychology* (Cosmides & Tooby, 1987).

4.2 TURNING THE TABLES ON LEARNING

By prematurely adopting the sort of 'evolutionary connectionism' outlined above, however, we may be missing a valuable opportunity for re-conceptualizing learning itself. Evolutionary theory has traditionally been dominated by learning theory in psychology; what would happen if we momentarily inverted this dominance relation and asked: given the already powerful adaptive process of evolution by natural selection, what could learning really add?

Johnston (1981b) analyzed the relative costs and benefits of learning from an evolutionary perspective, and concluded that learning is not always an adaptive thing to have. Fitness costs of learning may include longer infancy and adolescence, with delayed reproductive maturity (as Staddon, 1983, p. 1, observes, it is "sometimes better to be dumb and fast than intelligent and slow"), increased juvenile vulnerability during learning, increased parental investment during learning, the neural 'bookkeeping' cost associated with memory storage and the possibly greater connection complexity and density required for learning, and, perhaps most importantly, the developmental fallibility of learning: "the importance of *not* learning maladaptively is underestimated" (Shettleworth, 1984, p. 448) In particular, not learning the wrong things at all may be more important than learning the right things quickly (Revusky, 1984). Proposed benefits of learning include being able to adapt to changes and fluctuations in the environment, particularly when the environment may change unpredictably during the animal's lifetime (Slobodkin & Rapaport, 1974; Plotkin & Odling-Smee, 1979), and being able to exploit new niches when required (Davey, 1989).

All adaptive costs and benefits must be understood relative to evolutionarily available alternatives. There may be no a priori need for learning to evolve if other psychological mechanisms suffice to generate adaptive behavior. Why insert an intermediate adaptive process (learning) between the evolution of psychological adaptations and the online functioning of those adaptations in generating behavior contingent on current environmental input? Hardwiring may suffice. Descartes (1662/1972), for example, ignored learning and viewed all animal behavior as reflexive responses to current environmental events. Davey (1989, p. xiii) observed "it is surprising how relatively few species have abandoned fixed behavioral patterns in favor of learning abilities" , and Mayr (1974, p. 652) noted "considering this great [supposed] advantage of learning [i.e. adaptability to changing environments], it is rather curious in how relatively few phyletic lines genetically fixed behavior patterns have been replaced by the capacity for the storage of individually acquired information".

The prevalence of hardwiring among terrestrial organisms has several explanations. Staddon (1983, p. 1) suggests "direct stimulus-response mechanisms, plus some sensitivity to rates of change, are sufficient for a wide range of surprisingly intelligent behavior" (see also Braitenberg, 1984). This is particularly true for small, fast-breeding organisms, whose short generation time facilitates rapid evolutionary change in response to environmental change and limits how much time they have to exploit learned information during their lifespan. Staddon (1983, p. 395) also remarks "the longer an animal's life span, and the more varied its niche, the more worthwhile it is to spend time learning". He goes on to note that, given most animals have rather short life spans, "It is not surprising, therefore, that learning pays a rather small part in the lives of most animals" (ibid.). Moreover, "animals in invariant environments can rely on equally invariant patterns of behavior" (Mackintosh, 1987, p. 336).

Some science seeks to reduce the strange to the familiar; our goal in this section has been the reverse. We sought in turning the tables on learning to make an apparently commonsense adaptive process seem strange and problematic from an evolutionary point of view. Perhaps by turning up the heat of evolutionary theory on the cauldron of learning theory, we can perform the kind of theoretical annealing that has been so successful in other areas of adaptationist biology. One of the hottest questions to ask becomes, not why have any biological constraints on learning evolved, but why isn't all initially 'learned' behavior canalized into genetically hardwired psychological adaptations? What can a few years of learning really buy a cognitive system already fine-tuned by millenia of natural selection? Whereas connectionists have taken learning as the ultimate adaptive process, real evolving organisms always have an alternative: hardwire the knowledge.

5 TOWARDS A TAXONOMY OF ADAPTIVE FUNCTIONS FOR LEARNING

The most salient aspect of learning to Behaviorists was its dynamic ability to bring organismic behavior into a better fit with the current conditions of the environment. Learning was, quite intuitively, assumed to have a kind of environment-tracking function. It was a way for organisms to adapt to environmental changes faster than evolution could. This view carries over into many recent justifications of learning as an adaptive adjunct to natural selection (e.g. Belew, 1990). But we still wondered why an organism would evolve to allow environmental conditions to change how its behavior-generating mechanisms work (by 'learning'), rather than allowing natural selection to optimize those mechanisms (by 'hardwiring') just as its has optimized so many other physical adaptations (see Mayr, 1974; Staddon, 1983; Menzel, 1984)? How can we clarify and extend these intuitions about the evolution of learning?

We see three main adaptive functions for learning. First, and perhaps most importantly, 'learning' may serve to increase an organism's 'developmental leverage', allowing it to build a larger, more complex, more finely organized phenotype than it otherwise could, given a certain size genotype. Sensitivity to certain predictable environmental regularities during neural development, and the resulting sensory activation patterns, could guide the self-organization of an animal's behavior-generating mechanisms (e.g. see Singer, 1988). Learning may allow the genotype to 'store information in the environment' and let environmental regularities do much of the hard work of wiring up adaptive behavior-generators.[2] The environmental regularities used in this way may take a rather abstract form. For example, parental 'imprinting' in birds (Lorenz, 1937) can be viewed as a way of building a behavior-generator sensitive to the appearance of one's parent, based on the following environmental regularity: the first large moving thing one sees after hatching is very likely one's parent. Of course, the particular behavior-generator constructed by different birds for recognizing their parents will be different (the birds will 'learn' different parental images), but the species as a whole relies on the same environmental regularity when doing the construction.

Second, 'learning' construed as 'memory encoding' can assist in the spatio-temporal integration of environmental information. Behavior-generators guided only by environmental cues in the here and now may be inferior to behavior-generators sensitive the relevant environmental cues from the distant and past.[3] Animals may evolve to construct the functional equivalent of variable-delay neural delay lines (i.e. 'episodic memories') from certain sensory systems to certain behavior-generators, to expand the temporal scope of their sensitivity to environmental cues. That is, they may evolve to be able to 'bring to mind' information they recruited in the past that is not currently available in the environment. For mobile animals, broadened temporal sensitivity can translate into broadened spatial sensitivity as the animal moves about and recruits environmental information. (Constructing a

[2] There have been many demonstrations that experimentally depriving developing nervous systems of certain environmental regularities (e.g. certain kinds of visual stimulation) results in maladaptively organized topographic maps (e.g. in striate visual cortex -- see Hubel & Wiesel, 1965, Wiesel & Hubel, 1965) Such experiments should not be construed as demonstrations of the *impotence* of genetic programming and the *importance* of 'experience' and 'learning', but of how efficient evolved developmental mechanisms are at recruiting environmental regularities to assist in self-organization. The fact that neurologists can selectively eliminate those regularities in laboratories should not make us doubt their reliability and ubiquity in nature.

[3] However, Gibson (1966, 1979) and Johansson, von Hofsten, and Jansson (1980) warn against underestimating the informational richness of the present, proximal environment, or underestimating the range of adaptive behavior that it can guide

'mental map' of one's environment from experience gathered during sequential exploration is a paradigmatic example of using 'learning' to integrate information across space and time.) This function of learning neither constructs behavior-generators (function 1 above) nor modifies their online functioning (function 3 below), but simply expands the range of environmental information to which they are sensitive.

Third, as suggested earlier, 'learning' may allow organisms to adjust the online functioning of their behavior-generators faster than natural selection would allow, by conferring on those behavior-generators some sensitivity to dynamic changes in environmental conditions during an organism's life. In this case, learning adapts phenotypes to ongoing changes and *particularities* of the environment rather than depending on environmental *regularities* during phenotype-construction. For example, animals equipped with a special mechanism for learning motor skills to remove new varieties of parasites from their bodies may fare better than competitors lacking such a mechanism if a new species of parasite migrates to the area. Different parasites may have different modes of attachment to the animals' bodies, so may require different removal methods. A learning mechanism that allowed an animal to infer, practice, and perfect an appropriate removal method given, e.g., a visual assessment of the parasite's attachment method, might prove adaptive. In this case, the animals would not be using an environmental regularity to construct a behavior-generator during development (the new parasites weren't around then), or integrating information about the parasites across space and time (parasites currently attached to the animals' bodies are very much in the here and now), they are simply developing a new behavior-generator (e.g. a new method of parasite removal) adapted to a new environmental problem.

This proposed environment-tracking function includes many sub-functions not usually considered in learning theory. One reason for modifying the operation of behavior-generators during the lifetime of an animal is that the animal's body may be growing and changing -- i e the spatial relations among its sensory transducers and motor effectors, and between those and its surrounding environment, may be changing. Certain learning mechanisms may evolve to track such relations between the animal's gradually changing body and relatively stable aspects of the environment, rather than tracking 'objective' environmental dynamics per se. Another way of expressing this would be to say learning tracks not just the external environment, but also the corporeal environment of bone, sinew, and flesh in which the animal's behavior-generators (i.e. brains) are embedded.

A second reason for modifying the operation of behavior-generators in response to certain environmental conditions is that animals may need to track changes not only in their individual bodies (their basic phenotypes) but in their 'extended phenotypes' (Dawkins, 1982), including the location, health, and reproductive status of their kin and offspring. Animals may be able to rely on relatively fixed developmental sequences and internal clocks to modify the operation of their behavior-generators as they grow and age. But to track the whole of their extended phenotype, they need to actually observe when other copies of their genes are being instantiated (i.e. when kin and offspring are born), when those copies are gaining access to metabolic and genetic resources (i.e. growing, eating, and mating), and when they are being threatened (i.e. injured or dying). Animals may evolve learning mechanisms that permit acquisition and maintenance of an ongoing cognitive model of one's kin and social-exchange network, including how to recognize and assist them, and how to request assistance of them. Many learning mechanisms may be rather specifically tuned to promote this type of kin recognition and kin selection (see Hamilton, 1964).

Such complexities aside, environment-tracking is probably the adaptive function of learning most familiar to adult humans (e.g. learning a new restaurant location, or a new person's name), so it has been more commonly studied by psychologists. Yet it is likely to be a less common use of learning than experience-guided phenotype-construction throughout the animal kingdom. Several issues of scale in time and space arise in considering the adaptive functions of learning; these can illuminate why experience-guided development may be more widespread than environment-tracking learning. We might expect small and large organisms to use experience-guided development to almost the same extent, i.e. as much as possible. Big animals, by definition, must build large and often incredibly complex phenotypes given moderately sized genotypes. Although small animals do not have as much of a phenotype to build, they must build their phenotypes from very small gametes that contain very little genetic material.

However, larger phenotypes generally take longer to build, implying longer generation time and a slower rate of genetic evolution (the greater extinction rate of large species is generally thought attributable to their difficulty in genetically adapting to changing environments). Thus, larger phenotypes will generally have a harder time tracking environmental change, and their genotypes will generally lag farther behind being adapted to the current environment, so in general there may be unusually strong adaptive pressures for larger animals to evolve environment-tracking learning mechanisms. The salience of these adaptive pressures to humans, one of the larger and longer-lived species on the planet, may lead us to overestimate the importance and popularity of environment-tracking learning. But, as Davey (1989, p. 20) suggests, "learning ... is likely to evolve only when more fundamental processes of information gain (such as phylogenesis [i.e. evolution]) have reached an upper limit to the amount or rate of change that they can cope with".

Thus, we needn't suppose that small animals lack environment-tracking learning abilities because they are 'less advanced' than larger animals. We must see 'learning' as (1) as way of developmentally fleshing out the genotype during development, (2) informationally fleshing out the animal's current *Umwelt* (view of the world) over space and time, and (3) temporally filling in the adaptive function of environment-tracking in between generation times, when natural selection operates. Small, fast-lived organisms simply don't need much of the third function. Since small-bodied species as collective entities out-number, out-weigh, out-reproduce, and typically out-last large-bodied species, they objectively instantiate most of the adaptive agency on this planet. The adaptive functions of learning that are central to them must be considered central to psychology as the (non-anthropocentric) study of adaptive agency.[4]

6 A SIMULATION FRAMEWORK FOR EXPLORING ADAPTIVE AGENCY

A general understanding of adaptive agency cannot rest on experimentation, observation, and theory alone. Some adaptive processes happen on time-scales that preclude experimental manipulation or direct observation. Also, the canalization of terrestrial evolution along certain lines (e.g. using DNA for genotype material) makes us wonder about alternative possibilities not directly observable. And the stochastic, complex nature of adaptive processes make strict experimental control of real physical systems very difficult. For these reasons, researchers have recently turned to computer simulation as the most tractable way of exploring certain adaptive processes. Simulation allows strict control over specified parameters, exploration of alternate phylogenies and developments, and rapid observation of processes that take eons in the real world. Simulation bears on psychology not only as a method for modelling specific psychological adaptations and specific historical phylogenies, but as a way of generally exploring the adaptive processes that produced those phylogenies and adaptations. However, the facility with which such simulations can be developed and explored, and the inherent appeal of watching adaptive systems develop, can allow the proliferation of studies not well-grounded in a theoretical framework. Without theoretical grounding, results are difficult to interpret and to assimilate into a coherent picture. We hope the notion of psychology as the study of adaptive agency can help to unify and direct all such studies (observations, experiments, and simulations).

To capture all of the adaptive processes discussed earlier in a simulation, we must have methods of simulating genotypic evolution, the generation of behavior, and the ability to learn new behaviors. (We ignore 'cultural' transmission for now, but see Belew, 1990). Specifically, we use a genetic algorithm to evolve successive generations of a population of neural network architectures, which in turn control the behavior of simple creatures which can learn as they live in a simulated environment. This is an extension of earlier work exploring the use of genetic algorithms to design network architectures capable of learning specific input/output mapping tasks (Miller, Todd, and Hegde, 1989); here, the algorithm's measure of fitness depends not on learning an arbitrary task, but on behaving adaptively in the simulated environment.

In simulating the evolution of further adaptive processes, for instance specific behavior-generators and specific learning abilities, we must first specify some environment and what defines fitness in that environment. We then observe, through the course of the evolutionary simulation, which adaptive processes are most important for maximizing individual fitness by "solving" the relevant environmental problems. Ackley and Littman's (1990) sophisticated simulations, for example, show the utility of evolving motivational systems to guide learning. Nevertheless, their simulations are pre-set to operate either with learning or without, rather than set up with an environment whose adaptive problems allow the evolution of learning itself to be studied. We strive to create simulations in which the subsidiary adaptive processes that evolution can spawn are as open-ended as possible.

6.1 TYPES OF LEARNING

Since we are primarily interested in using simulation to explore issues in the evolution of learning, we must address the varieties of learning we could investigate. Tolman (1932) asked "Is there more than one type of learning?" The answer, of course, depends on what one means by 'type'. Different kinds of learning might be distinguished by at least three different criteria. (1) type of experimental paradigm used to investigate it; (2) neural mechanism implementing it; (3) adaptive functions served by it. Behaviorists concentrated on the first criterion, defining classical versus operant conditioning, for example, according to the sorts of environmental contingencies the experimenter sets up for the laboratory an-

[4] The suggested logical and historical primacy of experience-dependent development over environment-tracking learning is a conjecture not easily supported or refuted. It is appealing to us because learning mechanisms that evolved originally for experience-guided development of behavior-generators could conceivably serve as pre-adaptations for the evolution of environment tracking learning. That is, given a learning mechanism that recruits patterns of environmental stimulation to help in construction of a psychological adaptation, one could easily imagine how prolongation of that mechanism's sensitive period could confer longer, perhaps lifelong, adaptive flexibility to that mechanism. Selective pressures to prolong such sensitive periods, we conjecture, may underlie the evolution of most environment-tracking learning mechanisms. Johnston (1981) discusses how potentially adaptive concommitants of evolved learning abilities might serve as pre-adaptations for later learning abilities by conferring on animals certain "ecologically surplus abilities", i.e learning abilities not directly selected for. Rozin (1976) also suggests that learning mechanisms may have evolved first as isolated specializations and later became available to a wider range of behavior-generators.

imal. Connectionists view the first criterion in terms of structuring the training set and test set for the network, and the second in terms of the specific mathematical learning algorithm implemented in the network. Rarely has a taxonomy of learning been conceptualized in terms of adaptive function. The theoretical and empirical study of learning could concentrate on any of these three ways of distinguishing learning processes, or it could pursue a fourth strategy of attempting to elucidate general principles of operation governing all learning processes, understood in some sufficiently abstract way -- as in Shepard's (1987a) work towards a universal law of generalization.

Fortunately, there may be some correspondence between learning mechanisms construed in terms of experimental paradigm used to investigate them and learning mechanisms construed in terms of their adaptive functions. Learning mechanisms evolved to solve particular ecological problems; to the extent that different experimental paradigms present ecologically valid, adaptively isomorphic problems, they may map onto real adaptive functions of learning. But the fit between adaptive functions and neural mechanisms may be much looser: "analysis of learning in terms of functional problems does not map directly onto the learning theorists' analysis in terms of [neural] mechanisms" (Shettleworth, 1984, p. 431). Different adaptive functions might be implemented by similar neural mechanisms for changing synaptic weights, or the same adaptive function might be implemented in very different neural mechanisms in different species.

For these reasons, we chose to categorize learning mechanisms by adaptive function rather than by experimental paradigm or neural mechanism. But the problem remains: which adaptive function should be explored first? To investigate the evolution of learning for exploiting environmental regularities specifically as a means of maximizing phenotype size and complexity given limited genotype size, a simulation must include adaptive pressures or constraints on genotype length or specificity. Without such pressures or constraints, the genotype may simply expand to accurately specify (hardwire) the appropriate phenotype, rather than evolving developmental tricks that depend on internalizing environmental regularities during development. Clearly, such pressures depend on what developmental mechanisms exist or can evolve for generating phenotypes from genotypes. Since the vagaries of real neural development and of adaptive pressures on genotype size are still poorly understood, we have not found a satisfactory general method for simulating neural development or for imposing pressures on genotype length. So we have avoided simulating the evolution of learning as a way of achieving developmental leverage.

Exploring the third 'adapting to particularities' function of learning requires setting up an adaptive problem with environmental changes too rapid for genotypic evolution to track. But if the environment changes *too* rapidly, then extensive simulation of learning during an organism's lifespan would be required. Likewise, exploring the second information-integration function of learning would require simulating extensive interactions between creatures and worlds, including recurrent network dynamics or memory encoding and retrieval systems. We did not want to become mired in simulating constantly-changing environments or sophisticated dynamic learning mechanisms, so we opted for a kind of imprinting scenario, where experience-guided development uses an abstract environmental regularity to help build behavior-generators adapted to unpredictable particularities of the niche, as follows.

The simplest way to defeat natural selection is to make the genotype unable to know ahead of time which one of two alternate econiches it will find itself in during phenotypic development. If one econiche requires one kind of behavior-generation mechanism and another econiche requires a different kind, natural selection alone will be unable to select the proper mechanism to guide the phenotype's behavior. Natural selection must instead select for the evolution of a more general mechanism that can flip into one of two states depending on some assessment of which econiche it finds itself in. Thus, we chose a kind of "imprinting" or parameter-setting based on the early environment as the simplest possible case in which learning, construed as adaptation to specific environmental regularities, could evolve.

6.2 INCLUSIVE FITNESS AS THE ONLY NATURAL 'SUPERVISOR' FOR LEARNING

Many previous attempts to use genetic algorithms to evolve neural network architectures have evaluated architecture fitness by training the networks with a supervised learning procedure, (i.e. one with an externally provided "target vector" the network is to produce given each input vector), typically back-propagation (Belew, McInerney, & Schraudolph, 1990; Miller, Todd, & Hegde, 1989; for a review, see Weiss, 1990). While supervised learning paradigms may be appropriate in evolving connectionist systems for particular commercial applications, they are problematic and perhaps misleading in scientific studies of adaptive agency. In particular, to be biologically plausible, the source of the "targets" or other supervising feedback must be justified. Organisms as whole functioning agents in real environments rarely receive patterns of information analogous to training signals in back-propagation. Although the distinction between supervised and unsupervised learning procedures can be blurred, we have chosen to focus on the more defensible latter end of the spectrum, including self-organizing, associative, and simple feedback-based mechanisms. But even if we sidestep the issue of target-based training, the concept of feedback still raises problems.

Years of learning by "being taught" instill in us intuitions about the utility of corrective feedback to guide learning. But such intuitions make it easy to overlook the fact that it is at least as difficult for organisms to evolve the ability to perceive feedback signals from the environment to guide their learning, as it is to evolve the perception of any other complex external cues. Consider for example the complexities involved in registering the information that one has just been rebuffed in a social exchange. Feedback signals cannot be assumed to be just somehow "provided" to an organism for it to use in adjusting its behavior. Instead, feedback systems must be understood as special *sensory* systems evolved to provide information to special learning mechanisms that in turn adaptively change the functioning of certain behavior generators (e.g. in the simulations of Ackley and Littman, 1990). Feedback systems, whether motivational, emotional, volitional, or proprioceptive, evolve just like other aspects of adaptive agency -- by cumulative selection of incrementally better-adapted designs.

Ideally, an organism might prefer to guide its learning with direct information about how its inclusive fitness changes as a result of its behavior. But there is no such thing as an inclusive fitness transducer that can be used to supervise learning. Organisms must instead evolve to sense inclusive fitness indirectly, through whatever proximal sensory cues have been reliably associated with increased fitness in their environment. Thus, natural selection itself is ultimately the *only* source of 'supervision' for learning systems.

The indirectness of natural selection's supervision of learning leads to the complexity inherent in real evolved learning mechanisms. Humbled by this complexity, we decided not to clutter our initial simulations with the requirement of evolving a motivational system to provide supervising feedback during learning, *in addition* to evolving the learning system itself. So instead we chose to start by exploring the simplest set of unsupervised associative learning mechanisms we could conceptualize, as will be described below.

6.3 THE GENETIC ALGORITHM FOR EVOLVING NEURAL NETWORKS

To simulate the evolution of learning in our explorations of adaptive agency, we use a relatively standard form of Holland's (1975) genetic algorithm, combined with a simple "developmental" method which translates genotypes into neural network architecture phenotypes. In this method, a strong genetic specification scheme (as defined by Miller, Todd, and Hegde, 1989) interprets each genotype as a *connectivity constraint matrix* that directly specifies the nature of each unit and connection in the network architecture.

Once a network, instantiating the behavioral mechanisms of an individual creature, has been so constructed, it is evaluated in the simulated world over several time-steps representing the creature's lifespan. During each time-step, a creature's network receives sensory input based on the current external environmental cues available, processes that input according to its architecture and current weights, generates behavior based on the activation of its output units, and changes its connection weights based on an unsupervised learning rule (e.g. Hebbian association). The effects of the creature's behavior on the world and on its own fitness are then registered, and the next time-step begins.

7 A SIMPLE SCENARIO FOR THE EVOLUTION OF UNSUPERVISED LEARNING

For our first exploration of adaptive agency, we attempted to devise the simplest, cleanest scenario in which learning could prove adaptive, focusing on a kind of imprinting function. After analyzing the building blocks needed for associative learning, we analyzed what sorts of environments might exert adaptive pressures to evolve that type of learning. Finally, we constructed an appropriate simple world to see if learning would spread through a population of simulated creatures behaving in that world. We explain the scenario used by outlining a biological metaphor that specifies the structure of the econiche and the nature of the adaptive problem. (The scenario and results are described in more detail in Todd and Miller (1990); space constraints preclude making this section much more than an overview.)

Our scenario can be imagined as an underwater realm, in which parents emit eggs randomly into two different types of feeding patches. those where food is green and poison red, or vice-versa. Each creature in this world lives a fixed lifespan, eating or ignoring food and poison at each life-step, and amassing energy which determines its eventual number of offspring in the next generation. Eating food raises energy; eating poison drains energy. Food smells sweet and poison smells sour across all creatures, but with some perceptual error rate -- the smell-sense accuracy -- determined by the turbulence of the water in this world. Food and poison each have characteristic fixed colors *within* one creature's life, but the meaning of each color varies *between* creatures, food being red for some and green for others, depending on their patch as mentioned above. The color-sense is 100% accurate. Thus natural selection can 'predict' the association between smell and object, but not between color and object -- this will be the task for learning.

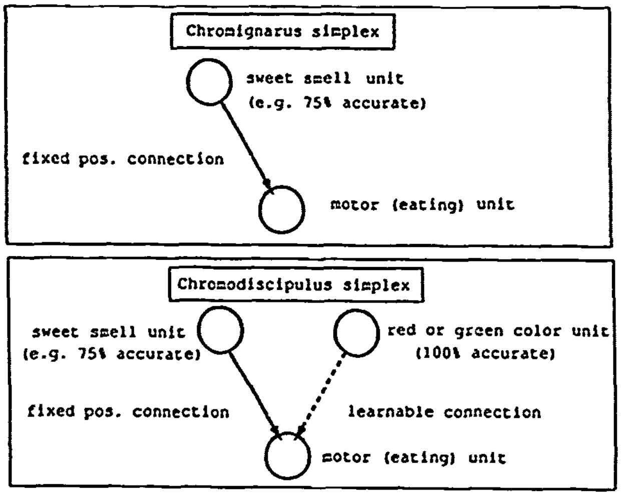


Figure 1. The network designs of *Chromignarus* (without color learning), top, and *Chromodiscipulus* (with color learning), bottom.

7.1 EVOLVING ADAPTIVE NETWORKS

As the genetic operators process genotypes again and again through many generations, recombination and mutation will sometimes produce particularly fit network designs. One reasonable design we can expect consists of a sweet smell sensing unit connected to a motor unit (which controls the creature's eating) by a fixed positive weight. A creature equipped with this nervous system will depend purely on smell to decide when to eat, with its behavioral accuracy dependent on the inherent noisiness of the smell sense in its world (i.e. by amount of turbulence). Although this design will sometimes make mistakes (that is, ignoring food or eating poison that smells wrong because of the turbulence), still on average it will eat more food than poison. Thus these creatures' energy, and number of offspring, will be higher than if they were just eating randomly. We call this design the *color-blind eater*, or, more fancifully, *Chromignarus simplex*, and it is shown in the top portion of Figure 1.

The best evolved creature design though is an elaboration on the color-blind eater chassis. Creatures of this type gain an adaptive edge by including a red or green color sensing unit, along with a learnable connection to the motor (eating) unit. This creature design we name *Chromodiscipulus simplex*, color-learning eaters, as shown at the bottom of Figure 1.

With this design, an appropriately excitatory link between the color unit and the eating unit will be built up over successive time-steps in the creature's life by a Hebbian correlational weight-change mechanism. Eventually, this weight will be large enough that the color unit alone can cause the eating unit to come on, *regardless* of what the sweet smell unit says to do -- the creature has now learned that a particular color means food. The creature can now rely on this completely accurate visual cue, rather than the inaccurate smell cue, and always choose to eat properly, thereby increasing its fitness further.

7.2 INITIAL SIMULATION RESULTS

We continued our investigation of the conditions under which learning could evolve with the more interesting question of how *quickly* learning would evolve, given various smell accuracies in different worlds. By tracking population average fitness values, it is possible to tell when the use of learning has spread through the population. Initially, the average fitness quickly rises to a plateau at which the fixed sweet-smell to eating unit connection is present in most of the creatures (*Chromignarus* temporarily dominates). After remaining fairly level at this fitness value for possibly many more generations, the average population fitness again jumps, indicating that the learnable connection from a color unit has penetrated the population (*Chromodiscipulus* ultimately rules). Average fitness then levels out again, this time around its final highest value. Thus recording when fitness jumps occur can tell us when the different creature designs predominate in the population.

For each of 17 smell accuracies between 50% and 100% we ran 20 populations of 100 individuals for 1500 generations each. Figure 2 shows how many generations it took each population to make each of the two jumps to new fitness-plateaus. These two jumps correspond to the widespread appearance of *Chromignarus* (without color learning) -- indicated by asterisks -- and of *Chromodiscipulus* (with color learning) -- indicated by bullets. The bottom curve shows the average number of generations taken to evolve *Chromignarus* across the 20 runs at each accuracy level, and the top curve indicates the analogous average generations to evolve *Chromodiscipulus*, and thus learning itself.

The fixed smell connection (the Chromignarus design) evolves rapidly, in less than 100 generations for most accuracy levels. The greater the accuracy of smell, the more quickly the fixed smell connection spreads, because the adaptive advantage to be gained from evolving it (i.e. the adaptive pressure) increases. More interesting is the effect of smell accuracy on time taken to evolve color learning (*Chromodiscipulus*). Here we found an unexpected U-shaped relationship. color learning evolved most quickly for smell accuracies around 75%, and took longer and longer for accuracies diverging on either side of that middle range (as shown by the upper, solid, curve in Figure 2).

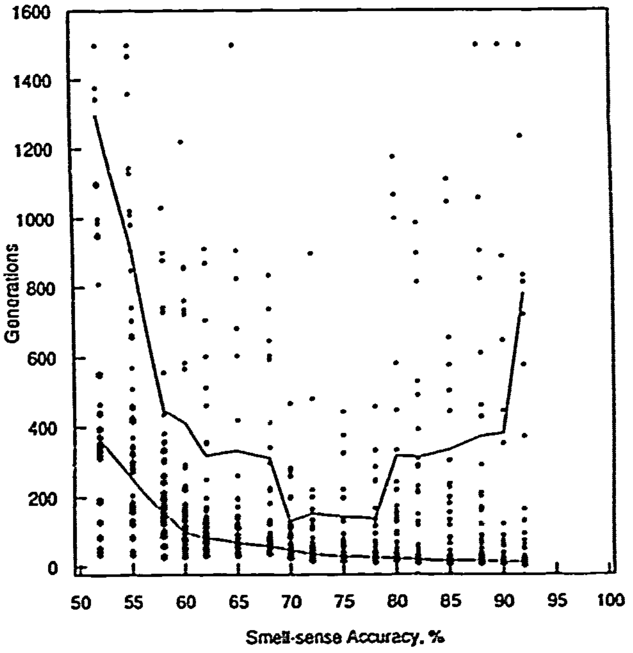


Figure 2. Generations to evolve *Chromignarus* (without color learning) and *Chromodiscipulus* (with learning) plotted against smell-sense accuracy. Asterisks indicate time to evolve *Chromignarus* for each of 20 runs at each of 17 smell-sense accuracy levels; the dotted line indicates average time across the 20 runs at each smell accuracy. Bullets indicate time to evolve *Chromodiscipulus* for the same 20 runs at each accuracy level, the solid line indicates average time across the 20 runs at each accuracy. Note that average time to evolve *Chromignarus* decreases monotonically as smell sense accuracy increases, but average time to evolve *Chromodiscipulus* follows a U shaped Learning Evolution Curve.

7.3 THEORETICAL INTERPRETATION OF THE U-SHAPED EVOLUTION FUNCTION

We view the U shape as emerging indirectly from a trade-off between the phylogenetic adaptive pressure to evolve learning (during species-wide evolution), and the ontogenetic ease of learning (during each individual creature lifetime). These forces interact at the various smell sense accuracy levels as follows.

At low smell accuracies, where *Chromignarus* does quite poorly, there is great phylogenetic adaptive pressure to evolve color learning, because it would add significantly to this creature's fitness by overcoming its color-blind, smell-guided error-prone behavior. However, this large potential benefit is offset by the ontogenetic difficulty of actually accomplishing learning at noisy low smell accuracies. In fact, learning can be so slow in this case that a learning creature's lifespan may elapse before it gains any benefit from this ability. Thus the ontogenetic difficulty of learning offsets its high phylogenetic adaptiveness at low smell accuracies, and learning will take a long time to evolve.

At high smell accuracies, in contrast, color learning would be easy to perform ontogenetically, because perceived smell, color, and substance type will be highly correlated, and the associations between them could build up quickly. However, there is little phylogenetic adaptive pressure to evolve color learning in this case, because the smell sense alone suffices to guide highly adaptive eating behavior. Since natural selection cannot distinguish *Chromignarus* from *Chromodiscipulus* if they are both doing almost perfectly, this 'ceiling effect' will keep *Chromodiscipulus* from proliferating. So again color learning will take a long time to evolve.

But for middle smell accuracies, color learning is relatively adaptive and relatively easy. Color learning gives a significant fitness increase over using smell alone, and learning can occur fairly quickly, since the eating unit comes on rather more often to food than to poison. Mid-level smell accuracy represents a happy medium between phylogenetic adaptive pressure and ontogenetic ease of learning, leading to the rapid evolution of color learning and its spread through the population.

8 PLANNED EXTENSIONS AND FUTURE RESEARCH

Our theoretical motivation will continue to be the exploration of adaptive agency and interactions among adaptive processes; our methodological strategy will continue to focus on the search for simple, elegant scenarios that reveal potentially general patterns and dynamics underlying adaptive agency. Given this orientation, we have gradually abandoned our earlier ambitions to create a general-purpose system for investigating the evolution of very complex nervous systems in very complex environments. The rush to build as much biological realism as possible into our simulations as quickly as possible, can, we fear, obscure those features of simulation that make it so useful in other sciences. parametric control, replicability, conceptual clarity, ease of analysis, and speed. Thus, we hope to develop more simple scenarios that not only capture the central features of certain adaptive problems, but that can reveal unanticipated patterns and complexities.

More specifically, we intend to develop as series of slightly more complex learning scenarios to investigate how natural selection, associative learning, and environmental dynamics interact. One could imagine that, given a series of results from such scenarios, a more general theory concerning the interaction of adaptive processes might emerge - not a formalistic model in terms of dynamical systems or information theory, but a concrete understanding of the interactions among adaptive pressures, cue structures in different environments, genetic representations and operators, developmental mechan isms, learning, behavior-generation, and information-processing. Later, we intend to address the adaptive problems of foraging, communication, and protean behavior. At each step, we hope to keep our motivations for simulation closely tied to resolving theoretical issues in the study of adaptive agency, while remaining sensitive to the sorts of unanticipated phenomena, patterns, and dynamics that simulation research so often reveals. In allowing research to be guided so strongly by a clearly articulated conceptual framework, we may give up some of the immediate richness and appeal of simulation-for-its-own sake, but we hope to achieve a theoretical depth and breadth, and a connection to major issues and perennial questions, that will, we believe, be more satisfying in the end.

Acknowledgements

This research was partially supported by National Science Foundation Graduate Fellowships to both authors. Any opinions, findings, conclusions, or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

We wish to thank Dave Rumelhart, Roger Shepard, John Tooby, Leda Cosmides, David Ackley, and Rik Belew, who have influenced and guided this research along the way.

References

Ackley, D.H., & Littman, M.S. (1990). Learning from natural selection in an artificial environment. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 189-193). Hillsdale, NJ: Erlbaum.

Belew, R.K. (1990). Evolution, learning and culture: Computational metaphors for adaptive search. *Complex Systems, 4*, 11-49.

Belew, R.K., McInerney, J., & Schraudolph, N.N. (1990). *Evolving networks: Using the genetic algorithm with connectionist learning* (Tech. Rep. No. CS90-174). San Diego: University of California, Computer Science and Engineering Department.

Boakes, R.A. (1984). *From Darwin to Behaviorism.* Cambridge, UK: Cambridge University Press.

Bolles, R.C. (1970). Species-specific defense reactions and avoidance learning. *Psychological Review, 77*, 32-48.

Bolles, R.C. (1988). Nativism, naturalism, and niches. In R.C. Bolles & M.D. Beecher (Eds.), *Evolution and learning* (pp. 1-15). Hillsdale, NJ: Erlbaum.

Bolles, R.C., & Fanselow, M.S. (1980). A perceptual-defensive-recuperative model of fear and pain. *Behavioral and Brain Sciences, 3*, 291-301.

Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology.* Cambridge, MA: MIT Press/Bradford Books.

Brant, D.H., & Kavanau, J.L. (1965). 'Unrewarded' exploration and learning in complex mazes by wild and domestic mice. *Nature, 204*, 267-269.

Breland, K., & Breland, M. (1961). The misbehavior of organisms. *American Psychologist, 16*, 681-684.

Chomsky, N. (1957). *Syntactic structures.* The Hague: Mouton.

Cosmides, L., & Tooby, J. (1987). From evolution to behavior: Evolutionary psychology as the missing link. In J. Dupre (Ed.), *The latest on the best. Essays on evolution and optimality.* Cambridge, MA: MIT Press/Bradford Books.

Curio, E., & Vieth, W. (1978). Cultural transmission of enemy recognition. *Science, 202*, 899-901.

Davey, G. (1989). *Ecological learning theory.* New York: Routledge.

Dawkins, R. (1976). *The selfish gene.* Oxford, UK: Oxford University Press.

Dawkins, R. (1982). *The extended phenotype.* Oxford, UK: Oxford University Press.

Dawkins, R. (1983). Universal Darwinism. In D.S. Bendall (Ed.), *Evolution from molecules to men.* Cambridge, UK: Cambridge University Press.

Descartes, R. (1662/1972). *Treatise on man.* Cambridge, MA: Harvard University Press. Translated by T.S. Hall.

Emlen, J.M. (1973). *Ecology: An evolutionary approach.* Reading, MA: Addison-Wesley.

Estes, W.K. (1984). Human learning and memory. In P. Marler & H.S. Terrace (Eds.), *The biology of learning* (pp. 617-628). New York: Springer-Verlag.

Galef, B.G. (1988). Evolution and learning before Thorndike: A forgotten epoch in the history of behavioral research. In R.C. Bolles & M.D. Beecher (Eds.), *Evolution and learning* (pp. 39-58). Hillsdale, NJ: Erlbaum.

Garcia, J., & Koelling, R.A. (1966). Relation of cue to consequence in avoidance learning. *Psychonomic Science, 4*, 123-124.

Gibson, J.J. (1966). *The senses considered as perceptual systems.* Boston: Houghton-Mifflin.

Gibson, J.J. (1979). *The ecological approach to visual perception.* Boston: Houghton-Mifflin.

Gluck, M.A., Bower, G.H., & Hee, M.R. (1989). A configural-cue network model of animal and human associative learning. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 323-332). Hillsdale, NJ: Erlbaum.

Gould, J.L., & Marler, P. (1984). Ethology and the natural history of learning. In P. Marler & H.S. Terrace (Eds.), *The biology of learning* (pp. 47-74). New York: Springer-Verlag.

Hamilton, W.D. (1964). The evolution of social behavior I and II. *Journal of Theoretical Biology, 7*, 1-52.

Herrnstein, R.J. (1979). Acquisition, generalization, and discrimination reversal of a natural concept. *Journal of Experimental Psychology: Animal Behavior and Processes, 5*, 116-129.

Hinde, R.A. (1973). Constraints on learning: An introduction to the problems. In R.A. Hinde & J. Stevenson-Hinde (Eds.), *Constraints on learning. Limitations and predispositions.* London: Academic Press.

Hinton, G.E., & Nowlan, S.J. (1987). How learning guides evolution. *Complex Systems, 1*, 495-502.

Holland, J. H. (1975). *Adaptation in natural and artificial systems.* Ann Arbor, MI: University of Michigan Press.

Hull, C.L. (1943). *Principles of behavior: An introduction to behavior theory.* New Yoik: Appleton.

Jenkins, H.M. (1984). The study of animal learning in the tradition of Pavlov and Thorndike. In P. Marler & H.S. Terrace (Eds.), *The biology of learning* (pp. 89-114). New York: Springer-Verlag.

Johansson, G., von Hofsten, G., & Jansson, G. (1980). Event perception. *Annual Review of Psychology, 31*, 27-63.

Johnston, T.D. (1981a). Contrasting approaches to a theory of learning. *Behavioral and Brain Sciences, 4*, 125-173.

Johnston, T.D. (1981b). Selective costs and benefits of learning. In J.S. Rosenblatt, R.A. Hinde, C. Beer, & M.C. Busnel (Eds.), *Advances in the study of behavior.* New York: Academic Press.

Kamil, A.C., & Roitblat, H.L. (1985). The ecology of foraging behavior: implications for animal learning and memory. *Annual Review of Psychology, 36*, 141-169.

Kline, L.W. (1898). Suggestions toward a laboratory course in comparative psychology. *American Journal of Psychology, 10*, 399-430.

Langton, C.G. (Ed.). (1989). *Artificial life.* New York: Addison-Wesley.

Lea, S.E.G. (1984). *Instinct, environment, and behavior.* London: Methuen.

Logue, A.W. (1979). Taste aversion and the generality of the laws of learning. *Psychological Bulletin, 86*, 276-296.

Lorenz, K.Z. (1937). The companion in the bird's world. *Auk, 54*, 245-273.

Lorenz, K.Z. (1952). The past twelve years in the comparative study of behavior. In C.H. Schiller (Ed.), *Instinctive behavior.* New York: International Universities Press.

Mackintosh, N.J. (1987). Learning. In D. McFarland (Ed.), *The Oxford guide to animal behavior* (pp. 336-346). New York: Oxford University Press.

Marler, P. (1984). Song learning. Innate species differences in the learning process. In P. Marler & H.S. Terrace (Eds.), *The biology of learning* (pp. 289-310). New York: Springer-Verlag.

Mayr, E. (1974). Behavior programs and evolutionary stiategics. *American Scientist, 62*, 650-659.

Meehl, P.E. (1950). On the circularity of the law of effect. *Psychological Bulletin, 47*, 52-75.

Menzel, R. (1984). Biology of invertebrate learning (Group Report). In P. Marler & H.S. Terrace (Eds.), *The biology of learning.* Berlin: Springer-Verlag.

Michotte, A. (1954). *La perception de la causalite,* (2nd ed.). Louvain: Publications Universite de Louvain.

Miller, G.F., Todd, P.M., & Hegde, S.U. (1989). Designing neural networks using genetic algorithms. In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 379-384). San Mateo, CA: Morgan Kaufmann.

Morgan, C.L. (1896). *Habit and instinct.* London: Edward Arnold.

Ohman, A., Dimberg, U., & Ost, L.-G. (1985). Animal and social phobias: Biological constraints on learned fear responses. In S. Reiss & R.R. Bootzin (Eds.), *Theoretical issues in behavior therapy.* New York: Academic Press.

Plotkin, H.C., & Odling-Smee, F.J. (1979). Learning, change, and evolution: An enquiry into the teleonomy of learning. *Advanced Studies of Behavior, 10*, 1-41.

Rescorla, R.A., & Wagner, A.R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A.H. Black & W.F. Prokasy (Eds.), *Classical conditioning II: Current research and theory.* New York: Appleton-Century-Crofts.

Revusky, R. (1977). Interference with progress by the scientific establishment: Examples from flavor aversion learning. In N. Milgram, L. Krames, & T. Alloway (Eds.), *Food aversion learning.* New York: Plenum.

Garcia y Robertson, R., & Garcia, J. (1988). Darwin was a learning theorist. In R.C. Bolles & M.D. Beecher (Eds.), *Evolution and learning.* Hillsdale, NJ: Erlbaum.

Rozin, P. (1976). The evolution of intelligence and access to the cognitive unconscious. *Progress in Psychobiology and Physiological Psychology, 6*, 245-280.

Rumelhart, D.E., & McClelland, J.L. (Eds.). (1986). *Parallel distributed processing: Explorations in the microstructure of cognition.* Cambridge, MA: MIT Press/Bradford Books.

Seligman, M.E.P. (1970). On the generality of the laws of learning. *Psychological Review, 77*, 406-418.

Seligman, M.E.P., & Hager, J.L. (1972). *Biological boundaries of learning.* Englewood Cliffs, NJ: Prentice-Hall.

Shepard, R.N. (1987a). Evolution of a mesh between principles of the mind and regularities of the world. In J. Dupre (Ed.), *The latest on the best: Essays on evolution and optimality.* Cambridge, MA: MIT Press/Bradford Books.

Shepard, R.N. (1987b). Toward a universal law of generalization for psychological science. *Science, 237*, 1317-1323.

Shepard, R.N. (1989). Internal representation of universal regularities: A challenge for connectionism. In L. Nadel, L.A. Cooper, P. Culicover, & R. M. Harnish (Eds.), *Neural connections and mental computation*. Cambridge, MA: MIT Press/Bradford Books.

Shettleworth, S.J. (1983). Function and mechanism in learning. In M. Zeiler & P. Harzen (Eds.), *Advances in analysis of behavior: Biological factors in learning (Vol. 3)*. New York: Wiley.

Shettleworth, S.J. (1984). Natı' ıl history and evolution of learning in nonhuman mammals. In P. Marler & H.S. Terrace (Eds.), *The biology of learning* (pp. 419-434). New York: Springer-Verlag.

Singer, W. (1988). Ontogenetic self-organization and learning. In J.L. McGaugh (Ed.), *Brain organization and memory: Cells, systems, and circuits*. New York: Oxford University Press.

Skinner, B.F. (1953). *Science and human behavior*. New York: Macmillan.

Slobodkin, L.B., & Rapoport, A. (1974). An optimum strategy in evolution. *Quarterly Review of Biology, 49*, 181-200.

Small, W.S. (1900). An experimental study of the mental processes of the rat. *American Journal of Psychology, 11*, 131-165.

Spencer, H. (1855). *Principles of Psychology* (Vols. 1-2). New York: Appleton.

Staddon, J.E.R. (1980). Optimality analyses of operant behavior and their relation to optimal foraging. In J.E.R. Staddon (Ed.), *Limits to action: The allocation of individual behavior* (pp. 101-141). New Yor' . Academic Press.

Staddon, J.E.R. (1983). *Adaptive behavior and learning*. Cambridge, UK: Cambridge University Press.

Stork, D.G., & Keesing, R.C. (1990). *Evolution and learning in neural networks: The number and distribution of learning trials affect the rate of evolution*. Paper presented at Neural Information Processing Systems: Natural and Synthetic, Denver, CO.

Sutton, R.S., & Barto, A.G. (1987). A temporal-difference model of classical conditioning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society* (pp. 355-378). Hillsdale, NJ: Erlbaum.

Thorndike, E.L. (1898). Animal intelligence: An experimental study of the associative processes in animals. *Psychological Review Monographs, 2*, 1-109.

Thorndike, E.L. (1911). *Animal intelligence*. New York: Macmillan.

Timberlake, W., & Lucas, G.A. (1989). Behavior systems and learning: >From misbehavior to general principles. In S.B. Klein & R.R. Mowrer (Eds.), *Contemporary learning theories* (pp. 237-275). Hillsdale, NJ: Erlbaum.

Tinbergen, N. (1951). *The study of instinct*. Oxford: Clarendon Press.

Todd, P.M., & Miller, G.F. (1990). Exploring adaptive agency II: Simulating the evolution of associative learning. In *Proceedings of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats*. Cambridge, MA: MIT Press.

Todd, P.M., & Miller, G.F. (in press). Exploring adaptive agency III: Simulating the evolution of habituation and sensitization. In *Proceedings of the First International Conference on Parallel Problem Solving from Nature*. Berlin: Springer-Verlag.

Vaccarino, F.J., Schiff, B.B., & Glickman, S.E. (1989). Biological view of reinforcement. In S.B. Klein & R.R. Mowrer (Eds.), *Contemporary learning theories: Pavlovian conditioning and the status of traditional learning theory* (pp. 111-142). Hillsdale, NJ: Erlbaum.

Watson, J.B. (1914). *Behavior: An introduction to comparative psychology*. London: Holt, Rinehart, & Winston.

Watson, J.B. (1919). *Psychology from the standpoint of a behaviorist*. Philadelphia: Lippincott.

Weiss, G. (1990). *Combining neural and evolutionary learning: Aspects and approaches* (Tech. Rep. No. FKI-132-90). Munich: Technical University of Munich.

Williams, D.R., & Williams, H. (1969). Automaintenance in the pigeon: Sustained pecking despite contingent non-reinforcement. *Journal of the Experimental Analysis of Behavior, 12*, 511-520.

The Evolution of Learning: An Experiment in Genetic Connectionism

David J. Chalmers
Center for Research on Concepts and Cognition
Indiana University
Bloomington, IN 47408.
dave@cogsci.indiana.edu

Abstract

This paper explores how an evolutionary process can produce systems that learn. A general framework for the evolution of learning is outlined, and is applied to the task of evolving mechanisms suitable for supervised learning in single-layer neural networks. Dynamic properties of a network's information-processing capacity are encoded genetically, and these properties are subjected to selective pressure based on their success in producing adaptive behavior in diverse environments. As a result of selection and genetic recombination, various successful learning mechanisms evolve, including the well-known delta rule. The effect of environmental diversity on the evolution of learning is investigated, and the role of different kinds of emergent phenomena in genetic and connectionist systems is discussed.

1 INTRODUCTION

Evolution and learning are perhaps the two most fundamental adaptive processes, and their relationship is very complex. This paper explores one aspect of this relationship. how, via the process of evolution, the process of learning might evolve. Taking this approach, we can view evolution as a kind of second-order adaptation. it is a process that produces individual systems that are not immediately adapted to their environment, but that have the ability to adapt themselves to many environments by the first-order adaptive process of learning. Here the learning mechanisms themselves are the object of evolution. Using a combination of methods drawn from connectionism and genetic search, we will start with a population of individuals with no ability to learn, and attempt to evolve useful learning techniques.

The details of the experiments performed here start in the second section. The remainder of this introduction is devoted to a somewhat philosophical discussion of the role of evolutionary computation in the modeling of cognitive systems, and in particular the importance of the phenomenon of emergence. A brief outline of related work combining connectionist and genetic methods is also given. Readers who wish to skip this background material may proceed directly to the second section.

1.1 EVOLUTION, EMERGENCE AND COMPUTATION

In recent years, the evolutionary approach to the computational modeling of cognitive systems has gained prominence, following the work of Holland and his colleagues on genetic algorithms (Holland 1975, Goldberg 1989), a class of methods of search directly inspired by biological systems. The attractions of this approach are many. To the biologist, ethologist or psychologist, evolutionary computation offers insight into the mechanisms that gave rise to adaptations present in existing living systems. To the computer scientist or the engineer, genetic search can yield a powerful method of problem solving. If we accept the often-repeated slogan, "Nature is smarter than we are", then harnessing the problem-solving methods that nature uses makes sound practical sense. Finally, to a cognitive scientist, evolutionary methods offer a paradigm of *emergence* that seems to have much in common with the kind of emergence found in contemporary connectionist systems. In both kinds of systems, complex high-level behavior is produced as a result of combining simple low-level computational mechanisms in simple ways.

The kind of emergence found in genetically-based systems differs, however, from that found in connectionist systems. Connectionist systems support *synchronic emergence*, or emergence over levels: at a given time, a host of low-level computations are taking place, which when looked at on another level can be interpreted as complex high-level functioning. By contrast, genetically-based systems support *diachronic emergence*, or emergence over time. primitive computational systems gradually evolve towards greater complexity.

To the cognitive scientist, while there is a good deal of interest in the *process* of evolution, the greatest interest lies in the *product*. One wishes that eventually, as a product of the appropriate sort of genetic search, we will be able to produce plausibly functioning cognitive systems. This is where the distinction between synchronic and diachronic emergence becomes important. The diachronic emergence of the evolutionary process gives no guarantee that the systems that result from the process will manifest the kind of emergence-over-levels that is so important in connectionist systems. For example, classifier systems, the most prominent application of genetically-based computation (Holland 1986), give rise via genetic search to a kind of production system, a paradigm of symbolic computation. To someone—a connectionist, for example—who believes that emergence-over-levels is fundamental to most cognitive phenomena, this is somewhat unsatisfying. This is not to say that classifier systems are uninteresting—they have yielded powerful insights into the process of learning, for instance (see e.g. Holland, Holyoak, Nisbett and Thagard 1986)—but it would be very interesting to see whether evolutionary methods could be used to develop systems that might be classed as emergent in their own right.

The road to achieving synchronic emergence through evolutionary methods seems clear: loosen the connection between genotype and phenotype.[1] In classifier systems (and in most other current applications of genetic algorithms) the connection between these is very direct. An element of the genome codes for a feature-detector or an action in the phenotype in a direct, symbolic fashion. This contrasts strongly with the human case, in which elements of the genome appear to determine phenotypic functioning in only a very indirect way, coding for low-level mechanisms that produce human behavior via "action at a distance". When the genotype encodes high-level features directly and symbolically, there is no room for synchronic emergence. Emergence-over-levels can only take place when levels are distinct. The key to achieving full emergence with evolutionary systems seems to lie in the ability to allow phenotypic characteristics to emerge indirectly from genetic information.

There is another motivation for indirect mappings from genotype to phenotype. open-ended search. Current genetically-based methods have been criticized for not allowing a sufficiently "open-ended" space of possibilities for evolution. It is a feature of current genetic search methods that a genotypic space is precisely specified in advance, and search cannot go outside this space. When this is coupled with a direct genotype-to-phenotype mapping, it translates directly into a strongly-delineated phenotypic space whose properties are well-understood in advance. The role of genetic search is reduced to that of optimizing over such a well-understood space. However, if the relationship between genotype and phenotype is indirect and emergent, phenotypic space need not be so strongly constrained. To be more precise, in such cases phenotypic space will be indirectly constrained by the genotypic space, but synchronic emergence guarantees that high-level phenotypic characteristics need not be limited in advance by the imagination of the designer.

There are at least two different ways to achieve such an emergent relationship between genotype and phenotype. The first is to make the genotype code for *subsymbolic* computation in the phenotype. it might, for example, determine the low-level computations of a connectionist system, without explicitly constraining high-level behavior. Secondly, a genotype might code for *developmental* processes in the phenotype. On this approach, the phenotype's final form would not be specified, but instead would result from a complex pattern of developmental processes over its lifetime. In this paper, we will use both kinds of emergent genotype-phenotype relationships simultaneously: the genotype will specify the dynamics of a developing system that interacts with an environment, and these dynamics will control low-level properties of a connectionist system. In this way, we will see whether adaptation at the level of evolution can give rise to adaptation at the level of the individual: specifically, whether we can produce, through evolution, the capacity for a connectionist system to learn.

1.2 GENETIC CONNECTIONISM

The combination of genetic search with connectionist computation is already popular. The power of genetic coding at the subsymbolic level is well-recognized. A key motivation behind this "genetic connectionism" stems from the fact that due to the synchronic emergence of connectionist systems, it is difficult to know in advance precisely which low-level computations are appropriate for a specific high-level behavior. In view of this difficulty, it makes sense to use genetic methods to search for an appropriate low-level computational form. Rather than construct the right kind of system or architecture by hand, why not let natural selection do the work?

Genetic connectionism to date has usually taken one of two forms. Genetic information has been used to encode the connection strengths in a neural network (e.g. Belew, McInerney & Schraudolph, 1990, Wilson 1990), or to encode the overall topology of a network (Miller, Todd & Hegde 1989; Whitley, Starkweather &

[1] The genotype is the collection of genetic information passed on between generations: in a genetic algorithm, this is typically a string of bits. The phenotype is the behavioral expression of the genotype, an entity that interacts with an environment and is subject to selection by differential fitness. In the human case, genotype=DNA, phenotype=person.

Bogart 1989). These methods have yielded interesting results on the optimization of network design. They have not, however, led to qualitatively new kinds of connectionist processes. The reason for this lies with the fact that the phenotypic spaces are qualitatively well-understood in advance (although, as pointed out above, the relationship between individual phenotypes and genotypes may not be). Genetic methods provide a powerful method of searching these spaces, but are unlikely to come up with surprising results. Soon we will look at an application where the nature of the phenotypic space is less clear in advance.

Much interesting recent work has focused on the relationship between evolution and learning. It has become apparent that genetic connectionism is an ideal modeling tool in this endeavor. genetic algorithms providing an elegant model of evolution, and connectionism providing simple but powerful learning mechanisms. Belew, McInerney and Schraudolph (1990), following up suggestions by Maynard Smith (1987) and Hinton and Nowlan (1987), have demonstrated the complementary nature of the two phenomena: the presence of learning makes evolution much easier (all evolution has to do is find an appropriate initial state of a system, from which learning can do the rest); and evolutionary methods can significantly speed up the learning process. Nolfi, Elman and Parisi (1990) have similarly investigated the ways in which the presence of learning might facilitate the process of evolution.

These studies have presumed a learning process to be fixed in advance—usually backpropagation—and have taken the object of evolution to be the initial state of a system, upon which the learning process may act. The genesis of the learning mechanisms themselves is not investigated. This study will take a different approach. Here, the learning process itself will be the object of evolution. We will start with systems that are unable to learn, subject these systems to genetic recombination under selective pressures based on their ability to adapt to an environment within their lifetime, and see whether any interesting learning mechanisms evolve.

2 EVOLUTION OF LEARNING IN NEURAL NETWORKS

2.1 GENERAL FRAMEWORK

The main idea in using genetic connectionism to model the evolution of learning is simple: use a genome to encode not the static properties of a network, but the dynamic properties. Specifically, the genome encodes a procedure for changing the connection strengths of a network over time, based on past performance and environmental information. In the experiment to be outlined here, the genome encodes no static properties at all. The initial configuration of a network is instead determined randomly (within a constrained framework). Thus, if genetic search is to be successful, it must evolve a learning procedure that can start from a wide variety of network configurations and reliably reach an appropriate result.

Working within this paradigm it is important to distinguish two levels of adaptation: learning—adaptation at the level of the individual—where an individual over time adapts to its environment, and evolution—adaptation at the level of the population—where over the course of evolutionary history a population gradually evolves mechanisms that improve the fitness of its members. In our case, the fitness of a single member of the population will be measured by its capacity for adaptation within its lifetime (learning), so the evolutionary mechanisms can be viewed as a second-order adaptive process. a population-wide adaptation that over generations produces better adaptive processes within the lifetime of an individual.

A vital component of the learning process is the *environment*. If the environment were relatively static, there might be little need for learning to evolve. Systems could instead evolve to a state where they have innate mechanisms to handle that environment. But if the environment is diverse and unpredictable, innate environment-specific mechanisms are of little use. Instead, individuals need general adaptive mechanisms to cope with arbitrary environments. In this way, a diverse environment encourages the evolution of learning.

Our basic *modus operandi* is as follows. A genome encodes the weight-space dynamics of a connectionist system—that is, it encodes a potential learning procedure. The fitness of a learning procedure is determined by applying it to a number of different learnable tasks. This is achieved by creating a number of networks and putting them in different environments for a specified amount of time. (In the experiments to be outlined here, an environment consists of a set of input patterns with associated training signals.) Each network has a random initial state, but its final state is determined by its interaction with the learning procedure and the environment. The fitness of the network is determined by how well it has adapted to its environment after the specified amount of time. The fitness of the learning procedure is derived from the fitness of the various networks that have used it. Learning procedures reproduce differentially over time, depending on their success in yielding adaptive networks. The hope is that from a starting population of essentially useless learning procedures, we will eventually produce something that enables powerful adaptation.

In principle, such a process could be of interest not only to biologists and psychologists but also to computer scientists and engineers. The space of possible learning mechanisms is very poorly understood. Genetic search allows the exploration of this space in a manner quite different from the usual combination of ingenuity and trial-and-error employed by algorithm designers. It is not impossible that genetic search could come up with a learning algorithm that rivals existing human-designed algorithms. At the very least, it might produce a learning algorithm already known to be useful. This, in fact, was the result of the experiments outlined here.

2.2 IMPLEMENTATION DETAILS

We may now move from broad generalities to specific implementation details. In this preliminary experiment, an effort was made to keep things as simple as possible, in the interests of minimizing computational requirements and maximizing the interpretability of results. The choices made below were appropriate for a study of the feasibility of the methodology outlined above, but many of them could be varied with potentially quite different results.

2.2.1 Type of Learning Task

First, we must choose what kind of learning we will try to evolve. The three standard categories of learning tasks are supervised learning (learning with full training feedback about desired actions), reinforcement learning (where the only training information is a scalar representing the utility of a given action), and unsupervised learning (learning without any external teacher) Any of these kinds of learning could potentially be investigated via evolutionary methods, but here we chose supervised learning, as it is both the easiest and the best-und' stood of the three varieties. The tasks will involve associating specific input patterns with specific output patterns, where the desired output patterns are presented to the network as a training signal.

2.2.2 Network Architecture

The next choice to be made is that of the topology of the networks. Under the assumption that the genome codes for weight-space dynamics, we need not fix weight values in advance, but it is necessary that some form of network design be fixed. (Another approach might be to evolve network dynamics and network topology simultaneously, but that was not pursued here.) The simplest non-trivial topology is that of a single-layer feed-forward network. These networks also have the advantage that a powerful learning algorithm--the delta rule, also known as the Widrow-Hoff rule—is known to exist in advance, at least for supervised learning tasks. This experiment employs fully-connected single-layer feed-forward networks with sigmoid output units, with a built-in biasing input unit to allow for the learning of thresholds. A maximum connection strength of 20 was imposed, to prevent possible explosion under some learning procedures.

2.2.3 Genetic Coding of Learning Mechanisms

We need to be able to code complex forms of weight-space dynamics into a simple linear genome. Clearly, we are not going to be able to express all possible kinds of weight-space dynamics under a single encoding; instead, the dynamics must be severely constrained. In this experiment, it was decided that changes in the weight of a given connection should be a function of only information local to that connection, and that the same function should be employed for every connection. For a given connection, from input unit i to output unit j, local information includes four items:

a_j the activation of the input unit j.
o_i the activation of the output unit i.
t_i the training signal on output unit i.
w_{ij} the current value of the connection strength.

The genome must encode a function F, where

$$\Delta w_{ij} = F(a_j, o_i, t_i, w_{ij}).$$

It was decided that F should be a linear function of the four dependent variables and their six pairwise products. Thus F is determined by specifying ten coefficients. (Note that this framework for weight-space dynamics does not exclude the delta rule as a possible candidate. This is not, of course, entirely coincidental, and the charge of using prior knowledge about learning to constrain the evolutionary process might be leveled. However, one advantage of the network topology we have chosen is that a sufficiently simple good learning procedure exists. Due to the simplicity of the procedure, we can allow it as a possibility without rigging the possibilities in too arbitrary a manner in advance.)

The genome specifies these ten coefficients directly, with the help of an eleventh "scale" parameter. We put

$$\begin{aligned}\Delta w_{ij} = \quad k_0(\quad & k_1 w_{ij} + k_2 a_j + k_3 o_i + k_4 t_i + \\ & k_5 w_{ij} a_j + k_6 w_{ij} o_i + k_7 w_{ij} t_i + \\ & k_8 a_j o_i + k_9 a_j t_i + k_{10} o_i t_i).\end{aligned}$$

The genome consists of 35 bits in all. The first five bits code for the scale parameter k_0, which can take the values $0, \pm 1/256, \pm 1/128, \ldots, \pm 32, \pm 64$, via exponential encoding. The first bit encodes the sign of a_0 (0=negative, 1=positive), and the next four bits encode the magnitude. If these four bits are interpreted as an integer j between 0 and 15, we have

$$|k_0| = \begin{cases} 0 & \text{if } j = 0 \\ 2^{j-9} & \text{if } j = 1, \ldots, 15. \end{cases}$$

The other 30 bits encode the other ten coefficients in groups of three. The first bit of each group expresses the sign, and the other two bits express a magnitude of 0, 1, 2 or 4 via a similar exponential encoding. If we interpret these two bits as an integer j between 0 and 3, then

$$|k_i| = \begin{cases} 0 & \text{if } j = 0 \\ 2^{j-1} & \text{if } j = 1, 2, 3. \end{cases}$$

For example, the delta rule could be expressed genetically as:

11011 000 000 000 000 000 000 000 010 110 000

where the coefficients decode to

$$4 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; -2 \;\; 2 \;\; 0$$

and the appropriate formula is thus

$$\begin{aligned} \Delta w_{ij} &= 4(-2a_j o_i + 2a_j t_i) \\ &= 8a_j(t_i - o_i). \end{aligned}$$

2.2.4 The Environment

Given that we are trying to evolve successful performance on supervised learning tasks, we need an appropriate environment consisting of a number of learnable tasks. Each "task" in the environment used here consists of a mapping from input patterns to output patterns. As we are using single-layer networks, learnable mappings must be linearly separable. Thirty diverse linearly separable mappings were used. These had between two and seven units in the input patterns, and always a single unit as output. (Any single-layer network with more than one output unit is equivalent to a number of disjoint networks with a single output unit each, so there is no point using more than one output unit at a time.) For each task, a network was presented with a number of training exemplars, each consisting of an input pattern and associated output.

Table 1 shows eight of the tasks that were used. Each of these was a linearly separable mapping from five input units to a single output unit. The eight rightmost columns of the table represent the correct outputs for the eight different tasks. The five leftmost columns represent the input units, common to all eight tasks. For each task, twelve training exemplars were presented, corresponding to the twelve rows of the table. For example, the first task shown was to detect whether the fifth unit in the input pattern was on or off. The second task involved recognizing a single specific pattern. The other tasks were more complex.

Table 1: Eight Tasks from the Environment.

a_1	a_2	a_3	a_4	a_5	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
1	1	1	1	1	1	0	0	0	1	0	1	0
0	0	0	0	0	0	0	1	0	0	1	0	1
0	1	1	1	0	0	0	1	0	1	0	1	0
1	1	0	0	0	0	1	1	1	0	1	1	1
1	0	1	0	1	1	0	0	0	0	0	1	1
0	1	1	0	0	0	0	1	0	1	1	0	1
0	1	1	1	1	1	0	1	0	1	0	1	0
0	1	0	0	0	0	0	1	1	1	1	0	1
1	1	0	0	1	1	0	0	1	0	1	1	1
1	0	0	1	0	0	0	1	1	1	0	1	1
1	0	1	1	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	0	0	1

2.2.5 Evaluation of Fitness

With this diverse "environment" in hand, evaluation of the fitness of a given learning algorithm proceeds as follows. A number of tasks (typically 20) are randomly selected at the start of every evolutionary run. For each learning algorithm in the population, we measure its performance on each task. Fitness on a given task is measured by the following procedure:

(1) Create a network with the appropriate number of input units for the task and a single output unit.

(2) Initialize the connection strengths of the network randomly between -1 and 1.

(3) For a number of epochs (typically 10), cycle through the training exemplars for the task, where for each exemplar we:

(3a) Propagate input values through the system, yielding output values; then

(3b) Adjust the weights of the system according to the formula specified by the learning procedure. on the basis of inputs, output, training signal, and current weights.

(4) At the end of this process, fitness on the task is measured by testing the network on all training exemplars, and dividing the total error by the number of exemplars, subtracting from 1, and multiplying by 100. This yields a fitness "percentage" between 0 and 100.

Fitness of the learning rule is obtained by evaluating its performance on each of the (typically 20) tasks, and taking the mean fitness over all tasks. In this way every learning procedure is assigned a fitness between 0 and 100%. A fitness of 50% represents chance performance (i.e., no learning over the lifespan of an individual network. A fitness of 100% indicates perfect learning, at least on the given tasks at the end of ten epochs, each mapping is learned perfectly. It should be noted that given the limited number of epochs, perfect fitness seems unachievable if we are using more than one or two tasks: even the delta rule has a fitness of only around 98%.

There is some stochastic variation in the fitness of a given learning rule, due to the random variation in initial weights of the networks. This variation is reduced by maintaining a history of each specific learning rule that appears in a given evolutionary process. If a learning rule has appeared already in an evolutionary run, its fitness is computed by the above procedure, but the fitness used for purposes of selection is the mean fitness from this and all prior appearances. This occurs until a given learning rule has appeared fifteen times, at which time the mean fitness is recorded for good and never recomputed, thus saving a significant amount of computational resources.

2.2.6 Parameters of the Genetic Algorithm

For those unfamiliar with the genetic algorithm, it works as follows. We start with a population of appropriate genomes—in this case random bit-strings of 35 bits each. In these experiments, the population size was always 40. Every generation, each genome is converted into a phenotype (here, a learning procedure), and its fitness is measured (as outlined above). Then we have a process of differential *reproduction*. Each genome probabilistically makes copies of itself, with the probability of reproduction being linearly proportional to its fitness. This yields a new population of 40 genomes, which is then subjected to the process of *crossover*: pairs of genomes are "mated" by taking a randomly selected string of bits from one and inserting it into the corresponding place in the other, and vice versa, thus producing two new genomes. In this experiment, 80% of the population are subject to this process, while the remaining 20% reproduce asexually by copying themselves directly into the new population. "Elitist" selection is also used: the best individual in the previous population is guaranteed a place in the new population by asexual reproduction. Finally, in the resulting population of 40 genomes of 35 bits each, each bit has a 1% chance of *mutation*—that is, of changing from 0 to 1 or vice versa. This process of fitness-measurement, reproduction, crossover and mutation is repeated for a number of generations, usually 1000.

Summary of genetic algorithm parameters. Population = 40, two-point crossover and elitist selection are used, crossover rate = 0.8, mutation rate = 0.01, number of generations = 1000.

3 RESULTS

When the genetic algorithm is run, initial results are as expected. At the start of the run, all individuals present in the population have fitness between 40% and 60%, indicating no significant learning behavior. Instead, weight-dynamics are essentially random over time. Within a few generations, differential reproduction begins to exploit even small differences in fitness, and the fitness of the best individuals in the population rises rapidly as simple adaptive mechanisms make their way into the weight-space dynamics. After 1000 generations, the maximum fitness is typically between 80% and 98%, with an mean of about 92%. Table 2 gives the results of ten separate runs with the same parameters but different random seeds.

Table 2: Best Learning Algorithms Produced on 10 Evolutionary Runs.

k_0	k_1	k_2	k_3	k_4	k_5	k_6	k_7	k_8	k_9	k_{10}	Fitness
0.25	0	0	0	0	0	0	0	-4	4	0	89.6%
-2.00	0	0	0	0	0	0	0	2	-2	0	98.0%
0.25	0	-1	-2	4	0	0	0	-2	4	-2	94.3%
0.25	0	-1	-2	4	0	0	0	-2	4	-2	92.9%
-0.25	0	0	1	-1	0	1	-1	4	-4	0	89.8%
-1.00	0	0	-1	1	0	0	0	4	-4	0	97.6%
4.00	0	0	1	-1	0	0	0	-2	2	0	98.3%
-0.06	0	0	0	-2	-1	2	2	4	-4	2	79.2%
-0.25	0	0	2	-1	0	-1	-1	2	-4	0	89.8%
0.25	0	-1	-2	4	0	0	0	-2	4	-2	93.2%

Note that even the worst learning rule has significant adaptive ability, with a fitness of 79.2%. This rule, then, enables a network to predict the correct output for a given input with approximately 80% accuracy after 10 epochs of training. Other rules are significantly better, with a mean fitness of 92.3%.

On the second of the ten runs, the genetic search process discovered a version of the well-known delta rule (or Widrow-Hoff rule). The learning rule here was:

$$\begin{aligned} \Delta w_{ij} &= -2(2a_j o_i - 2a_j t_i) \\ &= 4a_j(t_i - o_i). \end{aligned}$$

This rule unsurprisingly has a high fitness of 98.0%. Such an event was not unusual—the delta rule was discovered on perhaps 20% of all runs with similar parameters. In this set of ten runs, the delta rule evolved twice (the second occurrence has a much lower value of the "learning rate" k_0, and so a lower fitness of 89.6%). Slight variations on the rule also evolved twice, with high fitnesses of 98.3% and 97.6%. In the seventh run

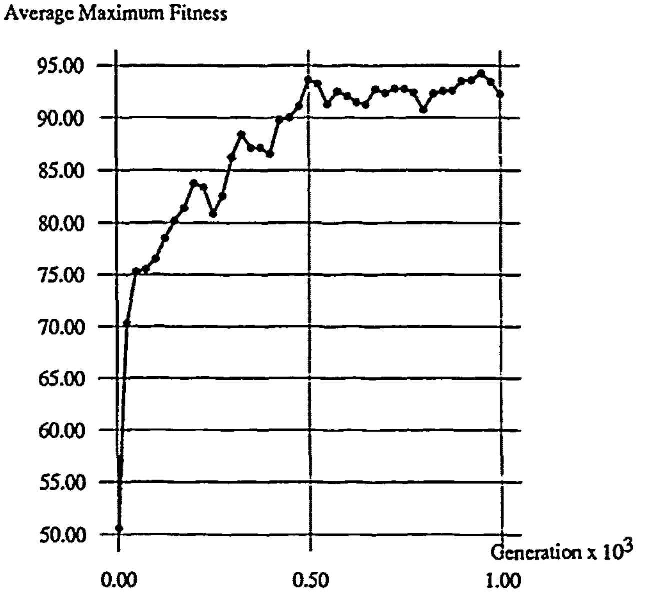


Figure 1. Evolution of Maximum Fitness in Population.

above, the rule is

$$\begin{aligned}\Delta w_{ij} &= 4(o_i - t_i - 2a_j o_i + 2a_j t_i)\\ &= 8(a_j - 0.5)(t_i - o_i).\end{aligned}$$

In the sixth run above, the rule is

$$\begin{aligned}\Delta w_{ij} &= -1(-o_i + t_i + 4a_j o_i - 4a_j t_i)\\ &= 4(a_j - 0.25)(t_i - o_i).\end{aligned}$$

The similarity of these rules to the delta rule is clear.

Typical evolutionary progress in fitness is shown in Figure 1. This is a graph of the fitness of the best individual learning rule in the population every 25 generations over the course of the run. The figures are averaged over the ten runs mentioned above.

3.1 EFFECT OF ENVIRONMENTAL DIVERSITY ON THE EVOLUTION OF LEARNING

Given the learning algorithms that this evolutionary method produces, it is natural to ask whether the algorithms are adapted specifically for the tasks that have been present in the environment during the evolutionary process, or whether the algorithms are in fact very general adaptive mechanisms capable of learning a wide range of tasks, including tasks that were not present during the evolutionary process. The delta rule clearly falls into the second category of general adaptive mechanisms, but for other kinds of weight-dynamics evolved by this process, the answer is not so clear *a priori*.

It might well be thought that given the limited number of tasks in the evolutionary environment, the evolved learning mechanisms would function well only on those tasks. At the very least, it is plausible that performance on such tasks might be superior to performance on tasks not in the evolutionary environment. On the other hand, it seems plausible that if enough tasks are present in the evolutionary environment, then it is unlikely that highly task-specific mechanisms would evolve. In the design of the experiment, 20 different tasks were used in the environment of a given evolutionary run precisely because this seemed a large enough number of tasks to minimize this likelihood.

The task-specificity of evolved learning mechanisms can be measured by applying such mechanisms to a selection of new tasks not present in the evolutionary environment. In this experiment, this measurement was made by starting from a pool of 30 tasks. On every run, 20 of these were randomly selected and designated as part of the evolutionary environment, and evolution proceeded on the basis of how well these 20 tasks could be learned. The other 10 tasks were designated as test tasks: at the end of an evolutionary run, the best learning algorithm in the evolved population was tested on these 10 new tasks, and mean fitness was measured.

For the 10 runs outlined in the previous section, the average fitness of the 10 final learning algorithms on the 20 tasks in their evolutionary environment was 92.3%. The average fitness these algorithms on the tasks not present in the evolutionary environment was 91.9%. This indicates that the evolutionary environment was sufficiently diverse that there was no significant tendency for task-specific mechanisms to evolve.

The next question of interest is that of how diverse the environment must be to force the evolution of general learning mechanisms. To investigate this, the number of tasks in the evolutionary environment was varied between 1 and 20 on a number of runs. Performance of a learning algorithm on this set of tasks is referred to as *evolutionary fitness*. After 1000 generations, the learning algorithm with the highest evolutionary fitness was removed and tested on 10 further tasks that were not in the evolutionary environment. Performance on this set of tasks—the *test fitness*—indicates the generality of the learning mechanisms that evolve. For each of a number of values of the number of tasks in the evolutionary environment, 10 runs were performed with different random seeds, and the results for these runs were averaged. Figure 2 graphs (1) the average final evolutionary fitness and (2) the average test fitness, as a function of the number of tasks in the evolutionary environment.

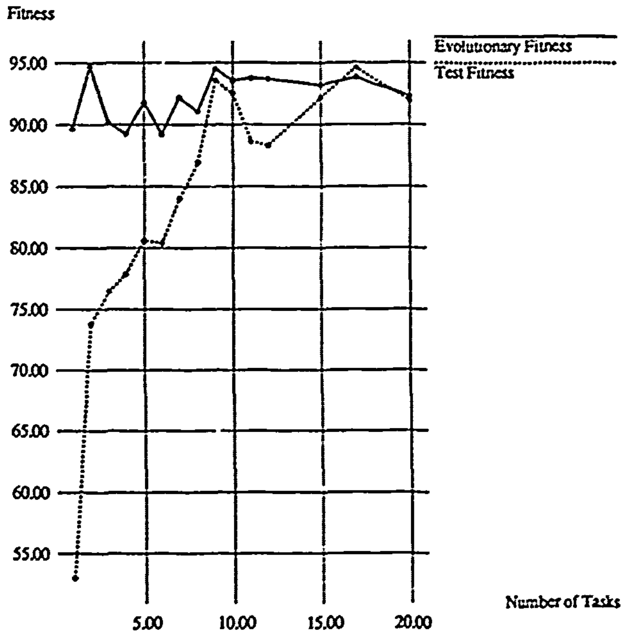


Figure 2: Evolutionary Fitness and Test Fitness versus Number of Tasks.

The first thing to note is that evolutionary fitness is fairly constant as a function of the number of tasks, always somewhere around 90%. In some ways, this is surprising—we might expect superior performance on a smaller number of tasks. It seems plausible that it would be easier to evolve good learning mechanisms for one task than for 20. The reason for the observed constancy perhaps lies with the fact that when there is only a single task, there are more ways to learn to perform it quite well but suboptimally. The greater ease of finding a successful learning mechanism may be counterbalanced by an increased chance that the evolutionary process will get stuck in a suboptimal local maximum, from which it is difficult to escape even by genetic recombination.

The more important thing to note is that test fitness does indeed decrease significantly as the number of tasks decreases. When there is a single task in the evolutionary environment, test fitness averages 53%, or only just above chance. This indicates that the kind of weight-dynamics that evolves, while performing well on the task in the evolutionary environment (fitness 89.7%), is not suited at all for other tasks. A task-specific mechanism has evolved, rather than a general learning mechanism. When the number of tasks is between 1 and 9, test fitness rises above chance but is well below evolutionary fitness, again indicating some degree of task-specificity in the learning mechanisms. When the number of tasks is about 10, test fitness rises to a value very close to evolutionary fitness, indicating that this number of tasks is sufficient to force the evolution of general learning mechanisms.

The moral here is clear: a sufficiently diverse environment brings about the evolution of general learning mechanisms. This makes intuitive sense. If the environment is fairly constant, there is no need for evolution to produce mechanisms that function in more general contexts—instead, we might expect organisms to inherit innate mechanisms adapted to that specific environment. When the environment is diverse and relatively unpredictable, we should expect individual organisms to inherit adaptive mechanisms capable of coping with a variety of different environmental features as they arise This is precisely what we have found here.

These results also have some bearing on the traditional controversy between nativists and empiricists. When the environment in these experiments was not diverse, the mechanisms that evolved to cope with that environment may be regarded as *innate*, at least in a weak sense. These mechanisms are present specifically to cope with certain tasks. Although the capacity to deal directly with the tasks is not actually present at the beginning of the lifetime of an individual, the capacity is triggered by an expected sequence of events from the environment on the basis of task-specific information present in the individual's genome. In this case, we might say that the weight-space dynamics represent a task-specific *developmental* process for an innate capacity, rather than a learning process. When the environment is diverse, on the other hand, the only mechanisms that individuals possess innately are general learning mechanisms. In a situation familiar to empiricists, the individual is a tabula rasa, ready to be imprinted with whatever regularities it finds in the environment.

We should be cautious about applying the results from these limited experiments to theorizing about natural biology and psychology. In particular, an important difference is that in these experiments, genetic information codes only dynamic properties of an individual, and specifies no structural information about the individual's initial state. In natural biological systems, the genome appears to carry much information about the initial structure of a system. This would make some differences to our conclusions about innateness, for example. But a general moral may still be drawn, if strewn with caveats: innate mechanisms are likely to be the product of a relatively constant evolutionary environment; adaptive, empiricist-style mechanisms may be a consequence of environmental diversity.

4 DISCUSSION AND FURTHER DIRECTIONS

We have seen that the methods of genetic search and connectionism can be combined to provide a demonstration that the capacity for learning (adaptation within the lifetime of an individual) may evolve in a natural fashion under the pressures of evolution (adaptation over the history of a population). This double adaptive loop seems to be a powerful mechanism for coping with a diverse environment. All that seems necessary for learning to evolve in this fashion is that the genome encode the long-term *dynamics* of the information-processing capacities of an individual. If genetic variation allows for variation in these dynamics, then a diverse environment will exert pressure towards dynamics that are adaptive over the lifetime of an individual; eventually, we should expect learning mechanisms to arise.

The kind of learning that we have allowed here has been very simple, to allow a simple proof of possibility and an illustration of the key issues. Supervised learning is a fundamental kind of learning, but it is not very plausible biologically, and it also suffers from the problem that we may understand it *too* well. At least in feed-forward networks, we already possess very powerful supervised learning methods in the delta rule and its generalization to multi-layer networks, backpropagation. It is unlikely that evolutionary methods will allow us to improve much on these methods. So while the methods here can yield some insight into the evolutionary processes behind learning, they have not given much insight into learning itself.

Other forms of learning are much less well-understood, however, and the methods of genetic connectionism outlined here may provide a novel means of insight. For example, reinforcement learning is much more difficult than the supervised variety. A number of learning algorithms are known (e.g. Barto 1985, Williams 1988), but none have been as successful as backpropagation has been. Using genetic search, we can investigate the space of possible dynamics of reinforcement learning systems, and it is not impossible that we might come up with novel algorithms. The extension, in principle, is a simple one. The problem of unsupervised learning, although more complex, could be attacked in a similar fashion.

Another possible generalization would be to a different class of network architectures. We could, for example, attempt to evolve learning algorithms for recurrent network. Another interesting approach would be to attempt to evolve static and dynamic properties of a network simultaneously. Network topology, the values of certain weights, and a learning algorithm could be simultaneously encoded by a genome, and genetic search might fruitfully explore the interaction between these factors.

There is one problem that is always lurking in the background here, and that is the problem of the genetic coding. How do we choose such a coding? Do we try to allow for as many kinds of weight-space dynamics as possible, or do we constrain the space using prior knowledge? How could we possibly find a coding of possible dynamics that includes as possibilities all the diverse learning algorithms proposed by humans to date? In the experiment described in this paper, the decision was easy. The networks were small, there was only a certain amount of relevant information, and it was known in advance that a simple quadratic formula could provide a good learning mechanism. The encoding of more ambitious mechanisms may not be so simple. To attempt to evolve backpropagation by this method, for example, we would need either a highly complex genetic coding, or else a simple but very specific coding that was rigged in advance to allow backpropagation as a possibility. When we do not know the form of a plausible learning algorithm in advance—and this is the most interesting and potentially fruitful application of these methods - the problem of the coding becomes very important. Only so much can be coded into a finite-length bit string.

One way around the limitation of prespecified codings of dynamic possibilities would be to move away from the encoding of learning algorithms as bit-strings, and instead encode algorithms directly as function trees. In a recent report, Koza (1990) has demonstrated the potential of performing genetic-style recombination upon function-tree specification of algorithms. This method of "genetic programming" uses recombination and selection in a fashion very similar to traditional genetic methods, but with the advantage that under evolutionary pressures such function-trees may become arbitrarily complex if necessary. This open-endedness may be a good way of getting around the limitations inherent in fixed genetic codings. Furthermore, the method is a very natural way of encoding dynamic, algorithmic processes of the kind we are investigating here.

Even if we are forced to constrain the possible kinds of learning algorithm, genetic methods provide a powerful way of searching the spaces of such learning algorithms. Unlike other phenotypic spaces such as those of classifier systems or network topologies, the space of learning algorithms is so poorly understood that even the gross qualitative properties of a given algorithm are very difficult to predict in advance. Genetic search may allow to us uncover algorithms that humans might never consider.

In sum, genetic connectionism provides a tool of analysis that may be of interest to biologists and psychologists, and also to computer scientists and engineers. Genetically-based methods provide a direct way to model evolution, a powerful method of search, and a paradigm of emergence-over-time. Connectionist methods have the potential for sophisticated forms of learning via their paradigm of emergence-over-levels. Combining genetic emergence-over-time with connectionist emergence-over-levels seems to provide a property that neither class of methods possesses alone: automated creativity in the design of adaptive systems.

Acknowledgements

Thanks to Liane Gabora, Gary McGraw and Bob French for helpful comments.

References

A. G. Barto (1985). Learning by statistical cooperation of self-interested neuron-like computing elements. *Human Neurobiology*, 4: 229-256.

R. Belew, J. McInerney & N. N. Schraudolph (1990). Evolving networks: Using the genetic algorithm with connectionist learning. CSE Technical Report CS90-174, University of California, San Diego.

D. E. Goldberg (1989). *Genetic algorithms in search, optimization and machine learning.* Reading, MA: Addison-Wesley.

G. E. Hinton & S. J. Nowlan (1987). How learning can guide evolution. *Complex Systems*, 1: 495-502.

J. H. Holland (1975). *Adaptation in natural and artificial systems.* Ann Arbor: University of Michigan Press.

J. H. Holland (1986). Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems. In J. Michalski, J. G. Carbonell & T. M. Mitchell (eds.) *Machine Learning II.* Los Altos, CA: Morgan Kaufmann.

J. H. Holland, K. J. Holyoak, R. E. Nisbett, & P. R. Thagard (1986). *Induction: Processes of Inference, Learning and Discovery.* Cambridge, MA: MIT Press.

J. Koza (1990). Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems. Technical Report STAN-CS-90-1314, Stanford University.

G. Miller, P. Todd & S. Hegde (1989). Designing neural networks using genetic algorithms. In *Proceedings of the Third Conference on Genetic Algorithms and their Applications.* San Mateo, CA: Morgan Kaufmann.

S. Nolfi, J. L. Elman & D. Parisi (1990). Learning and evolution in neural networks. CRL Technical Report 9019, University of California, San Diego.

J. Maynard Smith (1987). When learning guides evolution. *Nature*, 329: 761-762.

D. Whitley, T. Starkweather & C. Bogart (1990). Genetic algorithms and neural networks. Optimizing connections and connectivity. *Parallel Computing*, forthcoming.

S. Wilson (1990). Perceptron redux. *Physica D*, forthcoming.

R. J. Williams (1988). Toward a theory of reinforcement-learning connectionist systems. Technical report NU-CCS-88-3, Northeastern University.

Evolving Controls for Unstable Systems

Alexis P. Wieland*
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024-1596
alexis@cs.ucla.edu

Abstract

This paper considers the use of the genetic algorithm (GA) to create neural networks capable of carrying out a series of increasingly difficult control problems. The control problems considered are variations of the standard pole balancing problem, including a variable length pole, multiple poles on a single cart, and a jointed pole. This paper presents these problems and their equations of motion, discusses GA created networks for handling these problems, and compares the networks for control theoretic solutions. The paper concludes that the GA and neural nets are well suited to the difficult control problems presented.

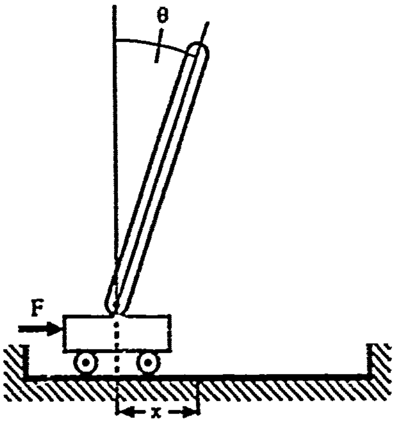


Figure 1: Single Pole Balancing Problem

1 INTRODUCTION

Balance and control are tasks that are often innate to natural systems but are annoyingly difficult to derive or compute. This paper considers artificial neural networks that have been created (or "evolved") by genetic algorithms (GA's); two paradigms that are loosely based on nature. These networks are used to control a series of pole balancing problems, including a single pole, a variable length pole, multiple poles on a single cart, and a jointed pole.

The standard pole balancing problem (also known as the cart-pole, broom balancer, or inverted pendulum problem) can be solved by a very simple neural network. The problem involves balancing a single rigid pole on a wheeled cart by exerting forces on the cart, as shown in Figure 1. This problem is of interest because it describes an inherently unstable system and represents a wide class of unstable mechanical systems. Unfortunately the pole balancing problem can and has been solved using a single computing "neuron," and as such does not address some of the more difficult areas of control. The variations of the pole balancing problem in this paper are intended to force the creation of networks that address these more difficult areas.

The remainder of this introduction surveys past work in pole balancing and provides a brief review of the relevant control theory and GA's. The next two sections describe the specifics of the neural networks and the genetic algorithms used for this work. Section 4 describes the different pole balancing problems in detail, the neural networks used to solve the problems, and compares the networks to the control theoretic solutions. The conclusions section discusses the significance of this work and suggests possible areas of future study. An appendix contains the equations of motion and parameter values for the pole balancing problems.

*The author is also a part-time/on-call employee of The MITRE Corporation, 7525 Colshire Drive, McLean, VA 22102-3481.

1.1 PAST WORK

Neural networks, genetic algorithms, and the pole balancing problem have each generated considerable literature. This section surveys past work in these areas.

1.1.1 Control Theory

The pole balancing problem is a standard control problem that has been examined in many control theory texts. This makes the pole-balancing problem particularly useful as a test problem for neural networks since it is possible to gauge the success or failure of a network by comparing it to known solutions.

The task of balancing a single pole on a cart has been used in introductory texts on control systems, such as (Cannon, 1967; Friedland, 1986). These texts variously consider the problem both with and without friction, with and without time delays in the response of the "motor," and with both unlimited and bounded track lengths.

The multiple pole and the jointed pole problems have also been addressed as a control theoretic problem, though to a much lesser extent. A control theoretic solution to balancing multiple poles on a single cart forms the core of a Stanford University doctoral dissertation, (Higdon, 1964). This work describes the regions of controllability and of stable chatter for a bang-bang controller. Similarly, a second Stanford University doctoral dissertation, (Schaefer, 1965), addresses the jointed pole problem.

1.1.2 Neural Networks

Neural networks and their predecessors have a long tradition of addressing the (single) pole balancing problem. The ADALINE model was trained to balance poles using the Widrow-Hoff LMS algorithm nearly thirty years ago (Widrow & Smith, 1963, Widrow, 1987). These simple systems were able to perform optimal bang-bang control using a four-component state vector composed of the cart position and velocity and the pole angle and angular velocity.

More recently, the now classic papers on reinforcement learning addressed the pole balancing problem, (Barto et al., 1983; Michie & Chambers, 1968a, 1968b). In these papers an "adaptive critic" is used to train the network.

Pole balancing continues to be a standard research task for areas including training networks with a teacher (Guez & Selinsky, 1988) and adaptively coding sensor ranges (Rosen et al., 1990).

Recent neural network control papers that address pole balancing consider simultaneously learning the forward model and control (Jordan & Jacobs, 1990), and dynamic reinforcement learning (Schmidhuber, 1990).

All of these works focus solely on the single unjointed pole balancing problem.

1.1.3 Genetic Algorithms

GA's have been used extensively for creating neural networks, e.g. (Harp et al., 1989). A good overview of evolving neural networks with an emphasis on combining GA and back-propagation learning is provided by (Belew et al., 1990).

GA's have been used to develop single pole balancers. There is a section in Goldberg's doctoral dissertation, (Goldberg, 1983), which discusses the pole balancing problem. Also, (Koza, 1990) evolves LISP S-expressions to solve a number of problems including pole balancing.

1.1.4 Other Past Work

The pole balancing problem has also been used to demonstrate the power and versatility of other computing paradigms. Of particular interest is the work on cellular automata with a steepest descent learning procedure used to balance both a single pole and a jointed pole, (Lee et al., 1990a, 1990b).

1.2 CONTROL THEORY

This section provides an introduction to the portions of control theory that are of particular relevance to this paper. Those interested in pursuing the area further should consult a text in the area, such as (Anand, 1984; Cannon, 1967).

In general, a control system receives one or more inputs and attempts to cause a physical system to produce matching outputs. Based on the inputs and the current state of the physical system the control system produces a control signal that it sends to the physical system. The difference between the inputs to the control system and the outputs of the physical system forms an error vector that indicates the success of the controller, see Figure 2.

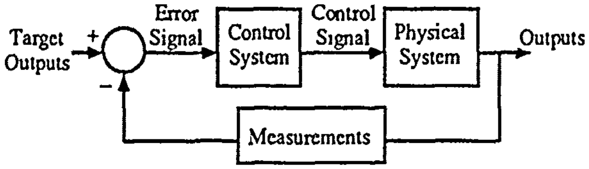


Figure 2: Basic Control System

In this paper the control system is implemented by a neural network, the control signal is a scalar specifying the force to apply to the cart, and the physical system is a computer simulation of the cart and pole(s). Further, since the goal is to keep the pole(s) vertical and

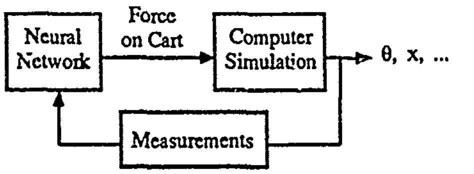


Figure 3: Neural Network Pole Balancer

the cart centered, the error signal is proportional to the cart position and the pole angle(s), see Figure 3.

The state of the physical system is specified by the measurements that it receives (and any "memory" of past states that it may contain). This so called "state vector" and the corresponding state space that it defines is fundamental to whether the system is capable of being controlled. The control system must receive (or be able to compute) some minimal state vector to be able to control the physical system.

A common but simple control scheme is proportional control, in which the control signal is proportional to the error signal. In neural network terms proportional control requires only a single computing neuron with a linear transfer function. Owing to the lack of velocity information, proportional control generally overshoots its target output, resulting in oscillations. The magnitude of the proportional controller's gain balances the steady state error of the system against this overshoot and corresponding oscillation frequency. This problem can be overcome by combining proportional control with derivative control and/or integral control, in which the derivative or integral of the state vector is also used to compute the control signal. Most pole balancing neural networks use some a form of proportional and derivative control.

By considering the Laplace transform of the system transfer function, that is the relationship between the desired and the actual output of the physical system, it is possible to derive the characteristic behavior of the control system. The characteristic behavior of the state variables is assumed to be periodic and is described by a sum of complex exponentials,

$$x(t) = \sum_i C_i e^{(\sigma - j\omega_i)t}$$

where $j = \sqrt{-1}$ and σ_i and ω_i are the real and imaginary parts of the poles of the transfer function.

It is helpful to plot the roots of the transfer function in the *s-space* defined by the real and imaginary parts of the complex exponentials. Roots in the right half-plane correspond to unstable behavior in the controller; the state variable $x(t)$ grows exponentially for $\sigma > 0$. Conversely, poles in the left half plane correspond to stable behavior, i.e., oscillations that die down. Figure 4 shows a plot in s-space of the roots of a second order system where both roots are in the left half plane and the corresponding system behavior. This is the behavior that we would hope to see from successful pole balancers.

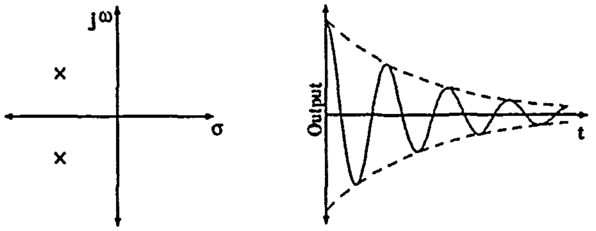


Figure 4: Sample Poles and Transient Response of Second Order System.

In any physical system there is a limit to the control signal that can be applied. Further, it can be shown that the fastest way to move a system to a desired state involves only applying maximal forces This so called *bang-bang* control is what is most often used by neural network controllers.

It is sometimes possible to determine under what conditions it is possible to bring the physical system to the desired state. For example, with the pole balancing problem, if the cart is at the far right of the track and the pole is leaning to the right then there is no way for the system to recover without either hitting the end of the track or allowing the pole to fall over. These *regions of controllability* have been derived for the multiple pole problem and will be presented below.

1.3 GENETIC ALGORITHMS

GA's, first introduced in (Holland, 1975), are a stochastic search method based loosely on the process of evolution and natural selection. With GA, the system to be evolved is encoded in a "gene," a string that can represent any of the class of systems. This gene then defines the GA search space. Sets or "populations" of these strings are created randomly and then a process of "natural selection" and "reproduction" continue until some goal is met.

It is this repeated selection and reproduction process that gives GA's their character. Selection of "parents" is based stochastically on their relative "fitness" in the population. The specific fitness measures and selection procedures that were used for this research are described in Section 3 below.

Reproduction is based on a series of "genetic operators." The three genetic operators used in this work are mutation, crossover, and inversion, shown pictorially in Figure 5. Mutation refers to randomly changing parts of a gene. Crossover takes part of the child's gene from one parent and the remainder of the gene

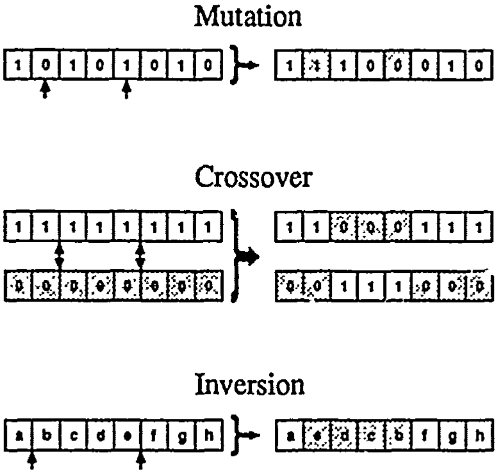


Figure 5: Genetic Operators

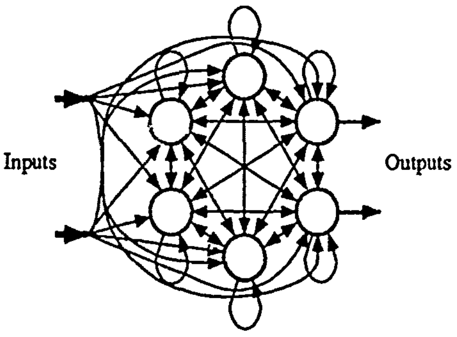


Figure 6: Fully Recurrent Network Topology

from the other parent. Inversion involves reversing the order of a fraction of the gene. Depending on how information has been coded in the gene, inversion can produce disastrous results, but in one case discussed below, inversion was used successfully.

The new text by David Goldberg, (Goldberg, 1989), is recommended to the reader interested in pursuing genetic algorithms further.

2 NEURAL NETWORKS

Neural nets are by their nature well suited to performing control tasks. Their distributed nature allows them to ignore noise and their general paradigm of weighing and combining information from many sources makes them good at integrating and filtering the many often redundant control signals that are available.

Many controllers require memory to compute derivatives or integrals. In neural network terms, this requires recurrent networks, i.e., networks that contain feedback. All the networks discussed in this paper are recurrent.

The bulk of this research was carried out on fully connected recurrent networks, shown in Figure 6. Every neuron in these networks receives an input that is the weighted sum of all the external inputs to the network and the previous state of all the neurons, including itself. A subset of the computing neurons are designated as output neurons and their outputs are used as control signals, but aside from this designation the output neurons are identical to the other neurons in the network.

Arbitrarily connected networks were also considered. Strings of structures containing *from*, *to*, and *weight* slots were used to evolve networks with arbitrarily complex interconnections. While this approach has been exceptionally successful for other hard problems, (Collins & Jefferson, 1990) it was only slightly more successful here.

Contrary to most other pole balancing work, in most of the simulations the nodes were not limited to binary outputs but could produce a range of outputs between -1 and 1. This not only gave the network more internal computational power but also allowed for finer control signals. This finer degree of control was helpful for the more difficult tasks addressed.

Some of the neural network design decisions were affected by the desire to run the simulations on both a conventional computer and a Thinking Machines Connection Machine. Weights, thresholds and node values were stored in single bits or in eight bit bytes. Bits were used to represent -1 and 1, and bytes were used to represent the values $-1, -\frac{253}{255}, -\frac{251}{255}, \ldots, \frac{253}{255}$, 1. Thus, all 256 values representable with one byte were used to represent numbers in the range -1 to 1, arranged symmetrically around zero. This representation has the interesting property of *not* allowing for any representation of zero. This made it impossible for these networks to exploit unstable equilibria since they were always producing "noise."

Both a sigmoid and a clipped linear transfer function were used at different times for the neurons, see Figure 7. The performance of the two transfer functions were qualitatively identical. Because of space and speed considerations, the results presented in this report were all computed using the clipped linear transfer function.

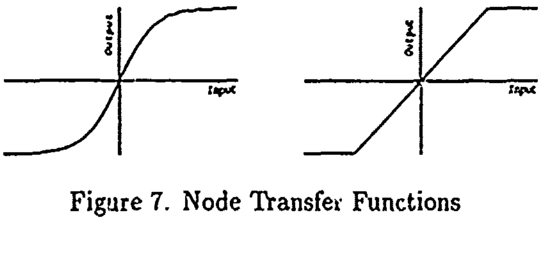


Figure 7. Node Transfer Functions

Network's genes were represented by binary strings. Fully connected networks were represented by a contiguous sequence of substrings for each neuron. Each node's substring contained one byte each for the node's initial value, threshold, and the weights of the connections into that neuron. A fully recurrent network with N nodes and M inputs requires $N(N+M+2)$ bytes.

Arbitrarily connected networks were represented by two substrings. The first substring contained the initial value and threshold for each node and the second substring contained an arbitrary number of 3-byte arc specifications (representing the *from*, *to*, and *weight* of each arc).

3 GENETIC ALGORITHM

GA's are well suited to creating neural network control systems. Many control systems require memory and therefore recurrent neural networks. Also the "correct" control signal is not always known ahead of time. GA's are able to deal successfully with both of these difficulties.

Recurrent networks pose special problems for gradient descent learning techniques that are not shared by GA's. With gradient descent learning it is generally necessary to correlate causes and effects in the network so that nodes and weights that cause the desired output are strengthened. But with recurrent networks the cause of a state may have occurred arbitrarily far in the past. Conversely, since GA's are only concerned with a scalar fitness of the network, the question of what caused any particular network state to occur is considered only in that the resulting state is desirable.

This inherent strength of GA's is in some ways also their weakness. While GA's do not rely directly on the cause and effect relations of nodes, ignoring this information when it does exist can make them inefficient. Fitness functions can be created to reflect much of our understanding of what the network should be doing, but in general GA's ignore much useful information.

The specific details of GA's vary widely between implementations. The remainder of this section describes specific fitness functions, selection criteria, and the genetic operators that were used. The basic use of these functions was introduced in Section 1.3 above.

The basic fitness of any network controller is defined simply as the time that it was able to keep the pole(s) from falling down or the cart from hitting an end of the track. "Falling down" means having the angle of the pole (from the vertical) pass some cutoff value. Cutoff angles ranging from 1° to 90° were tried, with little qualitative difference. In general the larger the angle the longer the evolving process took, but the less interesting the behavior that the balancer could exhibit. The data and plots in this report were made with a cutoff angle of 15°.

While the networks quickly evolved to keep the pole from passing the cutoff angle, it often took considerably longer to discover the relevance of the ends of the track. This was especially true for controllers that were able to balance the pole for hundreds of thousands of time steps without ever approaching the ends of the track. Therefore, fitness functions that penalized the network for leaving the center of the track were sometimes used. These functions were of the form

$$f(t+1) = f(t) + 1 - \lambda\left(\frac{x}{L}\right)^2$$

where $f(t)$ is the fitness at time t, x is the distance of the cart from the center of the track, L is half the length of the track, and $0 \leq \lambda \leq 1$ is a constant factor determining the degree of penalty. Also, when one or no inputs were provided to a network it was useful to *force* the network to generate the missing state variables. In these instances a similar penalty was given for not producing the correct value of the remaining state variables.

Parent networks were selected based on their fitness. The selection criteria used were determined in part by the desire to implement them efficiently on a Thinking Machines Connection Machine. In standard GA, a gene's likelihood of being selected is proportional to that gene's fitness relative to the fitness of the rest of the population. In these simulations a network was selected based on the *rank* of the gene's fitness. This both avoids problems having to do with translation invariance of fitness values[1] and allows parents to be selected without considering individual network's fitness values once the networks have been ranked.

Two selection criteria were used in this research. In the first method, parents were selected uniformly from a fixed percentage of the best networks, (Jefferson et al., 1990). In the second method a "selection power," α was chosen. A real valued random number was chosen from the range $(0, N^{1/\alpha})$, where N is the population size, and the resulting random variable was then raised to the α power. This allowed a continuous "knob" which controlled the selection process, the larger the α the more likely that only the best performing networks would be chosen as parents.

Crossover and mutation were used on the fully connected recurrent networks. Crossovers were allowed to occur anywhere within the bit-string, i.e., byte bound-

[1] Selection is generally based on fitness relative to the population's average. This causes fitness scores to be sensitive to translation; that is, if your fitness is 2 when the population's average is 1 you will be selected as a parent more often than if your fitness is 1002 and the average is 1001. This sensitivity can cause problems controlling the convergence of GA's, particularly when the fitness functions are heuristic measures of performance.

aries were not respected. Mutations corresponded to random changes in individual bits.

All three genetic operators, crossover, mutation, and inversion, were used in the arbitrarily connected networks. Any crossover point in the fixed length initial substring of the gene was at the same location in both parents. But, in the variable length connection substring, crossover points were only required to be at the same point in a 3-byte connection specification in both parents. In this system mutations were effected by adding a random value in the range [−32, 32] to an individual byte. Inversion was only allowed in the connection specification substring and inverted only the order of the 3-byte connection specifications but not the bytes in a specification itself.

4 POLE BALANCING

This section describes the results of the pole balancing problems and the GA created networks that solve them.

4.1 SINGLE POLE

The traditional single unjointed pole balancing problem can be solved optimally with a proportional bang-bang controller, (Cannon, 1967; Widrow, 1987). The pole can be balanced by exerting a force F on the pole given by:

$$F = F_{max}\ \mathrm{sgn}(k_1 x\ +\ k_2 \dot{x}\ +\ k_3 \theta\ +\ k_4 \dot{\theta}), \qquad (1)$$

where F_{max} is the maximum force, x and $\dot{x}$ are the position and velocity of the cart[2], θ and $\dot{\theta}$ are the angle and angular velocity of the pole, and k_1, k_2, k_3, and k_4 are coefficients that depend on the masses and frictions of the system. Equation 1 is also the equation of a single four input linear thresholding neuron. Thus, it is possible to solve the standard pole balancing problem with a neural network of a single neuron.

Let us consider the control theory solution to the one pole problem. The assumptions that the mass of the pole is evenly distributed along its length and, for this first case, that there is no friction and that the angles and velocities are small[3], yield the equations

$$\begin{aligned} (M+m)\ddot{x}\ +\ ml\ddot{\theta} &= F \qquad (2)\\ ml\ddot{x}\ +\ \frac{4}{3}ml^2\ddot{\theta}\ +\ mgl\theta &= 0 \qquad (3) \end{aligned}$$

where M and m are the mass of the cart and pole respectively l is the *half* length of the pole, and $g = -9.8m/s^2$ is the acceleration due to gravity.

Simple proportional control of these coupled second order equations produces an oscillatory system. The left plot in Figure 8 shows the poles of the proportional control transfer function with different amounts of feedback. In this system the force is simply proportional to the angle of the pole.[4] The open loop system, that is without any feedback, contains a pole in the right half plane. That means, not surprisingly, that with no control signal the pole falls over. As the feedback is increased, the poles of the transfer function move along the real axis meeting at the origin and then separate along the $j\omega$ axis. At this point the pole balancer exhibits undamped oscillations.

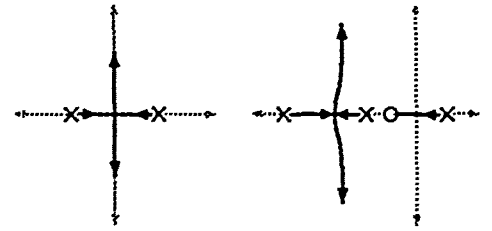


Figure 8: Proportional and Lead-Network Control for Balancing a Single Frictionless Pole.

In a real physical pole balancing system there are delays between the control signal and the application of the force on the cart. This, with the additional dynamic terms, tends to move the roots shown in the left plot of Figure 8 into the right half plane, causing the oscillatory system to be unstable. This effect is mitigated to some degree by friction in the system, but in general a proportional controller based on the angle alone is not stable. One method for stabilizing this system is the lead-network technique in (Cannon 1967, pp. 703-709). The right plot in Figure 8 shows the effective poles of the lead-network controller. This controller could be implemented using a four node recurrent neural network.

Of more interest to this paper is the single pole balancing problem with friction and without any of the small angle or small velocity approximations. This is the problem that has been addressed by (Barto et al., 1983) and others. The equations of motion for this system are given by Equations 9 and 10 in the appendix.

A slightly more difficult variation of this problem uses a two component state vector containing x and θ, but neither $\dot{x}$ nor $\dot{\theta}$. Stable control of this system requires the controller to compute an estimate of the velocities.

Using GA and fully recurrent networks the two input single pole balancing problem required less than a dozen generations to evolve to a point where most of the controllers can balance a pole indefinitely. Fig-

[2] The "dot" notation for time derivatives is used throughout this paper: $\dot{x} = dx/dt$ is the velocity of the cart, $\ddot{x} = d^2x/dt^2$ is the acceleration of the cart.

[3] The small angle approximation assumes that $sin(\theta) \approx \theta$ and $cos(\theta) \approx 1$ and the small velocity approximation assumes that factors containing $\dot{x}$ and $\dot{\theta}$ can be ignored.

[4] We are also assuming that the desired force never exceeds the system's limits.

ures 9 and 10 show the maximum, median, minimum, and a rectangle from the median of the larger half of the values to the median of the smaller half values of the values of the population's fitness at successive generations. Figure 9 used a fitness function that was simply the length of time that the pole remained balanced, while Figure 10 used a fitness function that was a function of the position of the cart. Both these simulations used a population of 511 fully recurrent networks, each network composed of six neurons. The selection factor was 10.0, and there were an average of 1.5 mutations per gene. In this and subsequent sections, a controller is considered successful if it can balance a pole for 10^6 time steps, approximately 5.5 hours of simulated time.

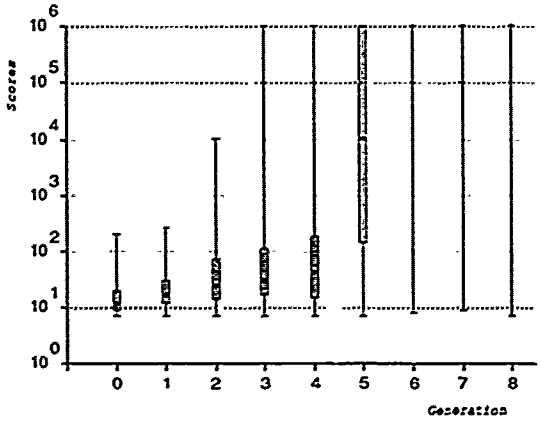


Figure 9: Evolution of a Two Input Single Pole Controller, Fitness is the Time the Pole is Balanced.

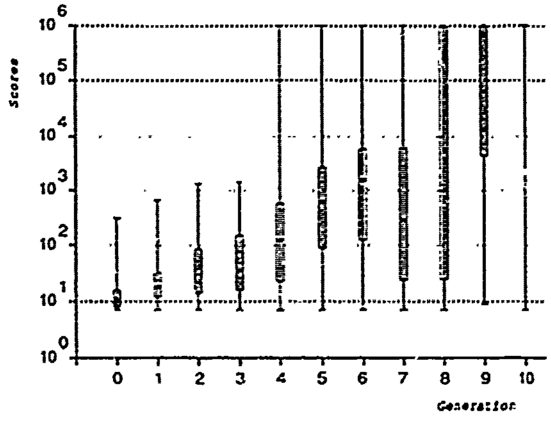


Figure 10: Evolution of a Two Input Single Pole Controller, Fitness is Based on Position of Cart.

Since the fitness of a controller was not directly affected by oscillations, it is not surprising that many of the successful controllers allowed the pole to oscillate during the entire simulation. In the early stages of evolution it was common to find controllers in which oscillations alternately grew and were damped out. These controllers becoming increasingly rare in later generations, presumably because they are not stable under GA reproduction.

Figures 9 and 10 show the difficulties of working with the single pole problem. Generally, populations evolved more quickly when fitness functions that penalized a network for being near an end of the track were used. In these particular examples however, the reverse is true. The single pole problem is so simple that compiling statistics becomes difficult.

Further variations of the single pole balancing problem were considered. Controllers that required only one input, the angle θ, were evolved. These tasks required slightly larger networks (ten neurons) and a consistent starting location (centered on the track). Using a fitness function that required two outputs, one for the force and the other for an estimate of the current x location, sped the rate of evolution considerably. These networks evolved in about thirty generations. Without such a fitness function the task was more difficult and required approximately 100 generations to evolve.

One further extension is to use a network with *no* inputs. For this problem the cart was consistently started in the center of the track with the pole vertical.[5] The most successful networks that were evolved for this problem were only able to balance a pole for a few hundred time steps. Even when a fitness function that required the network to produce an estimate of x and θ was used, these estimates quickly deteriorated (due largely to "round-off error"), causing the controller to fail.

A variation of the single pole balancing problem that surprisingly was *not* difficult considered variable length poles. Two related problems were investigated. In the first problem the length of a pole was randomly drawn between 0.1 and 1 meters but then fixed for the duration of the simulation. In the second, the length of the pole varied between 0.1 and 1 meter in a slow random fashion throughout a simulation. In both cases the mass of the pole varied in proportion to the length, that is the pole did not compress but rather the end "evaporated." While this task was slightly more difficult than the standard fixed length pole problem (a 10 neuron fully recurrent network requiring 25 to 30 generations with a selection power of 4 to evolve), there was no evidence that the network ever computed an estimate of the pole's length as anticipated. Further, most good standard pole balancers were able to balance the variable length pole systems for at least 10^4 time steps.

[5] Note that while this is an (unstable) equilibrium point, because the neurons used were unable to produce an output of zero (see Section 2) the network was not able to make use of this degenerate solution.

4.2 MULTIPLE POLES

An interesting and considerably more challenging version of the pole balancing problem involves balancing more than one pole on the same cart, a two pole example is shown in Figure 11. As long as the poles are of different lengths they will react differently to a force applied to the cart and can be balanced simultaneously. The equations of motion for the multiple pole problem are almost identical to the standard single pole balancing problem and are shown as Equations 11 and 12 in the appendix.

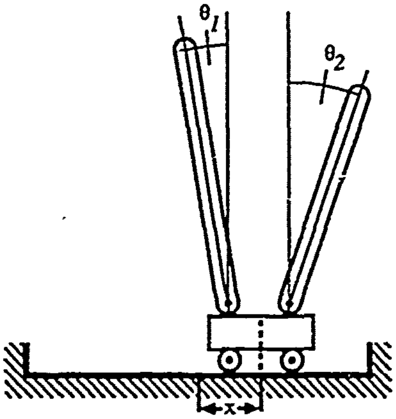


Figure 11: Double Pole Balancing Problem

It is worth taking a moment to examine what is involved in solving this problem. Equation 12 shows that the angular acceleration of a pole for a given $\ddot{x}$ is greater the more vertical the pole and is inversely proportional to the length of the pole Therefore, if the shorter pole is vertical and the longer pole is tilted, in order to bring both poles vertical a force must be applied so the longer pole leans over even further until the faster rotating shorter pole passes it. Then, when the shorter pole is leaning sufficiently more than the longer pole, the opposite force is applied and both poles are brought upright together. In order to balance both poles, the controller must make the current state of the system "worse" by leaning two nearly vertical poles over. Figure 12 shows the output of a neural network controller executing this maneuver.

It is possible to derive the conditions in which the poles are able to be returned to their upright positions at the origin. This region of controllability is derived for the multiple pole balancing problem in (Higdon, 1964). A brief summary of Higdon's observations is presented below.

The size of the region of controllability is determined by the ratio of the natural frequencies of the poles. The natural frequencies of the poles are given by the eigenvalues of the state vector's equations of motion.

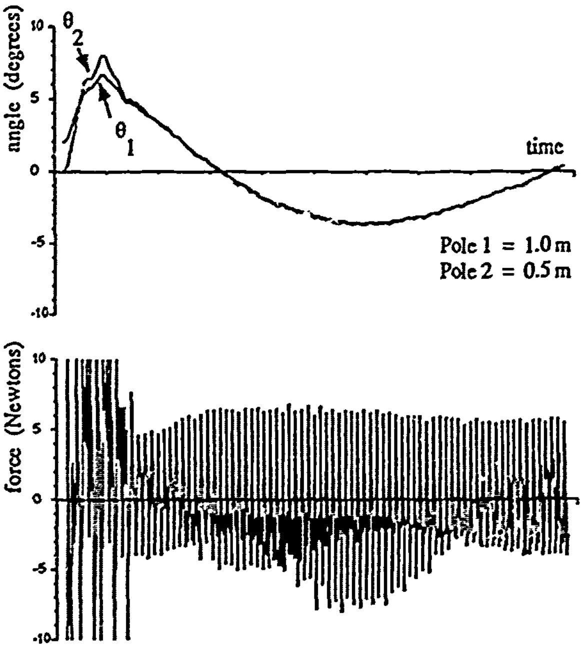


Figure 12: Control of Two Poles on One Cart

For the two pole problem these eigenvalues are [6]

$$\lambda_2 = 3\sqrt{\frac{g}{l_1}} \quad (4)$$

$$\lambda_4 = 3\sqrt{\frac{g}{l_2}} \quad (5)$$

$$\lambda_6 = \frac{\mu_c}{M} \quad (6)$$

The region of controllability can be described by a small number of "generalized coordinates." For the two pole problem these coordinates are given by

$$y_2 = -\theta_1 - \frac{1}{3}\sqrt{\frac{l_1}{g}}\,\dot{\theta}_1 + \frac{\mu_c/Mg}{1-(\mu_c/M)\sqrt{\frac{l_1}{g}}}\,\dot{x} \quad (7)$$

$$y_4 = -\theta_2 - \frac{1}{3}\sqrt{\frac{l_2}{g}}\,\dot{\theta}_2 + \frac{\mu_c/Mg}{1-(\mu_c/M)\sqrt{\frac{l_2}{g}}}\,\dot{x} \quad (8)$$

Figure 13 shows the region of controllability in terms of the generalized coordinates y_2 and y_4 for different values of λ_2/λ_4, including the two degenerate cases $\lambda_2/\lambda_4 = 1$ (equal length poles, cannot be controlled unless $y_2 = y_4$) and $\lambda_2/\lambda_4 = 0$. The region for a particular ratio is bounded by the symmetric lines around $y_4 = y_2$. The important point to notice is that the region of controllability shrinks quickly as the pole lengths approach one another.

[6] The same variable names are used here as in the appendix.

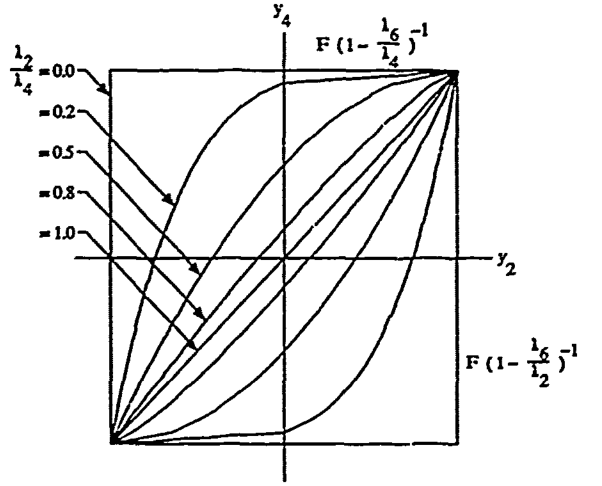


Figure 13: Regions of Controllability for 2 Poles

Network controllers for the two pole problem were evolved for the case where the poles had lengths of 1.0 and 0.1 meters ($\lambda_2/\lambda_4 = \sqrt{0.1/1.0} \approx 0.3$). This was a difficult task that required over 150 generations of a population of 2048 ten node fully recurrent neural networks.

Evolution was then continued while the shorter pole was lengthened by 1% increments until it was 0.9 meters long ($\lambda_2/\lambda_4 = \sqrt{0.9/1.0} \approx 0.95$). At this point the region of controllability is approximately 0.1° for vertical stationary poles. The process of slowly adjusting the pole length was similar to the "shaping" process that has been used with gradient descent learning paradigms, (Wieland & Leighton, 1988). It was most remarkable how *slowly* this shaping process proceeded. The controllers often required as much as a dozen generations to recover from the 1% change.

It is significant that the GA's were able to evolve controllers for the two pole problem, particularly when the pole lengths were 1.0 and 0.9 meters. The two pole problem was the hardest control problem considered in this paper. The resulting system corresponds to an extremely sensitive and delicate control problem.

4.3 JOINTED POLE

The final variation of the pole balancing problem considered a jointed pole, shown in Figure 14. As with the multiple pole problem, as long as the lengths and therefore the natural frequencies of the poles are sufficiently different it is possible to balance the system. The equations of motion for a pole with a single joint are shown as Equations 13, 14, and 15.

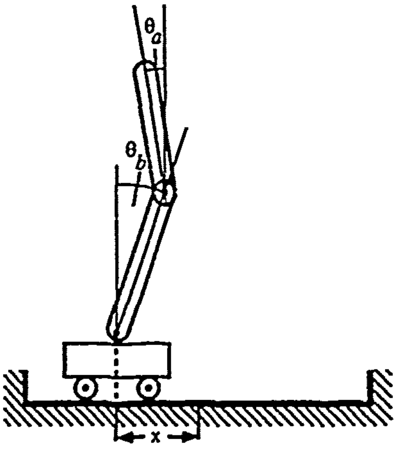


Figure 14· Jointed Pole Balancing Problem

The jointed pole problem proved to be simpler to evolve than the multiple pole problem. Unlike the multiple pole problem, the control signal that brought the poles closer to vertical was always the correct signal to use. For this reason it was possible to strongly favor selecting parents from a small set of best performers. 20 to 30 generations were required to evolve controllers for the jointed pole problem when the bottom pole was 1 meter and the top pole was 0.1 meter. A population of 512 ten node networks was used with selection coming either from the top 1% of the population or with a selection factor of 10.

5 DISCUSSION

This paper discussed a series of difficult control problems and the ways that GA's have been successfully used to create neural networks to solve those problems. With the sole exception of the difficult, and to my knowledge previously untried, problem of balancing a pole while receiving *no* feedback, networks were successfully created to solve each of the problems posed.

In most control theoretic work it is necessary to linearize the equations of motion by making small angle and small velocity approximations. This generally gives good results that are valid over the range of interest. However, one advantage of evolving controllers is that there is no restriction on the complexity of the model that can be controlled. This work has made a point of using accurate physical models, complete with friction and higher order velocity terms.

In general, the classic pole balancing problem was too simple a problem to be used with GA's and neural networks. The single pole balancing problem with two inputs is slightly more difficult, requiring the network to compute derivatives of its inputs. The next harder problem is the single input problem that required the network to estimate its current position, effectively requiring it to compute an integral, in addition to the derivatives. The multiple pole problem was the author's favorite. The equations of motion are simple and the problem can be made as challenging as desirable by adjusting the ratio of the pole lengths. The equations of motion for the jointed pole problem are considerably more complex but the underlying task was found to be relatively simple to evolve.

The difficulty that was encountered while "shaping" the multiple pole system points out an underlying difficulty with exclusively using GA's for creating neural networks. Even though small changes were made in the task, considerable effort was required by the GA to accommodate those changes. This is in sharp contrast to gradient descent learning paradigms that quickly adjust to small changes. GA's and gradient descent learning techniques move through the neural network search space in different but complementary ways. An ideal system would combine GA and gradient descent learning techniques.

This work was intended to address whether recurrent neural networks could be created using GA's to do hard control problems. In general, once a controller could be repeatedly evolved, questions of whether that controller could have been created more efficiently or with a smaller network were not pursued. Therefore the number of generations required to create the different controllers should be considered only in relative terms.

This work points to many areas for future research. Many of the details in comparing neural network controllers to theoretic control systems need to be completed. Similarly, the field of control is full of hard problems that could be addressed by these techniques. Finally, numerous questions still exist about the combined use of GA's and gradient descent learning. This paper has shown that hard control problems can be addressed; the next challenge is to harness that power.

APPENDIX

DETAILS OF POLE BALANCING SIMULATIONS

The equations of motion for the standard pole balancing problem are[7]

$$\ddot{x} = \frac{F - \mu_c \operatorname{sgn}(\dot{x}) + \tilde{F}}{M + \tilde{m}} \qquad (9)$$

$$\ddot{\theta} = -\frac{3}{4l}\left(\ddot{x}\, cos\theta + g \sin\theta + \frac{\mu_p \dot{\theta}}{ml}\right) \qquad (10)$$

where $\tilde{F}$ is the effective force of the pole on the cart

$$\tilde{F} = ml\dot{\theta}^2 \sin\theta + \frac{3}{4} m \cos\theta \left(\frac{\mu_p \dot{\theta}}{ml} + g\sin\theta\right)$$

and $\tilde{m}$ is the effective mass of the pole

$$\tilde{m} = m\left(1 - \frac{3}{4}\cos^2\theta\right)$$

The meaning of the parameters and the values used during simulations of the single pole problem are shown in Table 1.

Table 1: Symbols Used in Equations 9 and 10

Sym.	Description	Value
x	Position of cart on track.	[-2.4, 2.4] m
θ	Angle of pole from vertical	[-15,15] deg.
F	Force applied to cart.	[-10,10] N
g	Gravitational acceleration.	-9.8 m/s^2
l	Half-length of pole.	0.5 m
M	Mass of cart.	1.0 kg
m	Mass of the pole.	0.1 kg
μ_c	Coefficient of friction of cart on track.	0.0005
μ_p	Coefficient of friction of the pole's hinge.	0.000002

Similarly, the equations of motion for N unjointed poles balanced on a single cart are

$$\ddot{x} = \frac{F - \mu_c \operatorname{sgn}(\dot{x}) + \sum_{i=1}^{N} \tilde{F}_i}{M + \sum_{i=1}^{N} \tilde{m}_i} \qquad (11)$$

$$\ddot{\theta}_i = -\frac{3}{4l_i}\left(\ddot{x}\, cos\theta_i + g\sin\theta_i + \frac{\mu_{pi}\dot{\theta}_i}{m_i l_i}\right) \qquad (12)$$

where $\tilde{F}_i$ is the effective force from the ith pole on the cart

$$\tilde{F}_i = m_i l_i \dot{\theta}_i^2 \sin\theta_i + \frac{3}{4} m_i \cos\theta_i \left(\frac{\mu_{pi}\dot{\theta}_i}{m_i l_i} + g\sin\theta_i\right)$$

and $\tilde{m}_i$ is the effective mass of the ith pole

$$\tilde{m}_i = m_i\left(1 - \frac{3}{4}\cos^2\theta_i\right)$$

The meaning of the parameters and the values used during simulations of the two pole problem are shown in Table 2.

The equations of motion for the jointed pole are not as concise. For a pole with a single joint:

$$\begin{aligned}\ddot{\theta}_b = \frac{3}{4l_b}\Big(& (g + \dot{\theta}_a(\dot{x} - v_{bx}))\sin\theta_b \\ & + (v_{by}\dot{\theta}_a - \ddot{x})\cos\theta_b \\ & - \frac{\mu_b\dot{\theta}_b}{m_b l_b} \\ & - 2l_a\ddot{\theta}_a\cos(\theta_a - \theta_b)\Big) \qquad (13)\end{aligned}$$

[7] These equations are in a different but equivalent form to what has generally been used elsewhere.

Table 2: Symbols Used in Equations 11 and 12

Sym.	Description	Value
x	Position of cart on track.	[-2.4, 2.4] m
θ	Angle of pole from vertical	[-15,15] deg.
F	Force applied to cart.	[-10,10] N
g	Gravitational acceleration.	-9.8 m/s^2
l_i	Half-length of the ith pole.	[0.0,0.5] m, $l_1 = 0.5$ m
M	Mass of cart.	1.0 kg
m_i	Mass of the ith pole.	[0.0,0.1] kg, $m_i = l_i/5$
μ_c	Coefficient of friction of cart on track.	0.0005
μ_{pi}	Coefficient of friction of the ith pole's hinge.	0.000002

$$\ddot{\theta}_a = \frac{\Theta_{num}}{l_a\left(\frac{4}{3}m_a + m_b(1+3\sin^2(\theta_a-\theta_b))\right)} \tag{14}$$

where

$$\begin{aligned}
\Theta_{num} = {} & \\
& g\,[(m_a+m_b/2)\sin\theta_a \\
& \quad + \frac{3}{2}m_b\cos\theta_b\sin(\theta_a-\theta_b)] \\
& - \ddot{x}(m_a+2m_b)\cos\theta_a \\
& + \frac{3m_b v_{by}}{4l_a}(\dot{x}-v_{bx})(2\cos^2\theta_b - 1) \\
& + \frac{3m_b}{4l_a}((\dot{x}-v_{bx})^2 - v_{by}^2)\sin\theta_b\cos\theta_b \\
& + \frac{3}{2}(m_b\ddot{x}\cos\theta_b + \frac{\mu_b\dot{\theta}_b}{l_b})\cos(\theta_a-\theta_b) \\
& - 2m_b l_b\dot{\theta}_b\sin(\theta_a-\theta_b) \\
& - \frac{\mu_a\dot{\theta}_a}{l_a}
\end{aligned}$$

and

$$\ddot{x} = \frac{\mathrm{X}_{num}}{\mathrm{X}_{den}} \tag{15}$$

where

$$\begin{aligned}
\mathrm{X}_{num} = {} & \\
& (F-\mu_c\mathrm{sgn}(\dot{x}))(8m_a - 6m_b(4-3\cos^2(\theta_a-\theta_b))) \\
& + \frac{3}{2}[(m_a+2m_b)(2m_a+m_b)\sin(2\theta_a) \\
& \qquad -m_a m_b\sin(2\theta_b)]\,g \\
& - 6\frac{\mu_a\dot{\theta}_a}{l_a}(m_a+2m_b)\cos\theta_a \\
& - 6\frac{\mu_b\dot{\theta}_b}{l_b}(m_a+3m_b)\cos\theta_b \\
& + 9\frac{\mu_b\dot{\theta}_b}{l_b}(m_a+2m_b)\cos\theta_a\cos(\theta_a-\theta_b) \\
& + 9\frac{\mu_a\dot{\theta}_a}{l_a}m_b\cos\theta_b\cos(\theta_a-\theta_b) \\
& - 6m_b l_b\dot{\theta}_b^2(2m_a+m_b)\cos\theta_a\sin(\theta_a-\theta_b) \\
& + 6m_a m_b v_{by}\dot{\theta}_a\cos^2\theta_b \\
& + 3m_a m_b\dot{\theta}_b(\dot{x}-v_{bx})\sin(2\theta_b) \\
& + \frac{9m_a m_b}{4l_a}[2v_{by}(v_{bx}-\dot{x})\sin\theta_a\sin\theta_b \\
& \qquad -(v_{by}^2+2(\dot{x}-v_{bx})^2)\sin(\theta_a-\theta_b)]\cos\theta_b \\
& - \frac{9m_a m_b}{4l_a}v_{by}^2\sin\theta_b\cos(\theta_a+\theta_b) \\
& - 4m_a(m_a+2m_b)v_{by}\dot{\theta}_a \\
& - \frac{3m_b}{4l_a}[(\dot{x}-v_{bx})^2(2m_a+4m_b) \\
& \qquad + v_{by}^2(5m_a+4m_b)]\sin\theta_a \\
& - 2m_b l_b\dot{\theta}_b^2(4m_a+3m_b)\sin\theta_b
\end{aligned}$$

$$\begin{aligned}
\mathrm{X}_{den} = {} & -3m_a m_b\cos^2\theta_b \\
& + 3(m_a+2m_b)(2m_a+m_b)\cos^2\theta_a \\
& + 9m_b[2M\cos^2(\theta_a-\theta_b) \\
& \qquad -m_a\sin^2(\theta_a-\theta_b)] \\
& - 8(m_a+3m_b)M \\
& - 8(m_a+m_b)(m_a+3m_b/4) \\
v_{bx} = {} & \dot{x} + 2\,l_a\dot{\theta}_a\cos\theta_a \\
v_{by} = {} & 2\,l_a\dot{\theta}_a\sin\theta_a
\end{aligned}$$

The single (unjointed) pole problems were modeled using the Euler method and a time step on 0.02 seconds. The multiple pole and jointed pole problems were modeled with a time step of 0.01 seconds and a fourth order Runga-Kutta model.

Acknowledgements

I would like to thank David Jefferson, Rob Collins, Adam King, Russ Leighton, Dave Touretzky, and most of all my wife Annette Mercer for their comments and support.

The figures and plots in this report were created using the publicly available *GoodNeWS* software developed by the Turing Institute, Glasgow, Scotland.

References

D. K. Anand. (1984) *Introduction to Control Systems.* Oxford: Pergamon Press.

C. W. Anderson. (1989) "Learning to Control an Inverted Pendulum Using Neural Networks." *IEEE Control Systems Magazine*, 9(3):31-37.

A. G. Barto, R. S. Sutton, and C. W. Anderson. (1983) "Neuronlike Adaptive Elements That Can Solve Difficult Learning Control Problems." *IEEE*

Transactions on Systems, Man, and Cybernetics SMC-13:834-846.

R. K. Belew, J. McInerney, and N. N. Schraudolph. (1990) "Evolving Networks: Using the Genetic Algorithm with Connectionist Learning." Technical Report: #CS90-174, U. C. San Diego; La Jolla, CA.

R. H. Cannon, Jr. (1967) *Dynamics of Physical Systems.* New York: McGraw-Hill Book Company.

R. J. Collins and D. R. Jefferson. (1990) "An Artificial Neural Representation for Artificial Organisms." In R. Männer and D. E. Goldberg (eds.), *Proceedings, Parallel Problem Solving from Nature.* Berlin: Springer-Verlag (in press).

B. Friedland. (1986) *Control System Design, An Introduction to State-Space Methods.* New York: McGraw-Hill Book Company.

D. Goldberg. (1983) "Computer-Aided Gas Pipeline Operation Using Genetic Algorithms and Rule Learning." Ph.D. dissertation. Ann Arbor. University of Michigan.

D. Goldberg. (1989) *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA. Addison-Wesley.

A. Guez and J. Selinsky. (1988) "A Neuromorphic Controller with a Human Teacher." *IEEE International Conference on Neural Networks,* V. II, pp. 595-602.

S. A. Harp, T. Samad, and A. Guha. (1989) "Towards the Genetic Synthesis of Neural Networks." In J. D. Schaffer (ed.) *Proceedings of the Third International Convergence on Genetic Algorithms.* San Mateo, CA: Morgan Kaufmann Publishers.

D. T. Higdon. (1963) "Automatic Control of Inherently Unstable Systems with Bounded Control Inputs." Ph.D. Dissertation, Department of Aeronautics and Astronautics. Stanford University.

J. Holland. (1975) *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press.

D. Jefferson, R. Collins, C. Cooper, M. Dyer, M. Flowers, R. Korf, C. Taylor, and A. Wang. (1990) "The Genesys System. Evolution as a Theme in Artificial Life." In C. Langton, J. D. Farmer, S. Rasmussen, and C. Taylor (eds.), *Artificial Life II.* Reading, MA. Addison-Wesley (in press).

M. I. Jordan and R. A. Jacobs. (1990) "Learning to Control an Unstable System with Forward Modeling." In D. Touretzky (ed.), *Advances in Neural Information Processing Systems 2.* San Mateo, CA: Morgan Kaufmann.

J. R. Koza. (1990) "Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems." Technical Report. STAN-CS-90-1314, Stanford, California: Stanford University.

Y. C. Lee, S. Qian, R. D. Jones, C. W. Barnes, G. W. Flake, M. K. O'Rourke, K. Lee, H. H. Chen, G. Z. Sun, Y. Q. Zhang, D. Chen, and G. L. Giles. (1990a) "Adaptive Stochastic Cellular Automata: Theory." Technical Report: LA-UR 90-1229. Los Alamos, New Mexico: Los Alamos National Laboratory.

Y. C. Lee, S. Qian, R. D. Jones, C. W. Barnes, G. W. Flake, M. K. O'Rourke, K. Lee, H. H. Chen, G. Z. Sun, Y. Q. Zhang, D. Chen, and G. L. Giles. (1990b) "Adaptive Stochastic Cellular Automata: Applications." Technical Report. LA-UR 90-1227. Los Alamos, New Mexico. Los Alamos National Laboratory.

D. Michie and R. A. Chambers. (1968a) "BOXES. An Experiment in Adaptive Control." In E. Dale and D. Michie (eds.) *Machine Intelligence 2,* Edinburgh. Oliver and Boyd, pp. 137-152.

D. Michie and R. A. Chambers. (1968b) "'Boxes' as a model of pattern-formation." In C. W. Waddington (ed.) *Towards a Theoretical Biology,* vol. 1, *Prolegomena,* Edinburgh: Edinburgh University Press, pp. 206-215.

B. E. Rosen, J. M. Goodwin, and J. J. Vidal. (1990) "Adaptive Range Coding." *Neural Information Processing Systems, 1990* (to appear).

J. F. Schaefer. (1965) "On the Bounded Control of Some Unstable Mechanical Systems." Ph.D. Dissertation, Department of Electrical Engineering, Stanford University.

J. H. Schmidhuber. (1990) "Making the World Differentiable. On Using Supervised Learning Fully Recurrent Neural Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments." Technical Report: FKI-126-90, Technische Universität München; München, West Germany.

B. Widrow and F. W. Smith. (1963) "Pattern Recognizing Control Systems." In *Computer Information Sciences (COINS) Symposium,* 1963.

B. Widrow. (1987) "The Original Adaptive Neural Net Broom-Balancer." *Proceedings of IEEE International Symposium on Circuits and Systems,* pp. 351-357.

A. P. Wieland and R. R. Leighton. (1988) "Shaping Schedules as a Method for Accelerating Learning." *International Neural Network Society Meeting.*

Part IV

Temporal Processing

BACK-PROPAGATION, WEIGHT-ELIMINATION AND TIME SERIES PREDICTION

Andreas S. Weigend
Physics Department
Stanford University
Stanford, CA 94305, USA

David E. Rumelhart
Psychology Department
Stanford University
Stanford, CA 94305, USA

Bernardo A. Huberman
Dynamics of Computation Group
Xerox PARC
Palo Alto, CA 94304, USA

Abstract

We investigate the effectiveness of connectionist architectures for predicting the future behavior of nonlinear dynamical systems. We analyze the sunspot series as an example of a real world time series of limited record length. The problem of overfitting, particularly serious for short records of noisy data, is addressed both by using the statistical method of validation and by adding a complexity term to the cost function *(weight-elimination)*. We show why sigmoid units are superior in performance to radial basis functions for high-dimensional input spaces. The ultimate goal is prediction accuracy: we find that sigmoid networks trained with weight-elimination outperform traditional nonlinear statistical approaches. The prediction accuracy does not deteriorate when too many input units are used. Iterated single-step predictions are found to be better than direct multi-step predictions. Furthermore, we compare different sampling times (yearly and monthly), investigate the effect of preprocessing the data (square root and logarithmic transforms) and compare different error functions (corresponding to Gauss and Poisson statistics).

1 INTRODUCTION

In many instances, the desire to predict the future is the driving force behind the search for laws that explain the behavior of certain phenomena. Examples range from Newton's laws of motion to forecasting the weather and anticipating currency exchange rates.

The ability to forecast the behavior of a given system hinges on two types of knowledge. The first and most powerful one is knowledge of the laws underlying a given phenomenon. When this knowledge is expressed in the form of deterministic equations that can in principle be solved, the future outcome of an experiment can be predicted once the initial conditions are completely specified.

A second, albeit less powerful, method for predicting the future relies on the discovery of strong empirical regularities in observations of the system. The motion of the planets, the small amplitude oscillations of a pendulum, or the rhythm of the seasons carry within them the potential for predicting their future behavior from knowledge of their cycles without resorting to knowledge of the underlying mechanism.

There are problems, however, with the latter approach. Periodicities are not always evident, and they are often masked by noise. Even worse, there are phenomena – although recurrent in a generic sense – that seem random, without apparent periodicities.

We use feed-forward networks of the type introduced by Lapedes and Farber [LF87] to predict future values of time series by extracting knowledge from the past. In distinction to previous connectionist approaches for noise free, computer generated time series [LF87, MD89], we focus on noisy, real world data of limited record length. In this case, the problem of overfitting can become very serious. This problem is approached from two angles: by using internal validation [MB90] and by the method of weight-elimination [Rum88].

We analyze the time series of sunspots from the year 1700 to 1979, a benchmark used by many time series analysts. We show that the network leads to better predictions than the threshold autoregressive model of Tong and Lim [TL80], considered the best model in a recent review by Priestley [Pri88].

2 NETWORKS FOR TIME SERIES PREDICTION

Three ingredients are required to specify a model for short term prediction of time series.

1. Choose an embedding for the time series $\{x_t\}$: the short-term structure[1] can be captured by expressing the present value x_t as a function of the previous d values of the time series itself,
$$x_t = f(\text{past}) = \begin{cases} \mathbb{R}^d \to \mathbb{R} \\ (x_{t-1}, x_{t-2}, ..., x_{t-d}) \mapsto x_t \end{cases} .$$
The vector $(x_{t-1}, x_{t-2}, ..., x_{t-d})$ lies in the d-dimensional time delay space or **lag space**.
2. Approximate the points $\{x_t(x_{t-1}, x_{t-2}, ..., x_{t-d})\}$ by a smooth surface. Different approaches in time series prediction mainly differ in the choice of **primitives** (polynomials, splines, sigmoids, radial basis functions,...) and in the choice between one global fit in lag space vs. many local fits.
3. Choose a **cost function** that evaluates how well the points are approximated by the surface. The cost function reflects the assumptions about mesurement errors and statistics of the original data. Assuming Gaussian errors, the cost function is simply the the sum of the squared differences.

Then, given the embedding, the primitives and the cost function, find the parameters for the surface that minimize the cost function.

Once the surface has been determined, the prediction for the value following a point in lag space is given by the value of the surface above that point. The problem of prediction, usually framed as *extrapolation* in time, is re-framed for time invariant systems as *interpolation* in lag space. Following the approach by Lapedes and Farber [LF87], we train connectionist networks on examples from the past to find such surfaces.

2.1 ARCHITECTURE

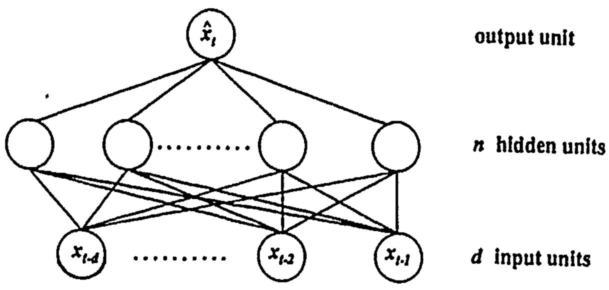


Figure 1: Architecture of a feed-forward network with one hidden layer. Units are shown as circles, connections as lines.

The networks are feed-forward networks with one hidden layer, as shown in Fig. 1. The abbreviation d-n-1 denotes the following network:

- The d *input units* are given the values $x_{t-1}, x_{t-2}, \ldots, x_{t-d}$.
- The n *nonlinear hidden units* are fully connected to the input units.
- The linear *output unit* is fully connected to the hidden units, producing the prediction $\hat{x}_t$ as the weighted sum of the activations of the hidden units.
- Output and hidden units have adjustable *biases*.
- The *weights* can be positive, negative or zero.
- There are no direct connections from input to output that skip the hidden layer.

The nonlinearities are located in the *activation function* (or nonlinear transfer function) of the hidden units. We used two activation functions: logistic activation, giving rise to sigmoid units, and Gaussian activation, giving rise to radial basis units.

2.1.1 Sigmoids vs. Radial Basis Functions

The success in learning crucially depends on the specific functions used to construct the smooth prediction surface above the d-dimensional input space. Specifically, it is important to understand how the input variables are treated in the two cases of sigmoids and decreasing radial basis functions.

Sigmoids. Let ξ_h denote the input of the network (including bias b_h) into a sigmoid (or logistic) hidden unit h,

$$\xi_h = \sum_{i=1}^{d} w_{hi} x_i + b_h = \vec{w}_h \cdot \vec{x} + b_h \quad . \qquad (1)$$

x_i stands for x_{t-i}, the value of input i, and w_{hi} is the weight between input unit i and hidden unit h. The contribution $\vec{x} \cdot \vec{w}_h$ is the projection of the input vector $\vec{x} = (x_1, x_2, ..., x_d)$ on the weight vector $\vec{w}_h = (w_{h1}, w_{h2}, ..., w_{hd})$.

The activation S_h of a hidden unit is given by

$$S_h = S(\xi_h) = \frac{1}{1+e^{-a\xi_h}} = \frac{1}{2}\left(1 + \tanh \frac{a}{2}\xi_h\right) \quad . \quad (2)$$

The sigmoid performs a smooth mapping $(-\infty, +\infty) \to (0, 1)$. The slope of the sigmoid, a, can be absorbed into weights and biases without loss of generality and is set to one.

Radial basis functions. A radial basis function (RBF) depends only on the distance $\eta = ||\vec{x} - \vec{\mu}_h||$ between the input, $\vec{x}$, and the center of the RBF, $\vec{\mu}_h$ (also of input dimension d),

$$f(\vec{x}) = f(||\vec{x} - \vec{\mu}_h||) = f(\eta) \quad . \qquad (3)$$

[1] The focus is on the short-term structure, since "chaotic" systems can be predicted on short time-scales, but not on long time-scales. This is discussed in [WHR90], see also [CFPS86, Sch88, Ger89, FS89, Cas89, EF90].

Choosing f to be *Gaussian* and $\| \ \|$ to be the Euclidean norm, the activation G_h of hidden unit h is given by

$$G_h = \exp\left(\frac{\|\vec{x} - \vec{\mu}_h\|^2}{-2\sigma_h^2}\right) = \exp\left(\frac{\sum_{i=1}^{d}(x_i - \mu_{h,i})^2}{-2\sigma_h^2}\right)$$

$$= \prod_{i=1}^{d} \exp\left(\frac{(x_i - \mu_{h,i})^2}{-2\sigma_h^2}\right) . \qquad (4)$$

The parameter σ_h indicates the width (or standard-deviation) of the Gaussian. It can be interpreted as the radius of the hyperspherical receptive field (in d-dimensional input space) of the hidden unit. The normalization of Gaussian RBFs is similar to that of sigmoids, *i.e.*, the activation lies between zero and one.

The crucial difference between RBFs and sigmoids lies in the treatment of multi-dimensional inputs. For Gaussian RBFs , as can be seen from Eq. (4), the inputs factor completely. Unless *all* inputs x_i are reasonably close to their centers $\mu_{h,i}$, the activation of hidden unit h is close to zero; an RBF unit is shut off by a single large distance between its center and the input in any one of the dimensions. This multiplicative feature resembles a logical AND. For sigmoids, there is no such factorization. a large contribution by one weighted input in the sum in Eq. (1) can often be compensated for by the contribution of other weighted inputs of the opposite sign.

This difference between sigmoids and RBFs increases with the number of input units. For one dimensional cases, the difference between fitting a function with sigmoids and Gaussians is not important [MD89, BL88]. For the Mackey-Glass time series, Moody and Darken [MD89] used four input units. Notice that thousands of RBFs were required to cover the four dimensional input space, compared to 40 sigmoid units (two layers with 20 each) in the network used by Lapedes and Farber [LF87] For equivalent performance in both cases, Moody and Darken showed that RBFs required a time series between 7 and 27 times longer due to the larger number of parameters.[2]

For real world data, we assume that the data set is noisy and limited in size. Some of the noise can be averaged out by a relatively large input dimension and a global approximation surface. If one is interested in optimal prediction, both the finite data set and the noise favor sigmoid units over radial basis functions, essentially due to the difficulty to fill high dimensional spaces with localized functions. This was confirmed in our experiments: none of the trials with up to 100 fully adaptive RBFs was successful on either series, whereas both series were learned reliably with significantly fewer sigmoids.

But even for sigmoid hidden units, the number of parameters is often comparable to the number of data points, entailing the danger of overfitting. This central issue is addressed in the following section on training.

2.2 TRAINING

We use the error back-propagation algorithm of Rumelhart *et al.* [RHW86] to train the network: the parameters are changed by gradient descent on the cost surface over the weights and biases. In the simplest case, assuming a Gaussian distribution for the errors, the cost function is the total residual variance, or sum of squared errors, for a set of examples, $\mathcal{S}$,

$$\sum_{k \in \mathcal{S}} (\text{target}_k - \text{prediction}_k)^2 = \sum_{k \in \mathcal{S}} (x_k - \widehat{x}_k)^2 \ , \qquad (5)$$

where x_k (target_k) is the true value of the time series at time k, and $\widehat{x}_k$ (prediction_k) is the output of the network for time k. This fitting error describes how well the points $\{x_k,\ k \in \mathcal{S}\}$ are approximated by the surface over the input space.

2.2.1 Overfitting

A serious issue in the application of a network to a problem domain is the size of the network as measured by the number of free parameters of the network. As for other methods of function approximation, such as polynomial, too many free parameters will allow the network to fit the training data arbitrarily closely, but will not necessarily lead to optimal prediction. Although there is no general method to determine the optimal size of the network for a particular task, there are statistical arguments which suggest that the number of training patterns required to fully determine the weights in a network is approximately proportional to the number of weights in the network [DSW, BH89]. A rule of thumb often cited is that the number of weights should be less than one tenth of the number of training patterns. With data sets of a few hundred patterns only, this constraint is quite restrictive.

We explored two methods for dealing with this general problem. The first method involves providing a large number of parameters for the network, but stopping training before the network has made use of many of its degrees of freedom . Such networks, whose number of weights is of the order of the number of training examples, are referred to as *oversized* networks. The second method involves a learning procedure seeking a *minimal* network capable of accounting for the input data. Both methods are discussed below.

[2] The advantage of RBFs is computational efficiency, obtained by only locally updating the relevant RBFs [MD89]. Sacrificing the flexibility of adaptive centers and widths, the remaining problem of determining the contribution of each RBF can be reduced to matrix inversion [BL88].

2.2.2 Oversized Networks need Validation

To a first approximation, the gradient learning process employed by the back-propagation procedure works as follows: initially, the hidden units in the network all do the same work, *i.e.*, they all attempt to fit the major features of the data (such as the average of the time series). As those features are accounted for, the major source of error in the network is determined by the second most important feature of the training data. The units then start to differentiate with some of them beginning to fit this second most important aspect of the data. This process of differentiation continues as long as there is error and as long as training continues. the *effective* number of degrees of freedom starts small and gets larger and larger as training proceeds. Assuming that sampling noise is small relative to other sources of variation in the data, we expect that early training will allow the network to fit the significant features of the data. It is only at later times that the network tries to fit the noise. A solution to the problem of overfitting is to stop training just before the network begins to fit the sampling noise.

The problem is to determine when the network has extracted all the useful information and is beginning to extract noise. The first method we employ is to split the whole available data set into three parts[MB90]. The earlier timespan, "the past", is divided into two sets: a **training set**, used for determining the values of the weights and biases, and a **validation set**, used for deciding when to stop training. The performance on the validation set is monitored. As long as this performance on the validation set improves, training continues. When it ceases to improve, we stop training. If we continued training, the oversized network would start to fit the noise. The last part of the record, the **prediction set**, acts as "the future". It is strictly set apart and never used in training. In particular, it must not be used to determine the stopping of the training process. Its only legitimate use is to estimate the expected performance in the future. For oversized networks, a validation set is necessary since the noise level in any real situation (as opposed to computer generated data) is not known a priori.

2.2.3 Minimal Networks through Weight-Elimination

The second method that addresses the problem of overfitting assumes that the network which generalizes best is the smallest network still able to fit the training data. Rumelhart [Rum88] proposed a method for accomplishing this within the framework of back-propagation learning. It has since been used by Hanson and Pratt [HP89], Chauvin [Cha90], Lang and Hinton [LH90] and others; an alternative is presented by Le Cun, Denker and Solla [LDS90]. The method of weight-elimination involves the extension of the gradient method to a more complex cost function. The idea is to begin with a network that is too large for the given problem, but associate a cost with each connection. The proposed cost function is the sum of two terms,

$$\sum_{k\in S}(\text{target}_k - \text{prediction}_k)^2 + \lambda \sum_{i,j} \frac{w_{ij}^2/w_0^2}{1+w_{ij}^2/w_0^2} \; . \quad (6)$$

The first term is the standard sum squared error term over the set of examples S. The second term describes a cost for each weight in the network and can be thought of as a complexity term [Ris89, Che90]. The sum extends over all connections in the network. The scale is given by w_0, for activations between 0 and 1, we set $w_0 = 1$. If the weight $|w_{ij}|$ is large compared to w_0, the cost is λ. For weights is close to zero, the associated cost is also close to zero.

The parameter λ represents the relative importance of the complexity term with respect to the performance term. If a given performance on the training set can be obtained with fewer weights, this cost function will encourage the reduction and eventual elimination of as many weights as possible. The learning rule is then to simply change the weights according to the gradient of the *entire* cost function with respect to the weights.

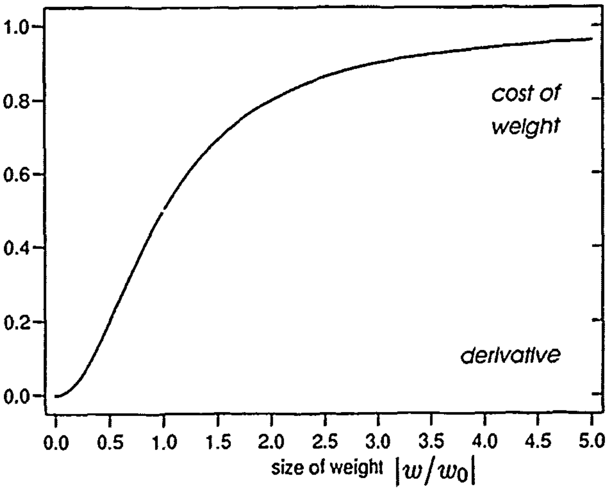


Figure 2: Cost for a weight in the network (solid line) and its derivative (gray line) with respect to the weight $|w/w_0|$ in units of λ.

The part of the change due to the additional term is proportional to the negative derivative of that term. The complexity part of the cost function and its derivative are shown as functions of $|w_{ij}/w_0|$ in Fig. 2. This complexity term is most important for medium size weights of order $w_0/2$.

There are a few technical points in the application of the procedure. It turns out to be useful to begin with λ at zero and to slowly increase the value of λ until performance begins to decline and thereafter increase

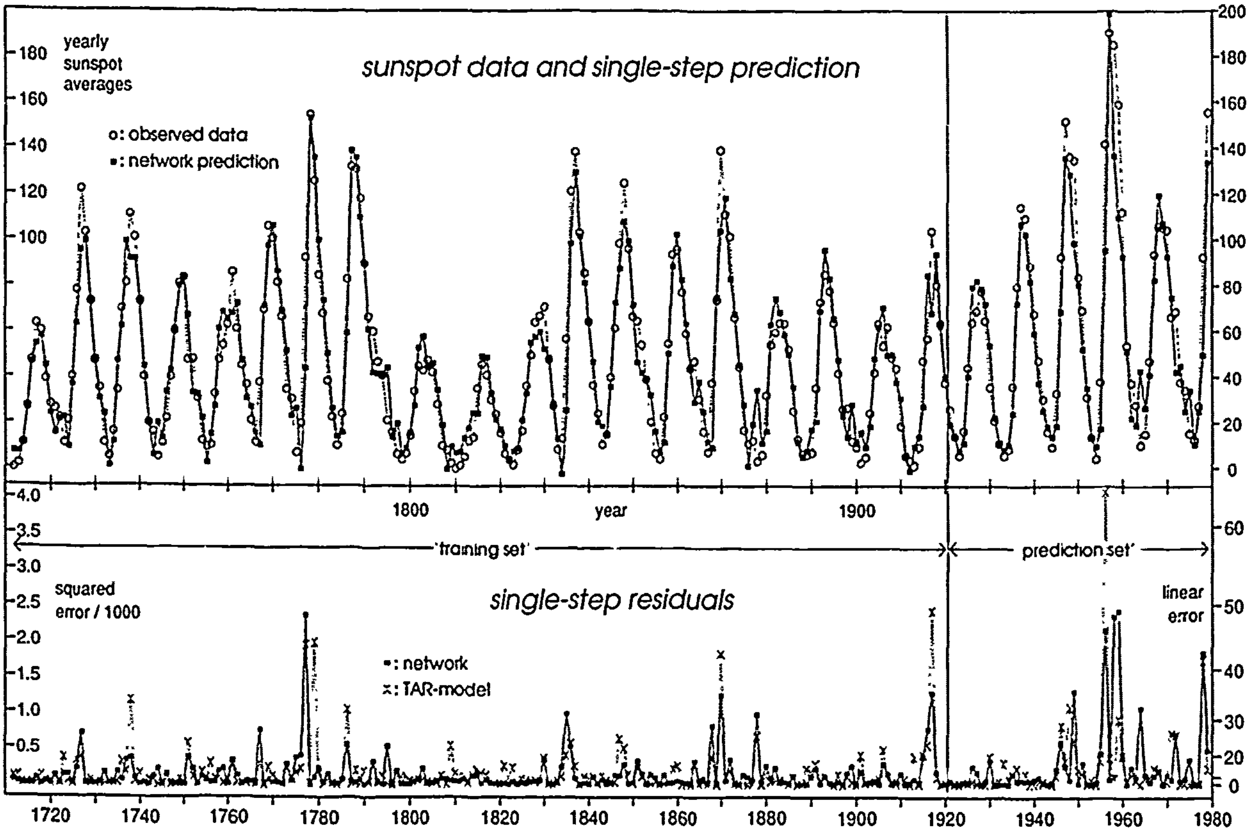


Figure 3: The sunspot series, the single-step network prediction and the residuals.

or decrease the value of λ so the error on the training set continues to decrease at a steady, slow rate. This is the procedure we call weight-elimination.

We are now equipped with the background to apply connectionist networks to the real world data set of observed sunspot numbers.

3 SUNSPOTS

Sunspots, often larger in diameter than the earth, are dark blotches on the sun. They were first observed around 1610, shortly after the invention of the telescope [Fou90]. Yearly averages have been recorded since 1700. The sunspot numbers are defined as $k(10g+f)$, where g is the number of sunspot groups, f is the number of individual sunspots, and k is used to reduce different telescopes to the same scale [Mar87]. The series is shown in Fig. 3. The average time between maxima is 11 years. Note, however, that the time between maxima ranges from 7 to 15 years.

The underlying mechanism for sunspot appearances is not exactly known. No first-principles-theory exists, although it is known that sunspots are related to other solar activities. For example, the magnetic field of the sun changes with an average period of 22 years. Sunspots usually appear in pairs, corresponding to magnetic dipoles. Sunspot pairs reverse their polarity from one cycle to the next, reflecting the underlying magnetic cycle.

The sunspot series has served as a benchmark in the statistics literature. Within the time delay or lag space paradigm, different models differ in the specific choice of the primitives for the surface above the input space. In the simplest case, a single hyperplane approximates the data points. Such a *linear autoregressive model* is a linear superposition of past values of the observable.

The evaluation of the network model, however, is carried out by comparison to a *nonlinear* model. In a recent evaluation of different models on the sunspot series, Priestley [Pri81, Pri88] favors the threshold autoregressive model (TAR) of Tong and Lim [TL80]. We here briefly sketch the model. For further discussion see Tong [Ton83, Ton90]. This globally nonlinear model consists of two local linear autoregressive models. Tong and Lim found optimal performance for input dimension $d = 12$. They used yearly sunspot data from 1700 through 1920 for training, and the data from 1921 to 1979 for evaluation of the prediction.

To make the comparison between network and TAR performance in sections 3.1 and 3.2 as close as possi-

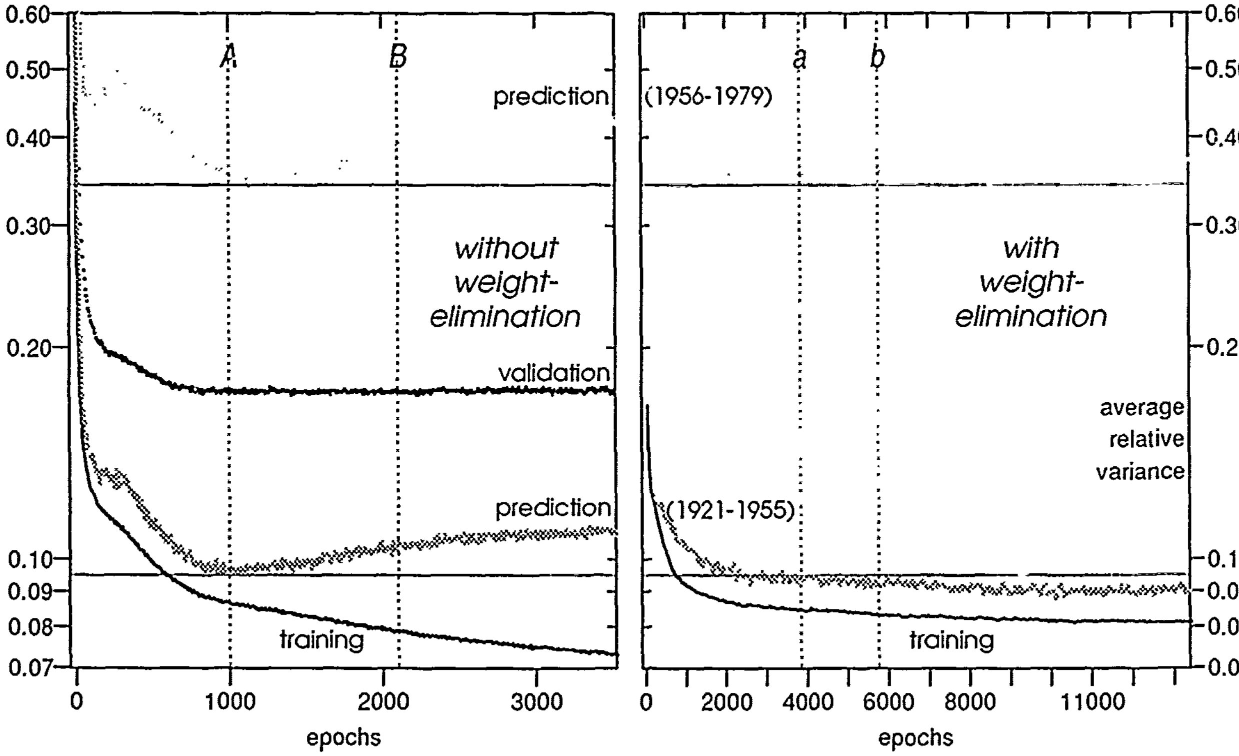


Figure 4: Learning curves of a 12-8-1 network as function of training time in epochs. The average relative single-step prediction variances are given for the training sets (solid lines) and the early (lower gray lines) and late prediction sets (upper gray lines) (as well as for the internal validation set for the network trained without weight-elimination on the left side). The vertical lines (A, B, a, b) indicate different stopping points.

ble, we use their exact data for training and evaluation, their choice for the input dimension, *i.e.*, 12 input units, and their error measure. The only remaining difference between the models lies in the choice of the primitives used for the fitting of the surface.

In Section 3 3, however, we present the result obtained by varying the number of input units from 1 to 41, and in Section 3.4 we use monthly data.

3.1 LEARNING THE TIME SERIES

3.1.1 Validation

The learning of the sunspot series of a 12-8-1 network is shown in Fig. 4 as a function of epochs. in one epoch the network sees each point from the training set exactly once. We define the average relative variance of a set $\mathcal{S}$ as

$$\operatorname{arv}(\mathcal{S}) = \frac{\sum_{k\in\mathcal{S}}(\text{target}_k \ \text{- prediction}_k)^2}{\sum_{k\in\mathcal{S}}(\text{target}_k - \text{mean})^2} = \frac{1}{\hat{\sigma}^2}\,\frac{1}{N}\sum_{k\in\mathcal{S}}(x_k - \hat{x}_k)^2 \quad . \qquad (7)$$

The averaging (division by N, the number of data points in set $\mathcal{S}$) makes the measure independent of the size of the set. The normalization (division by $\hat{\sigma}^2$, the estimated variance of the data) removes the dependence on the dynamic range of the data. This normalization implies that if the estimated mean of the data is used as predictor, arv = 1.0 is obtained.[3]

Expressed in terms of the *correlation coefficient* ρ between pairs of desired values and predictions, x_k and $\hat{x}_k$, the average relative variance is given by

$$\text{arv} = 2\,(1-\rho) \quad . \qquad (8)$$

This relationship is exact if and only if $\sum \hat{x}_k = \sum x_k$ and $\sum \hat{x}_k^2 = \sum x_k^2$.

In Fig. 4, the success in mastering the training set is indicated by the monotonic decrease of the lowest curve, indicating the fitting error. The case of standard error back-propagation without weight-elimination is shown in the left panel of Fig. 4. To get a feeling for the non-stationarity of the time series, the prediction set was split in two parts, 1921-1955 and 1956-1979. On

[3] If the variances of the individual sets differ, a choice has to be made. We have chosen to always used the variance of the entire record, $\hat{\sigma}^2 = \sigma^2_{\text{all}} = 1535$. Thus, in any of the three sets, a value of arv = 0.1 corresponds to an average *absolute* quadratic error of arv × σ^2_{all} = 0.1 × 1535. = 153.5 ≈ $(12.4)^2$. The alternative would have been to normalize each set by its own variance.

both prediction sets, the error first decreases, but then starts to increase. the network begins to use its resources to fit the noise of the training set, *i.e.*, it starts to pick out properties that are specific to the training set, but not present in the prediction sets. This overfitting leads to deteriorated generalization.

The question to be addressed is when the training should be stopped. Since prediction sets must not be used for this decision, a validation set is required for a statistically proper determination of the end of the training process. To get a feeling for the effect of the sampling error by picking a specific training set–validation set combination, we investigated several training set–validation set pairs.

The validation sets consisted of 22 years chosen at random from the time before 1920. Those points were removed in the corresponding training sets, reducing their size by 10 per cent. For the validation set of the example shown, the average relative variance approaches an asymptotic value, it happens not to increase. In this specific choice, the fitting of the noise of this training set happens to have no effect on the error of this validation set. Because the sunspot data set is rather small, different pairs of training and validation sets lead to results differing by factors of up to two. These variations are large compared to the variations due to different random initial weights and biases.

This approach is somewhat unsatisfactory because *(i)*, a certain part of the available training data cannot be used directly, *(ii)*, the results depend strongly on the specific pair of training set and validation set, and *(iii)*, it is not always entirely clear from the error of the validation set when the training process should be terminated. In the evaluation of the performance in Section 3.2, we compare the performance for two stopping points, A after 1000 epochs, and B after about 2100 epochs. As an alternative to the simple sum of squared errors cost function that requires a validation set, we next present the results of learning with weight-elimination.

3.1.2 Weight-Elimination

As in the case of back-propagation without weight-elimination, we start with a network large enough to guarantee a decrease of the error with training. The training curve for back-propagation with weight-elimination is shown in the right panel of Fig. 4. With the same learning parameters as without weight-elimination (zero momentum and a learning rate of 0.1), significant overfitting is avoided, even for training times four times longer. Since the entire training set is used, we are relieved from the uncertainty of a specific choice for a validation set. A decision, however, has to be made as to when the network reaches its asymptotic state. The performance of two solutions (*a* after 3900 epochs, *b* after about 5800 epochs) is compared in Section 3.2. It turns out that the exact stopping point is not important.

In the first 5000 epochs, the procedure eliminated the weights between the output unit and five of the eight hidden units. Since these five units did not receive signals in the backward pass any more, their weights to the input units subsequently decayed. In this sense, the weight-elimination procedure can be thought of as unit-elimination, removing the least important units

We analyzed the specific solution of the network that was stopped at point *b* and subsequently trained with a very small learning rate for a few epochs. (Details are given in the Appendix.) The main contribution to the first hidden unit comes from x_{t-9}, to the second hidden unit from x_{t-2}, and to the third hidden unit from x_{t-1}. In contrast to the output weights, only very few of the weights from the input units to the active hidden units were eliminated. The fact that the remaining weights are of relatively small size points to a relatively small use of the available nonlinearities. We show in [WHR90] how more elaborate measures of nonlinearity can be read off the network solution.

Predictions are obtained by adding the values of these three hidden units to the bias of the output unit. The solution of the network can thus be interpreted as a nonlinear transformation from the twelve-dimensional input lag space to the three-dimensional space of hidden units.

3.1.3 Avoiding Bad Solutions

For sigmoid units, good solutions were obtained in all of the hundreds of trials with different initial random weights and biases. We believe that this is mainly due to choosing relatively small initial weights, a relatively small learning rate, and a relatively large initial network size.

- If small initial random weights are chosen, the sigmoid units start out in their linear range. The gradient descent method moves the weights at the beginning towards the global minimum of the linear case. We chose the initial weights randomly with magnitude less than $\mathcal{O}(1/\sqrt{n})$ for activations in the unit interval, n denotes the fan-in into a given unit.
- Provided the learning rate is sufficiently small, nonlinearities are added only as needed. Typical learning rates are $\mathcal{O}(0.1/n) \cdots \mathcal{O}(1/n)$. No momentum term is used. The definitions of learning rate and momentum are given in [MR88].
- Starting with oversized networks rather than with tight networks seems to make it easier to find a good solution. However, as emphasized above, this approach requires a method to deal with the problem of overfitting.

Since the output is real valued, numerical differences between good and bad solutions are not as striking as in problems requiring binary outputs.

3.2 PREDICTING SUNSPOTS

So far, we have concentrated on the *learning* behavior of the network for different cost functions and activation functions. The ultimate goal, however, is to *predict* future values of the time series. In this section, we assess the predictive power of the network and compare it to the benchmark model. We first analyze single-step predictions and then turn to multi-step predictions.

3.2.1 Single-Step Prediction

The term *single-step prediction* (or *one-step-ahead prediction*) is used when all input units are given the actual values of the observed time series. To assess the single-step prediction performance, we use the average relative variance, arv, defined in Eq. (7). It is independent of the dynamic range of the data and of the record length of the series, allowing for comparisons across different time series.

The solution of the weight-eliminated network with sigmoid hidden units, explicitly given in the Appendix, gives

$$\text{arv}(\textit{train}) = 0.082\ , \quad \text{arv}(\textit{predict})_{1921-1955} = 0.086\ .$$

The corresponding values for the TAR model are

$$\text{arv}(\textit{train}) = 0.097\ , \quad \text{arv}(\textit{predict})_{1921-1955} = 0.097\ .$$

As can be seen by comparing this measure for the network with the TAR model, the single-step prediction qualities of the network and the benchmark model are comparable. Despite this similarity, however, significant differences will appear for predictions further than one step into the future.

3.2.2 Multi-Step Prediction

There are two ways to predict further than one step into the future. We first present the results of iterated single-step predictions and subsequently turn to direct multi-step predictions. In iterated single-step predictions, the predicted output is fed back as input for the next prediction and all other input units are shifted back one unit. Hence, the inputs consist of *predicted* values as opposed to actual observations of the original time series. The predicted value for time t, obtained after I iterations, is denoted by $\widehat{x}_{t,I}$.

The prediction error will not only depend on I but also on the time $(t-I)$ when the iteration was started. We wish to obtain a performance measure that smooths statistical fluctuations of a specific starting time, we average over M starting points and define the average **relative I-times iterated prediction variance** to be

$$\frac{1}{\hat{\sigma}^2}\,\frac{1}{M}\sum_{m=1}^{M}\left(x_t-\widehat{x}_{t,I}\right)^2\ . \qquad (9)$$

This measure is shown as a function of the number of iterations in Fig. 5. The average is taken from 1921, immediately following the end of the training period, through 1955. The differences between the different network solutions within each plot are not significant; they only indicate the spread of network performances.

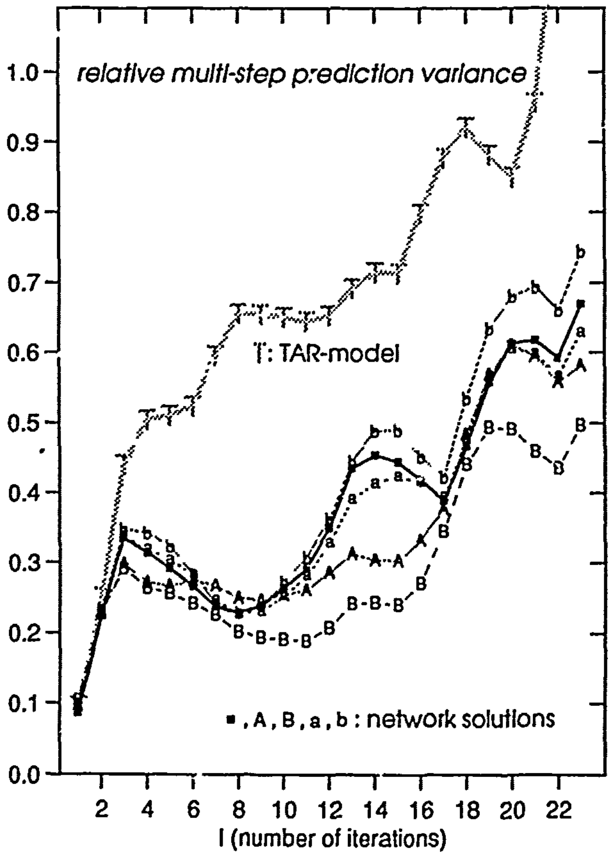


Figure 5: Multi-step prediction error (*average relative I-times iterated prediction variance*) for the sunspot series as function of the prediction time. Gray Ts indicate the performance of the TAR model. *A*, *B*, a, *b* refer to the different stopping points, shown in Fig. 4. Black squares show the performance of the weight-eliminated network given in the Appendix.

An alternative to this *iterated single-step* prediction is *direct multi-step* prediction: the network is trained to predict directly several steps ahead. On the sunspot data set, the prediction error for direct multi-step prediction was significantly worse than the error for iterated single-step prediction.

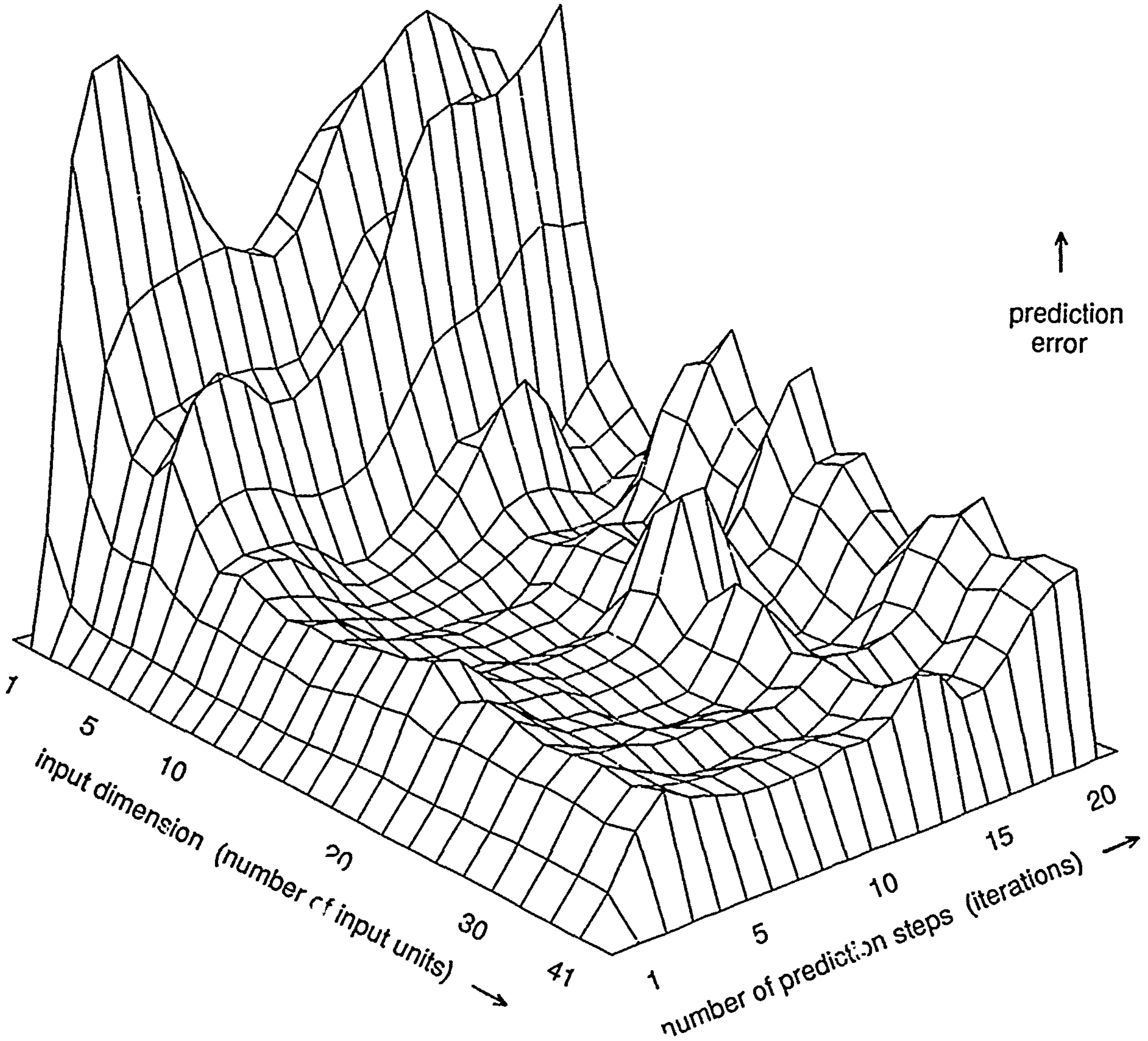


Figure 6. Prediction error as function of the number of input units of the network and the predictio. [illegible] into the future (iterated single-step predictions).

We also investigated the qualitative long term behavior of the models. The sunspot TAR model exhibits a periodic *eventual forecasting function* for iterated predictions [TL80], showing that TAR models can be self-exciting. This is an important improvement over global linear autoregressive models. We analyzed the eventual forecasting functions of several network solutions for several starting years. The whole range of possible dynamic behavior - fixed points, limit cycles and chaos - was displayed.

In summary, although we took extreme care not to gain any unfair advantage over Tong and Lim [TL80] (by taking the same input dimension, using identical data sets, minimizing the same sum of squared errors, *etc.*), the multi-step predictions by iteration of the network were found to be significantly better. On average, the prediction variances of the network were about half the prediction variances of the threshold autoregressive model. This concludes the comparison with the benchmark model. In the next section, we present the performance of the network as a function of the number of input units. In the last section we investigate preprocessing the data, compare Gauss and Poisson assumptions for the errors and explore effects of different sampling times.

3.3 VARYING THE INPUT DIMENSION

We varied the number of inputs units from one to 41. The prediction error for iterated single-step predictions (for the standard 1921-1955 set) is shown in Fig. 6 as surface above the number of input units of the network and the prediction time into the future.

Networks with one input unit already managed to capture two thirds of the single-step variance. reducing it to $\text{arv}(predict)_{1921-1955} = 0.33$. The solution was practically linear with an offset. Networks with two input units reduced the relative variance to 0.17; they began to use the available nonlinearities.

With increasing number of input units, the error reaches a roughly constant value. The performance does not degrade with input dimension several times larger than necessary: the network ignores irrelevant information. This important insensitivity to the input dimension is an advantage over other prediction methods such as the simplex algorithm employed by Sugihara and May [SM90].

To investigate this issue further, one input unit was only presented with noise. As expected, the weights from it to the hidden units became very small.

Numerical values of the single-step prediction error as well as the average of the 7, 8, 9 and 10-step prediction errors are given in Fig. 7. Note that the error for single-step predictions reaches its plateau with fewer input units than the error for iterated predictions. Due to the very limited sample size of the prediction set, no solid claim about the chaoticity of the sunspot data can be made. However, the similarity of the 7, 8, 9 and 10 year predictions speaks against the hypothesis that the system is chaotic, particularly when contrasted against the example of a computational ecosystem, discussed in [WHR90], where the prediction error increases exponentially with prediction time.

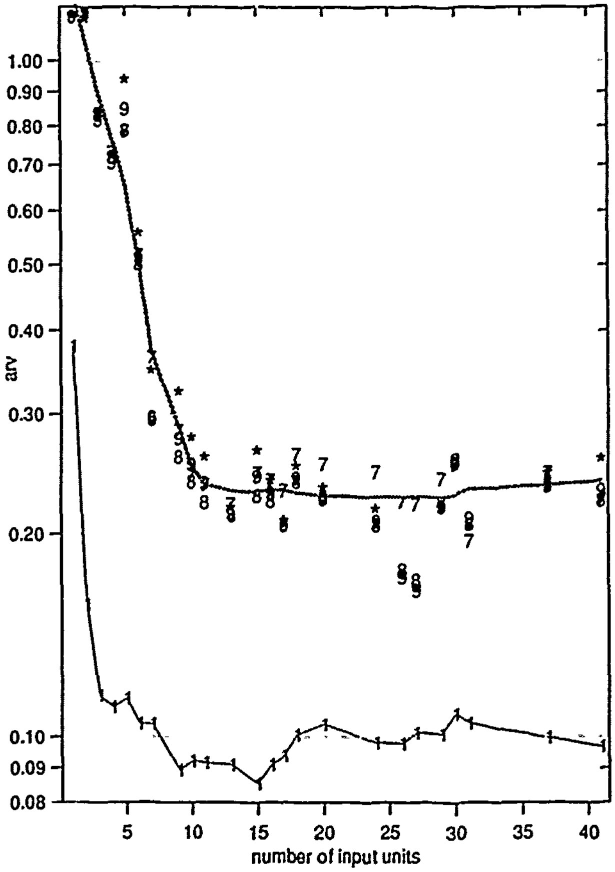


Figure 7: Relative prediction variance (1921-1955 average) as function of the number of input units. Solid line: single step prediction. Grey line: smooth average of 7...10 times iterated predictions. 1, 7, 8, 9, *: individual values of 1, 7, 8, 9, 10 year ahead predictions by iteration.

3.4 FURTHER RESULTS

We briefly summarize some further experiments on yearly sunspot data:

- **Preprocessing.** The distribution of the sunspot data (Fig. 3) is skewed towards small values. We trained networks on two data sets that were preprocessed to render less skewed distributions. x_t denoting "linear" values (after scaling into (0,1), as explained in the Appendix), we used square root transformed data, $X_t = \sqrt{x_t}$, and logarithmically transformed data,

  $$X_t = 0.98 + 0.35 \log(x_t + 0.06) \quad .$$

  The constant in the argument of the logarithm was chosen to make the distribution as symmetrical as possible. The other two numbers just scale and shift $\{X_t\}$ into the unit interval.

  It was easier for the networks to learn the square root transformed and the the logarithmically transformed data than the original data since the mean of both transformed sets was 0.5 as opposed 0.25 for the original data. The weight-eliminated networks tended to be slightly smaller. The prediction errors, however, computed the space of the original variable (after back-transforming the predictions) were slightly worse.

- **Poisson Statistics.** Replacing the assumption of Gaussian distributed errors (variance independent of predicted value) by assuming the errors to be Poisson distributed (variance proportional to predicted value) led to very similar prediction accuracy when evaluated with the sum squared error criterion.

Finally, we used *monthly* sunspot data from 1749 to 1976 (from [BCW88]) as input. Two tasks were designed. (A). predict the average sunspot number of the 12 months following the month corresponding to the latest input unit, and (M). predict the sunspot number of the month following the month corresponding to the latest input unit. Three networks were analyzed:

1. Task (A) alone, with a 80-20-1 network. The performance was slightly better than the performance obtained with yearly averages as input.
2. Task (M) alone, with a 80-20-1 network. We found the absolute prediction variance to be the same as in task (A).
3. Both task (A) and task (M), with a 80-40-2 network. The minimum error of this network was slightly above the sum of the squared errors of the two previous networks: adding a related task and doubling the resources (number of hidden units) did not help improve the prediction. We are presently designing an algorithm to prevent the network from overfitting on one task while it is still learning others.

4 SUMMARY

We investigated connectionist networks for short-term prediction of time series. Extending the work of Lapedes and Farber [LF87], we applied the networks to the sunspot series. On this noisy real world time series our networks outperformed the threshold autoregressive model by Tong and Lim [TL80], considered the best model in a recent review by Priestley [Pri88].

Several results in the domain of connectionist networks were also obtained. We presented a weight-elimination procedure as a solution to the related problems of network size and overfitting of the data. It dynamically reduced the number of hidden units to three. Furthermore, we compared different activation functions for the nonlinear hidden units. Whereas networks with sigmoid units converged reliably, serious problems were encountered with decreasing radial basis functions. The difference in performance was explained through the different treatment of the input variables.

Although the scope of this paper is limited to feed-forward networks, we are presently further investigating the effects of different cost functions and of training on additional tasks, as well as architectures for non-stationary time series and fully recurrent networks. Possible applications of these methods are in econometrics and finance, protein sequencing, seismic data, nonlinear predictive coding, and music.

We thank Richard Durbin and the lecturers and participants of the 1990 Connectionist Models Summer School for their comments, particularly in the workshop on time series prediction. This work was supported in part by a grant from the Office of Naval Research (N00014-87-K-0671).

Appendix: Parameters of the Network

For the comparisons between network and TAR predictions, the number of input units was set to 12. A network with initially 8 hidden units was trained with weight-elimination on the sunspot data for 5800 epochs with a learning rate of 0.1 and zero momentum. The weights were updated after each 20 patterns, presented in random order *(stochastic approximation)*. In order to remove effects of the specific order of the last presentation, the network was subsequently trained with a learning rate of 0.0001 for a few epochs with weight updates after each complete presentation of the whole training set.

The weights and biases for the three remaining hidden units are given in Table 1. The yearly sunspot values, tabulated both in Tong [Ton83] and Priestley [Pri88], were linearly compressed in the (0,1) range by dividing the raw values by 191.2 .

When simulated serially on a SPARCstation 1, the training of the 12-8-1 network for sunspot series with weight-elimination takes 1 minute for 100 presentations of the data. Once weights and biases are determined, predictions are extremely fast.

Table 1: Solution of the weight-eliminated network for sunspot prediction.

output bias: 0.798			
weights from hidden units to output:	−1.565	2.247	−1.599
biases of hidden units:	−0.858	−1.960	−0.512
(hidden unit	hu1	hu2	hu3)

weights from input:												
to hu1	0.153	−0.646	−0.328	−1.101	−0.060	−0.255	−0.129	0.500	0.030	0.317	−0.200	0.114
to hu2	−0.205	−0.094	−0.039	−0.048	1.078	0.414	0.000	0.533	−0.168	0.995	−3.103	0.814
to hu3	0.000	0.080	0.130	0.703	0.869	0.198	0.215	−0.208	−0.160	−0.923	−0.362	−4.010
(input	$t-12$	$t-11$	$t-10$	$t-9$	$t-8$	$t-7$	$t-6$	$t-5$	$t-4$	$t-3$	$t-2$	$t-1$)

References

[BCW88] Richard A. Becker, John M. Chambers, and Allan R. Wilks. *The New S Language*. Wadsworth and Brooks / Cole, 1988.

[BH89] Eric B. Baum and David Haussler. What size net gives valid generalization? *Neural Computation*, 1:151, 1989.

[BL88] David S. Broomhead and David Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321, 1988.

[Cas89] Martin Casdagli. Nonlinear prediction of chaotic time series. *Physica D*, 35:335, 1989.

[CFPS86] James P. Crutchfield, J. Doyne Farmer, Norman H. Packard, and Robert S. Shaw. Chaos. *Scientific American*, 255:46, December 1986.

[Cha90] Yves Chauvin. Generalization performance of overtrained networks. In L. B. Almeida and C. J. Wellekens, eds., *Neural Networks. Proceedings EURASIP Workshop Portugal 1990*, p. 46. Springer, 1990.

[Che90] Peter C. Cheeseman. On finding the most probable model. In Jeff Shrager and Pat Langley, eds., *Computational Models of Scientific Discovery and Theory Formation*, p. 73. Morgan Kaufmann, 1990.

[DSW87] John S. Denker *et al.* Large automatic learning, rule extraction and generalization. *Complex Systems*, 1:877, 1987.

[EF90] Stephen Eubank and J. Doyne Farmer. An introduction to chaos and randomness. In Erica Jen, ed., *1989 Lectures in Complex Systems*. Addison-Wesley, 1990.

[Fou90] Peter V. Foukal. The variable sun. *Scientific American*, 262:34, February 1990.

[FS89] J. Doyne Farmer and John J. Sidorowich. Exploiting chaos to predict the future and reduce noise. In Y. C. Lee, ed., *Evolution, Learning and Cognition*. World Scientific, 1989.

[Ger89] Neil Gershenfeld. An experimentalist's introduction to the observation of dynamical systems. In Bai-Lin Hao, ed., *Directions in Chaos*, vol. 2, p. 310. World Scientific, 1989.

[HP89] Stephen José Hanson and Lorien Y. Pratt. Comparing biases for minimal network construction with back-propagation. In D. S. Touretzky, ed., *Advances in Neural Information Processing Systems 1 (NIPS 88)*, p. 177. Morgan Kaufmann, 1989.

[LDS90] Yann LeCun, John S. Denker, and Sara A. Solla. Optimal brain damage. In David S. Touretzky, ed., *Advances in Neural Information Processing Systems 2 (NIPS 89)*, p. 589. Morgan Kaufmann, 1990.

[LF87] Alan S. Lapedes and Robert M. Farber. Nonlinear signal processing using neural networks: prediction and system modelling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, 1987.

[LH90] Kevin J. Lang and Geoffrey E. Hinton. Dimensionality reduction and prior knowledge in E-set recognition. In David S. Touretzky, ed., *Advances in Neural Information Processing Systems 2 (NIPS 89)*, p. 178. Morgan Kaufmann, 1990.

[Mar87] S. Lawrence Marple. *Digital Spectral Analysis with Applications*. Prentice-Hall, 1987.

[MB90] Nelson Morgan and Hervé Bourlard. Generalization and parameter estimation in feedforward nets: some experiments. In D. S. Touretzky, ed., *Advances in Neural Information Processing Systems 2 (NIPS 89)*, p. 630. Morgan Kaufmann, 1990.

[MD89] John Moody and Christian J. Darken. Fast learning in networks of locally tuned processing units. *Neural Computation*, 1:281, 1989.

[MR88] James L. McClelland and David E. Rumelhart. *Explorations in Parallel Distributed Processing*. MIT Press, 1988.

[Pri81] Maurice B. Priestley. *Spectral Analysis and Time Series*. Academic Press, 1981.

[Pri88] Maurice B. Priestley. *Non-linear and Non-stationary Time Series Analysis*. Academic Press, 1988.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart *et al.*, eds., *Parallel Distributed Processing*, p. 318. MIT Press, 1986.

[Ris89] Jorma Rissanen. *Stochastic Complexity in Statistical Inquiry*. World Scientific, 1989.

[Rum88] David E. Rumelhart. Learning and generalization. IEEE International Conference on Neural Networks, San Diego, 1988. Plenary Address.

[Sch88] Heinz-Georg Schuster. *Deterministic Chaos*. VCH Verlagsgesellschaft, 1988.

[SM90] George Sugihara and Robert M. May. Nonlinear forecasting as a way of distinguishing chaos from measurement error in time series. *Nature*, 344:734, April 1990.

[TL80] Howell Tong and K. S. Lim. Threshold autoregression, limit cycles and cyclical data. *Journal Royal Statistical Society B*, 42:245, 1980.

[Ton83] Howell Tong. *Threshold Models in Non-linear Time Series Analysis*. Springer, 1983.

[Ton90] Howell Tong. *Non-linear Time Series: a Dynamical System Approach*. Oxford University Press, 1990.

[WHR90] Andreas S. Weigend, Bernardo A. Huberman, and David E. Rumelhart. Predicting the future: a connectionist approach. *International Journal of Neural Systems*, 1:193, 1990.

Predicting the Mackey-Glass Timeseries With Cascade-Correlation Learning

R. Scott Crowder, III
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

The cascade-correlation learning algorithm has been shown to learn some binary output tasks 10-100 times more quickly than back-propagation. This paper shows that the cascade-correlation algorithm can be used to predict a real-valued timeseries. Results of learning to predict the Mackey-Glass chaotic timeseries using Cascade-Correlation are compared with other neural net learning algorithms as well as standard techniques. Learning speed results are presented in terms that allow easy comparison between cascade-correlation and other learning algorithms, independent of machine architecture or simulator implementation.

1 THE MACKEY-GLASS TIMESERIES

The Mackey-Glass timeseries is a good benchmark for learning programs because it has a simple definition, yet its elements are hard to predict (the series is chaotic.) Series prediction has many real-world applications in areas like signal processing, process control, and economic modeling. Another interesting feature of the Mackey-Glass problem is that real-valued outputs are required instead of the discrete output values found in most neural network benchmarks. Several other researchers have used the Mackey-Glass problem as a benchmark (Lapedes and Farber, 1987; Moody and Darken, 1988; Moody, 1989). However, while they have reported the quality of their solutions, the learning-time results have been reported in implementation-specific ways that make comparisons difficult.

The Mackey-Glass Series is derived by integrating the equation

$$\frac{dx[t]}{dt} = a\frac{x[t-\tau]}{1+x[t-\tau]^{10}} - bx[t].$$

When $a = 0.1, b = 0.2$, and $\tau = 17$, the integration produces a chaotic time series (Mackey and Glass, 1977). Figure 1 shows the portion of the series used for this study.

The goal of the task is to use known values of the timeseries up to the point $x = t$ to predict the value at some point in the future $x = t + P$. The standard method for this type of prediction is to create a mapping f from D points of the timeseries spaced Δ apart, *i.e.*, $(x[t-(D-1)\Delta], \ldots, x[t-\Delta], x[t])$, to a predicted future point $x[t+P]$. To allow comparison with earlier work (Lapedes and Farber, 1987; Moody and Darken, 1988; Moody, 1989), the values $\Delta = 6$ and $D = 4$ were used. Previous studies have used prediction interval values of $P = 6$ and $P = 85$. The choice of $P = 85$ is greater than the characteristic period of the series ($t_{char} \approx 50$). Predictions with $P > t_{char}$ have failed for standard methods like linear predictive coding and Gabor-Volterra-Wiener polynomial expansion (Gabor, 1960). However, by choosing $P = \Delta$ it is possible to predict the value of the timeseries at any multiple of Δ timesteps in the future, by feeding the output back into the input and iterating the solution. We choose to use $P = \Delta = 6$ for study since results can be compared with previous experiments where $P = 6$. By iterating the solution to $P = 84$, results may be

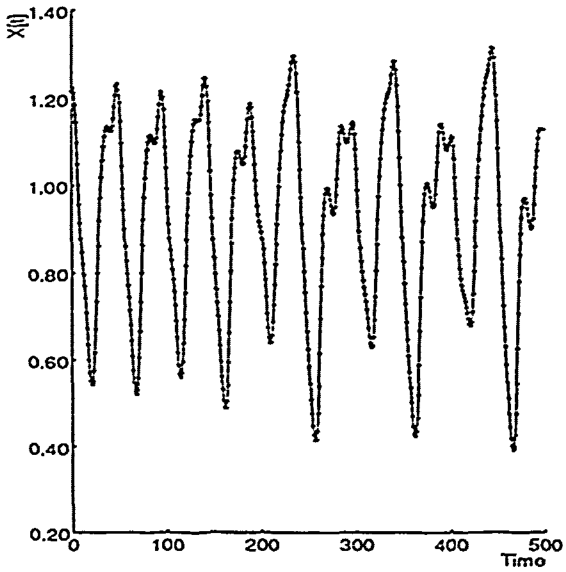


Figure 1: The Mackey-Glass Timeseries with $a = 0.1$, $b = 0.2$, and $\tau = 17$

compared with previous $P = 85$ studies.

Finally, a measure of performance must be specified. All error measures will be reported using the non-dimensional error index I, defined as the rms error divided by the standard deviation of the target series (Lapedes and Farber, 1987).

2 THE CASCADE-CORRELATION LEARNING ALGORITHM

Cascade-correlation is a supervised learning algorithm which builds a net as part of the learning process (Fahlman and Lebiere, 1990). The algorithm consists of two phases, output unit training and hidden unit training.

The output units receive input from all input and hidden units (initially just the inputs, as there are no hiddens.) In the first phase, output unit weights are trained to minimize the usual sum squared error measure, with error defined as the difference between the actual output values and the desired outputs. Once this training process levels off, a final epoch is run to record the residual error (difference between actual and desired outputs) for each unit on each training pattern. At the completion of output training, if the sum squared error remains above a certain threshold, a new hidden unit is inserted with weights determined by the hidden unit training phase.

During hidden unit training, a pool of potential hidden units is trained in parallel. Multiple candidates minimize the danger that an unfortunate choice of random initial weights will keep the new unit from contributing to the solution. The candidate hidden units receive input from both the input units and all previously-created hidden units. Thus, the hidden units form a cascade, as shown in Figure 2. The cascade architecture allows units to develop sophisticated higher-order representations.

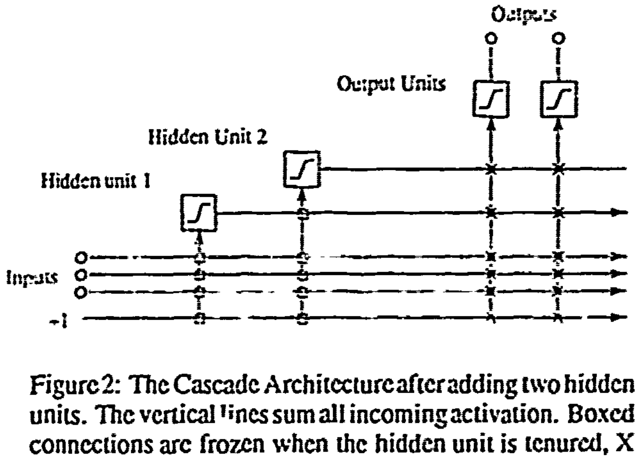


Figure 2: The Cascade Architecture after adding two hidden units. The vertical lines sum all incoming activation. Boxed connections are frozen when the hidden unit is tenured, X connections are trained repeatedly.

One problem with standard backprop is that when more than one hidden layer is added, learning seems to slow exponentially with the number of hidden layers. The cascade architecture realizes the benefits of multiple layers without suffering their exponential slowdown in learning speed, because only a single layer of weights is trained in each phase.

Candidate hidden units are trained to maximize S, the sum of the magnitude of the correlation between the output values for each pattern and the residual error from the previous output training phase. The measure S is defined as:

$$S = \sum_{o} \left| \sum_{p} (V_p - \overline{V})(E_{p,o} - \overline{E_o}) \right|$$

where o ranges over output units, p ranges over patterns, V_p is the candidate unit's output value for pattern p, and $E_{p,o}$ is the residual error of output unit o on pattern p. $\overline{V}$ and $\overline{E_o}$ are average values over all patterns.

Each candidate unit is trained separately. After the training reaches an asymptote, the candidate with the highest correlation score S is added to the network and tenured (its weights are frozen.) The network then repeats the output training phase. The two phases continue to alternate until the overall output error is small enough or the training run is declared a failure (a rare occurrence).

Since in both phases, the units being trained are directly connected to their inputs (no intervening layers with unfrozen weights), any gradient descent learning method may be used. The current implementation uses the quickprop algorithm (Fahlman, 1988) because of its speed. The only restriction placed on the hidden unit squashing function by the quickprop algorithm is that it must be differentiable. Most previous studies have used sigmoidal hidden units, but gaussian hidden units are possible, and in fact proved most effective for this problem.

2.1 REAL-VALUED OUTPUTS

All previous studies of cascade-correlation learning used tasks requiring binary outputs. For these tasks, sigmoidal output units are the logical choice. However, for real-valued problems the distribution of desired outputs is often gaussian rather than binomial, and the range of desired outputs may not be well-defined. The selection of a linear output unit addresses both problems by allowing equal likelihood for any output value and providing an unconstrained range. When the desired output distribution is bipolar, then sigmoidal output units yield a model that produces a maximum likelihood estimate for the correct weights. When the desired distribution is gaussian, linear units yield a maximum likelihood estimate.

Care must be taken when using linear output units in a constructive algorithm such as cascade-correlation. The error response of networks with sigmoidal outputs is fairly resistant to the addition of new units, because $\partial o/\partial net$ is very small for units whose outputs are close to 0 or 1. Therefore new hidden units tend to move the undecided outputs toward the correct response, but have a small affect on outputs that are already giving the correct response. New units can be added without drastically changing the part of the error space that must be explored during the output training phase. On the other hand, $\partial o/\partial net$ is constant for linear units. Any new hidden unit will effect all outputs equally. If the output weight initially assigned to the new unit is too high, the network may move to a completely new part of error surface. This can result in incomplete learning during the output phase, which will require the addition of extra hidden units to bring the solution back on track. This problem can be avoided by using a very small initial output weight for new hidden units.

3 BENCHMARK RESULTS

3.1 TRAINING RESULTS

The problem was run 50 times using cascade-correlation with gaussian hidden units selected from a pool of 8 candidate units. The network was trained until the error index I dropped to 0.025. The size of networks constructed ranged from 23 to 39 hidden units, with an average of 31.5 and median of 32 units. Figure 3 shows the distribution of the network sizes.

Hidden Units	Number of Trials	
23	1	•
24	0	
25	1	•
26	1	•
27	2	• •
28	2	• •
29	3	• • •
30	4	• • • •
31	7	• • • • • • •
32	10	• • • • • • • • • • •
33	6	• • • • • •
34	5	• • • • •
35	4	• • • •
36	2	• •
37	1	•
38	0	
39	1	•

Figure 3: Distribution of Network Sizes

The median size network had a total of 693 adjustable parameters, or about 28% more than the 40 hidden unit network used by Lapedes and Farber. The cascade architecture, which specifies that each unit is connected to *all* previous units, accounts for the higher number of parameters even thought the units count is smaller.

The simulations required an average of 3875 epochs (including both output and hidden unit phase epochs). The number of epochs obviously varied with the number of units used. The number of epochs per unit averaged 123, with smaller nets requiring slightly more epochs per unit than larger ones. Epochs are a good measure of learning performance for networks with a fixed architecture. Unfortunately, the notion of an "epoch" is more complicated with cascade-correlation. input epochs train a pool of fresh candidate units, while output epochs retrain the input connections of the output units. In both cases the number of weights being adjusted increases as hidden units are added.

A better objective measure of performance for network growing algorithms like cascade-correlation is the number of multiply/accumulates performed during the activation feed-forward and error back-propagation steps (Fahlman and Lebiere, 1990). This measure does ignore some of the computation performed during the simulation, but it is a better measure of amount of work required than epochs or machine running time.

Cascade-correlation constructed successful networks using between 141 and 360 million multiply/accumulates. The average trial required 265 million multiply/accumulates to solve the task.

Precise comparisons between cascade-correlation's learning speed and learning speed of the other algorithms are not possible. Previous researchers have reported their learning speed results in terms of computation time, because it is impossible to separate contributions from the algorithm, the hardware, and the details of the coding.

3.2 NETWORK BUILDING

Figures 4 and 5 show the building of one 25 hidden unit network. Each pair of figures shows the output of the network developed so far compared with the goal on the left and the output of the next hidden unit compared to the error of the pre-existing network on the right. Figure 6 illustrates how the addition of each unit helps to reduce the overall error in the network output.

The first few hidden units seem to be attempting to mirror the shape of the network error curves. After addition of the fourth hidden unit, the units begin lose their smooth response. This scatter is probably a result of using the covariance measure S to train the candidate units instead of a true correlation measure.

The current training strategy rewards units with extreme values, the sign of the $(V_p - \overline{V})$ term is more important than the actual unit output, V_p. In fact the highest possible S would be awarded to a unit that output 1 whenever $(E_{p,o} - \overline{E_o})$ is positive and 0 when $(E_{p,o} - \overline{E_o})$ is negative (or visa versa). This strategy works well in networks with sigmoid output units because it is not possible to overshoot the goal value. In linear output networks a more graded response would be beneficial. Use of a true correlation measure for hidden unit training would encourage the development of units with a more graded output.

The contribution of each unit to reducing the overall net-

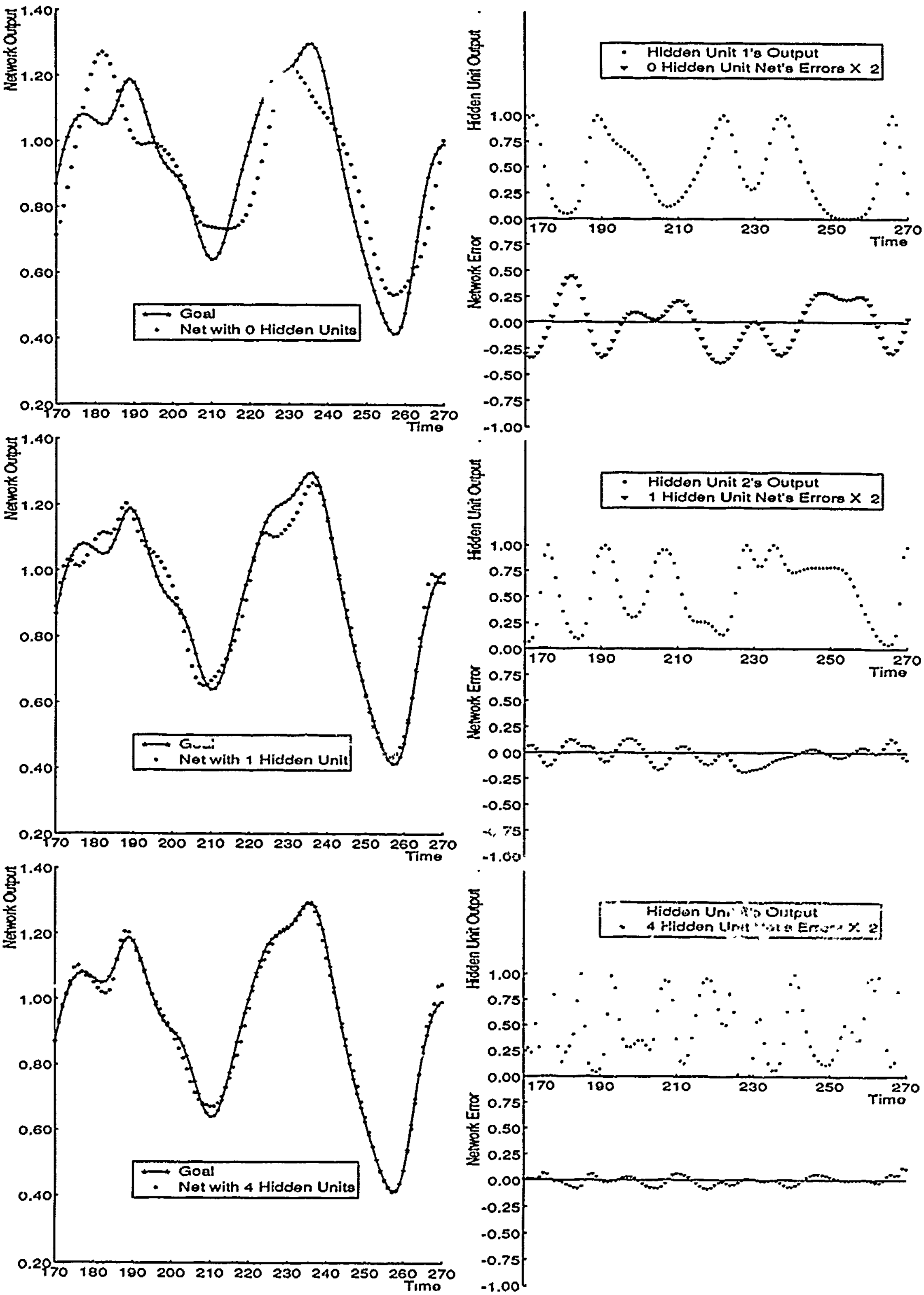


Figure 4: Network Evolution for a 25 Unit Network.

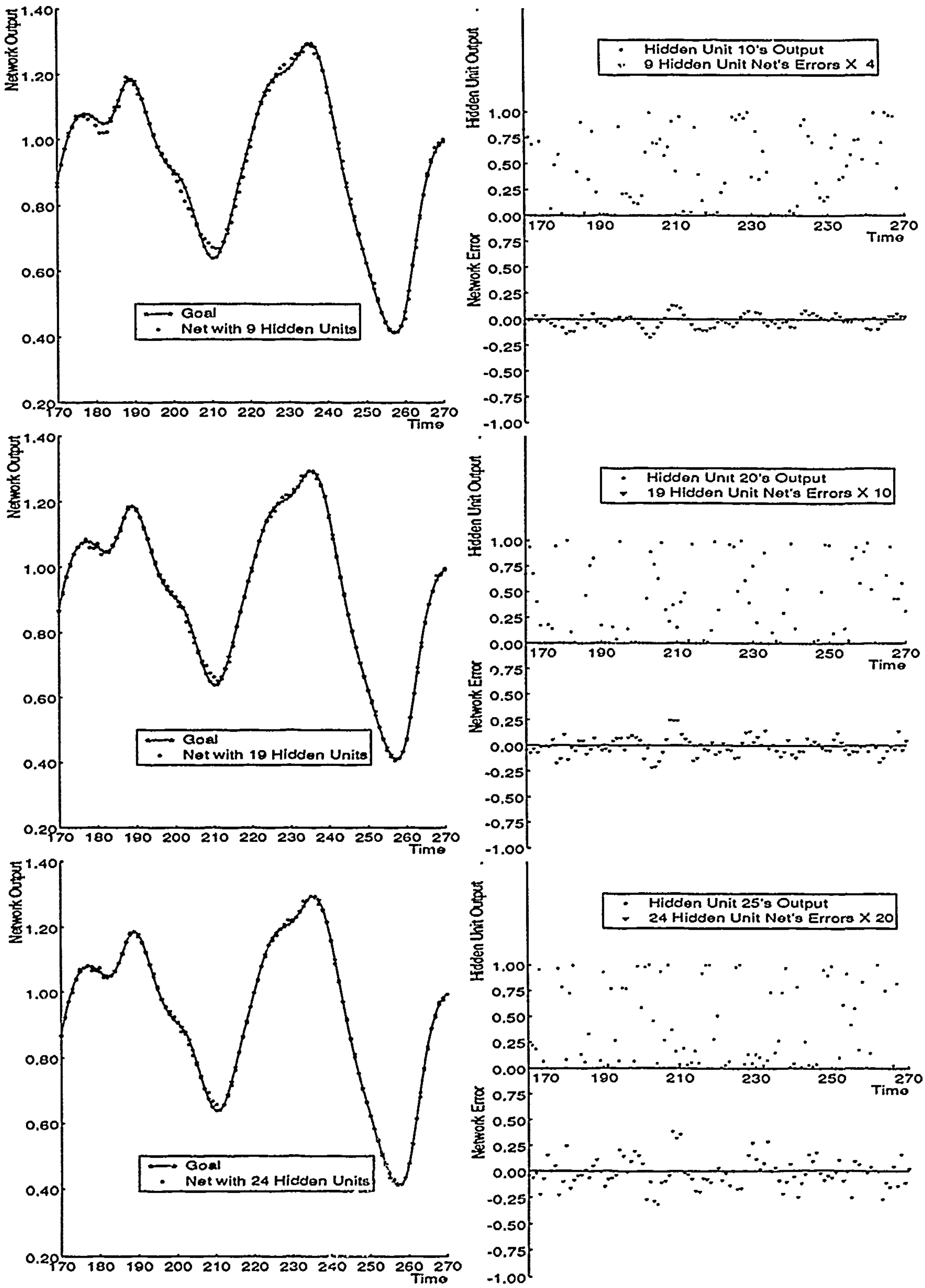


Figure 5: Network Evolution for a 25 Unit Network (Continued).

work error, I, is show in Figure 6. As expected, the first units, which mirror shape of the individual $E_{p,o}$, reduce I the most. It is also interesting to note that the addition of each new hidden unit allows the network to reduce I log-linearly.

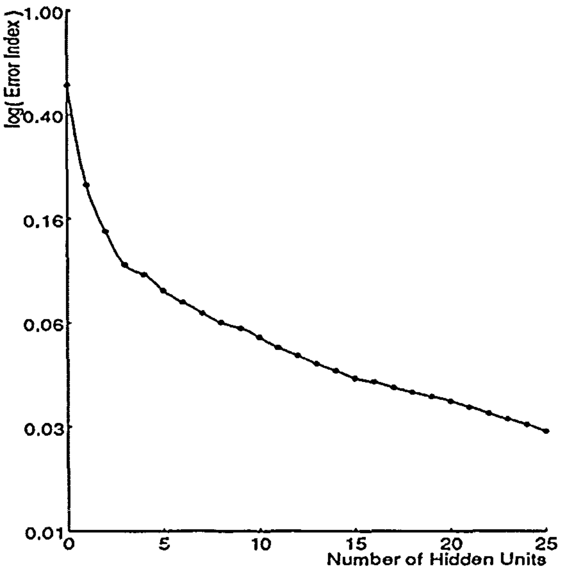


Figure 6: Network Error as a Function of the Number of Hidden Units

3.3 GENERALIZATION RESULTS

Generalization was measured by using each network to predict 500 points immediately following the training set. Generalization results for cascade-correlation are so far not as good as the best results for backprop reported by other neural network researchers. The results for $P = 6$ are presented in Table 1. The results show that all methods, except for the linear predictive model, are able to generalize fairly well for a small time into the future. The more challenging test of $P = 85$, Table 2, show that the neural-network techniques outperform standard techniques.

Table 1. Generalization Result Comparison for $P = 6$

Method	Training Cases	I
Cascade-Correlation	500	0.06
Back-Prop	500	0.02
6th-order polynomial	500	0.04
Linear Predictive Method	2000	0.55

The poor generalization performance of cascade-correlation compared to other neural network techniques is probably related to the choice of S used to train hidden units. A more effective training measure would result in smaller networks being built. There is evidence that the networks created by cascade-correlation are suffering from over-training. The generalization error for the networks reaches a minimum at about 20 hidden units and then increases sightly as each additional unit is added. The best generalization performance observed was $I_{P=6} = 0.04$ and $I_{P=84} = 0.17$.

Table 2: Generalization Result Comparisons for $P = 85$. Results for first four methods are generated by iterating the solution at $P = 6$. Results for Localized Receptive Fields(LRF) and Multi-Resolution Hierarchies(MRH) are for networks trained for $P = 85$.

Method	Training Cases	I
Cascade-Correlation	500	0.32
Back-Prop	500	0.05
6th-order polynomial	500	0.85
Linear Predictive Method	2000	0.60
LRF	500	0.10 - 0.25
LRF	10000	0.025 - 0.05
MRH	500	0.05
MRH	10000	0.02

4 Conclusions

Past studies have shown that for some binary output tasks cascade-correlation learning is much faster than standard network learning algorithms. This study shows that cascade-correlation is capable of solving a problem that requires real-valued outputs. Past neural network researchers have used the Mackey-Glass task to test their learning algorithms. Learning speed comparisons with these studies are not possible because their results are stated in machine specific terms. A machine and simulation-implementation independent measure of learning speed is given so that future researchers can compare results against cascade-correlation. However, the generalization performance of the networks created by cascade-correlation is not as good as networks created by other learning algorithms. Future work on cascade-correlation should focus on using a true correlation measure to train new hidden units instead of the measure used in the current implementation.

Acknowledgments

I would like to thank Scott Fahlman for guidance and insights that have helped to shape this work, and Dave Touretzky for his thoughtful comments and patience while editing this paper. This work was sponsored in part by the National Science Foundation under Contract Number EET-8716324 and by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 4976 under Contract F33615-87-C-1499 and monitored by. Avionics Laboratory, Air Force Wright Aeronautical Laboratories, Aeronautical Systems Division (AFSC), Wright-Patterson AFB, OH 45433-6543.

References

S. E. Fahlman. (1988) Faster-Learning Variations on Back-Propagation: An Empirical Study. In D. Touretzky, G. Hinton, and T. Sejnowski (eds.), *Proceedings of the 1988 Connectionists Models Summer School*, 38. San Mateo, CA: Morgan Kaufmann.

S. E. Fahlman & C. Lebiere. (1990) The Cascade-Correlation Learning Architecture. In D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems* 2, Morgan Kaufmann. A longer version is available as Technical Report CMU-CS-90-100, Carnegie Mellon University, School of Computer Science.

D. Gabor, W. P. Wilby, and R. Woodcock. (1960) *Proceedings IEEE*, 108B: 422.

K. J. Lang & M. J. Witbrock. (1988) Learning to Tell Two Spirals Apart. In D. Touretzky, G. Hinton, and T. Sejnowski (eds.), *Proceedings of the 1988 Connectionists Models Summer School*, 52. San Mateo, CA: Morgan Kaufmann.

A. S. Lapedes & R. Farber. (1987) Nonlinear Signal Processing Using Neural Networks: Prediction And System Modeling. Technical Report, Los Alamos National Laboratory, Los Alamos, New Mexico.

M. C. Mackey & L. Glass. (1977) Oscillation and Chaos in Physiological Control Systems. *Science*, 197:287.

J. Moody & C. Darken. (1988) Learning with Localized Receptive Fields. In D. Touretzky, G. Hinton, and T. Sejnowski (eds.), *Proceedings of the 1988 Connectionists Models Summer School*, 133. San Mateo, CA: Morgan Kaufmann.

J. Moody (1989) Fast Learning in Multi-Resolution Hierarchies. In D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 1*, 29. San Mateo, CA: Morgan Kaufmann.

Learning in Recurrent Finite Difference Networks

Fu-Sheng Tsung
Computer Science & Engineering
University of California, San Diego
La Jolla, CA 92093
tsung@cs.ucsd.edu

Abstract

A recurrent learning algorithm based on a finite difference discretization of continuous equations for neural networks is derived. This algorithm has the simplicity of discrete algorithms while retaining some essential characteristics of the continuous equations. A problem arising from using a discrete algorithm to learn sine wave oscillations is described and shown to be solved by this algorithm. This example explains a limitation of the discrete algorithm when learning to approximate continuous behaviors.

1 A FINITE DIFFERENCE ALGORITHM

The following coupled differential equations, with slight variations, have been widely used for continuous neural networks (for example, see Hopfield (1984), Pineda (1987), Pearlmutter (1989)):

$$\tau_k \frac{d\,y_k}{dt}(t) = -y_k(t) + f(x_k(t)) \tag{1a}$$

$$x_k(t) = \sum_j w_{kj}\, y_j(t) \tag{1b}$$

where $y_k\ (t)$ is the output of the unit k at time t, and τ_k is the time constant[1]. w_{kj} are the weights (coupling) from unit j to unit k; together with the transfer function $f()$ they specify the interaction among the units in the network. If no restrictions are placed on w_{kj}, then (1) may be an arbitrarily connected network. In particular, if cyclic connections exist among the network units, then such networks are called "recurrent nets". Recurrent nets are more powerful than feedforward nets (though perhaps not as efficient at certain tasks). One of the most important properties of recurrent nets is their potential to model temporal processes with feedback.

Gherrity (1989) and Pearlmutter (1989) have derived different learning algorithms for fully recurrent networks defined by continuous equations such as (1). On the other hand, Williams & Zipser (1989) derived a recurrent algorithm called RTRL (real-time recurrent learning)[2] based on discrete-time network dynamics, which is more common for feedforward networks. That is, the network dynamics is defined by

$$y_k(t+1) = f(x_k(t)); \quad x_k = \sum_j w_{kj} y_j \tag{2}$$

In both Gherrity (1989) and Pearlmutter's (1989) algorithms, the error gradients are calculated from the differential equations. The resulting equations then must be numerically integrated for simulation. In this paper we adopt a different approach, by first discretizing the differential equations, then calculating the gradients from the discretized equations. The resulting network can be considered as a cross between continuous and discrete networks. One advantage of this approach is that the learning algorithm is as straightforward as the RTRL algorithm, while the network retains some essential characteristics of the continuous network.

Among the elementary finite difference schemes for discretization, we consider the most basic one, the first order forward difference approximation:

$$\tau_k \frac{d\,y_k}{dt}(t) \cong \tau_k \frac{\Delta y_k}{\Delta t}(t) = \tau_k \frac{y_k(t+\Delta t) - y_k(t)}{\Delta t} \tag{3}$$

[1] Time constants determine the time scale of the system. One way to see this is by considering (1a) in the absence of $f()$. There the solution is $y=Ce^{(-t/\tau)}$ for some constant C. That is, after each time period of τ, y decays by $1/e$.

[2] Which has been independently discovered several times, see (Williams & Zipser, 1990) for references.

where Δt is a small, positive number[3].

Substituting the RHS (right-hand-side) of (3) into the LHS of (1a), and rearranging terms, we get:

$$y_k(t+\Delta t) = (1-\frac{\Delta t}{\tau_k})\, y_k(t) + \frac{\Delta t}{\tau_k} f(x_k(t)) \qquad (4)$$

To distinguish (4) from the continuous networks (1) and discrete networks (2), let us call networks defined by (4) "delta net". While (4) is derived from (1), it is related to the discrete equations (2) as follows. For numerical integration, it is normally desirable that the integration step Δt be small compared to the time constant τ. However, in the limit that Δt=τ, (4) reduces to (2). That is, we may consider discrete nets as a special case of delta nets, and that each iteration of the discrete net corresponds to jumping one time-constant forward in continuous time.

There is an intuitive interpretation for (4) from the discrete-time point of view. For example, if all the time constants are 1 and Δt is 0.1, then at each iteration step, the new activation for a unit is formed by (1-Δt) or 90% of the original activation, plus 10% of the contributions from the presynaptic units. Thus each unit decays exponentially, consistent with the original continuous system.

Having observed the similarity between (2) and (4), we proceed to derive a learning algorithm for the delta net in exactly the same way that Williams & Zipser (1989) derived the RTRL algorithm. If the desired output for unit k at time t is $d_k(t)$, then define the error of the system to be

$$E\,(t) = \frac{1}{2} \sum_k \left[d_k(t) - y_k(t)\right]^2 \qquad (5)$$

Weights are modified in proportion to the error's negative gradient with respect to the weights themselves:

$$\Delta w_{ij}(t) = -\eta \frac{\partial E}{\partial w_{ij}}(t) = -\eta \sum_k \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial w_{ij}}(t) \qquad (6)$$

Here η is a positive proportionality constant (the learning rate). By (5), the first term in the summation is simply

$$\frac{\partial E}{\partial y_k} = -\left(d_k(t) - y_k(t)\right)$$

The second term in the summation is calculated recursively, using (4):

$$p_{ij}^k(t+\Delta t) \equiv \frac{\partial y_k}{\partial w_{ij}}(t+\Delta t) = (1-\frac{\Delta t}{\tau_k}) p_{ij}^k(t) +$$

$$\frac{\Delta t}{\tau_k} [f'(x_k(t))] \left[\sum_l w_{kl} p_{ij}^l(t) + \delta_{ik} y_j(t) \right] \qquad (6)$$

where δ_{ik} is the delta function. The learning algorithm is thus

$$\Delta w_{ij}(t+\Delta t) = \eta \sum_k \left(d_k - y_k\right) p_{ij}^k (t+\Delta t) \qquad (7)$$

The only difference between the delta net algorithm and RTRL is in the unit update equations ((2) and (4)) and equations (6). Compare (6) with the corresponding equation from RTRL:

$$p_{ij}^k(t+1) \equiv \frac{\partial y_k}{\partial w_{ij}}(t+1) =$$

$$f'(x_k(t)) \left[\sum_l w_{kl} p_{ij}^l(t) + \delta_{ik} y_j(t) \right] \qquad (8)$$

we see that (6) differs from (8) in exactly the same way that (4) differs from (2). Therefore, in the limit Δt=τ, the delta net learning algorithm also reduces to the RTRL algorithm.

2 SIMULATIONS

While the delta net equations (4) is a discretized version of (1), it may not be true that (1) and (4) have the same dynamics for all possible functions f. However, for the logistic function

$$f(x) = \frac{1}{1+e^{-x}}$$

(1) and (4) do indeed seem to be analogs of each other. To see this, we take a two-unit fully connected network (Fig. 1a) and train it to oscillate as sine waves with the RTRL algorithm (Fig. 1b). This network has no inputs; the learned oscillation is a stable limit cycle of the two-unit dynamical system. The sine waves are normalized to between 0.1 and 0.9, and discretized with period T=30; that is, the discrete network steps through one period of the sine wave after 30 network iterations[4].

We then use the weights learned from the RTRL algorithm as the weights for an identical delta network, and run this network with Δt=0.1. That is, we are really running the same network with the delta net update equations (4) instead of the discrete equations (2). The oscillation thus generated is qualitatively indistinguishable. But note that now it takes iterations to complete one period (30 × 0.1). Phase-space plots (unit 1 against unit 2) of these oscillations are shown in Figure 2a and 2b. The same holds for Δt=0.01

[3] Also known as Euler's method. This approximation can be obtained directly from the mathematical definition of derivatives.

[4] The period T is in units of the time constant. In this and all simulations that follow, we use a time constant of 1 for all units.

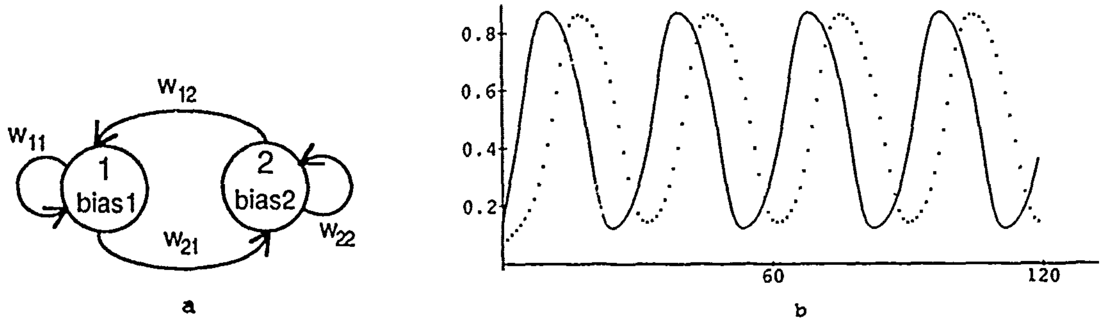
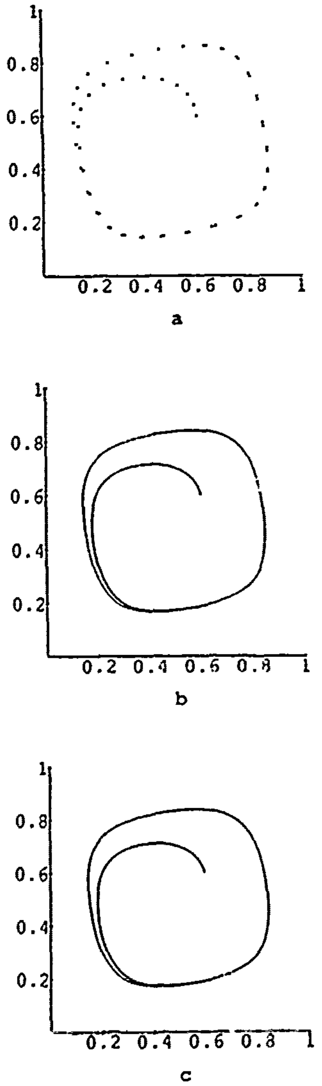


Figure 1: a. Diagram of the 2-unit fully connected network. The w_{ij}'s and biases are the weights. b. Sine waves learned with the RTRL algorithm, with period T=30 and phase difference of $\pi/4$. Unit 1 is the solid curve; unit 2 the dotted curve. The dots (unit 2) are actual values iterated by the network. For weights see Table 2a.



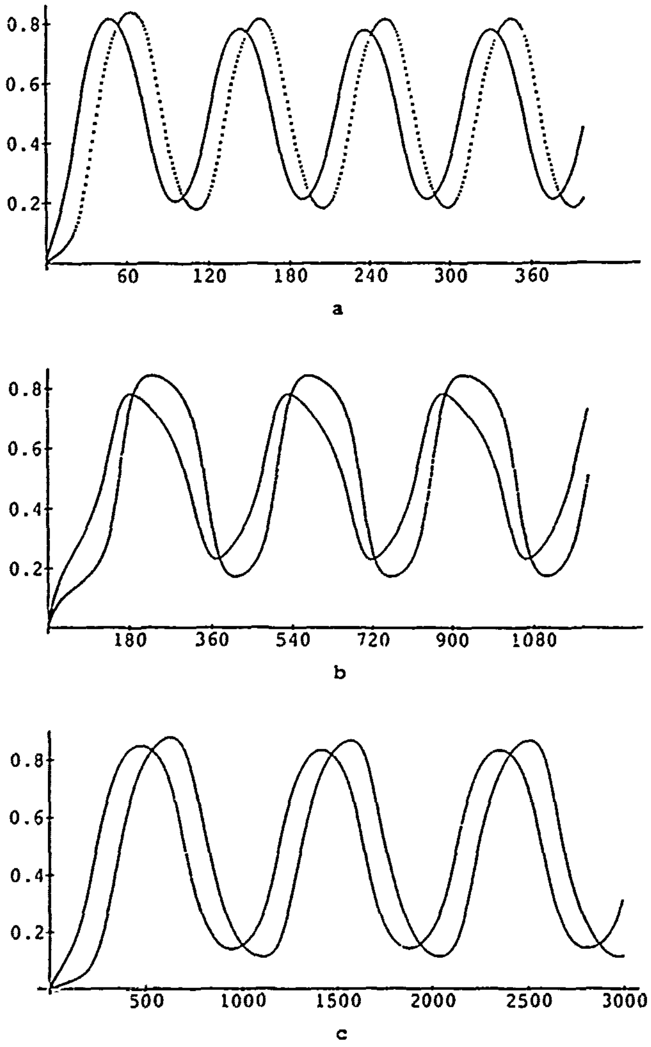


Figure 2: Phase plots of oscillations generated by the network of Fig. 1, but run with different Δt. a. Δt=1 (same as Fig. 1b). b. Δt=0.1. c. Numerically integrated with step size 0.01.

Figure 3: Oscillations learned by delta net algorithm. a. Period T=10, Δt=0.1. b. T=30, Δt=0.1. c. T=10, Δt=0.01. The distortion is small even though one period takes up to 1000 iterations to complete. Unit 1 is the solid line; unit 2, dotted line.

(3000 iterations per period). In fact, the same oscillation is generated using the continuous equations (1), which are numerically integrated with the 4th order Runge-Kutta method at a step size of 0.01.

The above tests shows that, at least for this two-unit network, the weights learned with the RTRL algorithm has the same behavior with the RTRL net, delta net, and continuous net. There the delta net is used simply as a first order numerical integrator. In Figure 3 we show oscillations generated with the delta net learning algorithm. In Fig. 3a, the two unit net is trained with period T=10, and Δt=0.1 (100 iterations/period). In Fig. 3b, T=30 and Δt=0.1; in Fig. 3c, T=10 and Δt=0.01. Table 1 shows the weights for these three cases.

These weights, obtained with the delta net algorithm, can also be run with different Δt from what they were trained with. For instance, the net from Fig. 3a is trained with Δt=0.1; it can be run with Δt=1 (≡ RTRL net), or Δt=0.01, and will generate the same wave forms.

The above tests supports the hypothesis that the delta net formulation is a good analog of both the continuous equations (1) and discrete equations (2). Whether this is true across a wide range of problems remains to be seen.

As a larger example, Fig. 4 shows a four-unit fully-recurrent network, two units trained to produce sin(t) and sin(2t), the other two are hidden units and are not plotted. Hidden units are necessary in this case because the phase space trajectory crosses with itself; at the point of intersection, additional information is needed for the net to figure out where to go next.

3 DISTORTED WAVE FORMS WITH THE RTRL ALGORITHM

Empirical observations show that it is necessary to use a technique called "teacher-forcing" in order to use the RTRL algorithm to train a network to oscillate (Williams & Zipser, 1989). The same is found to be true for delta nets. Teacher-forcing means, during training, instead of summing up the actual activations from the incoming units (which may be erroneous), each unit sums up the correct, teacher activations as its input for next iteration. One may visualize the effect of teacher-forcing by imagining that the network is learning to follow a trajectory; it goes astray (because the weights are wrong), but teacher-forcing puts the net back on its desired trajectory by setting the state of all the units to that of the teachers'. Teacher-forcing is applied periodically (by default, at every iteration) until the error criteria is met[5].

However, as an artifact of teacher-forcing, it can sometimes happen that after training, the error is consistently low (satisfying the error criteria), yet during free-running (when the teacher is removed), the learned oscillation appears very distorted from the teacher. An interesting example is from training the two-unit network with the RTRL algorithm to learn sine waves. When the period T is around 30 (see Fig. 1b), the net can approximate the sine waves fairly well. With T=60, that is, the net is to iterate 60 times during one period of the sine wave, training time can double, and the learned wave forms are slightly distorted. At T=100, if the net can oscillate at all, the wave forms are so heavily distorted they can hardly be called sine waves (Fig. 5). Weights for the three networks in Figure 5 are shown in Table 2. For periods much greater than 100 and smaller than 20, the net is unable to learn the sine waves with the RTRL algorithm. This is clearly undesirable for two reasons: First, if higher sampling frequency than 30 points per period is needed; for instance, if one wants to model a three-second oscillation, and precision down to millisecond level is desired. Second, one would like the teacher-forcing trained behavior to be the same as the free-running behavior, otherwise there is no obvious criteria for when the network has actually learned the teacher.

This distortion in RTRL nets is perplexing, too, because in section 2 we have shown oscillations by delta nets having up to thousands of iterations per period, without much distortion. Recall, also, that weights learned by a

[5] It is not entirely clear why teacher forcing is necessary in learning oscillations. One possible reason is to adjust the internal clock of the network: The net may oscillate correctly but if the phase is off, then the error will be consistently high. During training the net must learn to both create the correct internal clock and "adjust it to the right time".

Table 1: Weights for the 2-unit delta nets, a, b and c correspond to the networks that generated the oscillations in Figure 3a, 3b and 3c.

	T	Δt	w_{11}	w_{12}	w_{21}	w_{22}	bias 1	bias 2
a	10	0.1	7.053	-3.639	4.626	2.065	-1.703	-3.331
b	30	0.1	5.311	-0.954	1.828	3.642	-2.163	-2.726
c	10	0.01	9.372	-4.933	5.202	2.474	-2.271	-3.706

Table 2: Weights for the 2-unit RTRL nets; a, b and c correspond to the networks that generated the oscillations in Figure 5a, 5b and 5c. As the period T increases, the learned oscillations become more distorted.

	Period T	w_{11}	w_{12}	w_{21}	w_{22}	bias 1	bias 2
a	30	4.756	-1.159	1.022	4.620	-1.796	-2.807
b	60	4.894	-0.768	0.638	4.781	-2.069	-2.684
c	100	5.044	-0.772	0.632	4.954	-2.246	-2.745

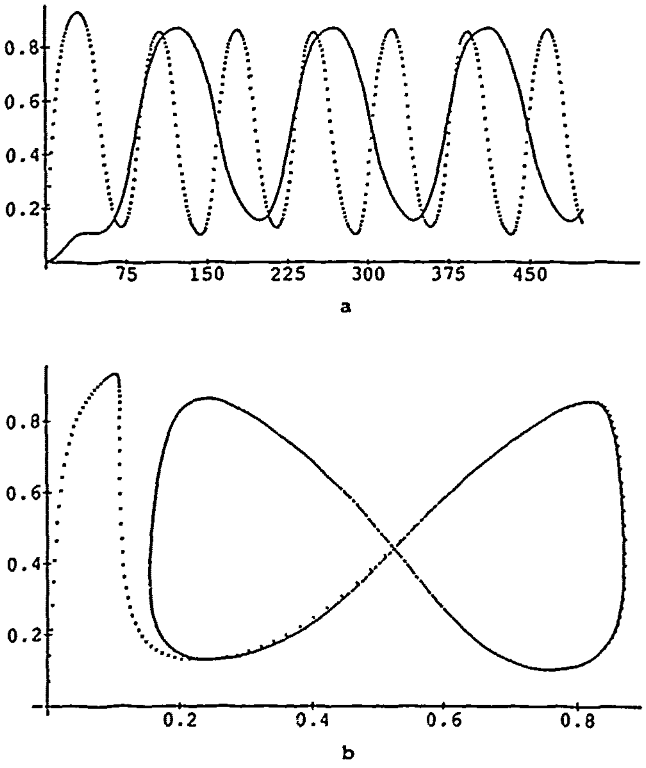


Figure 4: a. 4-unit delta net learned to produce sin(t) (solid line) and sin(2t) (dotted line). The two hidden units are not plotted. b. Phase plot of a.

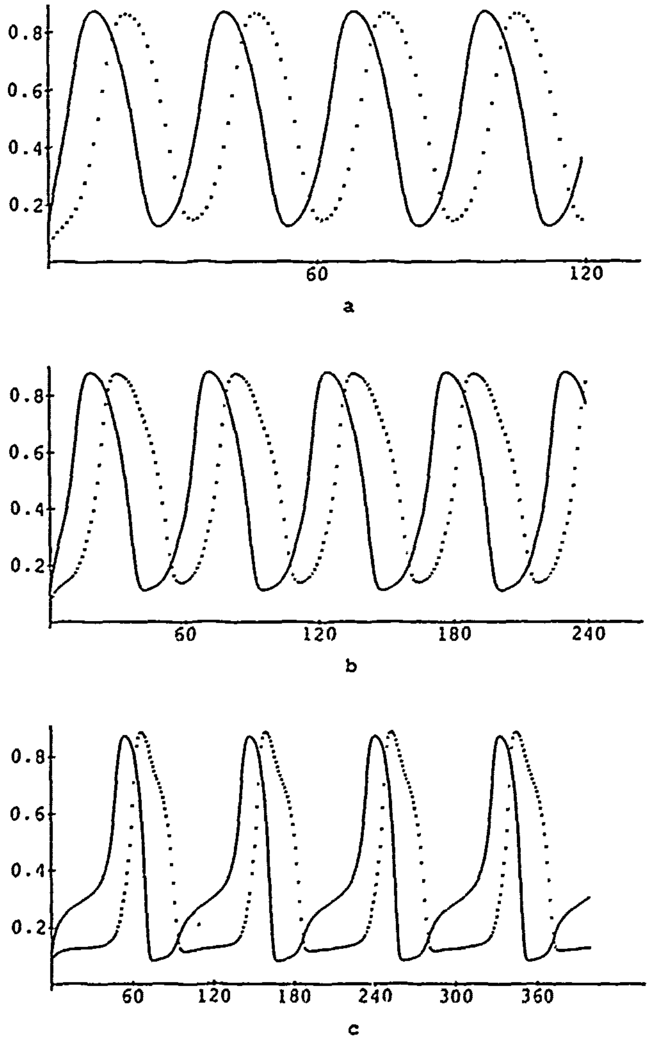


Figure 5: Oscillations learned by RTRL algorithm show distortions as the periods increase. a. Period T=30. b. T=60. c. T=100.

RTRL net (with period T=30) may be run with a Δt of 0.01, so as to have 3000 iterations per period without noticeable distortion (Fig. 2c).

We mentioned that the delta net equations (4) reduces to the discrete equations (2) when $\Delta t=\tau$. However, mathematically, something fundamentally different happens at $\Delta t=\tau$: the exponential decay term in the equation disappears; the old activation is completely forgotten with each new iteration. This suggests that a key advantage of the delta net (4) and the continuous net (2) is that the decay term acts as a direct memory input to the unit. As the curve (sine wave) is more finely sampled, the outputs of a unit from one time to the next (t to $t+\Delta t$) are very close to each other. The correct way to process this is to make Δt smaller, so the contribution from the decay term is greater, and more of the previous value is retained[6]. In the RTRL net, there is no such decay term serving as a "memory". What it ends up trying to do, say if a sampling rate of 100 points per period is needed (Fig. 5c), is to try to expand the oscillation over 100 time constants, which is the incorrect interpretation of "100 points per period".

In the RTRL net, the unit trying to achieve a value close to its previous value after an iteration must do so indirectly through the self-recurrent weight. Indirectly, because this input is "squashed" by the nonlinear transfer function. For a sigmoidal unit to retain its value near the maximum of 1, f must be close to 1, so the weight must be large (e.g. f(4)=0.98; f(5)=0.99, where f is the logistic function). Table 2 indeed shows that as T increases, the self-recurrent weights increase monotonically. On the other hand, influence from the other unit should decrease, in this 2-unit network, so that it takes longer to finish one oscillation. This trend is also observed in Table 2. Let us look at (1) again, in vector form:

$$\vec{y}\,'=-\vec{y}+\vec{f}(\mathrm{W}\vec{y}) \tag{5}$$

Defining the net input x,

$$\vec{x}=\mathrm{W}\vec{y};\ \text{so}\ \vec{x}\,'=\mathrm{W}\vec{y}\,'$$

and by multiplying **W** on both sides of (5) we get the equivalent system:

$$\vec{x}\,'=-\vec{x}+\mathrm{W}\vec{f}(\vec{x}) \tag{6}$$

This system is nonlinear exactly when f is nonlinear. The weights, W, are then the coefficients to the nonlinearities. As sine waves are linear harmonic oscillations, it is reasonable to assume that the network can approximate sine waves best when it can operate in the linear region; that is, when the weights are small or when the weights combine so that the nonlinear effects cancel. In the case of Table 2, increasing T forces the differences between a unit's self-recurrent weight (which becomes larger) and the weight from the other unit (which becomes smaller) to become large enough so that the network becomes a highly nonlinear oscillator. Hence the distortion observed in Fig. 5c.

Linear units will be much better at learning sine waves, but a linear system will not be able to provide stable limit cycles, which is important in biological oscillations.

The oscillation in Fig. 5c is interesting in its own right in that it resembles a class of nonlinear oscillators called relaxation oscillators. Relaxation oscillators are characterized by a phase where there is a relatively small change in the system, and a phase of very rapid changes. After the rapid phase, the system returns to the slow phase and starts over. Clearly, many biological oscillations fit this qualitative description, two immediate examples are the heartbeat and the action potential. A well studied relaxation oscillator is defined by the Van der Pol equation, where its near-harmonic oscillation becomes relaxation oscillation exactly as the coefficient of the nonlinear term becomes large[7]. It would be interesting to investigate the relationship between this class of differential equations and neural network oscillators.

4. DISCUSSION

The RTRL algori:hm is straightforward to implement for one familiar with back propagation[8]. Once it is implemented, the delta net modification can be made by changing as few as three lines of code. The only differences from RTRL net are the state equations (4) and the "p-values" (6). Also, note that the derivative of the logistic function,

$$f\,'=f(1\text{-}f)\quad \text{at time } t$$

can no longer be written as

$$f\,'(x(t))=y(t+1)*(1\text{-}y(t+1))$$

(which is done in back-prop and RTRL implementations), because

$$y(t+1)=f(x(t))$$

is not the update function anymore.

As with RTRL, the delta net algorithm suffers from high computational complexity: on the order of (n^4). For small networks (on the order of 10s of units in our experience), this complexity factor is not as important and

[6] This simply means the continuity property of (1) is preserved in (4).

[7] (Thompson,1986) contains a good but very brief introduction to relaxation oscillators that introduces the FitzHugh-Nagumo model of nerve axon response. A more thorough account on relaxation oscillators is by (Grasman, 1987).

[8] For implementation details, see (Williams & Zipser 1989).

the algorithm can be quite fast. For example, convergence of a 12-unit network trained to learn a biological oscillation (Tsung, Cottrell, & Selverston, 1990) on the Sun 4 takes less than three minutes with RTRL and about 20 minutes with delta net (in real time). This combination (easy implementation, small network) can be quite useful in the early stages of a modeling effort when changes in network architectures are made often. For more discussions on the pros and cons of different paradigms of recurrent learning algorithms, see Pineda (1989), and Williams & Zipser (1990).

During delta net training with a small Δt (0.1 or 0.01), we found that the net actually learned faster (by a factor of 5 to 10) when teacher-forcing was applied only intermittently, instead of at every step. For example, with a period of 10 and Δt of 0.01 (time constant of 1), teacher-forcing at every 50 iterations was sufficient and faster[9]. It was observed that with small Δt, teacher-forcing at every step would not give the net a chance to accumulate large errors, so learning would be slower[10]. Also, the error threshold should be selected with some care, as it depends on both the desired behavior (teacher) and the choices of Δt.

The second problem described in section 3, namely that the learned behavior does not always correspond to the teacher behavior when it is trained with teacher-forcing, even though the error cannot be reduced further, is an artifact introduced by teacher-forcing. One may contemplate various adaptive teacher-forcing ideas when training with delta net. For instance, gradually reduce the frequency of injecting teacher-forcing, say from once every 10 iterations, to once every 20, 30, ..., iterations. Another possible approach is to teacher-force a unit only when its error reaches some threshold. The latter suggests an "asynchronous teacher-forcing" method. These ideas have not been tried in simulation.

Finally, it should be mentioned that this particular discretization scheme chosen to relate the continuous equations and discrete equations is certainly not new. For example, Renals (1990) has studied the dynamics of neural networks at various degrees of symmetry and "discreteness" using the same discretization.

Acknowledgements

The author has benefitted from comments by Fernando Pineda, Peter Rowat, and David Zipser.

References

Gherrity, M. (1989) A learning algorithm for analog, fully recurrent neural networks. *Proc. IJCNN*, Washington D.C., June 18-22, I-643.

Grasman, J. (1987) *Asymptotic methods for relaxation oscillations and applications.* New York: Springer-Verlag.

Hopfield, J. J. (1984) Neurons with graded response have collective computational properties like those of two-state neurons. *Proc. Nat. Acad. Sci*, **81**, 3088-3092.

Pearlmutter, B. A. (1989) Learning state space trajectories in recurrent neural networks. *Proc. IJCNN*, Washington D.C., June 18-22, II-365.

Pineda, F. J. (1987) Generalization of back-propagation to recurrent neural networks. Physical Review Letters, 59, 19, 2229-2232.

Pineda, F. J. (1989) Time dependent adaptive neural networks. In D.S. Touretzky (ed.), *Advances in Neural Information Processing Systems*, **2**, 710-718.

Renals, S. (1990) Chaos in neural networks. In *Neural Networks*, L.B.Almeida & C.J.Wellekens (eds), Lecture Notes in Computer Science 412, Berlin: Springer-Verlag.

Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986) Learning internal representations by error propagation. In Rumelhart, D. E., McClelland, J. L. & The PDP Research Group, *Parallel Distributed Processing Vol. 1. Foundations*, MIT Press.

Thompson, J. M. T., Stewart, H. B. (1986) *Nonlinear dynamics and chaos.* Chichester: Wiley.

Tsung, F. S., Cottrell, G. W., Selverston, A.I. (1990) Some experiments on learning stable network oscillations. *Proc. IJCNN*, San Diego, June 17-21, I-169.

Williams, R. & Zipser, D. (1989) A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, **1**, 270-280.

Williams, R. & Zipser, D. (1990) Gradient-based learning algorithms for recurrent connectionist networks. Technical Report NU-CCS-90-9, College of Computer Science, Northeastern University, Boston.

[9] It might be better if the sampling frequency and teacher-forcing frequency are relatively prime with each other so the net is forced to all possible points on the trajectory at different times.

[10] David Zipser, personal communication.

Temporal Backpropagation: An Efficient Algorithm for Finite Impulse Response Neural Networks

Eric A. Wan
Department of Electrical Engineering
Stanford University
Stanford, CA 94305-4055
e-mail: wan@isl.stanford.edu

Abstract

The traditional feedforward neural network is a static structure which simply maps input to output. To better reflect the dynamics in the biological system a network structure is proposed which models each synapse by a Finite Impulse Response (FIR) linear filter. An efficient gradient descent algorithm is derived which will be shown to be a temporal generalization of the familiar backpropagation algorithm.

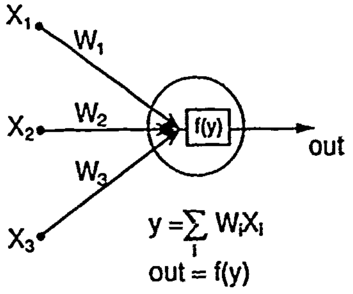


Figure 1: Static Model of Neuron

1 INTRODUCTION

A standard neural network models a synapse by a single variable weight parameter. In a feedforward structure this results in a *static* network which maps input to output. Real neural networks are of course dynamic in nature which is reflected in the temporal properties of the synapse along with such processes as impulse transmission and membrane excitation. While many accurate models of such processes do exist, from an engineering standpoint most are unrealistic to work with. The model we propose to use represents a synapse not by just a single weight parameter, but by an adaptive filter (Widrow, 1985). Further, we restrict the filter to be discrete time, characterized by a Finite Impulse Response (FIR) [1]. While biologically motivated, we make no claims that the structure is necessarily biologically plausible. With this we proceed to derive algorithms for adapting the synaptic transfer functions so as to train the network as a whole.

The structure of the FIR network presented here is similar to the Time-Delay Neural Networks (TDNN) used in speech recognition (Waibel, 1989). The primary differences are a matter of formalism and notation. As we will see, however, the proper choice of notation allows for a very elegant solution to the training algorithm. This algorithm may then be used as a computationally efficient method for training Time-Delay Neural Networks.

2 NETWORK STRUCTURE

2.1 STATIC MODEL

Consider first the traditional model of the neuron shown in Figure 1. The output of the neuron is simplistically taken to be a nonlinear function of the weighted sum of its inputs. Mathematically this is expressed as

$$y = \sum_i W_i X_i \tag{1}$$

$$out = f[y] \tag{2}$$

where X_i are the inputs to the neuron and W_i are the synaptic weights. By modeling the synapse with a single weight parameter the neuron can be considered a *static* structure mapping input to output. Clearly this is an oversimplified model of a biological neuron.

A common configuration is to arrange the neurons into a feedforward structure as shown in Figure 2. While this increases the class of functional mappings, it is still a static mapping from input to output.

[1] The Infinite Impulse Response (IIR) case has also been studied and will be presented in a future paper.

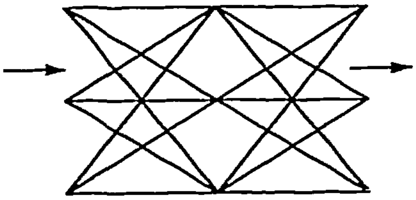


Figure 2: Static Feedforward Network

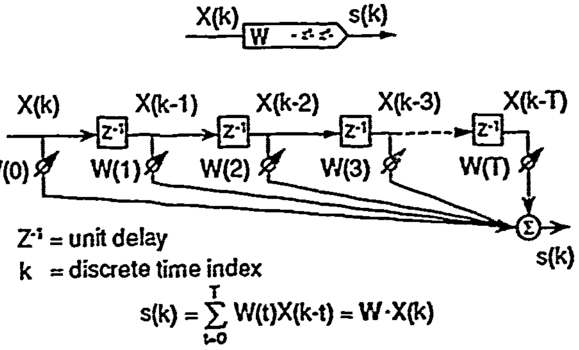


Figure 3: FIR Synapse

2.2 TEMPORAL MODEL

A more sophisticated model of the neuron results if time is incorporated into the structure by replacing the static model of the synapse by a FIR linear filter. This corresponds to a Markov model of transmission flow through the synapse. The synaptic model is shown in Figure 3. Defining k to be the discrete time index, the "output" of the synapse is now a weighted sum of delayed inputs

$$s(k) = \sum_{t=0}^{T} W(t)X(k-t). \tag{3}$$

The past input states can be represented in vector notation

$$\mathbf{X}(k) = [X(k), X(k-1), ..X(k-T)]. \tag{4}$$

Similarly we form a weight vector for the filter coefficients

$$\mathbf{W} = [W(0), W(1), ..W(T)]. \tag{5}$$

This allows us to express the operation of a synapse by a vector dot product, $\mathbf{W} \cdot \mathbf{X}(k)$, where time relations are now implicit in the notation.

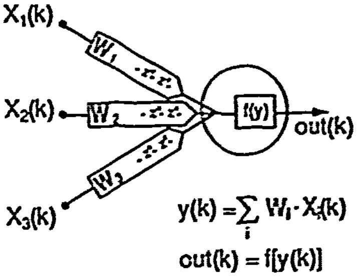


Figure 4: Temporal Model of Neuron

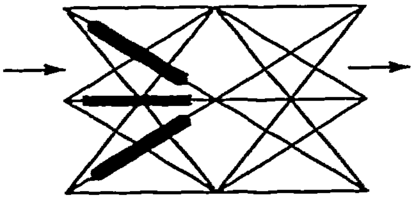


Figure 5: FIR Network

Using this model of the synapse we construct the neuron. The structure is the same as before except each static synapse is replaced by the FIR model. This is shown in Figure 4. Mathematically, the output of a neuron, now a function of time, is defined as

$$y(k) = \sum_i \mathbf{W}_i \cdot \mathbf{X}_i(k) \tag{6}$$

$$out(k) = f[y(k)] \tag{7}$$

Note the similarities in appearance between these equations and those of the static model in Equation 1. We have simply replaced scalars by vectors and multiplications by vector products. The convolution operation of the synapse is implicit in the notation. As we will show, these simple analogies will carry throughout subsequent derivations.

Finally, the temporal model of the neuron is used to construct a feedforward network as shown in Figure 5. Only feedforward structures will be considered at this time. However, the mapping for the feedforward network is no longer simply a static mapping. Dynamics are internal to the structure. Outputs depend on past values of the input.

3 TRAINING

Training will be based on a supervised learning algorithm in which a desired response vector is provided at each instance of time. Thus at each time increment the instantaneous squared error is defined as

$$e^2(k) = \sum_i [d_i(k) - out_i(k)]^2 = \sum_i e_i(k)^2 \qquad (8)$$

where $out_i(k)$ is the true value of output neuron i and $d_i(k)$ is the corresponding desired response. The sum is taken over all output neurons for the network. The goal is then to minimize the total squared error taken over all time

$$e^2 = \sum_k e^2(k). \qquad (9)$$

Note the choice of a Euclidean squared error metric is neither unique nor essential to the arguments in this paper. Other error metrics such as cross-entropy may also be considered. Furthermore, it is actually not necessary to provide desired response vector at *all* increments of time. It is the overall sequence of outputs that is important. At any given increment the desired output may not be known in which case the error metric can be taken as zerc This is the case, for example, with phoneme recognition.

Regardless of the choice of error metrics, learning will be based on traditional gradient descent in which we attempt to minimize total error over all time.

3.1 INSTANTANEOUS GRADIENT METHOD

Returning to thͤ squared error metric, the error gradient with respect to a given weight vector is normally expanded as follows:

$$\frac{\partial e^2}{\partial \mathbf{W}^l_{ij}} = \sum_k \frac{\partial e^2(k)}{\partial \mathbf{W}^l_{ij}} \qquad (10)$$

Subscripts are included so that $\mathbf{W}^l_{ij}$ specifies the synaptic filter connecting the output of neuron i in layer l to the input of neuron j in the next layer. By taking each term in the expansion as an unbiased instantaneous estimate of the gradient, we may form the on-line training algorithm:

$$\mathbf{W}^l_{ij}(k+1) = \mathbf{W}^l_{ij}(k) - \mu \frac{\partial e^2(k)}{\partial \mathbf{W}^l_{ij}(k)} \qquad (11)$$

in which the weight vectors are updated at each increment of time (μ is defined as the learning rate).

As we will show, this obvious expansion of $\partial e^2/\partial \mathbf{W}$ into the terms $\partial e^2(k)/\partial \mathbf{W}$ does not lead to a desirable learning algorithm for this structure. A less intuitive expansion, in fact, yields a more attractive algorithm which exploits the structure's FIR characteristics. For now we will proceed with this approach to show where the problems arise.

While direct calculation of the gradient terms is possible (Wan, 1990), the mathematics only tends to obscure the main features of the algorithm. A more illustrative approach is provided by using a graphical technique which involves *unfolding* the network in time. The general strategy is to try and remove all time delays by expanding the network into a larger static structure. Standard backpropagation (Rumelhart, 1986) can then be used to calculate the necessary gradient terms.

As an example, consider the very simple network shown in Figure 6. The network consists of three layers with a single output neuron and two neurons at each hidden layer. Each synapse is modeled as a second order (two tap) linear filter. Thus while there are only 12 synapses in the network there are actually a total of 30 variable filter coefficients. Now starting at the last layer each tap delay is interpreted as a "virtual neuron" whose input is delayed the appropriate number of time steps. A tap delay is then "removed" by replicating the previous layers of the network and delaying the input to the network accordingly. This is shown in Figure 7. The process in then continued backward through each layer until all delays have been removed. The final unfolded network is shown in Figure 8. This method produces an equivalent static structure where the time dependencies have been made external to the network itself by time windowing the input. Notice that whereas there were initially 30 variable coefficients the equivalent unfolded structure now has 150 "static" synapses. This can be seen as a result of redundancies in the weights. In fact, one can view a FIR network as a compact representation of a larger static network with imposed symmetries.

Given the unfolded static structure it is then a straight forward process to find the instantaneous error gradients for each weight using standard backpropagation. It is necessary, however, to keep track of which static weights are actually the same so that the gradients may be accumulated to find the total gradient for each unique parameter in the network. This need to do global bookkeeping is an immediate drawback to the instantaneous gradient approach. There is a loss of a sense of locally distributed processing. There is no longer a symmetry between the forward propagation of states and the backward propagation of gradient terms. No nice recursive formula for propagating all the error terms exists. The more layers in the network the more complicated the process becomes

From a practical standpoint, perhaps the greatest drawback to this approach is in the computational complexity of the algorithm. In a static network the number of computations associated with the standard backpropagation algorithm grows only linearly with

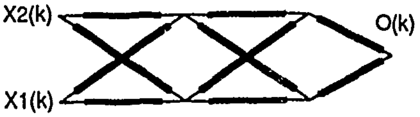


Figure 6: Simple Three layer FIR Network

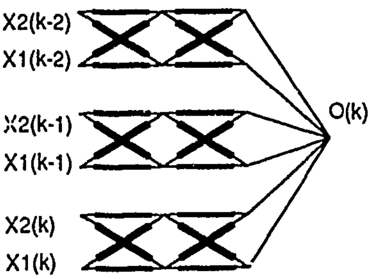


Figure 7: Unfolding Process

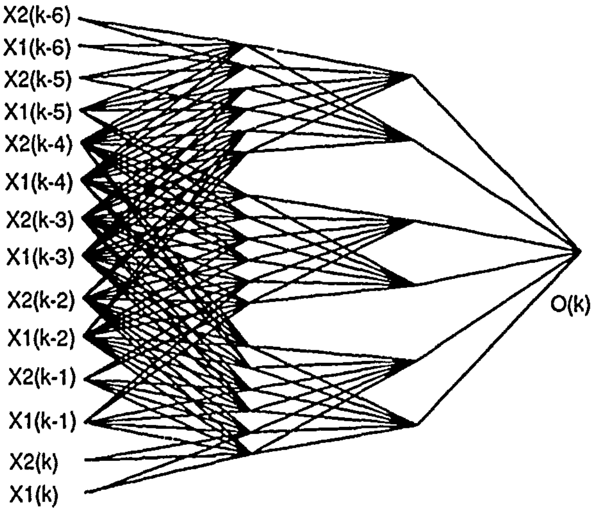


Figure 8: Final Unfolded Network

Table 1: FIR Networks vs. Static Equivalent

FIR Network Dimension		*Variable Parameters*	*Static Equivalent*
Nodes†	*Order*‡		
2×2×2×1	2:2:2	30	150
5×5×5×5	10:10:10	605	36,355
3×3×3	9:9	180	990
,.3×3×3	9:9:9	270	9,990
3 3×3×3×3	9:9:9:9	360	99,990
3^n	9^{n-1}	$(n-1)90$	$10^n - 10$

† Number of Inputs × Hidden Neurons × Outputs.
‡ Order of FIR synapses in each layer.

the number of weights. As we saw, a very small FIR network (30 parameters) was expanded to a much larger static network (150 weights). In fact, the size of the equivalent static network grows *geometrically* with the number of layers and tap delays. Consequently the number of computations required to train the FIR structure also grows geometrically with the size of the network. This would then be reflected in the actual training time. Table 1 shows the equivalent static size for various FIR networks. The TDNN system of Waibel used for speech recognition consisted of 521 variable coefficients. However, the expanded virtual network which was actually used to calculate the gradients consisted of a total of 6233 weights.

3.2 TEMPORAL BACKPROPAGATION

We now present an alternative learning algorithm which overcomes the problems associated with the instantaneous gradient approach. Unfortunately it is necessary to first complicate matters by introducing a bit of notation. The output of the synaptic filter connecting neuron i in layer l to the jth neuron in next layer is defined as

$$y_{ij}^{l+1}(k) = \mathbf{W}_{ij}^{l} \cdot \mathbf{X}_{i}^{l}(k) \tag{12}$$

where $\mathbf{W}_{ij}^{l} = [w_{i,j}^{l}(0), w_{i,j}^{l}(1), ..w_{i,j}^{l}(T^{l})]$ specifies the coefficients for the connecting synaptic filter. $\mathbf{X}_{i}^{l}(k) = [x_{i}^{l}(k), x_{i}^{l}(k-1), ..x_{i}^{l}(k-T^{l})]$ is the vector of delayed states along the synapse.

The *total* input to the jth neuron in layer l at time k is specified as

$$y_{j}^{l}(k) = \sum_{i=1}^{N_{l-1}} y_{i,j}^{l}(k) = \sum_{i=1}^{N_{l-1}} \mathbf{W}_{ij}^{l-1} \cdot \mathbf{X}_{i}^{l-1}(k). \tag{13}$$

The sum in the equation is taken over all N_l neurons in the layer. Finally, the output for the neuron is taken to be a nonlinear sigmoidal function of its input

$$x_{j}^{l}(k) = f(y_{j}^{l}(k)). \tag{14}$$

This notation completely defines the structure of the network. Note we can take $x_{j}^{0}(k)$ to specify an external input to the network while $x_j^L(k)$ specifies an output for an L layer network. Values of $x_j^l(k)$ for $1 \le l \le L$ specify internal states within the network.

We are now in a position to formally derive the new training algorithm. The original expansion of the total error gradient into a sum of instantaneous gradients is not unique. Consider the following:

$$\frac{\partial e^2}{\partial \mathbf{W}_{ij}^l} = \sum_k \frac{\partial e^2}{\partial y_j^{l+1}(k)} \cdot \frac{\partial y_j^{l+1}(k)}{\partial \mathbf{W}_{ij}^l} \tag{15}$$

This now yields an on-line version of the form:

$$\mathbf{W}_{ij}^l(k+1) = \mathbf{W}_{ij}^l(k) - \mu \frac{\partial e^2}{\partial y_j^{l+1}(k)} \cdot \frac{\partial y_j^{l+1}(k)}{\partial \mathbf{W}_{ij}^l}. \tag{16}$$

Note the time index runs over $y(k)$ and not $e^2(k)$. We may interpret $\partial e^2/\partial y_j^{l+1}(k)$ as the change in the total squared error over all time due to a change in the input to a neuron at a single instant of time. Furthermore,

$$\frac{\partial e^2}{\partial y_j^{l+1}(k)} \cdot \frac{\partial y_j^{l+1}(k)}{\partial \mathbf{W}_{ij}^l} \neq \frac{\partial e^2(k)}{\partial \mathbf{W}_{ij}^l(k)}. \tag{17}$$

Only the sum over all k is equivalent.

Now for any layer $\partial y_j^l(k)/\partial \mathbf{W}_{ij}^{l-1} = \mathbf{X}_i^{l-1}$. Furthermore, we may simply define $\partial e^2/\partial y_j^l(k) \equiv \delta_j^l(k)$. This allows us to rewrite Equation 16 in the more familiar notational form

$$\mathbf{W}_{ij}^l(k+1) \quad = \quad \mathbf{W}_{ij}^l(k) - \mu\delta_j^{l+1}(k) \cdot \mathbf{X}_i^l(k) \tag{18}$$

which now holds for any layer in the network. To complete the derivation, an explicit formula for $\delta_j^l(k)$ must be found. For the output layer we have simply

$$\begin{aligned} \delta_j^L(k) &\equiv \frac{\partial e^2}{\partial y_j^L(k)} = \frac{\partial e^2(k)}{\partial y_j^L(k)} \\ &= -2e_j(k)f'(y_j^L(k)). \end{aligned} \tag{19}$$

For any hidden layer we have

$$\begin{aligned} \delta_j^l(k) &\equiv \frac{\partial e^2}{\partial y_j^l(k)} \\ &= \sum_{m=1}^{N_{l+1}} \sum_t \frac{\partial e^2}{\partial y_m^{l+1}(t)} \frac{\partial y_m^{l+1}(t)}{\partial y_j^l(k)} \\ &= \sum_{m=1}^{N_{l+1}} \sum_t \delta_m^{l+1}(t) \frac{\partial y_m^{l+1}(t)}{\partial y_j^l(k)} \\ &= f'(y_j^l(k)) \sum_{m=1}^{N_{l+1}} \sum_t \delta_m^{l+1}(t) \frac{\partial y_{jm}^{l+1}(t)}{\partial x_j^l(k)}. \end{aligned} \tag{20}$$

But we recall

$$y_{jm}^{l+1}(t) = \sum_{k'=0}^{T^l} w_{jm}^l(k') x_j^l(t-k'). \tag{21}$$

Thus

$$\frac{\partial y_{jm}^{l+1}(t)}{\partial x_j^l(k)} = \begin{cases} w_{jm}^l(t-k) & \text{for } 0 \le t-k \le T^l \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

which now yields

$$\begin{aligned} \delta_j^l(k) &= f'(y_j^l(k)) \sum_{m=1}^{N_{l+1}} \sum_{t=k}^{T^l+k} \delta_m^{l+1}(t) w_{jm}^l(t-k) \\ &= f'(y_j^l(k)) \sum_{m=1}^{N_{l+1}} \sum_{n=0}^{T^l} \delta_m^{l+1}(k+n) w_{jm}^l(n) \\ &= f'(y_j^l(k)) \cdot \sum_{m=1}^{N_{l+1}} \Delta_m^{l+1}(k) \cdot \mathbf{W}_{jm}^{l+1} \end{aligned} \tag{23}$$

where we have defined

$$\Delta_m^l(k) = \left[\delta_m^l(k), \delta_m^l(k+1), ..\delta_m^l(k+T^{l-1})\right]. \tag{24}$$

Summarizing, the complete adaptation algorithm can be expressed as follows:

$$\mathbf{W}_{ij}^l(k+1) = \mathbf{W}_{ij}^l(k) - \mu\delta_j^{l+1}(k) \cdot \mathbf{X}_i^l(k) \tag{25}$$

$$\delta_j^l(k) = \begin{cases} -2e_j(k)f'(y_j^L(k)) & l = L \\ f'(y_j^l(k)) \cdot \sum_{m=1}^{N_{l+1}} \Delta_m^{l+1}(k) \cdot \mathbf{W}_{jm}^{l+1} & 1 \le l \le L-1 \end{cases} \tag{26}$$

An immediate observation is that these equations are seen as the vector generalization of the already familiar backpropagation algorithm. In fact, if we replace the vectors $\mathbf{X}$, $\mathbf{W}$, and Δ by scalars, the above equations reduce to precisely the standard backpropagation algorithm for static networks. Differences, however, in the *temporal* version are a matter of implicit time relations and filtering operations. To calculate $\delta_j^l(k)$ for a given neuron we *propagate* the δ's from the next layer backwards through the synaptic filters for which the given neuron feeds (see Figure 9). Thus δ's are formed not by simply taking weighted sums but by backward filtering. For each new input and desired response vector the forward filters are incremented one time step and the backward filters one time step.

By having manipulated the terms used to accumulate the error gradients we have preserved the symmetry between the forward propagation of states and the backward propagation of error terms. The sense of parallel distributed processing is maintained. Furthermore, the algorithm overcomes the computational complexity encountered in the first algorithm. The number of operations only grows *linearly* with the number of layers and synapses in the network. This savings comes as a consequence of the recursion which efficiently groups terms into products of sums instead of sums of products. Each unique coefficient is used

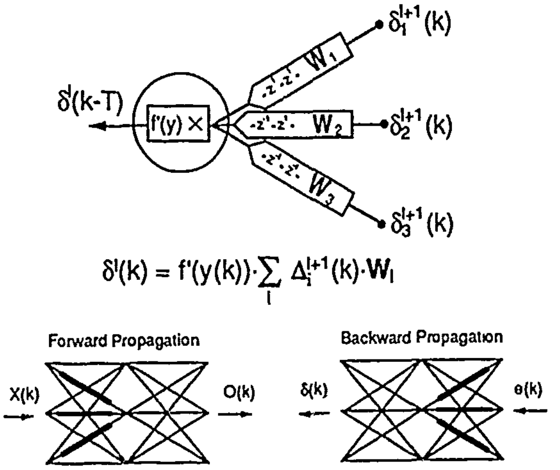


Figure 9: Backward filter propagation of gradient terms

only once in the calculation. There is no redundant use of terms as in the first case. Returning to the TDNN example where 521 weights expanded into 6233 virtual weights, the use of temporal backpropagation could result in a reduction in computer training time by as much as 90%.

3.2.1 Noncausality

The specification of the temporal backpropagation algorithm in Equation 26 hides an important subtlety encountered when actually implementing the algorithm. Careful inspection reveals that the calculations for the $\delta_j^{l-1}(k)$'s are in fact noncausal. The source of this noncausal filtering can be seen by considering the definition of $\delta_j^{l-1}(k) = \partial e^2/\partial y_j^l(k)$. Since it takes time for the output of any internal neuron to completely propagate through the network, the change in the *total* error due to a change in an internal state is a function of future values within the network. Since the network is FIR only a finite number of future states must be considered.

The exact time reference taken for adaptation purposes is not important. Making the system causal becomes a standard engineering task. This can be accomplished in a number of different ways by adding a finite number of simple delay operators at various locations within the network. One possible solution follows if we require that all weights vectors are to be adapted based on only the current error $e^2(k)$ and past values of the error. Given this information we may immediately form $\delta^L(k)$ for the last layer in order to adapt the weights in the last layer. For the next layer back, however, causality constraints imply that terms are only available to calculate

$$\delta_j^{L-1}(k-T) = f'(y_j^{L-1}(k-T)) \cdot \sum_{m=1}^{N_{l+1}} \Delta_m^L(k-T) \cdot \mathbf{W}_{jm}^L \tag{27}$$

based on the requirement $\Delta(k-T) = [\delta(k-T), \delta(k-T+1), ..\delta(k)]$ be composed of only current and past terms.

Since at time k only the time shifted value $\delta^{L-1}(k-T)$ can be computed, the states $\mathbf{X}^{L-2}(k-T)$ must be stored so that the product of the two may be formed to adapt the synapses in the layer. Continuing one more layer back, the time shift is simply twice as long. Rewriting the algorithm in this causal form yields

$$\mathbf{W}_{ij}^{L-n}(k+1) = \mathbf{W}_{ij}^{L-n}(k) - \mu\delta_j^{L+1-n}(k-nT) \cdot \mathbf{X}_i^{L-n}(k-nT)$$

$$\delta_j^{L-n}(k-nT) = \begin{cases} -2e_j(k) f'(y_j^L(k)) & n = 0 \\ f'(y_j^{L-n}(k-nT)) & 1 \le n \le L-1 \\ \quad \cdot \sum_{m=1}^{N_{l+1}} \Delta_m^{L+1-n}(k-nT) \cdot \mathbf{W}_{jm}^{n+1-L} & \end{cases} \tag{28}$$

While less aesthetically pleasing than the earlier equations, they differ only in terms of a change of indices. Summarizing then, we propagate the delta terms backward continuously *without* delay. However, by definition this forces the internal values of deltas to be shifted in time. Thus one must buffer the states $\mathbf{X}(k)$ appropriately to form the proper terms for adaptation. Added storage delays are necessary only for the states $\mathbf{X}(k)$. The backward propagation of delta terms require no added delays and is still symmetric to the forward propagation.

For simplicity in the above equations it was assumed that the order of each synaptic filter, T, was the same in each layer. This is clearly not necessary. If the order is different for each layer in the network we simply replace nT by

$$nT \leftrightarrow \sum_{l=L-n}^{L-1} T^l. \tag{29}$$

For the general case let T_{ij}^l be the order of the synaptic filter connecting neuron i in layer l to neuron j in the next layer. Then for the case of arbitrary synaptic filter order we have

$$nT \leftrightarrow \sum_{l=L-n}^{L-1} \max_{ij}\{T_{ij}^l\}. \tag{30}$$

The basic rule is that the time shift for the delta associated with a given neuron is equal to the total number of tap delays along the longest path to the output of the network.

3.3 DIFFERENCES IN ALGORITHMS

Both the first algorithm presented and the temporal backpropagation algorithm are based on gradient descent. They are not, however, equivalent. All gradient

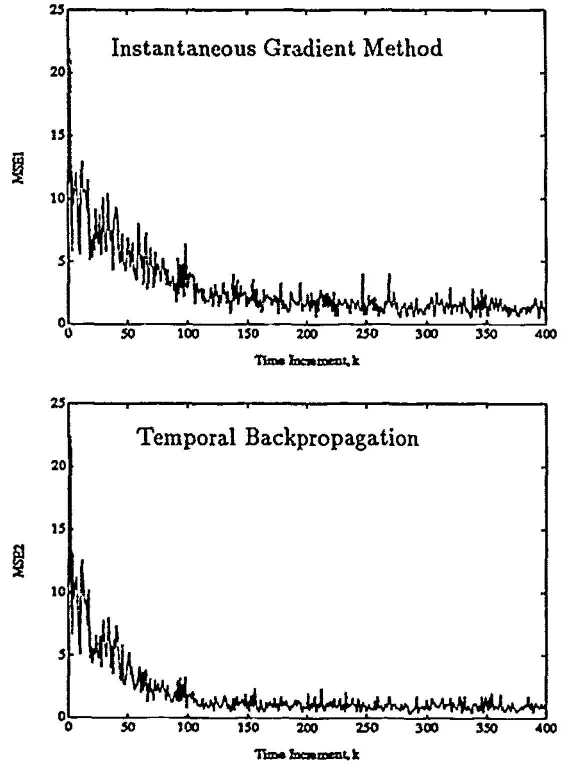


Figure 10: Learning Curves

derivations assumed that the weight parameters were fixed. During actual adaptation this is clearly not a valid assumption. Minor discrepancies in performance arise due to differences in the timing at which weights are adjusted relative to the calculation of the error gradients. Figure 10 shows averaged learning curves for the two algorithms used on a two layer network[2] modeling an unknown nonlinear system. Even for this simple example where combinatorial problems are not an issue, temporal backpropagation resulted in roughly a 40.5% reduction in computer simulation time. Furthermore, initial experimentation indicates the added benefit of less misadjustment using temporal backpropagation. For "small" learning rates differences in the learning characteristics are negligible. Mathematically the algorithms become equivalent as $\mu \rightarrow 0$. Alternatively, terms may be accumulated to find the total gradient for batch mode adaptation. In this case the algorithms are functionally identical apart from specific implementation and computational differences.

4 APPLICATIONS

By modeling each synapse as a linear filter, the neural network as a whole, may be thought of as an adaptive system with its own internal dynamics. Applications should thus include areas of signal processing, control, and pattern recognition where there is an inherent temporal quality to the data.

In nonlinear controls, for example, the design of a neural controller is commonly a two stage process in which both a feedforward controller and a system emulator are separately learned. A static neural network can be used for both components under the assumption that full state knowledge is available. However, the use of an FIR structure for both the emulator and controller can allow for the estimation of internal states based on available output observations.

As stated earlier, the definition of the FIR Neural Network also encompasses Time-Delay Neural Networks which have been investigated for use in speech recognition. TDNN's have been shown to be comparable to the more traditional method of Hidden Markov Models for identification of phonemes (Waibel, 1989). The temporal backpropagation algorithm may be used as a simple means for greatly improving the training time of such networks. An interest in implementing the algorithm for future versions of the TDNN speech recognition system has already been indicated (Hampshire, 1990).

5 CONCLUSION

This paper has defined an FIR neural network structure. A more sophisticated model of the synapse is used which replaces the static weight model by a linear Finite Impulse Response filter. A computationally efficient gradient descent algorithm was derived which was seen to be a temporal generalization of the familiar backpropagation algorithm.

References

B. Widrow and S. D. Stearns. (1985) *Adaptive signal processing,* Englewood Cliffs, NJ, Prentice Hall.

D. E. Rumelhart, J. L. McClelland, and the PDP Research Group. (1986) *Parallel Distributed Processing. Explorations in the Microstructure of Cognition*, Vol. 1, The MIT Press, Cambridge, MA.

A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. (1989) *Phoneme Recognition Using Time-Delay Neural Networks,* IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 37, No. 3, pp. 328-339, March 1989.

E. A. Wan. (1990) *Temporal Backpropagation for FIR Neural Networks,* International Joint Conference on Neural Networks, Vol. 1, pp. 575-580, San Diego.

J. Hampshire. (1990) *personal communications.*

[2] 1 input, 1 output, and 5 hidden units with 4-tap FIR filters for each synapse. $\mu = .05$

Part V

Theory and Analysis

Optimal Dimensionality Reduction Using Hebbian Learning

Asriel Levin
Department of Electrical Engineering
Yale University
P.O. Box 1968, Yale Station
New Haven, CT 06520

Abstract

The problem of learning efficient representation of a high dimensional input by a lower dimensional space is investigated. It is shown that a recurrent system of interconnected linear neurons, using Hebbian learning rule to update its connections, will converge w.p.1 to an orthogonal projection onto the space generated by the first q principal components of the p dimensional input covariance matrix. The problem is analytically tractable since the input-output map is expressed by a linear equation.

1 Introduction

The subject of dimensionality reduction has received extensive attention in the statistics literature (Jolliffe,1986) and has proven to be extremely useful when large dimensional data needs to be analyzed. In recent years a few algorithms were suggested, trying to achieve efficient dimensionality reduction using network techniques. Oja (Oja,1982) has shown that a linear neuron (i.e an element whose output is a weighted sum of its inputs) will extract the first principal component of the covariance matrix of its inputs if the weights are modified using a variation of the Hebb rule:

$$c_i(t+1) = c_i(t) + \gamma(t)[u_i(t) - c_i(t)y(t)]y(t)$$

c - the weight vector.
u - the input vector.
y - the output.
γ - learning coefficient.

An algorithm extracting the first q principal components was proposed in (Sanger,1989). The method suggested extracts the principal components sequentially and subtracts them from the input thus errors tend to accumulate and only the first few components can be estimated accurately.

In this paper we propose a new algorithm extending Oja's result to learning an orthogonal projection onto the space spanned by the first q principal components. First, the problem of dimensionality reduction is analyzed and conditions for global optimality for certain class of mappings are established. Using stochastic algorithm theory, it is shown that a two layered network with feedback connections and local Hebb type learning rule is guaranteed to converge to the optimal reduced representation. Learning is done on-line using local unsupervised rules, thus the algorithm may be used both as a practical method for data compression and as a building block in a network enabling it to achieve efficient internal representations. Self supervised backpropagation in a linear network would give similar results (Baldi & Hornik,1989) but would require error propagation through the layers.

In section 2 the problem is formally posed. Main results and proposed architecture for implementing the algorithm are stated in section 3. All proofs are deferred to the appendix.

2 Statement of The Problem

Similarly to self supervised BP (Rumelhart et al,1986) we define a representation to be optimal if given the reduced dimensional output we are able to reconstruct the original input with minimal mean square error. More precisely:

Given u a p dimensional random vector we wish to find f : $\mathcal{R}^p \rightarrow \mathcal{R}^q$ and g : $\mathcal{R}^q \rightarrow \mathcal{R}^p$ ($q < p$) such that

$$E\{\|\mathrm{u} - \mathrm{g} \circ \mathrm{f}(\mathrm{u})\|^2\}$$

is minimized.

In this paper we will restrict ourselves to the case where g is linear, i.e it can be represented by a $p \times q$ matrix C.

Defining $\mathbf{y} \equiv \mathbf{f}(\mathbf{u})$, our goal is to minimize

$$J = \mathbf{E}\{\|\mathbf{u} - \mathbf{C}\mathbf{y}\|^2\}$$

with respect to $\mathbf{C}$ and $\mathbf{y}$.

3 Main Results

1. **Optimal Feedforward Mapping.**
   If the feedback mapping is linear then the optimal feedforward mapping is also linear and is given by

   $$\mathbf{y} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{u} \equiv \mathbf{B}\mathbf{u}$$

2. **Learning Rule for C.**
   If $\mathbf{C}$ is adjusted according to:

   $$\mathbf{C}(t{+}1) = \mathbf{C}(t){+}\gamma(t)[\mathbf{u}(t){-}\mathbf{C}(t)\mathbf{y}(t)]\mathbf{y}^T(t)\ |_{\mathbf{y}=\mathbf{B}\mathbf{u}}$$

   Where γ is a decreasing sequence (like $1/t$) then $\mathbf{C}$ (and thus $\mathbf{y}$) will converge w.p.1 to a global minima of J.
   Note: As proved in (Baldi et al,1989) this means that the columns of $\mathbf{C}$ will span the space generated by the first q principal components of $\mathbf{E}\{\mathbf{u}\mathbf{u}^T\}$.

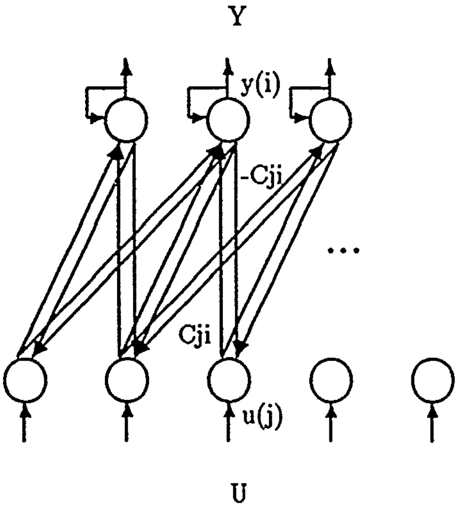


Figure 1: Diagrammatic Representation of Proposed architecture

3. **Proposed Algorithm.**
   It can be proved that with the above learning rule $\mathbf{C}^T\mathbf{C}$ is constant.Thus the optimal solution is not unique and depends on the initial value chosen for C. Particularly, choosing $\mathbf{C}^T\mathbf{C} = I$ will simplify the expression of $\mathbf{y}$ to $\mathbf{y} = \mathbf{C}^T\mathbf{u} = \mathbf{C}^T(\mathbf{u} - \mathbf{C}\mathbf{y}) + \mathbf{y}$. (An implicit assumption is that the delays in the system are neglegible compared with the intervals on which the input u is fixed). Define $\mathbf{x} \equiv \mathbf{u} - \mathbf{C}\mathbf{y}$. Combining all the above we get the following set of equations:

   $$\mathbf{x}(t) = \mathbf{u}(t) - \mathbf{C}(t)\mathbf{y}(t)$$
   $$\mathbf{y}(t+1) = \mathbf{y}(t) + \mathbf{C}^T(t)\mathbf{x}(t)$$
   $$\mathbf{C}(t+1) = \mathbf{C}(t) + \gamma(t)\mathbf{x}(t)\mathbf{y}^T(t)$$

   and the update rule for $\mathbf{C}$ becomes the familiar Hebb rule. The above algorithm is realized by a two layered linear network with antisymmetric connections between the layers. $\mathbf{x}$ is the input layer consisting of p units and $\mathbf{y}$ is the output layer consisting of q units. Because of the degenerate form of $\mathbf{y}$ stability is trivial.
   Note: If $\mathbf{y}$ is a scalar ($q = 1$) this reduces to the update rule proposed in (Oja,1982)

4 Discussion

A new, very easy to implement algorithm was proposed. Due to its local and on-line nature it should prove very useful when large dimensional data needs to be analyzed.

By limiting ourselves to linear feedback mapping the whole problem became linear and thus analytically tractable. Though no assumptions concerning the statistics of $\mathbf{u}$ where made, it is well known that this restricted class of mappings will be optimal only if $\mathbf{u}$ is a gaussian process. In many applications gaussian distribution can be assumed.

For other types of distributions the above would give only the optimal linear estimation. To get global optimality, more general feedback mappings would need to be considered. In such cases self supervised backpropagation through layered networks consisting of nonlinear elements might prove better (e.g (Saund,1989)) but for a specific task many layers may be required making the backpropagation method prohitively slow.

Thus, a further application might be achieved by inserting the above architecture as a component in a larger network containing nonlinear elements. As proposed in (Ballard,1987), achieving efficient internal representations should speed up learning rate, helping to overcome the inherent deficiencies of supervised methods. This direction is currently being pursued, trying to incorporate the effect of sigmoidal non-linearities in the above learning rules.

5 appendix

5.1 Proof of result 1

If $\mathbf{u}$ is ergodic J can be approximated by

$$J = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \|\mathbf{u}(n) - \mathbf{C}\mathbf{y}(n)\|^2$$

Given $\mathbf{u}, \mathbf{C}$ for $\mathbf{y}$ to be optimal

$$\delta J(\mathbf{y}, \Delta\mathbf{y}) = 0 \forall \Delta\mathbf{y} \in l^\infty$$

or

$$\lim_{\epsilon\to 0} \frac{dJ(\mathbf{C}, \mathbf{y}+\epsilon\Delta\mathbf{y})}{d\epsilon} = \lim_{\epsilon\to 0} \lim_{N\to\infty} \sum_{n=1}^{N} \mathbf{C}^T(\mathbf{u}-\mathbf{C}\mathbf{y})\Delta\mathbf{y}$$

Since $\Delta\mathbf{y}$ is arbitrary we get $\delta J = 0$ iff $\mathbf{C}^T(\mathbf{u} - \mathbf{C}\mathbf{y}) = 0$ And if the columns of $\mathbf{C}$ are linearly independent the unique solution for $\mathbf{y}$ is:

$$\mathbf{y} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{u} \equiv \mathbf{B}\mathbf{u}$$

5.2 Proof of result 2

Define $\tilde{J} \equiv J(\mathbf{C}, \mathbf{B}\mathbf{u})$. If $\mathbf{C}$ is adjusted in the gradient direction of $\tilde{J}$ i.e.

$$\frac{d\mathbf{C}}{dt} \alpha \frac{-d\tilde{J}}{d\mathbf{C}}$$

we are guaranteed to converge to an equilibrium point of $\tilde{J}$.

$$\frac{d\tilde{J}}{d\mathbf{C}} = \lim_{N\to\infty} \frac{1}{N} \lim_{\epsilon\to 0} \sum_{n=1}^{N} \frac{d}{d\epsilon} \|\mathbf{u} - (\mathbf{C}+\epsilon\Delta\mathbf{C})(\mathbf{y}+\epsilon\Delta\mathbf{y}\Delta\mathbf{C})\|^2|_{\mathbf{y}=\mathbf{B}\mathbf{u}}$$

$$\Delta\mathbf{C} \in L(R^q, R^p)$$

Ignoring second order terms in ϵ we get

$$\frac{d\tilde{J}}{d\mathbf{C}} = \lim_{N\to\infty} \frac{1}{N} [\sum_{n=1}^{N} (\mathbf{u} - \mathbf{C}\mathbf{y})\mathbf{y}^T \Delta\mathbf{C}^T + \sum_{n=1}^{N} \mathbf{C}^T(\mathbf{u} - \mathbf{C}\mathbf{y})]|_{\mathbf{y}=\mathbf{B}\mathbf{u}}$$

Second term is identically 0 so if we choose

$$\Delta\mathbf{C} = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} (\mathbf{u} - \mathbf{C}\mathbf{y})\mathbf{y}^T$$

we get

$$\frac{d\tilde{J}}{d\mathbf{C}} = -\epsilon(\Delta\mathbf{C})^2 < 0$$

and $\tilde{J}$ will converge to an equilibrium point. For general linear mappings $\mathbf{C}', \mathbf{B}'$ the landscape of

$$J' =< \|\mathbf{u} - \mathbf{C}'\mathbf{B}'\mathbf{u}\|^2 >$$

was investigated by (Baldi and Hornik,1989). Their main result is summarized in the following theorem.

Theorem: $\mathbf{B}', \mathbf{C}'$ correspond to a critical point of J' iff

$$\mathbf{C}' = \mathbf{A_q}\mathbf{W}$$

$$\mathbf{B}' = \mathbf{W}^{-1}\mathbf{A_q}$$

$$\mathbf{D} \equiv \mathbf{C}'\mathbf{B}' = \mathbf{P}_{\mathbf{A_q}}$$

$\mathbf{A_q}$ - a $p \times q$ matrix whose columns consist of q Principal Components of Q.
$\mathbf{P}_{\mathbf{A_q}}$ - Projection matrix onto the space spanned by the columns of $\mathbf{A_q}$
W - Any $q \times q$ invertible matrix.
If $\mathbf{D}$ is a projection to the space spanned by the first q Principal Components then it corresponds to a local and global minima of J'. All other critical points are saddle points.

From result 1 we know that y would be optimal if $y = Bu$. Hence, minimizing J' is equivalent to minimizing J.

Now $\tilde{J}$ has the same equilibria points as J and further we can prove:
Lemma:A pair $\mathbf{C}, \mathbf{B}(\mathbf{C})$ is a local minimum of $\tilde{J}$ iff it is a local minimum of J'.
The proof is long and not central to the derivation so it will not be given here.

So far calculation of $\frac{d\mathbf{C}}{dt}$ involved summation over long periods. To make the scheme more feasible we can used some results from stochastic algorithm theory (Ljung,1977) that in this context would translate to:
Theorem: Let $\gamma(t)$ be a sequence of positive numbers satisfying:

$$\sum \gamma(t) \to \infty \tag{1}$$

$$\sum \gamma(t)^2 < \infty \tag{2}$$

$$\gamma(t) \text{decreasing} \tag{3}$$

$$\lim_{t\to\infty} \sup[\frac{1}{\gamma(t)} - \frac{1}{\gamma(t-1)}] < \infty \tag{4}$$

If C is updated according to the rule

$$\mathrm{C}(t+1) = \mathrm{C}(t) + \gamma(t)[\mathrm{u}(t) - \mathrm{C}(t)\mathrm{y}(t)]\mathrm{y}^T(t) \; |_{\mathrm{y}=\mathrm{Bu}}$$

it will converge w.p.1 to a local minimum of $\tilde{J}$.

And thus, togather with the derivation above (i.e minima of $\tilde{J}$ coincide with minima of J and all minima are global) the above update rule would guarantee convergence to a global minima of J w.p.i.

5.3 Derivation of Proposed Algorithm

To complete the derivation we only need to prove that C^TC is constant.

$$\frac{d(C^TC)}{dt} = \dot{C}^TC + C^T\dot{C} =$$

$$y(y^TC^T - U^T)C + C^T(u - Cy)y^T \,|_{y=Bu}=$$

$$(c^TC)^{-1}C^Tu[u^TC(C^TC)^{-1}C^T - u^T]C$$

$$+C^T[u - C(C^TC)^{-1}C^Tu]u^TC(C^TC)^{-1} = 0$$

6 references

P. Baldi, K. Hornik. *Neural Network and Principal Component Analysis: Learning From Examples Without Local Minima.* Neural Networks, 2(1):53-58, 1989.

D.H. Ballard. *Modular Learning in Neural Networks.* Proc. Sixth National Conference on AI (AAAI-87), vol. 1: 279-284, 1987.

I. T. Jolliffe. *Principal Component Analysis.* Springer-Verlag, 1986.

L. Ljung. *Analysis of Recursive Stochastic Algorithms.* IEEE T-AC, Vol AC-22(4):551-575, 1977.

D.G. Luenberger. *Optimization by Vector Space Methods.* John Wiley, 1969.

E. Oja. *A Simplified Neuron as a Principal Component Analyzer.* J. Mathematical Biology, 15:267-273, 1982.

D.E. Rumelhart, G.E. Hinton & R.J. Williams. *Learning Internal Representations by Error Propagation.* In Parallel Distributed Processing, MIT Press, 1986.

T.D. Sanger. *Optimal Unsupervised Learning in Feedforward Neural Networks* MIT, M.Sc Thesis, 1989.

E. Saund. *Dimensionality Reduction Using Connectionist Networks.* IEEE T-PAMI, Vol 11:304-314, 1989.

Basis-Function Trees for Approximation in High-Dimensional Spaces

Terence D. Sanger
Electrical Engineering and Computer Science Dept.
Massachusetts Institute of Technology
Cambridge, MA 02139
tds@ai.mit.edu

Abstract

I describe a new algorithm for approximating continuous functions in high-dimensional input spaces. The algorithm builds a tree-structured network of variable size which is determined both by the distribution of the input data and by the function to be approximated. Efficient computation in this tree structure takes advantage of the potential for low-order dependencies between the output and the individual dimensions of the input. This algorithm is related to the ideas behind k-d trees (Bentley, 1975), CART (Breiman *et al.*, 1984), and MARS (Friedman, 1988). I present two examples.

1 INTRODUCTION

Basis function networks have proven to be useful for approximating functions in a variety of different domains (for reviews, see (Poggio and Girosi, 1989, Powell, 1987, Klopfenstein and Sverdlove, 1983)). Such networks are represented by equations of the form:

$$\hat{y} = \sum w_i \varphi_i(x) \qquad (1)$$

where y is the desired (scalar) output for input $x \in R^p$, $\hat{y}$ is the output approximated by the network, and the w_i's are learned scalar weights. The φ_i's are often radial basis functions of the form

$$\varphi_i(x) = \varphi(\|x - \xi_i\|) \qquad (2)$$

whose value depends only on the distance of the input x from the "center" ξ_i.

When the dimension p of the input space is high, the work required to calculate the output $\varphi_i(x)$ of any basis function increases. In addition, since the size of the space increases geometrically with p, the data will be very sparsely placed and the estimate $\hat{y}$ will be undefined for most inputs unless a prohibitively large number of basis functions is used. This problem is often referred to as the "curse of dimensionality". One might attempt to avoid such problems by noting that in some regions of the input space, the desired output function can be approximated using only a few dimensions of the input. This would occur if, for instance, the data were to lie on the line $x_2 = f(x_1)$ for any 1-1 function f, in which case the desired output y could be estimated from either x_1 or x_2.

In this report, I describe one method for reducing computational work which makes use of this idea. It is applicable in the case where the basis functions are separable, in the sense that they can be written

$$\varphi_i(x) = \phi_{r_1^i}(x_1) \cdots \phi_{r_p^i}(x_p) \qquad (3)$$

where x_j is the j^{th} component of x,

$$\phi_{r_d^i}(x_d) = \varphi(|x_d - (\xi_i)_d|)$$

is a scalar function of scalar input, and r_d^i specifies which basis function ϕ should be chosen along the j^{th} dimension. (An example of separable basis functions is the radially-symmetric Gaussian basis.) In this equation and in the rest of this report I assume for simplicity that the centers of the basis functions are located on a fixed regular lattice with N along each dimension so that there is a total of N^p basis functions. However, the technique can also be applied to basis functions with arbitrarily fixed centers, in which case, each function $\phi_{r_d^i}(x_d)$ is the projection of $\varphi_i(x)$ along the x_d dimension.

2 NETWORK STRUCTURE

In order to understand the structure of the proposed network, I will build the approximation up one dimension at a time. If the output can be determined from only the x_1 dimension, then it can be written

$$\hat{y}^{[1]} = \sum_{n=1}^{N} \alpha_n \phi_n(x_1) \qquad (4)$$

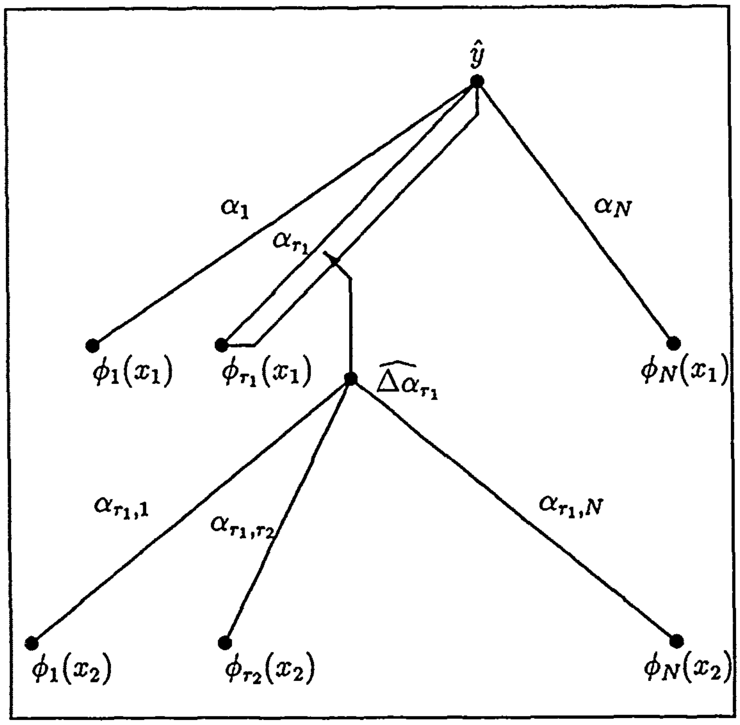


Figure 1: Two-layer tree constructed by the algorithm. x's are inputs, ϕ's are basis functions, and α's are weights. See text for explanation.

where the α_n's are the weights for the basis functions $\phi_n(x_1)$ along this dimension, and the superscript indicates that only one dimension has been used to compute the output.

In order to train the weights, we can use the LMS learning algorithm (Widrow and Hoff, 1960) to reduce the mean-squared output error $E[(y - \hat{y}^{[1]})^2]$ by

$$\Delta\alpha_n = \gamma(y - \hat{y}^{[1]})\phi_n(x_1) \tag{5}$$

where γ is a small rate term. Given sufficient input samples x_1, this algorithm will converge until the average value $E[\Delta\alpha_n] = 0$ for all n. The output $\hat{y}^{[1]}$ will then be the best linear approximation to y based on the values $\phi_n(x_1)$.

If y cannot be well approximated using only the value of x_1, then there will be some residual error $(y - \hat{y}^{[1]})$. Although this error will be uncorrelated with $\phi_n(x_1)$ for all n, the variances $E[(\Delta\alpha_n)^2]$ will be non-zero, indicating that there is pressure to change the weights. Although on average this pressure is 0, for particular instances the output error would be reduced if the weights could be either larger or smaller. We can thus improve the approximation of y by allowing the weights to vary based on information from the x_2 dimension. Suppose we pick r_1 to be the value of n for which $E[(\Delta\alpha_n)^2]$ is largest. We now want to add a term

$$\widehat{\Delta\alpha}_{r_1} = \sum_{n=1}^{N} \alpha_{r_1,n}\phi_n(x_2) \tag{6}$$

to the weight α_{r_1} which varies with x_2 (see figure 1). We use the LMS rule to learn the weights $\alpha_{r_1,n}$ according to

$$\Delta\alpha_{r_1,n} = \Delta\alpha_{r_1}\phi_n(x_2) \tag{7}$$

where the "delta" term is given by $\Delta\alpha_{r_1}$. This is equal to

$$\gamma(y - \hat{y})\phi_{r_1}(x_1)\phi_n(x_2) \tag{8}$$

and we see that the error term for the product basis function $\phi_{r_1}(x_1)\phi_n(x_2)$ is exactly the term which the regular LMS rule would supply. The approximation at the output is now

$$\begin{aligned}\hat{y}^{[2]} &= \hat{y}^{[1]} + \phi_{r_1}(x_1)\widehat{\Delta\alpha}_{r_1} \\ &= \hat{y}^{[1]} + \phi_{r_1}(x_1)\sum_{n=1}^{N}\alpha_{r_1,n}\phi_n(x_2)\end{aligned}$$

This procedure can be followed for every value of n at each of the p dimensions. This leads to the following recursive learning rule:

$$\begin{aligned}\Delta\alpha_0 &= \gamma(y - \hat{y}) \\ \Delta\alpha_{r_1} &= \gamma(y - \hat{y})\phi_{r_1}(x_1)\end{aligned}$$

$$\begin{aligned} &= (\Delta\alpha_0)\phi_{r_1}(x_1) \\ \Delta\alpha_{r_1,r_2} &= \gamma(y-\hat{y})\phi_{r_1}(x_1)\phi_{r_2}(x_2) \\ &= (\Delta\alpha_{r_1})\phi_{r_2}(x_2) \\ &\dots \\ \Delta\alpha_{r_1,\ldots,r_{d+1}} &= (\Delta\alpha_{r_1,\ldots,r_d})\phi_{r_{d+1}}(x_{d+1}) \end{aligned}$$

This formula makes it clear that the weight correction term $\Delta\alpha_{r_1,\ldots,r_d}$ functions like the error term $\gamma(y-\hat{y})$ for the next "layer", since the update equation for $\alpha_{r_1,\ldots,r_d,n}$ can be thought of as performing LMS learning with the target being given by $\Delta\alpha_{r_1,\ldots,r_d}$. The weights from successive basis functions are being trained to correct the weights of previous ones based on the context specified by additional dimensions of the input.

Note that this network structure is not limited to any particular set of basis functions. Any basis at all could have been used, and the choice of basis will determine the approximation ability of the network and the number of nodes needed to attain a given accuracy. Other possible bases include a Fourier basis along each dimension (the nodes will represent diagonally oriented filters), the eigenvectors of the input distribution (the nodes will represent cross-products of orthogonal outputs), the analog value x_d along each dimension (the nodes will represent monomials, and the network will find a polynomial approximation to y), or the individual bits of the binary representation of each input. If the basis is formed by dividing each dimension into disjoint regions along sharp boundaries, then the algorithm is exactly equivalent to a "k-d tree". The algorithm can thus be thought of as a generalization of the k-d tree to arbitrary overlapping or non-orthogonal bases.

The resulting network forms a tree of depth p where each node has N children. If the tree is completed, then the approximation $\hat{y}$ will contain all terms of the form $\alpha_{r_1,\ldots,r_p}\phi_{r_1}(x_1)\cdots\phi_{r_p}(x_p)$ and thus has at least as much approximation power as equation 1. $\hat{y}$ also contains additional terms involving combinations of fewer than p dimensions, and thus may have more approximating power.

However, if the tree is grown to its full size, there will be more coefficients than in equation 1, and this leads to more computational work. The work is decreased only if a sufficiently good approximation can be gained without growing the full tree. It thus makes sense to grow a subtree to modify a weight only when changing that weight would make a significant contribution to decreasing the output error for the network.

3 GROWING THE TREE

Since the ability of this type of network to save computation depends upon selectively growing the tree structure, the selection rule for determining when to create a subtree will determine the overall performance. Unfortunately, there is no general way to determine the optimal next subtree to grow. In the algorithm above, it was suggested that a tree be grown beneath the leaf of the current tree with maximum error variance given by

$$E\left[(\Delta\alpha_{r_1,\ldots,r_d})^2\right] \tag{9}$$

where the maximization is carried out between leaves at all depths d.

If we assume that the existing tree is fixed and the new subtree is made complete down to $p-d$ levels, then the decrease in expected error at the output $E[(y-\hat{y})^2]$ will be proportional to (9). In practice, it may not be practical to complete the subtree so that the weight error variance (9) is reduced to zero, but this error nevertheless is proportional to the maximal effect on the expected output error which a subtree at this node could have. It thus provides an upper bound on the usefulness of a partially grown subtree.

Unfortunately, because this heuristic is determined by the effectiveness of a theoretically perfect complete subtree, it does not tell us where to place the next single-layer subtree. In addition, if the desired y cannot in fact be approximated using this set of radial basis functions (or if there is significant noise), equation 9 will not predict the maximum effect of a subtree on the output. The subtree selection heuristic described here is thus intended only as a suggestion, and it is hoped that in particular applications more reliable selection methods could be found.

4 EXAMPLES

4.1 Predicting the Mackey-Glass Equation

As a simple example of performance, I will attempt to predict future values of the Mackey-Glass chaotic differential delay equation

$$\dot{x} = \frac{0.2x(t-30)}{1+x^{10}(t-30)} - 0.1x(t)$$

as suggested by (Farmer and Sidorowich, 1989, Lapedes and Farber, 1988). Many other authors have used networks to solve the same problem, including (Weigend *et al.*, 1990, Farmer and Sidorowich, 1989, Moody, 1989, Moody and Darken, 1989, Lapedes and Farber, 1988). I use the same parameters and method of error estimation as in (Farmer and Sidorowich, 1989) The network has six-dimensional input $x(t-6n)$ for $n = 0,\ldots,5$. Each input value is coded using 20 elements from the Fourier basis $sin(wx), cos(wx)$, $w = 1,\ldots,10$, as suggested in (Sanger, 1990). The task is for the network to learn to predict $x(t+6)$ while observing the continuously evolving time series. A new subtree was added (below the leaf with maximal error variance) every 400 time steps.

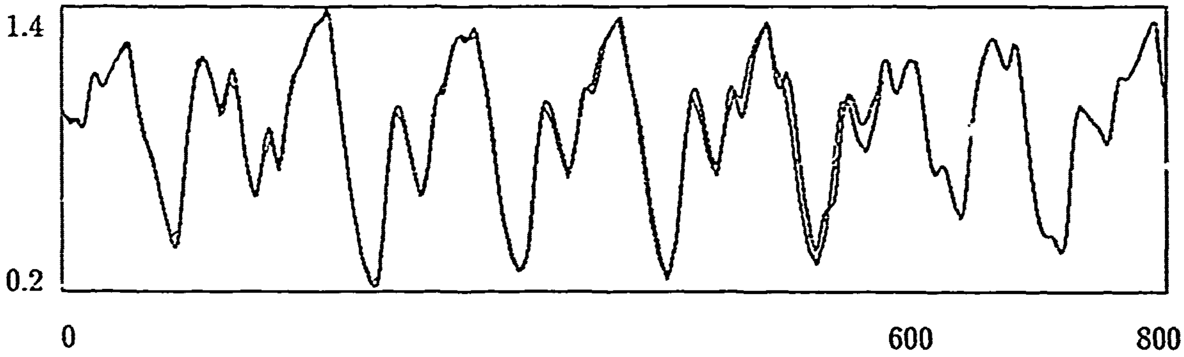


Figure 2. Mackey-Glass time series, 6-step ahead predictions (which are indistinguishable from the time series here), and iterated prediction time series up to 600 time steps into the future. The network has converged for 42,400 samples.

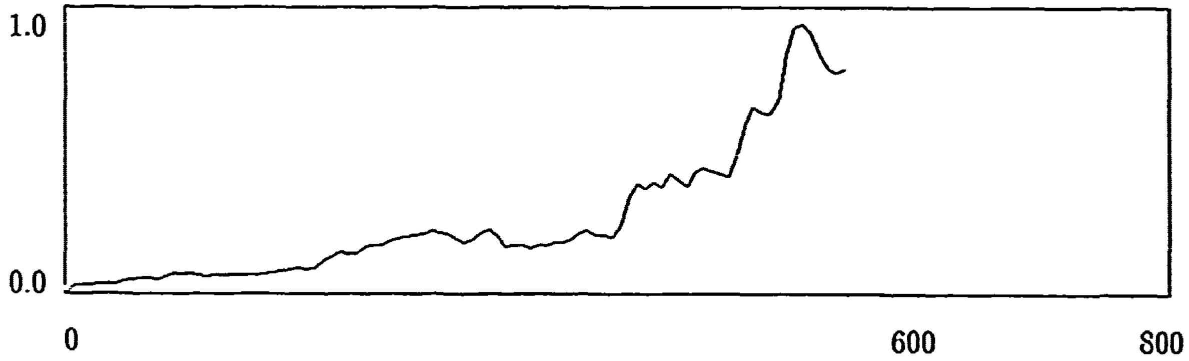


Figure 3. Iterated predictions for the Mackey-Glass equation, showing normalized mean-squared error as a function of prediction time up to 600 steps into the future.

Although the predictions and the true values are visually indistinguishable after only 20 subtrees have been added, the network was grown to 106 subtrees (about 40 minutes on a sparcstation) so that iterated predictions could be made. The 6-step normalized mean-squared error (NMSE, as defined in (Lapedes and Farber, 1988)) was 0.025, and iterated predictions 400 time steps into the future had NMSE $<$ 0.5. Figure 2 shows the time series, 6-step predictions, and 600 step iterated prediction time-series. Figure 3 shows NMSE as a function of iterated prediction time.

4.2 Nonlinear Image Filtering

For an example with higher-dimensional input, I applied the network to nonlinear smoothing of noisy images. 8-bit images of two faces were corrupted with additive random noise chosen from a uniform distribution between -64 and +63. The task was to predict the correct pixel value at the center of a 7x7 block of pixels, given only the 49 noisy pixel values. This is therefore a 49-dimensional problem. The network is given randomly selected 7x7 blocks of a noisy image as training inputs, and the correct center pixel (without noise) as the target output. A second image which did not provide training data was used to test generalization.

The network used a "bit-basis" of 16 basis functions per pixel consisting of the eight bits of the binary greylevel and their complements. While this basis is not particularly well-suited to the problem since it does not necessarily lead to smooth approximations, it has nice computational properties. In particular, no multiplications are ever performed in the network. Forward propagation consists of adding together all weights for which there exists an uninterrupted path of 1's at the nodes to the root (any 0 on this path would "multiply" the weight and cause it to have no effect.) Further, all weights were chosen to be integer-valued, so that weight updating involved only increment and decrement operations. The network output was scaled by a factor of 128 to slow down the learning rate.

Examples were chosen from random points throughout the training image, and noise was added independently to each example. A new subtree was added every 2000 examples, and the network was trained for a total of 40000 examples (20 subtrees). Figure 4 shows the original image, noisy image, filtered image, and residual error after only a single layer network has been built. This approximation is not much better than the optimal linear filter. Figure 5 shows the filtered image after 40 subtrees have been added, and demonstrates improved edge and detail resolution.

5 DISCUSSION

This formulation of basis function training has several advantages over more standard methods. It was motivated by an attempt to save computational work when approximating functions which can be calculated from only a few dimensions, and in this case both the learning time and the time required to compute the output $\hat{y}$ are reduced. If a minimum error is specified for approximation, then just enough α's can be calculated to achieve this criterion, and further terms do not need to be computed even if they do contribute.

An additional use for this network structure occurs when new dimensions may be added by the addition of, for example, new sensors. It is possible to construct the network so that the weights which have already been learned do not need to be re-learned to incorporate the new sensors. (Further improvement may be gained by modifying existing weights, but it will not be necessary to start over from scratch.)

As described here, the algorithm imposes an order on the dimensions, and if the dimension x_p is the most useful, the entire tree will have to be grown merely to access it. To avoid this problem, one can provide the entire basis set $\{\phi_n(x_d)\}_{n=1,\ldots,N}^{d=1,\ldots,p}$ at each level. This increases the size of the network by a factor of p, but it eliminates the need to choose an ordering of the dimensions, and hopefully will reduce the depth of the required tree.

There are several other network algorithms which are related to the one proposed here. Basis function approximation is a well-known technique in statistics, as is approximation by polynomials of increasing order (Gabor, 1961). The Perceptron algorithm (Rosenblatt, 1962) and Backpropagation (Rumelhart *et al.*, 1986) are related since they are both variants of the LMS rule (Widrow and Hoff, 1960). There are several algorithms which grow similar tree structures (Breiman *et al.*, 1984, Sun *et al.*, 1988, Bentley, 1975, Knerr *et al.*, 1990, Tenorio and Lee, 1989, Fahlman and Lebiere, 1990), although most (except (Fahlman and Lebiere, 1990)) are intended for classification tasks rather than approximation. There also exist algorithms for which the output of one network controls the behavior of another (Jacobs *et al.*, 1990, Hinton *et al.*, 1986). Perhaps the most closely related algorithm is the MARS algorithm (Friedman, 1988) which builds a tree of outdegree 2 using basis functions which are truncated polynomials. The tree structure is grown at each level by testing all possible subtrees and selecting the best according to a "lack of fit" criterion to the data.

The work I have presented allows a simple shortcut for approximating certain types of functions with a certain type of basis function network. It is easy to implement and forms a direct extension of standard

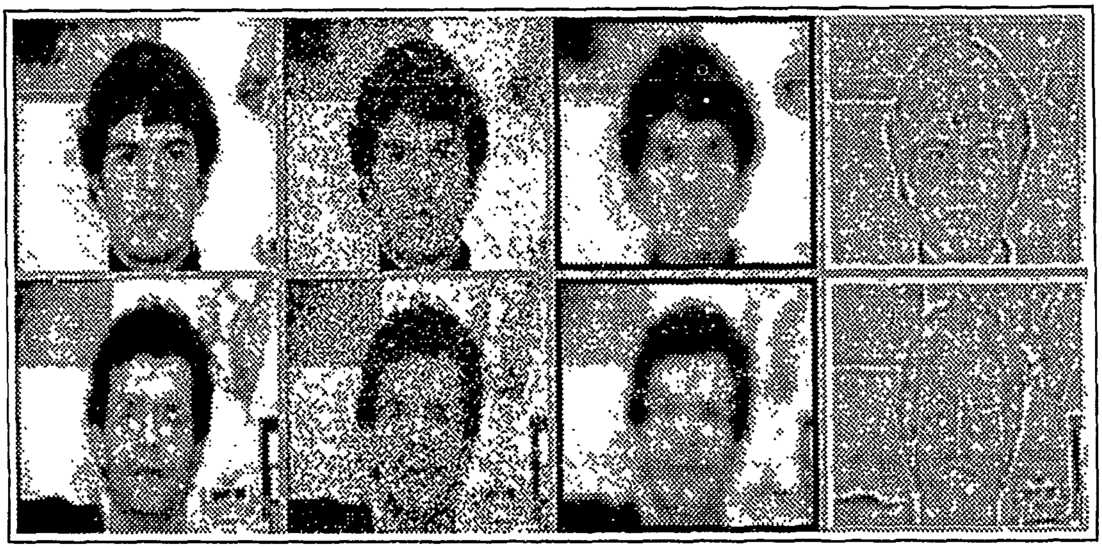


Figure 4· Nonlinear filtering with a single-layer tree. Top is training image, bottom is test image. From left to right are the original image, the noise-corrupted image, the filtered reconstruction, and the residual error.

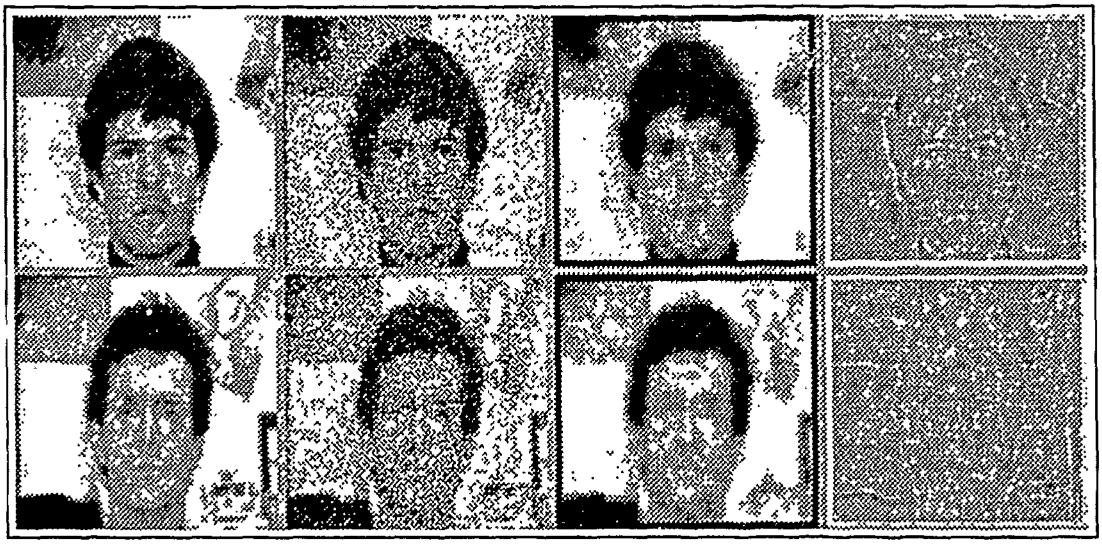


Figure 5: Nonlinear filtering with 20 subtrees. Top is training image, bottom is test image.

implementations of the LMS algorithm. It does not solve the "curse of dimensionality", but in applications it may make the use of basis function networks for high-dimensional input spaces practical.

Acknowledgements

This work started during a course taught at MIT by Chris Atkeson, Michael Jordan, and Marc Raibert. The image filtering task was first suggested to me by Ted Adelson, and Matthew Turk kindly provided the face images. This report describes .esearch done within the laboratory of Dr. Emilio Bizzi in the department of Brain and Cognitive Sciences at MIT. The author was supported during this work by the division of Health Sciences and Technology, and by NIH grants 5R37AR26710 and 5R01NS09343 to Dr. Bizzi. Parts of this report have been simultaneously submitted for publication in *Neural Computation.*

References

Bentley J. H., 1975, Multidimensional binary search trees used for associated searching, *Communications ACM*, 18(9):509-517.

Breiman L., Friedman J., Olshen R., Stone C. J., 1984, *Classification and Regression Trees*, Wadsworth Belmont, California.

Fahlman S. E., Lebiere C., 1990, The cascade-correlation learning architecture, Technical Report CMU-CS-90-100, Carnegie Mellon School of Computer Science, Pittsburgh.

Farmer J. D., Sidorowich J. J., 1989, Predicting chaotic dynamics, In Kelso J. A. S., Mandell A. J., Shlesinger M. F., ed.s, *Dynamic Patterns in Complex Systems*, pages 265-292, World Scientific.

Friedman J. H., 1988, Multivariate adaptive regression splines, Technical Report 102, Stanford Univ. Lab for Computational Statistics.

Gabor D., 1961, A universal nonlinear filter, predictor, and simulator which optimizes itself by a learning process, *Proc. IEE*, 108B:422-438.

Hinton G. E., McLelland J. L., Rumelhart D. E., 1986, Distributed representations, In McLelland J. L., Rumelhart D. E., The PDP Research Group , ed.s, *Parallel Distributed Processing*, pages 77-109, MIT Press, Cambridge, MA.

Jacobs R. A., Jordan M. I., Barto A. G., 1990, Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks, Technical Report COINS TR 90-27, U. Mass., Amherst.

Klopfenstein R. W., Sverdlove R., 1983, Approximation by uniformly spaced gaussian functions, In Chui C. K., Schumaker L. L., Ward J. D., ed.s, *Approximation Theory IV*, pages 575-580, Academic Press.

Knerr S., Personnaz L., Dreyfus G., 1990, Single-layer learning revisited. A stepwise procedure for building and training a neural network, manuscript.

Lapedes A., Farber R., 1988, How neural nets work, In Anderson D. Z., ed., *Neural Information Processing Systems*, pages 442-456, Am. Inst. Physics, NY, Proceedings of the Denver, 1987 Conference.

Moody J., Darken C., 1989, Fast learning in networks of Locally-Tuned processing units, *Neural Computation*, 1:281-294.

Moody J., 1989, Fast learning in Multi-Resolution hi erarchies, In Touretzky D. S., ed., *Advances in Neural Information Processing Systems 1*, pages 29-39, Morgan Kaufmann, San Mateo, CA.

Poggio T., Girosi F., 1989, A theory of networks for approximation and learning, MIT AI Memo 1140.

Powell M. J. D., 1987, Radial basis functions for multivariable interpolation: A review, In Mason J. C., Cox M. G., ed.s, *Algorithms for Approximation*, pages 143-167, Clarendon Press, Oxford.

Rosenblatt F., 1962, *Principles of Neurodynamics*, Spartan Books, New York.

Rumelhart D. E., Hinton G. E., Williams R. J., 1986, Learning internal representations by error propagation, In *Parallel Distributed Processing*, chapter 8, pages 318-362, MIT Press, Cambridge, MA.

Sanger T. D., 1990, Learning nonlinear features using eigenvectors of radial basis functions, Abstracts of the *Neural Networks for Computing* conference, Snowbird UT.

Sun G. Z., Lee Y. C., Chen H. H., 1988, A novel net that learns sequential decision process, In Anderson D. Z., ed., *Neural Information Processing Systems*, pages 760-766, American Institute of Physics, New York.

Tenorio M. F., Lee W.-T., 1989, Self organizing neural network for optimum supervised learning, Technical Report TR-EE 89-30, Purdue Univ. School of Elec. Eng.

Weigend A. S., Huberman B. A., Rumelhart D. E., 1990, Predicting the future. A connectionist approach, Technical Report PDP-90-01, Stanford PDP Research Group, submitted to *Int. J. Neural Systems.*

Widrow B., Hoff M. E., 1960, Adaptive switching circuits, In *IRE WESCON Conv. Record, Part 4*, pages 96-104.

Effects of Circuit Parameters on Convergence of Trinary Update Back-Propagation

Randy L. Shimabukuro
Naval Ocean Systems Center
San Diego CA 92152-5000

Patrick A. Shoemaker
Naval Ocean Systems Center
San Diego CA 92152-5000

Clark C. Guest
Univ.Calif.San Diego
San Diego CA 92093

Michael J. Carlin
Naval Ocean Systems Center
San Diego CA 92152-5000

Abstract

This paper discusses an error back-propagation learning rule that limits weight updates to one of three values: a small positive increment, a small negative increment, or zero. This rule is motivated by the relative ease with which it may be implemented in electronic circuitry. The effect on the convergence rate of this rule of likely circuit characteristics such as multiplication nonlinearity and function approximation are investigated. Results show convergence is still possible over a range of parameters even with these expected circuit constraints.

1. INTRODUCTION

There are currently many efforts underway to implement neural network models in analog or hybrid analog/digital integrated circuitry. While some have designed circuitry to take advantage of device physics [1,2] many architectures and algorithms have developed with primary regard given their computational capabilities or their suitability for modeling cognitive or neurobiological processes, rather than their suitability for implementation. Nonetheless, a number of experimental efforts at implementations, in addition to those cited above, have been reported recently [3-9] Some researchers have used fixed interconnection weights between processing units [6], while others have addressed the problem of modifiable weight circuitry [3-5,7,10-12] which is required for a network that is adaptive or programmable after manufacture.

Another focus of current research is on the inductive capabilities of adaptive neural network models. These employ "learning" algorithms, for modification of weight values based upon data presented to the network. Only a few efforts at implementation of adaptive networks with circuitry to carry out the learning process included on-chip [3,4] have been reported. Furman et al. have described an effort to implement the back-propagation learning algorithm [13], using circuitry for weight modification and storage that includes a dynamic memory cell.

We have investigated the use of a floating-gate MOS device as a modifiable and non-volatile mechanism for storage of weight values in analog Artificial Neural Network (ANN) circuitry [12,14]. In such a device, charge stored on an electrically isolated piece of conductor is used to represent a weight value. However, precise control of increments or rate of change of this charge, as would be required by many learning rules, is difficult to achieve with simple circuitry. In addition, if a unique change were to be made to each weight in parallel across an entire network, then complicated control circuitry would have to be replicated at each weight circuit, requiring a large area of silicon.

The difficulties associated with computing and imposing graded weight updates in parallel in analog hardware have led us to investigate simplified parallel learning procedures in which weight changes are very coarsely quantized. Some precedents for this approach are found in the simulation study of Peterson and Hartman [15], which examined the effect of update quantization into two states (increment or decrement) on the performance of a mean field theory learning algorithm, and in the hardware implementation of a stochastic learning network [3] whose hybrid digital/analog weights are also subject to fixed increments or decrements at each step in the learning process. In a previous study, [16] we examined a quantized form of back-propagation where weight updates are limited to one of three values: an increment, a decrement of the same magnitude, or zero. Due to its trinary nature we have dubbed this the Trit learning algorithm.

2. TRIT ALGORITHM

We consider feedforward networks with processing elements partitioned into an input layer of linear, unity-gain elements, and hidden and output layers of nonlinear elements. We adopt the notation of Rumelhart et.al. [13], writing for the output O_i of element i in this network:

$$O_i = \begin{cases} I_i & \text{(unit i an input unit)} \\ F\left(\sum_j W_{ij} O_j + \theta_i\right) & \text{(otherwise)}, \end{cases} \qquad (1)$$

where I_i is an external input to unit i, if it is an input unit, W_{ij} is the weight associated with the interconnection from the jth processing unit in the network to the ith unit, θ_i is a bias or offset term, and F is a sigmoidal activation function for the hidden and output units. We use the form of back-propagation corresponding to stochastic approximation [17,18] in which gradient descent is performed on the sum-square of network output errors for individual input/output associations, rather than in a "batch" or cumulative fashion. Learning constitutes an iterative procedure over a training set of input and desired output vectors or patterns. The components of the gradient of the square error with respect to the network parameters are computed as a product of two factors in the case of the weights:

$$\partial E/\partial W_{ij} = -\delta_i O_j, \qquad (2)$$

while for the biases,

$$\partial E/\partial \theta_i = -\delta_i, \qquad (3)$$

where

$$\delta_i = \begin{cases} F'_i E_i & \text{(unit i an output unit)} \\ F'_i \sum_j W_{ji} \delta_j & \text{(otherwise)}. \end{cases} \qquad (4)$$

Here E is the square-error summed over the network outputs, δ_i is a back-propagated "error" term, E_i is network output error for unit i, if unit i is an output unit, and F'_i is the derivative of the activation function F with respect to its argument for unit i. The δ_i in (4) can be regarded as calculated in parallel by a linear network topologically similar to the forward or signal propagation network, but with the signal flow between units in opposite direction

The update scheme detailed in this paper is used in an analogous iterative procedure and requires the quantities δ_i and O_j as calculated in standard back-propagation. These are used to quantize weight and bias updates according to the rules

$$\Delta W_{ij} = \begin{cases} \eta\, \mathrm{Sgn}(\delta_i O_j) & (|O_j| \geq \varepsilon_1 \text{ and } |\delta_i| \geq \varepsilon_2) \\ 0 & (|O_j| < \varepsilon_1 \text{ and } |\delta_i| < \varepsilon_2) \end{cases} \qquad (5)$$

$$\Delta \theta_i = \begin{cases} \eta\, \mathrm{Sgn}(\delta_i) & (|\delta_i| \geq \varepsilon_2) \\ 0 & (|\delta_i| < \varepsilon_2), \end{cases} \qquad (6)$$

where η, ε_1 and ε_2 are all positive constants. For constant η, the learning process corresponds to motion on a lattice in weight/bias space, which is in a direction of decreasing sum square error for each training pattern pair, although not generally in the direction of steepest descent.

Three different problems were studied using each learning rule in order to compare the convergence properties of the Trit and back-propagation algorithms. The first problem constituted recognition of sixteen orthogonal 32-component input vectors, obtained as samples of sixteen harmonically-related sine waves These were presented as analog values to a network with 32 input units, and the desired mapping was to a one-of-sixteen output code on the network's sixteen output units. In the second problem, the network had fifteen input and fifteen output units, and the training set consisted of 40 pairs of fifteen-bit strings, each derived from a three-by-five pixel representation of an alphabetic character, digit, or punctuation mark. The third problem, chosen for computational difficulty, amounted to finding the parities of a limited set of four-bit binary numbers. Of the sixteen possible four-bit patterns, one with even parity and one with odd were selected at random and withheld from the training set, to allow for future testing of network responses to novel patterns. A network with four input units and one output unit was used in the trials. The target output for these networks was 1 if an odd number of 1's were present in the input data, and -1 if the number were even.

Results of the study, summarized in Tables 1 and 2, show this Trit variant of back-propagation learning is capable of converging on solutions of the benchmark problems. Table 1 compares the iterations to convergence for Trit and backpropagation on three different problems. It converges in considerably fewer iterations than a comparable version of standard back-propagation (with fixed learning rate and without momentum) on all of these problems. Standard back-propagation displays behavior characteristic of gradient descent, in which rapid convergence in initial iterations is followed by slow and inconsistent improvement in performance; however, with the Trit rule the slowdown in convergence appears to be much less pronounced. Evidence suggests that this is at least partly due to the scaling imposed upon the vector of weight and bias updates by the quantization scheme. Table 2 presents the resistance to faults that each network provides. In table 2 the variable n is the number of runs completed for the indicated parameter values.

Table 1: Iterations to Convergence

	Orthogonal Input Vector Problem	Character-Mapping Problem	Limited Parity
Standard Back-Propagation:			
Mean:	16.3	123.8	245.9
St. Dev.:	4.1	12.4	72.3
Trinary Back-Propagation:			
Mean:	5.3	42.6	24.1
St. Dev.:	0.8	1.3	4.0

Table 2: Fraction of Outputs in Error

	Orthogonal Input Vector Problem	Character-Mapping Problem	Limited Parity
	$N_h = 8$	$N_h = 20$	$N_h = 5$
Standard Back-Propagation:			
Mean:	0.18	0.20	0.56
St. Dev..	0.04 (n=10)	0.02 (n=10)	0.08 (n=6)
Trinary Back-Propagation:			
Mean:	0.14	0.18	0.21
St. Dev.:	0.01 (n=10)	0.02 (n=10)	0.10 (n=2)
	$N_h = 32$	$N_h = 55$	$N_h = 15$
Standard Back-Propagation:			
Mean:	0.31	0.29	0.50
St. Dev..	0.03 (n=10)	0.02 (n=10)	0.13 (n=8)
Trinary Back-Propagation:			
Mean:	0.25	0.21	0.19
St. Dev.:	0.02 (n=10)	0.02 (n=10)	0.11 (n=10)
	$N_h = 96$	$N_h = 100$	$N_h = 40$
Standard Back-Propagation:			
Mean:	0.22	0.24	0.51
St. Dev.:	0.03 (n=10)	0.02 (n=10)	0.09 (n=10)
Trinary Back-Propagation:			
Mean:	0.22	0.18	0.19
St. Dev..	0.03 (n=10)	0.02 (n=10)	0.05 (n=10)

Tolerance of networks to variations of the weights from their final learned values was also examined. Normally distributed random numbers, with zero mean and several variances, were added to the weights, and the performances of the resulting networks on their respective training sets were tested. We see from the results of table 2 that the performance of the trinary networks compares favorably with that of the standard back-propagation networks.

3. EFFECTS OF CIRCUIT LIMITATIONS

The trinary rule compares quite well with back-propagation, giving a substantial reduction in required circuit complexity. There are however, several other factors which must be overcome in order to implement fully parallel learning in analog VLSI. We have been attempting to identify the most important issues by simulating circuit imperfections and evaluating their effect on learning. We present here preliminary results on two such effects; nonlinearities in multiplications and approximations to F', the derivative of the sigmoidal transfer function.

Networks with fifteen input units and forty output units, fully interconnected by a hidden layer which varied in size from 40 to 90 units, were used to judge these effects. The input pattern was identical to that used in the character mapping problem of the first study but in this study the desired output was a 1 of n vector code. Both inputs and desired outputs were binary, with values of +1 and -1. Networks that failed to converge within 1000 iterations were judged nonconvergent. The trinary rule can fail to converge because all back-propagated delta terms over a training set become smaller than an update threshold ε_2. This problem may be circumvented by simply reducing the ε_2 parameter. An adaptive feature was added to the trinary rule in that the values of ε_1 and ε_2 were reduced by 20% after any iteration (presentation of the entire pattern set) in which no update occurs. In all the trinary runs the initial values of ε_1 and ε_2 were set at .33 and .02 respectively. Each test of a set of parameters was repeated three times with a different set of initial random weights. The initial weights were uniformly distributed between ±0.3.

3.1 NONLINEARITIES IN MULTIPLICATION

Much of the attraction of neural networks is their inherent fault tolerance. This has rekindled an interest in analog computing with the hope that such fault tolerance could make up for the inaccuracies of analog computing elements. There have been studies which have looked at the effect of limiting the resolution of the weight values [19]. These studies are more applicable to digital or hybrid (analog inputs and outputs with digital weights) implementations since these would lead to round off errors rather than more systematic errors which may occur in a purely analog implementation.

In any multiplier circuit, one must trade off simplicity and size for accuracy to some extent. The multiplier proposed by Shimabukuro [14] can be realized with only two transistors at the synapse level. First order analysis of the multiplier shows nonlinearities of the form.

$$x \cdot w \rightarrow x \cdot w \cdot d_1 + d_2 \cdot x^2 \qquad (8)$$

where x is an input, w is a weight, and d_1 and d_2 are functions of the ratio of the transconductances of the two MOSFETs. In the ideal case, (perfectly matched transistors) $d_1 = 1$, and $d_2 = 0$. For a reasonable estimate of this effect, we looked at a mismatch in transconductances of plus and minus 10% which corresponds to a d_1, d_2 of (1.05, 0.075) and (.95, -0.075) respectively. This effect was included in the multiplication of back-propagated error terms as well as in the feed forward path.

3.2 APPROXIMATIONS TO F'

Another practical concern of any gradient type learning rule is the evaluation of F'. In one approach to this problem, suggested by Andes et al. [20] in their Madeline Rule III, F' is accounted for implicitly by perturbing the inputs to the neurons and looking at the corresponding change in the output error. While this does account for F', it does not lend itself well to a parallel implementation.

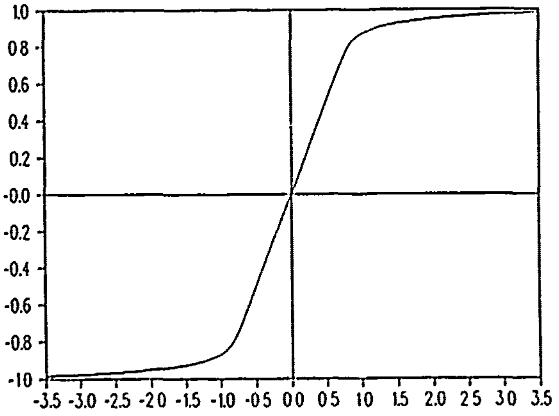


Figure 1. Activation Function F.

We have looked at a somewhat cruder approach, merely approximating F' by a simple function given in Eqn. 10. The activation function used in our simulations is shown in Figure 1. The function itself cannot be written in closed form but its inverse F^{-1} takes the form

$$F^{-1}(x) = C_1 x + C_2 \sinh(C_3 x). \qquad (9)$$

The form of this function is that of the input/output relationship for an operational amplifier in a current summing mode, with resistor and diodes in its feedback path. The derivative of this function is plotted by the solid line in Figure 2. We approximated this with the function V(F(x)) shown by the dotted line in Figure 2 and given by

$$V(\,F(x)\,) = \begin{cases} 1 & \text{for } |F(x)| > .85 \\ a & \text{for } |F(x)| < .85 \end{cases} \qquad (10)$$

where a is constant bias term which was varied between training runs. A similar function was used by Samad [21]. Rather than the constant bias a, he used a function which dropped off linearly in the outer regions.

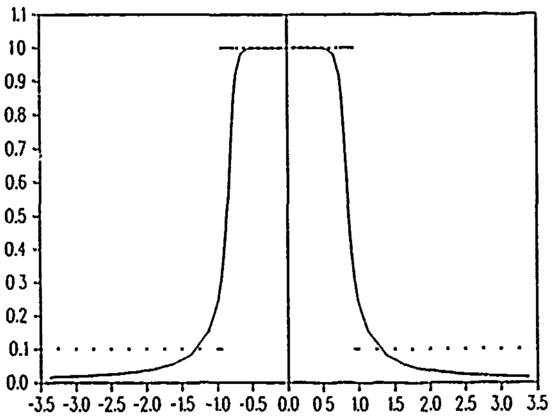


Figure 2 - Derivative of the Activation Function of Eqn. 9 and the Rect function of Eqn. 10.

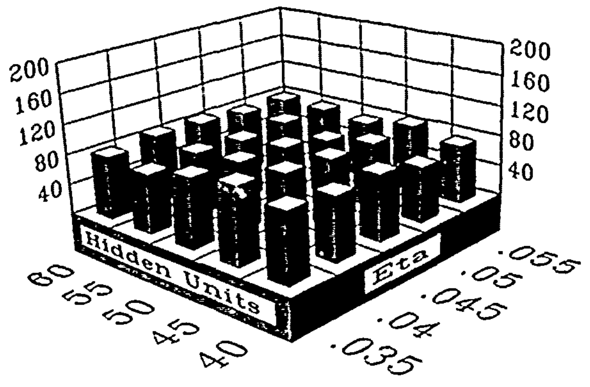


Figure 3 - Convergence Times for Back-Propagation. Iterations to Convergence Are Plotted Against Learning Rate and Number of Hidden Units.

4. RESULTS

To establish a baseline, the problem was first run using standard back-propagation learning. Figure 3 shows the convergence times for a range of η and different sized hidden layers. We see that it is fairly insensitive to these parameters over this range with a minimum of 72 ± 5.6 iterations. Figure 4 shows the results of running the same initial weight sets using the trinary algorithm. Again there is little sensitivity to the varied parameters. As observed in the previous study [16], the convergence times were significantly lower than those for back-propagation. In this case the minimum was 27 ± 1 iterations.

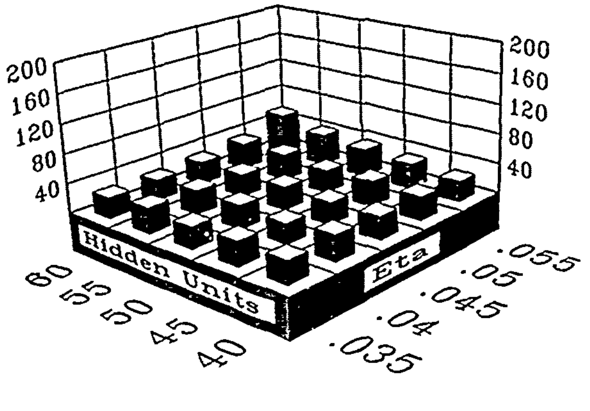


Figure 4 - Iterations to Convergence for Trit Algorithm.

4.1 NONLINEAR MULTIPLICATIONS

Nonlinearities of the form described in eq. 8 were introduced into the Trit learning algorithm. Figure 5a and 5b show convergence times for a second order term d_2 of -0.075 and +0.075 respectively. As expected, convergence times were significantly longer than in the linear case. Figure 5a has a minimum of 41 ± 5.3 iterations and figure 5b has a minimum of 96 ± 10 iterations. This data seems to indicate that positive values of d_2 have more of an effect on convergence, although it appears that the global minimum for figure 5b is probably at lower values of η and larger hidden layers. The asymmetry in the effects of positive and negative second order terms may be due to a bias in the number of input and desired output components of one sign versus the other.

4.2 F' APPROXIMATIONS

When the F' function is replaced with the Rect function estimate in networks with linear multipliers, there is no significant effect. Figure 6 shows results for runs with η fixed at 0.04 while varying the F' bias and hidden layer size. We see a similar flat response within the same range of hidden units as figure 4, with a minimum of 24.3 ± 2.3 iterations.

The two approximations combined have a much more dramatic effect. For runs with both nonlinear multiplications and F approximations a larger range of hidden layer sizes was needed to get an appreciable number of convergent runs. Once again when the second order term is positive, convergence times increase substantially. It is possible again that larger hidden layers or smaller values of η may not have such a big discrepancy. Figure 7a shows similar performance to figure 5a (at eta of 0.04) with a minimum of 55 ± 2. Figure 7b has a minimum value of 224 ± 43 iterations. In both figures runs with less than two out of three convergent runs were omitted.

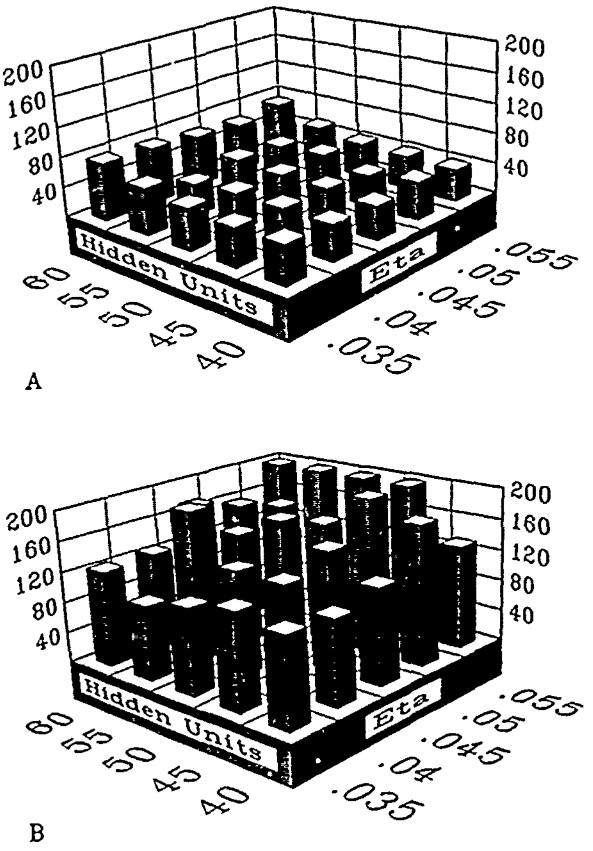


Figure 5 - a)Trit Convergence with a Second Order Term of -0.075. b)Trit Convergence with a Second Order Term of +0.075.

5. CONCLUSIONS

There have oeen many anecdotal statements to the effect that the fault tolerant nature of neural networks will allow for imprecise analog electronic components. This study is a first attempt at quantifying these claims. We have looked at two approximations which must be made in implementing the Trit learning algorithm; nonlinearities in the multiplication terms and a quantized form of the F' function. Both of these will affect the back-propagated error terms.

Within the range of the parameters in this study, it is apparent that convergence times are more sensitive to the nonlinearities in the multiplications. It should be pointed out that these results may not apply to different types of multiplier circuits. In other cases, the form of the resulting nonlinearities may be quite different.

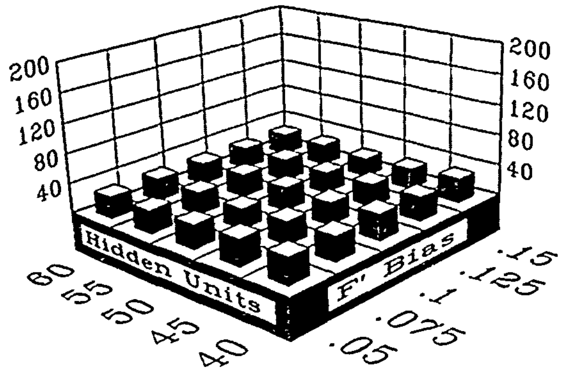


Figure 6 - Convergence with F' Approximation.

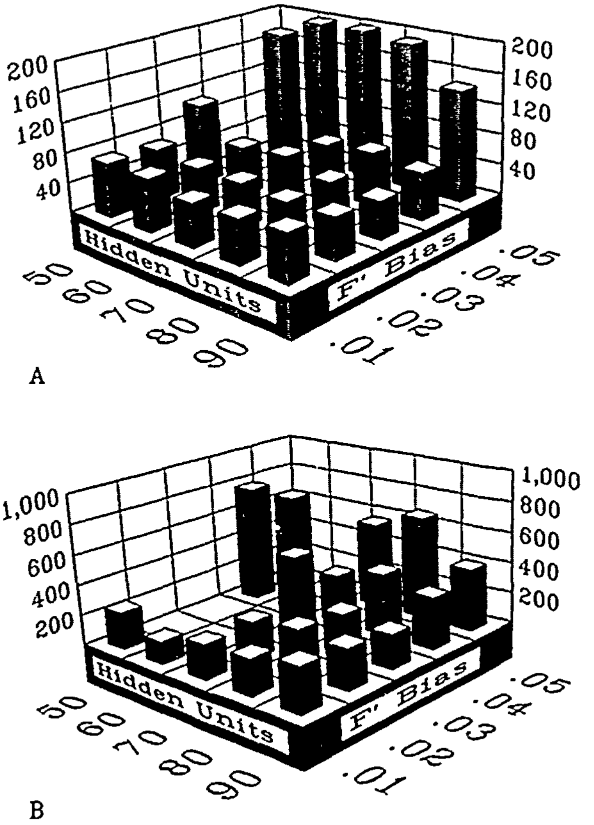


Figure 7 -a)Convergence with F' Approximation and a -0.075 Second Order Term. b)Convergence with F' Approximation and a +0.075 Second Order Term.

We feel that these results are significant in that they show convergence over a range of parameters in spite of expected circuit limitations. The issue of how these approximations affect the learning capacity of the networks remains an open question.

We believe that studies such as these are crucial to the development of learning analog neural network circuits. Such analysis can lead one to identify the more crucial components. Based on our results, we have been led to incorporate additional circuitry to improve on the linearity of our multipliers. We are currently working on similar studies of other effects.

References

1. Mead, C.A. (1989) Analog VLSI and Neural Systems. Reading, MA: Addison-Wesley.

2. Mead, C.A., & Mahowald, M.A. (1988) A silicon model of early visual processing. Neural Networks, 1, 91-97.

3. Alspector, J., Allen, R.B., Hu, V., Satyanarayana, S. (1988) Stochastic learning networks and their implementation. Proceedings, IEEE Conference on Neural Information Processing Systems -- Natural and Synthetic, Denver, 9-21. New York: American Institute of Physics.

4. Furman, B., & Abidi, A. (1988) CMOS analog IC Implementing the back propagation algorithm. First Annual Meeting, INNS, Boston. (Abstract) Neural Networks, 1, Sup. 1, 38.

5. Holler, M., Tam, S., Castro, H., & Benson, R. (1989) An electrically trainable artificial neural network (ETANN) with 10240 "floating gate" synapses. Proceedings, International Joint Conference on Neural Networks, Washington, 2, 177-182. New York: Institute of Electrical and Electronics Engineers.

6. Jackel, L.D., Graf, H.P., & Howard, R.E. (1987) Electronic neural network chips. Applied Optics, 26, 5077-5080.

7. Mueller. P., Van der Spiegel, J., Blackman, D., Chiu, T., Clare, T., Dao, J., Donham, C., Hsieh, T., & Loinaz, M. (1989) A general purpose analog neurocomputer. Proceedings, International Joint Conference on Neural Networks, Washington, 2, 191-196. New York: Institute of Electrical and Elec'ronics Engineers.

8. Murray, A.F., & Smith, A. (1988) Asynchronous VLSI neural networks using pulse-stream arithmetic. IEEE Journal of Solid-State Circuits, 23, 688-697.

9. Schwartz, D.B., Howard, R.E., & Hubbard, W.E. (1989) A programmable analog neural network chip. IEEE Journal of Solid-State Circuits, 24, 313-319.

10. Hu, V., Kramer, A., & Ko, P.K. (1988) EEPROMs as analog storage devices for neural nets. First Annual Meeting, INNS, Boston. Neural Networks, 1, Sup. 1, 385.

11. Moopenn, A., Thakoor, A.P., Duong, T., & Khanna, S.K. (1987) A neurocomputer based on an analog-digital hybrid architecture. In M. Caudill & C. Butler (Eds.), Proceedings, IEEE First International Conference on Neural Networks, San Diego, 3, 479-486. New York: Institute of Electrical and Electronics Engineers.

12. Shoemaker, P.A., & Shimabukuro, R. (1988) A modifiable weight circuit for use in adaptive neuromorphic networks. First Annual Meeting, INNS, Boston. (Abstract) Neural Networks, 1, Sup. 1, 409.

13. Rumelhart, D.E., Hinton, G.E., and Williams, R.J. (1986) Learning internal representations by error propagation. In D.E. Rumelhart & J.L. McClelland, (Eds.), Parallel Distributed Processing, Explorations in the Microstructure of Cognition, 1, 318-362. Cambridge: MIT Press.

14. Shimabukuro, R.L., Shoemaker, P.A., & Stewart, M. (1989) Circuitry for artificial neural networks with non-volatile analog memories. Proceedings, IEEE International Symposium on Circuits and Systems, Portland, 2, 1217-1220. New York: Institute of Electrical and Electronics Engineers.

15. Peterson, C., & Hartman, E. (1989) Explorations of the mean field theory learning algorithm. Neural Networks, 2, 475-494.

16. Shoemaker, P.A., Carlin, M.J., & Shimabukuro, R.L. (1990) Back-propagation learning with trinary quantization of weight updates. Neural Networks, to be published.

17. White, H. (1989) Neural-network learning and statistics. AI Expert, December, 48-52.

18. Robbins, H., & Munro, S. (1951) A stochastic approximation method. Annals of Mathematical Statistics, 2 2, 400-407.

19. Caviglia, D.D., Valle, M., & Bisio, G.M. (1990) Effects of weight discretization on the back propagation learning method: Algorithm Design and Hardware Realization. Proceedings, International Joint Conference on Neural Networks, San Diego, 2, 631-637.

20. Andes, D., Widrow, B., Lehr, M., & Wan, E. (1990) MRIII: A robust algorithm for training analog neural networks. Proceedings, International Joint Conference on Neural Networks, San Diego, 1, 533-536.

21. Samad, T. (1990) Backpropagation improvements based on heuristic arguments. Proceedings, International Joint Conference on Neural Networks, Washington, 1, 565-568. New York: Institute of Electrical and Electronics Engineers.

Equivalence Proofs for Multi-Layer Perceptron Classifiers and the Bayesian Discriminant Function

John B. Hampshire II
Dept. of Electrical and Computer Engineering
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Barak Pearlmutter*
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

This paper presents a number of proofs that equate the outputs of a Multi-Layer Perceptron (MLP) classifier and the optimal Bayesian discriminant function for asymptotically large sets of statistically independent training samples. Two broad classes of objective functions are shown to yield Bayesian discriminant performance. The first class are "reasonable error measures," which achieve Bayesian discriminant performance by engendering classifier outputs that asymptotically equate to *a posteriori* probabilities. This class includes the mean-squared error (MSE) objective function as well as a number of information theoretic objective functions. The second class are classification figures of merit (CFM_{mono}), which yield a qualified approximation to Bayesian discriminant performance by engendering classifier outputs that asymptotically identify the maximum *a posteriori* probability for a given input. Conditions and relationships for Bayesian discriminant functional equivalence are given for both classes of objective functions. Differences between the two classes are then discussed very briefly in the context of how they might affect MLP classifier generalization, given relatively small training sets.

1 INTRODUCTION

The use of multi-layer perceptron (MLP) classifiers in statistical pattern recognition requires that there be some mathematically defensible link between MLP outputs and the true *a posteriori* probabilities associated with the input random vector (RV) x being classified. We present a number of proofs that detail the link for an N-output MLP classifier and the N-class RV x, possessing an input feature space dimensionality of M. The number of classes N and the feature space dimensionality M of x are arbitrary, as is the *specific* parameterization (or connectivity) of the MLP classifier. For our purposes the term "multi-layer perceptron" is used to describe a backpropagation network using any continuous sigmoidal nonlinearity, although the proofs herein can be extended to networks employing other non-linearities.

Proofs of the relationship between both linear and nonlinear classifiers trained with the mean-squared-error (MSE) objective function and the Bayesian discriminant function are not new. Duda and Hart formulated the proof for a simple perceptron in [6] (pp. 154-155). More recently, [1, 3, 7, 11] have given variations of the proof for MSE-trained MLPs. We extend these proofs to the N-output MLP classifier trained with *any* objective function belonging to one of two broad classes. The proofs herein give detailed relationships among the MLP outputs, the Bayesian discriminant function, and the class conditional densities of x. In this sense, they have their conceptual basis in the proof of [6].

We show that the MSE proofs of [1, 3, 6, 7, 11] pertain to one specific member of a broad class of error measure objective functions. This class of "reasonable" error measures yields MLP outputs that converge to the Bayesian *a posteriori* probabilities $P(\omega_i \mid x)$ (where ω_i represents the ith class) for networks with sufficient functional capacity (see section 3.1.2) to classify asymptotically large sets of statistically independent training samples. The MSE and Cross Entropy (CE) [10] objective functions are members of this class of functions[1], as are other objective functions stemming from information theoretic learning rules (such as Maximum Mutual Information and Maximum Likelihood), and the Kullback-Liebler distance measure. These reasonable error measures all yield optimal Bayesian discriminant performance[2], given sufficient training data.

Given these results, one is inclined to conclude that all these objective functions yield equivalent classification perfor-

*Hertz Fellow

[1] Strictly speaking, the Cross Entropy objective function does not require that MLP outputs be compared with *binary* target values. Thus, it is fair to categorize the Cross Entropy objective function in this way *only* when binary target values are specified in its form.

[2] See section 2.

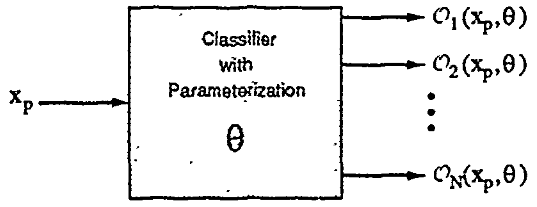


Figure 1: The general N-class classification problem.

mance, and that all MLPs are — in effect — no more than exotic estimators of Bayesian *a posteriori* probabilities. In fact, neither conclusion is correct. A broad class of objective functions called "N-monotonic Classification Figures of Merit" (CFM_{mono}) [8] are shown to approximate Bayesian classification performance under the same conditions for which the reasonable error measures yield Bayesian performance. However the CFM_{mono} class of functions *does not* produce MLP output activations that reflect *a posteriori* probabilities $P(\omega_i\,|\,\mathbf{x})$; instead it asymptotically identifies the maximum *a posteriori* probability for a given input $P(\omega_{max}|\mathbf{x}_p)$, as long as $P(\omega_{max}|\mathbf{x}_p) > 0.5$ (see section 4). Despite this limitation, [8] indicates that CFM_{mono}-trained MLPs can be more robust approximations to the Bayesian discriminant than their reasonable error measure counterparts, given small training sample sizes.

While the findings of [8] are not broad enough to be considered conclusive, they do argue against the maxim "all objective functions yield equivalent classification performance," when one's training set is limited in size. Section 5 contains some brief comments regarding the following proofs' applicability to real-world classification problems. Particular attention is paid to how the different objective functions might yield (or fail to yield) near-optimal classification boundaries for small training sets. These observations are made with an eye towards further investigation of MLP classifier generalization in the probabilistic context presented by this paper.

By the "asymptotic behavior" of a classifier we mean its behavior for an asymptotically large set of statistically independent training samples.

2 A GENERAL DESCRIPTION OF THE N-CLASS PROBLEM AND THE BAYESIAN DISCRIMINANT FUNCTION

In this section we give a brief description of the general N-class problem and the Bayesian discriminant function in the context of the connectionist and pattern recognition literature. The syntax and notation used herein is an expanded version of that used in [6].

The N-class classification problem is depicted in Figure 1. A random vector $\mathbf{x}$ is to be classified by a classifier with parameterization specified by the state variable θ. The classifier has N outputs, each one of which corresponds to one of N possible classes. Table 1 defines the variables used to describe the basic classification process. Simply stated, the objective is to associate a particular sample of the RV $\mathbf{x}$ — denoted $\mathbf{x}_p$ — with the correct class ω_c. The method for deciding the class of $\mathbf{x}_p$ yielding the fewest errors [6] (pp. 16-20) can be stated simply:

> *associate* $\mathbf{x}_p$ *with the class* ω_c *that has the largest a posteriori probability:*

$$P(\omega_c|\mathbf{x}_p) = P(\omega_{max}|\mathbf{x}_p) > P(\omega_j|\mathbf{x}_p) \quad \forall j \neq c$$

In simple terms, any function that implements this classification procedure constitutes the Bayesian discriminant function.

Clearly, being able to estimate all N $P(\omega_i\,|\,\mathbf{x}_p)$ accurately for each and every $\mathbf{x}_p$ allows one to implement the Bayesian discriminant function. Indeed, a large number of pattern classifiers do precisely this. The degree to which they succeed in the classification task is directly related to the accuracy with which they estimate the *a posterioris*. Another perhaps less obvious way to implement the Bayesian discriminant function is to consistently identify the largest $P(\omega_i\,|\,\mathbf{x}_p)$ for each and every $\mathbf{x}_p$ — an approach that does not require accurate estimation of the *a posterioris*. The salient point here is that while accurate estimation of the *a posterioris* is *sufficient* for Bayesian discriminant performance, it is *not necessary*. All that is necessary for Bayesian discrimination is accurate identification of the largest *a posteriori*.

These two approaches to implementing the Bayesian discriminant function lead to two broad classes of objective functions that one can use to train the classifier in Figure 1: the class of "reasonable error measures" achieves Bayesian performance by explicitly estimating the *a posterioris* associated with the input $\mathbf{x}_p$, while the Classification Figures of Merit (CFM_{mono}) achieve Bayesian performance by estimating the identity of the maximum *a posteriori* probability $P(\omega_{max}|\,\mathbf{x}_p)$.

3 REASONABLE ERROR MEASURES: BAYESIAN PERFORMANCE VIA ACCURATE ESTIMATION OF *A POSTERIORI* PROBABILITIES

The first class of objective functions that yield Bayesian discriminant performance comprises those error measures engendering classifier outputs that are true estimates of the *a posteriori* probabilities $P(\omega_i\,|\,\mathbf{x}_p)$. The necessary and sufficient conditions on the form of these functions are given below, followed by a number of familiar examples of the class and detailed proofs of their asymptotic Bayesian performance.

Table 1: Definitions of symbols used to describe the general N-class classification problem.

Symbol	Definition
x	The RV to be classified.
$\mathcal{O}_i$	The ith output of the N-output classifier.
ω_i	The ith of N classes to which x can belong.
x_p	The pth unique sample (or *prototype*) of x.
θ	The parameterization of the classifier. In the case of an MLP classifier, θ would represent the connections of the network.
$\mathcal{O}_i(\mathrm{x}_p, \theta)$	The ith output of the N-output classifier, given the input x_p and the classifier parameterization θ.
$\mathrm{P}(\omega_i \mid \mathrm{x}_p)$	The *a posteriori* probability of the ith class (ω_i), given the input x_p.
$\mathrm{P}(\overline{\omega_i} \mid \mathrm{x}_p)$	$1 - \mathrm{P}(\omega_i \mid \mathrm{x}_p)$.
$\rho(\mathrm{x} \mid \omega_i)$	The "class conditional" probability density function (PDF) for the RV x (given class ω_i).

3.1 THE NECESSARY CONDITIONS FOR REASONABLE ERROR MEASURES

Consider a class of error measures $\mathcal{E}[\mathcal{O}_i(\mathrm{x}_p, \theta), \mathcal{D}_i(\mathrm{x}_p)]$ that give the "loss" of a single output $\mathcal{O}_i(\mathrm{x}_p, \theta)$ when its desired or "target" activation is $\mathcal{D}_i(\mathrm{x}_p)$. Tables 1 and 2 define the symbols used to derive this class of error measures. The concept of a *prototype* of x introduced in these tabulated definitions warrants explanation.

3.1.1 Prototypes: bounds on the complexity of the class-conditional densities of the RV x

A prototype is a *unique* sample x_p of the RV x. Thus, if one obtains two identical yet statistically independent samples of the RV x, these samples are two instantiations of the same prototype. The notion of obtaining more than one *statistically independent* sample of x with the *exact* same value x_p is difficult to envision — even for large training sets. However, if one considers regions on the domain of x over which the class-conditional densities $\rho(\mathrm{x} \mid \omega_i)$ are essentially constant for all classes, one can associate each of these regions with a *prototypical* value of x. The prototype for the pth of such regions is given by x_p. For an input feature space of dimensionality M and a sufficiently large number of statistically independent samples of x, one might envision an (M+1)-dimensional histogram of the samples as an embodiment of this concept of prototypes. Such a view is consistent with the limited resolution of data acquisition systems used to measure real-world RVs.

Clearly this view of regions on x with constant class-conditional densities places an implicit restriction on the probabilistic nature of x. A simple yet elegant description of a 2-class problem (N = 2) involving a 2-dimensional RV x (M = 2) that does *not* have a bounded number of regions of constant class-conditional density is illustrated by the following: if one envisions a two dimensional fractal coastline forming the boundary between land and sea, one finds that in the vicinity of the boundary (shore line) there is no observation scale large enough to yield a bounded number P of regions x_p within which $\rho(\mathrm{x} \mid \omega_i)$ is constant on all sub-regions of each x_p for both classes. The RV x is therefore not "well behaved". Obviously, if x comprises a finite number of discrete states, then it will be well behaved.

The necessary conditions for reasonable error measures that follow — and all subsequent proofs in this paper — rely on this notion of prototypes. We assume that the RV x is well-behaved to the extent that P is bounded. This restriction places some limit on the complexity of the class-conditional densities of x that one can expect to model accurately using an MLP classifier — an issue that we discuss further in section 5.

3.1.2 The reasonable condition

In general, we assume that the outputs of the classifier are bounded on the closed interval [0,1], that there is minimal loss incurred when the output equals its target value, and that there is a symmetry to the loss function:

$$0 \leq \mathcal{O}_i(\mathrm{x}_p, \theta) \leq 1 \qquad \forall \mathrm{x}_p, \theta \tag{1}$$

$$\mathcal{E}[z, z] \;<\; \mathcal{E}[y \neq z, z] \tag{2}$$

$$\begin{aligned} &\mathcal{E}[\mathcal{O}_i(\mathrm{x}_p, \theta), \mathcal{D}_i(\mathrm{x}_p)] \\ &\quad = \mathcal{E}[\mathcal{D}_i(\mathrm{x}_p) - \mathcal{O}_i(\mathrm{x}_p, \theta), \overline{\mathcal{D}_i}(\mathrm{x}_p)] \end{aligned} \tag{3}$$

The symmetry constraint of (3) can be taken to mean that the reasonable error measure is a function of the absolute difference between the output and its target:

$$\mathcal{E}[\mathcal{O}_i(\mathrm{x}_p, \theta), \{\mathcal{D}\}] = f(|\mathcal{O}_i(\mathrm{x}_p, \theta) - \{\mathcal{D}\}|) \tag{4}$$

where

$$\{\mathcal{D}\} = \mathcal{D}_i(\mathrm{x}_p) \;\text{ or }\; \overline{\mathcal{D}_i}(\mathrm{x}_p)$$

Table 2: Definitions of symbols used to derive reasonable error measures.

Symbol	Definition
i	index denoting MLP output i of N, associated with class ω_i.
N	the total number of classes.
p	index denoting the pth prototype of x.
P	the total number of prototypes on the domain of x.
n_{pi}	the number of statistically independent occurrences of prototype x_p belonging to class ω_i.
$n_{p\bar{i}}$	the number of statistically independent occurrences of prototype x_p *not* belonging to class ω_i.
n_p	$n_{pi} + n_{p\bar{i}}$: the total number of *statistically independent* occurrences of prototype x_p in the training set.
n_i	$\sum_p n_{pi}$: the total number of statistically independent samples in the training set belonging to class ω_i.
$n_{\bar{i}}$	$\sum_p n_{p\bar{i}}$: the total number of statistically independent samples in the training set *not* belonging to class ω_i.
n_t	$\sum_p n_p$: the total number of statistically independent samples in the training set.
$\mathcal{D}_i(x_p)$	The target value for $O_i(x_p, \theta)$ when x_p belongs to class ω_i.
$\overline{\mathcal{D}_i}(x_p)$	The target value for $O_i(x_p, \theta)$ when x_p does *not* belong to class ω_i.
$\mathcal{E}[O_i(x_p, \theta), \mathcal{D}_i(x_p)]$	The error measure (or loss) for output $O_i(x_p, \theta)$ when its target value is $\mathcal{D}_i(x_p)$ (i.e., when x_p belongs to class ω_i).
$\mathcal{E}[O_i(x_p, \theta), \overline{\mathcal{D}_i}(x_p)]$	The error measure (or loss) for output $O_i(x_p, \theta)$ when its target value is $\overline{\mathcal{D}_i}(x_p)$ (i.e., when x_p does *not* belong to class ω_i).

Furthermore, if we choose binary targets for our error measure (which incidentally correspond to the upper and lower bounds on the classifier outputs)

$$\begin{aligned} \mathcal{D}_i(x_p) &\triangleq 1 \\ \overline{\mathcal{D}_i}(x_p) &\triangleq 0 \end{aligned} \qquad (5)$$

then (4) leads to the following functional description of the reasonable error measure:

$$\begin{aligned} \mathcal{E}[O_i(x_p, \theta), \mathcal{D}_i(x_p)] &\equiv f(1 - O_i(x_p, \theta)) \\ \mathcal{E}[O_i(x_p, \theta), \overline{\mathcal{D}_i}(x_p)] &\equiv f(O_i(x_p, \theta)) \end{aligned} \qquad (6)$$

Using the definitions in tables 1 and 2 with (6), we can express the average error produced by n_t samples of x. Note that these n_t samples are grouped into P prototypes; there are n_p samples of the pth prototype x_p:

$$\begin{aligned} \overline{\mathcal{E}} = \frac{1}{n_t} \sum_{p=1}^{P} \sum_{i=1}^{N} \Big\{ & n_{pi} \cdot f(1 - O_i(x_p, \theta)) \\ & + n_{p\bar{i}} \cdot f(O_i(x_p, \theta)) \Big\} \end{aligned} \qquad (7)$$

Equation (7) can be restated as

$$\begin{aligned} \overline{\mathcal{E}} = \sum_{i=1}^{N} \sum_{p=1}^{P} \frac{n_p}{n_t} \Big\{ & \frac{n_{pi}}{n_p} \cdot f(1 - O_i(x_p, \theta)) \\ & + \frac{n_{p\bar{i}}}{n_p} \cdot f(O_i(x_p, \theta)) \Big\} \end{aligned} \qquad (8)$$

The law of large numbers leads to the following asymptotic form for the average error:

$$\begin{aligned} \lim_{n_t \to \infty} \overline{\mathcal{E}} = \sum_{i=1}^{N} \sum_{p=1}^{P} \mathrm{P}(x_p) \{ & \mathrm{P}(\omega_i \mid x_p) \cdot f(1 - O_i(x_p, \theta)) \\ & + \mathrm{P}(\overline{\omega}_i \mid x_p) \cdot f(O_i(x_p, \theta)) \} \end{aligned} \qquad (9)$$

A necessary and sufficient condition for minimizing $\overline{\mathcal{E}}$ in (9) is $|\nabla_O \overline{\mathcal{E}}| = 0$, which requires that

$$\begin{aligned} \frac{d}{dO_i(x, \theta)} \overline{\mathcal{E}} &= \sum_{p=1}^{P} \mathrm{P}(x_p) \{ -\mathrm{P}(\omega_i \mid x_p) \cdot f'(1 - O_i(x_p, \theta)) \\ &\qquad + \mathrm{P}(\overline{\omega}_i \mid x_p) \cdot f'(O_i(x_p, \theta)) \} \\ &= 0 \quad \forall i \end{aligned} \qquad (10)$$

where

$$f'(u) \triangleq \frac{d}{du} f(u)$$

Equation (10), in turn, is satisfied if

$$\frac{f'(\mathcal{O}_i(x_p, \theta))}{f'(1 - \mathcal{O}_i(x_p, \theta))} = \frac{P(\omega_i \mid x_p)}{P(\overline{\omega_i} \mid x_p)} = \frac{P(\omega_i \mid x_p)}{1 - P(\omega_i \mid x_p)} \quad \forall x_p \quad (11)$$

Note that (11) is both a necessary and sufficient condition for satisfying (10) for *all* possible distributions of x_p (which are directly related to the class-conditional densities of x (see section 3.1.1)). While it is possible to satisfy (10) without satisfying (11) for *some* distributions of x_p (e.g., some sets of class-conditional densities $\{\rho(x \mid \omega_i)\}$), (11) must hold for (10) to hold independent of $\{\rho(x \mid \omega_i)\}$. As a trivial example, if $P(x_p)$ were zero for all but one prototype, satisfying (10) would require satisfying (11).

Clearly, the Hessian of the average error ($H_{\mathcal{O}}\,\overline{\mathcal{E}}$) must be positive definite in order for (11) to yield a minimum average error:

$$|H_{\mathcal{O}}\,\overline{\mathcal{E}}| > 0 \quad (12)$$

One can show that if $\overline{\mathcal{E}}$ in (4) is a strictly increasing function of $|\mathcal{O}_i(x_p, \theta) - \{\mathcal{D}\}|$, (12) will hold.

Equations (11) and (12) ensure a minimum of $\overline{\mathcal{E}}$ but they place no explicit condition on the form of $\mathcal{O}_i(x_p, \theta)$. Since we wish the outputs of the classifier to equal the *a posteriori* probabilities, we can assure this equivalence by constraining the reasonable error measure's functional form based on (11):

$$f'(\mathcal{O}_i) = \frac{\mathcal{O}_i}{1 - \mathcal{O}_i} f'(1 - \mathcal{O}_i) \quad 0 \leq \mathcal{O}_i < 1 \quad (13)$$

Any function satisfying the conditions of (3) – (6) and (10) – (13) is a reasonable error measure. Such a measure will yield classifier outputs that asymptotically equate to the *a posteriori* probabilities $P(\omega_i \mid x)$, provided the functional capacity of the classifier (i.e., the classifier's ability to model the function that maps the RV x to the *a posterioris* $P(\omega_i \mid x)$ for all x_p), embodied in the parameterization variable θ, is at least as great as the complexity of all the class-conditional densities $\rho(x \mid \omega_i)$. This statement relies on the assumption that these class conditional densities are restricted to those that are well behaved (see section 3.1.1). This, combined with the finding that a MLP with a single hidden layer of adequate connectivity can — under mild constraints consistent with our assumptions — approximate any continuous function mapping x onto the N-dimensional hypercube [4], assures that there exists a MLP that will accurately model the Bayesian discriminant functions of any well-behaved x, given a sufficiently large set of statistically independent training samples.

Finally, one can show that any positively scaled reasonable error measures is, itself, a reasonable error measure. That is, if $f_1(\mathcal{O})$ is a reasonable error measure, then $af_1(\mathcal{O})$ will also be reasonable if $a > 0$.

3.2 THE GENERAL REASONABLE ERROR MEASURE APPROXIMATION TO THE BAYESIAN DISCRIMINANT FUNCTION

If one defines the Bayesian discriminant function for the *i*th of *N* possible classes as

$$g_i(x) \triangleq P(\omega_i \mid x) \quad (14)$$

where

$$P(x) = \sum_{j=1}^{N} P(x \mid \omega_j) \quad (15)$$

one can define the *reasonable approximation error* for the *i*th discriminant function as

$$\epsilon_i \triangleq \int_{\mathbf{X}} [f(1 - \mathcal{O}_i(x, \theta)) \cdot g_i(x) + f(\mathcal{O}_i(x, \theta)) \cdot (1 - g_i(x))]\, \rho(x)\, dx \quad (16)$$

Additionally, one can define the *aggregate* reasonable approximation error as

$$\epsilon = \sum_{i=1}^{N} \epsilon_i \quad (17)$$

Given (9), one can express the asymptotic average reasonable error of the training set. One can in turn express the asymptotic average reasonable error in terms of the aggregate reasonable approximation error to the Bayesian discriminant function expressed in (16) and (17). Duda and Hart first showed such a relationship for the simple perceptron trained with the MSE objective function in [6] (pp. 154-155). The symbol "≍" should be read as, "asymptotically equals."

$$\begin{aligned} \lim_{n_t \to \infty} \overline{\mathcal{E}} &= \sum_{p=1}^{P} P(x_p) \sum_{i=1}^{N} \{P(\omega_i \mid x_p) \cdot f(1 - \mathcal{O}_i(x_p, \theta)) \\ &\quad + P(\overline{\omega_i} \mid x_p) \cdot f(\mathcal{O}_i(x_p, \theta))\} \\ &= \sum_{p=1}^{P} \sum_{i=1}^{N} \Big\{P(\omega_i, x_p) \cdot f(1 - \mathcal{O}_i(x_p, \theta)) \\ &\quad + P(\overline{\omega_i}, x_p) \cdot f(\mathcal{O}_i(x_p, \theta))\Big\} \end{aligned}$$

$$\begin{aligned}
&\asymp \sum_{i=1}^{N} \Big\{ \mathrm{P}(\omega_i)\,\mathrm{E}\left[f\left(1 - \mathcal{O}_i(\mathrm{x}, \theta)\right) \mid \omega_i\right] \\
&\quad + \mathrm{P}(\overline{\omega_i})\,\mathrm{E}\left[f\left(\mathcal{O}_i(\mathrm{x}, \theta)\right) \mid \overline{\omega_i}\right] \Big\} \\
&= \sum_{i=1}^{N} \Big\{ \int_{\mathbf{X}} f\left(1 - \mathcal{O}_i(\mathrm{x}, \theta)\right) \cdot \rho(\mathrm{x} \mid \omega_i) \\
&\quad \cdot \mathrm{P}(\omega_i)\, d\mathrm{x} \\
&\quad + \int_{\mathbf{X}} f\left(\mathcal{O}_i(\mathrm{x}, \theta)\right) \cdot \rho(\mathrm{x} \mid \overline{\omega_i}) \cdot \mathrm{P}(\overline{\omega_i})\, d\mathrm{x} \Big\} \\
&= \sum_{i=1}^{N} \Big\{ \int_{\mathbf{X}} f\left(1 - \mathcal{O}_i(\mathrm{x}, \theta)\right) \rho(\mathrm{x}, \omega_i)\, d\mathrm{x} \\
&\quad + \int_{\mathbf{X}} f\left(\mathcal{O}_i(\mathrm{x}, \theta)\right) \rho(\mathrm{x}, \overline{\omega_i})\, d\mathrm{x} \Big\} \qquad (18)
\end{aligned}$$

Since

$$\begin{aligned}
\rho(\mathrm{x}, \omega_i) &= \frac{d}{d\mathrm{x}} \mathrm{P}(\mathrm{x}, \omega_i) \\
&= \frac{d}{d\mathrm{x}} \mathrm{P}(\omega_i, \mathrm{x}) \\
&= \frac{d}{d\mathrm{x}} \left[\mathrm{P}(\omega_i \mid \mathrm{x}) \cdot \mathrm{P}(\mathrm{x})\right] \\
&= \mathrm{P}(\omega_i \mid \mathrm{x}) \cdot \rho(\mathrm{x}) \qquad (19)
\end{aligned}$$

and

$$\rho(\mathrm{x}, \overline{\omega_i}) = \mathrm{P}(\overline{\omega_i}\ '\mathrm{x}) \cdot \rho(\mathrm{x}) \qquad (20)$$

one can re-state the expression of (18) as

$$\begin{aligned}
&\lim_{n_i \to \infty} \overline{\mathcal{E}} = \\
&\sum_{i=1}^{N} \Big\{ \int_{\mathbf{X}} f\left(1 - \mathcal{O}_i(\mathrm{x}, \theta)\right) \mathrm{P}(\omega_i \mid \mathrm{x}) \cdot \rho(\mathrm{x})\, d\mathrm{x} \\
&\quad + \int_{\mathbf{X}} f\left(\mathcal{O}_i(\mathrm{x}, \theta)\right) \mathrm{P}(\overline{\omega_i} \mid \mathrm{x}) \cdot \rho(\mathrm{x})\, d\mathrm{x} \Big\} \\
&= \sum_{i=1}^{N} \left\{ \underbrace{\begin{matrix} \int_{\mathbf{X}} \left[f\left(1 - \mathcal{O}_i(\mathrm{x}, \theta)\right) \cdot g_i(\mathrm{x})\right. \\ \left. + f\left(\mathcal{O}_i(\mathrm{x}, \theta)\right) \cdot \left(1 - g_i(\mathrm{x})\right)\right] \rho(\mathrm{x})\, d\mathrm{x} \end{matrix}}_{\epsilon_i} \right\} \\
&= \epsilon \qquad (21)
\end{aligned}$$

Clearly then, minimizing the reasonable error measure of (7) also minimizes the reasonable approximation errors of (16) and (17). In order for ϵ in (17) and (21) to be zero, it is necessary that the MLP's functional capacity exceed the functional complexity of all the class-conditional densities $\rho(\mathrm{x} \mid \omega_i)$ (see section 3.1.2).

3.3 SPECIFIC EXAMPLES OF REASONABLE ERROR MEASURES

One family of reasonable functions, which can be derived by inspection of (13), is

$$f(\mathcal{O}) = \int \mathcal{O}^{r}(1 - \mathcal{O})^{r-1}\, d\mathcal{O} \qquad (22)$$

This family has two special cases of great practical importance.

3.3.1 $r = 0$: Information Theoretic objective functions

One function that satisfies the reasonable conditions is

$$\begin{aligned}
f(\mathcal{O}) &= \int (1 - \mathcal{O})^{-1}\, d\mathcal{O} \\
&= -\log(1 - \mathcal{O}) \qquad (23)
\end{aligned}$$

— the functional expression used to implement the Cross Entropy, Maximum Mutual Information, Kullback-Liebler distance, and Maximum Likelihood objective functions [7, 10].

3.3.2 $r = 1$: Mean Squared Error

The MSE objective function is also a special case of (22):

$$\begin{aligned}
f(\mathcal{O}) &= \int \mathcal{O}\, d\mathcal{O} \\
&= \frac{1}{2}\mathcal{O}^2 \qquad (24)
\end{aligned}$$

3.4 SOME "UNREASONABLE" ERROR MEASURES

Obviously, any objective function which does not satisfy the *necessary* reasonable conditions will be an unreasonable function for estimating *a posteriori* probabilities. Nevertheless, many such unreasonable error functions will still yield asymptotic Bayesian discriminant performance. If its outputs asymptotically reflect the correct *ranking* of the *a posterioris*, an unreasonable error measure will yield Bayesian discriminant performance. We discuss two classes of objective functions that are unreasonable.

3.4.1 Minkowski-R error measures

When the objective function is of the form $f(\mathcal{O}) = \mathcal{O}^R$ — which corresponds to a Minkowski-R (L_R) metric [9] — one finds that the reasonable condition is satisfied only when

$$\begin{aligned}
\mathcal{O}^{R-1} &= \frac{\mathcal{O}}{1 - \mathcal{O}}(1 - \mathcal{O})^{R-1} \\
\mathcal{O}^{R-2} &= (1 - \mathcal{O})^{R-2}
\end{aligned}$$

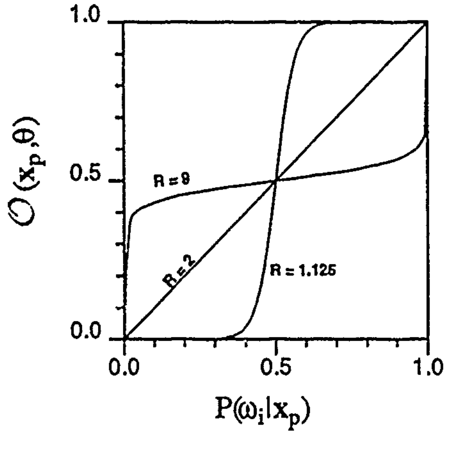


Figure 2: The L_R minimum error value of $\mathcal{O}_i(x_p, \theta)$ is plotted as a function of $P(\omega_i | x_p)$ for $R = 1.125, 2.0, 9.0$.

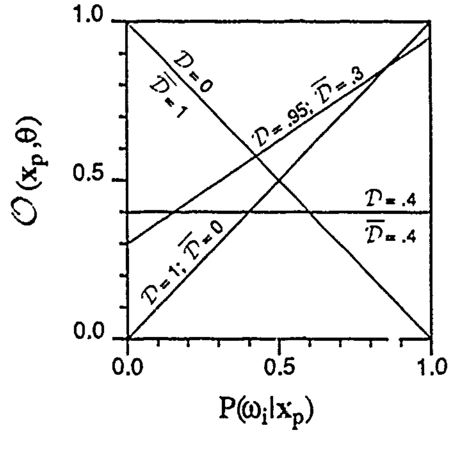


Figure 3. The minimum MSE value of $\mathcal{O}_i(x_p, \theta)$ is plotted as a function of $P(\omega_i | x_p)$ for various training target values.

or $R = 2$ ($r = 1$, in section 3.3.2). Another perspective is that $\overline{\mathcal{E}}$ is minimized when

$$\mathcal{O}_i(x_p, \theta) = \sqrt[R-1]{P(\omega_i | x_p)} \cdot \left[\sqrt[R-1]{P(\overline{\omega_i} | x_p)} + \sqrt[R-1]{P(\omega_i | x_p)} \right]^{-1}$$

which simplifies to $\mathcal{O}_i(x_p, \theta) = P(\omega_i | x_p)$ only when $R = 2$ (note that the L_2 metric is the MSE objective function). Since an L_R metric is reasonable only when $R = 2$, this argues against using L_R metrics other than L_2 when the output of the classifier is being interpreted as an *a posteriori* probability.

Figure 2 gives an intuitive feel for how various L_R metrics bias the output $\mathcal{O}_i(x_p, \theta)$ towards certainty (for $R \rightarrow 1$), or away from it (for $R \rightarrow \infty$): the minimum error value for $\mathcal{O}_i(x_p, \theta)$ is plotted as a function of $P(\omega_i | x_p)$ for various values of R. It should be noted that while L_R metrics are generally not reasonable error measures, they do in fact yield classifier outputs that asymptotically reflect the correct *ranking* of *a posteriori* probabilities.[3] Strictly speaking, they will yield Bayesian discriminant performance, and one can defend their use in training classifiers if the biases towards or away from certainty depicted in Figure 2 are not excessive for one's application.

3.4.2 Error measures with non-binary targets

Another class of unreasonable error measures is found if one employs otherwise reasonable error measures with non-binary targets $\{\mathcal{D}\}$. In such cases the resulting error measure will not be reasonable. Whether or not the resulting error measure reflects the correct *a posteriori* probability rankings depends on the choice of non-binary targets. We illustrate this point in the following sections as we derive the approximation error to the Bayesian discriminant function for the MSE and information theoretic error measures.

3.5 THE MSE APPROXIMATION TO THE BAYESIAN DISCRIMINANT FUNCTION

Using (16), one can define the *mean-squared approximation error* for the *i*th discriminant function as

$$\epsilon_i \triangleq \int_{\mathbf{X}} \left\{ [\mathcal{O}_i(\mathbf{x}) - g_i(\mathbf{x})]^2 - g_i(\mathbf{x})^2 + g_i(\mathbf{x}) \right\} \rho(\mathbf{x})\, dx \qquad (25)$$

Additionally, one can define the *aggregate* mean-squared approximation error as

$$\epsilon = \sum_{i=1}^{N} \epsilon_i \qquad (26)$$

One can express the average mean-squared error of the training set as

$$MSE \triangleq \frac{1}{n_t} \sum_{p=1}^{P} \sum_{i=1}^{N} \left\{ n_{pi} \cdot [\mathcal{O}_i(x_p, \theta) - \mathcal{D}_i(x_p)]^2 + n_{p\bar{i}} \cdot [\mathcal{O}_i(x_p, \theta) - \overline{\mathcal{D}}_i(x_p)]^2 \right\} \qquad (27)$$

Following the litany of section 3.1.2, one can express the asymptotic average mean-squared error as

[3] This is because $\mathcal{O}_i(x_p, \theta)$ is asymptotically a strictly increasing function of $P(\omega_i | x_p)$.

$$\lim_{n_t \to \infty} MSE =$$
$$\sum_{p=1}^{P} \mathrm{P}(\mathbf{x}_p) \sum_{i=1}^{N} \Big\{ \mathrm{P}(\omega_i \,|\, \mathbf{x}_p) \cdot \big[\mathcal{O}_i(\mathbf{x}_p, \theta) - \mathcal{D}_i(\mathbf{x}_p)\big]^2$$
$$+ \mathrm{P}(\overline{\omega_i} \,|\, \mathbf{x}_p) \cdot \big[\mathcal{O}_i(\mathbf{x}_p) - \overline{\mathcal{D}_i}(\mathbf{x}_p)\big]^2 \Big\} \qquad (28)$$

From this asymptotic form, one can show that the necessary condition for minimum MSE is

$$\lim_{n_t \to \infty} \mathcal{O}_i(\mathbf{x}_p, \theta) =$$
$$\mathcal{D}_i(\mathbf{x}_p) \cdot \mathrm{P}(\omega_i \,|\, \mathbf{x}_p) + \overline{\mathcal{D}_i}(\mathbf{x}_p) \cdot \mathrm{P}(\overline{\omega_i} \,|\, \mathbf{x}_p) \;\; \forall \mathbf{x}_p (29)$$

For the case in which binary targets as specified in (5) are used, the MSE objective function constitutes a reasonable error measure, and the classifier outputs asymptotically equate to the *a posterioris*. If $\mathcal{D}_i$ and $\overline{\mathcal{D}_i}$ are both set equal to the same value on the closed interval [0, 1], this will lead to a most undesirable asymptotic state in which all classifier outputs converge to $\mathcal{D}_i$ — a state of complete uncertainty analogous to that attained by the Minkowski-*R* error metric L_∞. For the case in which $\mathcal{D}_i > \overline{\mathcal{D}_i}$ and both targets are non-binary on [0,1], one finds that $\mathcal{O}_i(\mathbf{x}_p, \theta)$ is no longer an accurate estimate of $\mathrm{P}(\omega_i \,|\, \mathbf{x}_p)$, although it does remain a strictly increasing function of $\mathrm{P}(\omega_i \,|\, \mathbf{x}_p)$. For the bizarre case in which $\mathcal{D}_i < \overline{\mathcal{D}_i}$, $\mathcal{O}_i(\mathbf{x}_p, \theta)$ becomes a strictly increasing function of $\mathrm{P}(\overline{\omega_i} \,|\, \mathbf{x}_p)$ (or $1 - \mathrm{P}(\omega_i \,|\, \mathbf{x}_p)$). Figure 3 illustrates the effect of different target values on the asymptotic value of $\mathcal{O}_i(\mathbf{x}_p, \theta)$ plotted as a function of $\mathrm{P}(\omega_i \,|\, \mathbf{x}_p)$. As we shall see in the next section, this figure is relevant to information theoretic objective functions as well.

Returning to (28) one can derive the asymptotic mean-squared error (binary targets: $\mathcal{D}_i = 1, \overline{\mathcal{D}_i} = 0$) in terms of the aggregate approximation error to the Bayesian discriminant function (expressed in (25) and (26)). Using derivational procedures analogous to those of equations (18) – (21), one finds

$$\lim_{n_t \to \infty} MSE =$$
$$\sum_{p}^{P} \cdot \sum_{i=1}^{N} \Big\{ \mathrm{P}(\omega_i \,|\, \mathbf{x}_p) \cdot \big[\mathcal{O}_i(\mathbf{x}_p, \theta) - 1\big]^2$$
$$+ \mathrm{P}(\overline{\omega_i} \,|\, \mathbf{x}_p) \cdot \mathcal{O}_i(\mathbf{x}_p)^2 \Big\}$$
$$\asymp \sum_{i=1}^{N} \Big\{ \mathrm{P}(\omega_i)\, \mathrm{E}\Big[(\mathcal{O}_i(\mathbf{x}, \theta) - 1)^2 \,|\, \omega_i \Big]$$
$$+ \mathrm{P}(\overline{\omega_i})\, \mathrm{E}\Big[(\mathcal{O}_i(\mathbf{x}, \theta))^2 \,|\, \overline{\omega_i} \Big] \Big\}$$

$$= \sum_{i=1}^{N} \Big\{ \int_{\mathbf{X}} (\mathcal{O}_i(\mathbf{x}, \theta) - 1)^2 \, \rho(\mathbf{x}, \omega_i)\, d\mathbf{x}$$
$$+ \int_{\mathbf{X}} \mathcal{O}_i(\mathbf{x}, \theta)^2 \, \rho(\mathbf{x}, \overline{\omega_i})\, d\mathbf{x} \Big\} \qquad (30)$$

$$= \sum_{i=1}^{N} \Big\{ \int_{\mathbf{X}} \big[\mathcal{O}_i(\mathbf{x}, \theta)^2 - 2\,\mathcal{O}_i(\mathbf{x}, \theta) + 1\big]$$
$$\cdot g_i(\mathbf{x})\, \rho(\mathbf{x})\, d\mathbf{x}$$
$$+ \int_{\mathbf{X}} \mathcal{O}_i(\mathbf{x}, \theta)^2 \, \big[1 - g_i(\mathbf{x})\big]$$
$$\cdot\, \rho(\mathbf{x})\, d\mathbf{x} \Big\} \qquad (31)$$

$$= \sum_{i=1}^{N} \Big\{ \int_{\mathbf{X}} \mathcal{O}_i(\mathbf{x}, \theta)^2 \, \rho(\mathbf{x})\, d\mathbf{x}$$
$$- 2 \int_{\mathbf{X}} \mathcal{O}_i(\mathbf{x}, \theta)\, g_i(\mathbf{x})\, \rho(\mathbf{x})\, d\mathbf{x}$$
$$+ \int_{\mathbf{X}} g_i(\mathbf{x})\, \rho(\mathbf{x})\, d\mathbf{x} \Big\}$$

$$\lim_{n_t \to \infty} MSE =$$
$$\sum_{i=1}^{N} \left\{ \begin{array}{c} \int_{\mathbf{X}} \big[\mathcal{O}_i(\mathbf{x}, \theta) - g_i(\mathbf{x})\big]^2 \, \rho(\mathbf{x})\, d\mathbf{x} \\ \underbrace{- \int_{\mathbf{X}} (g_i(\mathbf{x}))^2 \, \rho(\mathbf{x})\, d\mathbf{x} + \mathrm{P}(\omega_i)}_{\epsilon_i} \end{array} \right\} \qquad (32)$$
$$= \epsilon$$

This result is the MLP analog of Duda and Hart's result for the MSE-trained perceptron ([6], pp. 154-155). A comparison of (32) and (25) confirms that each of the N terms in (32) is equivalent to the mean-squared approximation error term of (25). Thus, minimizing the MSE objective function of (27) (binary targets) also minimizes the mean-squared approximation errors of (25) and (26). Note that only the first term in (32) depends upon the output activations $\mathcal{O}_i$ of the MLP. In order for ϵ in (26) to be zero, it is necessary that the MLP's functional capacity exceed the functional complexity of all the class-conditional densities $\rho(\mathbf{x} \,|\, \omega_i)$ (see section 3.1.2).

Equation (32) illustrates the manner in which MSE is minimized during classifier training. The mean-squared approximation error term (ϵ) indicates that MSE is in fact a weighted integral sum of the squared errors between the MLP outputs $\mathcal{O}_i$ and their corresponding discriminant functions. The weighting factor is $\rho(\mathbf{x})$. The form of ϵ_i in (32) indicates that the approximation error minimization process focuses on the mode(s) of $\mathbf{x}$, where $\rho(\mathbf{x})$ is large. This issue is discussed further in section 5.

3.6 INFORMATION THEORETIC APPROXIMATIONS TO THE BAYESIAN DISCRIMINANT FUNCTION

Reference [7] shows that the information theoretic learning paradigms of Maximum Mutual Information, Kullback-Liebler distance, and Maximum Likelihood lead to a reasonable error measure known in the connectionist literature as the Cross Entropy (CE) objective function [10]. This error measure applied to a single input sample $\mathbf{x}_p$ belonging to class ω_i is expressed by

$$\begin{aligned} CE \overset{\Delta}{=} \\ & -\sum_{i=1}^{N} \Big\{\mathcal{D}_i(\mathbf{x}_p) \log\{\mathcal{O}_i(\mathbf{x}_p,\theta)\} \\ & + (1 - \mathcal{D}_i(\mathbf{x}_p)) \log\{1 - \mathcal{O}_i(\mathbf{x}_p,\theta)\}\Big\} \qquad (33) \end{aligned}$$

Given the Bayesian discriminant functions of (14), one can define the *cross-entropy approximation error* for the *i*th discriminant function as

$$\begin{aligned} \epsilon_i \overset{\Delta}{=} \\ & -\int_{\mathbf{X}} \Big[g_i(\mathbf{x}) \log\{\mathcal{O}_i(\mathbf{x},\theta)\} \\ & + (1 - g_i(\mathbf{x})) \log\{1 - \mathcal{O}_i(\mathbf{x},\theta)\}\Big] \rho(\mathbf{x})\, d\mathbf{x} \end{aligned} \qquad (34)$$

The *aggregate* cross-entropy approximation error is then given by

$$\epsilon = \sum_{i=1}^{N} \epsilon_i \qquad (35)$$

Given the definitions of tables 1 and 2, one can express the total cross entropy of the training set as

$$\begin{aligned} CE \overset{\Delta}{=} \\ & -\frac{1}{n_t} \sum_{p=1}^{P} \sum_{i=1}^{N} \Big\{ n_{pi} \cdot \big[\mathcal{D}_i(\mathbf{x}_p) \log\{\mathcal{O}_i(\mathbf{x}_p,\theta)\} \\ & + \big(1 - \mathcal{D}_i(\mathbf{x}_p)\big) \log\{1 - \mathcal{O}_i(\mathbf{x}_p,\theta)\}\big] \\ & + n_{p\bar{i}} \cdot \big[\overline{\mathcal{D}_i}(\mathbf{x}_p) \log\{\mathcal{O}_i(\mathbf{x}_p,\theta)\} \\ & + \big(1 - \overline{\mathcal{D}_i}(\mathbf{x}_p)\big) \log\{1 - \mathcal{O}_i(\mathbf{x}_p,\theta)\}\big]\Big\} \qquad (36) \end{aligned}$$

Following the litany of section 3.1.2, one can express the asymptotic cross entropy as

$$\begin{aligned} \lim_{n_t \to \infty} CE = \\ & -\sum_{p=1}^{P} \mathrm{P}(\mathbf{x}_p) \sum_{i=1}^{N} \Big\{ \mathrm{P}(\omega_i \,|\, \mathbf{x}_p) \big[\mathcal{D}_i(\mathbf{x}_p) \log\{\mathcal{O}_i(\mathbf{x}_p,\theta)\} \\ & + \big(1 - \mathcal{D}_i(\mathbf{x}_p)\big) \log\{1 - \mathcal{O}_i(\mathbf{x}_p,\theta)\}\big] \\ & + \mathrm{P}(\overline{\omega_i} \,|\, \mathbf{x}_p) \cdot \big[\overline{\mathcal{D}_i}(\mathbf{x}_p) \log\{\mathcal{O}_i(\mathbf{x}_p,\theta)\} \\ & + \big(1 - \overline{\mathcal{D}_i}(\mathbf{x}_p)\big) \log\{1 - \mathcal{O}_i(\mathbf{x}_p,\theta)\}\big]\Big\} \qquad (37) \end{aligned}$$

From this asymptotic form, one can show that the necessary condition for minimum Cross Entropy is

$$\begin{aligned} \lim_{n_t \to \infty} \mathcal{O}_i(\mathbf{x}_p,\theta) = \\ & \mathcal{D}_i(\mathbf{x}_p) \cdot \mathrm{P}(\omega_i \,|\, \mathbf{x}_p) + \overline{\mathcal{D}_i}(\mathbf{x}_p) \cdot \mathrm{P}(\overline{\omega_i} \,|\, \mathbf{x}_p) \;\; \forall \mathbf{x}_p (38) \end{aligned}$$

— precisely the same condition required for minimizing the MSE objective function. For this reason, the comments following (29) and Figure 3 accurately describe the dependence of information theoretic classifier outputs on target values: binary targets yield classifier outputs that asymptotically equate to the *a posterioris* $\mathrm{P}(\omega_i \,|\, \mathbf{x}_p)$.

Returning to (37) one can derive the asymptotic Cross Entropy (binary targets: $\mathcal{D}_i = 1$, $\overline{\mathcal{D}_i} = 0$) in terms of the aggregate approximation error to the Bayesian discriminant function (expressed in (34) and (35)). Using derivational procedures analogous to those of equations (18) – (21), one finds

$$\begin{aligned} \lim_{n_t \to \infty} CE = \\ & -\sum_{p=1}^{P} \mathrm{P}(\mathbf{x}_p) \sum_{i=1}^{N} \Big\{ \mathrm{P}(\omega_i \,|\, \mathbf{x}_p) \log\{\mathcal{O}_i(\mathbf{x}_p,\theta)\} \\ & + \mathrm{P}(\overline{\omega_i} \,|\, \mathbf{x}_p) \log\{1 - \mathcal{O}_i(\mathbf{x}_p,\theta)\}\Big\} \qquad (39) \\ \asymp \;& -\sum_{i=1}^{N} \Big\{ \mathrm{P}(\omega_i)\, \mathrm{E}\big[\log\{\mathcal{O}_i(\mathbf{x},\theta)\} \,|\, \omega_i\big] \\ & + \mathrm{P}(\overline{\omega_i})\, \mathrm{E}\big[\log\{1 - \mathcal{O}_i(\mathbf{x},\theta)\} \,|\, \overline{\omega_i}\big]\Big\} \\ = \;& -\sum_{i=1}^{N} \Big\{ \int_{\mathbf{X}} \log\{\mathcal{O}_i(\mathbf{x},\theta)\}\, \rho(\mathbf{x},\omega_i)\, d\mathbf{x} \\ & + \int_{\mathbf{X}} \log\{1 - \mathcal{O}_i(\mathbf{x},\theta)\}\, \rho(\mathbf{x},\omega_i)\, d\mathbf{x} \Big\} \end{aligned}$$

$$\begin{aligned}\lim_{n_t\to\infty} CE &= \\ &\sum_{i=1}^{N}\left\{\int_{\mathbf{X}} \log\{O_i(\mathbf{x},\theta)\}\,\underbrace{P(\omega_i\,|\,\mathbf{x})}_{g_i(\mathbf{X})}\,\rho(\mathbf{x})\,dx\right. \\ &\left.+\int_{\mathbf{X}} \log\{1-O_i(\mathbf{x},\theta)\}\,\underbrace{P(\overline{\omega_i}\,|\,\mathbf{x})}_{1-g_i(\mathbf{X})}\,\rho(\mathbf{x})\,dx\right\} \qquad (40)\\ &= \epsilon\end{aligned}$$

A comparison of (40) and (34) confirms that each of the N terms in (40) is equivalent to the cross entropy approximation error term of (34). Thus, minimizing the cross entropy objective function of (36) (binary targets) also minimizes the cross entropy approximation errors of (34) and (35). As with the MSE objective function, it is necessary that the MLP's functional capacity exceed the functional complexity of all the class-conditional densities $\rho(\mathbf{x}\,|\,\omega_i)$ in order for ϵ in (35) to be zero (see section 3.1.2).

Note that CE, much like its MSE counterpart, is a weighted integral sum of the cross entropy between between the MLP outputs $O_i(\mathbf{x}_p, \theta)$ and their corresponding discriminant functions. As with the MSE objective function, the form of ϵ_i for the CE objective function in (40) indicates that the approximation error minimization process focuses on the mode(s) of $\mathbf{x}$, where $\rho(\mathbf{x})$ is large (see section 5).

4 CLASSIFICATION FIGURES OF MERIT: LIMITED BAYESIAN PERFORMANCE WITHOUT EXPLICIT ESTIMATION OF *A POSTERIORI* PROBABILITIES

The N-monotonic CFM objective function [8] is given by

$$CFM_{mono} \overset{\Delta}{=} \frac{1}{n_t}\sum_{p=1}^{P}\sum_{i=1}^{N}\left\{n_{pi}\cdot\sigma\left[\Delta_i(\mathbf{x}_p,\theta)\right]\right\} \qquad (41)$$

where

$$\Delta_i(\mathbf{x}_p,\theta) \overset{\Delta}{=} O_i(\mathbf{x}_p,\theta) - \max O_{j\neq i}(\mathbf{x}_p) \qquad (42)$$

Thus, for n_{pi} cases of the prototype $\mathbf{x}_p$, output $O_i(\mathbf{x}_p, \theta)$ represents the correct class ω_i, while output $O_{j\neq i}(\mathbf{x}_p)$ is the most active output representing an incorrect class $\omega_{j\neq i}$. The function $\sigma\left[\Delta_i(\mathbf{x}_p, \theta)\right]$ is typically a strictly increasing continuously differentiable function of $\Delta_i(\mathbf{x}_p, \theta)$. The asymptotic form for CFM$_{mono}$ is

$$\begin{aligned}\lim_{n_t\to\infty} \mathrm{CFM}_{mono} &= \\ &\sum_{p=1}^{P} \mathrm{P}(\mathbf{x}_p) \underbrace{\sum_{i=1}^{N} \mathrm{P}(\omega_i\,|\,\mathbf{x}_p)\cdot\sigma\left[\Delta_i(\mathbf{x}_p,\theta)\right]}_{\mathrm{CFM}_{mono}(\mathbf{x}_p)} \qquad (43)\\ &\asymp \sum_{i=1}^{N} \mathrm{E}\left[\mathrm{P}(\omega_i\,|\,\mathbf{x})\cdot\sigma\left[\Delta_i(\mathbf{x},\theta)\right]\right] \qquad (44)\\ &= \sum_{i=1}^{N}\int_{\mathbf{X}} \underbrace{\mathrm{P}(\omega_i\,|\,\mathbf{x})}_{g_i(\mathbf{X})}\cdot\sigma\left[\Delta_i(\mathbf{x},\theta)\right]\,\rho(\mathbf{x})\,dx \qquad (45)\end{aligned}$$

Because $\sigma[\Delta]$ is a strictly increasing function of Δ, $\sigma'[\Delta] > 0$ and it is impossible to find the maximum of (43) by solving for the zero of its gradient with respect to the outputs $\{O\}$. Furthermore, the identity of $O_{j\neq i}(\mathbf{x}_p)$ in (42) is stochastic, so (43) is not a continuously differentiable function of the classifier outputs. As a result one cannot analytically determine the maximum CFM$_{mono}$ values of the classifier outputs $O_i(\mathbf{x}, \theta)$. Nevertheless, it is useful to consider how CFM$_{mono}$($\mathbf{x}_p$) — the CFM for a single prototype $\mathbf{x}_p$ — in (43) is maximized. Table 3 depicts the *a posterioris* and N CFM$_{mono}$($\mathbf{x}_p$) terms from (43) associated with the prototype $\mathbf{x}_p$. The *a posterioris* are ranked in decreasing order; their associated CFM$_{mono}$ terms are ranked in the same order. Thus $\mathrm{P}(\omega_{r1}\,|\,\mathbf{x}_p)$ is the largest *a posteriori* for $\mathbf{x}_p$ while $\mathrm{P}(\omega_{rN}\,|\,\mathbf{x}_p)$ is the smallest, and $\sigma\left[\Delta_{r1}(\mathbf{x}_p, \theta)\right]$ is the term involving output O_{r1} and its largest competitor O_{r2}.

We wish to show that if the *a posterioris* are ranked as shown in table 3, then the classifier output O_{r1} (underlined in the last column in table 3) corresponding to class ω_{r1} will be the most active of all outputs when CFM$_{mono}$($\mathbf{x}_p$) is maximized.

4.1 ASYMPTOTIC PERFORMANCE OF CFM$_{mono}$ FOR $\sigma[\Delta] = \mathrm{u}[\Delta]$

Figure 4 illustrates three different functions (normalized so that $-1 \leq \sigma[\Delta] \leq 0$) one might use to implement CFM$_{mono}$. The first of these functions is the Heaviside step function (denoted by u[Δ]). Clearly this function is an exception to the general rule stating that CFM$_{mono}$ is a strictly increasing continuously differentiable function of Δ. This functional form is of interest for two reasons. First, it is the MLP analog of the original perceptron learning criterion (e.g., [6], pg. 141); second, it leads to a very simple determination of the maximum CFM$_{mono}$($\mathbf{x}_p$) values for the classifier outputs. When this objective function is used to implement CFM$_{mono}$ learning, one can see readily from table 3 that CFM$_{mono}$ will be maximized if output O_{r1} is marginally bigger than any of its competitors. Because $\mathrm{u}'[\Delta] = 0\ \forall\ \Delta \neq 0$, there is no numerical incentive for O_{r1} to be made any more than marginally larger than its

Table 3: A ranking of the N *a posterioris* (and their corresponding CFM terms) of $\mathrm{CFM}_{mono}(\mathrm{x}_p)$ in (43).

$$\left.\begin{array}{lcl} \mathrm{P}(\omega_{r1}\,|\,\mathrm{x}_p) & : & \sigma\big[\Delta_{r1}(\mathrm{x}_p,\,\theta)\big] \\ \mathrm{P}(\omega_{r2}\,|\,\mathrm{x}_p) & : & \sigma\big[\Delta_{r2}(\mathrm{x}_p,\,\theta)\big] \\ \mathrm{P}(\omega_{r3}\,|\,\mathrm{x}_p) & : & \sigma\big[\Delta_{r3}(\mathrm{x}_p,\,\theta)\big] \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \cdot & & \cdot \\ \mathrm{P}(\omega_{rN}\,|\,\mathrm{x}_p) & : & \sigma\big[\Delta_{rN}(\mathrm{x}_p,\,\theta)\big] \end{array}\right\} = \left\{\begin{array}{l} \sigma(\underline{\mathcal{O}_{r1}} - \mathcal{O}_{r2}) \\ \sigma(\underline{\mathcal{O}_{r2}} - \mathcal{O}_{r1}) \\ \sigma(\underline{\mathcal{O}_{r3}} - \mathcal{O}_{r1}) \\ \quad\cdot \\ \quad\cdot \\ \quad\cdot \\ \sigma(\underline{\mathcal{O}_{rN}} - \mathcal{O}_{r1}) \end{array}\right.$$

competitors. The relative activation of outputs $\mathcal{O}_{r2} \rightarrow \mathcal{O}_{rN}$ is irrelevant beyond their being less that $\mathcal{O}_{r1}$. Thus, the Heaviside step functional form of CFM_{mono} implements the Bayesian discriminant function — albeit marginally – for asymptotically large training sets.

4.2 ASYMPTOTIC PERFORMANCE OF CFM_{mono} FOR STRICTLY INCREASING DIFFERENTIABLE FUNCTIONS OF Δ

In practice, learning with a discontinuous $\sigma[\Delta]$ like the Heaviside step is unstable. One can achieve stable learning using strictly increasing continuously differentiable functions of Δ [8]. One can analyze the asymptotic behavior of these functions by considering their effect on a set of classifier outputs in the initial equilibrium state for which all outputs are equal. If we define σ_e as the value of the CFM_{mono} function $\sigma[\Delta]$ when its argument $\Delta = 0$, and σ'_e as the derivative of the CFM_{mono} function at this same point (see inset in Figure 4), we can observe how the outputs will be perturbed from the equilibrium point as CFM_{mono} is maximized. A differential positive change in the value of $\mathcal{O}_{r1}$ results in a change to the over-all CFM_{mono} of

$$\begin{aligned} \frac{d\mathrm{CFM}_{mono}(\overline{\Delta} = 0)}{d\mathcal{O}_{r1}\uparrow} &= \\ &\sigma'_e \cdot \mathrm{P}(\omega_{r1}\,|\,\mathrm{x}_p) - \sigma'_e \cdot \mathrm{P}(\overline{\omega_{r1}}\,|\,\mathrm{x}_p) \qquad (46) \\ > 0 \;\; & \textit{iff} \;\; \mathrm{P}(\omega_{r1}\,|\,\mathrm{x}_p) > \mathrm{P}(\overline{\omega_{r1}}\,|\,\mathrm{x}_p) \\ & \text{or} \;\; \mathrm{P}(\omega_{r1}\,|\,\mathrm{x}_p) > 0.5 \end{aligned}$$

A differential negative change in the value of $\mathcal{O}_{r1}$ results in a re-ordering of the output rankings; $\mathcal{O}_{r1}$ becomes the *least* active output (all the other outputs remain unchanged), so all the terms in the right-most column of table 3 are altered to reflect this change in the identity of the most active competitor (refer back to (42)), and the net change in CFM_{mono} is given by

$$\frac{d\mathrm{CFM}_{mono}(\overline{\Delta} = 0)}{d\mathcal{O}_{r1}\downarrow} = -\sigma'_e \cdot \mathrm{P}(\omega_{r1}\,|\,\mathrm{x}_p) < 0 \quad (47)$$

One can show that altering any of the outputs $\mathcal{O}_{r2}$, $\mathcal{O}_{r3}$, $\ldots\mathcal{O}_{rN}$ independent of any alteration to $\mathcal{O}_{r1}$ from the equilibrium point always results in a net reduction in CFM_{mono}

$$\begin{aligned} \frac{d\mathrm{CFM}_{mono}(\overline{\Delta} = 0)}{d\mathcal{O}_{rj\neq r1}\uparrow} &= \\ &-\sigma'_e \cdot \mathrm{P}(\omega_{rj}\,|\,\mathrm{x}_p) + \sigma'_e \cdot \mathrm{P}(\overline{\omega_{rj}}\,|\,\mathrm{x}_p) \\ < 0 & \qquad (48) \end{aligned}$$

$$\frac{d\mathrm{CFM}_{mono}(\overline{\Delta} = 0)}{d\mathcal{O}_{rj\neq r1}\downarrow} = -\sigma'_e \cdot \mathrm{P}(\omega_{r_j}\,|\,\mathrm{x}_p) < 0 \quad (49)$$

Equations (46) - (49) do not indicate what the optimum values for $\mathcal{O}_{r1} \rightarrow \mathcal{O}_{rN}$ are when $\mathrm{P}(\omega_{r1}\,|\,\mathrm{x}_p) > 0.5$. These optimal values depend on the specific functional form of $\sigma[\Delta]$. When $\sigma[\Delta]$ is a linear function of Δ, the optimal values of the outputs are $\mathcal{O}_{r1} = 1$ and $\mathcal{O}_{r_j\neq r1} = 0$. Non-linear functional forms (such as the "maximally flat" one shown in Figure 4) tend to produce non-binary outputs $\mathcal{O}_{r1} < 1$ and $\mathcal{O}_{rj\neq r1} > 0$.

Equations (46) – (49) show that the equilibrium point yields sub-optimal CFM_{mono} only if the largest *a posteriori* is greater than 0.5. Since the class boundaries for an N-class problem are defined as the connected set of points on x at which all non-zero *a posterioris* are equal, continuously differentiable CFM_{mono} functions will (in theory) fail to form decision boundaries in regions on x where there are more than two non-zero *a posterioris* for asymptotically large training sets. Moreover, these equations indicate that continuously differentiable CFM_{mono} functions will set all classifier outputs equal in all regions on x where no *a posteriori* exceeds 0.5 — conceivably a large fraction of the domain of x when the number of classes N is large. While these asymptotic limitations would seem to render the CFM_{mono} class of objective functions useless for classification, in practice the functions compare quite favorably with reasonable error measures. Equation (44) may provide some insight into this apparent inconsistency.

Using the definition of correlation (denoted by ϕ), we can define the correlation between the ith CFM_{mono} term and its corresponding *a posteriori* by

$$\phi_i(\theta) = \left(\mathrm{E}\left[\mathrm{P}(\omega_i\,|\,\mathrm{x})\cdot\sigma\left[\Delta_i(\mathrm{x},\,\theta)\right]\right] - \mathrm{E}\left[\mathrm{P}(\omega_i\,|\,\mathrm{x})\right]\,\mathrm{E}\left[\sigma\left[\Delta_i(\mathrm{x},\,\theta)\right]\right]\right)\cdot\left(\mathrm{Var}\left[\mathrm{P}(\omega_i\,|\,\mathrm{x})\right]\cdot\mathrm{Var}\left[\sigma\left[\Delta_i(\mathrm{x},\,\theta)\right]\right]\right)^{-1/2} \quad (50)$$

As a result, it is possible to express (44) as

$$\mathrm{CFM}_{mono} \asymp \sum_{i=1}^{N}\left\{\sqrt{\mathrm{Var}\left[\mathrm{P}(\omega_i\,|\,\mathrm{x})\right]}\cdot\phi_i(\theta)\cdot\sqrt{\mathrm{Var}\left[\sigma\left[\Delta_i(\mathrm{x},\,\theta)\right]\right]} + \mathrm{P}(\omega_i)\cdot\mathrm{E}\left[\sigma\left[\Delta_i(\mathrm{x},\,\theta)\right]\right]\right\} \quad (51)$$

Since the terms $\sqrt{\mathrm{Var}[\mathrm{P}(\omega_i\,|\,\mathrm{x})]}$ and $\mathrm{P}(\omega_i)$ are not functions of the classifier's parameters θ, they are constants vis-a-vis optimizing CFM_{mono}in (51). Thus, maximizing CFM_{mono}tends to maximize the correlation between $\mathrm{P}(\omega_i\,|\,\mathrm{x})$ and $\sigma\left[\Delta_i(\mathrm{x},\,\theta)\right]$, along with the expectation and variance of $\sigma\left[\Delta_i(\mathrm{x},\,\theta)\right]$ for each class ω_i over the entire domain of x. In fact empirical studies bear this out. Classifiers trained with CFM_{mono} objective functions and relatively small sets of statistically independent samples tend to yield outputs with higher variance than their reasonable error measure counterparts; CFM_{mono}-trained classifiers also tend to yield multiple outputs with high activations when uncertain as to the class of a test sample. There is strong correlation between $\mathrm{P}(\omega_i\,|\,\mathrm{x})$ and $\sigma\left[\Delta_i(\mathrm{x},\,\theta)\right]$ as indicated by the CFM_{mono} classifiers' median error rate of 1.5% on a speaker-dependent /b,d,g/ phoneme recognition task [8].

5 COMMENTS ON THE APPLICABILITY OF THESE PROOFS TO THE STUDY OF GENERALIZATION IN MLP CLASSIFIERS

When one sets out to classify an RV x, the total number of statistically independent training samples n_t and the functional capacity (denoted by θ; see section 3.1.2) of one's classifier are two factors that will determine in large part the classifier's ultimate performance. We reproduce expressions for the MSE and information theoretic reasonable error measures and the analogous expression for the CFM_{mono} objective functions in order to illustrate the importance of these factors. Probabilities that rely on the asymptotic statistics of the training data are now presented as *estimates* (denoted by brackets "⟨⟩") of the true underlying probabilities:

$$MSE \asymp \sum_{i=1}^{N}\left\{\int_{\mathbf{X}}\left[O_i(\mathrm{x},\,\theta) - \underbrace{\langle \mathrm{P}(\omega_i\,|\,\mathrm{x})\rangle}_{g_i(\mathrm{X})}\right]^2\langle\rho(\mathrm{x})\rangle\,d\mathrm{x} - \int_{\mathbf{X}}\underbrace{\langle(\mathrm{P}(\omega_i\,|\,\mathrm{x}))^2\rangle}_{g_i(\mathrm{X})}\langle\rho(\mathrm{x})\rangle\,d\mathrm{x} + \langle\mathrm{P}(\omega_i)\rangle\right\} \quad (52)$$

$$CE \asymp -\sum_{i=1}^{N}\int_{\mathbf{X}}\left[\underbrace{\langle \mathrm{P}(\omega_i\,|\,\mathrm{x})\rangle}_{g_i(\mathrm{X})}\log\{O_i(\mathrm{x},\,\theta)\} + \left(1 - \underbrace{\langle \mathrm{P}(\omega_i\,|\,\mathrm{x})\rangle}_{g_i(\mathrm{X})}\right)\log\{1 - O_i(\mathrm{x},\,\theta)\}\right]\cdot\langle\rho(\mathrm{x})\rangle\,d\mathrm{x} \quad (53)$$

$$\mathrm{CFM}_{mono} \asymp \sum_{i=1}^{N}\int_{\mathbf{X}}\underbrace{\langle \mathrm{P}(\omega_i\,|\,\mathrm{x})\rangle}_{g_i(\mathrm{X})}\cdot\sigma\left[\Delta_i(\mathrm{x},\,\theta)\right]\langle\rho(\mathrm{x})\rangle\,d\mathrm{x} \quad (54)$$

Equations (52) – (54) indicate that the optimization of all of these objective functions depends on accurate estimates of the *a posterioris* and PDF of x — functions of the training set itself. Optimization also depends on sufficient functional capacity in θ. Finally, optimization — and thereby approximation of Bayesian discriminant performance — relies on the behavior of each objective function given some fixed number of training samples n_t and some fixed parameterization of the classifier θ.

In simplistic terms, there are four possible circumstances one will encounter:

- $n_t \rightarrow \infty$; θ sufficient: For the case in which one has plenty of independent training samples, one's training data will yield accurate estimates of the *a posterioris* and PDF of x over its entire domain. Furthermore, one's classifier will have sufficient functional capacity to model these estimates, and one will achieve Bayesian discriminant performance.
- $n_t \rightarrow \infty$; θ insufficient: For this case, the estimates of the *a posterioris* and PDF of x will be accurate, but the classifier will have insufficient functional capacity to

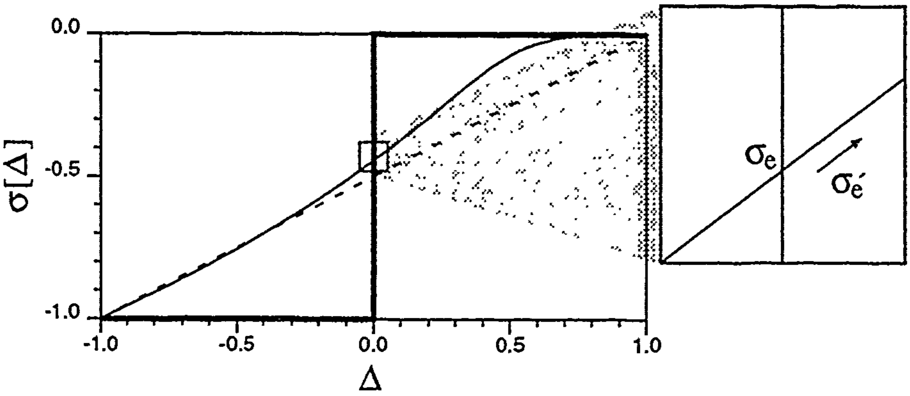


Figure 4. Three different functional implementations of the CFM_{mono} learning rule. Heaviside step, linear, and "maximally flat". Inset: a CFM_{mono} function in the vicinity of $\Delta = 0$.

model them. As a result, the classifier will not achieve Bayesian performance.

- $n_t \ll \infty$, θ sufficient. In practice one rarely has access to a sufficient amount of training data. In such cases the data will constitute poor estimates of the *a posterioris* and PDF of x. If the classifier has sufficient parameterization it will learn these inaccurate probabilistic estimates and will generalize poorly on disjoint test data ([6], section 3.8).
- $n_t \ll \infty$; θ insufficient: For the case in which one has insufficient data, it is often advantageous to train a classifier with reduced parameterization. In such cases the data will constitute poor estimates of the *a posterioris* and PDF of x, but the classifier will not have sufficient functional capacity to learn the less representative features of these inaccurate probabilistic estimates. Instead it will have only enough capacity to learn the gross features of these poor estimates, and generalization to disjoint test data will be as good as warranted by the training data.

This case has been studied in great detail from a number of different perspectives. PAC[4] analysis of learning using VC dimension applies a worst case analysis to the problem of learning from examples by deriving bounds on the number of exemplars needed to attain (with a desired probability) a desired accuracy. The VC dimension is defined only for concept classes that are discrete, or in connectionist parlance, for binary outputs (but potentially continuous inputs). Discrete concept classes require that the class-conditional densities of the input RV $\rho(x\,|\,\omega_i)$ be non-overlapping. For this reason the PAC formalism does not apply to our situation, in which estimates of *a posteriori* probabilities, or of the "best guess classification" (where the best guess might not be very accurate) are required.

VC PAC analysis has been applied to feedforward networks of binary threshold elements, to which we direct interested readers' attention [2].

The study of classifier generalization is typically viewed as a problem of determining the optimal parameterization for a classifier, given some fixed number of training samples. Interest in the functional form of the objective function used to train the classifier has rarely gone beyond establishing its validity as a learning metric on some information theoretic basis. But (52) – (54) clearly illustrate that different objective functions approximate the Bayesian discriminant function in markedly different ways. In this sense, given fixed n_t and θ, each objective function represents a different estimator of the Bayesian discriminant function. In detection and estimation theory, estimators are evaluated in terms of their bias and variance for finite sample sizes. Good estimators are those that are unbiased with minimal variance (as determined by the Cramér-Rao bound [5]); such estimators are characterized as "efficient". We feel that the study of generalization in MLP classifiers can be advanced by placing more emphasis on theoretical comparisons of objective functions as *estimators of the Bayesian discriminant function*. The derivations of this paper serve as a point of departure for such comparisons.

6 SUMMARY

Multi-Layer Perceptrons can be trained with two broad classes of objective functions to yield Bayesian discriminant performance. Reasonable error measures yield MLP outputs that (under conditions summarized below) asymptotically converge to the *a posteriori* probabilities associated with the input RV. Mean-squared error and information theoretic objective functions prove to be reasonable error measures. Classification Figures of Merit (CFM_{mono}) yield MLP outputs that generally reflect the identity of the max-

[4] Probably Approximately Correct.

imum *a posteriori* associated with any sample of the input RV for asymptotically large training sets.

The conditions necessary for MLP Bayesian discriminant performance (given that the classifier is trained with a reasonable error measure or a CFM_{mono} objective function) are

- The class-conditional densities of the input RV must be "well-behaved" to the extent that their complexity must be oounded (section 3.1.1).
- The functional capacity of the MLP classifier, expressed in its parameterization θ, must be sufficient to model the class-conditional densities of the input RV (section 3.1.2).
- The MLP training set must contain an asymptotically large number of statisticaliy independent training samples.

Given these results, we are inclined to view architecturally identical MLPs trained with different Bayesian objective functions as alternative *estimators* of the Bayesian discriminant function. We offer these results as a basis for evaluating MLP classifier generalization in the context of traditional detection and estimation theory.

Acknowledgments

This research was funded by grants from Bell Communications Research, ATR Interpreting Telephony Research Laboratories, and the National Science Foundation (NSF grant number EET-8716324). We wish to thank Professor B. V. K. Vijaya Kumar[5] and Dr. Dave Touretzky[6] for their critical review of this paper.

References

[1] A. Barron, "Statistical Learning Networks: A Unifying View" presented at 1988 Symposium on the Interface: Statistics and Computer Science.

[2] E. B.Baum and D. Haussler, "What Size Net Gives Valid Generalization?" *Neural Computation*, vol. 1, pp. 151-160, spring, 1989.

[3] H. Bourlard and C. Wellekens, "Links Between Markov Models and Multilayer Perceptrons," in *Advances in Neural Information Processing Systems, vol. 1*, Dave Touretzky, ed., San Diego: Morgan-Kaufmann, pp. 502-510, 1988.

[4] G. Cybenko "Approximation by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals and Systems*, vol 2, pp. 303-314, 1989.

[5] M. H. DeGroot, *Probability and Statistics, 2nd ed.*, Reading: Addison-Wesley, pp. 424-429, 1986.

[6] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, New York: John Wiley & Sons, 1973.

[7] H. Gish, "A Probabilistic Approach to the Understanding and Training of Neural Network Classifiers," in *Proceedings of the 1990 IEEE International Conference on Acoustics, Speech, and Signal Processing*, April, 1990, vol. 3, pp. 1361-1364.

[8] J. B. Hampshire II and A. H. Waibel, "A Novel Objective Function for Improved Phoneme Recognition Using Time-Delay Neural Networks", *IEEE Trans. Neural Networks*, vol. 1, pp. 216-228, June, 1990.

[9] S. J. Hanson and D. J. Burr, "Minkowski-*r* Back-Propagation. Learning in Connectionist Model with Non-Euclidean Error Signals", *Proceedings of the 1987 Neural Information Processing Conference, Am. Institute of Physics, D.*, D. Anderson, ed., pp. 348-357, 1988.

[10] G. E. Hinton, "Connectionist Learning Procedures", *Carnegie Mellon University Technical Report CMU-CS-87-115 (version 2)*, Dec. 1987, p. 14.

[11] R. Lippmann, "Pattern Classification Using Neural Networks" in *IEEE Communications Magazine*, vol. 27, No. 11, 1989.

[5] Dept. of Electrical and Computer Engineering, Carnegie Mellon University.

[6] School of Computer Science, Carnegie Mellon University.

A Local Approach to Optimal Queries

David Cohn
Dept. of Computer Science and Engineering
FR-35, University of Washington
Seattle, WA 98195

Abstract

We examine the problem of *self-directed learning* using queries in contrast to learning from random examples. We argue that, to provide any significant improvement over learning from random examples, a self-directed learning algorithm must make use of *sequential queries*, that is, new query points must be selected based on the answers to previous queries. Additionally, to perform well in a complex domain, such an algorithm must distinguish between degrees of the relative utility of querying various points in the domain. We present an algorithm that implements these two guidelines to produce "optimal" queries for a single linear threshold unit, in the sense that these queries minimize the bound on possible error. The algorithm is generalized to networks of such neurons using a local approach where each unit attempts to minimize its perceived error.

1 Overview

A large part of the research directed towards learning and generalization has concentrated on *learning from examples*, where a system is presented with a number of examples drawn at random from some distribution and is expected to exhibit generalization by being able to behave correctly when presented with novel examples from the same distribution. This approach is characterized by the work of Valiant (1984), Blumer *et al* (1989) and Haussler (1989). A recent area of interest in the field, contrasting with learning from examples, is that of *self-directed learning*. In this type of learning, the learner does not simply passively receive examples from the distribution, but takes at least some control over which examples it will see. Obviously, many natural systems exhibit self-directed learning, and in some formal situations, self-directed learning is provably more powerful than random sampling. Self-directed search in the form of queries has been studied theoretically by researchers such as Angluin (1986), and Eisenberg and Rivest (1990) as well as empirically by Hwang *et al* (1990) and others. This area is closely related to the field of control theory and optimal experiment design (Fedorov, 1972).

In Cohn *et al* (1990), we examined self-directed learning as a method for learning concept classification with a neural network. The strategy, called *selective sampling*, exhibited significant improvement over random sampling in the tests performed. This paper extends that work, demonstrating the utility of a sequential and graded-worth querying strategy. We define the learning problem in Section 2. In Section 3, we describe the difference between *batch* and *sequential* queries, and argue the need for any directed learning algorithm to use sequential queries. In Section 4 we describe selective sampling and show that one of its limitations stems from the inability to distinguish between different degrees of utility for querying points. Based on these observations, in Section 5 we give an algorithm for "optimal" querying in a single linear threshold unit and describe how this local approach may be generalized to incorporate multiple neurons and multiple layers.

2 Concept Learning and Generalization

We define the concept learning problem as follows: For some domain X we are given a concept class C consisting of a (possibly infinite) number of concepts $c \subseteq X$. There is some unknown target concept $t \in C$ such that for any point $x \in X$, if $x \in t$, then the correct classification (written $t(x)$) for x is 1, otherwise it is 0. The problem is to learn to approximate the correct binary classification over X.

We learn to approximate t by means of classified samples s of the form $(x, t(x))$. The sample points x may be drawn from some distribution $\mathcal{P}$ and the classification $t(x)$ obtained by querying an oracle. Based on a set of m such samples (written S^m), we choose a hypothesis $h \in C$ that is *consistent* with S^m, that is, for all $s \in S^m$, $h(x) = t(x)$. We define the *error* of h with respect to $\mathcal{P}$ as $error(h) = \mathcal{P}(\{x . x \in h\Delta t\})$, where $\mathcal{P}(A)$ is the measure of A according to $\mathcal{P}$, and Δ is the symmetric difference operator. The *generalization* of h is simply $1 - error(h)$.

For a given set A of unclassified points in X, define $\Pi_C(A) = \{A \cap c . c \in C\}$, that is, $\Pi_C(A)$ is the set of subsets of A that are "in" the various concepts in C. If $\Pi_C(A) = 2^A$, then we say that A is *shattered* by C. The *Vapnik-Chervonenkis dimension*[1] of a concept class C (written $\mathcal{VC}(C)$) is the largest number d such that there exists a set A, $|A| = d$ and A is shattered by C.

Blumer *et al* (1989) derived a bound for the number of random samples needed to learn a concept based on the the $\mathcal{VC}$-dimension of the class in consideration. For any class C, if $\mathcal{VC}(C) = d$ and if h is consistent with

$$m = \max\left[\frac{4}{\epsilon}\log_2\frac{2}{\delta}, \frac{8d}{\epsilon}\log_2\frac{13}{\epsilon}\right] \qquad (1)$$

randomly drawn training examples, then with probability at least $1 - \delta$, the error of h is bounded by ϵ (for $0 \leq \delta, \epsilon < 1$). Thus, if we know the dimension of our concept class, we can determine how many random samples we will need to draw and classify in order to have enough information to achieve good generalization with high confidence. This form of learning, bounding error by ϵ with confidence $1 - \delta$, is known as Probably-Almost-Correct, or PAC learning.

3 Generalization From Queries

In many situations, it is possible to not only draw and classify random samples from $\mathcal{P}$, but to query the oracle about arbitrary points in the domain. An immediate question is "How much can this new power improve our generalization?" Research along these lines has been pursued by Valiant (1984), Angluin (1986), Hwang *et al* (1990) and most recently by Eisenberg and Rivest (1990).

Eisenberg and Rivest address the question of PAC learning from random examples and queries. They show that for any unknown distribution meeting some smoothness criteria, the minimum number of examples needed to learn a concept is bounded from below by $\Omega(\frac{1}{\epsilon}\ln\frac{1}{\epsilon})$.

In this paper we make the distinction between *sequential* and *batch* queries. In sequential querying, the learner is allowed to know the result of a previous query before making the next one In batch querying, the learner is oblivious to the results of its queries until all queries have been made. We show that while sequential queries can, in many situations produce exponential improvements in generalization performance over random sampling (linear binary search, for example), batch querying will give us at *most* a constant factor improvement in our error bounds.

3.1 Bounding the error with batch queries

The concept of an *ϵ-transversal*, as described in Blumer *et al* (1989), is central to the idea of bounding error. For some set of regions (*concepts*) C in a domain X and a distribution $\mathcal{P}$ over X, an ϵ-transversal is a set of points A such that there is a point $x \in A$ inside every region in C that has probability mass greater than ϵ. An *ϵ-net* is a special case of an ϵ-transversal where the distribution $\mathcal{P}$ is uniform.

For a given target concept $t \in C$, we can define a set of regions $C_{\Delta t} = c\Delta t : c \in C$ (where Δ is the symmetric difference operator), as our *regions of error*. If a set of points A is an ϵ-transversal of $C_{\Delta t}$ then by classifying those points we will have samples that distinguish t from every other concept in C that differs from t by more than ϵ. The bound on m in the previous section is simply a bound on how many samples we must draw to have an ϵ-transversal of $C_{\Delta t}$ for an arbitrary $t \in C$ with high confidence.

The minimum number of points, m, necessary for an ϵ-transversal of an arbitrary class C has been bounded by Haussler and Welzl (1987) and most recently by Pach and Woeginger (1990) in terms of d, the $\mathcal{VC}$-dimensionof the class:

$$\frac{1}{48}\frac{d}{\epsilon}\log_2\frac{1}{\epsilon} < m \leq \frac{8d}{\epsilon}\log_2\frac{13}{\epsilon}. \qquad (2)$$

Blumer *et al* show that for any $t \in C$, $\mathcal{VC}(C) = \mathcal{VC}(C_{\Delta t})$. Thus, using the above bound, we can see that by drawing $\frac{8d}{\epsilon}\log_2\frac{13}{\epsilon}$ samples we have a non-zero probability of getting an ϵ-transversal of the regions of error.

It is important to realize, however, that a set of points that is an ϵ-transversal of $C_{\Delta t}$ for one target concept t is not necessarily an ϵ-transversal for some other target t' In order to bound the error of a hypothesis with absolute confidence, we must bound the error for *all* possible target concepts. For batch querying, this requires an ϵ-transversal of $C_\Delta^2 = \{c_1\Delta c_2 : c_1, c_2 \in C\}$, which is equivalent to a single ϵ-transversal of $C_{\Delta t}$ for all $t \in C$.

Claim: For a class C, if $\mathcal{VC}(C)$ is finite and $\emptyset \in C$ then $\mathcal{VC}(C) \leq \mathcal{VC}(C_\Delta^2) < 4 \cdot \log_2(6)\ \mathcal{VC}(C)$.

Proof: We can show the lower bound by realizing that if $\emptyset \in C$, C_Δ^2 contains C. This means that any set of

[1] For a more thorough explanation of the Vapnik-Chervonenkis dimension, see Blumer *et al* (1989), from which the above notation was taken.

points that class C is able to shatter is also shattered by class C^2_Δ. Thus, $\mathcal{VC}(C) \leq \mathcal{VC}(C^2_\Delta)$.

The upper bound may be demonstrated by application of Lemma 3.2.3 in Blumer *et al* (1989), an abbreviated version of which is described here. Consider a set $A \subseteq X$ and let $|A| = m$. By Proposition A2.1 of Blumer *et al*, fcr a class C such that $\mathcal{VC}(C) = d$, $\Pi_C(A) < (\frac{em}{d})^d$ (where e is the base of the natural logarithm), for all $m \geq d \geq 1$. Every set in $\Pi_{C^2_\Delta}$ is of the form $A_1 \Delta A_2$ where $A_1, A_2 \in \Pi_C(A)$. Therefore, $|\Pi_{C^2_\Delta}(A)| \leq (|\Pi_C(A)|)^2 < (\frac{em}{d})^{2d}$. For all $m \geq 4 \cdot \log_2(6) \cdot d$, it can be shown that $(\frac{em}{d})^{2d} < 2^m$. Therefore, if $|A| \geq 4 \cdot \log_2(6) \cdot d$, then $|\Pi_{C^2_\Delta}(A)| < 2^m$, and A is not shattered by C^2_Δ. So, $\mathcal{VC}(C^2_\Delta) < 4 \cdot \log_2(6) \cdot \mathcal{VC}(C)$. □

Thus, in order to bound the error of a hypothesis with perfect confidence[2], we need an ϵ-transversal of C^2_Δ, which would require, for an arbitrary class C, m points, where

$$\frac{1}{48}\frac{d}{\epsilon}\log_2\frac{1}{\epsilon} < m \leq \frac{32\log_2(6)d}{\epsilon}\log_2\frac{13}{\epsilon}. \qquad (3)$$

Comparing this with Equation 1, we can see that it is within a constant factor of the number of random samples needed PAC learn a concept over some unknown input distribution. This shows that, for learning by batch queries, we can expect at most a small improvement over random sampling in the number of samples needed to achieve a desired error rate, *even if the distribution is known*. We thus argue that any significant improvement in generalization ability made possible by querying must make use of sequential queries.

4 Selective Sampling as Sequential Querying

Consider the problem of concept learning from random examples, as described in Section 2. According to Equation 1, for fixed confidence δ, as we draw more samples from $\mathcal{P}$, the bound on our error decreases, but at an increasingly slower pace. This effect can intuitively be explained in light of declining efficiency of the sampling process. When learning from examples, the probability that a new sample will reduce our error is exactly the probability that our current h will misclassify the sample. As our learner progresses, fewer and fewer samples provide any new classification information.

Selective sampling (described in detail in Cohn *et al*, 1990), incorporates a formalization of the above argument. We can define the *region of uncertainty*, for C and set of samples S^m as

$$\begin{aligned} \mathcal{R}(S^m) \;=\; \{x : \exists c_1, c_2 \in C,\\ c_1, c_2 \text{ consistent with } S^m,\\ c_1(x) \neq c_2(x)\}. \end{aligned} \qquad (4)$$

The complement of this is the set of points that can provide us with no additional information, whose classifications are fixed by C and the already known training examples (see Figure 1). By drawing points from $\mathcal{P}$ but only querying their classification and training on them if they lie in $\mathcal{R}(S^m)$, we are "selectively sampling," effecting a simple form of self-directed learning[3].

This procedure may also be applied iteratively by redefining $\mathcal{R}(S^m)$ (and thus our sampling strategy) to incorporate the reduction of uncertainty after new points have been classified. In the limit, when $\mathcal{R}(S^m)$ is redefined after every sample, we have pure sequential querying: each point is chosen from a region based on the cumulative information from all previous points, and each one must reduce the region of uncertainty.

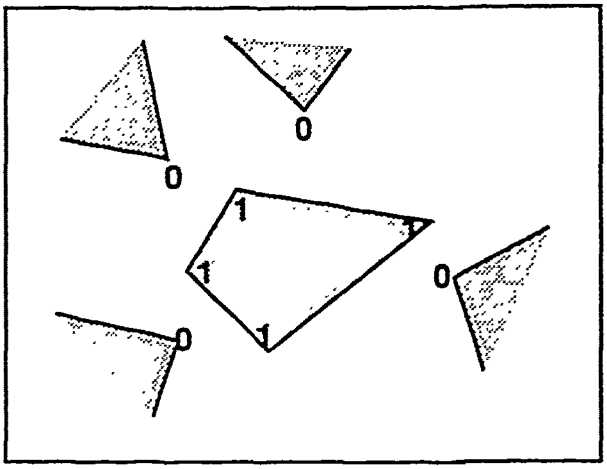


Figure 1: Example of a simple region of uncertainty. When learning the shape of a convex body, the classification of all points in the shaded areas is determined by the data already seen. The region of uncertainty is the star-shaped unshaded region.

[2] The confidence parameter in Equation 1 is a function of the probability of drawing "good" data. When one does not draw samples from a distribution, the confidence parameter disappears.

[3] If the distribution is known, then we can do away with sampling altogether and simply choose points inside $\mathcal{R}(S^m)$according to the distribution. In terms of the costs of training, selective sampling is only practical in problem domains where the cost of drawing a point from the distribution is small in comparison to the cost of classifying it.

4.1 Selective sampling with neural networks

It is not difficult to see how neural networks fit into this learning formalism: the set of all possible weights and parameters that may vary during training determine the concepts constituting the class C that the network is capable of learning. Baum and Haussler (1989), and Haussler (1989) examine this problem theoretically: given a network N, one can effectively bound $VC(N)$ and determine how many random examples one needs to train it on in order to achieve some error rate with high confidence.

In Cohn *et al*, a neural network implementation is used to approximate $\mathcal{R}(S^m)$ and selectively sample $\mathcal{P}$, thus effectively choosing the examples to train itself with. The network, called an *SG-network*, demonstrated significant increases in generalization over random sampling in several domains.

4.2 Limitations of selective sampling

One technical limitation of selective sampling is that for more complex classes, computation of $\mathcal{R}(S^m)$, or even its approximation, becomes very difficult. At some point, the computational cost of approximating $\mathcal{R}(S^m)$ can outweigh the benefit of requiring fewer training examples.

There is however, a more serious theoretical limitation. Selective sampling draws its power from the ability to differentiate a region of uncertainty from regions which can provide no further information. In cases where the representational complexity of the concept is large, however, $\mathcal{R}(S^m)$ can extend over the whole domain unless the concept is already well-learned. One very simple example of this problem is the class of the union of two circles in the plane (Figure 2). If one of the two circles is of size $\approx \epsilon$, then, until it has been "located" by a query, it could conceivably lie almost anywhere in the plane. Even though our maximum error may be close to ϵ, our region of uncertainty could contain the bulk of the domain. As a result, even though every query in $\mathcal{R}(S^m)$ will reduce our uncertainty, its probability of reducing our error is little better than that of a randomly drawn sample. As the complexity of the class being learned increases, this type of pathology becomes more and more common, in the limit reducing the effectiveness of selective sampling to that of random sampling.

Selective sampling was designed to be a "more optimal" method of gathering training points than random sampling. Its failing is that it cannot accommodate the fact that, among useful data points, not all samples will have the same value to the learning algorithm for reducing generalization error. While some "useful" queries will provide significant new information, in the limit of complex problems, a vast majority will provide only a very small incremental improvement.

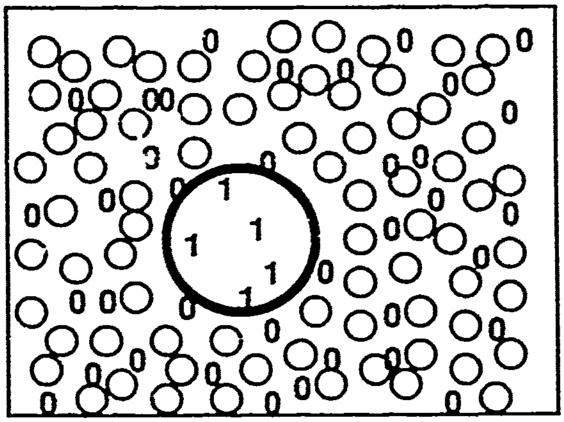


Figure 2: Example of the shortcomings of querying based on the region of uncertainty. We are trying to learn the union of two circles. One circle has been located, but until the other has been pinned down, the region of uncertainty covers almost the entire domain, even though the bound on our total error is small.

Thus, rather than merely asking "Will this point help us?" it is appropriate to ask "How *much* will this point help us?" We introduce a formalism to address this question in the next section.

5 Optimal queries

In previous sections, we have argued that unless a querying algorithm makes its queries sequentially, and distinguishes between graded values of querying points to reduce error, it can, in the most general case, offer little improvement over random sampling. In this section, we formalize a learning strategy that does both, and offer a measure of "optimality" for the value of querying arbitrary new points.

For a given class C, training set S^m and distribution $\mathcal{P}$, we can define a bound on the error as

$$\begin{aligned}\epsilon_{sup}(C, S^m, \mathcal{P}) \;&=\; \sup\{\mathcal{P}(c_1 \Delta c_2) : c_1, c_2 \in C,\\ &\qquad c_1, c_2 \text{ consistent with } S^m\}. \quad (5)\end{aligned}$$

This is just the maximum difference (weighted by the distribution) between any two hypotheses that are consistent with the training data. For the binary classification problem we are considering, we may measure the value of a new point $x \in X$ in terms of

$$\begin{aligned}\max[&\epsilon_{sup}(C, S^m \cup \{(x, 0)\}, \mathcal{P})\\ &\epsilon_{sup}(C, S^m \cup \{(x, 1)\}, \mathcal{P})], \qquad (6)\end{aligned}$$

and attempt to find an x that minimizes this value.

Determining this minimum is obviously not an easy problem. Calculating it for even a modestly sized network may be impractical (It may be noted that the

.ŠG-network used for selective sampling was designed to approximate ϵ_{sup}, and this took considerable time to determine a single case). What we propose as a tractable alternative is a local approach.

In the body of this section, we describe an algorithm for determining query points for a single linear threshold unit. We then demonstrate how this algorithm may be applied to a number of units in a single layer and to multi-layer networks as well.

5.1 Single element optimality

Consider the case of a single neuron learning to classify linearly separable data in $\Re^n$. For simplicity, we will assume that the distribution $\mathcal{P}$ on which we will measure error is known to be uniform over the unit n-cube. A set of labeled samples S^m restricts which hyperplanes may be used to separate the training data. Here, ϵ_{max} is just the maximum volume between two hyperplanes that are consistent with S^m.

A heuristic for approximating this volume is to enumerate the "extreme" hyperplanes. those consistent with S^m that border on exactly n points. Since the volume difference between hyperplanes is piecewise linear, we may visualize maximizing the difference between two hyperplanes by moving them apart until they abut constraining points in the training data. We may thus restrict ourselves to examining a finite (albeit large) number of candidate hyperplanes representing extreme hypotheses.

Using this metric, we may then determine the value of classifying a new point x by adding $(x, 1)$ to our training set, determining the new ϵ_{max}, and comparing it with the ϵ_{max} if we were to instead add $(x, 0)$ to our training set. The maximum of these two is the bound on our error if we know x, and thus is the value (to this particular neuron) of querying x.

We have created a simple implementation of this algorithm, employing a brute-force search for a point with near-optimal value. Figure 3 shows an example of the contours of value for a 2-dimensional case. Applying this algorithm iteratively, we compared the bound on generalization error for neurons trained on data from three runs of "optimal" queries with the bound for neurons trained on 12 runs of randomly sampled data The difference (see Figure 4) is dramatic. while the bounds for the randomly sampled data follow the roughly $\frac{1}{m}$ curve predicted by Blumer *et al*, the bound for the queried curve approaches zero exponentially fast.

An obvious drawback of the querying procedure described in this section is that it is computationally expensive; the number of hyperplanes that must be considered scales exponentially with input dimension. Additionally, (unlike the SG-network), the algorithm for determining queries is not built into the network structure; it operates *on* the classifier rather than being an intrinsic part of it. However, the procedure does appear to be structurally amenable to implementation using linear programming, which could be computationally tractable, or even relaxation search, which could be implemented in a neural network. These are currently being investigated by the author.

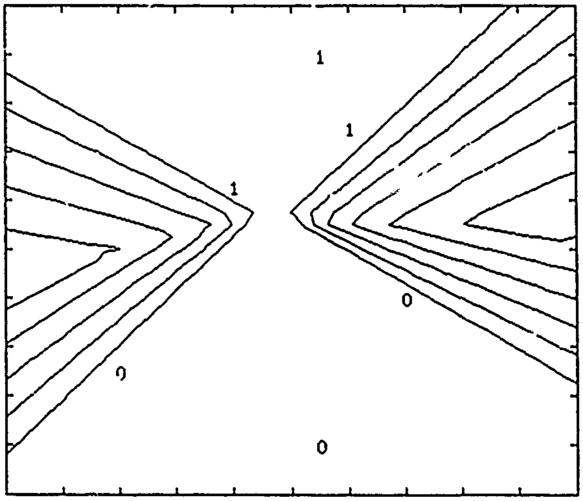


Figure 3: Contour map of the values of querying various point in the domain with respect to improvement on the bound of the possible error. Querying near (1.0, 0.58) will produce the greatest guaranteed decrease in the error bound.

5.2 Multiple element optimality

When a number of units take input from the same space, we may extend the above notion of querying with only minor adjustments. Intuitively, we may imagine simply computing the query value with respect to each unit and choosing the point with the overall greatest value to the layer.

The problem with this approach is that when thes units serve as an input layer for a more complex network the input data will likely not be linearly separable. Each unit will be distinguishing some part of the data set from some other, but none will be consistent with the entire set. What we end up with is the Credit Assignment Problem. which neuron should be responsible for separating what data?

There are two solutions to this problem. The globally optimal solution is to simply enumerate all sets of hyperplanes, one plane for each neuron in the layer per set and ensure that all distinguishable points are separated by some plane in the set. One would then repeat the maximization process with each of these sets. This is, of course, computationally intractable.

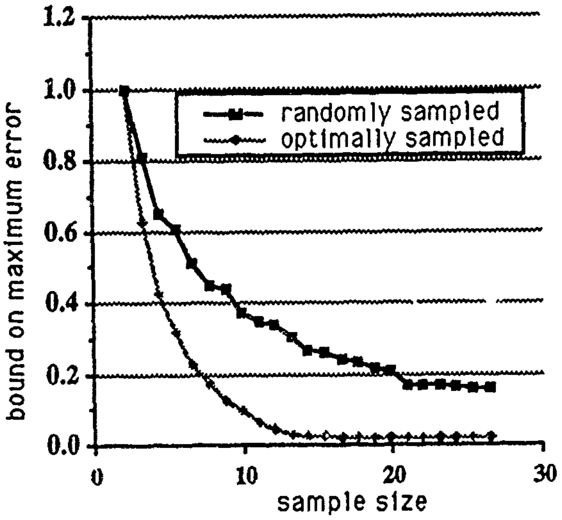


Figure 4: Comparative bounds on maximum error for runs of a single linear threshold unit using random sampling and locally optimal querying.

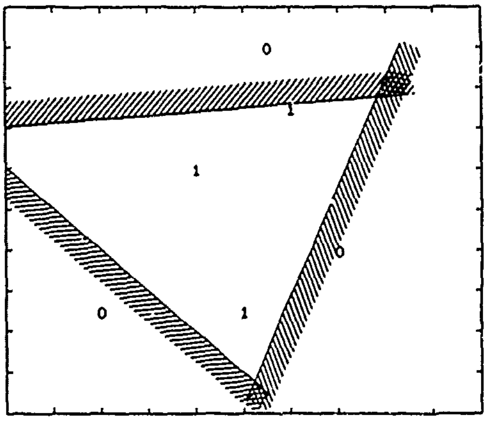


Figure 5. A more complex querying problem may be decomposed by first applying the classifier training algorithm and determining which neurons take responsibility for which subsets of the data.

A more reasonable (though not globally optimal) solution is to pursue a local approach to optimality: let each neuron, once it has settled on a part of its input domain, try to pursue what *it* believes is an optimal strategy for the points it separates. To illustrate, we consider the problem of learning a convex body using a single layer of neurons with outputs wired to a single AND gate.

Given an initial set of examples S^m, train the network on S^m and determine where the hyperplanes specified by the first layer units lie (Figure 5). For each hyperplane h defined by a unit, we may apply the above locally optimal querying algorithm, considering only the elements of S^m that are correctly classified by h (Figures 6a, b and c). The values of these queries may be summed over all input units, and point with the greatest overall value is chosen as the "optimal" point (Figure 6d). Note that this point may not be optimal in the global sense, it is a compromise between locally computed optima. The great advantage of this approach is that, since all but the final summation may be done with strictly local information, the computation scales linearly and may be parallelized.

5.3 Multiple layer optimality

We can view the multi-layer case as an extension of the single layer, multiple-element case. With any strictly feed-forward network, the computation may be decomposed into layers, where the job of each layer is to transform its k-dimensional input into an n-dimensional output subject to constraints imposed from above and below by the surrounding layers and training samples. It is this modularity that enables standard error backpropagation to employ the perceptron learning algorithm when training multi-layer networks of neurons.

Thus, we might expect the same techniques to suffice for determining, for each layer, what point in its input space it should query next. Since the queries we will be considering here are strictly in the input domain, we will have to "backpropagate" the interests of more remote layers to the input spaces of the lower, immediate ones. Experiments along these lines are currently being pursued.

6 Conclusion

In this paper, we have argued the need for a sequential querying method that makes judgements on the graded values of querying different points in the domain. We have presented a simple but effective local algorithm for making such queries to train a single linear threshold unit, and have described how this approach may be generalized for use with multiple-element, multiple-layer neural networks.

One of the motivations behind pursuing learning by queries is the possibility of training using far fewer examples than training by random sampling would require. The cost of determining which points to query must be weighed against this benefit. That the computational cost of the algorithm scales linearly with the number of units in a layer is good, but the brute-force method of computing optimality scales exponentially

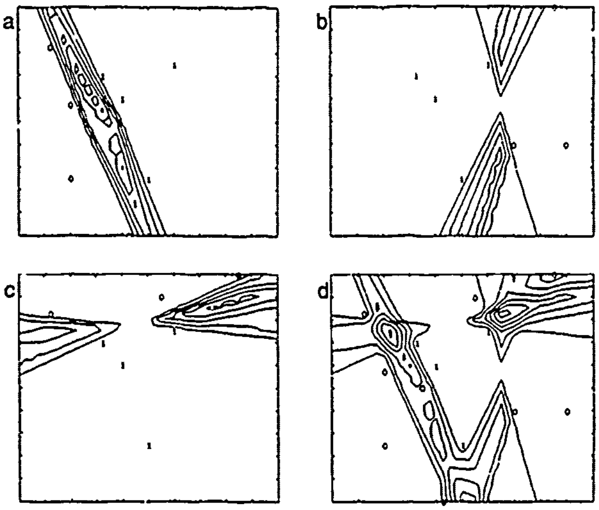


Figure 6: a), b), and c) The subset of the training data that is separated by each of the hyperplanes is considered in isolation to calculate the locally the best point to query. d) The locally determined "values" of querying various points are summed, giving a heuristic "best" point for the layer to query next.

in the dimension of the input space (and thus exponentially in the size of the previous layer, for a multi-layer network). More efficient computation strategies, such as those using linear programming, hold promise of bringing the complexity of the problem down sharply.

While many learning problems may not lend themselves to solution by self-directed learning and querying algorithms, those that do push us to search for insight in these intuitively appealing strategies.

Acknowledgements

Many thanks to David Haussler for the discussions and careful explanations of details of learning and the $\mathcal{VC}$-dimension, to Beatrice Golomb for the insight of learning using a local approach, and to my advisors, Richard Ladner and Les Atlas. This work was supported by the National Science Foundation and the Washington Technology Center.

References

Dana Angluin. "Learning regular sets from queries and counter-examples" Tech Report YALEU/DCS TR-64, Yale University, March 1986.

George Box, William Hunter, and J. Stuart Hunter. "Statistics for Experimenters" John Wiley and Sons, 1978.

Anselm Blumer, Andrej Ehrenfeucht, David Haussler, and Manfred Warmuth. "Learnability and the Vapnik-Chervonenkis dimension" *JACM 36, #4*, October 1989.

David Cohn, Les Atlas and Richard Ladner. "Training connectionist networks with queries and selective sampling" In *Advances in Neural Information Processing Systems 2*, Proceedings of the Neural Information Processing Systems Conference, Morgan Kaufmann 1990.

Bonnie Eisenberg and Ronald Rivest. "On the sample complexity of pac-learning using random and chosen examples" To appear in *ACM 3rd Annual Workshop on Computational Learning Theory*, Morgan Kaufmann 1990.

V. V. Fedorov. "Theory of Optimal Experiments" Academic Press, New York, 1972.

David Haussler and Emo Welzl. "ϵ-Nets and Simplex Range Queries" *Discrete Comput. Geom 2*:127-151, 1987.

David Haussler, Nick Littlestone, and Manfred Warmuth. "Predicting {0, 1}-Functions on Randomly Drawn Points" *Fundamentals of Computer Science 29*, 1988.

David Haussler. "Generalizing the PAC Model. Sample Size Bounds From Metric Dimension-based Uniform Convergence Results" *ACM 2nd Annual Workshop on Computational Learning Theory*, 1989.

Jenq-Neng Hwang, Jai Choi, Seho Oh and Robert Marks II. "Query learning based on boundary search and gradient computation of trained multilayer perceptrons" *IJCNN 90, San Diego, June 17-21, 1990.*

Daniel Schwartz, V. K. Samalam, Sara Solla and John Denker. "Exhaustive Learning" submitted to *Neural Computation*, 1989.

Leslie Valiant "A theory of the learnable" *Communications of the ACM*, 27:1134-1142, 1984.

Part VI

Modularity

A Modularization Scheme for Feedforward Networks

Arnfried Ossen
Institut für Angewandte Informatik*
Technical University of Berlin
Berlin W-1000/19, FR Germany

Abstract

This article proposes a modularization scheme for feedforward networks based on controllable internal representations. Control is achieved by replacing hidden units with pretrained modules that constrain internal patterns of activity to desired subsets. In the case of autoassociative feedforward networks these subsets can be seen as module interfaces. If enough *a priori* knowledge about a system is available, hierarchical systems with separately trainable anu exchangable modules can be built.

1 INTRODUCTION

Feedforward networks with backpropagation as a learning procedure have been successfully used for many problems. Generally speaking, they can approximate continuous nonlinear functions to any desired precision. But there are deficiencies: learning time efficiency is at least of polynomial order; generalization abilities of standard networks are insufficient; interpretation of network behavior is difficult, if not impossible.

Much work has been done on the optimization of learning time efficiency. The majority of the techniques developed are based on second order methods to minimize the number of steps the gradient descent procedure has to take. But even if significant progress could be achieved, it would not help improve generalization abilities of networks.

Le Cun [1989] has pointed out that the number of training examples required for good generalization scales like the logarithm of the number of functions which a specific network architecture can implement. Assuming that a small network is less general than a bigger one, i.e. that it can implement less functions, provided it is still able to compute the desired function, we can conclude that small networks yield better generalization than bigger networks, given the same amount of training data. Learning complexity per learning cycle decreases, then, simply because small networks contain less links and units. On the other hand, overall learning efficiency may deteriorate because of a decreasing convergence rate.

A couple of problem-independent strategies are known for the construction of small nets. Either an auxiliary error term is added to the cost function in order to penalize network configurations with many active links and/or units [Chauvin, 1989], or the relevance of links/units with respect to the error is determined, and the least relevant links/units are deleted [Mozer and Smolensky, 1989; le Cun *et al.*, 1990]. The problem with minimal network construction using auxiliary error terms is the difficulty of relative weighting of terms, which may cause convergence problems. Also, it might still be very difficult to understand the emerging patterns of activity in the hidden layers.

A more general approach is to minimize the number of free parameters in the network, e.g. by incorporating equality constraints between the weights of the network based on *a priori* knowledge of the problem [le Cun, 1989]. A significant reduction in learning complexity and better generalization can be obtained. On an even more general level, it is now recognized that the optimal network architecture for a given problem should be the one that can be described with the least number of bits, that is, the one with the *Minimum Description Length (MDL)* proposed by Rissanen.

A third approach is proposed in this article. *A priori* knowledge is used to select a constraint space for internal representations. The goal is to use the constrained internal representations as — within bounds -- flexible interfaces between modules of feedforward networks. In the first place, a break-down into modules would result in reduced learning complexity. In addition, if enough *a priori* knowledge is available, it should also be possible to tailor the interfaces in such a way that the number of free parameters is also reduced without losing the network's ability to imnlement the desired function, resulting in enhanced generalization abilities.

*Sekretariat FR 5-9, Franklin Str. 28/29, <ao@coma.cs.tu-berlin.de>

2 CONSTRAINING INTERNAL REPRESENTATIONS

This interface can be represented as a constraint space for patterns of activation at module boundaries. The patterns have to be controllable and interpretable, but must still be general enough to capture underlying regularities of the environment of the module during the learning procedure. On the other hand, they have to facilitate the encoding of external values and the decoding of adopted patterns of activity in terms of external concepts.

A promising way of defining these patterns is the use of coding schemes that restrict patterns of activation to specific subsets with well-defined values related to them. I have chosen a coarse coding [Hinton *et al.*, 1986] scheme of scalar values. In comparison to value-unit codings, it shows improved convergence [Hancock, 1989]. It also provides the resolution needed for the development of internal representations. The intended scalar value can simply be represented by sampling a unimodal function centered at this value [Saund, 1989]. Decoding of scalars is possible via auxiliary networks that map scalar representations to an activation value or by a least squares error procedure.

2.1 A CONSTRAINED NETWORK

Constraints are enforced by replacing the hidden layer of a standard feedforward network with a pretrained auto-associative network.

An auto-associative network m_1 trained for identity mapping of any desired scalar representations will generalize to reasonable representations at the outputs if clamped to unknown input data. If backpropagated to the input units, the error can be used to modify the inputs in a way that minimizes the error at the output units. If placed in the middle of a surrounding network m_2, adopted representations at the output units of m_1 serve as encodings of internal representations of m_2, while the backpropagated errors can be used to modify the weights between the input layer of m_2 and the input layer of m_1 (see figure 1). Thus two optimizations take place: (a) the output layer of m_1 will eventually generate almost perfect scalar representations without any weight changes in m_1 proper; (b) m_2 converges to its desired input/output mapping.

2.2 A CONSTRAINED ENCODER NETWORK

The scheme was tested on an *8-4-8* encoder whose hidden layer had been replaced by a *4-4-4* module. Three simulation runs were carried out (see figure 2). a) standard *8-4-8* system, b) *8-4-4-4-8* system (three hidden layers), c) *8-[4-4-4]-8* system with pretrained *4-4-4* abstract module. The weights of the links of the three hidden layers are copied from a *4-4-4* system that has learned to auto-associate arbitrarily-positioned coarse-coded scalars in a separate learning procedure. After being copied, the weights in the *4-4-4* module are fixed at their current values. Only weights from the input layer to the input layer of the *4-4-4* system and from the output layer of the *4-4-4* system to the output layer and the respective biases are subject to change by the backpropagation procedure. The *4-4-4* system merely serves to propagate activations and backpropagate errors, thereby constraining the internal representations of the encoder to scalar representations.

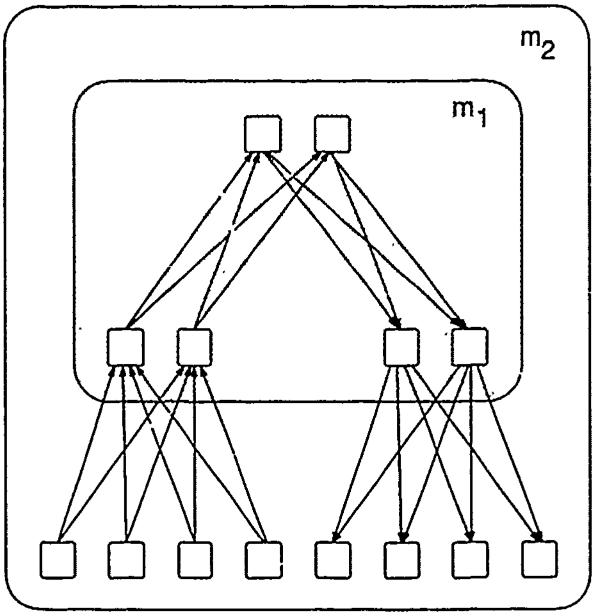


Figure 1: An auto-associative *2-2-2* module m_1 as an abstraction of the hidden layer of m_2.

The convergence of the modular system is very similar to the original encoder and about one order of magnitude better than a system without separate training. Of course, the training effort for the pretrained module has to be taken into account, too. But since backpropagation is typically of polynomial order, the additional complexity (40 free weights) is low in comparison to the complexity of the training cycles for the *8-4-4-4-8* (120 free weights) and *8-[4-4-4]-8* (80 free weights) systems.

3 APPLICATIONS

3.1 NONLINEAR DIMENSIONALITY REDUCTION

This system can be applied to nonlinear dimensionality reduction problems [Saund, 1989], which turn out to be the special case of equal representations at input and hidden layers. The constraining module has to be pretrained to auto-associate the chosen scalar coding, while the original modules auto-associate the higher dimensional data. The constraining module pressures internal patterns of activation into the desired coding, without need for spe-

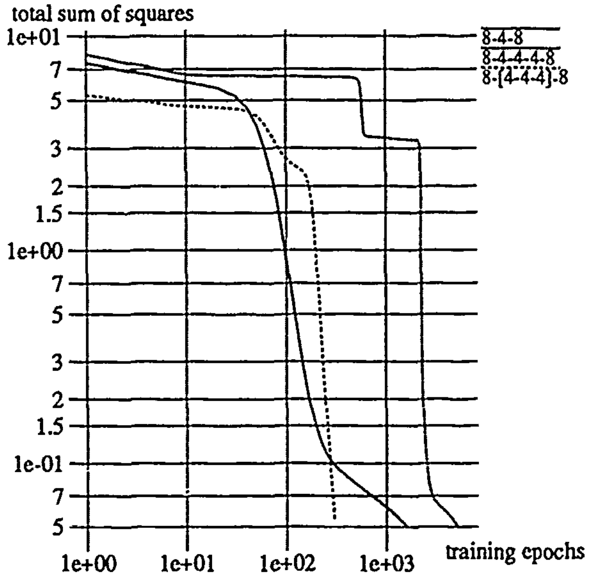


Figure 2: Convergence of *8-4-8* encoder, *8-4-4-4-8* encoder, and *8-[4-4-4]-8* encoder with pretrained abstract module.

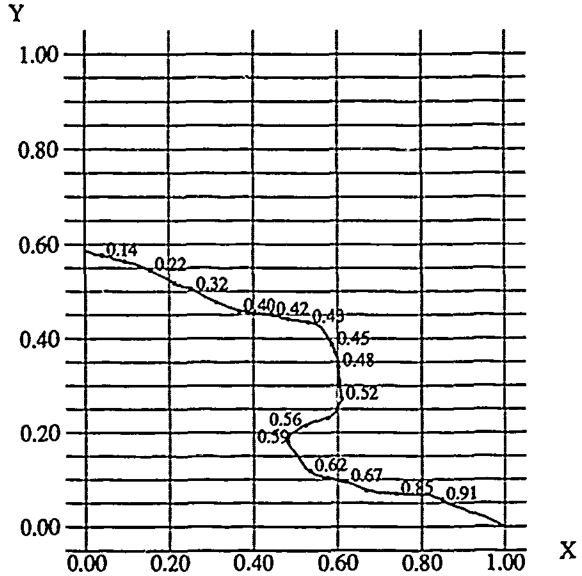


Figure 3: Nonlinear dimensionality reduction from two-dimensional curve data to one-dimensional data denoting points along the curve.

cial convergence strategies[1]. Figure 3 shows a nonlinear dimensionality reduction from two-dimensional data to a one-dimensional constraint using a *16-[8-8-8]-16* architecture.

3.2 INTERFACE DEFINITION

In truly modular systems, e.g. large computer programs, there is typically a design phase requiring the strict definition of interfaces before the construction of modules can take place. This includes a fixed input/output relation for all data to be processed. In addition, the representation of data at module boundaries is given by data types, that is, the range of possible values a parameter can adopt. Once a design is completed, modules can be constructed separately, which in general results in a significant reduction in implementation complexity. Also, modules can be exchanged, e.g. if implemented inefficiently, at any time without interfering with other parts of the system.

How might this scheme be transferred to modular neural networks? An important feature of neural nets is their ability to learn from examples, that is, to develop internal representations according to the underlying regularities of the environment. If internal representations are to be used as interfaces, all internal representations that can emerge during learning have to be known in advance in order to define module input/output relations, making the learning phase senseless.

This dilemma can partly be overcome if interfaces are defined more loosely. One way of achieving a loose coupling is to take advantage of the characteristics of scalar codings. In the case of an auto-associative feedforward network, where internal representations are forced into scalar codings and where inputs/targets are also presented in the form of scalar codings, a sufficient approximation of the targets is only possible if the scalar codings at the hidden layer evolve in orderly fashion. In other words, internal codings are almost equally distributed over the given interval and are in ascending or descending order with respect to corresponding inputs/targets, as shown above for the two-to-one dimensionality-reduction problem. The behavior very much resembles a simple topology-preserving mapping as known from Kohonen's self-organizing feature maps [Kohonen, 1989].

Given that, an interface can be defined by a type definition, that is, the range of valid scalar values, and the assertion that a neighborship relation between patterns of activation at inputs/targets and internal patterns will be preserved. On the other hand, a learning system consisting of more loosely coupled modules will lose some of the capabilities a strictly defined system has.

3.3 EXAMPLE

Let us consider a system where some raw data is to be processed in several steps. First, data is compressed into

[1] To obtain convergence, Saund [1989] has to use a simulated annealing schedule (smoothness of scalar coding) and a method of "encouraging" scalarized behavior by increasing peaks and decreasing valleys in a particular trial.

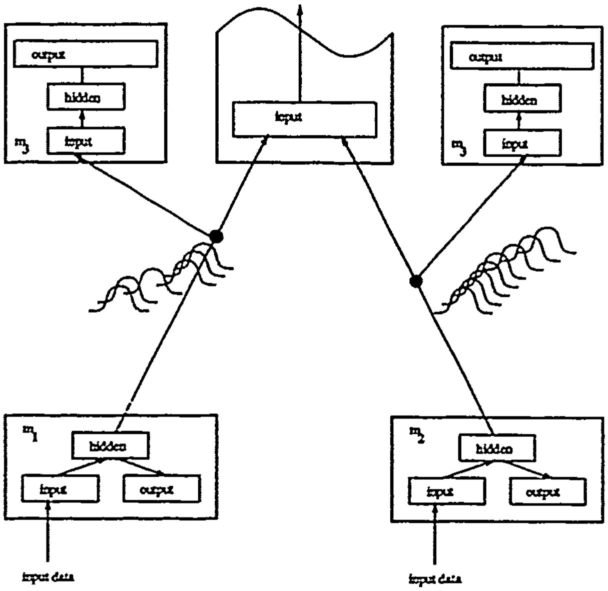


Figure 4: A modular system with scalar coded interfaces.

compact encodings, then a second step is applied, e.g. feature values derived from the encodings are displayed. Then further processing steps are carried out.

Such a system could be broken down into modules loosely coupled via scalar-coded interfaces. Figure 4 shows an example. Two modules (m_1, m_2) of auto-associative feed-forward networks learn to transform the raw data into scalar-coded internal representations. Assuming that the low-level configuration is appropriate for the task (*a priori* knowledge), two sets of scalar-coded sequences are learned. Since the possible values are known, it is sufficient to train one display module (m_3) to indicate the adopted values of both low-level modules. The display module can therefore be trained separately and may be exchanged.

4 DISCUSSION

Modularization is a promising way of combatting learning time complexity in backpropagation networks. On the other hand, the above modular system would be of little use if absolute feature values were important for further processing. It was not possible to train higher-level modules separately and no reduction in learning complexity was achievable. However, the topology-preserving behavior of scalar-coded interfaces might be a sufficient justification here, at least for a subset of problems. For these cases, a loosely-coupled modular system not only allows the required learning epochs to be reduced; a means of interpreting patterns of activity at module interfaces is also achieved.

A LEARNING PROCEDURE ADJUSTMENTS

All simulations were carried out using the standard back-propagation algorithm with a fixed learning rate of $\eta = 0.2$, a fixed momentum of $\alpha = 0.1$, weight update after each presentation (on-line mode), and sequential presentation of patterns. To speed up learning, the gradient was normalized before updating the weight. This follows from the empirical observation that the product of optimal learning rate and absolute value of the gradient remains almost constant over all learning cycles, see [Salomon, 1989]. Scalar codings were created using the derivative of the sigmoidial function $1/(1+e^{-x/t})$ at a temperature of $t = 1/8$.

Acknowledgement

I would like to thank Albrecht Biedl for his helpful comments on early drafts of the paper. My special thanks go to Geoffrey Hinton for his critical comments during revision of a previous published version of this paper.

References

[Chauvin, 1989] Yves Chauvin. A back-propagation algorithm with optimal use of hidden units. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems I*, pages 519–526. Morgan Kaufmann Publishers, San Mateo, California, 1989.

[Hancock, 1989] Peter J. B. Hancock. Data representation in neural nets: An empirical study. In David Touretzky, Geoffrey Hinton, and Terrence Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 11–20, San Mateo, CA, 1989. Morgan Kaufmann Publishers.

[Hinton *et al.*, 1986] Geoffrey E. Hinton, James L. McClelland, and David E. Rumelhart. Distributed representations. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, chapter 3. MIT Press/Bradford Books, 1986.

[Kohonen, 1989] Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer Verlag, 1989.

[le Cun *et al.*, 1990] Yann le Cun, John S. Denker, and Sara A. Solla. Optimal brain damage. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems II*, pages 598–605. Morgan Kaufmann Publishers, San Mateo, California, 1990.

[le Cun, 1989] Yann le Cun. Generalization and network design strategies. In Rolf Pfeiffer, Zoltan Schreter, Françoise Fogelman-Soulié, and Luc Steels, editors, *Proceedings — Connectionism in Perspective*. Swiss Group for Artificial Intelligence and Cognitive Science (SGAICO). Elsevier Science Publishers B.V., 1989.

[Mozer and Smolensky, 1989] Michael C. Mozer and Paul Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. Technical Report CU-CS-421-89, University of Colorado at Boulder, Boulder, January 1989.

[Salomon, 1989] Ralf Salomon. Adaptiv geregelte Lernrate bei Back-propagation. Technical Report 89-24, Technische Universität Berlin, 1989. Forschungsberichte des Fachbereichs Informatik.

[Saund, 1989] Eric Saund. Dimensionality-reduction using connectionist networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(3):304–314, March 1989.

A Compositional Connectionist Architecture

James R. Chen
Computer Science & Engr. Dept.
Univ. California at San Diego
La Jolla, CA 92093

Abstract

CompoNet is a connectionist architecture that facilitates modular knowledge composition. The network uses multiplicative links to control modular activations, as well as to dynamically recruit hidden units. Multiple relations between different sets of units can be loaded onto the same network at different time. Relations added later to the network can take advantage of established mappings and hidden unit representations, without disturbing previous trainings.

1 INTRODUCTION

The connectionist approach to the study of artificial intelligence has been flourishing in the past few years. Among many attractive features of the connectionist networks, most impressive are their adaptability to environmental information and noise, and their capability as universal approximator to arbitrary mappings [4]. Most of the research works, however, have focused on tackling a specific problem domain with specially designed network architecture, without addressing the problem of knowledge composition, i.e., systematic integration of network modules [2]. Network composition not only facilitates the incorporation of training efforts, but is also crucial to the role of connectionist networks in modeling symbolic reasoning systems [6,7,9]. Without compositional capability, networks are severely limited in their power of representation, and cannot be used as basic components of a knowledge base. Also, the lack of a general purpose network configuration makes it very difficult for connectionist networks to interface with other tools of computation such as databases, production systems, or even some special purpose connectionist processing module. Lastly, a common network configuration could also promote the realization of parallel connectionist hardware.

A general purpose connectionist architecture that facilitates knowledge composition must preserve the adaptive power of such networks. It must be able to accommodate multiple relations, and yet only a small part of the network should be activated at a time to handle the current problem, without invoking irrelevant information processing. Later training should not negatively disturb information stored previously in the network. And most importantly, the network must be able to take advantage of previously learned knowledge in later training.

More specifically, we need a way to control which nodes and links in the network, pre-trained or unexplored, are involved, given some specified sets of input and/or output. We also need some mechanism to decide which hidden units are to be allocated for current training.

In the next section, we present a connectionist architecture which provides a general purpose network configuration and systematically integrates different network modules. Simulation performance of the model in terms of knowledge composition and hidden unit recruiting will be discussed in section 3. Lastly we will summarize our research findings and possibilities of future research.

2 CompoNet: A COMPOSITIONAL CONNECTIONIST ARCHITECTURE

CompoNet is a network architecture that facilitates connectionist knowledge composition. The main idea behind CompoNet is to control modular activations with multiplicative gating on each node, and to recruit necessary hidden units by adjusting the weights of the links connected to the gates.

2.1 ARCHITECTURE

The basic architecture is depicted in figure 1. The network consists of a set of visible nodes as well as a set of hidden nodes. There are complete bi-directional connections among visible nodes, where reciprocal links

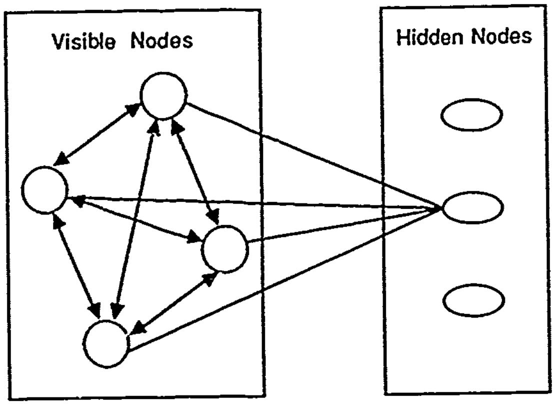


Figure 1. Basic Architecture - CompoNet consists of a set of visible nodes and a set of hidden nodes

can have arbitrary weights. There are no interconnections among hidden nodes. Between hidden nodes and visible nodes are complete bipartite connections. These links are uni-directional although the exact directions are not determined until their associated hidden nodes are recruited in training.

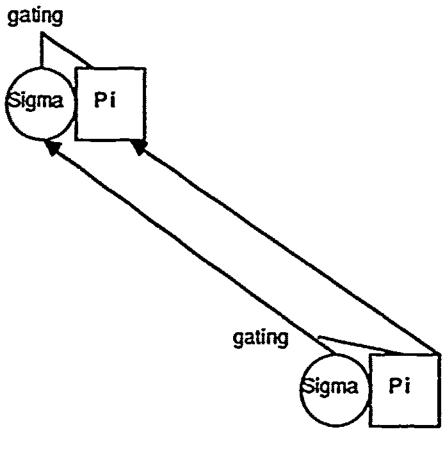


Figure 2: Sigma-unit And Pi-unit - two components of a node

As shown ᵗn figure 2, each node is actually composed of two units, a regular sigma-unit, and a pi-unit used to gate the output of the sigma unit. Both units take the sum of weighted incoming signals as their net input. The sigma-unit has continuous activation from -1 to 1, using a standard logistic squashing function, whereas the pi-unit is a threshold unit with binary activations of either 0 or 1. The pi-unit does multiplicative gating on its associated sigma-unit such that the sigma-unit's activation is only visible to others when the pi-unit is on.

Correspondingly, each connection in figure 1 actually consists of a sigma-link connecting sigma-units, and a pi-link connecting pi-units. A weight is associated with each link. In addition, each sigma link also carries a bias. Biases are typically associated with nodes in most connectionist networks. In CompoNet, however, it is necessary to keep the biases on the links since a visible node may participate in different mappings at different time, and thus requires different biases. Weights associated with pi-links are restricted to be non-negative, and they are initialized to zero. Weights and biases on the sigma-links are initialized to random values.

Also associated with each hidden node is a "used" flag which indicates that the hidden node has been recruited already. Similarly a "trained" flag is attached to each connection to signify that the connection has been used in training. Their usage will be discussed later.

2.2 ACTIVATION AND TRAINING

When a subset of visible nodes are clamped as input, the activation values of their sigma-units are set according to the input pattern, and their pi-units are turned on, that is, set to the value of 1. The values of both units are forwarded to other nodes they are connected to, through both pi-links and sigma-links. A pi-unit will be turned on if the sum of its weighted inputs exceeds a fixed threshold. Its associated sigma-unit then becomes active and forwards its value to other nodes. The route of forward propagation is determined by previous trainings since only trained pi-links have large enough weights to carry the information further. Such forward activations are continued only for a fixed small number of steps each time a set of input nodes are clamped. Thus the input nodes can only propagate information through trained connections to other nodes reachable in a finite number of forward steps.

During training, another subset of visible nodes are designated as target outputs. As in standard backpropagation learning [5], an output node computes and propagates error back to all nodes connected to it through sigma-links. Error propagation continues through active hidden nodes, but stops at any visible nodes or inactive hidden nodes. Only weights associated with sigma-links between active nodes are adjusted. An inactive node has zero activation hence change of weights on links to and from this node cannot contribute to reduce the global error measure.

Weights associated with the direct pi-links from input nodes to target output nodes are increased quickly till the output pi-units are turned on. Thus after training these output nodes will become active whenever the input nodes are clamped. Weights of pi-links involving hidden nodes are adjusted in the recruiting process.

2.3 RECRUITING

Since all pi-links initially have zero weights, no hidden nodes are used at the initial training stage. If the direct connections between input and output nodes become insufficient for the current mapping, new hidden nodes need to be recruited. This is done through adjusting weights of the pi-links connected to inactive hidden units. Each inactive hidden node connected from the clamped input has a potential activation value since untrained sigma-links carry random weights. This activation, however, is not visible to other nodes since it is blocked by the inactive pi-unit. Hidden units can also collect error information from target nodes. Based on this information, a hidden unit can compute the amount of improvement it could have contributed to the global error, if its pi-unit were turned on. Such measure of improvement is then used to adjust in proportion the weights of the hidden unit's incoming pi-links from the input nodes. Consequently, weights on pi-links connected to the most "useful" hidden unit will grow fastest, and thereby activate that hidden unit first. The sigma-unit of a recruited hidden node is used in back-propagation, changing the landscape of the weight space as well as the network's current position in that space. As a result, the "usefulness" of different inactive hidden nodes also changes and a previously growing pi-link weight may now move toward the opposite direction.

To compute the potential "improvement" it can make to a particular target node, a hidden node needs to know its own activation, the weight on the connection to the target node, as well as the current net input of the target node. Alternatively, the computation can be done at a target node and then propagated back to the hidden unit. In either case, the computation required for the recruiting algorithm is more costly than that for the standard back-propagation. Nevertheless, it is in the same order of computation complexity since both are bounded by the number of weights in the system. The same is true with parallel hardware provided that there is one processor for each connection. With one processor per node, however, the recruiting algorithm is more expensive than back-propagation since an inactive hidden unit cannot simply sum up information back-propagated from target nodes.

Growing the pi-links as described above does not assure some inactive hidden nodes can always be allocated. When the network has successfully learned most but not all of the training patterns, weights on the pi-links decline to zero since any newly added hidden node would likely have adverse effect on the majority of the training set. While such natural decay has the advantage of automatically reinitializing pi-links to zero weights, this also means that no hidden nodes can be further recruited even if the current configuration is not adequate for the training set. A positively biased factor is added to the "improvement" measure to alleviate this problem. As a hidden unit accumulates its potential improvement across training patterns, negative improvements are down weighted by a multiplier between 1 and 0. The multiplier can be adjusted according to some semi linear function of the elapsed time (number of iterations) from last recruiting. Such mechanism ensures that hidden nodes will be recruited when needed. It also helps back-propagation breaking out of local minimum since the recruiting is biased toward patterns of higher error.

A hidden node can only be recruited once. The aforementioned "recruited" flag is used for this purpose. This does not mean, however, that a hidden unit can only be involved in one particular mapping. When a new mapping is being trained, a previously trained hidden node may be activated by some subset of the current input nodes, and hence participate in current learning.

2.4 MINIMAL DISTURBANCE AND TRAINING REINFORCEMENT

The model presented here supports modularization through the use of pi-links. Given some specified input, only a subset of nodes in the network become active and participate in the current use or training. Weights of sigma-links to or from inactive nodes are never changed. CompoNet also facilitates knowledge composition. Later training employs previously learned knowledge since back-propagation from the target nodes involves not only the clamped input set, but also all visible and hidden nodes activated directly or indirectly by any subset of the input nodes. Caution must be taken, however, to ensure that previously trained relations will only be minimally disturbed by later trainings. Two issues are of particular concern. Firstly, we need to ensure that the activation of an established node cannot be altered unexpectedly due to the activation of some previously inactive nodes. A simple example is shown is figure 3. Hidden unit H is first recruited for the mapping from A to C. Later we need to train a relation from A and B to D, in which case both C and H will be activated and can be used in the next training. However, since B is now visible, C and H can take input from it and therefore may not have the right activation values.

To manage such problems, a "trained" flag is used on the links. The flag on a link from any active node to a visible node is set if the destination node is clamped as target, the flag on a link to a hidden node is set when the hidden node is just recruited. Active nodes in the network take input only from "trained" connections. Conversely, unrecruited hidden nodes collect input from all connections, which are necessar-

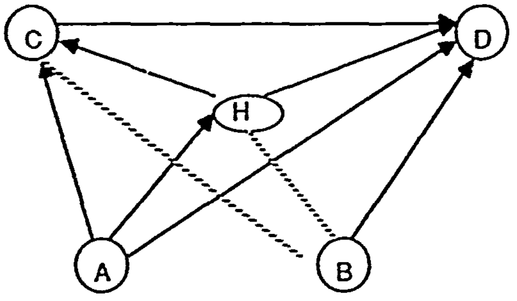


Figure 3: Example of Untrained Connections

ily "untrained". Note that activations of the pi-units change according to different input patterns, whereas the "trained" flag becomes permanent once set. The former is used to control modularization; the later is used to ensure minimal disturbance.

Actually, the "trained" flag is redundant since it corresponds to pi-links of none-zero weights. It is nonetheless included here for clarity.

A second issue of concern is whether later training can affect weights on established connections and thus distort previous mapping. In the above example, even though the connections from B to H and C are now disabled, weight associated with the link from A to H can change since the hidden unit H is involved in back-propagation (note that the direct link from A to C would not be affected since in CompoNet back-propagation does not go through visible nodes).

Disturbance to the established mapping from A to C is prevented through training reinforcement. When a free visible node (i.e., a visible node that is not clamped as either input or target) is activated, it automatically sets its own target value for back-propagation according to its current activation. For example, a free visible node with an act·vation of 0.95 would set its own target at 1. Thus while the back-propagation algorithm learns the new mapping for the target nodes, it also maintains the right activation for the free nodes.

Training reinforcement not only ensures minimum disturbance, but also further strengthens relations that are frequently used. Unfortunately, it also restrains our training pattern representation to a binary system, or at best, a system with graded activation intervals on visible nodes. With continuous representation, a visible node will not be able to memorize its own target across different training patterns without external clamping since its activation changes as learning proceeds.

2.5 MATHEMATICAL DESCRIPTION

The operation of CompoNet can be summarized with the following mathematical equations.

2.5.1 Forward Activation

Let

a_i^σ and a_i^π denote the activations of the sigma-unit and pi-unit of node i,

w_{ij}^σ and w_{ij}^π denote the weights associated with the sigma-link and the pi-link from node i to node j,

b_{ij}^σ denotes the bias associated with the sigma-link from node i to node j.

In forward activations,

$$a_j^\pi = \begin{cases} 1 & \text{if } \sum_i w_{ij}^\pi > C \\ 0 & \text{otherwise} \end{cases}$$

where C is a constant threshold.

$$a_j^\sigma = f(\sum_i u_{ij} * (b_{ij}^\sigma + w_{ij}^\sigma * a_i^\sigma) * a_i^\pi)$$

where

$f()$ denotes a logistic squashing function,

u_{ij} denotes the training flag associated with the link between node i and node j. It is set to 1 if the link has been trained, 0 otherwise.

2.5.2 Back Propagation Training

The standard back propagation algorithm is adopted here except that only active nodes and trained links are actually involved. Back propagation goes through active hidden nodes and stops at any visible nodes.

Let E denote the global measure of error

$$E = \frac{1}{2}\sum_{j \in T}(t_j - a_j^\sigma)^2$$

where

T denotes the set of indices of nodes currently clamped as target output,

t_j denotes the target activation of node j.

Define

$$\delta_j = -\frac{\partial E}{\partial net_j}$$

where net_j denotes the current sum of weighted input from incoming links to node j.

For output units

$$\delta_j = (t_j - a_j^{\sigma}) * f'(net_j)$$

For active hidden units

$$\delta_j = f'(net_j) * \sum_k u_{jk} * \delta_k * w_{jk}^{\sigma} * a_k^{\pi}$$

Adjustment of weights

$$\Delta w_{ij}^{\sigma} = \eta * \delta_j * a_i^{\sigma} * a_i^{\pi}$$

Adjustment of biases

$$\Delta b_{ij}^{\sigma} = \eta * \delta_j * a_i^{\pi}$$

2.5.3 Recruiting

The potential activation of an unrecruited hidden node is calculated differently:

$$p_j^{\sigma} = f(\sum_{i \notin T}(b_{ij}^{\sigma} + w_{ij}^{\sigma} * a_i^{\sigma}) * a_i^{\pi})$$

Its potential contribution to the improvement of global error is

$$imp_j = \sum_{k \in T}((t_k - f(net_k))^2 - (t_k - f(net_k + p_j^{\sigma} * w_{jk}^{\sigma}))^2)$$

Adjustment of weights associated with pi-links connected from node i to an unrecruited hidden node j is

$$\Delta w_{ij}^{\pi} = \theta * \beta(n) * imp_j$$

where

θ is the pi-link learning rate,

β is the positively biased multiplier, a function of the number of training epochs since last recruitment, n.

A possible choice of β could be

$$\beta(n) = \begin{cases} 1 & \text{if } imp_j > 0 \text{ } or \text{ } n < N_{min} \\ 0 & \text{if } imp_j < 0 \text{ } and \text{ } n > N_{max} \\ \frac{N_{max} - n}{N_{max} - N_{min}} & \text{otherwise} \end{cases}$$

where N_{min} and N_{max} are selected constants.

2.6 DESIGN VARIATIONS

Components of the CompoNet architecture can be varied without affecting the basic idea of network modualarization and knowledge composition. For example, complete asymmetric connections among visible nodes are not necessary. Actual connections can be customized for different purposes without changing the algorithm. The requirement of complete bipartite connections between visible nodes and hidden nodes can be relaxed to random connections given a large set of hidden nodes, since a small number of hidden nodes between every related pair of visible nodes would be sufficient. With some slight modification to the recruiting algorithm, one can also organize the hidden nodes into multiple layers instead of the single layer proposed here.

The estimate of potential improvement used in the recruiting algorithm is a little more expensive than standard back-propagation. To reduce computation effort, an inactive hidden unit can estimate the potential improvement by simply taking the sum of gradients back-propagated from the target nodes and multiply that by its own activation. The estimation will probably be accurate enough for recruiting purpose provided that the activation is small, which can be attained by restraining the initial random weights to be within a small interval.

The restriction of discrete representation can be removed if we allow only visible nodes to be shared by different relations, in which case training reinforcement would not be necessary.

The plausibility of these variations is yet to be verified by further experiments. Many other parts of the architecture and algorithm can be modified. CompoNet is designed to have flexible network configuration, and to provide easy interface to other connectionist models.

3 SIMULATION RESULTS

A software simulator was implemented to test the performance of CompoNet. We will first present experimental results in terms of modular knowledge composition, and later discuss the specific engineering performance of the recruiting algorithm. Unless specified otherwise, results presented were obtained with the same parameters. Weights were initialized randomly according to a normal distribution with mean zero and standard deviation two. Back-propagation learning rate was set to 0.1, with a momentum of 0.9. To avoid unnecessary recruiting, a history window of global errors in the past 256 epochs was used to adjust the learning rate on the pi-links. The learning rate is suppressed to 0.001 if the global error is reduced by 0.1% across window, and boosted back to 0.2 otherwise. The positively biased multiplier described in the recruiting algorithm follows a piecewise linear function with a value of 1 at 500 epochs (since last recruitment) or less, 0 at 1000 epochs or more, and $\frac{\#epochs-500}{500}$ in between. These parameters were chosen to attain more consistent performance across different tests. They are nevertheless quite arbitrary selections and are by no means optimal.

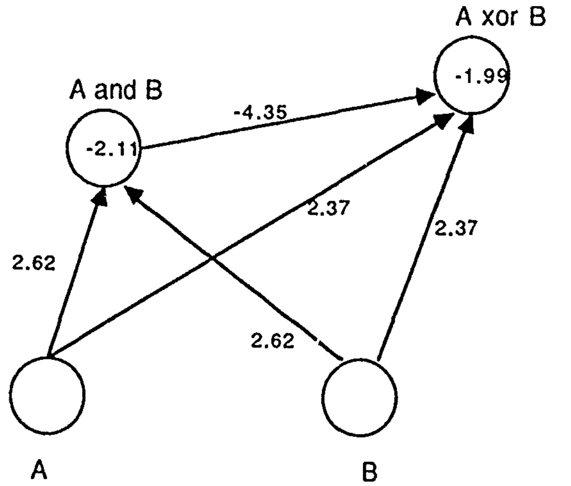


Figure 4. AND and XOR - An example of knowledge composition

A fixed configuration of sixteen visible nodes were used with the same number of hidden nodes available. For illustration purpose, simple boolean logics were used in the simulations. CompoNet, however, is not designed to model these functions only, nor is it our intention to propose a proper connectionist representation of boolean logic.

3.1 MODULAR KNOWLEDGE COMPOSITION

Previously trained knowledge can be used in later training. Such knowledge sharing can happen through the use of previously trained visible nodes, as well as established hidden nodes. In our first example, we trained an AND relation from two input visible nodes to another visible node designated as output. This was accomplished in 28 epochs. We then added an XOR relation from the same input nodes to a different output node. XOR typically can be trained in about 1000 epochs with one hidden unit recruited. In this case it used the output of AND and learned the relation in another 256 epochs. Figure 4 depicts the trained network from this example. Biases are written in the circles. Repeated trials of this experiment showed consistent performance, although the training time for the second XOR relation varied from 43 to 470 epochs.

Parity problems were used in another experiment of incremental learning. Training parity functions of four or more bits induced much difficulty on CompoNet due to the configuration of hidden nodes. This issue will be discussed further in this paper. In this experiment we first trained a 3-bit parity relation on the network, and later added in sequence 4-bit parity, 5-bit parity and 6-bit parity. Each training shared the set of input nodes used in the preceding training set, and also added one more input node for the extra bit. A different output node was used for each of the four relations. Figure 5 shows the performance of a typical training session. Mean squared errors were plotted against the number of training epochs. Each circle represents the addition of a hidden unit. For comparison purpose, we also plotted the performance of typical examples of individually trained parity problems. Note that the performance was not optimal in terms of the number of hidden nodes, as 4-bit parity recruited two hidden nodes instead of one even though it had 3-bit parity available as its input. The training time, nevertheless, was greatly reduced from that of separate trainings. Tesauro and Janessens in [8] suggested that the required training time for neural networks to learn parity functions with back propagation increases in second order polynomial with the number of input patterns. Although only very limited statistics were collected, our results suggested that with incremental learning, the increase of training time remained linear with respect to the number of input patterns. In fact, the linear order increase was not necessarily incurred by incremental learning, but probably bounded by the speed adjustment of the recruiting algorithm used to prevent unnecessary recruits.

Training reinforcement, i.e., the use of internally set target values, worked well in all experiments. No previously trained relations were ever disturbed by later trainings.

Another advantage of sharing previous knowledge is the easy generalization of some unknown new relations. Generalization in connectionist mapping refers to the capability of the algorithm to learn a particular relation without seeing the entire set of training patterns. The back-propagation learning algorithm can often generalize by effective use of the available hidden units. In addition, CompoNet is capable of a different type of generalization based on existing knowledge. To see this, we first train a relation like 4-bit parity onto the network. Later a subset of the same training patterns are presented to the same input nodes, but to a different output node. Since the first output node already carried the proper output information, a strong connection from the first output node to the second can be quickly established and no further training is needed. The new relation simply assumes the old one even though the new training set may actually be a subset of patterns for some other relations. Thus the system is biased toward existing knowledge, and always finds the easiest way to accomplish the new task.

Interestingly, such tendency to attain quick solution can sometimes have negative impact on the speed of learning. In one experiment we first trained a NOT-XOR relation and later added an OR relation from the same input nodes to a different output node. The OR relation can normally be trained easily in less than 30 epochs. With NOT-XOR pre-trained, however, it

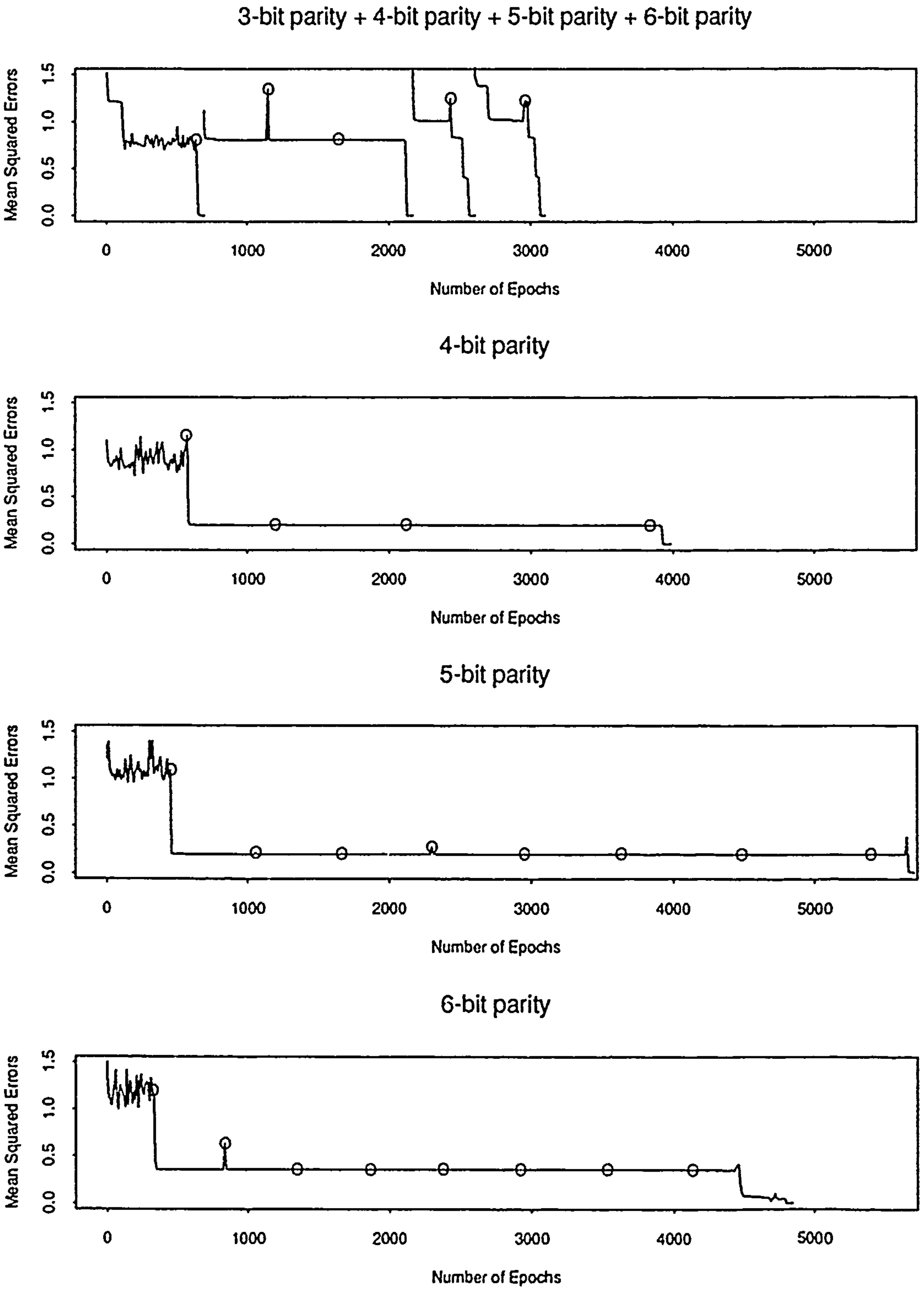


Figure 5: Incremental Learning of Parity Functions

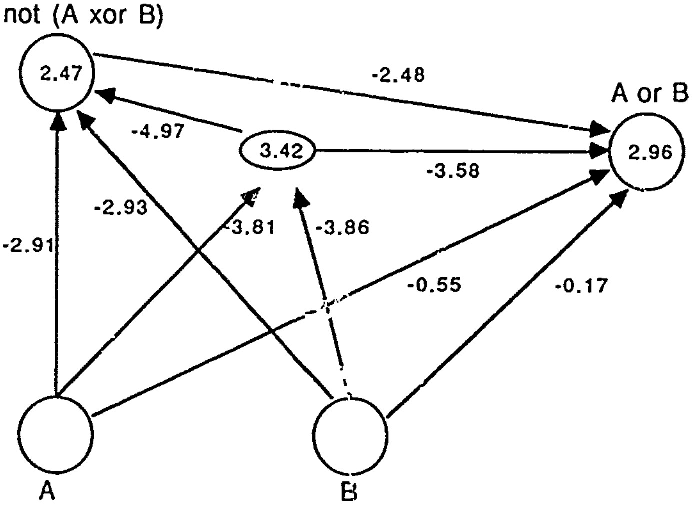


Figure 6: NOT-XOR and OR - An example of slow learning

sometimes takes many more epochs. The final network configuration of one such example is shown in figure 6. In the first training session, the hidden node recruited for NOT-XOR developed into a NAND. In the second session, the OR node first developed into an XOR and it took another 182 epochs before the right mapping was learned. The direct connections from the input nodes to the OR node were hardly used. This is mainly because after the output node had learned to behave like XOR, it became more effective to use the hidden NAND to fix the incorrect output. The correction process was slowed down considerably because the output node had values closer to the extremes of the logistic function hence had near zero derivatives with respect to the weights. The result of this experiment varies considerably depending on the meaning of the previously trained hidden node, as well as the initial random weights. There were also many cases in which the previously trained hidden node helped to speed up convergence.

3.2 PERFORMANCE OF RECRUITING

The recruiting algorithm was devised only for the purpose of hidden node allocation in a larger network. It is nonetheless interesting to study the engineering performance of such algorithm when it is used for individual problems. In particular, we are interested in the effect of recruiting on the size of network and on the speed of learning.

One major objective of research using network growing techniques is to attain minimum network size for a given problem [1]. The most obvious situation under which a new hidden node is needed seems to be when the network has settled into a local minimum. The recruiting algorithm in CompoNet relies on the growing of weights on pi-links. Even though the learning rate is adjusted according to progress of the training process, the exact time of recruiting cannot be tuned mechanically to correspond exactly with the detection of local minimum. Nevertheless, we expected that recruiting would consistently aid back-propagation to converge in reasonable time without allocating excessive number of hidden units.

To our surprise, the algorithm failed on one of the first problems tested. As we ran the 4-bit parity problem on the network, the system quickly learned fifteen out of the total sixteen training patterns after one or two recruits, but generated opposite output value for the one pattern left. The network then stayed in the same state with no further progress despite increase of hidden units. To see whether this problem was caused by the particular recruiting algorithm, we disabled recruiting and added hidden units manually one at a time, but still failed to escape from the local minimum even after adding over twenty nodes. Such problems occurred consistently over many trials. Figure 7 shows the network at an initial stage of one such example. Further study of the weights suggested that unless links connected to a recruited hidden node had very special initial weights, gradient descent could only push the network state further into the local minimum.

For comparison purposes, we deleted the direct connections from input to output and ran the experiment again. With this layered configuration, the system was able to learn 4-bit parity in less than 2000 epochs with either four or five recruited hidden nodes.

The problem was eventually solved by setting the targets at ±0.8 instead of ±1. In this setting the activations are pushed away from the extremes and thus the chance of fruitless training is greatly reduced. Parity problems of four or more bits remained difficult, but could usually be trained within reasonable time. The number of hidden nodes, however, often far exceeds the

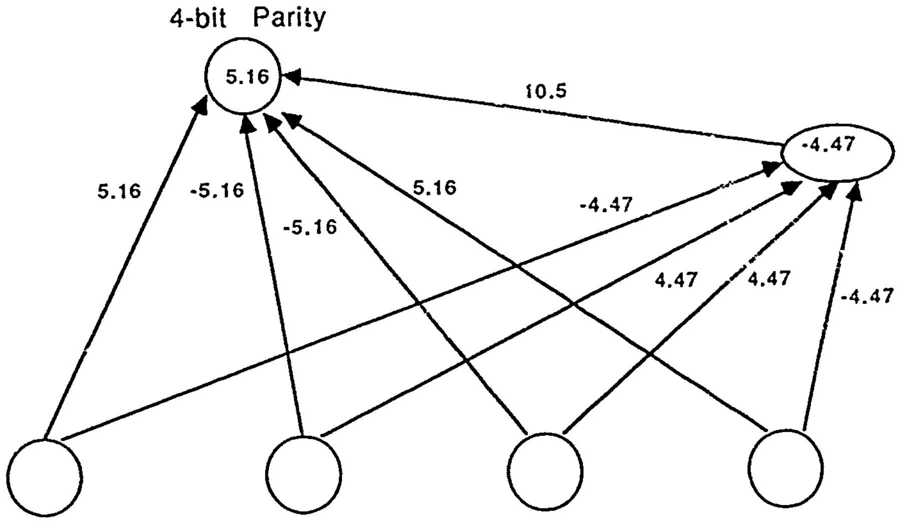


Figure 7: 4-bit Parity In A Local Minimum

minimum requirement. Typically, 4-bit parity could be trained in less than 5000 epochs, with 2 to 6 hidden nodes, 6-bit parity took from 4843 to 7371 epochs with 7 to 10 hidden nodes, 5-bit parity often was trapped in a local minimum for many epochs and took from 2045 to 19142 epochs with 4 to 11 hidden nodes. With direct connections between input and output, N-bit parity can be realized with $\lfloor N/2 \rfloor$ threshold hidden nodes.

Interestingly, the minimum solution to the parity problems with direct I/O connections is also conceptually more complicated than the same problem without I/O connections, even though the former requires fewer hidden units.

Experiments with parity problems suggested that recruiting or any other kinds of dynamic hidden node allocation method used along with back-propagation learning could not always help to determine near optimal architecture. The landscape of the weight space varies with particular problems as well as network configurations, and adding hidden nodes does not always help to escape from local minima. However, parity problems are known to be particularly difficult for back-propagation networks [5] and recruiting does seem to devise good architecture for most problems.

In terms of the speed of learning, i.e., the number of iterations/epochs needed to reach a solution, back-propagation with some scheme of dynamic node allocation compares unfavorably to that with a fixed architecture. Any recruiting algorithm based on the detection of local minima requires a lag time after each hidden node is allocated, such that the network can settle given the new configuration. This necessarily makes the algorithm more expensive computationally by an order of the number of hidden units. Simulation results confirmed such disadvantage as 4-bit parity without direct input to output connections was the only problem for which recruiting consistently performed better than fixed allocation. This is again due to the special weight space landscape of the particular problem. On most other problems, back-propagation with fixed allocation can usually converge to the right solution more rapidly, although sometimes the algorithm does get trapped in some local minimum. The local minimum problem can typically be unravelled by multiple trials or small variations on the number of hidden nodes. For example, with two fixed hidden nodes, 4-bit parity consistently converged into local minima, whereas with four fixed hidden nodes, it could typically reach the solution in about 200 epochs.

The recruiting algorithm, however, does not have to be based upon detection of local minimum. In some experiments we used a fixed fast learning rate for weights associated with the pi-links, and obtained very good results, including the only cases where we attained minimum configuration for 4-bit parity. The exact timing of recruiting played a significant role in the learning process, and early recruiting often prevented approaching local minima. The disadvantage of this approach is that it often recruited unnecessarily a few extra hidden nodes. The total number of hidden nodes allocated, however, was typically still well within the order of the number of input nodes. It is not clear whether few extra hidden nodes can have negative impact on the generalization capability of the network, or, in a larger framework like CompoNet, whether such hidden nodes will be less useful in terms of knowledge sharing.

A related question is whether recruiting can negatively affect generalization. As hidden nodes are recruited one by one, the learning algorithm is forced to attain best result given the current network configuration. Thus the learning process must follow some sequence of partitionings instead of traversing the multidimensional weight space from the beginning. This might hinder optimal partitioning, since an early move cannot be easily altered later with a local minimization algorithm such as gradient descent.

Another interesting observation on back-propagation with recruiting is that the algorithm seems to be less sensitive to its learning rate. With fixed architecture, the back-propagation learning rate needs to be well adjusted according to the size of the network as well as the number of training patterns. A large learning rate can impede close approximation to gradient descent across different patterns and often leads to oscillation. With recruiting it appears that at the same learning rate the amplitude of oscillation is greatly reduced. This is probably because the major part of the network has already settled into a relatively stable state, thus random oscillations caused by changes of weights on the new links across training patterns have less effect on the global error. Further study is needed to verify such conjecture.

4 SUMMARY

We presented CompoNet, a connectionist architecture which facilitates modular knowledge composition. The network uses multiplicative links to control modular activations, as well as to dynamically recruit hidden units. Multiple asymmetric relations between different sets of units can be loaded onto the same network at different time. Relations added later to the network can take advantage of established mappings and hidden unit representations, without disturbing previous trainings.

While the standard back-propagation learning algorithm is used in the model, both the modular activation and the recruiting mechanism are much customized to achieve best performance and to ensure robustness. Such customizations have made the model conceptually not as simple as we had hoped for. The algorithm, however, is local and well within the same order of computation as typical feedforward back-propagation networks. The design avoids the use of external knowledge of either the network structure or the training environment. Although the initial system set up is carefully tuned and tested, no further tuning on parameters such as learning rate is needed for the network to deal with different problems. Such consistency is particular important for CompoNet since its purpose is to handle multiple relations and to integrate them systematically.

Simulation performance of the CompoNet architecture satisfactorily supported its capability of knowledge composition and sharing. Previously trained relations were consistently preserved without disturbance. The model also facilitated generalization through existing knowledge. The recruiting algorithm functioned well for the purpose of hidden unit allocation. The significance of dynamic allocation with respect to the ability of learning is worth further investigation.

A serial version of the CompoNet model is currently under development. We would like to integrate both versions into one system which has the capacity to model both dynamic and static relations. We would also like to explore the design of a connectionist reasoning system based on CompoNet.

References

[1] T. Ash. Dynamic node creation in backpropagation networks. Technical Report ICS 8901, Institute of Cognitive Science, UCSD, 1989.

[2] J. A. Fodor and Z. W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. In S Pinker and J. Mehler, editors, *Connections and Symbols*. The MIT Press, Cambridge, MA, 1988.

[3] M. Furst, J. B. Saxe, and M. Sipser. Parity, circuits and the polynomial-time hierarchy. *Proceeding, 22nd IEEE Symposium on Foundations of Computer Science*, pages 260–270, 1981.

[4] K. Hornik, M. Stichcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5).359 366, 1989.

[5] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group. *Parallel Distributed Processing*, volume 1. The MIT Press, 1986.

[6] L. Shastri and V. Ajjanagadde. A connectionist system for rule based reasoning with multi-place predicates and variables. Technical Report MS-CIS-8905, Department of Computer Science, University of Pennsylvania, 1989.

[7] P. Smolensky. Proper treatment of connectionism. *Behavioral and Brain Sciences*, 11(1), 1988.

[8] G. Tesauro and B. Janssens. Scaling relationships in back-propagation learning. *Complex Systems*, 2:39–44, 1988.

[9] D. S. Touretzky, D. W. Wheeler, and G. E. Elvgren. Rules and maps II. Recent progress in connectionist symbol processing. Technical Report CS-90-112, Computer Science Department, Carnegie-Mellon University, 1990.

Part VII

Cognitive Modeling and Symbol Processing

From Rote Learning to System Building: Acquiring Verb Morphology in Children and Connectionist Nets

Kim Plunkett[*]
Institute of Psychology
University of Aarhus, Denmark

Virginia Marchman
Center for Research in Language
University of California, San Diego

Steen Ladegaard Knudsen
Dept. of Computer Science
University of Aarhus, Denmark

Abstract

The traditional account of the acquisition of English verb morphology supposes that a dual mechanism architecture underlies the transition from early rote learning processes (in which past tense forms of verbs are correctly produced) to the systematic treatment of verbs (in which irregular verbs are prone to error). A connectionist account supposes that this transition can occur in a single mechanism (in the form of a neural network) driven by gradual quantitative changes in the size of the training set to which the network is exposed. In this paper, a series of simulations is reported in which a multi-layered perceptron learns to map verb stems to past tense forms analogous to the mappings found in the English past tense system. By expanding the training set in a gradual, incremental fashion and evaluating network performance on both trained and novel verbs at successive points in learning, we demonstrate that the network undergoes reorganizations that result in a shift from a mode of rote learning to a systematic treatment of verbs. Furthermore, we show that this reorganizational transition is contingent upon a critical mass in the training set and is sensitive to the phonological sub-regularities characterizing the irregular verbs. The optimal levels of performance achieved in this series of simulations compared to previous work derives from the incremental training procedures exploited in the current simulations. The pattern of errors observed are compared to those of children acquiring the English past tense, as well as children's performance on experimental studies with nonsense verbs. Incremental learning procedures are discussed in light of theories of cognitive development. It is concluded that a connectionist approach offers a viable alternative account of the acquisition of English verb morphology, given the current state of empirical evidence relating to processes of acquisition in young children.

1 INTRODUCTION

Several accounts of the development of English inflectional morphology are couched in terms of a three-phase U-shaped pattern of acquisition. These accounts derive primarily from analyses of naturalistic production data which indicate that early in development, children produce the correct forms of English past tense or plural irregular forms, such as *went* or *sheep*. Later, these forms are incorrectly inflected and errors occur, e.g. *goed* or *sheeps*. Finally, the tendency to make such errors decreases, as some forms are identified as exceptions to the predominant pattern in the inflectional system.

The production of these overgeneralization errors has been interpreted to indicate that learning a language primarily involves the acquisition of *rule systems*, i.e., explicitly representable generalizations about linguistic regularities which allow the productive generation of forms that are not (or have not yet been) encountered in the input. The three stages in U-shaped development have each been interpreted as manifesting the application of different *mechanisms* or *strategies* for forming past tense or plural forms, each representing different modes or periods within the course of rule acquisition. During the first stage, a *rote* learning mechanism stores all forms, both regular and irregular, as independent items in a mental lexicon. During this period, both irregular and regular forms are correctly produced. However, systematic patterns which

[*] A substantial proportion of this work was carried out while the first author was a visitor at the *Center for Research in Language, University of California, San Diego* supported by a grant from the MacArthur Foundation. We would like to thank the PDPNLP discussion group for comments on this work, and especially Jeff Elman for making computing and office facilities available.

might characterize the input are not generalized to novel forms encountered by the child. The second stage of acquisition reflects the child's identification of patterns of regularity, represented in such terms as "add /-ed/" to form the past tense of a verb or "add /-s/" to form the plural of a noun. Overly general application of such rules results in the production of incorrect forms like *goed* and *sheeps*, as well as the tendency to regularize nonsense forms such as *nibbed*. Finally, after prolonged exposure to their native language, children pass into a third stage which involves discovering the exceptions to the rules, isolating these forms as independent entries in a mental lexicon. This final phase thus supposes the existence of two distinct mechanisms underlying children's ultimate knowledge of the English inflectional morphological system. One mechanism controls the default application of a general rule, responsible for the generativity of the regular paradigm in a given inflectional system. The second mechanism identifies exceptions and prompts the child to consult a separate knowledge store in the production and comprehension of irregular forms. This second mechanism is typically assumed to be closely related to, if not identical with, the mechanism underlying the rote learning which characterizes the first stage of acquisition.

In general, then, traditional accounts can be seen to attribute the onset of the production of erroneous forms to the transition from a stage in which production is determined primarily by rote-governed processes to a stage of rule construction and refinement, i.e., *system building*. The triggering of this transition in the child is not well understood (that is, what are the necessary and/or sufficient circumstances under which the transition occurs?), although a requisite amount of linguistic experience is typically assumed (Karmiloff-Smith [1986]). With respect to the English past tense, the transition into system building is at least partially dependent on sufficient exposure to suffixed past tense forms in order for the systematicities which define the regular rule to be extracted. Maturational factors which control the emergence of an inflectional system building device might also determine a U-shaped profile of development (e.g., Bever [1982]). However, maturational factors must be interpreted, at least to some degree, in interaction with input factors in order to account for observed time lags in the onset of productive behavior in different linguistic domains, e.g., the relatively early acquisition of the English plural system and the typically late acquisition of the past tense system (Brown [1973]; de Villiers & de Villiers [1985])[1]. Other explanations of the time lag between the acquisition of these two major inflectional systems in English incorporate children's developing conceptual understanding of time and number (e.g., Carey [1982]), in interaction with the character (e.g., transparency of form-function mappings) of the inflectional system of the language to be acquired (e.g., Slobin [1985]). Hence, while details about the timing and nature of the transition are still under debate, rule-based accounts generally assume that U-shape profiles of development are best explained via *dual mechanism architectures*, in which one mechanism is responsible for rote learning processes, and hence ultimately for the representation of the exceptions to the regular paradigm. A second mechanism is responsible for extracting the rule (or rules) which characterize the regular paradigm, and for constructing the underlying system guiding productive application and rule-governed language usage.

In an attempt to evaluate this account, we have argued elsewhere (Plunkett & Marchman [In press]) that it is important to distinguish between *macro* and *micro* patterns of the onset of errors when characterizing children's acquisition of the English past tense. Macro U-shaped development refers to a rapid and sudden transition into the second phase of system building, resulting in indiscriminate application of the "add /-ed/" rule to whole classes or categories of verbs. In contrast, a micro U-shaped developmental pattern is characterized by *selective* suffixation of English irregular verbs, and results in a period of development in which some irregular verbs are treated as though they belong to the regular paradigm while others are still produced correctly. The basis for selective application of the suffix may be defined with respect to certain representational characteristics of the verb stem (phonological, semantic or otherwise), or may result from the operation of a probabilistic device which determines the likelihood that the suffix will be applied to an irregular verb.

In reviewing sources of evidence regarding the patterns of overgeneralization errors in children, Plunkett & Marchman [In press] conclude that a macro characterization of past tense acquisition is inaccurate. For example, there appears to be little evidence that children overgeneralize the /-ed/ suffix indiscriminately, i.e., to all irregular verbs in their current vocabularies. Nor is there evidence to suggest the existence of a single well-defined stage of development in which erroneous behavior is observed (see also Derwing & Baker [1986]). Rather, children are likely to overgeneralize the suffix to only some irregular verbs (typically a small number) while, at the same time, they correctly produce the past tense forms of other irregular verbs. Furthermore, errors may occur across a protracted period with some irregular verbs recovering from erroneous treatment only to be overgeneralized again at a later point in development. In general, then, the evidence suggests that the transition from the first to the second phases is best characterized as a micro phenomenon, in which the onset of overgeneralization errors is both selective and gradual. An understanding of U-shaped past tense acquisition thus requires an account of:

[1] Unless we assume that different devices for controlling the acquisition of the English past tense and plural systems emerge at different points in development.

1. The mechanisms that trigger the transition from rote learning to system building.
2. The basis for the selective overgeneralization errors produced by children (featural or probabilistic).
3. The mechanism(s) by which overgeneralizations are eventually eliminated and appropriate performance on both regular and irregular verbs is ultimately achieved.

Two other factors further complicate an adequate account of children's acquisition of the English past tense. First, children's overgeneralizations do not always appear to result from the overapplication of an "add /-ed/" rule. For example, children produce stem and past tense forms like *eat* → *ated, sit* → *sit* or *pick* → *pack* (Marchman [1988]). In many respects, these *irregularizations* can be seen to be analogues to the mappings between stem and past tense of English irregular verbs (Bybee [1988]; MacWhinney [1987]). Second, as Pinker & Prince [1988] have noted, the set of irregular verbs in English (approximately 150 altogether) is not an unstructured list. Rather, it consists of a number of subregularities between the phonological form of the irregular stem and the type of transformation which relates the stem to its past tense form. For example, *all* English irregular verbs which have identical stem and past tense forms possess a stem final dental consonant (e.g., *hit* → *hit*). Similarly, verbs which undergo vowel suppletion tend to group into clusters of stem final vowel-consonant sequences, which form "family resemblance" patterns (e.g., *sleep* → *slept, weep* → *wept, keep* → *kept*).

The documented patterns of subregularity among English irregular verbs can be seen to predict the frequency of both overregularizations and irregularizations in children (Bybee & Slobin [1982]) and adults (Bybee & Moder [1983]). For example, Bybee & Slobin [1982] note that at certain periods in development, children tend to resist regularization of verb stems that already possess a dental final consonant, i.e., use the identical form in the past tense. Similarly, Derwing & Baker [1986], Marchman [1984] and Marchman [1988] note that vowel change errors on regular verbs often reflect the patterns of vowel suppletion which characterize the various clusters of irregular verbs. These findings suggest that children abstract and utilize more than one category of systematicity from among the corpus of verb stem/past tense pairs to which they are exposed. Given that children's early vocabularies contain a relatively large proportion of irregular verbs, it is not surprising that they may be misled into postulating alternative hypotheses of past tense formation based on the subregularities observed among the irregular verbs. A range of hypotheses for past tense formation may guide children's productions, and hence result in productive output which consists of both regularization and irregularization errors. However, it is surprising that irregularization errors tend to persist across development (Bybee & Slobin [1982]; Kuczaj [1977]), even past the point in lexical acquisition when the number of regular verbs greatly outnumbers irregular verbs. Thus, the tendency to irregularize may be just as *robust* as the tendency to regularize, even though suffixed forms may be more prevalent in children's productions when evaluated in terms of the number of regular verbs that are correctly inflected or the number of irregular verbs that are regularized. Note also that irregularization processes do not reflect the indiscriminate application of rule. Not all verbs that end in a dental are subject to a "no change" irregularization nor are all verbs that possess particular vowel/consonant sequences subject to vowel suppletion. Both processes of regularization and irregularization generate micro, as opposed to macro, profiles of U-shaped development. While these facts do not contradict the dual mechanism hypothesis, the timing and nature of the interaction between the two mechanisms requires further explication in order to account for these patterns of errors and recovery.

Recently, Rumelhart & McClelland [1986] have argued that a *single mechanism system* in the form of an *artificial neural network* is capable of extracting a range of regularities that characterize the English past tense system and producing patterns of regularization and irregularization analogous to the errors observed in children. Furthermore, this work has been interpreted as the first in a series of challenges to the widely accepted view that linguistic behavior should be understood in terms of *explicitly representable* rules and principles, and a separate store of exceptions to those rules. As a result, Rumelhart & McClelland [1986] has been subject to thorough analysis and criticism aimed both at the details of their model and at the applicability of artificial neural networks for models of language acquisition and processing. For our purposes, the crucial feature of Rumelhart & McClelland's [1986] work is the claim that a single mechanism is responsible for U-shaped acquisition and the representation of both the regularities and irregularities underlying the English past tense system. Here, the transition from rote learning to system building does not rely on a dual mechanism architecture to capture the distinction between regular and exceptional patterns. Rather, the model exploits the capacity of connectionist networks to simultaneously:

1. *Memorize* individual patterns and their transformations when the number of pattern types is sufficiently small.
2. *Generalize* on the basis of regularities observed in the input when the number of patterns is sufficiently large.

Rumelhart & McClelland [1986] initially trained their network on a subset of the vocabulary to which it would eventually be exposed. During the first 10 epochs of training only 10 verbs (8 of which were irregular) were presented to the network. Given the learning and representational characteristics of their network architecture (a single-layered perceptron), the model succeeded in learning the 10 verbs by "rote", i.e., without discovering any regularities among the individual verbs in the training set which were then generalized to new verbs or which interfered with the successful mapping of other verbs in the training set. After 10 epochs of training, Rumelhart & McClelland [1986] increased the size of the training set by 410 verbs. Most of new verbs were regulars. This sudden expansion in vocabulary size caused the learning algorithm (a probabilistic version of the Perceptron Convergence Procedure (Rosenblatt [1962])) to extract the "add /-ed/" regularity and to reorganize the mapping characteristics to reflect the dominant suffixation process. As a result, many irregular verbs displayed a sudden decrement in performance. Continued training on the input set allowed the network to gradually recover from the erroneous mappings. Analyses of the network's performance reveals that much of the success in modeling the classical U-shaped profile of development derives from the exploitation of the transition from item memorization to generalization that is inherent in these networks when the *number* and *structure* of mapping patterns is manipulated.

Pinker & Prince [1988] have pointed out that the discontinuities introduced into the training regime by Rumelhart & McClelland [1986] do not reflect plausible discontinuities in the input to children learning the past tense. First, they argue, there is scant evidence for such an abrupt increase in the total number of verbs to which children are exposed. Second, the evidence from children's productions (Brown [1973]) suggests that the actual relative proportions of regular and irregular verbs are less skewed than those represented in the Rumelhart & McClelland [1986] training set. For example, Pinker & Prince [1988] suggest that during early phases of acquisition, regular and irregular verbs are approximately evenly represented in children's production vocabularies. In general, current consensus has targeted the implausibility of the discontinuities in the original simulations, and hence the theoretical significance of the U-shaped learning demonstrated by the Rumelhart & McClelland [1986] model has been undermined.

However, Plunkett & Marchman [In press] and Plunkett & Marchman [1989] have demonstrated that several characteristics of micro U-shaped development can emerge in an artificial neural network which maps verb stems to past tense forms *in the absence of any discontinuities in the training regime*. Using a constant vocabulary size and structure throughout learning, several series of simulations were used to illustrate the conditions under which decrements in performance and subsequent recovery on individual verbs are observed. In all cases, the patterns of errors observed were attributed to the *competition* between the different types of mappings used in the simulations (arbitrary, suffixation, identity, vowel change), which are typical of the relationship between verb stem/past tense pairs in English. This work also showed that the capacity of these types of networks to learn inflectional verb morphology is highly sensitive to input parameters such as the *type* and *token frequency* of stems in the input set. For example, arbitrary mappings (*go* → *went*) tend to be mastered when they are few in number (low type frequency) and when each unique stem is repeated frequently (high token frequency). Interestingly, the frequency parameters that support realistic verb learning in artificial neural networks tend to converge on frequency typologies associated with natural languages. Plunkett & Marchman [In press] also showed that the types of errors observed in artificial neural networks are constrained by the phonological characteristics of the distinct verb mapping classes. Thus, some regular verbs which end in a dental are identity mapped, while, at the same time, regular verbs with stem final vowel-consonant pairs that are characteristic of the vowel change class are subject to irregularization.

In summary, Plunkett & Marchman [In press] observe a range of errors in their simulations which can be documented in the child language literature and which comprise the selective patterns of micro U-shaped development observed in children acquiring the English past tense. These results are achieved, not by introducing discontinuities into the training set, but by manipulating type and token frequency parameters in ways which reflect the characteristics of verbs in English. The errors characteristic of micro U-shaped development are thus shown to be a natural outcome of learning in a network required to map competing classes of verbs which vary in systematic, but not absolute ways with respect to mapping type, type and token frequency, and phonological predictability.

In attempting to demonstrate the ability of artificial neural networks to solve the mapping of competing verb classes in the absence of discontinuous input, Plunkett & Marchman [In press] deliberately held vocabulary size constant throughout training, i.e., at 500 verbs. Although the type and token frequency parameters used in these simulations were characteristic of English, it is unlikely that children attempt to learn an entire lexicon all of a piece at any point in learning. Naturalistic production and comprehension measures suggest that verb acquisition in children is an incremental (albeit non-linear) learning process. Because the *size* of the vocabulary used by Plunkett & Marchman [In press] precluded the network from achieving complete mastery of the vocabulary early in training, the marked transition from an initial overall high performance to a performance decrement that was achieved in the original Rumelhart & McClelland

[1986] model was not observed in these simulations. Indeed, one of the primary goals of the Plunkett & Marchman [In press] work was to demonstrate that an abrupt *transition* from conditions of rote learning to conditions of system building is *not necessary* for the emergence of systematic regularization and irregularization errors and the subsequent recovery from those errors. Thus, it is important to distinguish these findings from the observation that children pass from a period in which the past tense of all verbs are produced correctly, to a period in which regularization and irregularization errors are observed.

As noted above, the determinants of the transition from rote learning to system building in children are not well understood. The traditional view supposes that a *dual-mechanism* system is required to support the transition. An alternative, suggested by the work of Rumelhart & McClelland [1986], is that the transition is affected by *quantitative* and *structural* changes in the vocabulary of verbs that a *single mechanism* is required to learn. A sufficient evaluation of the single mechanism approach is hindered by the unrealistically abrupt vocabulary discontinuity that was introduced into Rumelhart and McClelland's training set. While they clearly demonstrate that a single mechanism can perform both rote learning and generalization, as well as make a transition between the two, further evaluation of learning in a single mechanism architecture under realistic learning conditions has so far been obscured. For example, how many verbs does the network need to experience before it attempts to extract regularities from the input and organize this knowledge base in a more systematic fashion? Is this event sudden or gradual? To what extent does the systematization of the current vocabulary impact on performance with verbs not yet acquired by the network? Are there characteristic patterns of errors associated with the transition from rote learning to system building? Does an early rote state of the knowledge base affect later patterns of acquisition? How do the developmental profiles observed in a network making the transition from rote learning to system building correspond to profiles of development in children acquiring the English past tense? How does such a single mechanism approach compare with the traditional dual mechanism architecture in accounting for the data?

In the remainder of this paper, we explore the performance of an artificial neural network required to learn mappings analogous to the English past tense when vocabulary size is expanded gradually, in an incremental fashion, across the course of learning. Our goal in this work is to determine the conditions under which an artificial neural network makes a transition from a stage of rote learning to a period of system building and to evaluate the characteristics of its performance in this period of transition and beyond. By comparing the characteristics of performance in the network with data from English children acquiring the past tense, we hope to demonstrate that artificial neural networks offer an alternative explanatory framework within which to understand the mechanisms underlying children's acquisition of inflectional verb morphology.

2 METHOD

2.1 OVERVIEW

All simulations involve training a multi-layered perceptron to map phonologically represented verb stems to their corresponding past tense forms. After initial training on a subset of 20 verb stems, the vocabulary is gradually expanded until it reaches a size of 500 verbs. Vocabulary expansion is performed following two types of training schedules, criterial expansion and incremental expansion. Several conditions of incremental learning are tested (see below). Learning in all simulations is evaluated at regular intervals during training in the following ways. First, the network's performance on verb stems in the current training set is evaluated in terms of the percentage of forms output correctly. For those forms that are produced incorrectly, errors are coded and categorized. Second, at every evaluation point in training, the network is tested on a set of novel verbs. In these cases, the mapping performed by the network on each novel stem is categorized in relation to the mapping characteristics of the training set.

A variety of control simulations have also been conducted to evaluate the role of various input and learning parameters on observed performance and learning effects. The details of these controls will be reported in the results section where appropriate. Each set of simulations that is reported here uses identical vocabularies and identical initial starting weights. However, the results of *all* sets of simulations have been replicated using different vocabularies and starting conditions.

2.2 VOCABULARY

A vocabulary of 500 verb stems is constructed from a dictionary of approximately 1000 stems. Each verb in the dictionary consists of a Consonant-Vowel-Consonant (CVC) string, a CCV string or a VCC string. Each string is phonologically well-formed, even though it may not correspond to an actual English word. Each vowel and consonant is represented by a set of phonological contrasts, such as voiced/unvoiced, front/center/back[2]. Table 1 summarizes the phonological representations for all consonants and vowels used in the simulations. Verb stems are assigned to one of four classes. Each class corresponds to a different type of transformation analogous to classes of

[2] See Plunkett & Marchman [In press] for a more thorough discussion of the phonological representation used here.

Table 1: Phonological representation

	PHONOLOGICAL FEATURE UNITS					
	CON/VOW	VOICING	MANNER		PLACE	
	#1	#2	#3	#4	#5	#6
/b/	0	1	1	1	1	1
/p/	0	0	1	1	1	1
/d/	0	1	1	1	1	0
/t/	0	0	1	1	1	0
/g/	0	1	1	1	0	0
/k/	0	0	1	1	0	0
/v/	0	1	1	0	1	1
/f/	0	0	1	0	1	1
/m/	0	1	0	0	1	1
/n/	0	1	0	0	1	0
/η/	0	1	0	0	0	0
/δ/	0	0	1	0	1	0
/θ/	0	1	1	0	1	0
/z/	0	1	1	0	0	1
/s/	0	0	1	0	0	1
/w/	0	1	0	1	1	1
/l/	0	1	0	1	1	0
/r/	0	1	0	1	0	1
/y/	0	1	0	1	0	0
/h/	0	0	0	1	0	0
/i/ (eat)	1	1	1	1	1	1
/I/ (bit)	1	1	0	0	1	1
/o/ (boat)	1	1	1	0	1	1
/^/ (but)	1	1	0	1	1	1
/u/ (boot)	1	1	1	1	0	1
/U/ (book)	1	1	0	0	0	1
/e/ (bait)	1	1	1	1	1	0
/ε/ (bet)	1	1	0	0	1	0
/ai/ (bite)	1	1	1	0	0	0
/æ/ (bat)	1	1	0	1	0	0
/au/ (cow)	1	1	1	1	0	0
/O/ (or)	1	1	0	0	0	0

past tense formation in English. The four classes of transformation are[3]:

Arbitrary mappings: There is no apparent relationship between the stem and its past tense form, e.g., '*go* → *went*'.

Identity Mapping: Past tense forms are identical to their corresponding verb stems. Such mappings are contingent upon the verb stem ending in a dental consonant i.e. /t/ or /d/, e.g., '*hit* → *hit*'.

Vowel Change: Certain vowels can be changed under the condition that they precede particular consonants. The following four vowel-consonant cluster changes are permitted:

1.	/iz/ → /ez/	'*fiz* → *fez*'
2.	/it/ → /εt/	'*kit* → *kεt*'
3.	/ais/ → /es/	'*lais* → *les*'
4.	/ail/ → /Ol/	'*rail* → *rOl*'

Regular mappings: A suffix is appended to the verb stem. The form of the suffix depends upon the final vowel/consonant in the stem:

1. If the stem ends in a dental (/t/ or /d/), then the suffix is /-id/, e.g., '*pat* → *pat-id*'.
2. If the stem ends in a voiced consonant or vowel, then the suffix is voiced /d/, e.g., '*dam* → *dam-d*'.
3. If the stem ending is unvoiced, then the suffix is unvoiced /t/, e.g., '*pak* → *pat-t*'.

The suffixes on the regular past tense forms are represented non-phonologically as three distinct patterns across two output units, i.e., 0 1, 1 0, and 1 1. A fourth pattern (0 0) corresponds to the absence of a suffix, as is the case for stems in the irregular classes (i.e., arbitrary, identity and vowel change).

Stems are assigned randomly from the dictionary to each of the four classes, with the constraint that stems possess the appropriate characteristics of a given class. The resulting 500 verb vocabulary contains 2 stems in the arbitrary class, 458 stems in the regular class, 20 stems in the identity class and 20 stems in the vowel change class. Each of the four vowel-consonant clusters defining the vowel change class contains 5 members. Stem assignment to the arbitrary and regular classes are not contingent upon any particular criteria, and these classes may contain stems which have phonological characteristics of identity mapping or vowel change stems. The number of stem assigned to each verb class, i.e., the vocabulary configuration, is based on results from Plunkett & Marchman [1989] in which a wide range of verb frequencies were evaluated for their learning consequences, in light of their similarity to supposed vocabulary configurations of English. The current choice parallels a configuration which demonstrated a high level of learning in the earlier work.

Appropriate past tense forms are constructed for each vocabulary item in each of the four classes. In the case of stems in the arbitrary class, a past tense form is chosen that does not share any consonants or vowels with the stem, nor corresponds to the stem or past tense form of any other verb in the training set.

2.3 TRAINING SCHEDULE

After 500 verbs have been assigned to the four class types, a subset of 20 verbs is randomly selected from the vocabulary for use in the initial phase of training. The initial training set is comprised of 2 arbitrary stems, 10 regular stems, 4 identity stems and 4 vowel change stems. The token frequencies (i.e. the frequency with which any given stem is likely to be repeated during a single training epoch) for this initial phase of learning are 15 for the arbitrary stems, while regular, identity and vowel change stems have a token frequency of 5[4]. The relatively high token fre-

[3] A more fine-grained classification of the past tense of English is provided by Bybee & Slobin [1982], Pinker & Prince [1988]. However, the current four way distinction serves to capture many of the phenomena of interest.

[4] Pilot simulations indicated that the network would fail to learn all of the regular stems if a token frequency of less than 3 was used.

quency of the arbitrary stems is based on results from Plunkett & Marchman [1989] in which arbitrary mappings are only acquired under similar token frequency conditions.

The network is trained on this small vocabulary until all verb stems are mapped to their appropriate past tense forms. The vocabulary is then gradually expanded in size. Two general types of expansion schedules have been tested. On the first schedule, *criterial expansion*, the vocabulary is expanded one verb stem at a time and trained until the new verb is successfully mapped by the network. At this point, a new verb is introduced, training continues until that verb is mapped correctly, another verb is added and trained until it is mapped correctly, and so on. On the second training schedule, *epoch expansion*, a new verb is introduced to the vocabulary and trained for a set number of epochs. Another verb is then introduced into the training set *irrespective of the level of performance on the mapping of the previously introduced new verb*. This process is repeated until the vocabulary has reached 500 verbs. In the current set of simulations, incremental schedules of 1, 2, 5, 10 and 25 epochs per new verb have been evaluated. In all epoch expansion simulations, training is reduced to 1 epoch per verb after 80 new verbs have been introduced (i.e., the total vocabulary has reached 100 verbs).

The order in which new verbs enter the vocabulary is determined by a weighted random selection process which is based on an 80% likelihood that the new verb is taken from the regular class and a 10% likelihood that the verb is taken from the identity or vowel change classes. Each new verb entered into the training set after the initial set of 20 are assigned a token frequency of 3, until the vocabulary size reaches a total of 100 verbs. Thereafter, verbs that are introduced (predominantly regulars) are trained using a token frequency of 1. This frequency profile is chosen to accommodate the data set to the observation that children are more likely to hear, and thus have a greater opportunity to learn, verbs with a high token frequency. A summary of the changing structure of the vocabulary is provided in Table 2.

2.4 NOVEL VERBS

A further 100 stems were selected from the dictionary for testing the generalization properties of the network. Of the 100 novel verb stems, 10 end in a dental final consonant (/t/ or /d/), 10 stems possess the characteristics of each of the 4 clusters defining the vowel change class (40 verbs), and 50 stems are legal stems but possess none of the previously mentioned characteristics. These sub-classes of novel stems permit evaluations of the manner in which the network has tuned its response characteristics to stems which are candidate (though not definitive) members of the various implicit stem classes making up the training set. The network's performance on the novel verbs is evaluated at regular intervals during training.

Table 2: Vocabulary Structure by Verb Class

TOTAL	ARBS	REGS	IDS	VCS	TOK
20	2	10	4	4	5[a]
80	2	61	8	9	3
100	2	77	10	11	1
140	2	112	11	15	1
200	2	163	15	20	1
260	2	221	17	20	1
269	2	227	20	20	1
380	2	338	20	20	1
500	2	458	20	20	1

[a] Arbitrary verbs have a token frequency of 15.

2.5 NETWORK CONFIGURATION

All the simulations are run using the RLEARN simulator (Center for Research in Language, UCSD) using a back propagation learning algorithm. Back propagation involves the adjustment of weighted connections and unit biases when a discrepancy is detected between the actual output of the network and the desired output specified in a teacher signal. In multi-layered perceptrons (containing hidden units), error is assigned to non-output units in proportion to the weighted sum of the errors computed on the output layer.

All networks contain 18 input units, 30 hidden units and 20 output units. All layers in the network are fully connected in a strictly feed-forward fashion. Previous work (Plunkett & Marchman [In press]) demonstrates that a multi-layered perceptron is required to solve mapping problems of the type encountered here. There is no generally acknowledged criterion for selecting appropriate numbers of hidden units for an arbitrary problem. The modeler must, therefore, experiment with network capacities in order to find a configuration suited to the problem. The final choice of 30 hidden units reflects a compromise between the attempt to achieve an optimal level of performance and the aim to maximize the generalization properties of the network. Minimizing the number of hidden units in a network encourages the system to search for regularities in the input stimuli.

Training in the simulations follows a pattern update schedule, i.e., a pattern is presented to the net, a signal propagates through the net, the error is calculated, and the weights are adjusted. Pattern update is preferred to batch update (in which error signals are averaged over a range of input patterns before the weights are adjusted) for this problem since children are unlikely to monitor an average error on their output, but are more likely to monitor the error associated with individual pattern tokens. Learning rate and momentum are held constant throughout the simulation at values of 0.1 and 0.0, respectively. (As with the choice of network configuration, learning rate and momentum parameters are typically determined through experimentation rather than principled criteria). Verb stems are presented randomly to the network within each epoch of trainıng.

2.6 OUTPUT ANALYSIS

On each presentation of an input pattern, any error on the output units is recorded and the weights adjusted accordingly. The weight matrix for the network is saved at regular intervals; first, when the net has just mastered the initial 20 verbs and then each time a new verb is introduced but before any training on the new verb has occurred. These weight matrices provide snapshots to evaluate the accuracy of the network in producing the correct past tense form for each unique stem at different points in the network's development. The output of the network is evaluated in terms of the "closest fit" (in Euclidean space) to the set of phonemes that map the output space, defined by the teacher signal to the network (see Table 1). For each class of stems, error analyses provide an overall hit rate (i.e. % correct) and error types are classified by verb class to determine the proportion of stems that are incorrectly mapped as identity stems, vowel change stems, blends, etc. Similarly, ncvel verb stems are tested on the saved weight matrices. The different categories of novel verbs are analyzed separately to determine their output tendencies, i.e. whether they are regularized, irregularized or handled in some other fashion by the net.

3 RESULTS

3.1 CRITERIAI TRAINING

Under the criterial expansion condition, vocabulary size is increased one verb at a time and training is continued on each new verb (as well as the initial set) until that new verb is successfully mapped by the network. Typically, training on the initial set of 20 verbs requires 15 to 40 epochs to reach criterion, depending on the initial configuration of random weights. Performance on subsequent training is also sensitive to the initial set of random weights. Several initial configurations were tested. However, in none of the criterial learning simulations was it possible to continue vocabulary expansion beyond approximately 27 verb stems. Training continued for a considerable number of epochs (up to 1000) or until it was clear that the error gradient had reached asymptote at a non-zero level.

The inability of networks in the criterial expansion condition to learn a large number of verb stem/past tense mappings reflects the propensity of networks of this type to be caught in "local minima". In order for networks to avoid entrenchment in specific areas of weight space, training must ensure that a variety of weight changes occur. If the network is repeatedly trained on a limited and fixed number of patterns, where a series of similar weight changes occur, further training may fail to promote necessary reorganizations or may even enhance the network's entrenchment in a particular region in weight space. This training schedule was therefore abandoned as a method of vocabulary expansion that is appropriate to the current task.

The following sections report on results from simulations in which verbs are added to the vocabulary *irrespective* of the level of performance of the network on the previously added verb.

3.2 EPOCH EXPANSION

3.2.1 Overall Performance

Figure 1 summarizes the hit rates (%) on stems in the regular and irregular classes (arbitrary, identity and vowel change combined) as a function of vocabulary size in simulations which use the following expansions schedules: (a) 1 verb per epoch, (b) 1 verb every 5 epochs, and (c) 1 verb every 25 epochs[5].

First, note that a high level of performance is achieved on both regular and irregular verbs by the end of training (i.e., a total vocabulary size of 500 verbs) in all expansion schedules. This consistently high level of performance contrasts with that reported in earlier work (Plunkett & Marchman [In press]) in which performance on similar vocabulary configurations did not exceed 80% for the regular and vowel change classes. It is noteworthy that final level of performance appears to vary depending on the particular expansion schedule condition used. In particular, level of mastery on the set of regular verbs decreases as a function of the epoch increment, i.e., overall performance on regulars tends to diminish slightly as the network is trained more on each new verb before yet another new verb is introduced. For example, overall performance on the regular verbs in the 1 epoch condition reached 99%, whereas, only 95% of the regular stems were correctly mapped in the 25 epoch increment condition.

Interestingly, the relationship between expansion schedule and final level of performance on regular stems contrasts to the pattern of learning observed early in training. When vocabulary size is increased at a rapid rate (e.g., 1 epoch increment), performance on regulars does not improve as quickly and greater decrements are observed, compared to the other expansion schedules. Thus, while exposure to a constant vocabulary for a substantive period (i.e., 25 epochs) results

[5] Similar patterns of results are found for the 2 epoch and 10 epoch expansion schedules.

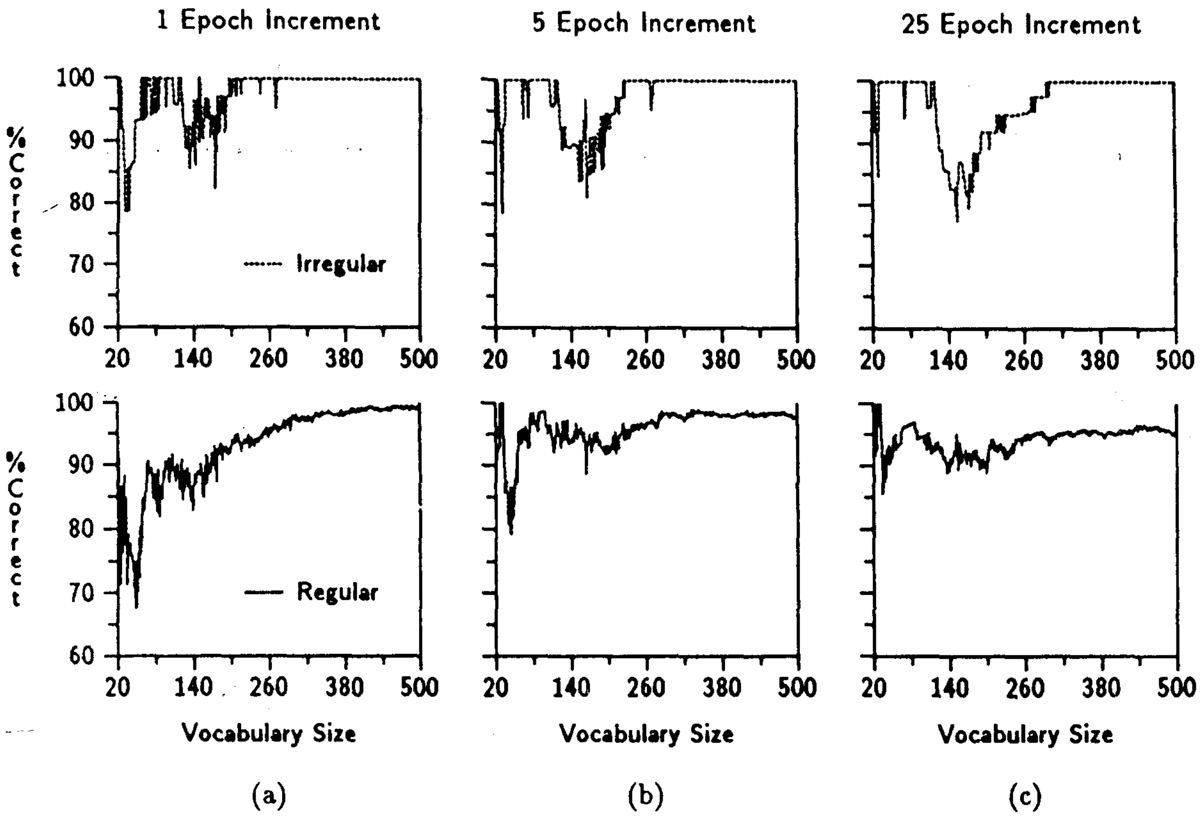


Figure 1: Hit Rates for Regular and Irregular Verbs

in a more even level of performance across training, this appears to be at the expense of overall mastery of the set of regular mappings. It is possible that this pattern of results derives from the general inability of these networks to master a large number of mappings when training follows a criterion learning schedule (see section 3.1 above). When the training set is fixed for a number of training epochs, the network may have difficulty recovering from erroneous configurations of the weight matrix. Thus, the criterion learning condition may be seen as corresponding to an indefinitely long expansion schedule.

In order to rule out the effect of initial set size on final level of performance, a control simulation was performed in which the network was trained on an initial vocabulary of 1 verb (compared to 20), and the vocabulary was expanded incrementally every two epochs to a total of 500 stems. The results from this control simulation revealed a similar level of performance on the regular verbs, achieving 100% performance for all verb classes. While more complete analyses are required, these results suggest that the size of the initial subset of the vocabulary *per se* (e.g., 20 verbs) is not a necessary determinant of the high level of performance on the regular verbs observed in these simulations.

Second, Figure 1 also reveals that, in all conditions, performance on both regular and irregular verbs suffers a marked decrement in the period immediately following the initial period of training. This is the point in training when vocabulary expansion is first initiated, and hence, the network is now faced with the task of learning a continually increasing vocabulary. Note, however, that comparisons across the expansion conditions illustrated in Figure 1 reveal that expansion schedule influences the *degree* to which performance deteriorates during this period. In particular, the higher the rate of expansion, the greater the observed decrement in performance. Thus, the 1 epoch expansion schedule results in a deterioration to 68% for regular verbs. In contrast, performance on the regular verbs in the 25 epoch expansion schedule only deteriorates to 88%. The level of deterioration in performance on the 2 epoch and 10 epoch conditions (not shown in Figure 1) round out this pattern of results at 77% and 84%, respectively. Given the dynamics of learning in these networks, this result is not surprising. When the rate of expansion is decreased and the network is trained for a greater number of epochs on each new pattern, the opportunity to accommodate the weight matrix to the new pattern is correspondingly increased. A high rate of verb introduction allows only a minimal amount of training on each new verb and hence, performance on the overall set of verbs is decreased.

More surprising, however, is that recovery from this period of initial decrement is first manifest when vocabulary size reaches around 50 verbs *irrespective of* *the expansion schedule condition*[6]. The fact that recovery occurs at approximately the same level of vocabulary size regardless of the depth of the performance decrement, and hence, training schedule, suggests that absolute vocabulary size, i.e., *critical mass*, may be an important factor in determining the network's recovery from erroneous performance.

The simulations described in Figure 1 deliberately confound the size of the training set with the number of learning trials that each verb stem/past tense pair receives. In order to investigate a "critical mass" interpretation of these results, we conducted a series of control simulations in which expansion of the vocabulary is halted at various points. Training is then continued with a constant vocabulary size in order to observe whether performance recovers in the manner observed in the test simulations. Using a 2 epoch expansion schedule, performance was evaluated in networks with final vocabulary sizes of 30, 40, 50, 60, 70 and 80 verbs. For all vocabulary sizes, the network eventually recovers from its initial erroneous performance to achieve 100% correct performance on all verb classes. Indeed, similar patterns of trajectories of recovery were observed for networks learning vocabularies of these sizes. Thus, we conclude that the size of the training set does not appear to influence the ability of the network to recover from erroneous performance on verbs in the training set, i.e., on *trained* verbs. However, the results from these control simulations will illustrate an important contrast between trained vs. novel verbs. As presented below, the final size of the training set determines the generalization properties of the network, and hence can be seen to contribute to an explanation of the network's treatment of verbs introduced later in training.

Finally, Figure 1 reveals that for all training schedules, irregular verbs undergo a substantial decrement in performance during the middle period of training, i.e., when vocabulary size reaches approximately 100 verbs. During this period in training, the number of irregular verbs increases from 23 to 37 (including an extra 5 IDS and 9 VCS). Two factors contribute to this decrement in performance. First, recall that *all* verbs introduced after the 100 vocabulary mark are introduced with a token frequency of 1. We have shown in previous work (Plunkett & Marchman [In press]) that irregular mappings (in particular, arbitrary and vowel change verbs) are difficult for the network to master in the context of other types of mappings if those stems have a low token frequency. Second, as discussed in more detail in the following section, the generalization properties of the network during this stage of training are such that at least 30% of novel verbs are treated as regulars, regardless of whether they possess phonological properties characteristic of verbs in the irregular classes.

Note, however, that performance decrements on the irregular verbs are not identical in duration and magnitude across the expansion schedules. For example, in the 1 epoch increment schedule, the deterioration in performance is not as great nor as persistent compared to that observed in the 25 epoch increment condition Further discussion of this finding and a detailed breakdown of the type of errors observed during this period of training is provided in section 3.2.2 (Error Analysis).

As mentioned above, the ability of these networks to map certain classes of verbs (especially arbitrary and vowel change verbs) is greatly facilitated by high levels of token frequency relative to the dominant mapping type (Plunkett & Marchman [In press]). It is possible, however, that the overall improved learning under the incremental expansion conditions obviates the need to construct training vocabularies in light of token frequency differences across verb classes. Therefore, a series of control simulations were conducted in which all verbs were introduced with the same token frequency (i.e., 1 token) and trained following a 2 epoch incremental expansion schedule. Briefly, the results indicate that incremental learning does not result in an improved level of performance on the arbitrary and vowel change verbs (measured by hit rate). Thus, the effects of token frequency outlined in earlier work appear to be robust across the range of training schedules tested here. These results, therefore, further justify the attempt to carefully configure training vocabularies in light of the general token frequency characteristics of English.

3.2.2 Error Analysis

All incorrectly generated mappings are analyzed in terms of the proportion of errors by verb class. We refer to these errors as *general learning errors* in subsequent discussion. (Patterns of micro U-shaped learning, i.e., errors on verbs which are successfully mastered followed by erroneous mapping and subsequent recovery are presented below.) Table 3 summarizes the categories and timing of errors, as well as the hit rate (%) for each verb class[7] for the 5 epoch expansion condition. Successive rows in the table represent expanding vocabulary levels (and increasing number of epochs). Error coding categories are:

[6] It should be stressed that decrements in performance plotted in Figure 1 do *not necessarily* indicate the U-shaped "un-learning" of individual verbs. New verbs are continually introduced into the training set and may contribute to the overall decrement in performance even though old verbs continue to be mapped appropriately (c.f., the criterial expansion schedule above). Analyses of the patterns of U-shaped learning in these simulations are discussed in section 3.2.2.

[7] An analysis of the arbitrary class is not presented here since they perform at optimal level throughout the expansion schedules.

Table 3: Errors by Class

		REGULARS				IDENTITY			VOWEL CHANGE				
VOC	EPO	HIT	SUF	ID	BLD	HIT	SUF	VC	HIT	SUF	ID	BLD	VC
20	40	100	0	0	0	100	0	0	100	0	0	0	0
30	90	90	0	0	0	100	0	0	71	0	0	0	0
40	140	80	20	0	0	100	0	0	100	0	0	0	0
50	190	85	20	20	20	100	0	0	100	0	0	0	0
60	240	95	50	50	0	100	0	0	100	0	0	0	0
70	290	94	0	33	0	100	0	0	100	0	0	0	0
80	340	98	0	100	0	100	0	0	100	0	0	0	0
90	390	97	0	100	0	100	0	0	100	0	0	0	0
100	440	97	0	50	0	100	0	0	100	0	0	0	0
110	490	94	20	20	0	90	100	0	100	0	0	0	0
120	540	94	40	20	0	100	0	0	100	0	0	0	0
130	590	97	33	66	0	100	0	0	85	0	50	50	0
140	640	93	57	28	0	100	0	0	80	0	66	33	0
150	690	95	40	20	0	100	0	0	81	0	33	0	33
160	740	93	37	37	0	91	0	100	88	0	50	0	0
170	790	94	28	42	0	84	0	50	88	50	0	0	50
180	840	95	14	57	0	92	100	0	78	0	25	25	50
190	890	92	9	54	0	92	100	0	78	0	25	25	50
200	940	92	15	53	0	93	100	0	95	0	0	100	0
210	990	94	10	60	0	93	100	0	95	0	0	100	0
220	1040	95	0	66	11	93	100	0	100	0	0	0	0
230	1090	95	11	55	0	100	0	0	100	0	0	0	0
240	1140	96	0	71	14	100	0	0	100	0	0	0	0
250	1190	96	0	62	12	100	0	0	100	0	0	0	0
260	1240	96	12	62	12	100	0	0	100	0	0	0	0
270	1290	97	0	83	0	95	100	0	100	0	0	0	0
280	1340	96	0	62	12	100	0	0	100	0	0	0	0
290	1390	98	25	75	0	100	0	0	100	0	0	0	0
300	1440	98	25	75	0	100	0	0	100	0	0	0	0
310	1490	98	20	60	0	100	0	0	100	0	0	0	0
320	1540	97	14	42	28	100	0	0	100	0	0	0	0
330	1590	98	20	60	0	100	0	0	100	0	0	0	0
340	1640	98	25	50	0	100	0	0	100	0	0	0	0
350	1690	98	25	75	0	100	0	0	100	0	0	0	0
360	1740	98	40	60	0	100	0	0	100	0	0	0	0
370	1790	98	40	40	0	100	0	0	100	0	0	0	0
380	1840	97	28	57	14	100	0	0	100	0	0	0	0
390	1890	98	16	66	16	100	0	0	100	0	0	0	0
400	1940	98	14	71	14	100	0	0	100	0	0	0	0
410	1990	98	16	66	16	100	0	0	100	0	0	0	0
420	2040	98	16	66	16	100	0	0	100	0	0	0	0
430	2090	98	20	40	20	100	0	0	100	0	0	0	0
440	2140	97	12	50	25	100	0	0	100	0	0	0	0
450	2190	98	28	28	42	100	0	0	100	0	0	0	0
460	2240	98	16	33	49	100	0	0	100	0	0	0	0
470	2290	98	16	33	49	100	0	0	100	0	0	0	0
480	2340	98	16	33	49	100	0	0	100	0	0	0	0
490	2390	97	22	33	33	100	0	0	100	0	0	0	0
500	2440	97	20	30	30	100	0	0	100	0	0	0	0

SUF: The stem is regularized. For regular stems this indicates that an inappropriate suffix is affixed.

ID: The stem and past tense have the same form.

VC: The stem undergoes a vowel change. For vowel change stems this indicates that an inappropriate vowel change occurs.

BLD: The stem is blended i.e., it undergoes both vowel suppletion and suffixation.

These categories account for the overwhelming majority of errors observed. Residual errors are mostly incorrect mapping of consonants. Scores indicate percentage of total errors in a given class. The exact pattern and timing of errors differs across expansion training schedules. However, only those findings applicable to all expansion conditions, represented by the epoch 5 condition, are presented.

First, note that the overall level of errors is low and circumscribed to a limited range of error types. Second, different verb classes are susceptible to different patterns of errors. For example, regular stems are typically identity mapped, given an inappropriate suffix, or blended. While identity mapping and inappropriate suffix errors occur throughout the training period, blends are more likely to occur later in training. Following the phonological systematicities in the vocabulary, regular stems which are identity mapped are likely to end in a dental final consonant. However, in some cases, non-dental final regular stems are also identity mapped. Identity stems are typically suffixed or undergo a vowel change. Interestingly, all identity stems which undergo an erroneous vowel change possess the requisite vowel/consonant stem final combination. It is also noteworthy that identity stems *never* undergo blending. Finally, vowel change stems display the greatest range of error types, generally consisting of identity mapping, inappropriate vowel changes, and blending. Regularizations are generally rare on vowel change stems.

The previous analysis summarized the proportion of all errors generated by the network for each class, irrespective of whether the network had mastered any of those forms at an earlier point in training. In order to investigate *reorganizational* processes in these networks, Table 4 presents the proportion of stems in each verb class that undergo U-shaped development in each of the five epoch increment conditions. Here, a stem is defined as undergoing U-shaped development if it is correctly produced by the network, then at some subsequent point in training, it is incorrectly mapped and finally, again correctly mapped by the network.

Table 4: Proportion (%) of U-Shaped Verbs by Verb Class

	EPOCH EXPANSION				
CLASS	1	2	5	10	25
ARB	0.0	0.0	0.0	0.0	0.0
REG	20.7	18.3	17.7	12.7	14.0
ID	35.0	25.0	25.0	20.0	20.0
VC	35.0	40.0	30.0	30.0	30.0

These data suggest the following generalizations: First, the proportion of each class of stems which is correctly output and then subsequently erroneously produced is greater for the irregular than regular verbs. Second, as the rate of vocabulary expansion increases, the proportion of U-shaped stems tends to increase for both regular and irregular verbs.

Analyses of error types for the U-shaped errors on regular stems reveal patterns that are consistent with the general learning error analysis presented in Table 3. That is, when a regular verb is U-shaped, it is most likely to be identity mapped, blended and inappropriately suffixed. Interestingly, comparisons across training condition suggest that as expansion rate is slowed (i.e., 10 and 25 epoch increments), the tendency for blending increases and the tendency for inappropriate suffixation decreases. For irregular verbs, U-shaped errors are most likely to result from suffixation, blending, or incorrect vowel changes. These patterns appear to be consistent across training condition. However, the small absolute number of U-shaped irregular stems precludes the identification of stable trends.

U-shaped errors were also analyzed in terms of the point during training in which they occurred, i.e., at what point in training after the verb was correctly mapped did the first *incorrect* output occur. In general, this analysis reveals that the onset of U-shaped errors tends to be distributed relatively evenly across the entire training cycle. However, the data also suggest that there is an interaction between the proportion of verbs undergoing U-shapes during any given period of training and the training condition. In particular, the data suggest that increasing the rate of vocabulary expansion tends to increase the tendency to make an U-shape error early in training. Table 5 summarizes this finding, presenting the proportion of U-shape *onsets* observed in four training periods, for each of the five epoch expansion schedules.

Table 5. Proportion (%) of U-Shape Onsets on Regular and Irregular Verbs by Training Period

	EPOCH EXPANSION				
PERIOD	1	2	5	10	25
20–140	36.7	19.6	17.2	19.1	5.4
141–260	32.1	40.2	38.7	42.6	51.4
261–380	15.6	20.6	21.5	19.1	24.3
381–500	11.9	16.5	21.5	20.6	21.6

An error type by training period analysis on these onsets indicates a relatively even distribution of types of U-shapes across the training period for the *regular verbs*. That is, blends and identity mapping U-shapes were equally likely to occur early and late in the training period. However, there is a tendency for inappropriate suffix U-shapes to occur early in training.

In contrast to the regular verbs, the great majority of the U-shapes on irregular verbs (52 of 58 or 89.7%) occur during the first half of training. The occurrence of irregular U-shapes is split fairly evenly across this period in the five training conditions, although there is some evidence to suggest that U-shapes are more likely to occur during the later phases of learning as the rate of epoch expansion is decreased (i.e., in the 10 and 25 epoch increment condition). Thus, the pattern of U-shaped errors on stems in the regular and irregular classes is quite similar across epoch expansion conditions.

3.2.3 Novel Verbs

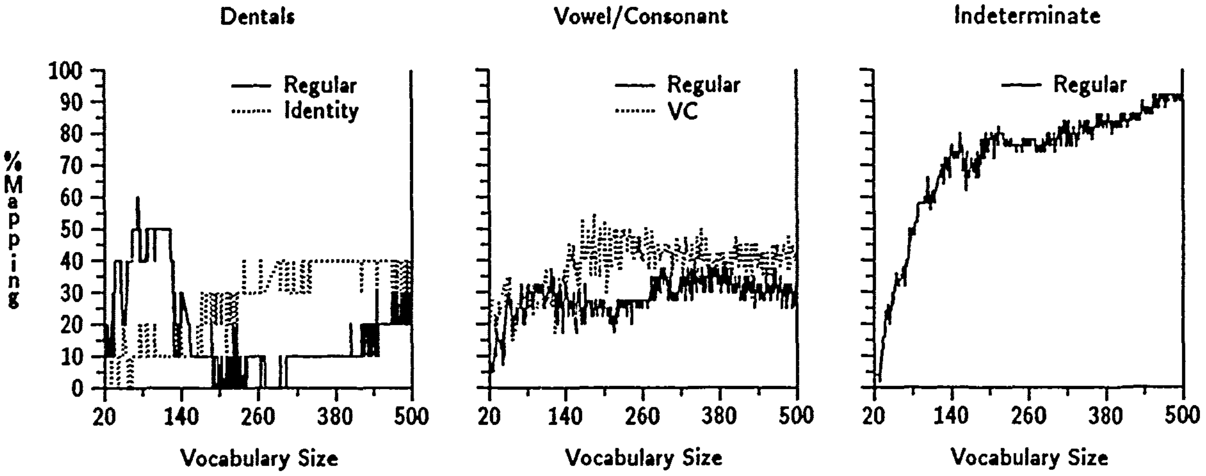
The preceding analyses concentrate on the network's ability to produce the appropriate past tense forms of stems that were members of the training set, i.e., trained verbs. The following analyses investigate network performance when it is required to produce the past tense forms of stems that it has never seen. Figure 2 outlines responses to novel stems in the 5 epoch expansion condition. As with the error analyses presented above, the findings discussed are applicable to all epoch expansion conditions. Figure 2(a) plots the tendency of the network to treat novel stems that end in a dental (Dental-Final) as if they were regular, identity mapping, or vowel change forms. Figure 2(b) plots performance on novel stems which possess stem final vowel-consonant clusters characteristic of vowel change verbs (Vowel Changes). Lastly, Figure 2(c) plots network performance on novel stems which do not possess any particular phonological sub-regularities (Indeterminates). These data are relevant to understanding the network's sensitivity to the predictable phonological characteristics of a stem (or lack thereof) when generating past tense forms, as well as changes in the tendency of the network to generalize regularities across learning.

These figures reveal that the network is indeed sensitive to the phonological properties of stems when generating the past tense forms of novel verbs. First, note that dental final verb stems are more likely to be identity mapped (28%)[8] than verb stems which do not end in a dental (7%). Similarly, novel stems that possess particular Vowel/Consonant stem final clusters are more likely to undergo vowel suppletion (37%) than those which do not possess those sub-regularities (11%). In addition, the network appears to be sensitive to a lack of phonological properties in its tendency to regularize novel stems, although this pattern is less absolute. For example, there is a striking tendency for indeterminate novel stems to undergo regularization (71%); however, a substantial proportion of the dental and vowel/consonant novel stems are also regularized (18% and 26% respectively), especially towards the end of training. Note that very few indeterminate stems undergo identity mapping (3%) or vowel suppletion (0.4%).

Focusing on changes in the tendency to regularize indeterminate novel stems across learning, we observe that the regularization properties of the network alter dramatically between early and late training. Early in training, i.e., immediately after the network has mastered the initial vocabulary of 20 verbs, indeterminate novel stems are treated in an unsystematic fashion. Network output is unclassifiable in terms of the mapping categories used in Table 3[9]. However, as vocab-

[8] The percentages quoted here are means for the whole training period.

[9] During this early period of training, only one of the



(a) (b) (c)

Figure 2: Performance on Novel Stems

ulary size expands (and training continues), the tendency for the network to add a suffix to indeterminate novel stems increases quickly and substantially. Interestingly, this rapid increase occurs at about the same point in training (50 verbs) at which performance on trained verb stems begins to show recovery from the initial performance decrement (cf., Figure 1(b) with Figure 2(c)). Furthermore, while a major proportion of indeterminate stems are regularized by the network by the 140 vocabulary mark (76%)[10], the tendency of the network to regularize novel stems continues to increase, albeit at a less marked rate, as vocabulary size (and training) also increases.

The sudden onset of the systematic treatment of novel stems indicates that abrupt reorganization processes are occurring in the weight matrix of the network. However, it is unclear whether these changes are the result of prolonged training or the result of the network's exposure to an increasing number of different stems. In section 3.2.1, we report the sudden recovery from error on *trained* verbs. Figure 3 plots the performance of the network in mapping *indeterminate novel verbs stems* to regular past tense forms as a function of training (by epoch) on a series of vocabulary sizes[11]. These generalization curves indicate that final vocabulary size, rather than amount of training (number of epochs), is a better predictor of final level of generalization. That is, the ability of the network to generalize the suffix to novel stems continues at a low rate when the final vocabulary size is small. However, the relationship between changes in final vocabulary size and final level of generalization is non-linear. In particular, a substantial increase in generalization tendencies is observed when final vocabulary size is increased beyond 30 verbs.

4 DISCUSSION

In comparison to previous neural network models of the acquisition of the English past tense, the simulations reported here achieve a high level of performance within a reasonable period of training. For example, using similar network architectures and token frequency manipulations on an identical vocabulary configuration, Plunkett & Marchman [In press] report only an average of 80% mastery on verbs in the regular and vowel change classes. The current set of simulations result in overall learning levels of 98% and 100% on these verb classes, respectively, averaged across the five epoch increment conditions.

The improved performance in the current simulations can be attributed directly to the use of an incremental epoch expansion learning schedule, rather than training the network on the entire set of 500 verbs from the initial point in training. On problems that are radically different from the mapping of verb stems to their past tense forms, other researchers (Cottrell & Tsung [1989]; Elman [1989]) have also noted improved overall learning in networks that are trained on subsets

novel indeterminate stems is treated in a systematic fashion. It is regularized. The tendency to treat Dental-Final novels and Vowel/Consonant novels systematically is greater, including 3 regularizations, 3 identity mappings and 11 vowel changes.

[10] Note that this point in training corresponds to a period in which performance on *trained* irregular verbs is low (see Figure 1).

[11] Bullets (•) indicate the point on the generalization curve, where vocabulary expansion is halted for each control simulation. The open circle (o) indicates where vocabulary expansion starts for each control simulation. Vocabulary sizes are indicated in Figure 3.

of the data prior to expanding to the full set. However, the facilitatory effect of an expansion training procedure is not well-understood. Given the statistical nature of these systems, it is likely that limiting the size and/or sampling of a problem domain reduces the probability that the network will extract spurious correlations in the data set. Provided that initial data sets are sufficiently representative of the overall problem space, training on limited data sets increases network efficiency in uncovering the *principal components* of variation that define the problem domain. Once these principal components are encoded in the weight matrix of the network, training on an expanded sample serves to reinforce the initial organization of the weight matrix and reduces the identification of spurious correlations. In addition, the network is better equipped to extract lower order components of variation. In the current context, lower order correlations might correspond to the phonological sub-regularities that characterize the irregular classes.

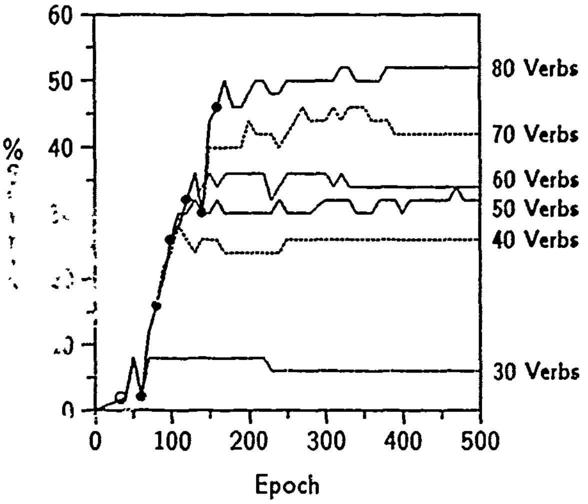


Figure 3· Generalization on Indeterminate stems by Vocabulary Size

Several prominent theories of cognitive development have explored the relationship between the current knowledge state of the child and the nature of the problem domain to which the child is exposed with respect to determining the child's success or failure. For example, Piaget [1953] introduces the notion of *moderate novelty* to summarize the finding that children display the most advancement in those conditions where new problems only exert moderate demand on their current knowledge state. Incorporating slightly different components, Vygotsky [1962] utilizes the notion of *zone of proximal development* for similar purposes. In these simulations, the interaction between the current knowledge state of the network, as encoded in the weight matrix, and the nature and size of the problem space to which the network is exposed can be seen to account for many of the observed behaviors. Networks, like children, appear to benefit from a moderate novelty effect, such that if the initial knowledge state of the network is undifferentiated with respect to the problem at hand (e.g., a random weight configuration), overall performance is enhanced if the learning set is initially restricted and then gradually expanded.

However, recall that these networks have considerable difficulty mastering the past tense mapping problem if the conditions for expansion involve learning each verb to criterion before any additional verbs are added to the vocabulary. Indeed, the upper limit on the number of verbs that the network was able to master was several orders of magnitude smaller in the criterial expansion condition (i.e., 25-30 stems) than in the epoch expansion conditions (i.e., 500 stems). Criterial learning results in weight changes which are derived from only one training pattern (i.e., the new verb), given that the error signal resulting from old verbs has already been minimized. In effect, the network is no longer attempting to solve a global mapping problem (i.e., at the level of the overall problem domain), but it is instead attempting to adjust the weight matrix to the mapping characteristics of a single pattern. Eventually, the network gradually entrenches itself in an increasingly isolated partitioning of the weight space, such that it is impossible for the gradient descent learning algorithm to accommodate to the mapping demands of the new verb stem. The network is trapped in a local minimum, and even indefinite training on the single new verb has little or no effect on performance.

Many theories of cognitive development emphasize the importance of continuous variation of environmental stimulation for cognitive advancement (though also acknowledging the role of maturational factors and internal reorganization processes (Karmiloff-Smith [1986])). It is unlikely that children attempt to learn verb morphology, or other types of tasks for that matter, by limiting the domain of exemplars to just one greater than what has already been mastered. Most current account of lexical acquisition assume that children learn many items in parallel, and *partial* mastery of a lexical items or inflectional systems may best characterize much of development. The gradual but continuous expansion of the problem space that was illustrated in these connectionist simulations can be seen to parallel those aspects of learning in children which ensure that they do not become entrenched in the irrelevant details of a particular problem and are forced to satisfy a multiplicity of constraints characterizing the problem domain.

One major goal of this work was to determine the nature of the mechanisms that trigger the transition from rote learning to system building in children acquiring the English past tense. Two sets of results from the simulations presented here indicate that artificial neural networks can also be observed to pass from a period of rote learning to a period of system building early in learning. A third set of results illustrates that, in these networks, this transition is triggered by quantitative factors relating to the number of verbs which share specific mapping properties and characteristic phonological features.

First, after the initial set of 20 verbs are learned to criterion, the network does not generalize the regularities in that set to new stems. In the period immediately following initial training, both regular and irregular verbs are prone to incorrect mapping (see Figure 1). In addition, the errors produced by the network during this period are not systematic, i.e., they do not appear to reflect the mapping characteristics of the initial learning set (see error tabulations in Table 3). Interestingly, however, the initial 20 verbs are likely to continue to be mapped correctly by the network, even in the presence of the other erroneous mappings[12]. This pattern of results indicates that the network has failed to extract any pattern of regularities from the initial set and treats each stem/past tense pair as an independent mapping problem, i.e., the network has *memorized* the initial set of 20 stem/past tense pairs. Overall performance on new verbs (particularly on regular verbs) continues to deteriorate as new verbs are added to the training set (see Figure 1).

The deterioration in performance on newly introduced verbs is reversed when the vocabulary reaches a specific size (around 50 verbs). At this point in training, performance on new verbs improves rapidly. In addition, errors are now observed on several irregular verbs that were members of the initial training set. The timing of this turnaround in performance appears to be stable across all epoch expansion schedules tested here, and the same patterns of learning have been replicated in networks learning a different random selection of the total vocabulary. These findings suggest that the network has reorganized its representation of the mapping problem, to the extent that the network now treats new verbs entering the vocabulary in a systematic fashion. The fact that the timing of the turnaround is remarkably consistent across the training conditions tested here suggests that the trigger for the transition from rote learning to system building in these networks is associated with the *quantity* of verbs in the training set which undergo systematic mapping processes.

Secondly, the performance of these simulations on novel verbs also confirms the interpretation that the network can be seen to pass from a stage of rote learning to system building. In the period following the initial training on the 20 verbs, responses to novel stems is unsystematic, irrespective of their phonological characteristics (see Figure 2). Thus, correct performance on the initial 20 verbs does not transfer to novel verbs, and the corresponding successful output on verbs in the training set again cannot be attributed to the generalization of mapping characteristics. As vocabulary size is expanded (and training continues), the tendency to treat novel verbs in a systematic fashion increases rapidly. For example, there is a clear tendency to regularize those novel stems which do not possess phonological subregularities characteristic of the irregular classes (i.e., the indeterminates). In addition, novel dental or vowel change stems are now likely to be mapped in accordance with their phonological shape, or they are regularized.

Crucially, the onset of the systematic treatment of novel verbs stems *coincides with the turnaround in the performance on trained verbs*. Thus, the network is consistent in its treatment of trained and novel stems. At this point in training, the network appears to have modified its representation of the verb mapping problem from one in which individual stem/past tense form mappings are encoded independently to one in which classes of mappings can be differentiated into categories. As a result, the process of regularization can be seen to act like a "default" transformation, in that it is likely to be applied to the majority of novel stems, especially when a stem lacks a characteristic phonological shape. The various processes of irregularization, in contrast, are applied to a smaller subset of novel stems and are contingent on the presence of phonological sub-regularities.

A third set of simulations demonstrates that the *quantity* of stems in the training set which undergo systematic mapping serves to trigger the transition from the period of rote learning to the period of system building in these networks. The control simulations illustrated in Figure 3 suggest that size of training set is the best predictor of the regularization of indeterminate novel verb stems. In particular, the size of the training set must be increased beyond 30 verbs before substantial regularization tendencies are observed. This finding clearly does warrant the conclusion that an absolute critical mass of verbs can be defined as a prerequisite for generalization. However, the non-linear relationship between the tendency to generalize the regular mappings and size of training set suggests that repeated training on a small set of mappings will not in itself lead to generalization. The network must be exposed to a sufficient range (types) of mappings. Yet, adequate sizes of training sets cannot be defined independently from the specific network architecture employed and the overall set of mapping characteristics that the network must learn.

In general, these results support the view that a *single mechanism* learning system can offer an alternative account of the transition from rote learning processes to system building in children's acquisition of

[12] It will be recalled (see footnote 4) that regular verbs in the initial training set are only correctly mapped if they have a high token frequency another symptom of rote learning in these networks.

English verb morphology. In contrast to the traditional view which posits an interaction between two qualitatively distinct mechanisms supporting different modes of representation (i.e., rote and rule), a connectionist account posits a single mechanism driven by a general learning algorithm which is capable of both memorization and generalization processes. The network's transition from rote-like to rule like processes are not triggered by the interaction of qualitatively distinct mechanisms, but instead by *quantitative* increments in the size of a structured training set.

The behavior of these networks can be seen to mimic several aspects of the type and timing of children's pattern of morphological acquisition. However, it is as yet impossible to determine the degree to which children's transition from rote learning to system building is driven by similar quantitative changes in the size of the learning set. There is abundant evidence that many children pass first through a prolonged period of development in which they only produce a limited number of verbs and their corresponding appropriate past tense forms. This period is then superceded by one in which verb vocabulary expands at a fairly rapid pace, and errors are likely to occur on irregular (and sometimes regular) past tense forms. However, sufficient longitudinal evidence is not yet available to determine whether vocabulary size operates in a critical mass fashion for children. Furthermore, it is difficult to anticipate the contributions of comprehension or other linguistic processing factors in determining the nature and timing of this transition in children.

However, it is interesting to note that similar relationships between set size and the onset of reorganizational processes have been hypothesized with respect to the "vocabulary (or naming) spurt" that is observed in young children during the second half of their second year Several researchers (Bates, Bretherton & Snyder [1988]; Nelson [1973]; Plunkett [1990]) have noted that there is a shift in the rate of vocabulary acquisition after the point in learning when overall vocabulary reaches around 50 words. Although the interpretation of this finding is controversial, many researchers (e.g., Barrett [1982], Bowerman [1982]) argue that such a transition is associated with a *reorganization* in the structure of children's vocabularies. Prior to this point, word meanings tend to be encoded independently as separate and distinct lexical items, whereas, the vocabulary spurt signals that children's lexicons are now organized into coherent semantic systems This interpretation is consistent with the view that the transition from rote learning to system building can derive from processes inherent in a single mechanism connectionist-like system, driven by quantitative changes in the size of the learning set. A similar view has been espoused by Karmiloff-Smith [1986].

The categories of general learning errors reported for these simulations resemble those reported in earlier work (Plunkett & Marchman [In press]). In particular, patterns of multi-lateral interference are observed between the mapping classes resulting in both regularizations and irregularizations. Furthermore, corroborating earlier findings, different verb classes are susceptible to different types of errors. When errors are made, regular verbs are most likely to be identity mapped; identity verbs are most likely to be suffixed. Vowel change verbs are subject to the widest range of errors. The patterns of timing of errors are also similar to that reported in earlier work, e.g., blending errors tend to be a characteristic of late training. However, because of the high level of performance in these simulations, errors on irregular verbs in the second half of training are virtually absent. Instead, errors on irregular verbs tend to be clustered around the beginning and middle periods of training. Errors on regular verb stems are observed throughout training.

We have argued elsewhere (Plunkett & Marchman [In press]) that the timing and pattern of errors produced in these networks resemble those produced by young children acquiring English verb morphology. Children (and adults) produce incorrect past tense forms over a wide span of development and the types of errors produced suggest a sensitivity to the phonological properties of the irregular classes. Such sensitivities can sometimes interfere with the appropriate production of regular past tense forms, just as the pattern of suffixation interferes with the appropriate production of irregular forms. These multi-lateral patterns of interference are an inherent property of an interconnected connectionist system attempting to map several competing classes of verb types.

In our analyses of the errors produced by these networks, we distinguish errors produced across the course of *learning* from errors that resulted from an apparent *unlearning* of stem/past tense mappings (i.e., those stems that have been produced correctly at some point earlier in training). Much of the empirical foundation for the traditional view of acquisition has been concerned with the latter type of error, perhaps because the phenomena of U-shaped regressions in performance is more theoretically striking than the hypothesized processes which guide the gradual linear acquisition of a verb. Clearly, however, the results from these simulations predict that children's erroneous output results from both types of errors. In these networks, general learning errors typically occur early in training, whereas the occurrence of U-shaped errors is distributed fairly evenly throughout the training period. Interestingly, as with children, the proportion of irregular verbs that are U-shaped is greater than the proportion of regular verbs that are U-shaped. However, irregular mappings are all mastered by the 300 vocabulary size mark. Unfortunately, empirical data comparing U-shaped errors to patterns of general learning errors are not available for children.

Lastly, we tested several types of learning schedules, focusing primarily on the rate at which vocabulary size is expanded across training. The findings suggest that high expansion rates (i.e., 1 epoch increment) are likely to result in:

1. Greater decrements in performance on both regular and irregular verbs early in training.
2. A higher final level of performance on regular verbs.
3. A shorter and less marked performance decrement on irregular verbs in the middle portions of the training period.
4. A greater proportion of stems undergoing U-shaped learning, though these are more likely to be restricted to early periods in training.

At this point, these results can only offer speculative predictions for the study of language acquisition. However, we suggest that the rate of vocabulary expansion that we have modeled in these networks might prove to be analogous to the rate with which young children attempt to assimilate new verbs into their vocabularies. It is possible, for example, that children who vary in their rate of vocabulary acquisition (e.g., "late talkers" vs. "early talkers" (Bates & Thal [In press])) may also vary in the type and timing of overgeneralization errors. In particular, the results from these series of simulations predict that early language learners would be more likely to produce errors earlier than later in acquisition, and U-shaped errors would be less likely to occur later in development. Late language learners, in contrast, would be likely to achieve a slightly lower level of final performance and produce a greater proportion of errors on verbs that do not match the dominant pattern, i.e., the irregular verbs. These predictions are clearly only speculations which must await empirical findings. Nevertheless, the simulation work does suggest that qualitatively different patterns of learning can arise from quantitative manipulations of the input and training schedules.

5 CONCLUSION

Our current understanding about children's acquisition of English inflectional morphology is typically based on naturalistic observations of children's actual productions, and experimental studies using nonsense words (e.g., Berko [1958]). Naturalistic studies typically report that overgeneralization errors comprise a small portion of children's productive vocabularies, however, children do produce a range of regularization and irregularization errors. In contrast, studies using nonsense forms typically demonstrate an overwhelming preference for regularization. However, several more recent studies suggest that children are indeed sensitive to the phonological aspects of nonsense forms. For example, (Derwing & Baker [1986]) have shown that novel stems which end in a dental tend to be identity mapped and novel stems with specific vowel/consonant clusters will sometimes undergo vowel suppletion.

In these simulations, analyses of network performance on *trained* verbs is analogous to naturalistic observations of children's performance on real English words. Correspondingly, network performance on novel verbs is best viewed to correspond to the experimental elicited production studies with nonsense verbs. Interestingly, the overall pattern of results suggests a considerable correspondence between children and networks across these two types of measures. For example, the pattern of errors observed across learning for verbs in the training set are multi-lateral i.e., irregular verbs are regularized and regular verbs are irregularized. In contrast, responses to novel verbs are overwhelmingly more likely to involve regularization. For those novel stems that phonologically resemble irregular verbs, the tendency to regularize is still prominent but identity mappings and vowel suppletions do occur. Thus, evaluations of network performance on novel verbs (in comparison to patterns of errors across learning) suggests that the network, like the child, is best characterized as a rule-governed system. We suggest that evaluations of acquisition using experimental studies using nonsense verbs, while clearly illustrating the generalization abilities of young children, may have succeeded in biasing our view of the nature of the phenomenon to be explained. While we accept that experimental findings contribute important insights into the developmental process, they do not obviate the need for detailed naturalistic studies of children's acquisition of verb morphology. Indeed, it is precisely such data that are needed to evaluate the predictions of the present model in a stringent fashion.

References

Barrett, M. D. [1982], "Distinguishing between Prototypes: The early acquisition of the meaning of object names," in *Language Development*, S. A. Kuczaj, ed. #1: Syntax and Semantics, Lawrence Erlbaum Associates, Hillsdale, NJ.

Bates, E., Bretherton, I. & Snyder, L. [1988], *From First Words to Grammar. Individual Differences and Dissociable Mechanisms*, Cambridge University Press, Cambridge, MA.

Bates, E. & Thal, D. [In press], "Associations and dissociations in child language development," in *Research on child language diorders. A decade of progress*, J. Miller & R. Schiefelbusch, eds., Pro-Ed, Austin, TX.

Berko, J. [1958], "The child's learning of English morphology," *Word* 14, 150–177.

Bever, T. G., ed. [1982], *Regressions in mental development: Basic phenomena and theories*, Lawrence Erlbaum Associates, Hillsdale, NJ.

Bowerman, M. [1982], "Reorganizational process in lexical and syntactic development," in *Language Acquisition: The State of the Art*, E. Wanner & L. Gleitman, eds., Cambridge University Press, Cambridge, MA.

Brown, R. [1973], *A first language: The early stages*, Allen & Unwin, London.

Bybee, J. & Moder, C. L. [1983], "Morphological classes as natural categories," *Language* 59, 251–270.

Bybee, J. & Slobin, D. I. [1982], "Rules and schemas in the development and use of the English past tense," *Language* 58, 265–289.

Bybee, J. L. [1988], "Morphology as lexical organization," in *Theoretical morphology*, M. Hammond & M. Noonan, eds., Academic Press, New York, NY.

Carey, S. [1982], "Semantic Development: The state of the art," in *Language Acquisition: The State of the Art*, E. Wanner & L. Gleitman, eds., Cambridge University Press, Cambridge, MA.

Cottrell, G. & Tsung, F. [1989], "Learning simple arithmetic procedures," *Proceedings of the Eleventh Annual Cognitive Science Society Conference, Ann Arbor, MI*, Hillsdale, NJ.

Derwing, B. L. & Baker, W. J. [1986], "Assessing morphological development," in *Language acquisition: Studies in first language development. Second edition*, P. Fletcher & M. Garman, eds., Cambridge University Press, Cambridge.

Elman, J. L. [1989], "Representation and Structure in Connectionist Models," Center for Research in Language, University of California, San Diego, Technical report #8903.

Karmiloff-Smith, A. [1986], "From meta-processes to conscious access: Evidence from children's metalinguistic and repair data," *Cognition* 23, 95–147.

Kuczaj, S. [1977], "The acquisition of regular and irregular past tense forms," *Journal of Verbal Learning and Verbal Behavior* 16, 589–600.

MacWhinney, B. [1987], "The competition model," in *Mechanisms of language acquisition*, B. MacWhinney, ed., Lawrence Erlbaum Associates, Hillsdale, NJ.

Marchman, V. [1984], "Learning not to overgeneralize," *Papers and reports on child language development* 24, 69–74.

Marchman, V. [1988], "Rules and regularities in the acquisition of the English past tense," *Center for Research in Language Newsletter* 2.

Nelson, K. [1973], "Structure and strategy in learning to talk," *Monographs of the Society for Research in Child Development* 38.

Piaget, J. [1953], *The origin of intelligence in the child*, Routledge & Kegan Paul, London.

Pinker, S. & Prince, A. [1988], "On language and connectionism: Analysis of a parallel distributed processing model of language acquistion," *Cognition* 28, 73–193.

Plunkett, K. [1990], "Identifying formulaic expressions in early language acquisition: The case for articulatory/fluency criteria," *Center for Research in Language Newsletter* 4.

Plunkett, K. & Marchman, V. [1989], "Pattern association in a back propagation network: Implications for language acquisition," Center for research in language, University of California, Technical Report # 8902, San Diego.

Plunkett, K. & Marchman, V. [In press], "U-Shaped learning and frequency effects in a multilayered perceptron: Implications for child language acquisition," *Cognition*.

Rosenblatt, F. [1962], *Principles of neural dynamics*, Spartan, New York.

Rumelhart, D. E. & McClelland, J. L. [1986], "On learning the past tense of English verbs," in *Parallel distributed processing: Explorations in the microstructure of cognition, #2: Psychological and Biological Models*, J. L. McClelland, D. E. Rumelhart & PDP Research Group, eds., MIT Press, Cambridge, MA, 216–271.

Slobin, D. I. [1985], "Crosslinguistic evidence for the language-making capacity," in *The crosslinguistic study of language acquisition #2: Theoretical issues*, D. I. Slobin, ed., Lawrence Erlbaum Associates, Hillsdale, NJ.

de Villiers, J. G. & de Villiers, P. A. [1985], "The Acquisition of English," in *The Crosslinguistic Study of Language Acquisition (Vol 1)*, D. I. Slobin, ed., Lawrence Erlbaum, Hillsdale, NJ.

Vygotsky, L. [1962], *Thought and language*, MIT Press, Cambridge, MA.

Parallel Mapping Circuitry in a Phonological Model

David S. Touretzky
School of Computer Science
Carnegie Mellon
Pittsburgh, PA 15213-3890

Abstract

M^{3}P is a computational model of phonology based on parallel mapping matrices. Rules govern the mappings between successive levels of representation of an utterance, from morphphonemic, to phonemic, to phonetic. The mapping matrix architecture performs several types of sequence manipulation. M^{3}P shows that human phonological processing can take place in a highly constrained architecture, using purely feed-forward circuitry and tight limits on depth of derivations.

1 Introduction

A connectionist architecture that uses a mapping scheme can perform arbitrary combinations of insertions, deletions, and mutations on input sequences in parallel. These operations are useful for describing the effects of human phonological processes, which derive surface phonetic forms from underlying phonemic ones. Processing time can be made independent of the length of the sequence and the number of operations requested. The mapping matrix is a key component of M^{3}P, the "Many Maps Model of Phonology" currently under development by Deirdre Wheeler and myself (Touretzky & Wheeler, 1990a, 1990b; Wheeler & Touretzky, in press).

In addition to performing sequence manipulation, the mapping matrix can also compute efficient projections of sequences, e.g., extracting all the vowels in an utterance. Projections are useful for implementing processes such as umlaut and vowel harmony which operate on a series of vowels, ignoring intervening consonants.[1]

Another component of the M^{3}P architecture recognizes clusters of adjacent segments that meet some feature specification; the cluster may then be operated on as a unit. Like the projection operation, this has special significance for phonology. It is also possible to apply clustering to the output of a projection.

M^{3}P is labeled a connectionist model because its primitive operations are constrained to be efficiently implementable by feed-forward circuits constructed from simple, neuron-like elements. This constraint strongly influences the phonological theory. The long term goal of our work is to devise an architecture that is not only a computationally plausible model of human phonological behavior, but also a source of novel predictions and insights into its nature.

2 Sequence Manipulation Via a Change Buffer

A phoneme sequence may be viewed as a string of segments, each of which is a vector of binary features. Consider the word "cats," whose underlying phonemic form[2] is /kæt+z/. The phoneme /z/ is described by the feature vector [−syllabic, +consonantal, −sonorant, +anterior, +coronal, +continuant, +strident, +voice]. In the surface form, [kæts], the final segment has been changed to have the feature [−voice], giving [s]. Two other types of changes that may occur when deriving surface from underlying forms are insertion and deletion of segments.

M^{3}P uses a "change buffer" to explicitly represent the changes a sequence is to undergo. An input sequence plus change buffer are input to a parallel mapping matrix, which then derives the output sequence by applying the requested changes. The details of the map are discussed in the next section. Figure 1 shows examples of a mutation, an insertion, and a deletion via this mechanism. Although in each case only a single change is shown in the change buffer, any number of insertions, deletions, and mutations can be processed simultaneously.

[1] In autosegmental phonology terms, such processes are said to look only at selected "tiers," with vowels and consonants occupying separate tiers.

[2] The tradition in linguistics is that each morpheme be represented by a single underlying form. For reasons having to do with the overall simplicity of the analysis, the underlying representation of the English plural morpheme is assumed to be /z/.

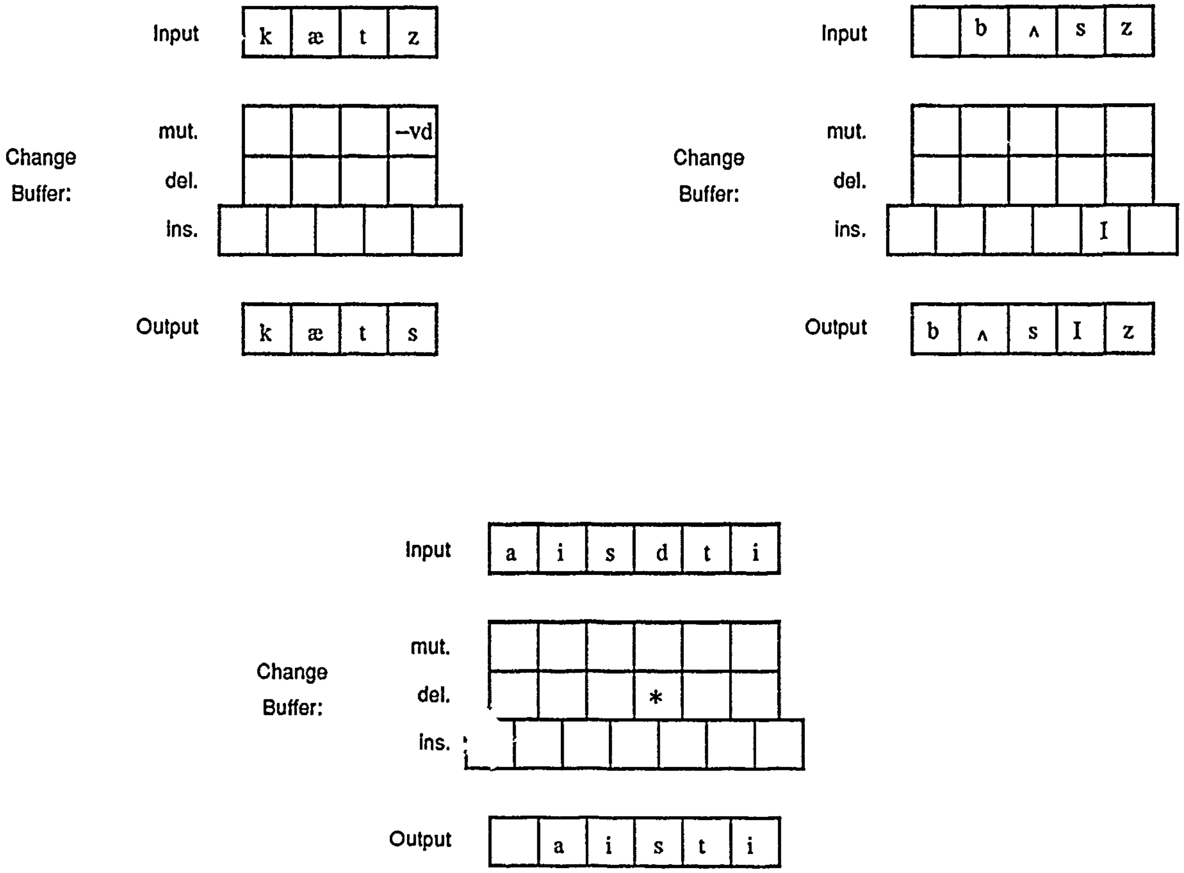


Figure 1. Examples of (a) a mutation in the derivation of "cats"; (b) an insertion in the derivation of "buses", (c) a deletion in the derivation of "iced tea".

Rules in M^3P are implemented by binary threshold units that look at some region of the input buffer and, if they fire, write some pattern into the change buffer. Rule units are replicated at each input buffer position so that they may apply to the entire utterance in parallel.

The change buffer approach is fundamentally different from other connectionist models that manipulate phoneme sequences. For example, the Rumelhart and McClelland (1986) verb learning model employed a single layer of trainable weights to directly map words, represented as sets of Wickelfeatures, from input to output form. MacWhinney's model of Hungarian verb inflection (personal communication) uses a syllabically-structured sequence representation and a layer of hidden units, but it too maps inputs directly to outputs. Both these models can learn much more complex transformations than can be expressed in M^3P's change buffer formalism. This fact is crucial to the Rumelhart and McClelland model, since it combines morphological knowledge with phonological processing in order to derive exceptional past tense forms such as "go"/"went" using the same mechanism as for regular forms like "kiss"/"kissed". But the direct mapping approach has several disadvantages, as noted by Pinker and Prince (1988); one is that it is underconstrained.

3 Operation of The Mapping Matrix

The mapping matrix is designed to produce an appropriately changed string, right justified in the output buffer, containing neither internal gaps nor collisions. Without this device, gaps in the string could arise due to deletions requested by the change buffer, and collisions could occur as a result of insertions. Figures 2 and 3 show how the matrix handles the derivation of the word "parses" (underlyingly /pars + z/), in some New England dialects. Three process apply in this example: fronting of the /a/ vowel before /r/, r-deletion, and insertion of an /I/ as part of the regular process of English plural formation, producing the surface form [pæsIz].

Each square of the mapping matrix is a register that can hold one segment (phoneme.) The first step in the operation of the matrix is to copy each segment of the input buffer down to all the squares in its column, as shown in Figure 2. If an input segment is to undergo a mutation then it is the mutated segment that is copied down to the matrix. For example, the change buffer indicates that the /a/ is to undergo the change [−back]; it therefore shows up as /æ/ in the matrix. If an input segment is to be deleted, the deletion bit being turned on in the change buffer disables all the squares in that column of the matrix. Hence the /r/ does not appear at all in the matrix; the disabled squares are marked with a gray line. Finally, if an insertion is indicated in the change buffer, the inserted segment is copied down to all the squares in its column. Input segments are assigned to *every other* column of the mapping matrix in order to leave room for possible insertions.

Note that a to-be-inserted segment cannot simultaneously undergo a change, because it is not present in the input buffer. However, it is possible for an input segment to undergo multiple changes. For example, a vowel could be simultaneously undergo rounding and nasalization by independent rules; the changes would combine in the mutation slot of the change buffer.

Another architectural limitation is the number of insertions that may take place between adjacent input segments. The change buffer currently has room for only one insertion between each pair of adjacent inputs. This limit could be raised at the cost of extra hardware. For example, we could choose to provide two insertion slots between every pair of input segments, and align the inputs with every *third* column of the mapping matrix. However, human languages appear to require only a single insertion.[3] Permitting multiple insertions would raise another issue: suppose two rules both try to insert a segment *A* or *B* between the adjacent inputs *xy*. Some sort of arbitration mechanism would be needed to prevent the rules from writing into the same insertion slot. And if they each write into a different slot, should the result be *xABy* or *xBAy*? Resolving such problems would require additional circuitry and complicate both the model and the phonological theory.

Once segments have been loaded into the mapping matrix, the next step is to read off the output string. There are several ways this might be done, with differing cost/performance characteristics. In the simplest approach, which uses $O(n^2)$ units each with $O(n)$ connections, every active square in the matrix inhibits all the squares to its left in the same row, and all the squares below it in the same column. After the matrix settles, there will be at most one square active in any row or column, as shown in Figure 3. One can then read out the string by or'ing together all the squares in each row.

The problem with this simple approach is that the settling time of the matrix is linear in the length of the string. Consider the /z/ column; let z_i refer to the cell in row i of that column, with row 1 being at the top. Initially both z_1 and z_2 are active, so z_2 inhibits all the squares to its left, including I_2. But z_1 inhibits z_2, so when z_2 turns off, I_2 can turn back on again, which causes I_3 to turn off, and so on. The rightmost squares stabilize first; the leftmost ones may flip on and off several times before settling into their final states.

The matrix can be made to settle in constant time by using slightly more complex circuitry. In this scheme, each column counts the number of active squares to its right in the topmost row. If a square is in row i and there are exactly $i - 1$ active squares to the right of its column, then that square will remain active; otherwise it will shut itself off. After a single update cycle, all squares will reach their final state.

Let $u_{i,j}$ be the unit governing the mapping matrix square in

[3] This claim is somewhat controversial. Counterexamples have been proposed, but in some cases the data is unclear, and in others the insertion may be a morphological process rather than a phonological one.

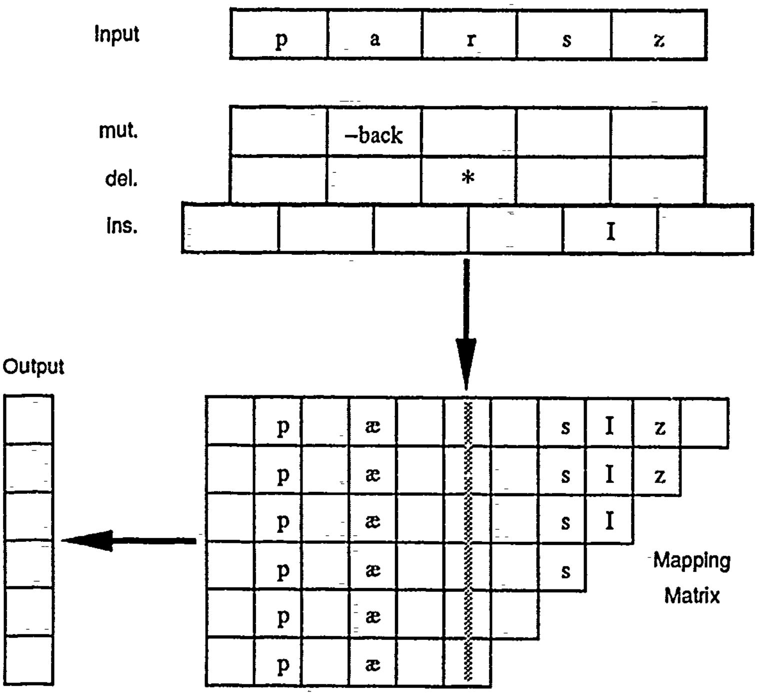


Figure 2: First step in derivation of the word "parses": copying input segments into the matrix.

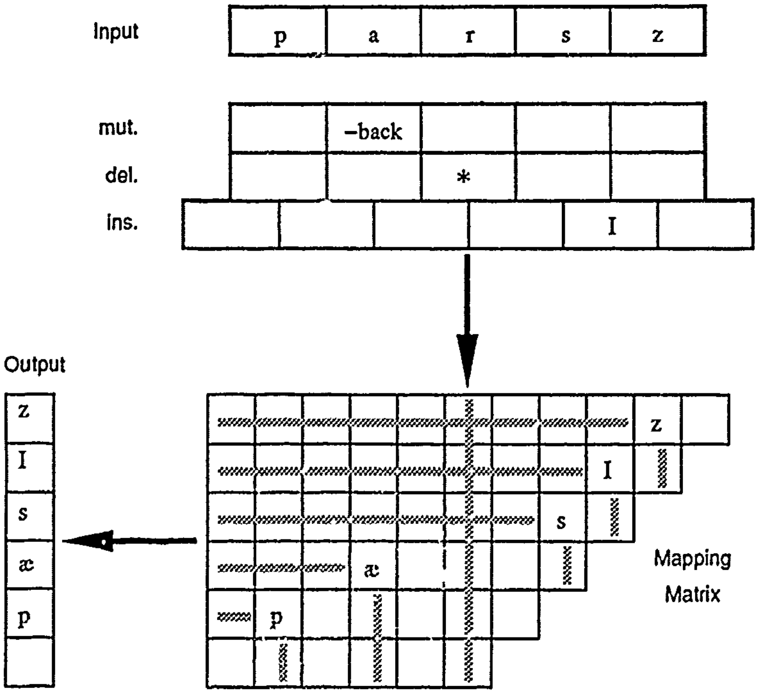


Figure 3: State of the matrix after settling.

row i, column j. This unit will have excitatory connections of weight 1 from the rightmost $j-1$ squares in the topmost row of the matrix; its threshold will be $i - 0.5$. It will also have an inhibitory connection from unit $u_{i+1,j}$. Note that this circuit is strictly feed-forward. After two update cycles (one for units to become activated, and one for them to be inactivated by the inhibitory connections from units representing higher counts), all the $u_{i,j}$ will have reached their final states. At this point the $u_{i,j}$ can be used to gate the corresponding squares of the mapping matrix. The extra cost of this scheme for constant-time mapping is just $O(n)$ units per column to keep the counts. Thus, the architecture as a whole still requires just $O(n^2)$ units, each with $O(n)$ connections.

Further savings are possible if we assume a bound on the change in length of the string. Let Ins_i and Del_i be the total number of insertions and deletions, respectively, that apply to columns i through n. Suppose that for every i, $|\text{Ins}_i - \text{Del}_i| \leq k$ for some constant k. This is a reasonable assumption for human languages. In this case, the upper triangular matrix can be replaced by a band of width $2k+1$, which requires only $O(kn)$ units. Figure 4 shows the shape of the band for the case $k = 1$.

4 Projections

The projection of a sequence with respect to some predicate is the subsequence consisting of only those segments that satisfy it. The most common example of projection in phonology is the vowel projection. Some phonological rules operate on sequences of vowels, ignoring any intervening consonants. If rules are implemented by single binary-threshold units as in M^3P, those rules that are required to skip over variable numbers of consonants cannot look directly at the input buffer; they must look at its vowel projection.

The mapping matrix, without a change buffer, can be used to take projections, as shown in Figure 5. Columns that do not contain segments of the appropriate type (e.g., vowels) are shut off; the remaining segments are then collected together to form a contiguous sequence in the output buffer.

In order for rules to operate on the segments of a projection, they need to be able to *back-project* to the original string. For example, imagine a rule that rounds a vowel if the following vowel is round.[4] The rule would have to look at the vowel projection in order to determine adjacency of vowels separated by an arbitrary number of consonants. As shown in Figure 5, the vowel projection of the input string /karestifu/ is /aeiu/. The rule applies to the vowel /i/ because the vowel to its right, /u/, is round. Therefore the rule should write [+round] into the mutation slot of the /i/ in the change buffer. The problem is how to find this slot. The /i/ is the third segment in the vowel projection buffer, but it is the seventh segment in the input buffer. By back-projecting its changes through the map, the rule can deposit them in the appropriate change buffer slot. The back projection is shown as a dotted line in the figure; the change buffer itself is omitted to save space.

5 Clustering

Phonological phenomena commonly involve *iterative* processing. Consider voicing assimilation in Russian, where the voicing quality of the rightmost member of a consonant cluster spreads to the entire cluster. One way to describe this process is to have a rule that proceeds right-to-left through an utterance, voicing (or devoicing) consonants if their right neighbor is a consonant and is voiced (or voiceless, respectively):

/	mcensk	bï	/	'if Mcensk'
	|			*voicing assimilation*
	mcensg	bï		
	|			*voicing assimilation*
	mcenzg	bï		
	|			*voicing assimilation (null effect)*
[	mcenzg	bï	]	

More modern theories, such as autosegmental phonology (Goldsmith, 1990) treat iteration as a spreading phenomenon; the rule applies only once, but that application may spread the feature [+voice] to any number of adjacent segments. Iterative processes in M^3P are modeled using special grouping circuitry for recognizing clusters of segments that should be treated as a whole. The voicing assimilation rule would be represented this way:

Cluster type:	[−syllabic]
Direction:	right-to-left
Trigger:	[+consonant, −sonorant, αvoice]
Element:	[+consonant]
Change:	[αvoice]

The trigger of a cluster is a consonant; the elements are the preceding consonants. The result of the rule is that the elements of the cluster all become voiced. Figure 6 shows how the Trigger and Element bits are set for the utterance /mcensk bï/. When a cluster rule writes its changes into the change buffer, the change is recorded only for those segments whose Element bit is set.

Another process commonly described using iterative rule application is vowel harmony, whereby properties of a trigger vowel such as height or roundness spread to one or more succeeding vowels, ignoring any intervening consonants.

[4] Such "umlaut" rules exist in many of the world's languages, but not English. In Icelandic, for example, a pair of umlaut processes convert underlying /fatnaδ + um/ "suit of clothes" (dative plural) to surface [fötnuδum]. See (Anderson, 1974) for one analysis For clarity of exposition, in this paper I use an artificial example rather than the actual Icelandic rule.

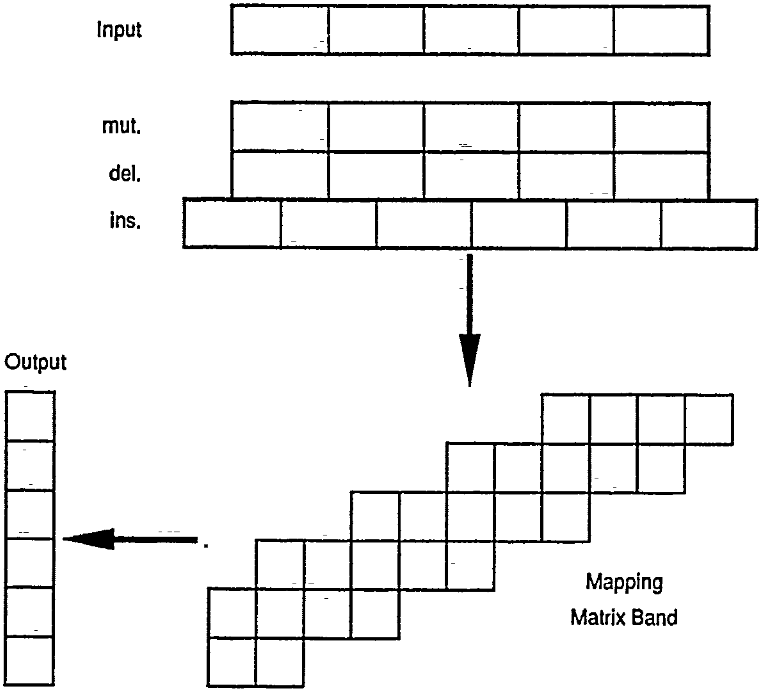


Figure 4: Minimum band of the mapping matrix required when $k = 1$.

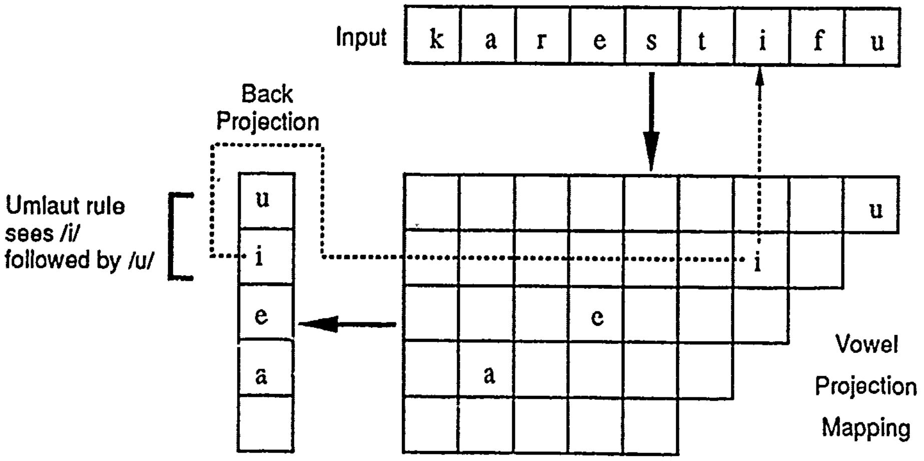
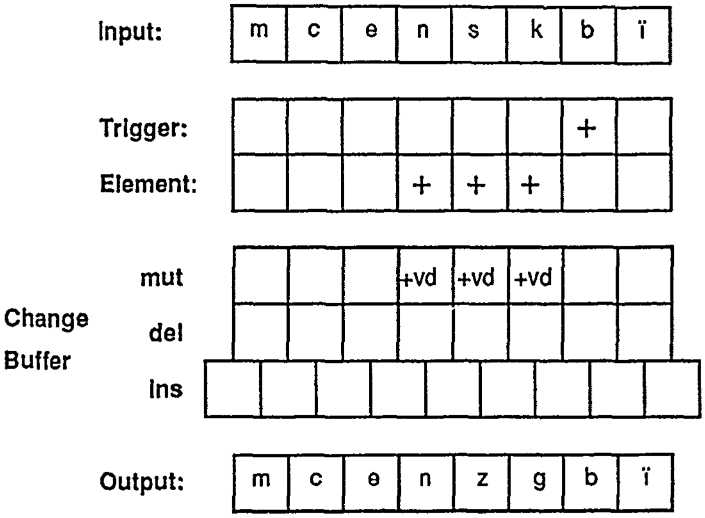


Figure 5: Taking the vowel projection of a stri..3, and back-projecting to apply the result of a rule.



Input:	m	c	e	n	s	k	b	ï	
Trigger:							+		
Element:				+	+	+			
Change Buffer mut				+vd	+vd	+vd			
Change Buffer del									
Change Buffer Ins									
Output:	m	c	e	n	z	g	b	ï	

Figure 6: Parallel implementation of voicing assimilation via clustering.

Vowel harmony is implemented in M^{3}P by applying clustering to the vowel projection. See (Touretzky & Wheeler 1990a) for an example.

6 M^{3}P: The Big Picture

The M^{3}P model is organized as a collection of maps. Based on proposals by Goldsmith (1990) and Lakoff (1989), the model utilizes three levels of representation, called M (morphophonemic), P (phonemic), and F (phonetic). M is the input level and holds the underlying form of an utterance; P is an intermediate level; and F is the output level, containing the surface form. The model supports just two levels of derivation: M-P and P-F, as shown in Figure 7. The clustering and projection circuitry is omitted from this diagram. Cluster rules write their requested changes into the appropriate change buffer in parallel with ordinary rules.

Another portion of the M^{3}P model, also not shown in the diagram, deals with syllabification. M-level strings that do not meet a language's syllabic constraints trigger insertions and deletions (via the M-P mapping matrix) so that the P-level string is always organized into well-formed syllables (Touretzky & Wheeler, 1990b). Several types of phonological processes are sensitive to syllabic structure, the most notable being stress assignment. Syllabic information is available to rules through a set of onset, nucleus, and coda bits set at M-level by the syllabifier, and transmitted to P-level via the M-P map.

7 Discussion

This work was inspired by Goldsmith's and Lakoff's earlier proposals for three-level mapping architectures where rules could apply in parallel. The crucial feature distinguishing their work from ours is the reliance on intra-level well-formedness constraints in addition to inter-level rules. In their models, constraints can interact with rules and with each other in complicated ways. Their models therefore require some form of parallel relaxation process, perhaps even simulated annealing search.[5] Neither has been implemented to date, due to the complexity of the computation involved.

We were able to implement M^{3}P using simple, feedforward circuitry because our model permits only inter-level rules. But denying ourselves the luxury of a powerful and essentially unconstrained relaxation process forced us to drastically rethink the model's structure. Specialized clustering and syllabification primitives were introduced in compensation.

M^{3}P is thus a highly constrained architecture. It cannot, to use a now famous example, reverse all the segments of an utterance (Pinker & Prince 1988), or perform many other elaborate sequence manipulations which are not found in human languages. It also permits only limited depth derivations. This constraint gives rise to some interesting predictions. For example, if vowel harmony is treated as a P-F process because vowels inserted by the M-P syllabification mechanism can undergo harmony, then since there are no further derivational levels after P-F, the model predicts that (across all human languages) vowel harmony will not feed

[5] Goldsmith in fact makes explicit reference to Smolensky's Harmony Theory (Smolensky, 1986).

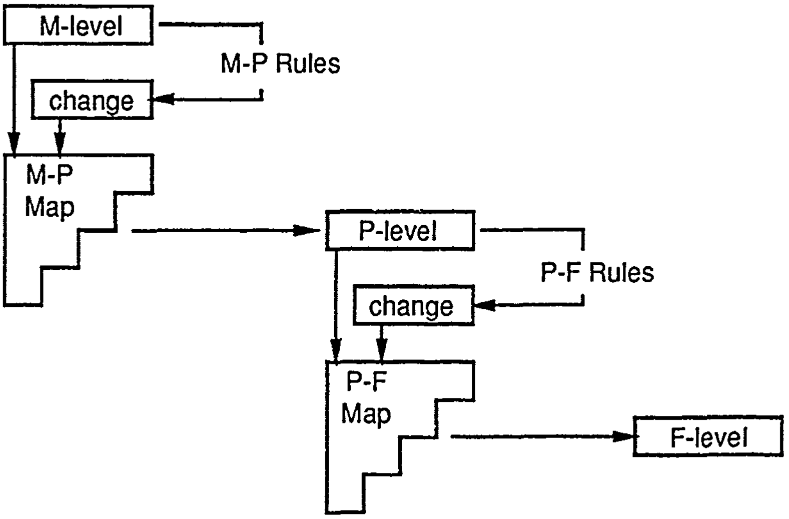


Figure 7: Overview of the M^{3}P model.

other phonological processes.

The model has been applied successfully to limited sets of data drawn from a variety of languages (English, Mohawk, Yawelmani, Turkish, Slovak, Gidabal, Russian, Korean, and Icelandic.) We are currently extending it to deal with stress and tone, and planning a more comprehensive analysis of a single language as a further test of the model's validity.

Acknowledgements

This research was supported by a contract from Hughes Research Laboratories, by the Office of Naval Research under contract number N00014-86-K-0678, and by National Science Foundation grant EET-8716324. I thank Deirdre Wheeler for her many contributions as co-developer of the M^{3}P model, and Gillette Elvgren III for his work on implementing the simulations.

References

Anderson, S. (1974) The Organization of Phonology. New York: Academic Press.

Goldsmith, J. A. (1990) *Autosegmental and Metrical Phonology*. Oxford, UK: Basil Blackwell.

Lakoff, G. (1989) Cognitive phonology. Draft of paper presented at the UC-Berkeley Workshop on Constraints vs Rules, May 1989.

Pinker, S., & Prince, A. (1988) On language and connectionism: analysis of a parallel distributed processing model of language acquisition. In S. Pinker & J. Mehler (eds.), *Connections and Symbols*. Cambridge, MA: The MIT Press.

Rumelhart, J. L. and McClelland, D. E. (1986) On learning the past tenses of English verbs. In J. L. McClelland and D. E. Rumelhart (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 2. Cambridge, MA: The MIT Press.

Smolensky, P. (1986) Information processing in dynamical systems: foundations of harmony theory. In and D. E. Rumelhart and J. L. McClelland (eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1. Cambridge, MA: The MIT Press.

Touretzky, D. S., and Wheeler, D. W. (1990a) A computational basis for phonology. In D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2*, pp. 372-379. San Mateo, CA: Morgan Kaufmann.

Touretzky, D. S., and Wheeler, D. W. (1990b) Two derivations suffice: the role of syllabification in cognitive phonology. In C. Tenny (ed.), The MIT Parsing Volume, 1989-1990. MIT Center for Cognitive Science, Parsing Project Working Papers 3.

Wheeler, D. W., and Touretzky, D. S. (in press) A connectionist implementation of cognitive phonology. In J. Goldsmith (ed.), *Proceedings of the UC-Berkeley Phonology Workshop on Constraints vs. Rules*. University of Chicago Press.

A modular Neural Network Model of the Acquisition of Category Names in Children

Philippe G. Schyns
Dept. of Cognitive and Linguistic Sciences
Brown University
Providence, R.I., 02912

Abstract

A neural network architecture in which names and concepts are learned in separate modules is presented. The categorizing module extracts concepts and provides a complete conceptual interpretation of exemplars to which the naming module associates category names. The output of the categorizing module--the conceptual map--plays an important role in encoding and transmitting important information to the naming module. This paper investigates how the development of a conceptual representation constrains specifically the acquisition of a simple lexicon. The model's behavior is compared to lexical development in children.

1 INTRODUCTION

The lexical categories of young children are often broader than those of adults (Clark, 1973, 1983; Chapman et al., 1986). Consequently, when asked to po'nt for example at dogs, children may group together all the pets that have four legs such as dogs, cats, iguanas and so forth. Eventually, after gradually reorganizing and restructuring their early conceptual knowledge, the children's use and comprehension of category terms narrow down to match adults concepts. What are the mechanisms, representations and constraints underlying these misnamings of categories known in psychology as overextension errors?

I will present in this paper a possible answer to this question using a modular model in which the acquisition of category labels is constrained by the development of conceptual knowledge. Two functionally independent operations, *categorizing* and *naming* are implemented by two structurally different modules. In modular systems, knowledge is "informationally encapsulated" (Fodor, 1983) and the modules communicate only at their input and output stages. Therefore, representational issues are of prime importance in this paper. The interface between two modules, in particular its encoding capacities, plays a crucial role in propagating the informational flow through the system. The topic of concept learning will provide a nice medium to illustrate how such codes self-organize to allow specific information to flow, and thereby constrain what is learnable by other modules.

1.1 CHILDREN'S REPRESENTATION AND NAMING OF CATEGORIES

Recent work on the acquisition of meaning by young children has concentrated on the early constraints children honor when they are exposed to new words (Au & Glusman, 1990; Markman, 1990). In principle, when a child faces a new word, the meaning that he or she could assign to this label is underconstrained. For example, if the child hears "iguana" when confronted with a particular species of lizard, he or she could consider that the label refers to the iguana's color, or its shape, or one of its parts, or its weight and so on. One way in which children could initially constrain word meaning is by assuming that new labels always refer to a whole object rather than to its parts or any of its properties. This initial constraint on the meaning of words is referred to as the Whole Object Constraint (Markman, 1990). The Mutual Exclusivity Constraint is another important initial assumption (Markman, 1990). It states that children constrain word meaning by assuming that words refer to mutually exclusive sets of objects. Put differently, children would assume at first that each object has only one label. These initial constraints provide a broad framework to understand a phenomenon such as the overextension of category terms. The current research will present a possible account of the processes and representations underlying Mutual Exclusivity.

Early constraints on word meaning depend ultimately on a specific theory of concept organization. The organization of young children's conceptual knowledge seems compatible with the Prototype Theory of concepts (Knapp & Anderson, 1984; Posner & Keele, 1968; 1970; Smith & Medin, 1981). According to Prototype Theory, a category is represented by its central tendency, an abstraction called the prototype. In this account of category representation, concept learning is straightforward. As children see more and more exemplars from a particular category, e.g. *dog*, they eventually pick out a set of statistically relevant features that compose the prototype, e.g. *has-four-legs*, *barks*, etc. while leaving out of the representation features particular to some exemplars, e.g. *has-pointed-ears*, *brown*, *has-a-curly-tail*, and so forth. Categorization is also very simple, it proceeds by comparing an instance of a category to each known concept--i.e., each prototype. In Prototype Theory, the output of categorization is graded and complete. Before categorization, there is no way to know what stored prototype the exemplar has to be compared to. In order to have a complete and graded conceptual interpretation, it is necessary that the output of categorization indicates how *typical* the exemplar is with respect to *each* known category. *Gradedness* and *completeness* of conceptual judgments will be two representational constraints imposed on my model of concept learning.

1.2 MOTIVATIONS FOR A MODULAR ARCHITECTURE

Traditionally, models of concept learning have assumed an omniscient teacher that provides the category names to associate with exemplars (Amari, 1977; Gluck & Bower, 1988; Knapp & Anderson, 1984). However, recent studies have shown that concepts can be acquired without the help of category terms (Chapman et al., 1986, Quinn & Eimas, 1986). Indeed, an argument can be made that adults do not have a one-to-one relationship between lexical and conceptual knowledge either (Murphy, in-press). Just think, for example, of the different styles of *chair* we are able to discriminate without always knowing different names for them. The independence between words and representations points out the important *label-concept independence constraint* that should be imbedded in a model of concept learning. the acquisition and the development of conceptual representations must be independent from their lexical tagging.

Label-concept independence stresses the idea that learning to apply a category term to a set of referents can have an unsupervised and a supervised learning component. Thus, learning to represent categories could be described, to a certain extent, as an unsupervised learning task (Grossberg, 1976; Rumelhart & Zipser, 1985). Since no system can guess the correct category name to associate to an exemplar, lexical tagging is definitely an instance of supervised learning.

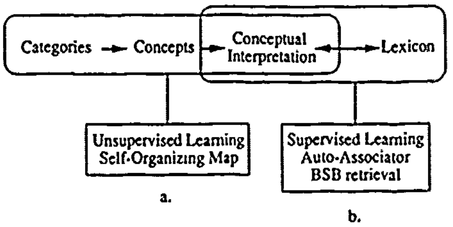

Categories
Concepts
Conceptual Interpretation
Lexicon
Unsupervised Learning Self-Organizing Map
Supervised Learning Auto-Associator BSB retrieval
a.
b.

Figure 1: Abstract Representation of the Implementation of *Categorizing* and *Naming*.

Figure 1 presents the architecture. Part (a) of Figure 1 represents the module that performs the unsupervised learning task, an operation I call categorizing. This module is implemented with a learning scheme that is derived from self-organizing maps (Kohonen, 1982, 1984; Ritter and Kohonen, 1989; Schyns, 1990a; von der Marlsburg, 1973). Part (b) of Figure 1 includes the naming module. A simple pattern associator and an error-correction learning rule realize lexical tagging. Note on Figure 1 the conceptual map that constitutes the medium of communication between the categorizing and the naming modules.

1.3 DESCRIPTION OF THE CATEGORIZING MODULE

The categorizing module is implemented with an architecture derived from the work of Kohonen (1982, 1984). Structurally, this architecture is characterized by an n-dimensional input vector x fully connected to a two dimensional map of output units. Therefore, each output unit o_i is fully connected to the input vector i with a n-dimensional weight vector w_o. The architecture is linked to a discriminant function and a learning rule. Formally, it segregates the input space into distinct subspaces and encodes them on the map with regions of correlated levels of activation.

The discriminant function described in (1) computes the activation of each unit on the map by taking the inner product of the input vector x and each weight vector w_i. A Winner-Take-All scheme selects the output unit o_w with the highest level of activation.

$$o_w = \max_j (w_j^T \cdot x) \qquad (1)$$

The learning rule presented in (2)-(3) assumes that each output unit on the two dimensional map is connected to its neighbors with local excitatory connections lc (o_i, o_j). The fixed strength of each connection is a Gaussian function of the distance between a particular output unit and its neighbors.

$$\Delta w_i = x \; [(1 - | o_w |) \cdot lc\,(o_i , o_w)] \quad \text{for } i \text{ in } N_w \qquad (2)$$

$$\Delta w_i = 0 \quad \text{for } i \text{ not in } N_w \qquad (3)$$

The learning rule adds a fraction (the portion of Equation 2 between the brackets) of the input sample to the weights of the output units locally connected to the winner o_w. Therefore, with learning, the weights of the output units in N_w rotate in weight space in direction of the input sample in input space, and the region defined by N_w has its output units correlated in their activation values. If we generalize this scheme to input samples of different sorts, we get different regions on the map that respond selectively to each kind of sample.

The neighborhood size and the gain parameter have to decrease with time for the network to converge on a correct solution (Kohonen, 1984). In my version of the learning rule, these two conditions occur as by-products of (2). As learning goes on, when the activation of an output unit tends to saturate, the modification of its weight vector tends to 0 because the gain parameter, $(1 - | o_w |)$, goes to 0. This property implements the decreasing of a gain parameter. The connections linking the output units to the winning unit become less efficient because of this reduction in gain. Since they already had a low value due to the Gaussian distribution, with learning, the neighborhood size will implicitly shrink. Moreover, these two parameters need never be reset. Due to (2), the system has an automatic learning mechanism whose onset is triggered by novel categories of objects: an atypical exemplar will produce a low activation state on the map, and learning will reoccur automatically.

1.4 DESCRIPTION OF THE NAMING MODULE

The naming module is implemented with a simple autoassociator using an error-correction learning rule (Widrow & Hoff, 1960). The idea behind error-correction is to minimize, i.e. to correct, the error between a teacher output and the actual output by modifying a connection matrix. In this model, the teacher vector t will be composed of two parts: first the output of the categorizing module, the complete conceptual interpretation of an exemplar, and second, the category name to associate with the exemplar. Formally, learning to associate category names to conceptual interpretations can be described as follows

$$f = W \cdot f \qquad (3)$$

$$e = t - f \qquad (4)$$

$$\Delta W = \alpha \cdot e \cdot f^T \qquad (5)$$

The f component of (3) is decomposed into two parts: the conceptual interpretation f' and the category name f'', i.e. f = f' | f'' (where | stands for the operation of concatenation). The modification of the weight matrix will be proportional to the error vector e and the learning parameter α (see (4) and (5)). When the error vector is null, i.e. when perfect learning will be achieved, $\Delta W = 0$.

Once the system has learned its lexicon of category terms, a mechanism has to be provided to label the conceptual interpretation produced by an exemplar. In other words, how is it possible to retrieve f'' given f'? Since the conceptual map provides a complete and graded conceptual judgment, retrieving the category name can be thought of as a disambiguation process. The interesting dynamic properties of the Brain State in a Box model (Anderson et al., 1977) will achieve the disambiguation task. Formally, the dynamics of the system are described by:

$$x (t + 1) = \lim(\alpha\, x (t) + \beta \; W x (t)) \qquad (6)$$

where
$$\lim(x_i) = \begin{cases} 1 \text{ if } x_i > 1, \\ -1 \text{ if } x_i < -1, \\ x_i \text{ otherwise,} \end{cases}$$

and the equation to measure energy is

$$E(x) = - \; 1/2 \, x^T \cdot W \cdot x \qquad (7)$$

BSB is a dynamical system that evolves to stable states, or attractors by implementing a gradient descent algorithm (Anderson, Siverstein, Ritz & Jones, 1977; Golden, 1986; Kawamoto & Anderson, 1985). W represents the cross-connections between the elements of the different f' stored in the associative memory. To retrieve a particular category name at test phase, the conceptual interpretation f' is the initial state of the dynamic search, so at t = 0, x(t) = f' | 0. Since the information about the name f'' is stored in the cross-connection matrix, the search strategy repeatedly passes this vector through W to reconstruct the missing information. On its way to an attractor, x(t) will be similar to f = f' | f'', and the category name will be revealed. Note that the retrieval scheme implemented here is synchronous and deterministic. Equation (7) is the Liapunov energy function described by Hopfield (1982); it provides a measure of distance from the attractor. The number of iterations required by BSB to fill f'' will be used as an analogy to reaction time in psychological experiments.

2. EXPERIMENT

In this experiment, I will demonstrate how the system described above overextends and narrows down category terms. The architecture will be initially loaded with three concepts in the categorizing module mapped in a one-to-one relationship to three labels in the naming module. For the sake of presentation, the categories of objects have been called *dog*, *cat* and *bird*, although they really are abstract drawings. The way in which this initial knowledge is learned is described in Schyns (1990b). Therefore, I will only sketch the major properties of the two modules and concentrate in the present research on the problem of learning new categories with prior conceptual knowledge. The behavior of the categorizing module agrees with Prototype Theory on most of its major points. Prototype Theory proposes that concepts are stored as prototypes; the self-organizing map encodes prototypes in its weight vectors. In Prototype Theory, a concept is compared to each known prototype. Here the output of the categorizing module, the conceptual map, represents a complete conceptual interpretation of the input sample, where zones of activation reflect the typicality between the exemplar and a particular prototype. Thus, the categorizing module accounts for the type of concept learning that doesn't need a teacher to inform about category membership. The conceptual code built on the conceptual map, the complete and graded conceptual interpretation of exemplars, constrains specifically what is learned in the naming module. A category name is learnable if the conceptual map distinctively encodes this category. Therefore, lexical tagging is dependent on the development of conceptual knowledge. As the prototypes of the categories give rise to unambiguous conceptual interpretations on the map, their category name is retrieved faster than those of exemplars. To be more specific, the state vector consisting of the unlabelled conceptual interpretation of a prototype $(f'_p \mid 0)$ starts closer to an attactor in state space than the unlabelled conceptual interpretation of the exemplar $(f'_e \mid 0)$. Thus, the performance of the naming module is modulated by the nature and the quality of the information that flow between the modules (Schyns, 1990b).

The next experiment is composed of three phases. In the first phase, the system loaded with the three concepts and the three labels will be briefly exposed to examples from a new category, *wolf*. This category will be quite similar to the *dog* category. During the second phase, the unsupervised module of the network will have extensive experience with exemplars from the three known categories, plus *wolf*. When a reorganization of conceptual knowledge occurs, the system in the third phase will correctly associate the new category term "wolf." At each phase of the experiment, I will stress the constraints on word learning imposed by the internal conceptual codes on the naming module.

2.1 PARAMETERS OF THE EXPERIMENT

The self-organizing architecture presented earlier was characterized by the following parameters. The input vector of dimensionality 100 was fully connected to a two dimensional map of 10 x 10 output units. Each output unit had local connections with each of its neighbors up to a Euclidian distance of 2.83. This distance determines a square of plus or minus two units away from each unit on the map. The value assigned to each local connection strength was a Gaussian function (with sigma = 1.5) of the distance between a particular output and each of its locally connected neighbors.

2.1.1 Stimuli of the Categorizing Module.

The categories were composed of distortions around prototypes (Knapp & Anderson, 1984; McClelland & Rumelhart, 1985; Posner & Keele, 1968). Each prototype was a vector of dimensionality 100 composed of values equal to 1 (white) or -1 (black). For a methodological reason that will become apparent later, the prototypes are represented as simple drawings on a 10 x 10 array, as shown in Figure 2. These drawings are supposed to capture an abstract visual representation of the prototypes of the three categories.

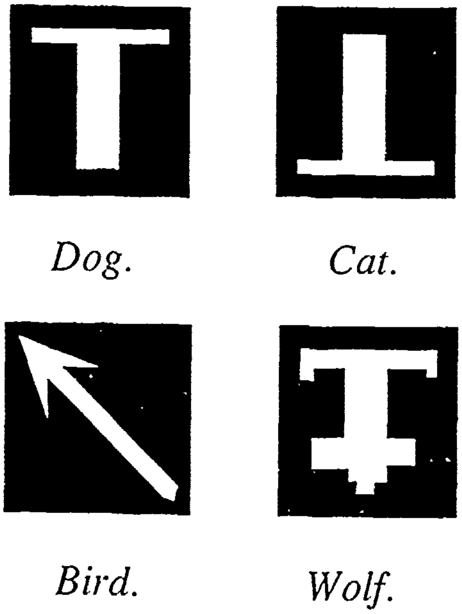


Figure 2: Prototypes of the Categories *Dog* , *Cat*, *Bird* and *Wolf*.

The inter-prototype vector cosines were 0.52 for *dog* and *cat*, 0.16 for *dog* and *bird* and 0.12 for *cat* and *bird*. These values reflect the intuition that the first two categories were fairly similar, and both were somewhat different from the last category (see Figure 2). The new prototype, *wolf*, was more similar to the prototype *dog* than to any other prototype. The inter-prototype vector cosines were 0.5 for *wolf* and *dog*, 0.14 for *wolf* and *cat* and 0.02 for *wolf* and *bird*. To create an exemplar, a prototype was chosen and a noise vector was added to it. The features composing the noise vector were constrained to lie around the contour of the prototype drawings of Figure 2. In addition, each unit with a white, "on," value was turned on with a probability of 0.75 to prevent the category to be defined by *singly necessary and jointly sufficient* features. Given these characteristics, the vector cosine between an exemplar and its prototype varied between 0.72 and 0.92.

2.1.2 Stimuli of the naming module

The f input vectors had dimensionality 132. f', the 10 x 10 conceptual interpretation, was stored in the first 100 units and f'' in the last 32 units. In order to keep the label representation simple, "dog," "cat," "bird" and "wolf" were coded with an ASCII binary code. Each character was represented by its ASCII number on eight units, with 1 and -1 standing respectively for the binary values 1 and 0. When a label had three characters, a hyphen was added to it (e.g., "cat-").

Relearning phase

The network was initially loaded with the concepts of *dog, cat, bird* and their respective labels. The relearning phase was decomposed into two steps. First, for 250 iterations of relearning, the categorizing module was exposed to exemplars of the three "old" categories plus a new one, *wolf*. At that stage, neither new nor old names were associated with the conceptual interpretations. Second, after relearning, the four names, i.e. "dog," "cat," "bird" and "wolf" were associated to conceptual interpretations for 300 iterations.

Testing phase

Two tests were undertaken in two different states of lexical development. First, before any new label was learned, snapshots of the weight portraits and snapshots of the evolution of the conceptual interpretations were taken after 10, 40 and 250 iterations of relearning. The naming performances of the system were also measured after 250 iterations. Second, after the new symbol "wolf" was learned, the capacity to name categories was tested by recording BSB number of iterations to fill f''.

3. RESULTS AND DISCUSSION

In Figure 3, the conceptual interpretation of *wolf's* prototype--the right column--is compared with the conceptual interpretation of *dog's* prototype--the left column--after 10, 40 and 250 iterations of relearning. The activation of the output units of the categorizing module is represented with densities of points where black means no or little activation (0.0-0.2) and white means high activation (0.9-1.0). The coordinate system represents the effective location of each output unit on the conceptual map. The lower right region of the map responds preferentially to instances of *dog* and although it is not shown on the figure, the upper right region responds preferentially to instances of *cat* and the middle left to instances of *bird* (see Schyns, 1990b). Since the prototype of *wolf* is more similar to the prototype of *dog* than to any other, a wolf is interpreted as an instance of *dog* at the beginning of relearning (see Figure 3, the right column, a). If BSB attempts to name this conceptual interpretation, "dog" comes out of the naming module, as expected. This result illustrates the constraints concepts can impose on the interpretation of the outside world. Since the new category has not yet been represented in the system's conceptual knowledge, all of its exemplars are regarded as instances of the closest concept that can interpret the new category. As the network gains more and more experience with *wolf*, it eventually learns how to distinguish this category from the others. This is illustrated in the right column of Figure 3, where we can see a new conceptual region that emerges gradually from the *dog* region to interpret the new category as a separate piece of knowledge.

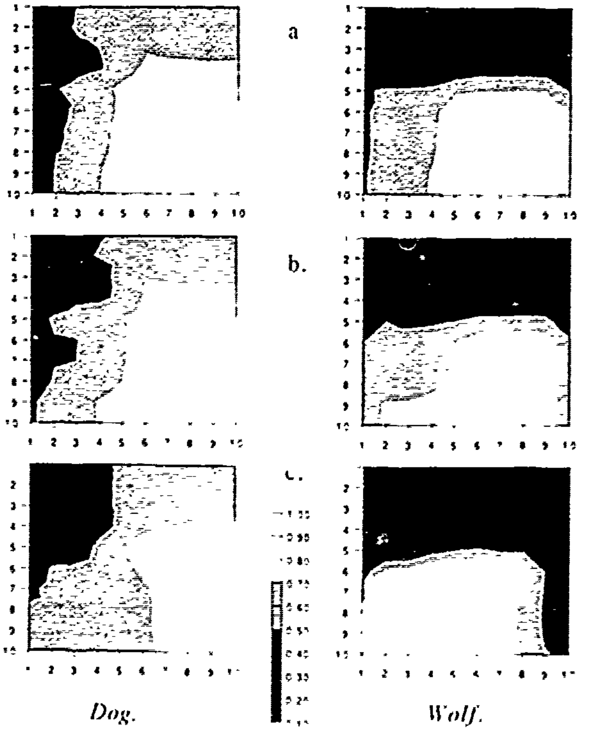


Figure 3. Conceptual Interpretations of the Prototypes of *Dog* and *Wolf* after 10 (a), 40 (b) and 250 (c) Iterations of Relearning.

The weights of the categorizing module play a crucial role in storing knowledge by extracting the structure of the category. To understand how these weights evolve and how they subsequently determine the activation of the conceptual map, snapshots of the weight vector afferent to each output unit were made after 10, 40 and 250 iterations of learning. In figure 4, each weight vector is represented as a 10 x 10 square of pixel values. *The coordinates of the pictorial representation of each vector are exactly its output unit coordinates.* Each of the 10 x 10 connection strengths is coded by a level of grey where black and white mean respectively maximally inhibitory and maximally excitatory connection. After 10 iterations, the new weight vectors overlap with some of the weight vectors that were dedicated to represent *bird* and *dog*. Eventually, as the weights underlying the new conceptual region incrementally pick out the relevant features that characterize the new category, its prototype is represented distinctively. The prototype of *wolf* is represented in the weight vectors of the lower left corner of Figure 4, part (c).

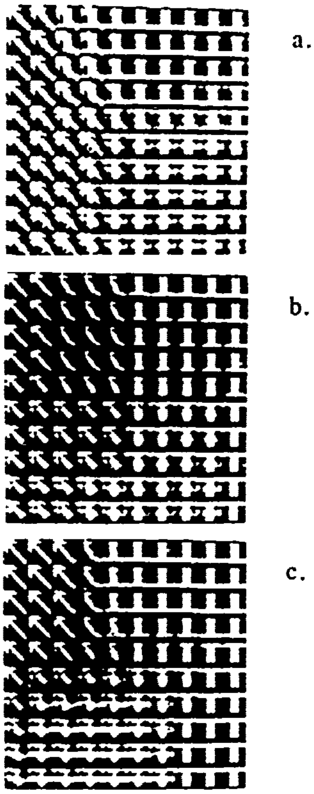


Figure 4. Weight Portraits after 10 (a), 40 (b) and 250 (c) Iterations of Relearning.

It should be noted that the new conceptual region is close to the region that responds preferentially to *dog* (the lower right region of both the conceptual interpretations and the weight portraits of Figure 3 and Figure 4, the left columns, a). This is not accidental. The self-organizing learning algorithm updates the weights by first selecting a winner. We have seen that the region for *dog* is the one that responds the most to wolves in the beginning of learning. Therefore, the weights that will first be updated to represent wolves will be in the *dog* region. With learning, the new region will migrate to a corner where the competition for space allocation is less intense. This kind of migration on the map is due to a *repelling* effect, a by-product of the learning algorithm (see Schyns (1990b) for a more detailed description).

After 250 iterations of relearning, the naming of the conceptual interpretations resulting from the four prototypes gives three labels. Then four category names are provided for 300 more iterations. Importantly, once all labels are learned, a one-to-one relationship between concepts and labels is achieved. The BSB reaction time for each prototype is shown in Table 1. The left part of Table 1 shows the reaction times before the name "wolf' is learned, and the right part shows the BSB reaction times once the four category names are learned.

Table 1: BSB Reaction Times for the Retrieval of Names in the two Conditions of the Experiment.

PROT.	NAME	RT	NAME	RT
dog	"dog"	14	"dog"	20
cat	"cat"	11	"cat"	12
bird	"bird"	8	"bird"	10
wolf	"dog"	27	"wolf"	24

Much like children's overextension errors and their eventual corrections, in this model, when the new category *wolf* is learned, conceptual interpretations of its exemplars are labelled as "dog." At this stage, the lexical item "dog" is overloaded. Its set of referents is overgeneralized since it includes not only the dogs, but also the wolves. However, the conceptual map interprets wolves distinctively from dogs. Therefore, a new label can be associated to something specific, the set of conceptual interpretations for wolves. When this new category term is learned, the initially overextended category name "dog" narrows down to the correct set of referents while "wolf" refers to the the remaining referents as shown on the right part of Table 1.

By explicitly grounding the acquisition of a simple lexicon on the development of conceptual knowledge, the modular architecture of the current research proposes a specific implementation of Chapman et al. (1986) and Quinn and Eimas (1986)'s proposal that a new category term could follow, rather than precede, the acquisition of a new concept in a kind of bottom-up category learning. We saw in this experiment how items of a lexicon could label conceptual interpretations through a disambiguation process. When lexical items couldn't be mapped to concepts with a one-to-one relationship, category terms were overextended to denote different sets of referents. By making the adequate learning and correct usage of a new category term ultimately contingent upon the development of a distinct concept, I provided a possible implementation of the mechanism by which the early lexical categories of children narrow down to the adult ones. These mechanisms also honor Mutual Exclusivity.

In general, Mutual Exclusivity states that children start off with the assumption that each object in the world is referred to by only one label (Markman, 1990). While different names might be applied to the same object in different phases (e.g. an overextended label such as "dog" preceding an appropriate label such as "wolf"), only one name is applied at any one phase. The architecture presented here naturally honors Mutual Exclusivity through the bias of contrasted conceptual interpretations. Category terms are mutually exclusive if they are mapped onto mutually exclusive sets of objects. Thus, non overlapping internal codes provide the representational basis for the mutual exclusivity of category terms. However, Mutual Exclusivity is really honored if the one-to-one association of internal codes to category names is an *internal* constraint of the naming module. By internal, I mean a constraint that ensues solely from the dynamics of the module, as opposed to an external constraint such as a teacher that provides one name per category. If Mutual Exclusivity were violated, i.e. if one code were accidentally associated with two different labels, the naming module wouldn't work correctly. Technically, the autoassociator would most likely learn a category name that results from the linear combination of the two labels associated with the same conceptual interpretation (see Knapp & Anderson, 1984, for the formal development of this argument). Moreover, even if this problem was solved with a more powerful learning scheme, another problem would still occur if Mutual Exclusivity were violated as explained above. Since BSB is deterministic, it could not retrieve one word sometimes and the second word other times given the same conceptual interpretation as initial state. Thus, Mutual Exclusivity is a fundamental internal constraint on the proper working of the naming module presented here.

4. CONCLUSIONS AND FUTURE WORK

The account of concept learning presented here is realized by a two-stage process in which conceptual maps play the role of an "informational relay." This relay could encode graded category membership and thereby reflect the status of an exemplar with respect to conceptual knowledge. I have shown elsewhere that this encoding property would be preserved if the knowledge were hierarchically organized, i.e. the conceptual map reflected the inclusion relationships between concepts (Schyns, 1990b). This code is rich enough to be handled by a minimal naming device such as the pattern associator presented here. In the current experiment, this code has constrained learning in the naming module and provided the representational foundation of a possible account of Mutual Exclusivity

As explained earlier, Mutual Exclusivity is a constraint on word meaning observed in young children. Of course, Mutual Exclusivity needs to be overcome in order to learn, for example, hierarchies of names. Since children eventually outgrow the Mutual Exclusivity Constraint, complete models of human category naming should demonstrate this important development.

This model of concept learning is essentially bottom-up. The encoding of a new category on the conceptual map is determined by the atypical character of its instances. To instantiate top-down category learning, a new category term would constrain the state of activation of the map. This could be easily achieved by clamping a new category term on the f^{*} component of f. After a few iterations, a state of activation $f^{\prime}$ would appear on the map. Then, it suffices to select the winner output unit and update its weight vector, as well as those of its neighbors with equation (2). By so doing, a new class of conceptual interpretation would emerge as a result of top-down constraints. They are top-down in the sense that it is the new category name that forces the encoding of a new category, not an atypical instance as is the case in bottom-up learning. It should be noted that this implementation of top-down concept learning could preserve the property that similar categories are encoded in nearby regions. However, to achieve this result, a reorganization of the coding system would be necessary. This reorganization would result from repeated exposures to exemplars from the different categories. Future research will address this topic.

Acknowledgments

This work has been supported in part by grant MH 41704 from the National Institute of Mental Health. The author would like to thank Jim Anderson, Greg Murphy and Dave Touretzky for helpful comments on the manuscript. The computer simulations were run on the Stellar GS1000 minisupercomputer of the Department of Cognitive and Linguistic Sciences, Brown University.

References

Amari, S. I. (1977). Neural theory of association and concept formation. *Biological Cybernetics*, 26, 175-185.

Anderson, J. A., Silverstein, J. W., Ritz, S. A., & Jones, R. S. (1977). Distinctive features, categorical perception, and probability learning: Some applications of a neural model. *Psychological Review*, 84, 413-451.

Au, T. K., Glusman, M. (1990). The principle of mutual exclusivity in word learning: To honor or not to honor? *Child Development*, 11, 393-416.

Chapman, K. L., Leonard, L. B., & Mervis, C. B. (1986). The effect of feedback on young children's inappropriate word usage. *Journal of Child Language*, 13, 101-117.

Clark, E. V. (1973). What's in a word? On the child's acquisition of semantics in his first language. In T.E. Moore (Ed.), *Cognitive development and the acquisition of language*. New York: Academic Press.

Clark, E. V. (1983). Meaning and concepts. In J. H. Flavell & E. M. Markman (Eds.), *Manual of Child Psychology: Cognitive development* (Vol 3, pp. 787-840). New York: Wiley.

Fodor, J. A. (1983). *The modularity of mind*. Mit Press, Cambridge: MA.

Golden, R. M. (1986). The "brain-state-in-a-box" neural model is a gradient descent algorithm. *Journal of Mathematical Psychology*, 30, 73-80.

Grossberg, S. (1976). Adaptative pattern classification and universal recoding. I. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23, 121-134.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational capabilities. *Proceedings of the National Academy of Sciences*, 79, 2554-2558.

Kawamoto, A. H., & Anderson, J. A. (1985). A neural network model of multistable perception. *Acta Psychologica*, 59, 35-65.

Knapp, A. G., & Anderson, J. A., (1984). Theory of categorization based on distributed memory storage. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 10, 616-637.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43, 59-69.

Kohonen, T. (1984). *Self-organization and associative memory*. Berlin: Springer-Verlag.

Markman, E. M. (1990). The Whole Object, Taxonomic and Mutual Exclusivity Assumptions as initial constraints on word meanings. To appear in J. P. Byrnes and S. A. Gelman (Eds.), *Perspectives on Language and Cognition: Interrelations in Development*. Cambridge: Cambridge University Press.

McClelland, J. L., & Rumelhart, D. E. (1985). Distributed memory and the representation of general and specific information. *Journal of Experimental Psychology: General*, 114, 159-188.

Murphy, G. L. (in-press). Meanings and Concepts. To appear in: *The Psychology of Word Meaning*, P. Schwanenflugel (Ed.). Hilldsale, NJ: Lawrence Erlbaum.

Posner, M. I., & Keele, S. W. (1968). On the genesis of abstract ideas. *Journal of Experimental Psychology*, 77, 353-363.

Posner, M. I., & Keele, S. W. (1970). Retention of abstract ideas. *Journal of Experimental Psychology*, 83, 304-308.

Quinn, P. C., & Eimas, P. D. (1986). On categorization in early infancy. *Merrill-Palmer Quarterly*, 32, 4, 331-363.

Ritter, H., & Kohonen, T. (1989). Self-organizing semantic maps. *Biological Cybernetics*, 62, 4, 241-255.

Rumelhart, D. E. & Zipser D. (1985). Feature discovery by competitive learning. *Cognitive Science*, 9, 75-112.

Schyns, P. G. (1990a). Expertise acquisition through the refinement of a conceptual representation in a self-organizing architecture. *Proceedings of the International Joint Conference on Neural Network*, Washington DC, 1, 236-240.

Schyns, P. G. (1990b). A modular neural network of concept acquisition. Manuscript submitted for publication.

Smith, E. E., & Medin, D. L. (1981). *Categories and concepts*. Cambridge, MA: Harvard University Press.

Von der Marlsburg, C. (1973). Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 15, 85-100.

Widrow, B., & Hoff, M. (1960). Adaptative switching circuits. *IRE WESON Convention Record*, NY: IRE, 96-104.

A Computational Model of Attentional Requirements in Sequence Learning

Peggy J. Jennings and Steven W. Keele
Department of Psychology
University of Oregon
Eugene, OR 97403
pegj@oregon.uoregon.edu

Abstract

This paper presents a computational model of attentional requirements in sequence learning. Cohen, et al. proposed two fundamental operations in sequence learning. An associative mechanism mediates learning of patterns with unique associations (1-5-4-2-3). These associations do not require attention to be learned. Such an associative mechanism is poorly suited for learning sequences with repeated elements and ambiguous associations (3-1-2-1-3-2). These sequences must be parsed and organized in a hierarchical manner. This hierarchical organization requires attention. The simulations reported in this paper were run on an associative model of sequence learning developed by Jordan (1986). The simulations modeled closely the keypressing task used by Cohen, Ivry and Keele (1990). The simulations replicate the empirical findings, and suggest that imposing hierarchical organization on sequences with ambiguous associations significantly improves the model's ability to learn those sequences. Implications for the analysis of fundamental computations underlying a system of skilled movement are discussed.

This paper examines whether a connectionist model of sequence learning developed by Jordan (1986, 1990) provides insight into the attentional requirements of sequence learning as investigated by Cohen, Ivry, and Keele (1990; see also Keele, Cohen, & Ivry, 1990). The first section describes the central phenomena explored by Cohen, et al. The second section describes aspects of Jordan's model and how it relates to the empirically discovered phenomena.

1 BEHAVIORAL CHARACTERISTICS OF SEQUENCE LEARNING

Cohen, et al. adopted a paradigm established by Nissen and Bullemer (1987) to investigate the learning of sequential representations. Successive presentations of a visual stimulus appeared at one of 3, 4, or 5 locations on a screen. Subjects responded to each stimulus by pressing a key corresponding to the stimulus location. Reaction times were recorded. Unknown to the subjects, a large proportion of the successive signals appeared in a particular repeating sequence of locations. One type of sequence, called Unique, involves a sequence of 5 unique signal positions, an example of which is 1-5-4-2-3. The numbers refer to signal positions. After the last position in a cycle, the sequence repeated with no detectable break. Sequences were presented to subjects in 8 blocks of 20 cycles through the sequence. A second type of sequence, called Ambiguous, involves only 3 signal positions. Each position is repeated within the sequence, but each occurrence is followed by a different successor. An example of an Ambiguous sequence is 1-3-2-3-1-2.

Cohen, et al. found that, with practice, subjects learn both of these types of sequences in the absence of a distraction task. Sequence acquisition was demonstrated by steady improvement in reaction time over 8 blocks of training, with a significant increase in reaction time on blocks of trials presented after training in which the signals occurred at random rather than in the structured sequence. When a secondary task was performed simultaneously, diverting attention from the primary reaction time task, Unique sequences were learned but Ambiguous ones were not. That is, performance to signals occurring in the structured Ambiguous sequence never became faster than performance to randomly occurring signals. Something about learning the Ambiguous sequence appears to require attention. Cohen, et al. also observed that if an Ambiguous sequence is altered by replacing one of the repeated events with an event that occurs only once in the cycle, such as 1-4-2-3-1-2 (the underline shows the unique event), subjects learn this sequence at a rate similar to that for the Unique sequences. These sequences are called Hybrid because they involve a mixture of repeated and unique events.

Cohen, et al. offered the following explanation of the effects of sequence structure on learning. The Unique sequences can be acquired by forming associations between adjacent events. Such learning appears to require little or no attention to the relationships among items. Associational learning is not well suited for learning Ambiguous sequences, however, because on different occasions a particular event is ambiguously followed by a different event. Ambiguous sequences can, however, be learned by a mechanism that divides the sequence into parts. This process of hierarchic organization seems to require attention. Why the Hybrid sequences are learned more like Unique than Ambiguous sequences is unclear.

2 A COMPUTATIONAL MODEL OF SEQUENCE LEARNING

The goal of the present research is to gain insight into (1) how the structure of a sequence interacts with component operations of a sequence learning system; and (2) how parsing and hierarchic representation may be implemented in a neural network. These goals were approached using Jordan's (1986, 1990) recurrent network model. His model, as we implemented it, is illustrated in Figure 1. An input layer has units of two types: plan units and context units. These input units are connected to a layer of hidden units, and those hidden units feed into output units that we call prediction units. Activation of these latter units represents a prediction, or priming, of the upcoming event. The prediction units can be viewed variously as representing a prediction of the upcoming stimulus or, because the stimulus determines which response to make, the upcoming response. Stimulus information constitutes the target output that is used to calculate error in the prediction units and forms the basis for learning using backpropagation. Changes in connection weights are made over the course of learning based on the amount of prediction error on each trial.

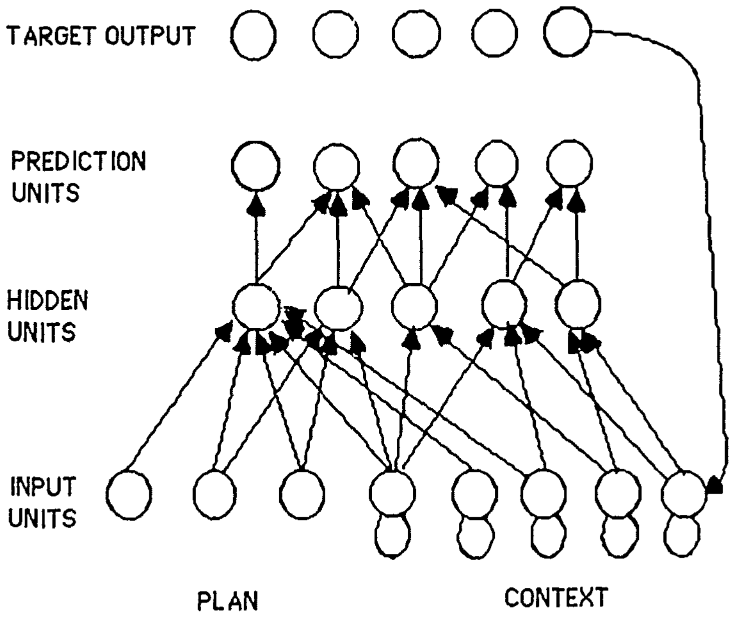


Figure 1: Jordan (1986) Recurrent Network Model. (Not all connections are shown.)

During training, information about the target response feeds back to the context units with a fixed connection weight of 1. Output representation is local rather than distributed, and each target output unit is connected to one and only one context unit. Each context unit also feeds a proportion of its previous activation back on itself. The recurrent connections produce context unit activations that retain a history of recent events. In an

Ambiguous sequence such as 1-3-2-3-1-2, the context following stimulus 3 is somewhat different each time the 3 occurs because in each case it is preceded by different events. The influence of those events, while diminishing over successive stimuli, is partially retained by their representation in the context units.

Sequence learning is achieved as follows: at the beginning of a sequence a pattern of activation appears on the plan units. In the general case, this pattern persists in unmodified form throughout the training session. At the beginning of a block of training, the context units are all set to zero. The combination of plan and initial context values feed through the hidden layer to produce a prediction. A "stimulus", or target response, provides target output information. Prediction error is calculated, and weight changes occur via backpropagation. The "stimulus" information also feeds back to the context units, and the context units reverberate a portion of their previous values. On subsequent iterations, the updated context values in combination with the plan values constitute the input patterns associated with consecutive target responses.

In these simulations, the model was trained on two distinct types of sequence organization. One condition involved presenting 20 cycles through each sequence type with no higher-level organization. In another condition, the input patterns were parsed into distinct subparts. Two implementations of parsing were examined. One implementation involved resetting the context units to zero at the beginning of each cycle through the sequence. This implements a form of parsing whereby the boundaries of a sequence are marked. The second implementation of parsing involved assigning different plan values to each subpart of a sequence. When a subsequence is finished and a new one is to start, the activation pattern on the plan units changes accordingly. In each implementation, parsing is made explicit in the input patterns.

The intent of these simulations is to determine how parsing influences the learning of the three sequence types: Unique, Hybrid, and Ambiguous. Parsing, in this conception, corresponds in the empirical case to an attention-dependent organizational process that mediates learning of Ambiguous sequences.

2.1 SIMULATION 1: ASSOCIATIVE PROCESSES IN SEQUENCE LEARNING

The goal of Simulation 1 was to replicate the behavioral characteristics of sequence learning under divided attention as reported by Cohen, Ivry, and Keele (1990). Under dual-task conditions, performance of Unique sequences (e.g., 1-5-4-2-3) and Hybrid sequences (e.g., 1-4-2-3-1-2) steadily improves with training, while performance of Ambiguous sequences (e.g., 1-3-2-3-1-2) remains no better than performance to randomly presented stimuli.

2.1.1 Training Set

Input to the model was composed of 2 elements. (1) a plan value; and (2) a representation of recent target keypress history. Plan values remained constant throughout the training session for each sequence. Target keypress values were represented by a vector of binary values across 5 units. For example, an output pattern of 00100 represents a press of the middle key. Context values represented the sum of the desired output at time *t* and a proportion of the context value at time *t*-1. Training sets consisted of 20 cycles through a sequence, consistent with a single block of training in the empirical task. Each input pattern is associated with a desired output value corresponding to a keypress. At the beginning of a block of training, context units were set to zero. Performance measures consisted of (1) total sum of squares of prediction erro. summed over individual trials and recorded after each block of training; and (2) number of blocks required to learn the sequence to specified accuracy criterion (less than 0.04 sum of squares prediction error over 20 cycles). The model was trained on three exemplars of each sequence type: Unique (1-5-2-4-3; 1-4-5-3-2; 1-3-4-2-5), Ambiguous (1-2-3-1-3-2; 1-2-3-2-1-3; 1-3-2-3-1-2), and a Hybrid of unique and ambiguous associations (1-2-3-1-3-4; 1-2-3-2-4-3; 1-4-2-3-1-2). Six separate training sessions per sequence type were run; two sessions per sequence. Mean values from sequence type groups were analyzed.

2.1.2 Results of Simulation 1

An analysis of variance of number of blocks required to learn the sequence to criterion showed a significant effect of sequence type (F(2,15) = 19.742, p <= 0.001). Multiple comparisons (Tukey, p < .05) reveal that the model requires fewer blocks of training to learn the Unique sequences (mean blocks = 62.2) and the Hybrid sequences (mean blocks = 68.0) than to learn the Ambiguous sequences (mean blocks = 106.0). The Unique and Hybrid conditions are not significantly different.

Figure 2 displays the rate of change in prediction error over the first 10 blocks of training. While learning the Unique sequences, the reduction in prediction error is rapid and steadily decreasing with training, approaching zero. While learning the Amibiguous sequences, however, prediction error levels off at a relatively high value after an initial brief and rapid decline. These error measures may be compared with improvement in reaction time reported in the empirical case, resulting from the subject's increasing ability to correctly anticipate events. In the Ambiguous condition, prediction error as well as empirical reaction times remain relatively high as anticipation of response is not much better than chance.

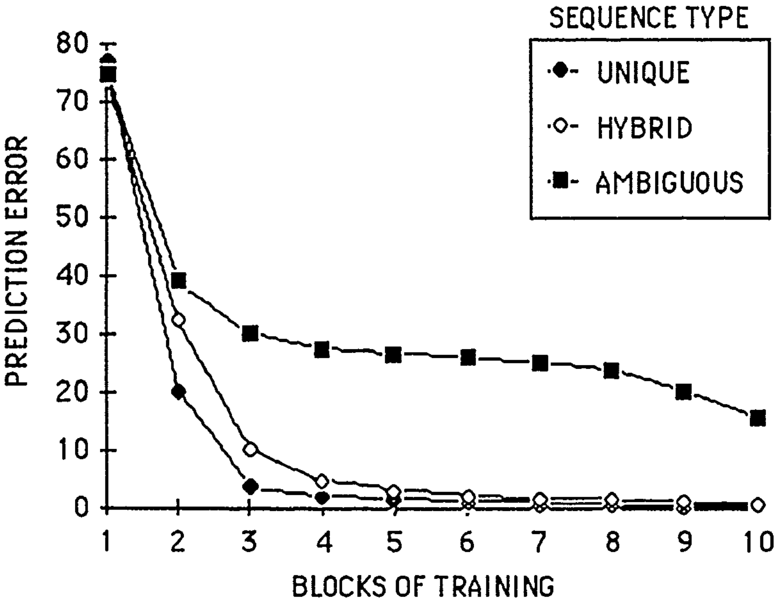


Figure 2. Prediction Error as a Function of Training, by Sequence Type.

2.1.3 Discussion

Learning in the simulation is denoted by reductions in error between prediction values and target response values while learning in humans is documented by reductions in reaction time. Human reaction time is improved by priming or expectancy. Model error as well as empirical reaction times, because block means are reported, reflect accuracy of prediction over the course of 20 cycles through a sequence.

In this simulation, both Unique and Hybrid sequences were learned at essentially the same rate. These results agree qualitatively with the results of human learning in which learning of Unique and Hybrid sequences also occurs at similar rates when a distraction task is used. In contrast to the equivalence of Unique and Hybrid sequence learning rates, the model learns the Ambiguous sequences more slowly. This result is qualitatively similar to the empirical result in which a distraction task prevents learning of the Ambiguous sequences. It must be noted, however, that Ambiguous sequence learning does occur in the simulation. For the Unique and Hybrid sequences, the model required over 50 blocks to learn the sequences, while human subjects show learning within 8 blocks of training. It is not clear how the time frames between the model and the human are related, but given that the model required 106 blocks to learn the Ambiguous sequence, it is possible that humans may also show evidence of learning the Ambiguous sequences with more extensive practice.

2.2 SIMULATION 2 -- HIERARCHICAL ORGANIZATION OF INPUT SEQUENCES

Cohen, et al. proposed that the role of attention in sequence learning is to mediate parsing of sequences containing ambiguous associations. The goal of Simulation 2 was to examine the effects on learning of imposing an hierarchical organization on the input sequence. The associative learning mechanism implemented in Simulation 1 required significantly more training (106 blocks) to learn the Ambiguous sequences than to learn the Unique and the Hybrid sequences (65 blocks). Preorganizing the input patterns into parsed subsequences should allow the associative mechanism to now learn the Ambiguous sequences as quickly as it learns Unique and Hybrid sequences. Because the associative mechanism is already efficient at learning Unique and Hybrid sequences, performance on them should remain virtually unchanged by delineating boundaries in the sequence patterns.

2.2.1 Training Set

The training sets in Simulation 2 were similar to those used in Simulation 1 except that, in Simulation 2, parsing was imposed on the input patterns. Parsing was achieved in two alternative ways. In one condition (Simulation 2.1), context unit activation was set to 0 at the beginning keypress of each cycle of the sequence within a block of 20 cycles. Unique, Hybrid, and Ambiguous sequences were parsed in this way and presented to the model. In this condition, the internal structure of the sequence is left intact, but the beginning of each cycle is delineated by a change in context. Here parsing is achieved by making explicit changes in activation values of the context units, while leaving plan unit values constant. In another condition (Simulation 2.2), which affected only the Ambiguous sequences, parsing was achieved by modifying the value of the plan units at the beginning of each subsequence within the Ambiguous sequence. For example, the sequence 3-1-2-1-3-2 had a plan value of 110 for the 3-1-2 subsequence, and 111 for the 1-3-2 subsequence. During training, therefore, the subplan values alternated as the sequence progressed. In this condition, the Ambiguous sequence is represented as two alternating subsequences. Here parsing is achieved by making explicit changes to the plan units, while leaving the context unit values dependent on model function.

2.2.2 Results of Simulation 2.1

In the first parsing condition, resetting context values to zero, the Ambiguous sequences were learned as quickly as were the Unique and Hybrid sequences (mean blocks 50.8, 60.7 and 57.0, respectively). The three sequence types are not significantly different (F(2,15) = 1.483, p< 0.26).

An analysis of variance of the number of blocks of training required to learn the sequences in both Simulation 1 (unparsed) and Simulation 2.1 (parsed, zero context) revealed significant effects of sequence type (Unique, Hybrid, or Ambiguous) (F(2,30) = 7.991, p < 0.002), sequence organization (unparsed vs. parsed) (F(1,30) = 33.659, p < 0.000), and a significant interaction of sequence type and organization (F(2,30) = 18.088, p< 0.000). Unique and Hybrid sequences are learned more quickly than Ambiguous sequences (mean blocks = 61.4, 62.5, and 78.4, respectively). Parsed sequences of all three types are learned more quickly than unparsed sequences (mean blocks = 56.2 and 78.7, respectively). The main effects of sequence type and organization, as well as the interaction, are strictly the result of improvement in learning rate for the Ambiguous sequences in the parsed condition over the rate in the unparsed condition. Learning rates for Unique and Hybrid sequences are unimproved by parsing.

Results from post hoc comparisons (Tukey, p < .05) of learning rates for the three sequence types in the parsed and unparsed conditions support the hypothesis that unparsed Unique and Hybrid sequences as well as parsed Unique, Hybrid, and Ambiguous sequences are all learned at the same rate (mean blocks = 59.74). Only unparsed Ambiguous sequences are significantly more slowly learned (mean blocks = 106.00). The groupings are as follows:

GROUP A: Unparsed (Unique, Hybrid), Parsed (Unique, Hybrid, Ambiguous)

GROUP B: Unparsed (Ambiguous)

2.2.3 Results of Simulation 2.2

In the second parsing condition, involving alternating subplans in Ambiguous sequences, the parsed sequences were learned in 55.3 blocks. An analysis of variance of blocks to learn the Ambiguous sequences under three types of organization (unparsed, parsed (zero context), and parsed (alternating subplans)) reveals a significant effect for organization (F(2,15)=24.958, p<0.000). Post hoc comparisons show that the parsed sequences are learned at the same rate, while the unparsed sequences require significantly more exposure to the sequence to learn it. In this case, the particular implementations of parsing are not significantly different. Figure 3 summarizes the results of Simulation 2, displaying the rate of improvement in model prediction as a function of training for parsed and unparsed Ambiguous sequences. The learning rate for both implementations of parsing is similar to that required to learn the parsed Unique and Hybrid sequences, while the learning rate for the unparsed Ambiguous sequence remains at a relatively high level of prediction error.

2.2.3 Discussion

Each particular method of implementing parsing had the same effect on sequence learning. When parsed in either of these two ways, Ambiguous sequences are learned by an associative mechanism at a rate similar to that required to learn Unique and Hybrid sequences. These results agree with the empirical results in which removal of distraction (presumably allowing an attention-dependent parsing process to function) enables, or at least greatly enhances, learning of the Ambiguous sequences.

3 GENERAL DISCUSSION

The two forms of parsing used in the simulation correspond to two different mechanisms, both of which could be involved in human sequence learning. One form of parsing involves resetting context units to zero at the beginning of each cycle of the sequence while leaving plan values intact. This form of parsing may be seen as mimicking a working memory function. The sequence is represented by activation across the plan units, but learning is facilitated by identifying the boundaries of the sequence. When a cycle of the sequence ends, plan values denote that the same sequence will follow, but the change in context values provides information about where the sequence begins and ends as well as indirect information about the length of the sequence.

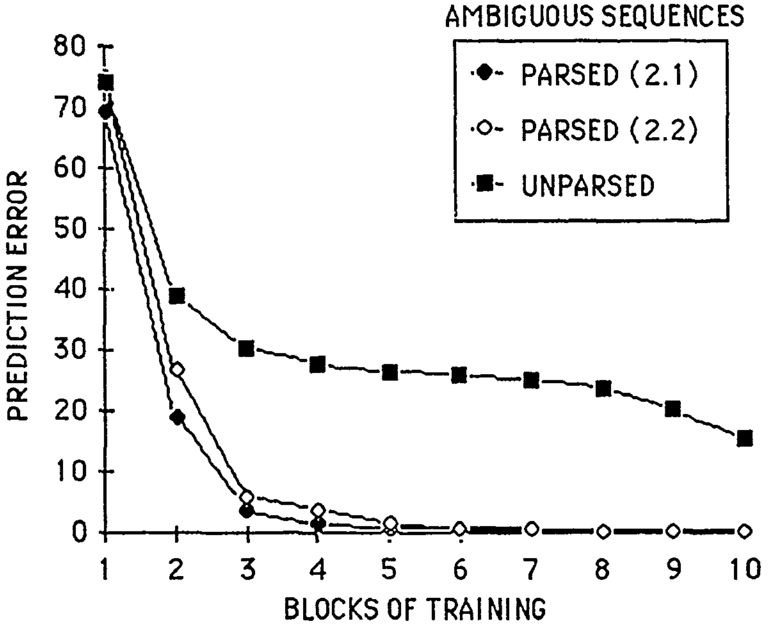


Figure 3. The Effect of Sequence Organization on Learning Ambiguous Sequences

The second form of parsing was one in which plan values were altered to represent sequence subparts, while leaving the context units to function uninterrupted. This notion of parsing is somewhat closer to traditional notions of hierarchic representation in which a node stands as a symbol for a more complex and lower order sequence of events. It may be seen as simulating the function of higher level object recognition processes. The alternating subplans represent the final products of a mechanism which has identified the subparts of a sequence whole.

The components of the Jordan model provide further insights into the various computations involved in learning complex motor sequences. The foundation for this learning system is the associative mechanism represented in the model by connections strengths between input patterns and response predictions. A simple two-layer system is fully capable of learning the Unique sequences in these simulations. Hidden units mediate higher level abstractions of input patterns. The hidden units perform an intermediate synthesis of input features. Context units function in a way that is analogous to visuospatial working memory. And finally, plan units reflect the contributions of representational or symbolic processes.

The model used here to simulate sequence learning represents the major components of a sequence learning system. Some of these component processes are dependent on attention, while others are not. It appears that various types of motor sequences require differing levels of interaction with these representation and memory components depending on the structure of the sequence. Unique sequences may require only a strict associative mechanism. Hybrid sequences may require association and synthesis as well as working memory capacity. Ambiguous sequences may need the full system of association, synthesis, working memory, and symbolic representation. These simulations have shown that learning of ambiguous associations are facilitated by higher level processes of organization. Facilitation of learning may occur at the level of working memory by providing cues to pattern boundaries. Facilitation may also occur as the result of an attention-dependent process which parses input sequences into subparts in a hierarchical representation. These effects of different parsing implementations provide insight into the reason that Hybrid sequences are learned by human subjects and by the simulation at a rate similar to that required to learn Unique sequences. The unique elements in the Hybrid sequence may serve to provide cues to boundaries within the sequence. These cues may function at the level of working memory and are independent of attention. This would allow Hybrid sequences to be learned during distraction, as the empirical data show. The Ambiguous sequences contain no clues to pattern boundaries, so they must be analyzed into subparts to be learned. This hierarchical organization process requires attention. The empirical data show that human subjects are able to learn the Ambiguous sequences when attention is not distracted to another task, but do not show evidence of learning the Ambiguous sequence under dual-task conditions. Future simulations will involve examining further the nature of these interactions between sequence structure, attention, and the fundamental operations that underly sequence learning.

Acknowledgements

We are grateful for the financial and moral support from the Office of Naval Research (Contract N00014-87-K-0279) in the conduct of this research. We have benefitted greatly from discussions with Asher Cohen, Richard Ivry, and Clifford Keele.. We extend our thanks to Jay McClelland for making available a test version of Macintosh software on which these simulations were run. Discussions with Jim Tanaka during the early work on these simulations were helpful and very much appreciated. And thanks to Mike Posner for comments and discussions that are forever thought-provoking and insightful.

References

Cohen, A., Ivry, R. I., & Keele, S. W. (1990). Attention and structure in sequence learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 16,* 17-30.

Jordan, M. I. (1986). *Serial order: A parallel distributed processing approach* (ICS Report 8604). La Jolla, CA: Institute for Cognitive Science, University of California, San Diego.

Jordan, M. I. (1990). Learning to articulate: Sequential networks and distal constraints. In M. Jeannerod (Ed.), Attention and performance XIII. Hillsdale, NJ: Erlbaum.

Keele, S. W., Cohen, A., & Ivry, R. I. (1990). Motor programs: Concepts and issues. In M. Jeannerod (Ed.), *Attention and performance XIII.* Hillsdale, NJ: Erlbaum.

Nissen, M. J., & Bullemer, P. (1987). Attentional requirements of learning: Evidence from performance measures. *Cognitive Psychology, 19,* 1-32.

Recall of Sequences of Items by a Neural Network

Stefano Nolfi
Inst. of Psychology
National Research Council
Rome-Italy

Domenico Parisi
Inst. of Psychology
National Research Council
Rome-Italy

Giuseppe Vallar
Inst. of Neurology
University of Milan
Milan-Italy

Cristina Burani
Inst. of Psychology
National Research Council
Rome-Italy

Abstract

A network architecture of the forward type but with additional 'memory' units that store the hidden units activation at time 1 and re-input this activation to the hidden units at time 2, is used to train a network to free recall sequences of items. The network's performance exhibits some features that are also observed in humans, such as decreasing recall with increasing sequence length and better recall of the first and the last items compared with middle items. An analysis of the network's behavior during sequence presentation can explain these results.

1 INTRODUCTION

Human beings possess the ability to recall a set of items that are presented to them in a sequence. The overall capacity of the memory systems used in this task is limited and the probability of recall decreases with increasing sequence length. A second relevant feature of human performance in this task is that the last (recency effect) and the initial (primacy effect) items of the sequence tend to be recalled better than the middle items. These serial position effects have been observed both in a free recall condition, in which subjects may recall the stimuli in any order they wish, and in a serial recall condition, in which subjects must preserve the presentation order. (See reviews concerning free and serial recall of sequences and the recency effect in: Glanzer, 1972. Crowder. 1976; Baddeley and Hitch, 1977. Shallice and Vallar, 1990).

In this paper we report the results of a simulation experiment in which we trained neural networks to recall sequences of items. Our purpose was to explore if a particular network architecture could function as a memory store for generating free recall of sequences of items. Furthermore, we wanted to determine if the recall performances of our networks exhibited the two features of human free recall that we have mentioned, that is, decreasing probability of recall with increasing sequence length and an U-shaped recall curve (for related works see: Schneider and Detweiler, 1987; Schreter and Pfeifer, 1989; Schweickert, Guentert and Hersberger, 1989).

2 THE SIMULATION

The type of network that we have used (Figure 1) is a feed-forward network with a layer of hidden units intermediate between the layer of input units and the layer of output units. However, the network has an additional set of units, which we call 'memory units (Jordan, 1986, Elman, 1988), one for each of the hidden units. Each memory unit receives a connection link with a fixed weight of 1 from its corresponding hidden unit and it sends connections with learnable weights to all hidden units. When a stimulus is applied to the input units, the input pattern is elaborated (transformed) into a pattern on the hidden units through the connection weights from the input to the hidden units. Then, this internal representation is transformed into the output pattern through the connection weights from the hidden to the output units. This is what normally happens in all feedforward networks. However, in networks with memory units, the activation pattern on the hidden units which is the internal representation of the stimulus is copied in the memory units and kept there until the successive stimulus is input to the network. When the new stimulus is applied to the input units, its internal representation on the hidden units depends on both the stimulus itself and the internal representation of the preceding stimulus which is stored in the memory units. This stored pattern acts as an additional input determining the internal representation of the new stimulus. As the connection weights from the input to the hidden units transform the input pattern of the actual stimulus, in the same way the connection weights from the memory to the hidden units trasform the internal representation of the preceding stimulus stored in the memory units. Hence, how the network reacts to the new stimulus, i.e. the output resulting from the new input, is a function of both the new input and the preceding context. The memory process is a continuing one. Since the internal representation of the new stimulus is also copied in the memory units (replacing the activation pattern which was the internal representation of the first stimulus), it will influence how the network will react to the third stimulus, and so on. In other words, what is stored in the memory units at any given time is a compressed trace of all preceding stimuli and this compressed trace of the preceding stimuli influences the manner in which the network reacts to each new stimulus.

The architecture of a network with memory units can be used in a number of different tasks. For example, it can be used to train networks to predict the next item in a sequence of items if the network has been repeatedly exposed to sequences of a certain type (Elman, 1988). Or, it could be used to make the reaction of a network to a given item sensitive to the preceding context, as it seems to be necessary for language understanding. We have used the architecture to simulate free recall of sequences of items. When a system is given a sequence of items as input and it is asked to recall all the items of the sequence after the last item has been presented, the reaction to the last item must be very context-sensitive in the sense that the system must react to the last item by giving all the items of the sequence that have preceded it in addition to the last item. In order to obtain that networks must be trained in the particular task. We have used the backpropagation procedure (Rumelhart, Hinton, and Williams, 1986) for training our networks. After each successive item of a sequence the network is told which is the desired output, i.e. the set of items that have presented up to that point. These items should appear in the network's output at that time. The network compares the desired output (the items to be recalled up to that point) with its actual output (the actually recalled items) and uses the resulting discrepancy to change the connection weights in the direction of better recall. Notice that the backpropagation procedure will change not only the connection weights from the input to the hidden units and from the hidden to the output units, as is usual in feedforward networks, but also the connection weights from the memory to the hidden units. (Remember that the memory units function as an additional set of input units to the hidden units.) By progressively adjusting the connection weights the network learns to construct memory traces that compress information about an increasing number of items (depending on sequence length) in such a way that from these traces the actual items constituting a sequence can be recovered (recalled).

The network's architecture is shown in Figure 1. The network has 5 input units and 20 output units. The number of hidden units varied in three different simulations from 11 to 15 to 19. The memory units were the same number as the hidden units.

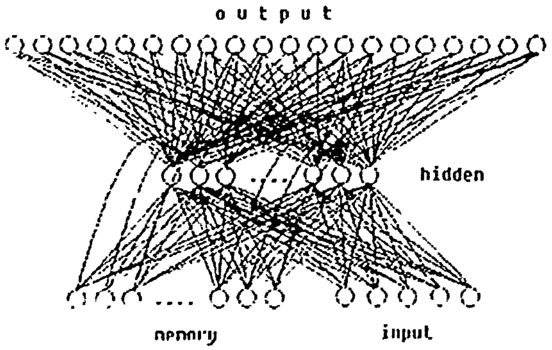


Figure 1: The Network

Each item of the sequence to be recalled is represented in input as a distributed activation pattern on the set of 5 units, and is represented localistically in output as the activation of a single specific unit out of the total 20 output units. We used a universe of 20 different items that are conventionally identified with letters. Our input sequences were composed of 7 items selected from the universe of 20 letters. Tables 1 and 2 show how the 20 letters are represented at the input level and at the output level.

Table 1: Input Patterns

A	=	0 0 0 0 1	M	=	0 1 1 0 0
B	=	0 0 0 1 0	N	=	0 1 1 0 1
C	=	0 0 0 1 1	O	=	0 1 1 1 0
D	=	0 0 1 0 0	P	=	0 1 1 1 1
E	=	0 0 1 0 1	Q	=	1 0 0 0 0
F	=	0 0 1 1 1	R	=	1 0 0 0 1
G	=	0 1 0 0 0	S	=	1 0 0 1 0
H	=	0 1 0 0 1	T	=	1 0 0 1 1
I	=	0 1 0 1 0	U	=	1 0 1 0 0
L	=	0 1 0 1 1	V	=	1 0 1 0 1

Table 2: Output Patterns

A	=	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
B	=	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
C	=	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
D	=	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
E	=	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
F	=	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
G	=	0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
H	=	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
I	=	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
L	=	0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
M	=	0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
N	=	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
O	=	0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
P	=	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0
Q	=	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
R	=	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0
S	=	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
T	=	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
U	=	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
V	=	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

When a sequence of letters is given in input to the network, first the activation level of the memory units is set to 0 and the activation level of the input units is set equal to the input pattern corresponding to the initial letter of the sequence. Following a spreading of activation, a pattern of activation appears on the output units. Then the activation level of the input units is modified according to the second input pattern (letter) and there is a new spreading of activation. After the last item of the sequence has been processed, the network is said to have recalled the sequence if its output units show a pattern of activation in which the units corresponding to the letters of the presented sequence are on and the other units off.

For example, if we present to the network the sequence of 7 letters TGEMBAD, after the last spreading of activation we want the activation pattern shown in Table 3 on the output units (since the activation values of the output units are continuous, the items corresponding to the 7 most activated units are considered as the items recalled by the network):

Table 3: Example of Output Pattern

1 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0

If the network produces the output pattern shown in Table 4, instead of the pattern of Table 3, this means that the network has recalled the sequence TGEMBVD instead of TGEMBAD (i.e. it has recalled the letter 'V' instead of 'A').

Table 4: Example of Output Pattern

0 1 0 1 1 0 1 0 0 0 1 0 0 0 0 0 0 1 0 1

How can the network remember the letters of an input sequence? When the network processes the first input pattern (letter) the input units and the weights on the connections between the input and the hidden units generate a particular pattern on the hidden units. This pattern is stored (copied) in the memory units due to the connections with fixed weight 1 from the hidden units to the memory units. When the input pattern corresponding to the second letter is activated, the activation pattern that is generated on the hidden units depends on both the new letter and the previous pattern stored in the memory units which send connections with learnable weights to the hidden units. This new pattern on the hidden units is stored in the memory units and it will influence the network's reaction to the third item, and so on. In this way many input patterns (letters) are stored as a single pattern in the network's memory units. This pattern itself, when the sequence ends, determines the final activation state of the output units from which it can t. ascertained how many letters in the sequence have been remembered.

The weights of the connections among the units (with the exception of the fixed weights from the hidden units to the memory units) are learned by back-propagation. The network starts with random weights but it is trained with a set of letter sequences as input and a set of desired outputs (correct recalls) as teaching input. During the training the weights are modified, using the teaching input information, in the direction of improved performance. After the training, the network is tested for its recall of the particular sequences it has experienced during the training (old sequences) and also of new sequences that were not included in the training set (new sequences).

During training we presented to the network 87 sequences of 7 randomly selected letters (see Table 5) for 500 times, and we asked the network, after each spreading, to recall all the letters of the current sequence which had been presented up to that point. For example, for the sequence DELNGAII, after presentation of the first letter, we gave the network a teaching input of 1 for the output unit corresponding to the letter 'D' (the fourth output unit) and a teaching input of 0 for all the other output units. Then, after the presentation of the second item, we gave a teaching input of 1 for the two output units corresponding to the letters 'D' and 'E' and a teaching input of 0 for all the other units, and so on.

Table 5: Training Sequences

ECAIVRU	HONGVTA	BARIDEQ
LPFONST	TSRGBQN	OMDFLSE
LTBPNFV	UDESGRH	RVNDELM
ICGASBP	EBQINPG	MHCTDVF
VEIDMLB	UGTFHQO	IHELOUC
TMAQSPB	FRBHTOL	MDPAGEN
REGVLHP	NOTFSBQ	UVSTMQC
OLDEIHF	AQUMFHG	BONRICF
DTGOVMP	FLSERIQ	MRCQUGF
AITPLBS	ASOCRIB	FLPGOHM
OBCVLDT	EFABGHI	SEHTMIU
PQBRCGF	UCRHINQ	TOSLGPM
OCUAMLB	FTDESPH	MOGSCAL
BTHVEFD	QINEUFR	LCSMPVH
SFHANLR	QTDOEMG	CSULOID
EQFGPMN	BSCVTDP	ILRNMOQ
MVAEIBC	CDSLGHR	VOBCQGM
UDHAFTS	HEVFUCG	QSRMPIO
QBANUOR	GCSLDVM	PQDMGEB
LNSFRUI	CQPTDSI	FHGVAOR
LNTFBPS	VODMIUE	FPMVBAU
NCGTOEL	AGUDHQT	BSNROCE
SLGOMHR	PQCIDFE	MQPAOBG
CHFTDEU	TNLESRI	OAMBHGF
EAQUMON	LICHDVP	ODNABMU
FECTSPG	BNMVGIT	CAQPHEF
QEHLCPO	GSDBAVN	LHTGSOR
VNMIEUB	PUONQDA	RUICVNA
DRLUANV	NUVQPTA	HRIUTAV

Three different simulations were run, one with 11, one with 15, and one with 19 hidden units. The networks used in the three simulations all have 5 input units and 20 ouput units. A learning rate of 1.0 and the same training set of sequences were used in all three simulations. After learning, the networks were tested with an additional set of 87 sequences of 7 letters also randomly selected from the universe of 20 letters (see Table 6).

Table 6: New Sequences

IFSOENA	ULBCHTD	UPRVCLO
QHEGFMI	MESFUHQ	ANPIGBO
HMBSQLN	TCGAFOV	HSMCQTR
PBIONGV	DPOHENV	GALTSMQ
SLEBADI	QOCTUGF	VDOTPUQ
REGINLH	CEOLDNS	GFIPRMQ
ABOVMQC	DFEHTLN	FRTMBAI
NPEULCH	LPDGHQR	VFNEASC
CSLFIDU	AVRQNOP	IDHMFEL
USQRBTG	QPCIDVR	HFGOMUS
DMGUVNH	ELROQFI	GCDRLSO
PHTBMIQ	EQHRMOC	LTDIFVG
GMOUTBD	PERIQFN	COQPUSA
GITBDNR	USTHRQC	OMLDIGE
DILMVTP	ERUCBNQ	BLTSCIV
EPFUMGH	TOHQMIP	SUCFDLG
UARHTMN	SLPCGOI	FCTPBAV
MOGULDI	PDRFTUE	LNQMHSI
PAIRCUV	GSETFMH	ESPLFOU
IGNTBVC	OUPQASN	FHDLEMG
MABVICR	DETOFHS	OASFBEI

GNCPLMD
ATFOMQB
IAUCQST
BREUSIQ
VFBGIUE
NVADGRB
HDVAPRL
HQVNRAS
RLFSVGD
CNEGUIL
TRNSBDC
DLONMFH
NGAVOPM
RVAESBT
ATVDHCB
BANEPVU
MNQHPAO
DPMBVFN
QHFIVOM
LTDCOPQ
BQVEOCF
LGUARHT
HBLVEUA
SUPNARM

3 RESULTS

All the three networks (with 11, 15, and 19 hidden units, respectively) learned to recall the letter sequences they saw during the training and were also able to generalize this ability to new sequences they had never seen before.

The global performance of the three simulations is shown in Figure 2. The Figure shows the percentage of letters correctly recalled by each network when tested with old and new sequences (chance level is 35%).

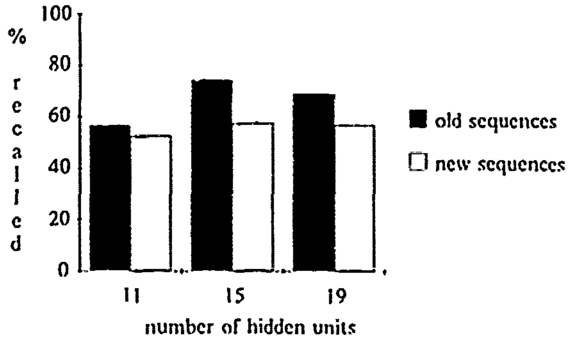


Figure 2: Total Recall of Old and New Sequences as a Function of Number of Hidden Units

As the Figure shows, the networks with 15 and 19 hidden units yielded the best results for the training sequences, with the network with 15 units being slightly better than the network with 19 units. This seems to show that there is an optimal number of hidden units and that adding more hidden units does not improve performance. However, the results for the new sequences are almost identical for the three networks.

If we consider the recall performance for letters in different sequential positions we obtain the results shown in Figure 3. The Figure gives the percentage of letters correctly recalled in new sequences as a function of the serial position of the letter in the sequence and of number of hidden units.

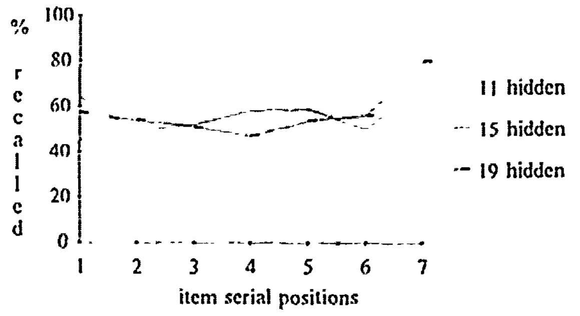


Figure 3: Recall of Items in Different Serial Positions

An analysis of variance was performed on recall of new sequences. The analysis of variance included two factors. The first factor (Letter position) had seven levels (from first through seventh position). The second factor (Number of hidden units) had three levels (11 vs 15 vs 19 hidden units). There were 87 sequences for each type of network. The analysis was performed taking sequences as source of variation. The results show a significant effect of letter position (F (6,1806) = 17.13, p < .001), but no effect of number of hidden units (F (2,1806) = 1.51, p = .218). The interaction between Letter position and Number of hidden units was significant (F(12,1806) = 3.86, p < .001)

These results show that the probability for a letter in new sequences to be correctly recalled varies as a function of the letter's serial position in the sequence. More specifically, the last letters of a sequence seem to be better recalled than the letters in the middle part of the sequence. The first letters appear also to be somewhat better recalled than the middle letters.

The data shown in Figure 3 represent the percentage of letters correctly recalled by the network at the end of a sequence, i.e. after the last letter has been presented. However, unlike what can be done with real human subjects, we can also look at a network's performance during the presentation of a sequence by testing how much the network recalls of the presented letters after 1, 2, 3, 4, etc., letters have been presented. In a sense, it is as if we had done a number of different experiments with letter sequences of different length.

One type of result that we can obtain from this type of analysis is a global measure of recall for sequences of increasing length. The result, shown in Figure 4, is that total recall decreases with increasing sequence length.

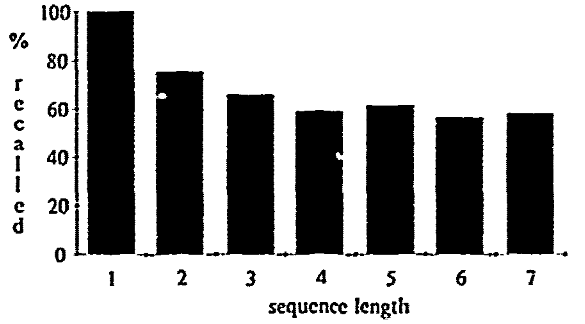


Figure 4: Total Recall for Sequences of Increasing Length

Another result concerns the shape of the recall curve during the course of sequence presentation. Figure 5 shows the percentage of letters correctly recalled in new sequences as a function of the serial position of the letter in the sequence for the 15 hidden network (i.e. the network with the best performance), when the network is tested after 1, 2, 3, ... 7 letters respectively have been presented. Curves have different lengths because we are considering only the letters already presented. For example, the curve of length 2 represents the percentage of letters correctly recalled after the first 2 letters of each sequence have been presented.

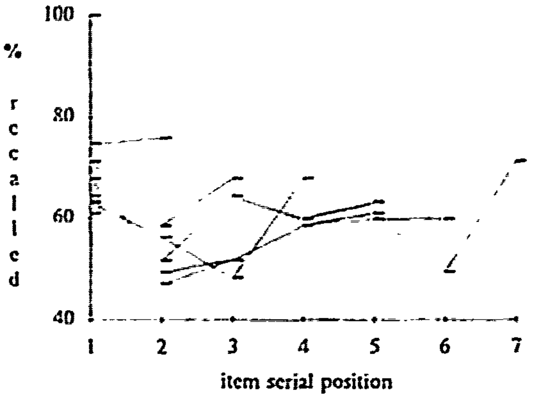


Figure 5: Recall of Letters in Different Serial Positions When Performance is Tested After Each New Letter

As we have said, the data in Figure 5 refer to the 15-hidden units network. However, similar results were obtained with the 11- and the 19-hidden units networks. The results appear to show that, even during the course of sequence presentation, the first and the last letters presented are better recalled than the other letters. For example, after presentation of the 6th letter (see the curve that stops at the 6th position), the sixth letter and the first one are better recalled than the others. Then, after the presentation of the seventh letter (see the curve that stops at the 7th position), the percentage of recall of the 6th letter goes down and the seventh and first letters become the best recalled. We can conclude that the U-shaped recall curve which is obtained at the end of a sequence presentation is constructed dynamically during the course of the presentation of the sequence and that the partial recalls also have an U shape.

The data reported in Figure 5 can be analyzed from another perspective in order to clarify the recall process during sequence presentation. We can determine how the n-th letter of a sequence is recalled after an increasing number of subsequent letters have been presented. For example, we may want to know how the first letter in a sequence is recalled after the first letter is presented, then after the second letter, after the third letter, and so forth. The results ar- shown in Figure 6.

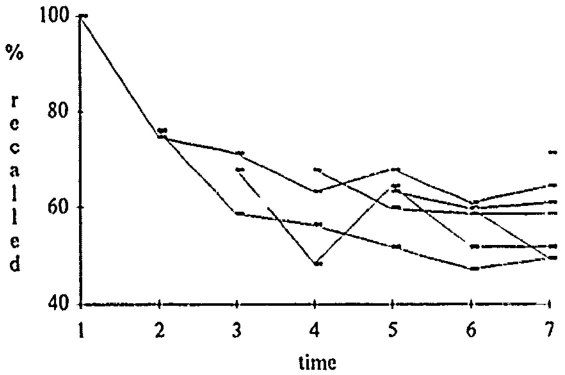


Figure 6: Recall of Letters in Different Serial Positions After an Increasing Number of Successive Letters has been Presented.

In this case, the curve of length 7, which starts at time 1, represents the percentage of correct recall of the first letter after just the first letter has been presented (100% recall), and then after the second, the third, until the seventh and last letter, has been presented. The curve of length 6, starting at time 2, represents the percentage of correct recall of the second letter after presentation of the second letter, after the third, and sɔ on until the last letter has been presented.

Figure 6 is based on data of the 15-hidden unit network but in this case too similar results are obtained for the other two networks. The figure ʰows two results. The first result is that letters are best recalled immediately after presentation and then their recall decreases as more letters are subsequently presented. For example, the 4th letter goes down from a percentage of 67,8 correct recall immediately after the 4th letter has been presented to a percentage of 58,6 three letters after, i.e. after the last letter has been presented. (Cp. the curve that starts at position 4.) This indicates that letters in the middle positions which are not well recalled at the end of a sequence, may be better recalled during sequence presentation. It is the subsequent occurrence of other letters that interferes with their recall at the end of the sequence. The second result is that letters in the first positions of a sequence are better recalled than letters in the last positions when they are first presented. For example, the very first letter is perfectly ıecalled (100%) just after presentation whereas the second letter has a recall percentage of 75.9% and the third of 67.8% in the same conditions. This shows that letters start with a better recall if they are not preceded by too many other letters.

4 DISCUSSION

We have been able tc ،nstruct a neural network that can free recall a sequence of items that the network has been given as input. After training, during which the network processes a list of sequences and a corresponding list of correct answers, the network is able to correctly remember (to some extent) new sequences which it has never seen before. In other words, the network developed a general ability to recall sequences of items. This general ability is incorporated in the weight matrix that has been modified during learning but is fixed when the network is storing the inᶠormation necessary to recall a new sequence of items.
In order to recall sequences of stimuli the network ،ıust be able to acquire various abilities:

(a) It must learn to recognize the different items that are presented in input. The items are represented by patterns that partially overlap. Hence, the network must learn how to distinguish among them in order to associate each distributed pattern in input with the appropriate localist pattern in output.

b) The network must be able to compress various input patterns into a single pattern which is the pattern stored each time in the memory units. This task is related to the previous one because, in order to compress information in a small space, ıt is useful to eliminate redundancies.

c) The network must be able to translate the compressed pattern stored in the memory units into the required output pattern.

The first two abilities are involved in every task in which a network with memory units, in order to appropriately process the current input, must take into account the preceding inputs it has already processed. Therefore it seems interesting to try to understand the properties of our of networks.

Another interesting issue is that our networks seem to perform in a way that appears similar to human performance in that they show:

1) better performance on shorter than on longer sequences (see Figure 4).

2) better performance on the last items of a sequence and also, although to a less extent, on the very first item (see Figure 3 and 5).

From Figure 4 it is clear that the network, even if it has only been trained with sequences of length 7, performs better with shorter sequences than with longer ones. The percentage of items correctly recalled is about 75 for sequences of two items, 65 for sequences of three items, and so on.

There might be two possible explanations for this effect of sequence length on recall. A first explanation, which could be called structural, is based on a capacity limitation of the memory store used by the network to recall sequences of items. The network is provided with a limited amount of space, i.e. a limited number of hidden and memory units, to store many items. This can explain why it does not have perfect recall and why its performance decreases gradually with the increase of the number of items to be recalled.

A second explanation has a more dynamic character since it puts an emphasis on the process of memory trace formation during the presentation of a sequence. We know that this process implies a compression of information from the preceding trace with information from the new item. This process is repeated as many times as are the number of items in the sequence to be recalled. There might be a limitation in the number of times this process can go on without seriously damaging the memory trace, and this independently of the capacity of the memory store, i.e. of the number of hidden and memory units.

The results of the present simulations may not be sufficient to test which of the two explanations might be the correct one, although the fact that increasing the number of hidden (and memory) units from 15 to 19 does not lead to a better performance could be interpreted as against the first explanation based on capacity limitation. A more direct test would be to see if by substantially increasing both the number of hidden (and memory) units and sequence length, the same global performance is obtained as in our simulations, or a much worse performance. In the latter case, sequence length, i.e. number of memory trace compressions, would be the real limiting factor, as proposed by the second, "dynamic", explanation.

However, aside from this global effect, we may ask why the network's performance varies as a function of item serial position and why the last items and the first one are better recalled than the middle items. This effect appears to be independent of the number of hidden units and of sequence length.

It is important to keep in mind that in this kind of network the process of pattern compression is a dynamical process. Each time a new item is given in input to the network, all the preceding items, or more correctly, the compressed pattern that collectively represents the preceding items, must be processed again to generate a new compressed pattern that includes the new item. (What we call "compression" is the process of generating a new internal representation on the hidden units based on information coming from the new input and from the compressed representation of the preceding items as stored in the memory units.) This fact can explain why, during the presentation of a sequence, items are recalled better immediately after they have been presented and the percentage of correct recall of an item gradually decreases as new items are presented (cp. Figure 6). Each time the memory representation of the preceding items is re-processed to include a new item there is the possibility of losing something included in that rapresentation. In other words, new items interfere with the recall of the preceding ones (retroactive interference).

On the other hand, items just presented are not always correctly recalled with a probability of 100%, with the single exception of the first item (cp. Figure 6). In other words, if the recall of an item is tested just after the item has been presented, there is perfect recall only if the item is the first in the sequence, but the immediate recall of an item becomes worse for items in successive serial positions. This can be explained by the fact that the more the items are already represented in the compressed pattern stored in the memory units (i.e. the more the items preceding the current item), the more becomes it difficult to compress the new item with the already compressed rappresentation of the preceding ones. In other words, the preceding items make it more difficult to process and recall the current item (proactive interference).

These two facts taken toge'her could expl:in why the last items and the first item of a sequence are better recalled than the middle items. The last items are those that are re-processed a smaller number of times and so have less chance to be damaged (i.e. they are less affected by retroactive interference). The first item is more easily processed the first time because it is compressed with an empty prior context (i.e. it is the least affected by proactive interference). But another logical step is still missing from this reasoning. Retroactive and proactive interference are two opposite forces since the first decreases the probability of the first items and the second, the probability of the last items being correctly recalled. As a consequence, only if these two forces have non-linear effects should we expect significant differences of correct recall among different item positions of the kind we get. Otherwise, if the effects of the two opposite forces of retroactive and proactive interference were linear, we should expect that the two forces would balance and neutralize each other and we should get the same amount of recall for all serial positions in a sequence. As a matter of fact, our data appear to show that the effects of retroactive and proactive interference, while being opposite, are non-linear. For example, if we look at how retroactive interference affects the recall of the first item (i.e. how much the percentage of recall of the first item decreases with the arrival of new items), we get the curve shown in Figure 7. (The curves for all item positions are shown in Figure 6.)

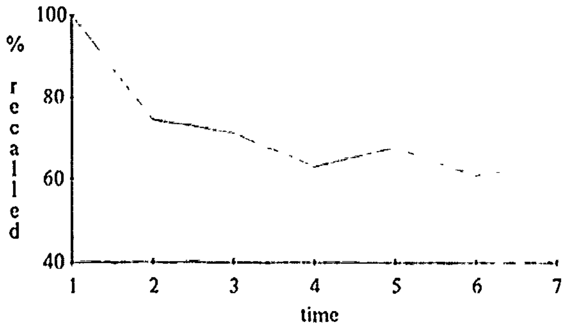


Figure 7: Recall of the First Letter as a Function of Number of Successive Letters

The percentage of correct recall of the first item is 100% when the preceding context is empty (i.e. at time 1), it decreases dramatically when the second item is presented, and then it decreases more gradually.

Similarly, if we consider how proactive interference affects the recall of the last item as a function of how many items have already been presented, we get the curve shown in Figure 8.

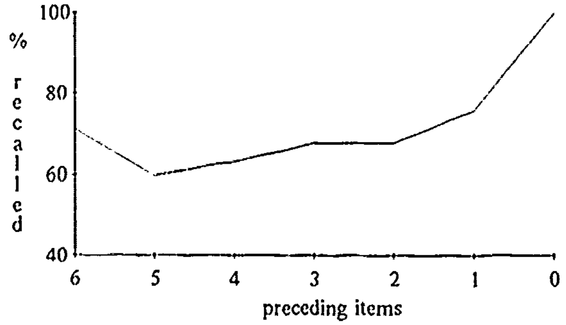


Figure 8: Recall of the Last Letter as a Function of Number of Preceding Letters

The percentage of correct recall of the last item is 100% if the previous context is empty (i.e. if the sequence has length 1), it decreases dramatically if a single item has already been presented, and then decreases gradually with the increase of the number of items already presented. (Surprisingly, the percentage of correct recall of the last item is better if the context contains 6 items instead of 5. This could be due to the fact that the network has been always trained with sequences of length 7.)

We conclude that a network architecture in which memory units store the activation pattern on the hidden units elicited by one stimulus and re-input this pattern to the hidden units when the next stimulus is processed, can simulate free recall of sequences of items. We do not imply that this same architecture underlies human performanc on this task. However, it is an interesting resu.. that two specific properties that are observed in humans when they free recall item sequences, i.e. decreasing recall with increasing sequence length and an U-shaped recall curve, are also observed in the performance of our networks and can be explained as natural consequences of the functioning of their architecture.

Acknowledgements

This research was supported by the ESPRIT Basic Research Action 3207 ACTS and by "FONDAZIONE SIGMA TAU", Rome

Rererences

Baddeley A.D., Hitch G.J. (1974). Recency re-examined. In S. Dornic (Ed.). Attention and performance (Vol. 6). Hillsdale, NJ:Erlbaum, pp. 647-667.

Crowder R.G. (1976). Principles of learning and memory. Hillsdale, NJ: Erlbaum.

Glanzer M. (1972). Storage mechanisms in recall. In G.H. Bower (Ed.). The Psychology of learning and motivation. Advances in research and theory. (Vol. 5). New York: Academic Press, pp. 129-193.

Elman, J.L. Finding structure in time. (1990). Cognitive Science, 14, 179-211.

Jordan, M.I. (1986). Serial order: A parallel distributed processing approach. Institute for Cognitive Science. Report 8604. University of California, San Diego.

Shallice T., Vallar G. (1990). The impairment of auditory-verbal short-term storage. In: G. Vallar and T. Shallice (Eds.). Neuropsychological impairments of short-term memory. New York: Cambridge University Press, pp.11-53.

Rumelhart, D.E., Hinton G.E., & Williams,R.J. (1986). Learning internal representations by error propagation. In D.E. Rumelhart, and J.L. McClelland, (Eds.), Parallel Distributed Processing: Explorations in the microstructure of cognition (Vol.1) (pp.318-362). Cambridge, Mass.: MIT Press.

Schneider, W., & Detweiler, M. (1987). A connectionist control architecture for working memory. In G.H. Bower (Ed.) The Psychology of learning and motivation vol 21. New York: Academic Press.

Schreter, Z., & Pfeirer, R. (1989). Short term memory and long term memory interactions in connectionist simulations of psychological experiments on list learning. In L. Personnaz and G. Dreyfus (Eds.), Neural Network. From models to applications. Paris: I.D.S.E.T.

Schweickert, R., Guentert, L., & Hersberger, L. (1989). Neural Network Modles of Memory Span. Preceedings of the Eleventh Annual Conference of the Cognitive Science Society. Ann Arbor, Michigan.

Binding, Episodic Short-Term Memory, and Selective Attention, Or Why are PDP Models Poor at Symbol Manipulation?

Rainer Goebel*
Department of Psychology
Philipps University of Marburg
D-3550 Marburg, FRG

Abstract

To model all aspects of human cognition it is reasonable to develop systems which exploit the advantages of both parallel distributed processing and symbol manipulation because the advantages of each approach are complementary. To approach this goal it is proposed to start with simple paralled distributed processing models and to modify them so that they are also capable of symbolic manipulation. A theoretical analysis of the essential aspects of physical symbol systems and their relation to human information processing, and the analysis of feedforward and recurrent PDP models converge to the same conclusion that the crucial steps towards symbol processing lie in incorporating an episodic short-term memory for serial order information and a selective attention mechanism in PDP models. An important first step towards constructing these mechanisms is based on a neurally inspired solution of the binding problem which assumes that the exact timing of neuronal activity may play an important role for information processing in the brain. Based on these ideas a new model is developed which posesses a flexible short-term memory and an attention mechanism The model overcomes many problems of ordinary PDP models, e.g., representing multiple objects, structured (hierarchical) distributed representations, position independent application of knowledge and explicit rule following. The model is a promising step towards a PDP cognitive architecture, of power and generality previously attained only by symbol manipulation models such as ACT* and SOAR.

1 INTRODUCTION

There are currently two competing approaches for modeling human cognition and for building intelligent systems: the classical symbolic approach and the connectionist approach, each with non-overlapping strengths and weaknesses. However, to deal with all aspects of human cognition it is reasonable to develop systems which exploit the advantages of both approaches. How can this be done? One way is to build *hybrid systems*, where traditional symbolic algorithms and connectionist networks work together

A more challenging and appealing way is to build a totally connectionist model with the ability to do both a kind of symbol manipulation and parallel distributed processing. The human brain — a huge and very complex "connectionist system" — is an existence proof that this is possible.

There are three ways of approaching this goal. The first is to build *functionally hybrid connectionist systems* consisting of two subnetworks, one with symbolic and one with PDP properties. They combine their strengths by communicating with each other. (For example a "symbol module" capable of following explicit rules can train a"PDP module", Goebel, 1990b).

The second way is *top-down*. start with classical symbol systems and *implement* them in a connectionist network hoping that on the way down connectionist strengths like parallel constraint satisfaction can be added. Most work in this direction has been done using *localist* (structured) connectionist networks (e.g., Shastri, 1988), but some success is also achieved with networks using distributed representations (e.g., Touretzky and Hinton, 1988). Unfortunately all these approaches suffer from several drawbacks. The most important problem of such networks is that almost all connections have to be "hand-wired", thus requiring a large amount of human effort. Because these networks are implementations of symbolic schemes (production systems or semantic nets), they lose many advantages of distributed connectionist models, especially the

*address after November 1, 1990: Department of Psychology; University of Braunschweig, Spielmannstr. 19, D-3300 Braunschweig, FRG ability of gradual learning.

The third way — the way which is adopted in this study — is "bottom-up". This way proposes to start with simple PDP networks and to move towards symbol manipulation *without sacrificing any advantage of parallel distributed processing.*

According to this approach the following questions a-e raised:

- What are the essential aspects of symbol processing? What and how much of human information processing is best characterized as symbol manipulation?
- What are the essential aspects of parallel distributed processing? What and how much of human information processing is best characterized as parallel distributed processing?
- Is it possible to modify PDP models so that they are also capable of symbol manipulation without sacrificing their natural properties?

The answers to these questions should lead to some insights into the nature of the problem and hopefully point to promising directions for building a new model.

2 SYMBOL SYSTEMS

According to Newell (1980) a *physical symbol system* consists of a set of symbols, which can occur as components of expressions (symbol structures). A symbol system also contains a collection of processes that operate on expressions to produce other expressions. Especially important are the following notions:

- *Interpretation*: Symbol systems are able to interpret expressions through the fundamental distinction between process and data.
- *Designation.* An expression can designate other expressions. This allows, e.g., to bind a symbol to arbitrary expressions and to recursively expand small symbols to larger ones and to package large expressions to smaller ones.

The ability of symbol systems to construct *arbitrary* combinations out of other symbols or symbol structures is called *compositionality* (Fodor & Pylyshyn, 1988). A symbol may also designate a process (operation). A process is applied to data in a *structure-sensitive* way.

To illustrate these concepts, consider the following example In LISP, the function `CAR` returns the first element of a list, e g , `(CAR '(A B C))` produces the element `A` The whole expression serves as data and is interpreted by the `EVAL` function. The expression is analyzed or parsed by `EVAL` resulting in the recognition that `CAR` designates an operation wh'ch is to be applied to the expression `'(A B C)`. The `QUOTE` sign (`'`) determines that the list `(A B C)` serves as data for the `CAR` function producing the symbol `A` as output. The application of `CAR` is structure-sensitive because it is performed *without caring about the concrete symbol appearing in the first position or how many and which elements appear after the first position in the list.* The only crucial fact is the *structure* of the expression, namely `(CAR '(<first element> <other elements>)` producing `<first element>`. The parts of the structure are *variables* which can be filled with any symbols.

This example shows that symbols are treated as atomic elements. Symbol structures are interpreted and serially manipulated by a processor. The ability of symbol systems to construct and manipulate *arbitrary* symbolic structures built from a set of elementary symbols together with the distinction between process and data allows symbol systems to do *universal computations.* This is in contrast to PDP models which are generally task-specific networks.

3 PARALLEL DISTRIBUTED PROCESSING

Connectionist networks using *distributed representations* stress the rich internal structure of entities. A symbol is not an atomic entity but an activity pattern over a large number of units. These activation patterns are not interpreted but directly evoke other activity patterns through connection weights. Each weight contributes a small influence (a 'weak constraint', a 'soft rule' or a 'microinference') to the mapping from one activity pattern to the next.

Distributed representations naturally lead to the advantages of parallel distributed processing as shades of meaning, similarity-based generalization, simultaneous consideration of many pieces of knowledge. Additionally, PDP models show how symbols with rich internal structure might emerge and how they change their appearance through simple learning rules. These rules slowly modify the weights to improve performance on a given task.

4 HUMAN COGNITION

Before the central assumptions of this paper are presented it is important to specify the concrete views of connectionist models and symbol systems considered here. This is important because in general both frameworks offer universality (e.g., connectionist models are implemented on serial computers). In the following the terms 'symbol manipulation' and 'symbol system' are meant in the sense of a LISP interpreter. 'Parallel distributed processing' or 'subsymbolic interactions' is meant in the sense of simple PDP models, especially feedforward and recurrent back-propagation networks.

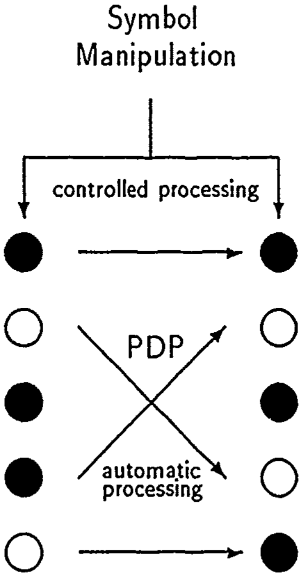


Figure 1. The dualistic nature of a symbol

The central theses of this paper can be summarized as follows: (see figure 1):

- The real power of human cognition lies in the fact that it can *simultaneously* view a symbol as an atomic entity and as an entity with rich internal structure.
- Human *controlled processing* corresponds to some extent to symbol manipulation. *Short-term memory for serial order information* and *selective attention* constitute the basis of (concatenative) compositionality and structure-sensitive processing.
- Human *automatic processing* corresponds to some extent to associative processing as produced by PDP models.
- Since human cognition is always a mixture of controlled and automatic processing it is best described as the coevolution of symbol manipulation and associative processing.

The distinction of a symbol as an atomic entity and as an entity with rich internal structure corresponds to Smolensky's (1988) distinction between symbols and subsymbols. Smolensky argues further that symbol manipulation is an emergent property of subsymbolic interactions. Although this is generally true, in this paper it is argued that symbol manipulation can not emerge from subsymbolic interactions of the kind found in ordinary PDP models. Although PDP models are able to produce implicit rules (e.g., Rumelhart & McClelland, 1986), hierarchical data structures (e.g., Elman, 1988; Pollack, 1988) and even structure-sensitive processing (Chalmers, 1990) it is argued that this works only for *learned regularities*: PDP models are unable to appropriately represent and handle *new* constellations of multiple objects which are against learned regularities.

To reach the universality of symbol systems other mechanisms are proposed which are assumed to be manifest in human short-term memory and selective attention. The goal of this paper is to modify PDP models so that they possess a kind of short-term memory and attention but otherwise work as ordinary PDP models. This allows symbol manipulation and PDP-like processing to evolve side by side over time. This is different than, for example, the distributed connectionist production system (Touretzky & Hinton, 1988). Although this model uses distributed representations the parallel interactions work *only* at the symbol level and thus loses the spirit of subsymbolic computation.

The assumption of the coevolution of symbol manipulation and parallel distributed processing also implies that the PDP approach is the natural one to start with. Compared with the manipulation of 'high-level' symbols it requires a lot of time for a symbol system to *precisely* compute the effects of subsymbolic interactions because the single processor must 'slip' in each processing element to serially compute the parallel interactions. Thus a serial symbol system is unable to manipulate high-level symbols and to produce the associative effects of subsymbolic interactions at the same time scale.

In general human cognition is viewed as a fruitful mixture of controlled processing (best described as symbol manipulation) and automatic processing (best described as associative processing in the spirit of PDP models). Language processing (syntax and semantic aspects) and the gradual development of (cognitive) skills are considered to be prime examples of this interaction.

In the following section the proposed relation between symbol manipulation and controlled processing is elaborated.

4.1 CONTROLLED PROCESSING AS SYMBOL MANIPULATION

Psychologists distinguish between controlled (or deliberate) and automatic processing (Shiffrin & Schneider, 1977). A central characteristic of controlled processing is that it requires *attention* while automatic processing does not. Controlled processing also depends on a flex:ble but limited *short-term memory*. If a task requires too much information to be retained in this "working r. emory" it is overloaded resulting in performance errors.

Short-term memory for serial order information

Human short-term memory is able to retain item and serial order information. Another important point is that it is able to store this information with only *one presentation* of a sequence. This seems trivial but it is important to note that humans can store and reproduce sequences they may never have seen or heard before (as long as short-term memory is not overloaded). An example is to retain a new seven-digit phone number until it is dialed. Even simple rules can be stored, reproduced and correctly applied after one presentation (Hadley, 1990).

The term 'episodic' short-term memory (eSTM) is used henceforth to emphasize the challenge for connectionist models to retain item and order information with 'one-shot' or 'single-trial' learning.

Episodic STM is important because it shows that humans have the capability, albeit limited, to immediately store arbitrary combinations of elementary objects which is required for compositionality.

Selective attention for structure-sensitive processing

Attention is studied primarily in visual and auditory perception. In this study, however, the important role of attention for symbolic manipulation is stressed, that is how it controls read-out from eSTM.

It is assumed that attention allows the *segregation* of arbitrary segments of the information present in eSTM. The segments can then be processed further.

To illustrate the proposed role of attention consider how a *human* (familiar with LISP) might evaluate the expression `(CAR '(A B C))`. A simplified description might be as follows:
First attention is focused primarily on the left side of the expression. This leads to the recognition of the `CAR` function. The retrieved knowledge about this function lets the human now focus attention at the position following the `'(` where the letter `A` is found.

This example shows that humans are able to process information as symbol manipulators: the LISP expression is 'interpreted', that is attention is focused on the left side to identify what to do next. Through the segregation of the left side the symbol `CAR` is found. The knowledge about the `CAR` function is retrieved recognizing that it is a function. This corresponds to the designation process. It is also retrieved that `CAR` returns the first element of its argument. Thus attention is focused on the first position of the list `(A B C)` and the element A is segregated. This is a kind of structure-sensitive processing because what counts is not the concrete symbol but only the position where it appears.

Note that in this example the task was treated as presented as a visual processing task, thus attention could be guided to *spatial* locations. However humans can do this task even if the expression is presented auditorily or if elements of the expression are presented sequentially in the same retinal position. Thus selective attention must also work on the representations in short-term memory.

4.2 AUTOMATIC PROCESSING AS PARALLEL DISTRIBUTED PROCESSING

Consider again a simple LISP expression. `(CAR '(PDP is important))`. According to the previous section, attention is first focused on the `CAR` symbol, then on the first symbol in the list, leading to the answer PDP. Is this really all what happens?

Although humans can do a kind of structure-sensitive processing, they are *unable to do only structure-sensitive processing* because the processed symbols *directly evoke associations*. In contrast to the effortful process of controlling attention, the processed symbols *automatically* provide a cloud of associations. Thus if humans do the LISP task they can not focus on the symbol `PDP` without having "in mind" associated ideas, as, for example, knowing that PDP is an acronym for paralled distributed processing. Furthermore it is difficult to ignore the other symbols in the list as is typical for 'real' structure-sensitivity. one simply *must read the sentence*. More generally, it is impossible to totally suppress the *context*, which always do influence processing in parallel.

Context may operate on different levels, e.g., a human focusing on the `CAR` function in the context of evaluating a LISP expression is likely unaware of the fact that `CAR` may also represent an automobile in other contexts.

These examples illustrate the central thesis of this paper that controlled processing (symbol manipulation) and automatic processing (directly evoked associations) always evolve side by side over time.

In the following section the claim that PDP models are unable to do symbol manipulation is explained. This claim is based on several reasons. The most important one for the present study is the inability of PDP models to appropriately represent and handle *arbitrary combinations of multiple objects*. This is required for compositionality. The nature of the problem can be clearly revealed in the context of the *binding problem* (Hinton et. al., 1986; Smolensky, in press).

5 THE BINDING PROBLEM OR WHAT GOES WITH WHAT

The binding problem is the problem of "what goes with what". It is mostly studied in the visual domain where it is also known as the feature integration problem (Treismann & Schmidt, 1982). In this study its general relevance for representing multiple objects at each processing level is stressed. In the visual domain it is well known that features of objects (e g , shape, color, motion) are analyzed in different areas (maps) of the brain. What is the problem with that?

Suppose there are four local units, two for representing the shape of the letters A and B respectively and two for the colors red and blue (see figure 2). Now suppose a red A is presented at the retina for some time. Then the units representing red and A are active (see figure 2a). The information that a red A is presented is represented through the simultaneous activity of the units for red and A

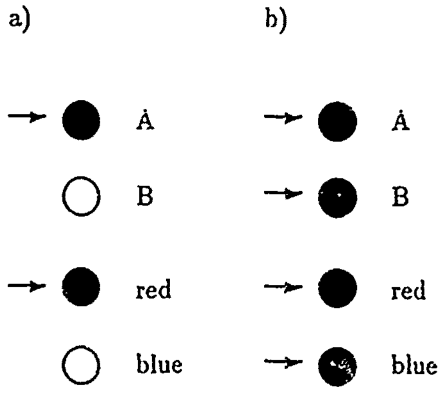


Figure 2. The whole as the set of active units

Now suppose that additionally a blue B is presented for a while. Then also the B unit and the blue unit are active (see figure 2b) Note that from looking at all four units it is now impossible to decide wether there is a red A and a blue B or whether there is a blue A and a red B. The representation does not specify what goes with what thus all combinations or groupings of activated units are possible leading to "illusory conjunctions".

5.1 WHOLES AS THE SET OF ACTIVE UNITS

This problem arises because the simultaneous activity of units can not represent which active units go together, the activity only represents the whole expression as the *unordered set* of active units. Instead of representing the whole as composed of the constituents red A *and* blue B something is represented which is red, blue, an A and a B. Thus the fundamental problem of PDP networks is that they must represent whole expressions and their relations to the constituting parts with the set of active units. This is very difficult.

The proposed solution to these problems is to use *binding or conjunctive units* (e.g., Smolensky, in press) which detect combinations of occurring objects. This solution works for specific tasks but is not appropriate in general (von der Malsburg, 1981). To represent all imaginable groupings, huge numbers of conjunctive units are necessary. Even more severe is that they must exist in advance to foresee all possible groupings. *Combinations of objects which are not anticipated simply can not be represented.* Additionally, the representation buries the individuality of the elements bound together leading to further diifficulties (Fodor & McLaughlin, 1990).

Conjunctive units are of course important to capture higher-order regularities but they should not have the burden of representing all possible bindings. To solve the binding problem a much more flexible solution is desirable which establishes temporary bindings at the time they are really needed.

5.2 OSCILLATING UNITS: WHOLES AS DESYNCHRONIZED PARTS

An interesting solution to the binding problem is based on ideas proposed by Legendy (1970) and more concretely by von der Malsburg (1981, 1986). In contrast to most connectionist models assuming that only the *average output activity* of neurons encodes information between neurons, they suggest that the *exact timing* of neuronal activity (firing of individual neurons or 'bursting' of cell groups) plays an important role for information processing in the brain. The central idea is that a unit receiving *constant* input does not respond with a constant output but with *oscillating behavior.* This allows objects to be labeled with different time phases. if multiple objects are to be represented, the parts of one object can be linked together through synchronized (phase-locked) activity and separated from other objects through firing at different phases.

Synchronization can be stabilized through synaptic modulation of fast weights (see also Hinton & Plaut, 1987; Feldman, 1982): Units oscillating in phase increase the weights connecting them and units out of phase decrease the connecting weights. Thus if two objects are desynchronized for a while and then both are presented at the same time the fast weights (together with some noise) tend to desynchronize them again.

Consider again the previous example. first a red A is presented stimulating the corresponding units for red and A. These units now respond with phase-locked activity (see figure 3a). Then a blue B is additionally presented Figure 3b shows that although now all 4 units are stimulated, unit red and A on the one hand and unit blue and B on the other hand are grouped together through phase-locked oscillations and both are seperated from each other because they oscillat·: in different phases. Thus the whole "a red A *and* a blue B" is represented correctly because the parts (the red A and the blue B) are both distinctly represented.

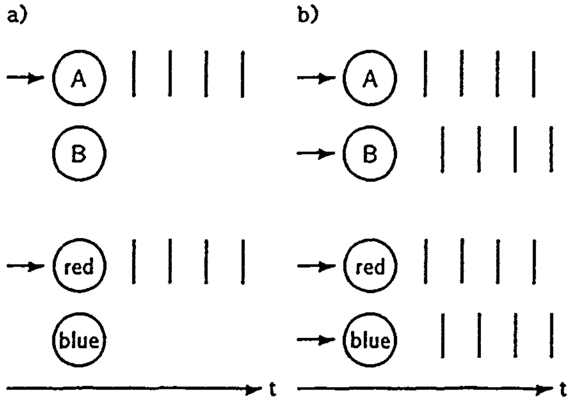


Figure 3. The whole as desynchronized parts

In this example synchronization was established through simultaneous stimulus onset but grouping information could also come from location information. Generally, long-term knowledge and attention can also be sources of grouping information.

6 PADSYMA, THE NEW MODEL

The described binding mechanism constitutes the first important step to achieve the dualistic treatment of symbols: multiple objects, each maybe a large distributed pattern of activity, can be represented simultaneously yet preserving its individuality. The distinctness of the objects is represented on a very rapid time scale. For a very short time one object is active and then the other object etc. On a somewhat slower time scale – e.g., 'a psychological moment' consisting of some oscillations – the experience of the whole expression is represented.

However some further steps must be done to achieve an eSTM and an attention mechanism as the basis of symbol manipulation with distributed objects. The mechanism as described so far can group objects together and seperate them from other objects. However, this is not sufficient for representing sequences because the serial order information between distinct objects must be represented, too. If all elements of a sequence would oscillate in a different phase they would be represented individually but no order information would be established. If all elements would oscillate in the same phase nothing is gained because within a phase the whole is represented as the set of active units in the same way as in ordinary PDP models.

The first steps done to resolve these and further problems lead to the development of a new model, PADSYMA (*pa*rallel *d*istributed *sy*mbol *ma*nipulation). The name of the model emphasizes th·:t it is designed to combine the strengths of PDP models and symbol manipulation through the treatmeant of symbols both as atomic and as distributed patterns.

6.1 OVERVIEW OF THE MODEL

Figure 4 shows a sketch of the model. The architecture of the model is very simple to demonstrate the general ideas of the proposed eSTM and selective attention mechanism, e.g., there are currently no modules for different sensory modalities or motor control.

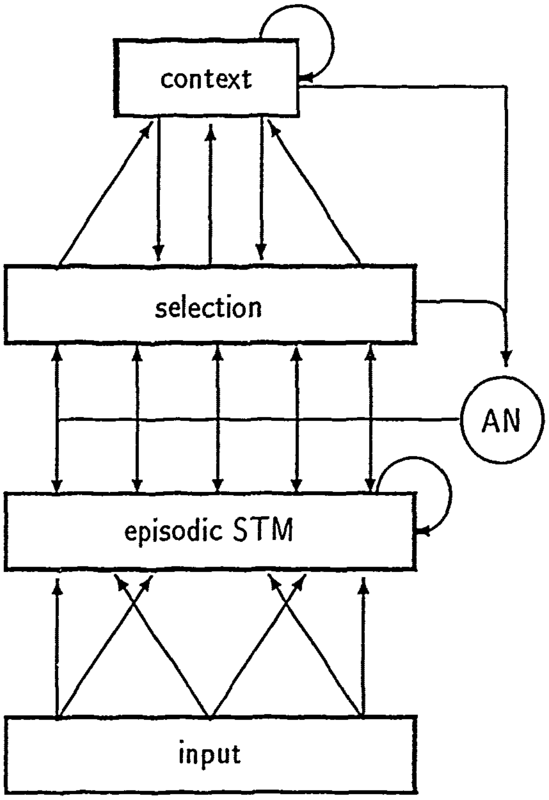


Figure 4. A sketch of PADSYMA

An input sequence is presented to the input layer which feeds into the episodic short-term memory. The eSTM units are totally connected with each other except to themselves. The weights are modified according to a Hebb-like learning rule on a fast time scale. If these units are stimulated for some time they exhibit bursting behavior. activation rises until a gliding average of activity is reached followed by a 'refractory period' where the unit is quiet even if it is further stimulated. Then the next burst can occur.

A special unit in the eSTM, the inhibitory unit H (not shown in figure 4), limits the total activity of the eSTM layer and plays an important role for synchronizing and desynchronizing blocks of activated units. All eSTM units project to the inhibitory unit with fixed small positive weights and receive connections with small negative weights.

The eSTM units have one–to–one bidirectional connections to corresponding units in the selection layer with fixed weights of 1. These connections are for simplicity one-to-one in the current model but can be any bi-directional weight pattern. The selection layer holds the information which is currently in the focus of attention. The selection layer feeds to the context layer and receives recurrent connections from this layer. The loop - selection layer, context layer, selection layer - constitutes the higher processing levels. Long-term memory is represented in the weights between these layers and the weights projecting to the attention node. This allows top-down effects to influence the knowledge in eSTM and to control attention.
The attention node (AN) gates the input from the eSTM to the selection layer thus determining the strength with which information present in the eSTM reaches the selection layer and vice versa. Focusing attention is represented in a high (oscillating) activity of the attention node. The attention node gets its input from the selection and context layer. Note that if the attention node oscillates in a special phase the selection layer 'sees' only the information of that phase.

6.2 SYSTEM DYNAMICS

The units of the input layer are simply turned on or off. The weights between the input layer and the eSTM are fixed.

The units in the eSTM must be able to exhibit oscillating behavior. There are many ways to achieve this. In the present version of the model oscillation behavior is not modeled on the level of individual spike trains but on the averaged activity of cell groups (Schneider & von der Malsburg, 1986). Each cell group is represented with one eSTM unit. The net input of eSTM unit i is computed according to the following difference equation:

$$\begin{aligned} net_i(t+1) &= \alpha a_i(t) + \sum_{I=1}^{n_I} w_{iI}a_I + \sum_{j\neq i}^{n} w_{ij}a_j + \\ & a_{S_i}a_A + w_{iH}a_H + r(t) \end{aligned}$$

Thus the net input of unit i at time step $t+1$ depends on its previous activity determined by the parameter α which is set to 0.92 in all subsequent simulations. The net input also depends on the influence of the n_I input units and the other $(n-1)$ eSTM units. Unit i also receives activation from the corresponding unit of the selection layer a_{S_i} multiplied by the activity of the attention node a_A. Additionally, it gets negative input from the inhibitory unit ($w_{iH} = -0.2$) and small noise values $r(t)$ randomly chosen from an interval ranging from 0 to 0.01. Noise is important to break accidental synchronization between weakly coupled units.
The final activation is computed according to

$$a_i(t+1) = R_i(t)\frac{1}{1+e^{(0.5-net_i)/0.2}}$$

The refractory function $R_i(t)$ takes the values of 1 or 0 depending on the value of the gliding average $G_i(t)$:

$$G_i(t+1) = (1-\delta)G_i(t) + \delta a_i(t)$$

If $G_i(t)$ reaches the upper threshold value $g_u = 0.5$, $R_i(t)$ is set to zero. Then also $a_i(t)$ is 0 and the gliding average decays with time constant $(1-\delta)$ until the lower threshold $g_l = 0.02$ is reached ($\delta = 0.35$). Then the refractory period is over, $R_i(t)$ is set back to 1 and unit i may respond to receiving signals.
The incoming weights w_{ij} from the other eSTM units (all weights are positive) are changed according to a Hebb-like rule:

$$w_{ij}(t+1) = \gamma w_{ij}(t) + \begin{cases} \epsilon a_i a_j & \text{if } R_i \leq F_i \\ \epsilon a_i a_j \frac{F_i}{R_i} & \text{otherwise} \end{cases}$$

At each time step the weights decay slowly towards 0 with time constant $\gamma = 0.98$. Each unit has a fixed amount of "weight substance" $W = 0.7$ that it can distribute among its incoming weights. The available weight substance at the current time step is $F_i(t+1) = W - \sum_{j=1}^{n} \gamma w_{ij}(t)$. If there is not enough weight substance for all requested weight changes ($R_i(t+1) = \sum_{j=1}^{n} \epsilon a_i a_j$) then $F_i(t+1)$ is distributed among the incoming weights in proportion to the amount of the individual weight change requests.

The inhibitory unit H is updated as follows ($\beta = 0\ 63$; $w_{Hi} = 0.01$):

$$a_H(t+1) = 1/(1+e^{0\ 5-(\beta a_H(t)+w_{Hi}\sum_i^n a_i(t))/0\ 2})$$

The attention node is influenced by the n_S selection units and the n_C units of the context layer ($n_R = n_S + n_C$) (all weights positive) which can drive its activity towards its maximum value 1. Without stimulation the attention node decays quickly ($\lambda = 0.6$) to a small resting level ($b = 0.2$):

$$a_A(t+1) = b + \lambda a_A(t) + (1-a_A(t))\sum_{R=1}^{n_R} w_{AR}a_R$$

Long-term memory

Long-term memory is established in the recurrent loop - selection layer, context layer, selection layer. Since this paper focuses on short-term memory and attention long-term memory effects are not considered in detail. In the following simulations necessary weights are set by hand. In this section the primarily functions of long-term memory based on first simulation experiments is briefly described.

While the eSTM weights store the *temporal* bindings (associations) between elements, the weights in the recurrent loop contain the long-term knowledge. The weights between the layers of the recurrent loop change on a much slower time scale than the fast weights of the eSTM layer. The weights are updated according to the back-propagation learning rule adopted for recurrent networks (Goebel, 1990a). The error at each time step is determined by the difference between the actual activation of the selection layer and the activity produced by the recurrent loop. Thus the recurrent loop is trained to *predict* the next state of the selection layer (Elman, 1988). This develops useful representations in the context layer which can be used for pattern completion and the recognition and production of known sequences. New sequences, rehearsed in eSTM, can be learned by heart.

Due to the long-term memory weights each distributed symbol active in the selection layer evokes its typical associations. The serial order information between these elements determines how the associations of the single elements interact to produce a coherent representation of the whole sequence (e.g. the meaning of a sentence).

Long-term knowledge is primarily applied to information currently in the focus of attention. But also the unattended information modulates processing because it too reaches the selection layer however with weaker strength. Long-term memory also controls the activation of the attention node.

6.3 SHORT-TERM MEMORY FOR SERIAL ORDER

The following simulations demonstrate how the eSTM is able to store the serial order of a presented sequence (no long-term memory effects are considered). The model must cope with the most difficult case that is that only one element is presented at each time step (imagine an auditorily presented input or that the elements are displayed successively). For simplicity first a local representation is used.

Suppose the sequence (, A, B, C, and) is presented in successive order. Serial order information can only be established if some information about the past is retained over time If the elements are presented, information about past elements is retained through resonance of the eSTM units with the corresponding selection layer units (the attention node is at its resting level). If the first letter A appears the unit in STM gets activity and can also activate its corresponding unit in the selection layer. The connections between these layers are bidirectional so that the selection layer activates the unit in the STM layer and vice versa. If the input stimulus is strong enough both units gets more and more active until the STM unit reaches the upper threshold of the gliding average. After this burst the unit is quiet and therefore also the corresponding selection unit.

The important point is that this process needs some time. Thus if the next element is presented after a short time interval the first one is still active and so on. The result is that there is some *overlap* between the elements which is proportional to the proximity of the elements in the presented sequence. This simple overlap - resulting naturally of the temporal behavior of the units - is sufficient to store the serial order information in the fast weights of the eSTM layer. The resulting weight pattern after one presentation of the sequence is shown in figure 5.

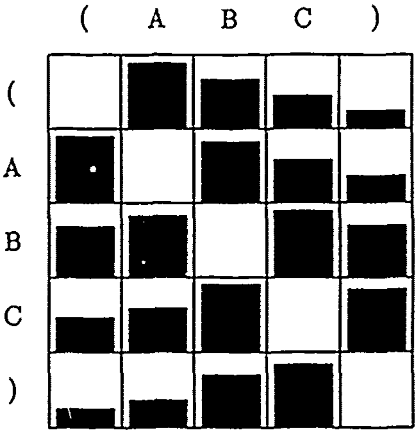


Figure 5. Fast weights after presenting (A B C)

Note that the weights resemble a (graded) context-sensitive representation! However this representation is not hand-designed (Rumelhart & McClelland, 1986) or slowly learned for a special task (Mozer, 1990), but developed "on the fly", at the time it is really needed.

Rehearsal

Now consider how the order information can be retrieved. Assume that only the start symbol (gets top-down activity at the selection layer. Then the activity of the corresponding eSTM unit rises and activates through the learned (fast) weights the next sequence elements. The magnitude of this 'priming' effect decreases with the distance of the elements in the sequence as reflected in the established fast weighs (see figure 5). As time proceeds all elements burst in the right order (see figure 6). Thus the problem of storing both item and order information is solved as follows. Item information is present because each element of the sequence oscillates in a distinct phase, serial order infor..ation is present because the elements are activated in the appropriate order.

Note that there are symmetric weights in both directions. This has the effect that primed elements also 'help' the priming elements to reach their upper threshold.

During this retrieval process the (slowly decaying) weights are also strengthened again. Through repeating this process the serial order information can be retained as a *spatio-temporal pattern* over time as long as desired.

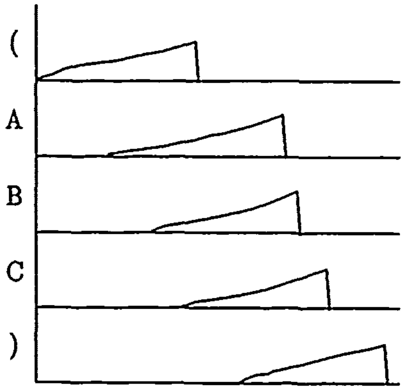


Figure 6. Spatio-temporal pattern of (A B C)

The same simulation was repeated with distributed (partially overlapping) representations for the elements. The obtained results were qualitatively the same as the reported one. Since all units of one element are activated at the same time they possess the highest fast weights among themselves. This makes the retrieval process even more robust so that the described behavior is relatively unsensitive to parameter changes.

Although the described mechanism can store and reproduce sequences it requires a high presentation rate. Note that it is only critical to achieve the described overlap between elements. If elements are presented at slower speeds other mechanisms must establish this overlap. This has not been done yet, but a possible solution is to assume that the sequence elements stored so far are rehearsed all the time and that the overlap depends on the number of joint rehearsals.

Representing repeating subsequences

The eSTM has the capability, albeit limited, to store and to retrieve sequences containing repeated subsequences because through the described priming effect the necessary information about the right order is present. Generally, the limits of representing repeating subsequences are those of context-sensitive representations. In PADSYMA this limit is reached if an element occurs more than 3 times in succession (see the weights in figure 5). However there is another problems if a unit in the eSTM layer has bursted it can't be activated again until its refractory period is over. Interestingly the difficulties of PADSYMA with repeated subsequences on a fast presentation rate seem to fit nicely with those of humans (Kanwisher, 1987).
It is assumed that on slower presentation rates long-term knowledge enhances the capability to represent subsequences in that it allows to recode the information (e.g., A A as double-A).

6.4 SELECTIVE ATTENTION

The attention mechanism has two apparently distinct functions: to segregate information of the eSTM and to bind elements by synchronizing their activity. In this study only the first function is demonstrated.

Segregation is accomplished through the oscillation of the attention node in a special phase. All elements active in that phase reach the selection layer with high intensity. This has the important consequence that *the selection process has not to be determined by specific connections.* It is sufficient to have simple one-to-one connections between the STM layer and the selection layer and let the activation of the attention node determine which specific information goes through. This is in contrast to complicated pull-out networks (Mozer, in press; Touretzky & Hinton). The point in time when the attention node is active determines which specific information arrives at the selection layer. Note that if the attention node is active over a longer time period (e.g., as long as one period) the selection layer "sees" the whole information but if it oscillates in a special phase over time it has a 'stroboscopic view' of knowledge of that oscillation phase. Thus the system can focus on the whole expression or on arbitrary parts!.

To demonstrate this consider again figure 3b. A simulation was conducted with the 4 units of the figure placed in the eSTM layer and corresponding units in the selection layer. Furthermore a positive weight was established from unit A of the selection layer to the attention node.
If unit A in the eSTM layer bursts, its corresponding unit in the selection layer gets some activity, too. Due to the positive weight from that unit to the attention node all eSTM units active in this phase reaches the selection layer intensified. Thus at the selection layer unit A and also unit red are highly activated. Although attention is focused only on the element A the bound element red is also retrieved! The activity of the attention node decreases quickly to its resting level. When the 'unattended' B and blue units burst in the eSTM only a fraction of their activation reaches the selection layer. Thus over time the selection layer 'sees' only the

red A.

There is an analogy to spot-light models of visual attention. To narrow down the spotlight equals narrow to a special phase, to shift the spotlight in space equals to shift the focused oscillation phase. This corresponds to serially scanning the representation in the eSTM and is assumed to be mediated by top-down influences.

7 PROPERTIES OF THE MODEL

PADSYMA offers interesting solutions to many problems of ordinary PDP models. First simulation results are very encouraging. Due to space limitations only the main ideas are presented briefly.

7.1 POSITION-INDEPENDANT KNOWLEDGE ACQUISITION

In many connectionist models multiple roles are represented in that an input or output layer is divided in some parts or 'slots' each holding the information about a slot-filler. This apprach has some disadvantages, e.g., that only a fixed representation scheme can be used. Another important problem is the following: if different slots are filled with the same object (in the same input pattern or in different input patterns) then the knowledge acquired about an object (filler) in one role is seperated from the knowledge about the same object in another role. If, e.g., the network has learned that a person appearing in the subject slot is a woman then the network does not know this if the same person appears in the object slot! This happens because knowledge about an object is learned *position-dependant* in the weights emanating from the slots.

A possible solution would be to replace slots by units for roles and to represent a role-filler relation as the simultaneous activity of the involved role and filler units. However this again leads to the binding problem if more then one role-filler combination is to be stored.

The solution proposed in PADSYMA is to store each role-filler combination as a two-element sequence (e.g., ROLE1 FILLER1) which are bound through fast weights. This allows the same filler to appear in combination with different roles and the acquired knowledge about the filler in different roles rests mainly in the weights emanating from the filler.

7.2 HIERARCHICAL (STRUCTURED) DISTRIBUTED REPRESENTATIONS

The previous section also points in a promising direction how to achieve very flexible structured distributed representations as is required, e.g., to represent a syntactic tree of a parsed sentence. A flexible representation requires that an object may be represented by multiple role-filler combinations; each role also may have multiple arguments or multiple fillers. This can be represented in PADSYMA with oscillating subsequences, each representing one role-filler combination (e.g., ROLE1 ARG1 ARG2; ROLE2 ARG1). This flexibility is achieved because spatio-temporal patterns exploit concatenative compositionality which allows arbitrary structured representations (cf. LISP-lists).

7.3 VARIABLE BINDING AND EXPLICIT RULE FOLLOWING

Variable binding counts as a difficult problem for PDP models (e.g., Norman, 1986). An anlysis of this problem using binding units can be found in Smolensky (in press).

Variable binding is especially important for explicit rule following. Consider the rule $X < Y \rightarrow Y > X$. If values are bound to X and Y on the left side then they must also be instantiated on the right side. The solution of this problem in PADSYMA is very appealing. Suppose the mentioned rule is presented sequentially to the model. Suppose further that the value 3 is bound to X and the value 5 to Y (a matching process is not yet established) that is the variable-value pairs have positive bidirectional fast weights. Thus if the sequence is rehearsed the variables on the right side will produce the corresponding values due to the fast weights!

Thus variable binding and variable instantiation are only special cases of the general binding method used in PADSYMA. The simulation of the attention mechanism (section 6.4) shows how the value of a variable can be retrieved in general. attention is focused on the variable (in the example the letter A) and the value appears automatically in the selection layer (in the example the color red).

These ideas will be pursued in further research and also the problem of matching within the framework of the model will be addressed. However the possibility of PADSYMA to store a sequence (of limited length) immediately and to do variable binding promise an elegant way towards a system capable of following explicit rules. Since the higher levels of the model are trained to predict the information present in the selection layer, PADSYMA also shows a mechanism how explicit rules can be 'compiled' to automatically applied implicit rules.

8 DISCUSSION

PADSYMA is preliminarily in nature, much has to be done. The model is based on the idea that the exact timing of activity exchange among units is important. This relatively old idea (Legendy, 1970, von der Malsburg, 1981, recently also proposed by Damasio, 1989) seems now mature to stimulate concrete modeling activity. The recent interest is primarily due to empirical findings demonstrating that synchronized oscillations occur in the cat visual cortex presumably to link (bind) features of objects together (Eckhorn, et. al., 1988; Gray & Singer, 1989).

Some modeling activity related to this work can be found in Lange et.al., (1989) and Shastri & Ajjanagadde (1990). They also use oscillating units to solve the variable binding problem. However their work is done within the framework of local (structured) connectonist networks.
Horn & Usher (1990) and Wang et. al. (1990) show how long-term knowledge can be used to decompose a whole expression in its (distributed) parts. This indicates another interesting role for the long-term memory of PADSYMA.

Although PADSYMA is in an early state of development it touches a broad range of psychological phenomena ranging from short-term to long-term time scales. Therefore the model will be studied carefully, modified to explain concrete data and its limits identified. In the present state only qualitative descriptions can be made. Facing the model with psychological data also provides a way to fix the free parameters determing the time behavior of the model.

On a very short time scale PADSYMA [illegible] be related to such phenomena as illusory conjunctions (Treisman & Schmidt (1982) and repetition blindness (Kanwisher, 1987). Although these phenomena require additional sub-modules, e.g., for *spatial* attention (Mozer, in press), the properties of the model seem relevant to explain these phenomena.

PADSYMA's short-term memory is not (only) represented as an activity state but also as an evolving weight pattern. Instead of two time scales as in ordinary PDP models, there is an additional one bridging the gap between the very fast activity dynamics and the very slow long-term weight dynamics. This allows for example to explain how a human interrupted in doing task A (e.g. through a phone call) can rapidly recover the state of task A before he was interrupted. Fast weights can even be used to do a limited form of recursive processing.

On a long-term time scale PADSYMA points to a deeper understanding of the relation between controlled and automatic processing. This should lead to new models of skill acquisition as oppposed to traditional accounts (Anderson, 1982).

8.1 CONCLUSION

The weaknesses of PDP models concerning symbol manipulation are expressed e.g., by Fodor & Pylyshyn (1988), Norman (1986) and Pinker & Prince (1988).

PADSYMA offers a connectionist solution of the raised problems. The solutions are not as 'perfect' as the corresponding processes in symbol manipulation systems. However the limits of PADSYMA seem to resemble those of humans making the model especially interesting for modeling psychological phenomena. Due to the dualistic treatment of high-level symbols as atomic elements and as entities with large internal structure the model preserves all advantages of parallel distributed processing and thus *is not merely an implementation of symbol manipulation*. Instead the advantages of both approaches are brought together in a fruitful way.

The most important contribution of the proposed model is that it clearly indicates a promising way towards a PDP cognitive architecture, of power and generality previously attained only by symbol manipulation models such as ACT* and SOAR.

Due to the explicit relation of the model to psychological phenomena the further elaboration of the model promises to lead to a deeper understanding of the microstructure and its relation to the macrostructure of cognition.

Acknowledgements

copy I am especially grateful to Mike Mozer for his careful critique and helpful comments during the writing of this paper. Many thanks are also due to Geoffrey Hinton for helpful suggestions. [illegible] thanks go to Helga Bauer, Thomas Goettsch[illegible] Henning Schulze, Dirk Vorberg and Kai-Uwe Wagner.

This work was supported by a grant from the Deutsche Forschungsgemeinschaft (Vo 212/2-4).

References

Anderson, J. R. (1982). Acquisition of cognitive skill. *Psychological Review*, 4, 369-406.

Chalmers D. J. (1990). Syntactic transformations on distributed representations. *Connection Science*, 2.

Damasio, A.R. (1989) The Brain binds entities and events by multiregional activation from convergence zones. *Neural Computation*, 1, 123-132.

Eckhorn, R, Bauer, R, Jordan, W., Brosch, M., Kruse, W., Munk, M. & Reitboeck, H.J. (1988) Coherent Oscillations: A mechanism of feature linking in the visual cortex? *Biological Cybernetics*, 60, 121-130

Elman, J.L. (1988) Finding structure in time (CRL Tech. Rep. 8801). La Jolla, CA: University of California, San Diego, Center for Research in Language.

Feldman, J. A. (1982). Dynamic connections in neural networks. *Biological Cybernetics*, 46, 27-39.

Fodor J. & McLaughlin, B. P. (1990). Connectionism and the problem of systematicity: Why Smolensky's solution doesen't work. *Cogniton*, 35, 183-204.

Fodor, J. & Pylyshyn, P. (1988). Connectionism and cognitive architecture: A critical analysis. *Cogniton*, 28, 3-71.

Goebel, R. (1990a). Learning Symbol Processing with Recurrent Networks. In R. Eckmiller (Ed.), *Proceedings of the International Conference on Parallel Processing in Neural Systems and Computers (ICNC)*, Dusseldorf, F.R.G., 19-21 March, 1990.

Goebel, R. (1990b). A connctionist approach to high-level cognitive modeling. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, Cambridge, MA.

Gray, M. C. & Singer, W. (1989). Stimulus-specific neuronal oscillations in orientation columns of cat visual cortex. *PNAS USA*, 86, 1698-1702.

Hadley, R. F. (1990). Connectionism, rule following and symbolic manipulation. In *Proceedings of the eigth national conference on artificial intelligence.*

Hinton, G. .E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In D. E. Rumelhart and J. L. McClelland (Eds.). *Parallel Distributed Processing.* Volume I, MIT Press, Cambridge, MA.

Hinton, G.E. & Plaut, D. C. (1987). Using fast weights to deblur old memories. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society*, Seattle, WA.

Horn, D. & Usher, M. (1990) Associative transitions in an oscillatory neural network. School of Physics and Astronomy, Tel Aviv University.

Kanwisher, N. G. (1987). Repetition blindness: Type recognition without token individuation. *Cognition*, 27, 117-143.

Lange, T. E., Vidal, J. J., & Dyer, M. G. (1989). Artificial neural oscillators for inferencing. Technical Rep. UCLA-AI-89-11.

Mozer, M. C. (1989). A Focused Back-Propagation Algorithm for Temporal Pattern Recognition. *Complex Systems*, 3, 349-381.

Mozer, M. C. (1990). Discovering faithful 'wickelfeature' representations in a connectionist network. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, Cambridge, MA.

Mozer, M. C. (in press). *The perception of multiple objects: a connectionist approach.* MIT Press / Bradford Books.

Newell, A. (1980). Physical symbol systems. *Cognitive Science*, 4, 135-183.

Norman, D. A. (1986). Reflections on Cognition and Parallel Distributed Processing. In D. E. Rumelhart and J. L. McClelland (Eds.). *Parallel Distributed Processing.* Volume II, MIT Press, Cambridge, MA.

Pinker, S. & Prince, A. (1988). On Language and Connectionism. Analysis of a Parallel Distributed Processing Model of Language Acquisition. *Cognition*, 28, 73-193.

Pollack, J.B. (1988) Recursive Auto-Associative Memory. Devising Compositional Distributed Representations. In *Proceedings of the Tenth Annual Conference of the Cognitive Science Society.* Hillsdale, NJ.

Rumelhart, D.E., & McClelland, J.L. (1986). On Learning the Past Tenses of English Verbs. In D. E. Rumelhart and J. L. McClelland (Eds.). *Parallel Distributed Processing.* Volume II, MIT Press, Cambridge, MA.

Shastri, L. (1988) *Semantic Nets: An Evidential Formalization and Its Connectionist Realization.* Los Altos, CA: Morgan Kaufmann.

Shastri, L. & Ajjanagadde, V. (1990). From simple associations to systematic reasoning. a connectionist representation of rules, variables and dynamic bindings, Tech. Report MS-CIS-90-05, Dept. of Computer Science, Univ. of Pennsylvenia.

Shiffrin, R. M., & Schneider, W. (1977). Controlled and automatic human information processing: II. Perceptual learning, automatic attending, and a general theory. *Psychological Review*, 84, 127-190.

Smolensky, P. (1988). On the proper treatment of connectionism. *The Behavioral and Brain Sciences.* 11, 1-23.

Smolensky, P. (in press). Tensor product variable binding and the representation of symbolic structures in connectionist networks. *Artificial Intelligence.*

Touretzky, D.S. and Hinton, G.E. (1988). A distributed connectionist production system. *Cognitive Science*, 12(3), 423-466.

Treisman, A. & Schmidt, H. (1982). Illusory conjunctions in the perception of objects. *Cognitive Psychology*, 14, 107-141.

Von der Malsburg, C. (1981) The correlation theory of brain function. Internal Report 81-2, Department of Neurobiology, Max-Planck-Institute for Biophysical Chemistry.

Von der Malsburg, C. & Schneider, W. (1986) A neural cocktail-party processor. *Biological Cybernetics*, 54, 29-40.

Wang, D., Buhmann, J. & von der Malsburg, Ch., (1990). Pattern Segmentation in Associative Memory. *Neural Computation*, 2, 94-106.

Analogical Retrieval Within a Hybrid Spreading-Activation Network*

Trent E. Lange
Eric R. Melz
Computer Science Department
University of California, Los Angeles
Los Angeles, CA 90024

Charles M. Wharton
Keith J. Holyoak
Department of Psychology
University of California, Los Angeles
Los Angeles, CA 90024

ABSTRACT

The mechanisms by which people are initially reminded of analogies remain poorly understood. Psychological evidence suggests that memories retrieved tend to be semantically similar and structurally consistent with the new situation. This paper describes a hybrid symbolic/connectionist model of analogical retrieval that is influenced by the language understanding process. In this model, activation is spread through a semantic network that disambiguates and makes inferences about the input cue. A network of units is then dynamically created to represent competing mappings between the cue's winning interpretation and the activated subset of potential targets. Connections created from the semantic network to these new mapping units imposes priming by semantic similarity, with links between mapping units enforcing structural consistency with the cue. The mapping network then settles by constraint satisfaction into the most coherent analogical match to the cue, after which the winning mapping units feed activation back into the semantic network, where the most highly-activated target is retrieved as the analog.

1. INTRODUCTION

Human memory retrieval involves more than just matching text against items in memory. Comprehension processes, such as disambiguation and inferencing, will alter the effective retrieval cue. Thus a realistic model of episodic reminding must integrate the process by which the retrieval cue is understood with the process by which it is used to recall information from memory. Considerable evidence indicates that a primary influence on reminding is the degree of direct semantic similarity between the cue and objects in memory (Gentner & Landers, 1985; Gick & Holyoak, 1980; Holyoak & Koh, 1987; Ross, 1989). Though the evidence is not as compelling as for semantic similarity, some recent work has shown that structural consistency (i.e., analogy) also influences the retrieval process. *Structural consistency* requires that if two propositions are placed in correspondence, then their predicates and arguments should also correspond (Falkenhainer, Forbus, & Gentner, 1989; Holyoak & Thagard, 1989). Figure 1 illustrates a simple example of variation in structural consistency. Suppose a person has studied the sentences *The sailboat followed the dolphin* and *The porpoise followed the ferry*, and is then cued with *The boat followed the whale*. If the cue is viewed as being mapped to potential targets in memory, then the former target yields a consistent mapping in which similar objects fill the corresponding agent and object roles, whereas the latter target generates an inconsistent "cross mapping" in which similar objects play dissimilar roles. Ross (1989) found that cross mapping impaired retrieval of formulas to solve story problems when the analogs involved similar objects.

We believe, however, that the effect of analogy on reminding will be influenced by several other factors. First, cue/target semantic similarity is a necessary condition for structural consistency to affect reminding. If two situations are dissimilar, then the retrieval cue will likely fail to make contact with (activate) a stored representation of the individual concepts, in which case configural properties will be irrelevant. Second, there is considerable evidence that human memory is sensitive to retrieval interference effects (Nickerson, 1984; Crowder, 1976; Barnes & Underwood, 1959). Because of retrieval competition, a stored potential analog that maps inconsistently to the retrieval cue may be less likely to be recalled if a rival analog with a consistent mapping to the cue is also stored in memory (see Figure 1).

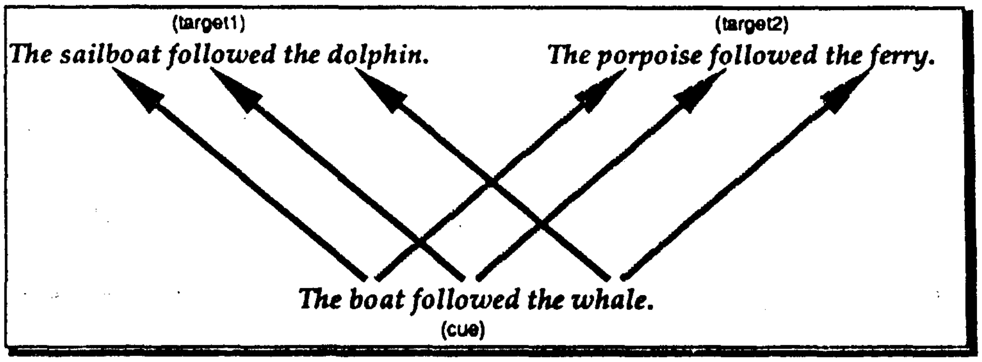

(target1)
The sailboat followed the dolphin.
(target2)
The porpoise followed the ferry.
The boat followed the whale.
(cue)

Figure 1. Cue that shares similar concepts with two targets, but maps consistently to one only (heavy arrows).

Finally, the impact of structural consistency and retrieval competition influences and is influenced by the comprehension processes involved in lexical disambiguation. The reversal of case role fillers, which can alter the structural consistency of a mapping, can also alter preferred interpretations of individual lexical items. For example, the fish in *The surfer ate the fish* is small, dead, and cut up, whereas the fish in *The fish ate the surfer* is very large, alive, and whole. In such cases, a role reversal can effect the interpretation of lexical items, which in turn can alter the similarity of individual concepts in the cue to the concepts in a stored potential analog, as well as altering configural resemblance. The inferences needed for the comprehension process are also crucial to retrieval — without a minimal understanding of how the concepts and actions of a cue are related, it is unlikely that a reasoner will retrieve a proper analogy to a given cue. This is especially true if the potential memories are indexed in ways that can only be inferred indirectly from the cue.

This paper describes SAARCS (Spreading-Activation Analog Retrieval by Constraint Satisfaction), a hybrid symbolic/connectionist model that performs both language comprehension and analogical retrieval in order to model the effects of inferencing and disambiguation on the memory retrieval process.

2. CONNECTIONIST MODELS

Symbolic artificial intelligence and case-based reasoning models (e.g., Kolodner, 1984; Owens, 1989) have so far been the main models to integrate complex language understanding and analogical (or episodic) retrieval. Connectionist models, however, have appeal because of their massive parallelism, their generally simpler and more universal processing mechanisms, and their potentially more realistic cognitive performance. Several classes of connectionist models exist, each with its own strengths and weaknesses for language understanding and retrieval.

2.1. DISTRIBUTED MODELS

Distributed connectionist (or PDP) models represent knowledge as patterns of activation across a large number of simple numerical processing units. They are valuable because they can perform noise-resistant associative retrieval, are robust against damage, and can usually be trained by learning mechanisms that perform stochastic category generalization. However, few distributed models have been built to perform semantic natural language understanding. Two of the most advanced models are those of Miikkulainen and Dyer (1989) and St. John (1990), which are able to use recurrent networks to learn to process short texts based on scripts (such as going to a restaurant) or script-like stories (Schank & Abelson, 1977). Unfortunately, there may be significant problems in scaling such distributed models to more complex language understanding that goes beyond recognizing and instantiating previously-trained scripts. Much of language understanding involves the inference of causal relationships between events for completely novel stories in which no script or previously-trained input/output pair can be recognized. This requires *dynamic inferencing* — a process of constructing chains of inferences over simple known rules, with each inference resulting in a potentially novel intermediate state (Touretzky, 1990).

Other distributed models explicitly encode variables and rules, such as the models of Derthick (1988), Touretzky and Hinton (1988), and Dolan and Smolensky (1989). Because of this, they are able to perform some of the dynamic inferencing necessary for language understanding. Unfortunately, however, the types of rules they can currently encode are generally limited. More importantly, they are serial at the knowledge level because they can fire only one rule at a time. This is a serious drawback for natural language understanding, particularly for ambiguous text, in which multiple alternative interpretations must often be explored in parallel (Dyer, 1990).

2.2. MARKER-PASSING MODELS

On the other end of the connectionist spectrum are marker-passing models, which spread symbolic markers (i.e., structured pointers) across labelled semantic networks in which concepts are represented by individual nodes. Their symbolic nodes are far more complex than those of distributed networks, and they do not have distributed networks' learning mechanisms. However, they are able to easily represent variable bindings with their symbolic markers, and more importantly, propagate them across the network in parallel to generate inference paths representing alternative interpretations of the input. Because of this, marker-passing models have been able to perform much of the high-level inferencing necessary for language understanding (cf. Charniak, 1986; Granger, Eiselt, & Holbrook, 1986; Norvig, 1989; Riesbeck & Martin, 1986). The drawback to marker-passing models is that because of the generally all-or-nothing nature of marker-generated paths, they must use a separate symbolic path evaluation mechanism to select among the alternative paths generated. This presents a serious problem for ambiguous text (Lange, in press).

2.3. SPREADING-ACTIVATION MODELS

Structured spreading-activation models are connectionist models that represent knowledge in semantic networks like those of marker passing networks, but in which the nodes are simple numeric units with weighted interconnections. The activation on each conceptual node generally represents the amount of *evidence* available for its concept in a given context. As in marker-passing networks, structured connectionist networks have the potential to pursue multiple candidate interpretations of a story in parallel as each interpretation is represented by activation in different local areas of the network. In addition, however, structured spreading-activation networks are ideally suited to perform lexical disambiguation because it is achieved automatically as related concepts under consideration provide graded activation evidence and feedback to one another in a form of analog constraint relaxation (cf. Cottrell & Small, 1982; Waltz & Pollack, 1985).

On the other hand, structured networks have had difficulties representing variable bindings and encoding rules. While recent structured spreading-activation models have some variable binding and inferencing abilities (e.g., Ajjanagadde & Shastri, 1989; Barnden, 1990; Holldobler, 1990), they resemble marker-passing models in that the networks no longer perform disambiguation. An exception is ROBIN (Lange & Dyer, 1989), a structured spreading-activation model which propagates *signature* activation patterns to generate possible interpretations of an input text, while at the same time using the network's evidential constraint satisfaction to select the most plausible interpretation in a given context. However, even with the variable binding and rule-firing abilities of recent structured models, such purely spreading-activation models do not use the powerful symbolic search and comparison mechanisms of marker-passing systems, features that are potentially useful for analogical retrieval.

3. A HYBRID SPREADING-ACTIVATION MODEL OF DISAMBIGUATION AND RETRIEVAL

SAARCS is a hybrid symbolic/connectionist model that integrates language comprehension and analogical retrieval. The system combines aspects of a language understanding model (Lange & Dyer, 1989; Lange, in press) with a model of analog retrieval (Thagard, Nelson, Holyoak, & Gochfeld, in press). Because we are interested in modelling the processes of disambiguation and inferencing and their effects on analogical retrieval, SAARCS combines marker-passing with a structured spreading-activation network in a single integrated model.

The previous retrieval model to which SAARCS is related is ARCS (Thagard *et al.*, in press), which uses a parallel constraint-satisfaction approach involving sensitivity to configural relations and competitive retrieval. Thagard *et al.* (in press) describe similarities and differences between ARCS and earlier retrieval models that lack some of its key properties (e.g., Anderson & Thompson, 1989; Carbonell, 1983; Hammond, 1989; Kolodner, 1984). None of these models, including ARCS, deals directly with comprehension and its effects on recall. Related models of comprehension include the work of Kintsch (1988) and Lange and Dyer (1989).

SAARCS consists of a structured connectionist network that encodes a knowledge base of concepts (e.g., objects, actions, plans, and goals) and general knowledge rules for inferencing between concepts. Also indexed into this semantic network are nodes representing long-term memory episodes that are potential targets for retrieval. Using this network, the understanding and analog retrieval process consists of four major stages:

(1) Activation is spread through the semantic network to disambiguate and infer an interpretation of the cue.

(2) Symbolic markers (cf. Charniak, 1986) are propagated from the nodes of the winning inference path to find the targets that are semantically similar in the current context to the cue's interpretation.

(3) A network of units is dynamically built to represent the possible competing mappings between the cue's interpretation and the semantically similar targets found by the spread of markers. The excitatory and inhibitory connections between units of this new mapping network enforce semantic and structural consistency with the cue.

(4) The new mapping network is settled by a constraint-satisfaction process; the mapping units active after settling constitute the most coherent match to the cue.

Because the units in the mapping network formed by the spreading-activation and marker-passing process feed back into the corresponding units in the semantic network, the activation of the target most semantically and structurally similar to the cue increases. The target episode in the semantic network with the highest activation is retrieved.

3.1. UNDERSTANDING AND CUE DISAMBIGUATION: ROBIN

As previously mentioned, SAARCS is built upon ROBIN (Lange & Dyer, 1989), a structured connectionist model

```
(FRAME Ingest-Food Action (Roles       (Actor  (Animate       0.05))
                                       (Object (Edible-Object 0.60)))
                          (Phrase      (P-Devoured            1.0 (Actor  Subject))
                                                                  (Object Direct-Obj))
                          (Plan-For    (G-Satisfy-Hunger      0.7 (Actor  Actor))
                          (Refinements (Carnivore-Ingest-Human 1.0 (Actor  Actor)
                                                                  (Object Object))
                                       (Human-Ingest-Food     1.0 (Actor  Actor)
                                                                  (Object Object))
```

Figure 2: Simplified definition of the frame representing the action Ingest-Food. The weights (numbers) from each of the concepts correspond to the relative amount of evidence they provide for Ingest-Food.

that uses these features to allow it to perform lexical and pragmatic disambiguation and reinterpretation (Lange, in press), while also being able to represent the variable bindings and perform some of the general knowledge rules necessary for high-level inferencing and understanding. This section gives a short overview of how ROBIN (and therefore SAARCS) performs inferencing, but a detailed description may be found in (Lange & Dyer, 1989).

ROBIN uses structured networks of simple connectionist nodes to encode semantic networks of frames representing world knowledge. Each frame has one or more roles, with each role having expectations and selectional restrictions on its fillers. Every frame is related to one or more other frames, with pathways between corresponding roles (representing general knowledge rules) for inferencing.

As in nearly all structured models, ROBIN's knowledge base is hand-built. Figure 2 gives an example of how knowledge is defined in ROBIN (and SAARCS). It defines the conceptual frame Ingest-Food, which represents a general eating action, and which has two roles: an Actor that does the eating (which must be an Animate), and an Object that is being eaten (which must be an Edible-Object). The rest of Figure 2 defines Ingest-Food's relations to other frames: it is directly accessed by the phrase P-Devoured (as in *The man devoured the steak*), it is a Plan-For the goal G-Satisfy-Hunger, and it has two possible refinement frames, Carnivore-Ingest-Human and Human-Ingest-Food[1]. The relations and their role correspondences shown in the figure also define the network's general knowledge rules, e.g.:

```
R1: [Subject X P-Devoured Direct-Obj Y]
        == phrase ==>
    [Actor X Ingest-Food Object Y]
```

```
R2: [Actor X Ingest-Food Object Y]
        == refinement ==>
    [Actor X Carnivore-Ingest-Human Object Y]
```

Finally, the numbers in Figure 2 represent the connection weights from those concepts to Ingest-Food, and are chosen on the basis of how much evidence they provide for it. For example, if a human has just eaten something (Human-Ingest-Food), then the network can definitely infer that an Ingest-Food has happened, so the weight from Human-Ingest-Food to Ingest-Food is maximal (1.0). On the other hand, if something edible has been mentioned in a story, then there would be substantial, though not certain, evidence that it will be eaten, so a corresponding "middling" weight of 0.5 from Edible-Object to Ingest-Food's Object role is given[2].

Definitions of frames such as that of Ingest-Food describe the knowledge base needed for a given domain. These definitions are used to *construct* the actual networks' structure before any processing begins. The basic networks thus constructed are much like other structured connectionist models, with a single node in the network representing each frame or role concept. Relations between concepts are represented by weighted connections between the nodes. Activation on frame and role nodes is *evidential*, corresponding to the amount of evidence available for that concept and the likelihood that it is selected in the current context.

Simply representing the amount of evidence available for concepts, however, is not sufficient for complex inferencing and understanding tasks. The network must also be capable of representing dynamic variable bindings and allow them to be propagated according to general knowledge rules. ROBIN handles this problem by having additional structure in the network which can hold uniquely-identifying activation patterns, termed *signatures*. Every concept in the network has a set of *signature nodes* that output its signature, a constant activation pattern different from all other signatures. A dynamic binding exists

[1] Refinements of frames are useful because they allow more specific inferences to be made when role-bindings are known. For example, if the network has inferred that a human is eating or wants to eat (Human-Ingest-Food), then it could infer that possible plans for it are cooking or going to a restaurant. However, quite different inferences would be made were a crocodile to eat a person (Carnivore-Ingest-Human).

[2] The actual weights chosen are clearly arbitrary. What is important is that they be in a *range* reflecting the amount of evidence the concepts provide for their related concepts in a certain knowledge base.

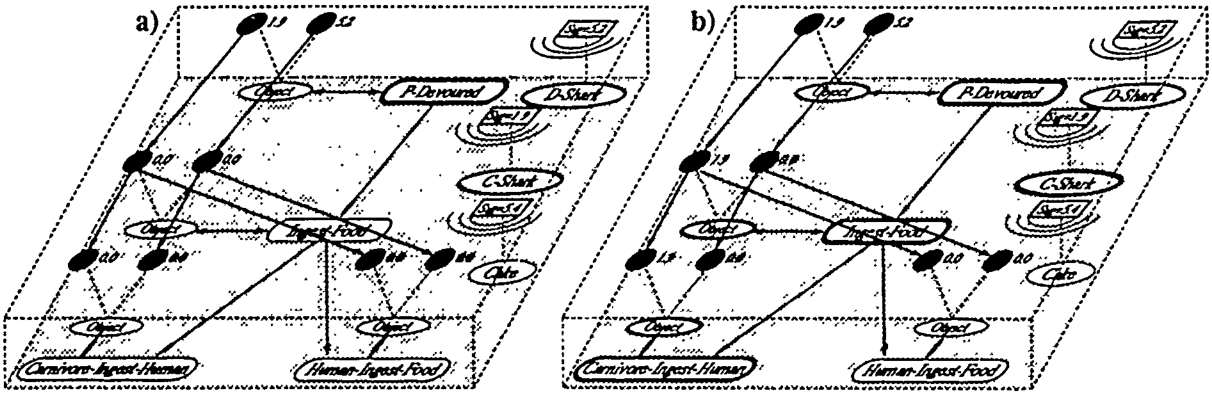


Figure 3. Simplified ROBIN network segment at two different cycles during processing of *The shark ate the diver*. Each figure shows the parallel paths over which evidential activation (bottom plane) and signature activation (top plane) are spread for inferencing. Signature nodes (outlined rectangles) and binding nodes (solid black circles) are in the top plane. Thickness of node boundaries (ovals) represents their levels of evidential activation. (Node names do not affect the spread of activation in any way. They are simply used to initially set up the network's structure and to aid in analysis.)

when a role or variable node's *binding nodes* hold an activation pattern matching the activation pattern of the bound concept's signature.

Figures 3a and 3b illustrate how the network's structure automatically propagates signatures to perform inferencing over rules such as R1 and R2. For simplicity, the signatures are arbitrarily-chosen unique scalar values. Evidential activation for disambiguation is spread through the paths between conceptual nodes on the bottom plane (i.e., Ingest-Food and its Object role), while signature activation for dynamic role-bindings is spread across the parallel paths of corresponding binding nodes (solid black circles) on the top plane. Nodes and connections for the Actor roles are not shown. Initially there is no activation on any of the conceptual or binding nodes.

Figure 3a shows the state of the network after it has been presented with input fo. the simple sentence *The shark devoured the diver* (**Killer Shark**). In this network, the word *shark* has two alternative meaning senses, C-Shark (a large, carnivorous shark) and D-Shark (a cut-up dinner shark). When input for **Killer Shark** is presented, the lexical concept nodes for each of the words in the phrase are clamped to a high level of evidential activation, directly providing activation for concepts C-Shark, D-Shark, P-Devoured, and Diver. To represent the role-bindings given by the phrase, the binding nodes of each of P-Devoured's roles are clamped to the signatures of the concepts bound to them[1]. The binding nodes of its Actor role are clamped to the activations (1.9 and 5.2) of the signatures for C-Shark and D-Shark (Figure 3a), representing the candidate bindings from the word *shark*, and a binding node of its Object role is bound to the signature for Diver (not shown).

The activation of the network's conceptual nodes is equal to the weighted sum of their inputs plus their previous activation times a decay rate. The activation of the binding nodes, however, is equal to the maximum of their unit-weighted inputs, allowing signatures to be propagated without alteration.

As activation starts to spread after the initial clamped activation values in Figure 3a, Ingest-Food receives evidential activation from P-Devoured, representing the strong evidence that something has been eaten. Concurrently, the signature activations on the binding nodes of P-Devoured's roles propagate to the corresponding binding nodes of Ingest-Food (Figure 3b), since each of the binding nodes calculates its activation as the maximum of its inputs. Notice, however, that the network does not allow role bindings to propagate blindly — the links are actually gated by nodes in the network (not shown) that calculate when a frame's selectional restrictions have been violated. In this case the signature for D-Shark (5.2) did not propagate to the corresponding binding node of Ingest-Food's Actor (Figure 2), since it expects only Animates as its filler[2]. Similarly, as time goes on, evidential and signature activation reach Carnivore-Ingest-Human, but not Human-Ingest-Food (a human was being eaten, and not doing the eating) (Figure

[1] ROBIN does not currently address the problem of deciding upon the original syntactic bindings, i.e. that *"shark"* is bound to the Actor role of phrase P-Devoured. Rather, ROBIN's networks are given these initial bindings and use them for high-level inferencing.

[2] The network structure that gates the spread of activa tion based on selectional restrictions (essentially by comparing the signature activations of the bindings to those of the selectional restrictions) is not important for this paper, but is described in (Lange & Dyer, 1989).

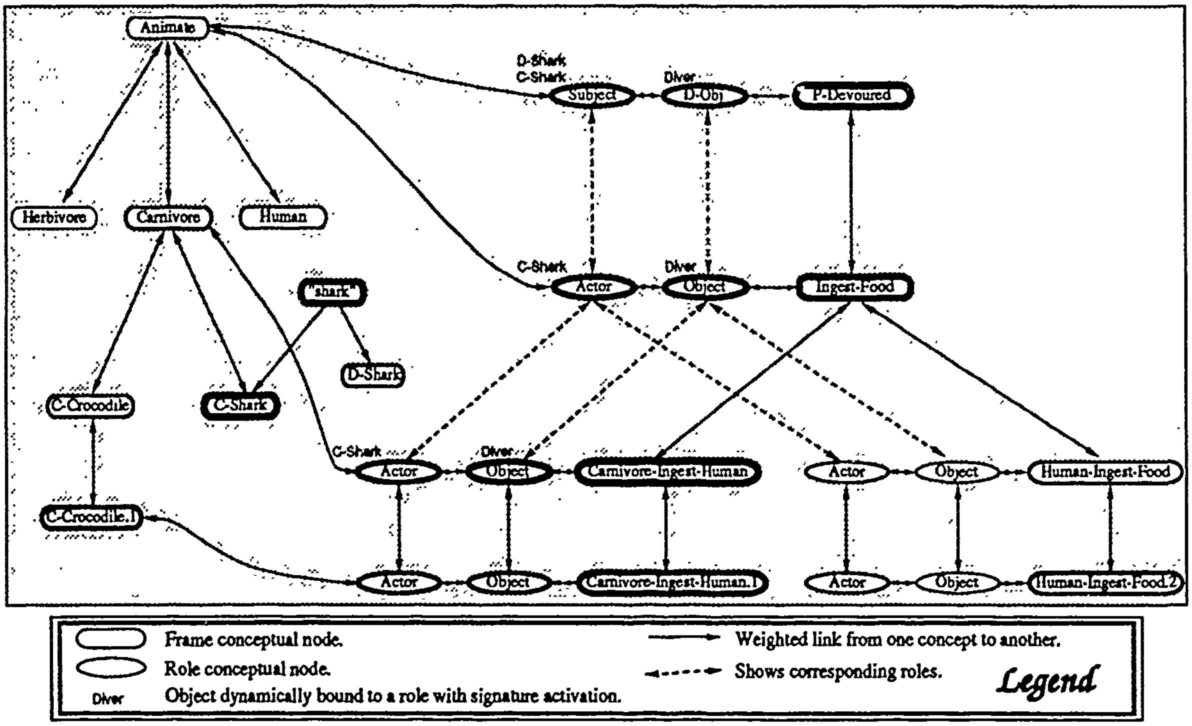


Figure 4. Expanded area of Figure 3 showing how two target episodes, Carnivore-Ingest-Human.1 and Human-Ingest-Food.2, are connected to the network. Shown are the activation of the evidential layer's nodes after presentation of input for *The shark devoured the diver*. The labels next to roles (i.e. D-Shark, C-Shark, and Diver) show the concepts inferred over the binding nodes with the signatures.

3b). Inferencing would continue in a larger network having specific frames related to the newly-inferred instantiation of Carnivore-Ingest-Human.

When the network has settled, C-Shark ends up with more evidential activation than D-Shark (thicker node boundary), so that the word *shark* has been disambiguated to the large, carnivorous kind. The interpretation of the input is then the most-highly activated path of frames and their signature role-bindings, in this case simply (Actor C-Shark Ingest-Food Object Diver) ↔ (Actor C-Shark Carnivore-Ingest-Human Object Diver).

As can be seen, inferencing is crucial to understanding even simple sentences such as **Killer Shark**. If the original phrase's role-fillers had been reversed (*the diver ate the shark*), then D-Shark would have won because of feedback from the role-bindings allowed to propagate to Human-Ingest-Food. The role of inferencing and the evidential combination of the evidential network is even more important in more complicated examples, such as *John put the pot inside the dishwasher because the police were coming*, which requires a complex plan/goal analysis to understand (Lange & Dyer, 1989).

3.2. FINDING SIMILAR TARGETS

SAARCS' networks are built from ROBIN's networks, but in addition have nodes representing the episodes in its long-term memory. Each of the elements of these episodes is an instance of a frame in the semantic network, and so is connected (without signature binding paths) to the evidential nodes of those frames. The strength of those weights is relative to how "well" the episodes have been remembered: particularly salient episodes will have high connection weights, and "fading" memories will have low connection weights.

Figure 4 shows an example of how a simple episode is connected to the network. In figure 4, the episode *The crocodile ate the swimmer* is represented by the instance Carnivore-Ingest-Human.1, whose Actor is connected to C-Crocodile.1 and whose Object is connected to Swimmer.1 (not shown). *The sailor consumed the fish* is represented by Human-Ingest-Food.2.

The spread of activation used to understand the input has the side effect of activating targets that are semantically similar to the interpretation of the cue. For example, in Figure 4, which shows the results of the processing of *The shark ate the diver* (from figure 3), the target Carnivore-Ingest-Human.1 has become strongly activated due to activation from Carnivore-Ingest-Human.

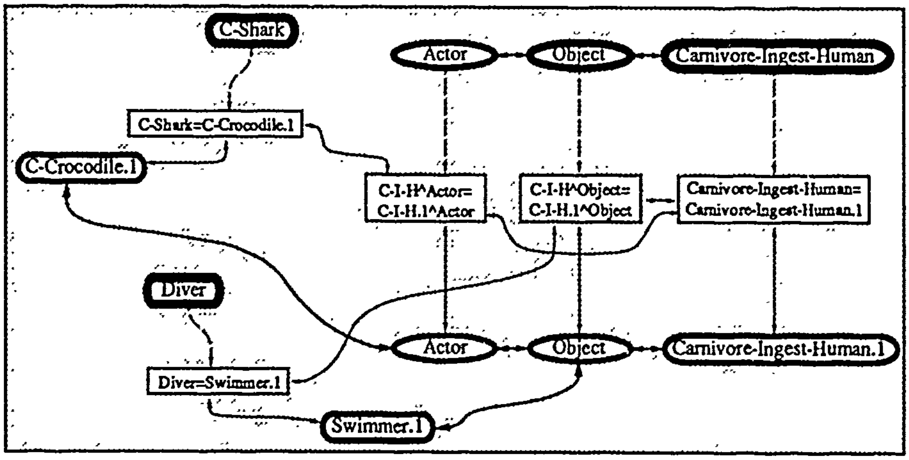


Figure 5. Some of the mapping units (rectangles) created by passing of markers in **Killer Shark.** All links shown are excitatory. Unidirectional dashed lines have weights proportional to the total weight distance between that particular concept and the target being mapped.

To start memory retrieval, symbolic markers are spread from the frame and role nodes of the cue's winning interpretation. These markers hold both a symbolic backpointer to their originating node and a strength equal to the numeric product of the link weights they have propagated over. The frame and role markers only propagate over links between corresponding frames and roles, respectively, and only over active portions of the network.

This propagation of markers finds, in parallel, all of the instances in memory that are semantically similar to the cue in the current context. Equally important is that the markers' backpointers tell exactly which part of the cue they are similar to. For instance, in Figure 4, one marker will reach Carnivore-Ingest-Human.1 from the inferred Carnivore-Ingest-Human, and another marker will reach C-Crocodile.1 from C-Shark[1].

This marker-passing naturally constrains the search for similar targets because of two features of the network: (1) instances not semantically similar to the cue in the current context will have little or no activation, and so will not be reached (e.g., Herbivore), and (2) instances that are active, but which are semantically distant from a cue concept (such as C-Crocodile.1 from Diver) will not be reached because of their separation in the network.

3.3. BUILDING THE MAPPING NETWORK

The marker-passing process finds large numbers of partially-active long-term instances that are semantically similar to part of the cue. Each of these correspondences are potential analogs. However, to retrieve a single coherent episode most analogous to the cue, these isolated correspondences must compete against each other. This competition is driven by parallel satisfaction of the two main types of constraints, semantic similarity and structural consistency, that are believed to operate in both analogical retrieval (Thagard *et al.*, in press) and analogical mapping (Holyoak & Thagard, 1989).

To perform this competition, a mapping network is dynamically formed whose units represent the possible mappings between each pair of semantically similar concepts, as in the ARCS model of analogical retrieval (Thagard *et al.*, in press). In SAARCS, these are the pairs found by propagation of the markers. For example, in **Killer Shark**, markers hitting nodes for the target *The crocodile ate the swimmer* would cause mapping units to be created for the hypotheses that C-Shark=C-Crocodile.1, Diver=Swimmer.1, and Carnivore-Ingest-Human=Carnivore-Ingest-Human.1. Target roles that receive markers also create units representing the possible mappings between those roles and the markers' originating roles. This in itself enforces partial structural consistency, since only corresponding roles that can be reached over inferencing paths (dashed lines in Figure 4) will receive markers. The units created for the potential mappings between **Killer Shark** and *The crocodile ate the swimmer* are shown in Figure 5.

As in ARCS, structural consistency is enforced by excitatory connections between corresponding mapping units. As shown in Figure 5, excitatory connections are created between units mapping two roles (e.g., C-I-H^Actor=C-I-H.1^Actor) and the units mapping their frames (e.g.,

[1] Markers are used here rather than signatures since each instance in the targets may be semantically similar to multiple concepts in larger cue stories, and thus need to hold several markers at once. Since signatures are activation patterns, binding units can hold only one signature "backpointer" at a time. This is also true of other current spreading-activation variable binding approaches, such as the use of different segments of a phased clock (Ajjanagadde & Shastri, 1989).

Carnivore-Ingest-Human=Carnivore-Ingest-Human.1). Units mapping two concepts that serve as the fillers of two mapped roles also have excitatory connections (e.g., between C-Shark=C-Crocodile.1 and C-I-H^Actor=C-I-H.1^Actor).

All of the above types of connections between structurally consistent mapping units have a small positive value (0.05). Excitatcry weights are also constructed to mapping units from the nodes in the semantic network that they map, with the link weights being proportional to the total path weight product between the concepts (0.05 * strength of the marker that caused the mapping unit to be built). These weights thus give importance to both (a) semantic similarity, since the weights to mapping units for two very similar concepts will be higher than those for two less similar concepts, and (b) pragmatic relevance, since important and relevant goals will have more basic activation in the semantic network, thus biasing retrieval towards units mapping those goals.

Competition between potential mappings is facilitated by inhibitory links between all rival mappings (-0.20 in the simulation). For instance, there will be an inhibitory link between Diver=Swimmer.1 and the mapping unit created for Diver=Sailor.2 (from the target *The sailor consumed the fish*).

3.4. COMPETITION AND RETRIEVAL

During and after creation of the units representing candidate analogical mappings, the new mapping network is settled using a constraint-satisfaction algorithm. The mapping units that are most active after the network has settled will be those that constitute the most coherent match to the cue (Thagard *et al.*, in press).

Because mapping units are created with bi-directional links from their target nodes in the semantic network (e.g., from C-Shark=C-Crocodile.1 to C-Crocodile in Figure 5), activation from the winning mappings feeds back into the targets. This boosts the evidential activation of targets most analogous to the cue, so that they tend to become highly activated. The target retrieved is the episode with the highest evidential activation.

4. SIMULATION RESULTS

SAARCS has been implemented in the DESCARTES connectionist simulator (Lange, Hodges, Fuenmayor, & Belyaev, 1989). The model has been tested on three different types of competitive and non-competitive retrieval: examples in which the targets can be retrieved solely on the basis of semantic similarity after interpretation, examples in which analogical similarity plays a crucial role, and examples in which plan/goal analyses of the cue must be made before retrieval is possible.

The first class simulated are retrievals in which a single target is clearly the most semantically similar to the cue after interpretation. In such examples, the interpretation process activates the similar target much more highly than any others. This is the case when *The crocodile ate the swimmer* and *The sailor consumed the fish* are the closest potential targets for *The shark ate the diver*. The carnivorous crocodile episode is so similar to **Killer Shark** that it becomes highly active just from the inferencing process, especially in relation to *The sailor consumed the fish* (see Figure 4). Structural similarity pressures from the mapping network only marginally helps retrieval in these kind of cases.

In other cases, however, multiple targets can have approximately the same semantic similarity to the cue, so structural consistency plays a larger part in retrieval. An example of this is that the target *The sailboat followed the dolphin* (Ptrans-Follow.1) is a better analogy for *The boat followed the whale* than is *The porpoises followed the ferry* (Ptrans-Follow.2). In this kind of case, the pressures due to structural consistency allow the better analogy to be retrieved first.

Figure 6a shows the activations of these target episodes during retrieval in SAARCS. Activation reaches the two targets after presentation of the cue at cycle 16, and the semantic network settles by about cycle 39. Although activations of the cue (not shown) clearly indicate that a Boat was Ptrans-Following a Whale, the activations of the two targets is about the same, essentially because they both involve sea mammals and boats following each other. At this point, markers propagate from the cue's interpretation, so that at cycle 41 the competing mapping units for Ptrans-Follow=Ptrans-Follow.1 and Ptrans-Follow=Ptrans-Follow.2 are formed (Figure 6b). Because of the excitatory connections enforcing structural similarity between them and the other newly created mapping units, Ptrans-Follow=Ptrans-Follow.1 soon begins to win, dominating by about cycle 80. This activation feeds back into the semantic network, driving Ptrans-Follow.1 to saturation and allowing Ptrans-Follow.2 to decay. *The sailboat followed the dolphin* is thus retrieved as the best analogy for the cue.

The final set of simulations have tested SAARCS' ability to perform retrievals which require a plan/goal analysis of the cue. For instance, to understand *John put the pot inside the dishwasher because the police were coming*, the ROBIN portion of the network must first make multiple inferences to decide that John was most likely trying to hide his marijuana from the police inside an opaque object (the dishwasher), because he didn't want to get arrested. These inferences combined with ROBIN's spread of activation allow the network to disambiguate and form an interpretation of the cue (see Lange & Dyer, 1989).

Once a reinterpretation has been made to Hiding Marijuana due to confluences of evidential activation from the inferred plan/goal analysis, SAARCS' retrieves the analogous episodes *Bill hid the cocaine in the stove so that he wouldn't be arrested*, as opposed to the previously most analogous episode *Mary put the cooking pot in the dishwasher to clean it.*

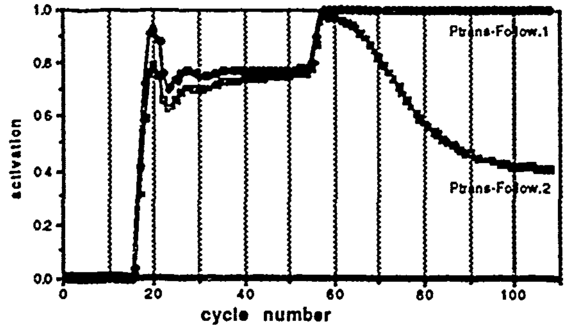


Figure 6a. Activation of target nodes Ptrans-Follow.1 (*The sailboat followed the dolphin*) and Ptrans-Follow.2 (*The porpoises followed the ferry*) after presentation of *The boat followed the whale.*

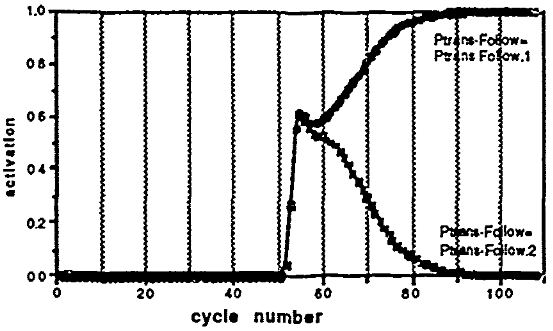


Figure 6b. Activation of mapping units Ptrans-Follow=Ptrans-Follow.1 and Ptrans-Follow=Ptrans-Follow.2 after they were created due to presentation of *The boat followed the whale.*

5. DISCUSSION

Although it is in the early stages of testing, SAARCS integrates language understanding and analogical retrieval, and does so in a manner that is driven by the primary constraints that seem to drive analogical reminding: semantic and structural constraints. To review, SAARCS' networks have three interacting spreading-activation structures, each performing different tasks in parallel: the permanent signature role-binding and inferencing paths, the permanent evidential semantic network structure, and the dynamically-constructed mapping network.

One of the main strengths of SAARCS is the signature inferencing paths and evidential semantic network structure from ROBIN (Lange & Dyer, 1989). Propagation of signatures allows SAARCS to hold role-bindings and make chains of dynamic inferences, possibly activating several alternative interpretations or plan/goal analyses of the cue. The concurrent spread of evidential activation allows the network to select the most plausible of those ambiguous interpretations in a given context.

When long-term target instance nodes (episodes) are built into the evidential network by connections to the semantic frames of the inferencing network, as in SAARCS, the instances that are semantically similar to the interpretation of the cue being processed automatically become activated. Propagation of symbolic markers from the frames and roles of the cue's interpretation finds these potential analogs. Because the markers only spread across active areas of the network, the only targets that have the potential to be mapped are those that are semantically similar to part of the cue, as appears to be the case in human retrieval (Ross, 1989).

Retrieval itself is driven by the constraint satisfaction of the mapping network created by collisions of cue markers with active targets. Each collision causes a mapping unit to be built representing the possibility of that element being retrieved as the mapping for the marker's cue node. The soft pressures of semantic similarity on retrieval are imposed by connections from the evidential semantic network to the corresponding mapping units, while excitatory connections between consistent mapping units and inhibitory connections between inconsistent units impose the soft pressures of structural similarity. The mapping network then settles by constraint satisfaction into the most coherent analogical match to the cue, after which the winning mapping units feed activation back into the evidential semantic network, where the most highly-activated target is retrieved as the analog.

A final aspect about to note about SAARCS is how language understanding and retrieval processes come full circle. The analogy retrieved depends crucially on the interpretation of the cue from the spreading-activation network's inferences. Once an analogy is retrieved, it in turn primes the activation of the evidential spreading-activation network, perhaps leading to a different disambiguation and therefore interpretation of the next cue.

5.1. OTHER RETRIEVAL MODELS

Although there have been numerous symbolic and connectionist models of language understanding and analogical retrieval, few have dealt with how the two processes are integrated and effect each other. Here we compare SAARCS to the main models that can retrieve extended episodes: symbolic case-based retrieval models, and the connectionist CONPOSIT model of case-based retrieval (Barnden & Srinivas, in press) and ARCS model of analogical retrieval (Thagard *et al.*, in press).

5.1.1. Case-Based Reasoning Models

Symbolic memory retrieval models (e.g., Kolodner, 1984) and case-based reasoning models (e.g., Hammond, 1989; Kolodner, Simpson, & Sycara, 1985; Owens, 1989; Schank & Leake, 1989) attempt to retrieve the (symbolically represented) episode from memory that is most likely to help them in their current task. To do this, they search memory for episodes that share analogous goals, plans, enablements, or failures with the current problem situation, depending on the reasoning task. To limit the number of retrievals, they generally apply a symbolic/conditional classification of the current problem to index similarly-classified episodes in memory. These classifications initially search memory with fairly specific indices, but use more and more general indices each time the search fails. For example, when SWALE (Schank & Leake, 1989) attempts to find a relevant explanation for the anomalous event of a young race horse dying of a heart attack, it initially searches for episodes of premature deaths of race horses. If no episode is found, it searches for episodes of premature deaths of horses in general, and then finally of any animals.

Whenever the indexing mechanism retrieves episodes from memory, the case-based reasoning models check them for their relevance to the current problem and eliminate those that are not. MEDIATOR (Kolodner *et al.*, 1985), for example, first eliminates any episodes in which the goal relationship is different than that of the current case (a structural similarity rule), and then eliminates any episodes where the derivation of the goal relationship is different than that of the current case (a partially structural and partially semantic similarity rule). If all of the episodes were rejected, then the models may go back and generate more general indices with their classification mechanism and try again.

Both SAARCS and case-based reasoning (CBR) models use structural and semantic considerations for retrieval. SAARCS, however, has much looser semantic constraints — any form of semantic similarity that is activated between target episodes and elements of the cue or inferences from it can lead to a potential mapping, whereas CBR models only look for specific semantic relationships (indices) depending on the problem for which they are looking for a solution. The difference in enforcement of structural similarity between SAARCS and the CBR models is similar — structural differences between a cue and a target in SAARCS form a negative pressure against that target's retrieval which can be overridden by the combined activation from semantic similarity, whereas pertinent structural differences in CBR models generally completely rule-out retrieval. Part of these differences can be explained by the different objectives of SAARCS and case-based retrieval models. Most CBR models try to retrieve only those episodes that will provide useful information to the task at hand. SAARCS, on the other hand, is an attempt to model the less ideal world of general human memory retrieval, where semantic similarity appears to play a more important role, and in which remindings are not always so specific or helpful.

On the basis of ability to represent and retrieve long and complex episodes from memory, the symbolic case-based models have a definite advantage over SAARCS because of the current relative difficulty of representing and processing complex structured knowledge and rules in connectionist networks. On the other hand, most of the case-based models do not currently integrate language understanding with the retrieval process as SAARCS does. The ones that do (e.g., Kolodner, 1984) simply use normal symbolic parsers as front-ends to the retrieval system. However, for stories with the types of inferences that SAARCS is able to process, SAARCS seems more psychologically plausible and better able to handle ambiguity and its effects on the retrieval process. Symbolic parsers, for instance, have a very difficult time with ambiguous stories such as *John put the pot inside the dishwasher because the police were coming*, and so could not form the proper indices for retrieval. This is not a problem in SAARCS. Another example is the effect of priming on retrieval in SAARCS. For instance, SAARCS normally retrieves the target *Jacques Cousteau studied the sharks* when given the cue *The astronomer saw the star.* However, when the network is primed by activation of Celebrity, it interprets *star* as a Movie-Star, so instead retrieves the target *Bill saw Magic Johnson.*

5.1.2. CONPOSIT

Barnden and Srinivas (in press) have recently proposed a case-based reasoning extension to CONPOSIT, a connectionist model that performs rule-based reasoning (Barnden, 1990). CONPOSIT uses relative-position encoding to represent complex, symbolic data structures in two-dimensional "Configuration Matrix" (CM) arrays of neural subnetwork "registers", each holding a dynamically-changing activity pattern representing a symbol and binary flags. The structure of the episode in a CM is encoded by the adjacency relationships of the symbols and flags in its registers. In the CBR version of CONPOSIT, each episode is represented by a separate CM. To retrieve one of those episodes, a winner-take-all mechanism arbitrarily chooses one of the proposition symbols of the cue's CM to be broadcast to all episodic CMs. Any CM that contains a symbol matching the broadcast one adds the "degree" of that match to its "total degree of match" activation. After all of the cue's propositions have been broadcast in this manner, the CM episode retrieved is the one with the highest total degree-of-match.

Because CONPOSIT's Configuration Matrices can encode more complex structures than can currently be encoded with signatures, it can store and retrieve more complex episodes than SAARCS can. However, as in symbolic CBR models, CONPOSIT's retrieval mechanism is relatively brittle. To match a symbol in an episodic CM, the cue proposition and its role-bindings broadcast must be exactly the same. To match similar propositions with non-identical role-bindings (e.g. matching *John kissed Mary* to *Peter kissed Susan*), CONPOSIT must randomly "hide" symbols in both the cue and episodic CMs, until eventually a match is found (e.g. the above will match when *John, Mary, Peter, and Susan* are eventually all hidden). Besides being expensive, this hiding process throws out both structural and semantic similarity, so that *Billy kissed Becky* is no more likely to match *Johnny kissed Sue* than is *The beauty kissed the beast.*

One important ability that CONPOSIT has that SAARCS currently lacks is the ability to add "advice" from the retrieved episode as inferences to the cue. On the other hand, as an integrated model of language understanding and retrieval, one of CONPOSIT's main drawbacks is that such advice is its only means of performing inferencing. There thus appears to be no obvious way to handle the ambiguity of language and its effects on both understanding and retrieval, both effects that SAARCS' spreading-activation inferencing mechanism is able to simulate.

5.1.3. ARCS

ARCS (Thagard *et al.*, in press) is the analogical retrieval model most closely related to SAARCS. It does retrieval by creation of a mapping network similar to SAARCS', but which is created by normal symbolic search mechanisms. These symbolic search mechanisms look for target elements that are semantically similar to elements of the cue and decide which mapping units to connect to each other. The biggest advantage SAARCS has over ARCS is its language understanding abilities, which ARCS lacks altogether. In addition, SAARCS' spreading-activation understanding processes can potentially account for many psychological phenomena involving priming effects in human memory retrieval. These phenomena include increased retrieval due to to repetition, recency, and prior semantic priming, all of which can be modelled by variations in activation levels prior to presentation of the cue. Similarly, SAARCS can produce partial retrieval of target episodes (when some elements of the episode do not become activated), whereas ARCS can only yield all-or-none retrieval.

5.2. FUTURE WORK

In the future, there are three main areas that we would like to explore:

(1) Improved understanding and retrieval. SAARCS' analogical retrieval abilities are currently limited by the semantic network's comprehension abilities. Though the ROBIN portion of the network is capable of variable binding and some general knowledge rules, its inferencing and comprehension abilities are limited in complexity compared to symbolic models. Handling longer and more complex examples is a particularly important area of research (Lange & Dyer, 1989).

(2) Simplifying the connectionist model: It would be desirable to replace the marker-passing that SAARCS uses to find correspondences between semantically similar concepts with a purely spreading-activation approach, such as an extension to signatures. More importantly, the dynamic creation of mapping units is quite expensive computationally, so we are looking into ways to instead temporarily recruit pre-existing mapping units.

(3) Psychological Experiments: Many predictions can be made from SAARCS about the effect of language comprehension on memory retrieval, and vice versa. We are currently performing a number of psychological experiments to test these predictions.

ACKNOWLEDGEMENTS

This research was supported by Contract MDA 903-89-K-0179 from the Army Research Institute and by a grant from the Keck Foundation. We would like to thank David Touretzky and Bruce Burns for their helpful comments on previous drafts of this paper.

REFERENCES

Ajjanagadde, V., & Shastri, L. (1989). Efficient inference with multi-place predicates and variables in a connectionist system. In *Proceedings of the Eleventh Annual Meeting of the Cognitive Science Society*, Ann Arbor, MI.

Anderson, J. R., & Thompson, R. (1989). Use of analogy in a production system architecture. In S. Vosniadou & A. Ortony (Eds.), *Similarity and analogical reasoning*, p. 267-269, Cambridge: Cambridge University Press.

Barnden, J. (1990). The power of some unusual connectionist data-structuring techniques. In J. A. Barnden and J. B. Pollack (Eds.), *Advances in connectionist and neural computation theory*, Norwood, NJ: Ablex.

Barnden, J., & Srinivas, K. (in press). Overcoming rule-based rigidity and connectionist limitations through massively-parallel case-based reasoning. *International Journal of Man-Machine Studies.*

Barnes, J. M., & Underwood, B. J. (1959). 'Fate' of first-list associations in transfer theory. *Journal of Experimental Psychology, 58*, 97-105.

Carbonell, J. (1983). Learning by analogy: Formulating and generalizing plans from past experience. In R. Michalski, J. Carbonell, & T. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (pp. 137-161). Palo Alto, CA: Tioga Press.

Charniak, E. (1986). A neat theory of marker passing. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA.

Cottrell, G., & Small, S. (1982). A connectionist scheme for modelling word-sense disambiguation. *Cognition and Brain Theory, 6*, 89-120.

Crowder, R.G. (1976). *Principles of learning and memory.* Hillsdale, NJ: Lawrence Earlbaum.

Derthick, M. (1988): *Mundane reasoning by parallel constraint satisfaction.* (Tech Rep. CMU-CS-88-182), Ph.D. Dissertation, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA.

Dolan, C. P. & Smolensky, P. (1989). Tensor product production system: A modular architecture and representation. *Connection Science, 1 (1).*

Dyer, M. G. (1990). Symbolic neuroengineering for natural language processing: A multi-level research approach. In J. A. Barnden and J. B. Pollack (Eds.), *Advances in connectionist and neural computation theory*, Norwood, NJ: Ablex.

Falkenhainer, B., Forbus, K. D., & Gentner, D. (1989). The structure-mapping engine. Algorithm and examples. *Artificial Intelligence, 41*, 1-63.

Gentner, D., & Landers, R. (1985). Analogical reminding: A good match is hard to find. *Proceedings of the International Conference on Systems, Man and Cybernetics*, p. 607-613. Tucson, AZ.

Gentner, D., & Toupin, C. (1986). Systematicity and surface similarity in the development of analogy. *Cognitive Science, 10*, 277-300.

Gick, M., & Holyoak, K. J. (1980). Analogical problem solving. *Cognitive Psychology, 12*, 306-355.

Granger R. H., Eiselt, K. P., & Holbrook, J. K. (1986). Parsing with parallelism: A spreading activation model of inference processing during text understanding. In J. Kolodner and C. Riesbeck (Eds.), *Experience, memory, and reasoning*, p. 227-246. Hillsdale, NJ: Lawrence Earlbaum.

Hammond, K. (1989) *Case-based planning.* Boston: Academic Press.

Holldobler, S. (1990). A structured connectionist unification algorithm. *Proceedings of the Ninth National Conference on Artificial Intelligence*, Boston, MA.

Holyoak, K. J., & Koh, K. (1987). Surface and structural similarity in analogical transfer. *Memory & Cognition, 15*, 332-340.

Holyoak, K. J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science, 13*, 295-355.

Kintsch, W. (1988). The role of knowledge in discourse comprehension: A construction-integration model. *Psychological Review, 95*, 163-182.

Kolodner, J. (1984). *Retrieval and organizational strategies in conceptual memory: A Computer Model.* Hillsdale, NJ: Lawrence Earlbaum.

Kolodner, J., Simpson, R., & Sycara, K. (1985). A process model of case-based reasoning in problem solving. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA.

Lange, T. & Dyer, M. G. (1989). High-level inferencing in a connectionist network. *Connection Science, 1 (2)*, p. 181-217.

Lange, T. (in press). Lexical and pragmatic disambiguation and reinterpretation in connectionist networks. *International Journal of Man-Machine Studies.*

Lange, T., Hodges, J., Fuenmayor, M., & Belyaev, L. (1989). DESCARTES: Development environment for simulating hybrid connectionist architectures. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, Ann Arbor, MI.

Miikkulainen, R. & Dyer, M. G. (1989). Encoding input/output representations in connectionist cognitive systems. In D. Touretzky, G. Hinton, and T. Sejnowski (Eds.). *Proceedings of the 1988 Connectionist Models Summer School.* San Mateo, CA: Morgan Kaufmann.

Nickerson, R.S. (1984). Retrieval inhibition from part-set cuing: A persisting enigma in memory research. *Memory & Cognition, 12*, p. 531-552.

Norvig, P. (1989). Marker passing as a weak method for text inferencing. *Cognitive Science, 13*, p. 569-620.

Owens, C. (1989). Integrating feature extraction and memory search. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, Ann Arbor, MI.

Riesbeck, C. K. & Martin, C. E. (1986). Direct memory access parsing. In J. Kolodner and C. Riesbeck (Eds.), *Experience, Memory, and Reasoning*, p. 209-226. Hillsdale, NJ: Lawrence Earlbaum.

Ross, B. H. (1989). Distinguishing types of superficial similarities: Different effects on the access and use of earlier problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 15*, 456-468.

Schank, R. C. & Abelson, R. (1977). *Scripts, plans, goals and understanding.* Hillsdale, NJ: Lawrence Earlbaum.

Schank, R.C., & Leake, D.B. (1989). Creativity and learning in a case-based explainer. *Artificial Intelligence, 40*, 353-385.

St. John, M. F. (1990). *The story gestalt: Text comprehension by cue-based constraint satisfaction.* (Tech Rep. # 9004), Department of Cognitive Science, University of California, San Diego.

Thagard, P., Holyoak, K. J., Nelson, G., & Gochfeld, D. (in press). Analog retrieval by constraint satisfaction. *Artificial Intelligence.*

Touretzky, D. S. (1990). Connectionism and compositional semantics. In J. A. Barnden and J. B. Pollack (Eds.), *Advances in connectionist and neural computation theory*, Norwood, NJ: Ablex.

Touretzky, D., & Hinton, G. (1988). A distributed connectionist production system. *Cognitive Science, 12*, 423-466.

Waltz, D., & Pollack, J. (1985). Massively parallel parsing: A strongly interactive model of natural language interpretation. *Cognitive Science, 9*, 51-74.

Appropriate Uses of Hybrid Systems

Daniel E. Rose
Computer Science and Engineering Department C-014
University of California, San Diego
La Jolla, California 92093
rose%cs@ucsd.edu

Abstract

While the techniques of connectionism and symbolic AI have been applied to many problems individually, there are far fewer systems which attempt to combine these techniques. This paper presents a possible set of factors one might use to determine when such hybrid approaches are appropriate. These factors include characteristics of the problem as well as the intended uses of the system. SCALIR, a hybrid system for retrieving legal documents, is presented, and is used to illustrate the motivating factors. Problems unique to hybrid systems are discussed, and possible solutions are given.

1 Introduction

Different problems demand different solutions, in artificial intelligence (AI) as in any domain. In recent years, solutions to AI problems have fallen primarily into two classes, the traditional "symbolic" approach and the neurally-inspired "connectionist" approach. Yet the problems to which these techniques are generally applied do not fall neatly into the same two classes. In some cases, *hybrid* solutions which utilize both approaches are most appropriate. In this paper, I investigate what characteristics make a problem well-suited for solution by a hybrid system. I then consider some difficulties facing hybrid systems, and describe how one system — SCALIR, a system for conceptual retrieval of legal documents — attempts to resolve them while exploiting the advantages of connectionist and symbolic techniques.

For the purposes of this paper, I will use the term "symbolic" to refer to systems which depend on axiomatized inference methods for the explicit manipulation of discrete symbols. These characteristics are often found in the tools and techniques of mainstream AI research, such as production systems, logic, and semantic networks. In contrast, I will refer to those systems which employ unstructured statistical inference on graded, continuous representations as "associative." Their representations are often created or modified by learning procedures. The properties of associative inference are essential to most connectionist network models, but some of them can also be found in many traditional AI systems, such as MYCIN (Davis et al., 1977), which employ probabilistic representations. I will often refer to connectionist implementations of the associative paradigm.

The term "hybrid" refers to explicit combinations of symbolic and associative techniques, such as Hendler's "Marker Passing Over Microfeatures" (Hendler, 1989), Lange's ROBIN system (Lange and Dyer, 1989), and SCALIR (Rose and Belew, to appear), which will be described in Section 4. Note that this does not include what might be considered "functionally hybrid" systems — associative systems trained to replicate symbolic ones.

To understand how the hybrid approach is motivated in the case of SCALIR, it is useful to understand the system's domain, legal research. When an attorney (or paralegal) wishes to investigate a particular legal issue, he or she is often interested in finding all of the court decisions which pertain to that issue. There are several traditional sources that have been used for this task, as well as some newer, computer-based tools. Some of these are conceptually linked together by West Publishing's hierarchical taxonomy of the law.

West's taxonomy first divides law into seven general areas: Persons, Property, Contracts, and so on. These areas are divided further; for instance, "Property" contains such topics as Automobiles, Copyrights and Intellectual Property, and Franchises. Further subdivisions continue until the legal issues are quite specific, e.g. what constitutes infringement of copyright for sound recordings. Cases can be characterized by several topics, but they must be drawn from the preexisting list.

The legal researcher's first source is the set of "reporters" such as the *Supreme Court Reporter*[1] which contain the full text of all cases of a given court system, in chronological order. The most common reporters, produced by West,

[1] Supreme Court Reporter is a registered trademark of West Publishing Company.

contain additional "headnotes" before each case, summarizing how the case bears on a particular legal issue. The issues are labeled with "key numbers" which indicate their location in West's legal taxonomy.

In addition to reporters, there are "digests" which are organized by areas of the law. West's digests use the same key number system as its reporters, so a researcher can use the digest to find all cases which contained a headnote labeled by a given key number.

Finally, there is the *Shepard's Citations*[2] "citator," a backward-pointing index (similar to the Science Citation Index) which lists all cases which cited a given case. Many of the citations listed in Shepard's are annotated with a label describing the nature of the relationship. Some of these labels refer to the *history* of the case (e.g. whether the cited case was affirmed on appeal); others refer to its *treatment* (whether the decision was criticized, limited, etc., by the citing case).

Most of these traditional resources are now available online through the WESTLAW[3] and LEXIS[4] information services. A WESTLAW user can, for example, ask to see all cases which have a certain key number headnote, or all cases which cited a given case. In addition, these systems support Boolean term searches. However, there are many limitations with systems like these (Blair and Maron, 1985). They are not well suited for the kinds of complex conceptual requests that a human researcher could handle, such as "find me all the cases about copyrights on computer programs, especially those related to *Apple v. Franklin*." Notions like "about" and "related to" cannot be defined exactly and so are are difficult for symbolic systems to handle. Should a case on video game copyrights be retrieved? Other cases involving Apple Computer? Cases cited by *Apple*? Cases citing it? It is this broader set of conceptual retrievals that SCALIR is designed to support.

While it is tempting to try to develop the necessary relationships by relying on the associative mechanisms of a connectionist network, this approach alone is unlikely to handle the entire task. For example, such a system would not easily support logical relations like "overturned-by," nor would it perform well in the real-world domain where literally tens of thousands of cases must be considered. As will be discussed below, these are some of the reasons for employing both symbolic as well as associative mechanisms in SCALIR.

2 Motivating a Hybrid Solution

There are some general reasons why one might want to use a hybrid solution for a given task. One might believe that hybrid approaches offer "the best of both worlds." However, this ignores the added costs that the complexity of a hybrid system incurs. (Some of these will be considered in Section 3.) It is important to consider specific aspects of the problem domain and of the system goals; together these factors may determine whether a hybrid is appropriate.

Since this paper is presented in a forum for connectionist research, I will not attempt to motivate the use of associative methods. Their success at problems involving learning and statistical inference is widely described in the literature. But there are many cases in which the problem can be addressed more effectively by combining symbolic methods with associative ones. Thus I will focus on cases in which the additional use of symbolic techniques is appropriate. Each of the following factors motivates the use of hybrids; in each case I will indicate how this factor is present in SCALIR's domain, legal information retrieval.

Symbolic characteristics of the problem. Some problem domains, such as symbolic integration, inherently involve the use of discrete representations. In the law, there are many such representations, such as the hierarchical taxonomy of the law formed by West Publishing's key numbers used to index cases. A symbolic component in a legal research system allows us to take advantage of the constraints provided by the existing representation.

Task involving artifacts. Humans solve many problems through the use of cognitive artifacts, artificial devices (such as an airline pilot's checklist) that enhance cognitive abilities (Norman, in press). If we want a computer system to interact with humans and assist them at existing tasks, that system should be able to use our artifacts. For example, we use a certain notation for writing music which normally supports the representation of twelve distinct tones. Even though the frequency of sound waves is not inherently quantized or "symbolic," a computer-based music composition system would do well to accept and present the conventional notation. Many artifacts are used by lawyers to solve the legal research problem. For example, *Shepard's Citations* (described in Section 1) lists not only what cases cited a case in question, but in what way -- e.g. criticized, questioned, etc. Furthermore, the labels drawn from Shepard's standardized set have taken on the status of "reserved words," conveying meanings beyond (and often different from) their natural-language counterparts.

A priori knowledge. One of the primary virtues of associative systems is their ability to learn by experience. However, if the system designer has a priori knowledge of the domain, this type of inductive learning may be inappropriate. First, the task may not be well-suited to inductive learning. For example, in his connectionist model of beam-balance physics, McClelland suggests that the highest level of understanding — using the concept of torque to predict which way the beam will fall — cannot be learned from examples and requires explicitly transferring knowledge (McClelland, 1989). Second, many learning

[2] Shepard's is a registered trademark of McGraw-Hill, Inc.

[3] WESTLAW is a registered trademark of West Publishing Co.

[4] LEXIS is a registered trademark of Mead Data Central.

algorithms are often designed to extract previously unknown regularities from their environment. Thus the representations they develop may not be comprehensible to their users (Belew and Forrest, 1988). Finally, if the relevant features are already known (as is the case with the Shepard's citations mentioned earlier), it makes little sense to train an associative representation when these relationships could be explicitly embedded in the architecture.

Efficiency. Associative systems have had difficulty scaling up to large problems. In some cases, there may be computationally efficient ways to symbolically solve certain sub-parts of the problem. Also, simply decomposing the problem may reduce the overall complexity. (This has also been a factor in the growing use of modular connectionist systems such as the Meta-Pi network (Hampshire and Waibel, 1990).) The information retrieval domain involves large databases; for example, SCALIR contains over 7500 nodes just to represent approximately 1500 documents — the Supreme Court and Federal Appeals Court cases for one small area of law. While the system's rules for propagating activity and for learning *could* be implemented by a pure associative system, it would require an order of magnitude more links, which would render it impractical for real-time use.

3 Problems with Hybrids

Hybrid systems face two problems not found (or found to a lesser extent) in their single-paradigm counterparts. First, the symbolic and associative components in a hybrid system must be able to communicate. Second, if learning is to be supported, credit must somehow be assigned to each component.

The communication problem arises because the two paradigms use different representations. In most connectionist implementations of the associative paradigm, the state of the system is contained in the real-valued activations of the units, while the long-term knowledge is embodied in the connection strengths, which can be expressed as a weight matrix. In contrast, state information in a symbolic system is contained in the positions and labels of a discrete set of tokens, such as the database of a production system or the markers in a semantic network. Long-term knowledge is represented as a set of static structures such as the production system rule base or the IS-A and other labeled links of the semantic net. Associative systems are generally good at statistical inference which involves the gradual accumulation of evidence for a solution, while symbolic systems excel at logical inference (though not necessarily that of first-order predicate calculus).

One solution to the communication problem is to subdivide the problem, delegating one set of tasks to each component. This approach might be appropriate for certain tasks, but it restricts the effectiveness of the hybrid, the components cannot benefit from each other's information when solving their individual tasks.

An alternative is to convert the representations of each component into a common language, or simply use one of the original representations for this purpose, reducing the amount of translation required. A variant of this is for each representation to operate on different aspects of the same medium. For example, Shastri's variable-binding network (Shastri and Ajjanagadde, 1990) uses phase to represent "symbolic" information and amplitude for associative activity. Similarly, SCALIR uses amplitude for activity and vector dimension (viewed conceptually as "color") for labels. This is explained in more detail in Section 4.2.

The credit assignment problem has haunted machine learning systems for many years: how to determine which part of the program should be rewarded (punished) when positive (negative) feedback is supplied. If a game-playing program loses, what parameters should be changed? Which ones were responsible for the loss? The problem can be largely solved by using a homogeneous representation and adjusting all "learning parameters" with a single mechanism. This is exactly the approach used in most connectionist systems; the learning parameters are link weights, and the single mechanism is a learning rule such as back-propagation. Thus each unit of the system can be modified in proportion to its contribution to the result.

However, hybrid systems are heterogeneous by definition. In addition to determining, for example, how much to change each weight in the connectionist network, one may need to know how much — or whether — that network played a role in that particular result. There are several possible approaches to this problem. The simplest is to restrict learning to only the associative system. A more powerful but more complex approach is to train the individual components separately, fix their parameters, and then train the combination together. This is similar to the technique used on some modular networks. Unfortunately, it does not work in real-time learning domains such as SCALIR's. Finally, the systems can attempt to learn simultaneously. Though there is a risk of instability, the system can be damped by making the learning parameters from both components 'compete" for the same resources As Section 4.3 explains, it is essentially this approach used by SCALIR to assign credit to its connectionist and semantic networks.

4 The SCALIR System

SCALIR, which stands for "Symbolic and Connectionist Approach to Legal Information Retrieval," is a hybrid system for assisting legal research about copyright law. When completed, SCALIR will contain all (approximately 4000) federal court decisions dealing with copyright law, in addition to the relevant statutes. As of this writing, about 1500 cases were actually installed in the system. The design of SCALIR was largely inspired by Belew's Adaptive Information Retrieval (AIR) system (Belew, 1986), though

AIR did not use a hybrid architecture.

Each case and statute section is represented by a node in SCALIR's network. In addition, there are nodes for terms which occur in the cases and statutes. The nodes are connected by various types of links — some with variable weights, unlabeled, and others with labels and constant weights. Thus the different link types form two interleaved networks, one connectionist and one similar to a semantic net. The network is unlayered and contains many recurrent connections. In most cases, a (directed) link from one node to the other is accompanied by a second link in the opposite direction, though their weights and/or labels are usually not symmetric. Nodes related in a structured, a priori manner (such as one case overturning another) get connected by "symbolic" links known as S-links. Nodes related by statistical inference (such as a case and a term that occurs relatively frequently in it) are connected by connectionist links called C-links. In addition, there is a third class of link which shares some properties with both of the others.

To retrieve documents, a user indicates possible terms or cases in which he or she is interested. For instance, Figure 1 shows a user interested in retrieving information about the terms "computer" and "software" and the case *Apple v. Franklin*.[5] The system then propagates activity through the network using a hybrid activation scheme described below. Nodes whose activity surpasses a certain significance threshold are presented as part of the response set. Since the weights on outgoing links sum to 1.0, and nodes' activities are "squashed" by a sigmoidal function, the total activity of the system gradually decreases. This process is hastened by halting activity flow from nodes which fall below a quiescence threshold. To accommodate the activity loss, the significance threshold decays to allow nodes related to the query by a set of longer, weaker chains of association to be considered relevant. The propagation algorithm is summarized in Figure 2.

A typical result is shown in Figure 3. At this point, the user can indicate (by selecting items with a mouse) how to pursue the search by *pruning* certain nodes or *expanding* on them. In addition to interpreting the feedback signal as a new query, SCALIR uses it as a reward or penalty for a reinforcement learning rule. Thus the network is gradually modified to agree with users' relevance judgements.

4.1 Building the Network

As the SCALIR network is constructed, initial weights and labels are assigned based on a statistical analysis of the documents and various properties of the domain.

All initial C-links connect case or statute nodes to term nodes. Each document is analyzed to determine which words occurring in it are most predictive of its content. This is a standard practice in information retrieval research (Salton, 1968); it also establishes the relative association strength or "term weight" from document i to term j. The following formula is used:

$$TW_{ij} = \frac{F_{ij} \times \log(N/DF_j)}{\sqrt{\sum_{k=1}^{N}(F_{ik} \times \log(N/DF_k))^2}}$$

where F_{ij} is the number of occurrences of term j in document i, DF_j is the number of documents in which term j occurs, and N is the total number of documents in the corpus.

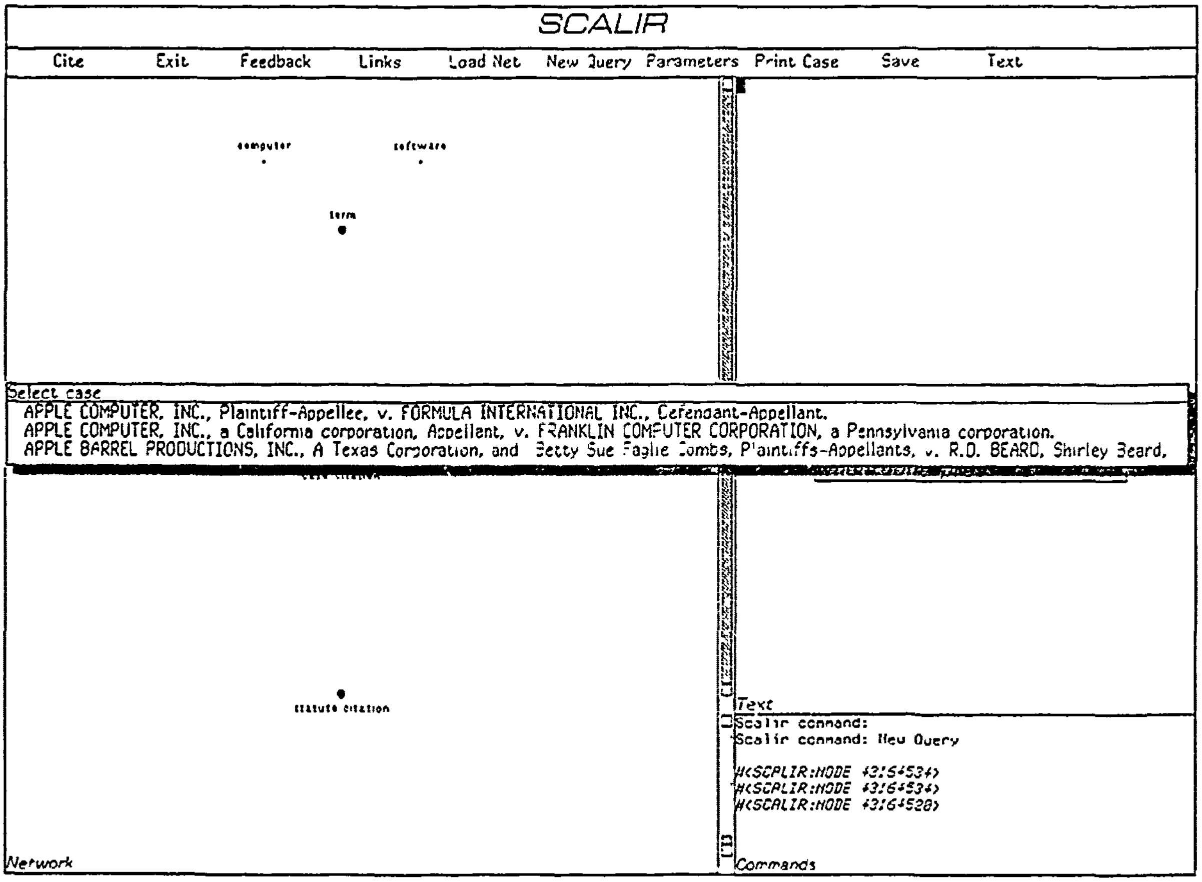
The most highly-weighted terms (usually ten to fifteen of them) are then chosen to be installed as term nodes in the network and linked to the documents that contain them. In order that SCALIR's propagation scheme (approximately) conserve the initial amount of activity introduced by a query, all outgoing weights from a node must sum to unity; the term weights are thus normalized appropriately to become C-link weights. The inverse (term-to-document) links are also installed, weighted inversely proportional to the number of document nodes to which they are connected. The C-link weights can later be modified through the learning rule by user feedback.

S-links represent fixed, *a priori* knowledge of relationships or constraints in the domain. For example, each case is indexed by West Publishing Co.'s "key numbers" (described in Section 1) indicating what areas of law the case touches upon. Since these keys form a hierarchical taxonomy of the law, S-links can be constructed to reproduce this hierarchy in a manner similar to IS-A links in a traditional semantic net. S-links are also used to indicate indisputable relationships between cases, such as that case A overturned case B on appeal. Finally, S-links are used to indicate dependencies in statutes, as shown in Figure 4.

Some relationships can be labeled explicitly and reasoned about logically, but are not indisputable facts. An example of this is a citation from one case to another. Did the judge in the citing case criticize, limit, or question the holding of the cited case? These decisions are made by the editors of *Shepard's Citations*, also described in Section 1. *Shepard's* identifies each citation as falling into a small, fixed set of categories. The decision about which category label to use is a subjective one, made by an editor with limited time and resources to read the citing case and describe the judge's intentions.

For these labeled but subjective relationships, SCALIR uses a third type of link, called an H-link (for "hybrid"). Like S-links, H-links bear labels and can be the subject of logical inference. But like C-links, they have modifiable weights. Furthermore, similar Shepard's "treatment phrases" (the categories mentioned above) are represented as links with the same label, but different weights. For example, both "harmonized" and "followed" citations are expressing agreement on the legal issue or rule, though the latter is stronger than the former. Thus SCALIR represents them as two instances of an H-link labeled "similar

[5] Recall that the entire database contains cases on copyright law, so the user does not need to specify the term "copyright."



Select case

APPLE COMPUTER, INC., Plaintiff-Appellee, v. FORMULA INTERNATIONAL INC., Cefendant-Appellant.
APPLE COMPUTER, INC., a California corporation, Appellant, v. FRANKLIN COMPUTER CORPORATION, a Pennsylvania corporation.
APPLE BARREL PRODUCTIONS, INC., A Texas Corporation, and Betty Sue Faglie Combs, Plaintiffs-Appellants, v. R.D. BEARD, Shirley Beard,

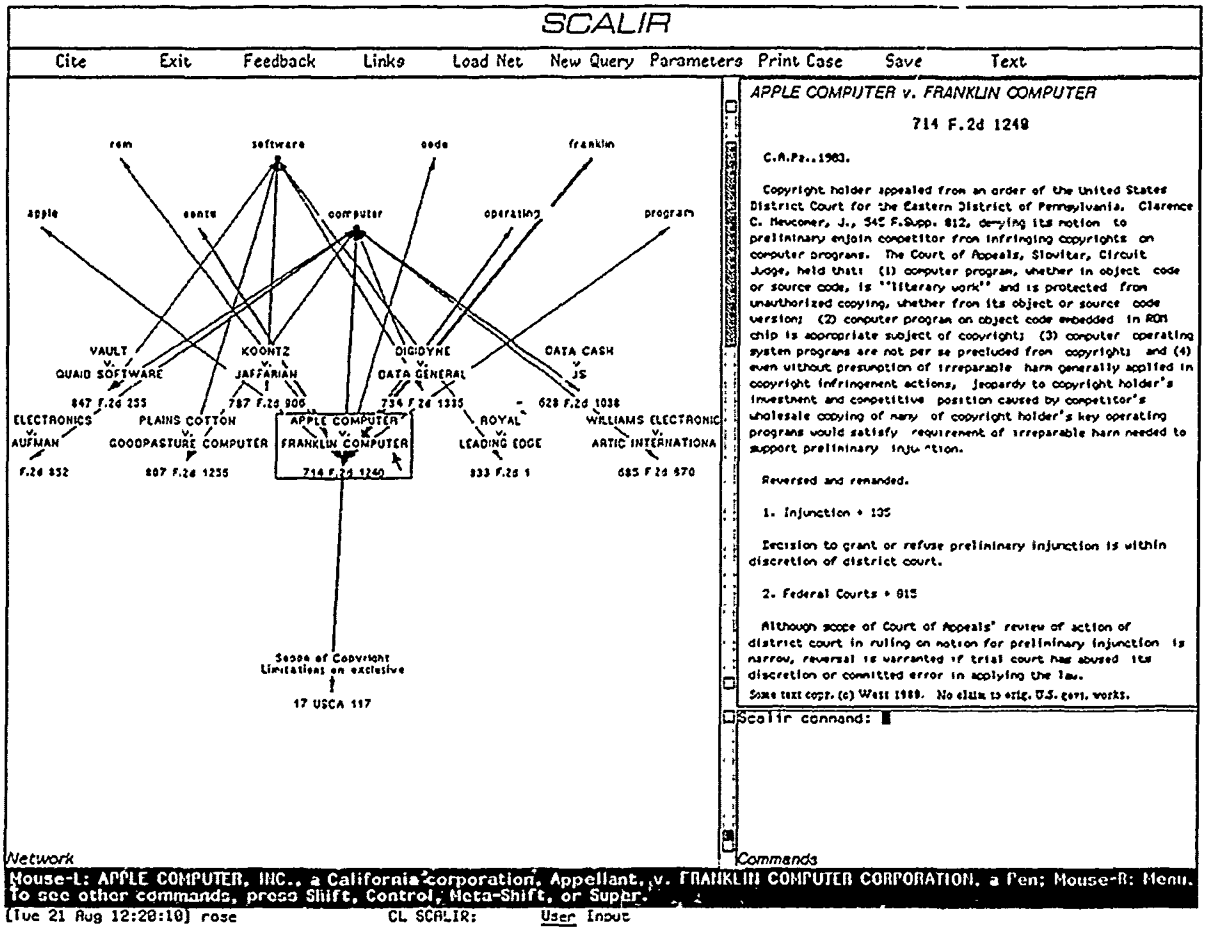
```
Scalir command:
Scalir command: New Query

#<SCALIR:NODE #3:54534>
#<SCALIR:NODE #3:64534>
#<SCALIR:NODE #3:64520>
```

Figure 1: Querying the system.

```
Set significance-threshold to initial value.
Set ACTIVE-NODES to the set of nodes queried by the user.
Repeat {
        Set RESPONSE-SET to all nodes in ACTIVE-NODES with activity above significance-threshold.
        Remove from ACTIVE-NODES all nodes with activity below quiescence-threshold.
        Add neighbors of nodes in ACTIVE-NODES to ACTIVE-NODES.
        Update activities of all nodes in ACTIVE-NODES.
        Decay significance-threshold.
} Until ACTIVE-NODES is empty or RESPONSE-SET reaches maximum size.
```

Figure 2: SCALIR's retrieval algorithm. Lower-case labels are real-valued propagation parameters, upper-case labels are sets of nodes.



APPLE COMPUTER v. FRANKLIN COMPUTER

714 F.2d 1240

C.A.Pa..1983.

Copyright holder appealed from an order of the United States District Court for the Eastern District of Pennsylvania. Clarence C. Newcomer, J., 545 F.Supp. 812, denying its motion to preliminary enjoin competitor from infringing copyrights on computer programs. The Court of Appeals, Sloviter, Circuit Judge, held that: (1) computer program, whether in object code or source code, is "literary work" and is protected from unauthorized copying, whether from its object or source code version; (2) computer program on object code embedded in ROM chip is appropriate subject of copyright; (3) computer operating system programs are not per se precluded from copyright; and (4) even without presumption of irreparable harm generally applied in copyright infringement actions, jeopardy to copyright holder's investment and competitive position caused by competitor's wholesale copying of many of copyright holder's key operating programs would satisfy requirement of irreparable harm needed to support preliminary inju^ction.

Reversed and remanded.

1. Injunction • 135

Decision to grant or refuse preliminary injunction is within discretion of district court.

2. Federal Courts • 815

Although scope of Court of Appeals' review of action of district court in ruling on motion for preliminary injunction is narrow, reversal is warranted if trial court has abused its discretion or committed error in applying the law.

Some text copr. (c) West 1989. No claim to orig. U.S. govt. works.

```
Scalir command: █
```

Mouse-L: APPLE COMPUTER, INC., a California corporation, Appellant, v. FRANKLIN COMPUTER CORPORATION, a Pen; Mouse-R: Menu.
To see other commands, press Shift, Control, Meta-Shift, or Super.

```
[Tue 21 Aug 12:28:10] rose          CL SCALIR:        User Input
```

Figure 3: Result of processing the *Apple* query.

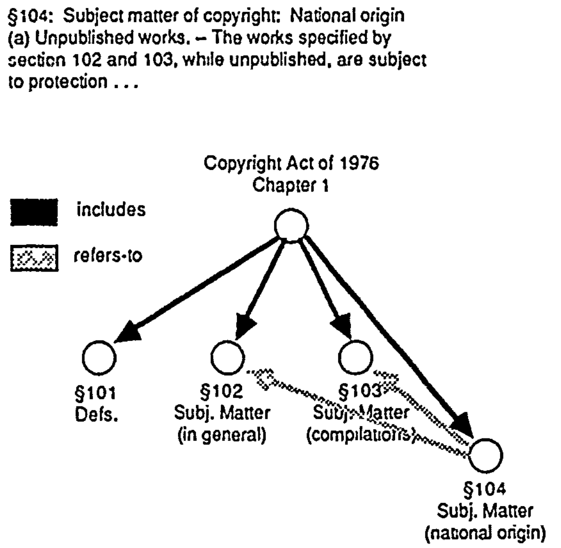

§104: Subject matter of copyright: National origin
(a) Unpublished works. – The works specified by section 102 and 103, while unpublished, are subject to protection . . .
Copyright Act of 1976
Chapter 1
includes
refers-to
§101
Defs.
§102
Subj. Matter
(in general)
§103
Subj. Matter
(compilations)
§104
Subj. Matter
(national origin)

Figure 4: Some dependencies in the Copyright Act of 1976.

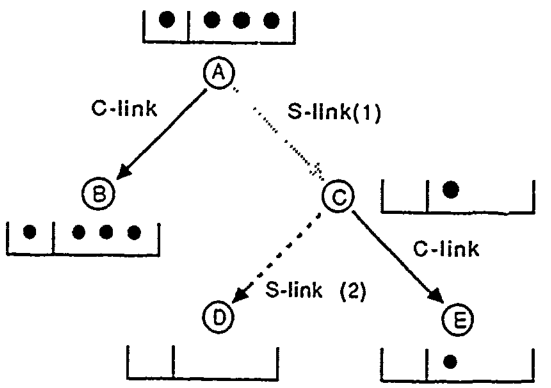


Figure 5: Hybrid activity propagation

rule." In contrast, the citation of a dissenting opinion is orthogonal, and has its own H-link label.

The three classes of links are summarized in Table 1.

4.2 Hybrid Activity Propagation

A typical connectionist network employs spreading activation, the activity of a node is the result of repeatedly computing a real-valued function of the activity of its neighbors and the weights joining them. Once a node's activity is computed, it does not matter (for the purposes of determining the next state or the output of the system) which of the infinite possible combinations of weights and neighbors' activations was actually present.[6]

In contrast, the marker-passing in a semantic network — though also generally termed "spreading activation" — has quite a different character. There may be several different types of markers and link labels, and the decision of a node to pass or not pass a marker may be a complex logical function of these labels. Some semantic nets also contain provisions to prevent the unbounded spread of activity; the constant weights on SCALIR's S-links play a similar role.

Since SCALIR contains both connectionist and symbolic[7] links, it also must support both types of spreading activation. Furthermore, since the networks are fully interleaved (i.e., the different links share the same nodes), the system must provide a mechanism for communication between them. This is accomplished by conceptually dividing up activity into a *vector of several components*. Each component corresponds to activity filtered by a certain type of S-link, except for one component reserved for the "miscellaneous" activity carried by C-links.

When activity traverses a C-link, all components of the activity vector are individually multiplied by the weight on the link. When activity traverses an S-link, only the component which matches the link type is allowed to pass; the others are set to zero. We can think of the different components of activity as different wavelengths (colors) of light. C-links serve as grey filters, modulating the intensity of all wavelengths. S-links are like colored filters, allowing only light of the right wavelength to pass.

An example of the two kinds of propagation is shown in Figure 5. The large circles in the figure represent nodes in the SCALIR network; the different arrows represent different kinds of links. C-links are shown as solid arrows, S-links (of various types) as arrows with various shadings. The rows of dots represent the activity components for each node; the single dot on the left hand side corresponds to "miscellaneous" activity, while the other dots are specific types of activity. Suppose the node A at the top of the figure is activated with all colors. Since the link to node B is a C-link, it passes all of A's colors of activation to B, attenuated by the weight on the link. In contrast, the link to node C is an S-link of type 1; so only the corresponding component of activity gets passed along. Finally, note that node D gets *no* activity, since the type of the link from C to D does not match the type of activity present at C.

This approach has two immediate benefits. First, and perhaps most obviously, it supports both associative (connectionist) retrievals and logical (symbolic) retrievals. Sec-

[6] We have referred to this property elsewhere (Rose and Belew, 1989) as *interchangeability*.

[7] Since H-links are labeled and support logical inference, they will be treated as a special case of S-links in the discussion of activity propagation.

LINK TYPE	INFERENCE	LEARNING	INSTANCES
C-link	no	yes	term ↔ case (word occurrence) term ↔ statute (word occurrence) any nodes (from feedback)
H-link	yes	yes	case ↔ case (Shepard's treatment phrases)
S-link	yes	no	statute ↔ statute (structure) statute ↔ statute (cross-reference) term ↔ term (West key number taxonomy) case ↔ case (Shepard's history)

Table 1: SCALIR's three classes of links.

ond, the logical operations are completely local to individual nodes. For example, suppose one is interested in only cases which supported a certain decision. The user could draw a "supported-by" link to a node representing some unspecified case. The first node would thus be instructed to pass activity only along the corresponding vector. After activity traversed the first S-link, however, no other nodes would need this instruction. Due to the filtering of the S-links, only the desired component of activity would be present at successive nodes. It is as if all other S-links were instantaneously disabled. This feature makes the system especially amenable to future implementation on parallel hardware.

4.3 The Hybrid Learning Rule(s)

Typical connectionist systems use either supervised, unsupervised, or reinforcement learning. However, as Belew notes in his discussion of AIR's learning rule (Belew, 1987), these three approaches do not exhaust the possibilities. AIR used what Belew calls a "localized reinforcement" [illegible] , in which the reward or penalty is a vector, rather than a scalar value. In the information retrieval domain shared by AIR and SCALIR, relevance judgements are made by actual users, whose feedback is then used to train the weights.

Thus SCALIR uses the localized reinforcement ideas from AIR, but implements them with weight adjustments similar to the type used in error-correction rules such as backpropagation. In a simple error-correction rule, the new weight w' is computed from the old weight (from node j to node i) as follows:

$$w'_{ij} = w_{ij} + \eta(t_i - a_i)a_j$$

a_i and a_j are the activities of nodes i and j, respectively, and t_i is the "target" — the correct value that node i should have produced. η is a learning rate, which is generally kept small in order to avoid "overshooting" the correct weights. In SCALIR, user feedback is used in lieu of a target. In other words, when a user says "I liked that case, give me more like it," the system treats this as "that node should have been maximally activated." When a user says "I didn't like that one, don't show it to me," the system interprets this as "that node should not have been activated." Early experiments suggest that scaling the weight adjustment by an amount proportional to the activity a_j of the node receiving feedback (as might be suggested by error-correction rules) does not improve performance. Accordingly, a simplified version of SCALIR's learning rule is:

$$w'_{ij} = w_{ij} + \eta f_i a_j$$

where f_i is the (binary) feedback given to node i by the user. We are experimenting with various values of η, on the order of 0.1.

For nodes that do not explicitly receive user feedback, the f values are computed by propagating the user's feedback signals "back" through the network. "Back" simply means the link directions are reversed; other than that, feedback propagation uses the same algorithm as activity propagation (shown earlier in Figure 2). As with activity propagation, the fact that the total activity never increases (since outgoing weights sum to 1.0), and is attenuated by both the squashing function and the quiescence threshold, guarantees that the system will not get stuck in an infinite loop. Restricting the set of active nodes (those which get to participate in propagation) to just those with "significant" levels of activity further speeds up the process of moving to a stable state.

There are two final considerations for updating weights. First, recall that in order to conserve activity, SCALIR requires that all outgoing weights from a node must sum to 1. This means that all weights from a node must be renormalized whenever any one of them is changed. This also means that C-links and H-links must compete for a share of the total weight, thus addressing the heterogeneous aspect of the credit-assignment problem.

Second, since SCALIR contains several types of links, and nodes have vector-valued activity, we need different learning rules to handle the different classes of links. S-links have fixed weights, so they actually do not learn at all. C-links pass all components of activity, so we can replace the scalar-valued activities a_j and feedback f_i with vectors a_j and f_i to obtain the following rule:

$$w'_{ij} = w_{ij} + \eta \mathrm{f}_i \mathrm{a}_j$$

H-links, in contrast, each pass only one type of activity. Thus they are changed in the following way:

$$w'_{ij} = w_{ij} + \eta f_i^{\tau}(a_j^{\tau} + a_j^{\mu})$$

where the superscript μ indicates the "miscellaneous," or unfiltered component of activity, while τ is the "typed" component passed by that link. In words, this rule says to change the H-link weight in proportion to "postsynaptic" feedback of the correct type and any "presynaptic" activity that can traverse the link (namely, miscellaneous activity and activity matching the link type).

Although there is a superficial similarity between SCALIR's learning algorithm and back-propagation (BP), it lies only in the idea of "reversing" the propagation of activity. The differences are more striking. BP works for layered networks with scalar-valued activity and a single link type. BP also approximates gradient descent to any desired degree, and generally finds error minima, though possibly local. SCALIR's learning procedure works for unlayered, (potentially) fully recurrent networks with vector-valued activity and a multiplicity of link types, some symbolic and some connectionist. Furthermore, since relevance judgements cannot be made for all possible future queries, and the set of "outputs" changes according to the query order, we have no way to measure a "correct answer," and thus it is unknown whether a global error is being optimized. SCALIR's learning rule is a heuristic which, based on early tests and previous experience with AIR, seems to produce the desired behavior.

5 Discussion and Conclusions

We have seen that there are many factors which may motivate the use of a hybrid system, and that the difficulties introduced by hybridization can be overcome. SCALIR, though just one example, illustrates both of these points. Yet many observers familiar with the recent successes of connectionism may feel that the presence of any symbolic mechanism is unnecessary.

Many connectionist systems have been designed to "emulate" symbolic behavior. These often serve as existence proofs of the capabilities of connectionist models, and argue for the plausibility of these models in various domains. In a hybrid system, then, it *is* often possible to replace the symbolic component with an associative one trained to reproduce the symbolic behavior. Some may view this as an inherently desirable substitution, since the resulting system is more uniform, easier to analyze, and — by some measures — simpler. The challenge to the hybrid system designer, then, is to determine if and when this substitution should be carried out.

If the designer of the system is engaged in engineering — that is, an attempt to design the best solution for a particular problem — then arguments based on "purity" or "plausibility" carry no weight. If the most effective system is a hybrid (because certain computations can be performed more efficiently, for example), then there is no reason to try to replace the symbolic mechanisms with associative ones.

However, suppose the designer of the system is engaged in science, attempting to investigate (but not necessarily model) certain phenomena. Simplicity of explanation may indeed be a consideration, but which approach — hybrid or single-paradigm — provides this simplicity depends on the phenomenon being studied. The hybrid may still be appropriate if some of the other conditions from Section 2 obtain. There may for example be explicit constraints on the problem which must be represented; embedding these in an underlying associative representation may confound the analysis of the system.

Finally, one may be interested in a specific modelling task, in which the components of the connectionist system are meant to stand for components or groups of components in the actual system exhibiting emergent behavior and possessing no explicit symbol-manipulation capabilities. (While the elements of the system being modeled are often neural, they need not be.) We would not be satisfied with a model that had an omniscient rule-based control mechanism intervening in the network's behavior. Under these circumstances, the success of the model can be established only by discarding the hybrid solution in favor of a purely connectionist approach.

The ultimate question for hybrid systems is whether the whole will be greater than the sum of its parts. As we have seen, the answer is both no and yes. In one sense, any sufficiently powerful representational system can emulate the computations of another; associative systems can be trained to duplicate the results of a hybrid. But as SCALIR demonstrates, the different strengths of the two paradigms *can* be combined effectively, resulting in a system with behavioral properties different from and better than a system constructed using only one approach.

Acknowledgements

I would like to thank Rik Belew, Susan Gruber, and Dave Touretzky for their helpful comments on earlier drafts of this paper. I would also like to thank West Publishing Co. and Shepard's/McGraw-Hill for making their data available for use in SCALIR.

References

Belew, R. K. (1986). *Adaptive Information Retrieval. Machine Learning in Associative Networks*. PhD thesis, University of Michigan.

Belew, R. K. (1987). Designing appropriate learning rules for connectionist sytems. In *Proceedings of the IEEE First International Conference on Neural Networks*, San Diego, CA.

Belew, R. K. and Forrest, S. (1988). Learning and programming in the classifier system. *Machine Learning*, 3(2).

Blair, D. C. and Maron, M. E. (1985). An evaluation of retrieval effectiveness for a full-text document-retrieval system. *CACM*, 28(3):289–299.

Davis, R., Buchanan, B., and Shortliffe, E. (1977). Production rules as a representation for a knowledge based consultation program. *Artificial Intelligence*, 8:15–45.

Hampshire, J. B. and Waibel, A. (1990). Connectionist architectures for multi-speaker phoneme recognition. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems* 2, pages 203–210. Morgan Kaufmann, San Mateo, CA.

Hendler, J. A. (1989). Marker-passing over microfeatures. *Cognitive Science*, 13(1):79–106.

Lange, T. E. and Dyer, M. G. (1989). Frame selection in a connectionist model of high-level inferencing. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, Ann Arbor, MI.

McClelland, J. L. (1989). Parallel distributed processing: Implications for cognition and development. In Morris, R. G. M., editor, *Parallel Distributed Processing: Implications for Psychology and Neurobiology*. Clarendon Press, Oxford.

Norman, D. A. Cognitive artifacts. In Carroll, J. M., editor, *Designing Interaction: Psychology at the Human-Computer Interface*. Cambridge University Press, New York. In press.

Rose, D. E. and Belew, R. K. A connectionist and symbolic hybrid for improving legal research. *International Journal of Man-Machine Studies*. (To appear).

Rose, D. E. and Belew, R. K. (1989). A case for symbolic/sub-symbolic hybrids. In *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society*, pages 844–851, Ann Arbor, MI.

Salton, G. (1968). *Automatic Information Organization and Retrieval*. McGraw-Hill, Inc., New York, NY.

Shastri, L. and Ajjanagadde, V. (1990). From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings. Technical Report MS-CIS-90-05, Computer and Information Science Department, University of Pennsylvania.

Cognitive Map Construction and Use: A Parallel Distributed Processing Approach

Ronald L. Chrisley*
New College
University of Oxford
Oxford OX1 3BN
United Kingdom

Abstract

The Connectionist Navigational Map (CNM) is a parallel distributed processing architecture for the learning and use of robot spatial maps. It is shown here how a robot can, using a recurrent network (the CNM *predictive map*), learn a model of its environment that allows it to predict what sensations it would have if it were to move in a particular way. It is shown how this predictive ability can be used (via the CNM *orienting system*) to enable the robot to determine its current location. This ability, in turn, can be used, when given a desired sensation, to generate sequences of goal states that provide a route to a place with the desired sensory properties. This sequence is given to the CNM's *inverse model*, which in turn generates a sequence of actions that effects the desired state transitions, thus providing a sort of "content-addressable" planning capability. Finally, the theoretical motivation behind this work is discussed.

1 INTRODUCTION

The Connectionist Navigational Map (CNM) is a system being developed to impart to an autonomous robot the ability to map its spatial environment and use this map to navigate in that environment. The primary theoretical motivation for constructing the system is to understand how a robot can make the the transition from pre-conceptual to conceptual representations of space.[1]

The CNM is being designed with a Heathkit Hero 2000 robot as the intended testbed. The features of the Hero 2000 that are relevant to this paper are the fact that it is autonomous and mobile, its primary sense is sonar range-finding, and its spatial environment is a floor of research offices in a modern building. It is this experimental situation that is implicit in the following discussion and illustrating examples. For instance, the notion of "sensations" in the case of Hero can be taken to be a vector of n numbers, the first indicating, roughly, the distance to the closest surface directly ahead, the next number indicating the distance to the closest surface at a heading of $\frac{2\pi}{n}$ radians to the right of front, etc. Also, it is important to the proper functioning of the CNM that the sensations to which the robot attends are static, i.e., the sensation vector for a given location in the world is constant.

There are three main components of the CNM system.

- the forward model, or *predictive map*;
- the ability to determine to which place on the map a particular sensation might correspond, or *orienting system*[2];
- the ability to generate the actions that will reach a desired state, which is provided by the *inverse model*.

Each of these components will be discussed in turn.

2 THE PREDICTIVE MAP

One can view the process of robot map construction as the learning of a model of the environment, such that once the model is learned, the robot can use it to predict what sensations would occur if it moved in a

*Also affiliated with Xerox PARC Systems Sciences Lab, 3333 Coyote Hill Road, Palo Alto, CA 94304. Email. chrisley@csli.stanford.edu

[1] See section 6.

[2] The usage of "orienting system" here should not be confused with the usage of "orienting subsystem" in Carpenter and Grossberg's various papers on Adaptive Resonance Theory, e.g. (Carpenter and Grossberg '87). "Orienting" is used here in a way similar to the way one does when talking of orienting oneself using a field map and compass, or the north star.

particular ego-centrically specified manner (e.g "rotate $\frac{\pi}{4}$ radians to the right", "move forward 10 feet").[3]

Of course, this will require the agent to have some kind of representation of its current location, since, in general, the mapping from actions to sensations is dependent on where one is in the world. That is, the mapping *sensations* × *actions* ↦ *sensations* is one-to-many, since more than one place can have the same sensory properties. Thus, the spatial environment, and therefore a model of it, can be seen instead as a function from current state and current action to predicted sensations. The input consists of a state vector, corresponding to the current location l of the robot, and a vector representing the move m being made. The output of the network is a vector that is supposed to be equal to the sensation vector the robot would receive from its senses if it were actually at the place that is reached by making the move m at location l.

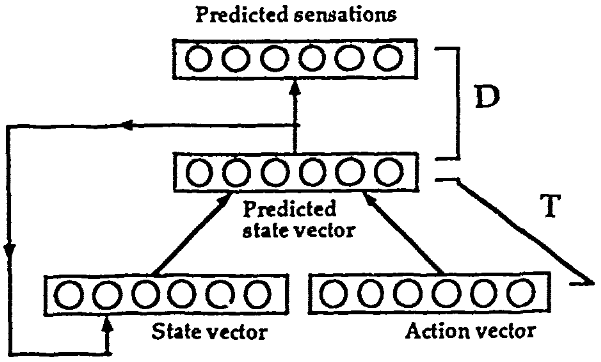


Figure 1: A PDP architecture for learning a predictive mapping (*states* × *actions* ↦ *sensations*) by composing a topological mapping T (*states* × *actions* ↦ *states*) with a descriptive mapping D (*states* ↦ *sensations*).

2.1 TOPOLOGICAL AND DESCRIPTIVE MAPPINGS

There is more structure to space than the mapping *states* × *actions* ↦ *sensations* indicates. Specifically, location and action determine a new location, which itself determines the sensations of the robot. Thus, it might be easier to learn a model if its structure reflects this regularity of the spatial environment. Consider the PDP architecture depicted in figure 1.

As said before, the input consists of a state vector and an action vector, the output is a vector that is supposed to be equal to the sensation vector the robot would receive from its senses if it were at the place that is reached by making the move at the location represented by the state vector. The network is a composition of two mappings. a T mapping and a D mapping (explained below). The recurrence between the hidden layer[4] and the state vector portion of the input layer allows this network to be a viable architecture for a predictive map. The activity pattern on the hidden units at time t is sent back to become the state representation in the input at time $t+1$.[5] This imposes a relation on the patterns representing states, so that they become meaningful and useful. For example, if the robot considers moving forward and then turning right, it can use the map to predict what sensations it would have after those moves by calculating $D(T(T(l, \text{forward}), \text{right}))$, where l is a state vector corresponding to the robot's location before the moves, and D and T are the mappings indicated in figure 1. It is the recurrent connection in the network that allows this composition of the T mapping.

The mapping also becomes learnable, since all necessary inputs and outputs are provided or can be calculated.

Learning a model of the environment, then, can be decomposed into learning two mappings: a *topological* mapping T from states[6] and actions to states, and a *descriptive* mapping D from states to sensations.

[3] It should be made clear at the outset what is meant by a "map". "Map" is not taken to imply any kind of visual or imagistic reoresentation. Rather, a map is any data structure that associates representations with actual locations, and with other representations that indicate the properties that that represented location has. Furthermore, the location representations are related to each other in a way that indicates what kind of motion would be required to get from one to the other. The point is that there isn't much weight being put on the word "map" itself, and if one prefers, one can just mentally replace it with "descriptive/topological data structure" The reason why this kind of data structure should be of any interest is brought out in section 2.1.

[4] In all discussions of network architectures in this paper, I suppress any mention of a hidden layer unless the activation patterns of that layer are used somewhere else in the network. For example, it is consistent with figure 1 (and is presumed by my discussion) that there may be a hidden layer between the input and output layers of the descriptive mapping, D.

[5] Such a use of recurrence has been suggested by (Jordan '86) and (Elman '88). A notable difference is that Jordan's network has recurrent links from the outputs, while Elman's network has recurrent links from the hidden units, as does the architecture in figure 1.

[6] I will hereafter use "state" instead of "location" not only because the network will have to represent orientations as well as locations, but because I want to allow activity patterns in the network to be interpreted in ways that do not involve the objective, absolute places that the word "location" connotes.

2.2 *A PRIORI* AND EMPIRICAL TOPOLOGIES

There are two approaches to having a mapping network learn a model of the environment. In one approach, the topological mapping is assumed to be known *a priori*, and the network only has to learn the descriptive mapping. The structure of space is known; all that remains to construct the map is to "fill it in."

In the second approach, no such *a priori* knowledge is assumed; the network must learn both mappings. Although there are many reasons why such a "tabula rasa" strategy would be more appropriate for the purposes of psychological modeling and philosophical understanding, one should not underestimate the advantages of an *a priori* topology for the purposes of engineering a working navigational system. The "empirical topology" that would result from learning both mappings would most likely fail to guarantee such essential properties as idempotency under the null action, invertibility, and composability, in the mappings it produces. For example, it would be possible for a network to learn a topology where the following are true:

$$\begin{aligned} &T(x,\text{null}) \neq x;\\ &T(x,a) = y \text{ but } T(y,a^{-1}) \neq x;\\ &T(x,a) = y \text{ and } T(y,b) = z,\\ &\qquad\qquad \text{but } T(x, b \circ a) \neq z; \end{aligned}$$

where a^{-1} is the inverse of an action (moving back 2 feet as opposed to moving forward 2 feet), and $b \circ a$ is an action that is equivalent to performing a, then b.

Thus, PDP learning methods might produce topologies that can play an important role in explaining navigation behavior in animals, and in understanding concept acquisition,[7] but it is hard to see how they could produce a topological mapping that would be as accurate and general as a hand-crafted topology that had these three desirable properties, *inter alia*, built-in. This is not to say that such topologies could not be learned; but the development of such general topologies from experience alone, while theoretically of great import, and even potentially superior in terms of sheer performance, is not a prerequisite for a working system.

2.3 LEARNING THE TOPOLOGICAL AND DESCRIPTIVE MAPPINGS

The best strategy for learning the predictive map will depend on which of the two kinds of topologies, *a priori* and empirical, is being used. All the learning strategies considered in this paper assume the kind of learning situations available to back-propagation of error in multi-layer, feed-forward networks. Specifically, in learning a mapping $F : X \mapsto Y$, the network is presented with a number of samples, (x_i, y_i), where $x_i \in X$, $y_i \in Y$, and $F(x_i) = y_i$. This leaves much of the learning process unspecified (e.g., whether the samples are presented simultaneously, allowing epochal learning, or serially, requiring learning on a particular sample before the next sample can be examined), thus allowing for several distinct possible learning strategies. However, before discussing these different strategies, an observation needs to be made about the holistic nature of maps.

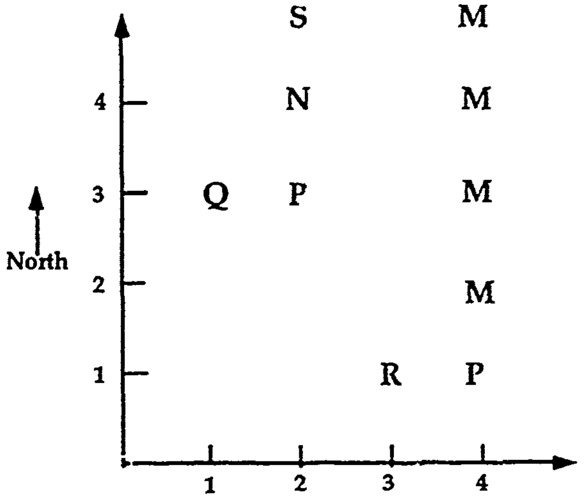


Figure 2. A schematic spatial environment used to illustrate the holistic nature of the *activity pattern* ↦ *world state* referential relation. Capital letters indicate the (idealized) sonar sensation at that location.

2.3.1 Maps are holistic

The place that a particular symbol on a conventional road map represents is not an intrinsic property of the symbol, but is determined by where the symbol is in some reference frame: how the symbol is spatially related to other symbols. Similarly, which world state a particular CNM activity pattern represents is not determined merely by the intrinsic properties of that pattern, or even by that pattern and the sensory properties assigned to it under the descriptive mapping; rather, a pattern's referent is also determined by its connections, under the topological mapping, to other activity patterns and their sensory associations given by the descriptive mapping.

For example, consider the spatial environment depicted in figure 2.

Note that in this toy world there are two qualitatively identical places, (2,3) and (4,1), each with the idealized sonar sensation P. Now suppose that there happen to be two activity patterns in the CNM, x_{10} and x_2, that, when input to the descriptive mapping, yield

[7] An elaboration of such philosophical benefits can be found in section 6.

P. That is:

$$D(x_{10}) = D(x_2) = P.$$

It is impossible to tell from this information alone which of the two places each pattern represents. Thus it would not be possible to determine what input pattern to give to the network in order for it to learn a mapping involving the place (2,3).

But suppose that the topological mapping had the following properties:

$$T(x_{10}, \alpha) = x_3;$$
$$D(x_3) = Q;$$
$$T(x_2, \alpha) = x_4;$$
$$D(x_4) = R;$$

where α is an action that results in the robot moving to the west.

In this case, there would be an indication that x_{10} represents (2,3) and x_2 represents (4,1): the topological relations between patterns as defined by T helps determine the referents of the patterns themselves. But note that the four properties above do not conclusively determine the reference of x_{10} and x_2: the relation of x_{10} and x_2 to other states in the CNM might indicate just the opposite, with the above mappings being a local error. For example, we might have, for the north-bound action β:

$$T(x_i, \beta) = x_{i+1} \text{ for } 10 \leq i \leq 14;$$
$$D(x_i) = M \text{ for } 10 \leq i \leq 14;$$
$$T(x_2, \beta) = x_1;$$
$$D(x_1) = N;$$

which would indicate just the opposite: that x_2 represents (2,3) and x_{10} represents (4,1); the evidence for this conclusion would be even stronger if Q and R were very similar. Thus, not only do the referential properties of states depend on *both* the topological and descriptive mappings, but reference also depends on the *entire* mappings.

2.3.2 Empirical topologies: learning both mappings at once

As mentioned above, in the standard approach to getting a PDP network to learn a mapping, a database of input/output pairs for which the mapping holds is required. For the CNM predictive map, this would require a set of pairs where each pair comprises a state vector and an action vector as input, and a sensation vector as the desired output.

However, the spatial world does not provide an agent with explicit information as to which state it is in. A mapping network which does not assume an *a priori* correspondence between states of the network and states of the world will have to generate its own system of representation of world states. Thus, the point just made about the holism of maps is relevant in choosing a learning strategy. Since, as we saw, the representational relationship between an activity pattern and a world state depends on both the topological and the descriptive mappings, the pattern that corresponds to a particular world state will change as the two mappings in the network change. This means that the input/output mapping to be learned will be dynamic. The network's learning task will be a "moving target" situation[8]: the desired output for a given input will change as the weights of the network, and hence the mappings they realize, evolve.

Several standard approaches to collecting the input/output pairs for the learning process would be of little use then. For example one could not just select the samples on the basis of the world states one wants the network to learn about. It might be easy enough to get the desired output, but what should the state vector input be? One would have to come up with some way of determining, given a network and its current weights, what pattern represents a particular location. We have already seen that this procedure would be dependent on the state of the entire network, and that the topological evidence can be contradictory. It is very likely, then that the computational cost of the procedure would be prohibitively high. Furthermore, it would require some teacher to actually choose and collect this data.

An alternative strategy is to let the robot generate the samples itself by moving through the environment. That is, make the samples causally and temporally related to their predecessors and successors. The robot starts with an arbitrary state vector and makes some move. The desired output is the sensations it receives at the location reached by the move. Thus, the first training sample is created. Another action is chosen, and it, along with state vector on the hidden units, would become the input for the next sample, etc.

Note that this is not merely a matter of incorporating context. It is common, for example, in training networks for speech recognition or text-to-speech tasks to present to the network the samples as they are encountered in the environment, as this facilitates incorporating context effects. The network gets its input from a window on the input stream, and the window is moved down the input stream to provide another input. The actual order of the windowed input is irrelevant, however. the window could be placed anywhere on the stream to generate a useful input sample. But this *assumes* an input stream that exists independently of the network's states, whereas the strategy of using environment-determined inputs for the CNM is in order to generate such a stream. Nor is the importance of ordering related to superficially similar schemes that

[8] Both (Mikkulainen and Dyer '88) and (Chalmers '90) explicitly mention the "moving target" nature of their networks' learning situation. In both cases, the networks were successful in developing an appropriate representational system.

improve learning times, accuracy, or convergence.

This iterative means of actually generating the training samples does not mean that learning has to be serial. One could, after collecting enough data, use an epochal learning strategy, in which all weight changes from each sample are calculated before any are made. But this would necessitate either a human assistant or prior navigation routines in order to generate actions for the robot while the data is being collected, since the CNM would not be able to be used for navigation until after the data for the epochal learning had been collected and some learning had taken place.

In a non-epochal system, the weights are changed after each sample presentation. (Jordan '90) has shown that it is possible in such cases to learn the inverse model at the same time as the forward model. In the case of CNM, this means that navigational abilities should be able to be learned at the same time as the predictive map. If so, the robot's navigational system could choose the actions to be taken, even at the very first stages of learning, as Jordan's systems do. Thus, no human teacher would be required; the system would be capable of generating all of its inputs and desired outputs.[9]

2.3.3 Learning with *a priori* topologies

There are two kinds of *a priori* topology: fixed and variable. In a fixed topology, the weights that implement the topological mapping are frozen[10], with learning in the CNM only changing the weights that implement the descriptive mapping; as said before, all that remains to construct the map is to "fill it in". A variable *a priori* topology, on the other hand, initializes the topological mapping to some ideal state, but allows subsequent modification and refinement on the basis of experience.

To learn a predictive map with a fixed *a priori* topology, one would initialize the robot's state vector to the one that corresponds, in the *a priori* topology, to the initial location of the robot. Then, the robot would chose a move to make. The topology is again consulted to yield the state vector that corresponds to the location that robot would occupy if it were to make that move. This vector is used as input to the descriptive mapping, which yields a predicted sensation as output. The robot makes the move, and uses the difference between what it actually observes and what it expected to observe as an error signal for weight change in the descriptive mapping network.

However, there are several reasons why an *a priori*, fixed topology might be unacceptable:

- **Error due to noise, failure, or unforeseeable events.** This isn't just a problem for fixed topologies: *any* means of predicting future states from the current one will be faced with the uncertainty and error that comes from being embedded in a complex physical world. This is addressed in the CNM by the orienting system, which refines the robot's estimate of its current location based on the coherence between the robot's current sensations and the sensory expectations provided by the predictive map. See section 3.
- **Systematic error.** General orienting and localization abilities are necessary, as just mentioned, but other, systematic sources of error could be handled with simpler methods in order to prevent an over-reliance on the orienting system, which could be computationally expensive or unreliable. Specifically, the CNM should be able to reflect the fact that the world *isn't* topologically uniform, but there are pockets of topological regularity in a heterogeneous space. For example, the topological mapping should be able to take into account that the kind of state transition that a move m effects in a room with thick carpeting is different from the transition effected by m in a tiled corridor, that floors can be sloped in some areas, etc. These deviations from a uniform topological mapping are systematic enough that they could be addressed directly by allowing slight modifications to the topological mapping, rather than relying exclusively on orienting strategies.
- **Limited resources.** A network has a finite quantity of representational resources, which it must distribute over space, which is infinitely dense. A fixed *a priori* topology will have a distribution of representational states over world states that will be independent of the qualitative character of those world states. For example, the amount of actual distance corresponding to the minimal effective difference between state vectors will most likely be constant in an *a priori* fixed topology. Even if it isn't constant, it certainly won't vary in a way related to the presence of obstacles, doors, walls, etc. in the environment. A more optimal use of the limited representational resources of a network would be to allow, e.g., a sparse representation of regions in which there are few or no obstacles, while regions around doors, or areas cluttered with permanent furniture are modeled in finer detail, with a smaller grain size. This requires an adaptive, variable topology.

The point is that the only major drawback of an empirical topology was that it seemed unlikely that a fully general topology could be learned through the weak method of error minimization. However, if the

[9] How the inverse model is to be learned and used for navigation is discussed in sections 4 and 5.

[10] Actually, in the case of a fixed topology, there seems to be no practical reason for having the topological mapping implemented in a PDP network at all.

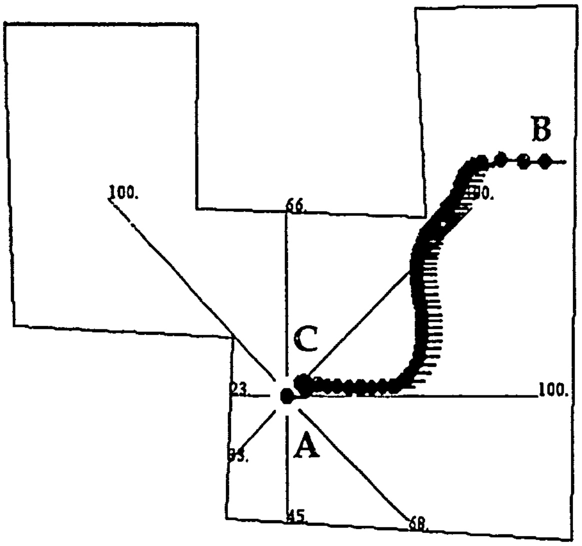


Figure 3: An illustration of the orienting system. A bird's-eye view of a two dimensional world is displayed. Rectilinear solid lines indicate walls. The robot is actually at location A, from whence it receives its sensory data (radial lines). Its state vector initially corresponds to location B. The state vector is iteratively modified so as to reduce the difference between expected and actual sensations. The trajectory of successive state modifications is shown, with the final fixed-point, C, being a rather good estimate of the actual location. (The orientation of the robot, which remained constant throughout all simulations described in this paper, is indicated by the horizontal bar extending from the robot's circular image.)

topological network is given a very good initial state, then perhaps one can have both a general and adaptive topology. Of course, simply allowing the weights of a "correct" topological network to be modified does not guarantee that it will successfully compensate for regular distortions in space, nor that it will optimally distribute its representational resources. Nor does it guarantee that the generality of the initial topological mapping can be maintained. Rather, these possibilities will have to be explored in future research.

3 THE ORIENTING SYSTEM

Two observations explain the need for an orienting system:

- Most means of using a map for navigation require the robot to know which map representation (a position on the paper in the case of a standard map; a state vector in the case of the CNM) corresponds to the current world state (location of the robot).
- Even with the best topological models and most accurate sensors, odometry-based location estimations drift away from the actual location quickly.

The orienting system, then, is a system that, given the robot's movement history, the predictive map, and recent sensory information, attempts to the find the activation pattern that corresponds to the current location.

3.1 STATE ESTIMATION: GRADIENT DESCENT IN ACTIVATION SPACE

Suppose the robot has constructed an acceptable, but not necessarily highly accurate, predictive map for a given region. But suppose, for some reason, the robot is in a different world state (p_{actual}) than the one that corresponds to the current state vector (x). Then it is no surprise that $D(x) \neq S(p_{actual})$, where S is the function from world states to the sensations they cause in the robot. Thus, there is error in the CNM: there is a difference between what is expected and what is observed.

Normally, error in a network is reduced or eliminated by modifying the weights of that network. But we are assuming that the map is more or less correct. *Ex hypothesi*, the source of error in this robot's CNM is its erroneous state estimation, not its predictive map as a whole. The way to reduce error becomes clear, then. use the error to modify the erroneous state vector, not the weights. Back-propagate the error signal from the output of the descriptive mapping network to the input, and change the activation patterns of the input units in proportion to the negative gradient of the error. Propagate this new input through the descriptive mapping again. If the output is still substantially different from what is observed, then back-propagate this error again, and so on.[11]

A state vector x_s is a fixed-point of this iterative process if $D(x_s) = S(p_{actual})$. That is, the network will, if the process converges,[12] find a state vector that is consistent with the current sensory data. This process is depicted in figure 3.

[11] The idea of using back-propagation to alter activations instead of weights has been considered before. (Williams '86) introduced the idea; (Mikkulainen and Dyer '88) used it to generate better internal representations; (Linden and Kindermann '89) and (Kindermann and Linden '90) provide good analyses of the procedure, and apply it to pattern completion problems, as well as other uses, (Chen, Belew, and Salomon '90) apply the idea to finding fixed points in iterative associative memories.

[12] A rigorous study of the convergence properties of the orienting system has not yet been performed, but informally tested cases generally converge, even with a poor predictive map.

3.2 SIMULATION DETAILS

To generate behavior like that illustrated in figure 3, a 2D world, with walls and a robot, was simulated. The sonar sensations of the robot at any given time were calculated by finding the distance to the nearest wall at each of the angles $\frac{2\pi k}{8}$; $k = 1, 2, \ldots, 8$ from the current orientation of the robot. Each component of the input vector was one of these distances, or 100, whichever was less (to reflect the range limitations of sonar).

A fixed *a priori* topology was used for the sake of simplicity and experimental control, so the robot only had to learn the descriptive mapping. Each sample in the training data consisted of an activation vector corresponding to a location, and the sensory input the robot had at that location. This data was collected by moving the robot around the environment, periodically storing the current location coordinates and the current sensations (distances at the eight angles). The coordinates were scaled to be within the range $0 \cdots 100$. Each scaled coordinate and (pre-scaled) distance was then translated into a four-unit activation vector using a representation scheme roughly based on one mentioned in (Hancock '88). A simplified description of the translation procedure is: if the value to be coded is 0, 33, 66, or 100, then all the units have 0 activation except for the unit 1, 2, 3 or 4, respectively, which has an activation of 1; if the value to be encoded is 0.25 of the way between 33 and 66 (i.e., 41.25), then unit 2 has an activation of 0.75 and unit 3 has an activation of 0.25, etc. These four-unit vectors were composed to make the appropriate input and output vectors (i.e., the input vector was composed of two four-unit vectors, one for each 2D coordinate, while the output vector had 32 units, 4 units for each of the 8 sensed distances). Roughly 200 samples were collected in this manner. The absolute orientation of the robot was kept constant.

The network was trained on this data, using Scott Fahlman's Quickprop algorithm (Fahlman '88), and his Lisp implementation of this algorithm. The network had 8 hidden units, and converged to within 1% of its final error within 500 trials.

After convergence, the behavior displayed in figure 3 was generated as follows. The robot was placed at the location A, and its current location coordinates were encoded into an input vector via the process described above. This input was forward-propagated through the network to produce an output vector, o. This was compared to the actual sensations the robot was receiving, s, to yield an error signal $\frac{1}{2}(s - o)^2$. This error was back-propagated in the normal manner (Rumelhart, Hinton, and Williams '86). The error signal at each input unit was normalized by dividing it by the number of weights from which the unit received an error signal. This error value was then multiplied by a scale factor of 0.2, and then added to the activation of each input unit. Activation values less than 0 or greater than 1 were truncated to 0 and 1, respectively. This process was iterated until a fixed-point was reached. Each change of input activation corresponds to a revision of the network's estimate of the robot's location, and is depicted in figure 3 by the chain-like succession of robot images.

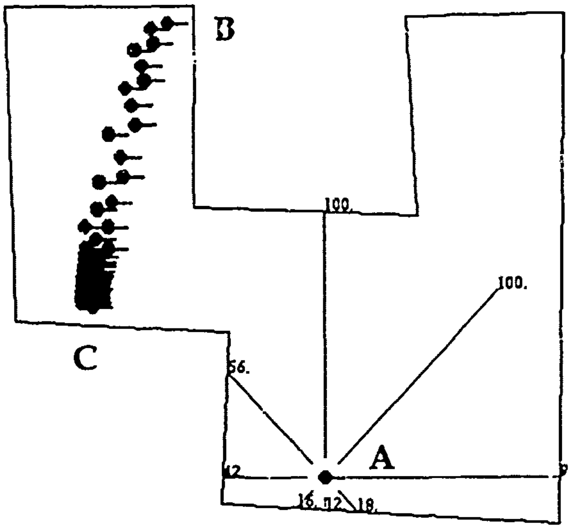


Figure 4: An illustration of how the simple orienting system is dependent on initial conditions. The system converges on the location (C), closest to the initial guess. that best matches the actual sensory data. The robot is roughly in the same actual location (A) as it was in figure 3, but since it has a different initial guess (B) as to where it is, the orienting system produces a different (and in some sense incorrect) estimate of the current location. This ambiguity can be overcome, if desired, by chaining constraints (see text).

3.3 OVERCOMING AMBIGUITY: CHAINING CONSTRAINTS

Of course, since any state vector x_i for which $D(x_i) = S(p_{actual})$ is a fixed-point of the orienting system, there is no guarantee that the state converged to corresponds to p_{actual}. Rather, the orienting system will (most likely) converge to the state vector, closest to the initial state, for which $D(x) = S(p_{actual})$. See figure 4, which was produced using the simulation just described for figure 3.

Note that even though this orientation scheme is susceptible to local minima, this does not mean that it can't be useful in navigation. For example, odometer drift is a constant problem, but it seldom happens that a robot's estimate is so wrong that the orienting system would converge to the wrong location. Thus, even the "nearest-neighbor" orienting system just described would be of use whenever the odometer drift is bad enough to make corrections based on the predictive map to be of use (i.e., whenever the odometer drift is greater than the relatively high margin of error of the predictive map), but not so bad that the robot could drift into another sensory "well" between consultations of the orienting system.

Furthermore, the final state vector is not (necessarily) the closest in actual space, but in activation space. Thus, if the network uses a state vector space of a dimensionality higher than actual space, and it uses an empirical (or at least variable) topology, it might be able to construct the mapping between state vectors and world states to be such that minimizing distance in activation space would correspond to some interesting trajectory in real space. For example, the orienting system might converge to a state vector for which $D(x) = S(p_{actual})$, but that also has some higher-order similarity to the initial state, such as "being in a room with one exit", rather than being merely the closest in a raw spatial sense. Such possibilities are suggested by the analysis in (Servan-Schreiber, Cleeremans, and McClelland '89) of the internal representations of a network that learns a finite state grammar, and warrant further exploration.

These qualifications aside, it is nevertheless in general desirable to have an orientation system that will work even when the initial location estimate is very poor, and when the state vector topologies are of little use. In such a case, the orienting system can operate by *chaining constraints.*

To chain constraints for orienting, a cascade structure like the one in figure 5 is used. All D and T mappings are copies of the descriptive and topological mappings, previously learned. SV_1 is initialized to the initial state estimate, and the current sensations are stored at AS_1. All other values are as yet undetermined.

The basic orienting process occurs between SV_1 and AS_1: SV_1 is modified until PS_1 matches AS_1. Then a move is made, and the activation pattern for this move is imposed on M_1. SV_1 and M_1 are propagated to PSV_2, and SV_2 is initialized to PSV_2. The move at M_1 is made.

The sensations after the move are stored at AS_2, and the orienting process occurs between SV_2 and AS_2. If the first orienting process yielded an incorrect guess as to the location before the move, then there will most likely be a discrepancy between PSV_2 and SV_2. This is a source of error that can be back-propagated to the original SV_1, which is changed accordingly, and the entire process repeats from the beginning. Of course, one move may not be enough to disambiguate, or the initial guess might have been correct. In such a case, the process is extended to SV_3, and so on. In order to prevent memory overflow, older states, moves, and sensations can be forgotten.

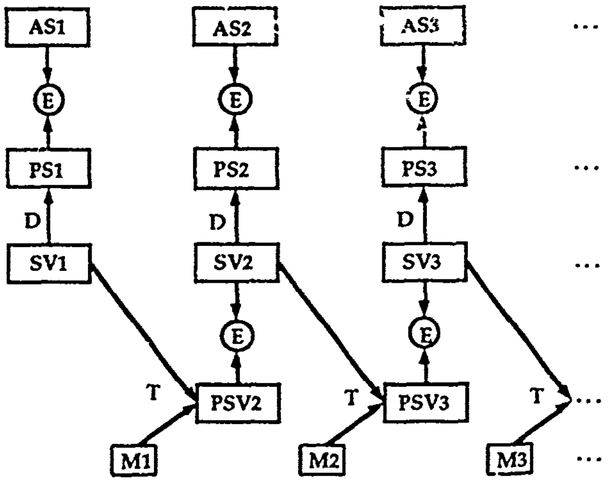


Figure 5: Using the method of chaining constraints in order to find the state vector that corresponds to the current world state. The modification to a state vector is determined not only by what error it yields under the descriptive mapping, but also by the error produced under the descriptive mapping by state vectors that follow and depend upon it. SV = state vectors; PSV = predicted state vectors; AS = actual sensations; PS = predicted sensations; M = moves; D = descriptive mappings; T = topological mappings; E = sources of error.

Simulations examining this more robust means of orienting are currently underway.

4 THE INVERSE MODEL

At least part of the task of navigation is this: given where I am and where I want to be, how do I get there? In the case of the CNM, this might correspond to the robot being given two state vectors, a and b, with a representing its current location and b representing its desired location, and the task being to come up with a move or set of moves that will take the robot from $R(a)$ to $R(b)$, where $R(x)$ is the world state to which x refers.

Of course, it would be very unlikely that this problem could be solved in general by a simple mechanism, for states that are arbitrarily distant from each other, separated by arbitrary obstacles. Rather, a simple mechanism can only provide moves that make transitions between relatively local states, and it must be up to some other system to find the sequence of states to be traversed Such a system will be discussed in the next section; for now we will consider how the CNM could compute the function P : *states* $\times$ *states* $\mapsto$ *actions* for nearby states.

4.1 DERIVING THE INVERSE MODEL FROM THE PREDICTIVE MAP

There are several ways a robot could use the predictive map to compute the function P:

- ***A priori* solutions.** Any CNM with an *a priori* topological mapping might just as well have an *a priori* inverse topological mapping. For instance, if the *a priori* mapping used the Cartesian coordinate system to represent states, and had associated with each move a corresponding vector, $(\Delta x, \Delta y)$, that indicated the change of state that action produces, then it should be trivial to also have the inverse association, from the difference between the actual and desired state, to a move that makes that transition.[13]
- **Heuristic search.** The robot could input a into the map, and test, for various moves m, whether $T(a,m) = b$. If not, another m is tried. However, it is unclear what means of sampling the space of moves (other than obvious heuristics such as "If you have just tried move m, don't try m^{-1}") would make a feasible search strategy.
- **Gradient descent in activation space.** In a manner similar to the mechanism proposed for the orienting system, the inverse model could be computed by inputting the a and an initial move "guess" m to the mapping T. The error $T(a,m)-b$ is back-propagated to the input units, and the move vector is changed so as to effect a gradient descent reduction of error.[14]

Although this last idea seems very promising, and is currently under investigation,[15] the possibility of actually learning the inverse model directly (as opposed to being given it, or deriving it from the forward model) should be considered.

4.2 LEARNING THE INVERSE MODEL

(Jordan '90) discusses how a network might learn an inverse model similar to the kind we are considering. One major difference is that Jordan assumes that the state information is available, or in other words, that the sensory information also serves as the state information. Nevertheless. his analysis is appropriate as long as care is taken to remember these discrepancies.

One idea of how to learn an inverse model is to try to learn it directly, by training a network on input/output pairs of the form $(a,b)/m$. In Jordan's framework, the robot could generate these training samples by actually performing an action in the world and observing what states result. The initial and final state are used as the input, the action that led from one to the other is the desired output. But, as pointed out before, in the case of spatial mapping the state vector corresponding to a particular location is not directly observable, so the best a robot could do would be to generate the training samples by inputting a state a and a move m to the forward topological mapping (T), to produce $(a, T(a,m))/m$ as a training sample.

Jordan provided three reasons for rejecting this method:

- **One-to-many.** Since there is more than one way to move from one state to another, the training data for a direct inverse model is potentially one-to-many. Back-propagation handles one-to-many training situations by averaging the desired outputs, which in general will not be a meaningful solution. This might be able to be avoided by restricting the set of actions being considered in forming the inverse model. For example, one might want to restrict the possible state transitions learned to be those that can be achieved by a simple action: motion in a straight line for some small distance, or rotation clockwise or counterclockwise through less than π radians. This would make the *states* × *actions* $\mapsto$ *states* mapping one-to-one, so its inverse, the *states* × *states* $\mapsto$ *actions* mapping, would be one-to-one as well.
- **Not goal-directed.** Direct inverse modeling samples action space to generate its training samples, and, at least in the case of Jordan's applications, using this method does not guarantee that the network will learn the mapping for the "desired" inputs (state vectors) in which one is interested. (However, it is unclear whether the proper "goal" for the CNM inverse model is to learn about specific desired states or actions in general.)
- **No direct connection to the world.** Jordan warns about using the network's forward model as a means of generating training data. It is much better to use the world itself when one can. The problem is that the world does not directly provide the state information that is needed to generate the training data, so the network must rely on its own predictions of what state will result (but see below).

Jordan proposes instead a means of indirect inverse

[13] Of course, such a topology will have all the problems of inflexibility mentioned before; a variable. *a priori*, inverse topology might overcome some of these problems. The learning involved in such a topology will be similar to that in an empirical topology; see section 4.2.

[14] Although gradient descent (the third item) is, strictly speaking, a form of heuristic search (the second item), I mean to make a distinction here between heuristics on the (conceptual) level of *actions* (heuristic search) and those on the (non-conceptual) level of *activations* (gradient descent). For a discussion of the conceptual/non-conceptual distinction, see section 7 and (Cussins '90).

[15] This idea is also being investigated by others; see (Thrun, Möller, and Linden '90).

modeling, or *forward modeling*. The idea is to compose the inverse (T^{-1}) and forward (T) mappings,[16] and learn an identity mapping across this composition. A desired state vector b and current state vector a is input to T^{-1}, which yields an action vector m. In the cases that Jordan is considering the error $T(a,m) - b$ is ignored; rather, the action m is executed and the state vector b^* is obtained directly from the world via the senses. The error $b^* - b$ is back-propagated though the $T \circ T^{-1}$ composition, but only the weights in T^{-1} are modified.

In this manner, two of the above limitations of the direct inverse approach have been addressed. By composing the T^{-1} and T mappings, the inverse model is forced to converge to a *particular* inverse solution, not plagued by the one-to-many problem. Also, forward modeling is goal-directed in Jordan's sense, since the input to the system is a desired state (but again, it is unclear whether this is an important constraint for the CNM inverse model).

However, for the spatial map learning tasks relevant to the CNM, an unmodified forward modeling approach remains susceptible to the third criticism mentioned above: it is not directly connected to the world. There is no immediate way to observe what the actual world state is after making a move, so there is no b^* available. Thus, the network is forced to use its own prediction, b, as a training output, a method which Jordan warns us against. Modifications need to be made if the CNM is to use forward modeling to learn its inverse model.

One idea is to apply forward modeling to the composition $D \circ T \circ T^{-1}$. That is, the error that is back-propagated is $D(T(a,m)) - S(p_{actual})$, with S being the sensory function from world states to sensations and p_{actual} being the current location, as defined before in section 3.1. This allows the error to be calculated, but not at the expense of having the network generate its own desired outputs: the world actually provides the error signal.

Another way of addressing this problem is to use the orienting system described in section 3 to generate the training outputs for the composition $T \circ T^{-1}$. In this case, $\hat{b}$, a world-dependent estimate of b^* (as opposed to the completely model-dependent estimate, b) is indirectly inferred from the current sensations by running the orienting process on the current sensations, $S(p_{actual})$. The pattern b is input to D as an initial guess for $\hat{b}$. The error $D(\hat{b}) - S(p_{actual})$ is calculated and back-propagated to the input in order to make changes to $\hat{b}$ that result in a better estimate of b^*. This process will converge on a state vector that, according to the predictive map, represents the closest place that matches the current sensory data. This estimate, $\hat{b}$, is used as a training output (via the error term $\hat{b} - b$) for $T \circ T^{-1}$ with the input b. Thus, the network's model, $D \circ T$ is involved in the generation of the training output, but so is the actual world, in the form of $S(p_{actual})$.

5 NAVIGATION

The purpose of having an inverse model is that it can assist the robot in navigating its environment. As pointed out in the previous section, the inverse model can only be expected to generate actions for relatively close state transitions. In order to be of use, the inverse model must be given an appropriate sequence of states from some other system. This section looks at a few ways that such sequences could be generated and used.

5.1 STATE-BASED NAVIGATION

State-based navigation takes as input an ordered sequence of N state vectors s_i and generates from them an ordered sequence of moves m_i such that after executing the nth m, the robot will be in the world state $R(s_n)$ (or at least guarantees that after the execution of all of the m_i, the agent will be in s_N).

This type of navigation could be useful for following previously encountered routes. While defining a route, the robot can periodically store the current state vector, such that after reaching its destination, it will have stored a sequence of state vectors. Then, if it ever finds itself in one of those states later, and it wants to go to another one of those states, it can input the current and neighboring state into the inverse model, which will output an action to take it to that next state, and the process can be iterated until the destination is reached.

Of course, things don't always go as planned. The action performed, m, will not always bring the robot to the desired state, s_i. There are three basic strategies for dealing with this:

- The robot can plan the next step as normal, ignoring that it is not where it should be (this might be satisfactory for small errors).
- The robot can use the orienting system to find the current state and then plan to the next desired state, s_{i+1}.
- Or the robot can be more conservative, and assume that it must reach s_i before it can reach s_{i+1}. Therefore, it uses the orienting system to find the current state, but plans to the same desired state as before, s_i.

[16] Although in some sense the inverse mapping is not a formal inverse of the forward mapping T, it will be denoted by T^{-1} here to emphasize the fact that $T \circ T^{-1}$ is an identity mapping.

5.2 CONTENT ADDRESSABLE PLANNING: DESIRED SENSATIONS

The CNM can also be provided with a means of generating actions based on desired sensations, to result in what is effectively a limited sensation-based inverse model: when given the current state and desired sensation, an action is output that will lead to a (nearby) world state that has those sensory properties. This is done by using a (state-based) inverse model and the orienting system again. The input to the D mapping network is initialized to some guess b (obtained by calculating $T(a, m)$), and back-propagating the error $D(b) - d$, where d is a sensation vector representing the desired sensations, through the network to change b until it is a state vector that indicates the nearest place that has the desired sensory properties. At this point, $T^{-1}(a, b)$ should indicate a move that could be made to reach a place with the sensory signature d.

Another way to achieve a similar result is to use the orienting system in a novel way. Instead of finding the current location, given the current sensory data and the initial guess b, the orienting procedure can be used to find a route in state space from the current location to a place that meets a sensory description. Given some desired sensations d, the state vector representing the current location, a_0, is input to the descriptive mapping D, and the error $D(a) - d$ is used to change a_0 in a gradient descent fashion to a new vector, a_1. The two state vectors, a_0 and a_1 are then input into the inverse model to produce an action to effect the transition. The process is then repeated, with a_1 as the next input to D. If the scale factor (conventionally denoted by η) used in the back-propagation is small enough, $T^{-1}(a_i, a_{i+1})$ will in general be meaningful (i.e., it will denote a performable and correct move).

This process can be illustrated by re-interpreting figure 3. The sensations at A are desired. The current location is B, and the current state vector denotes B. The orienting process generates a sequence of states (the trail of states shown in the diagram), which can be input to the inverse model in a pair-wise manner to generate a sequence of actions that will reach location C. Again, things will not always go as planned, so the robot might want to periodically use the orienting system in its proper capacity to ensure that its current state vector continues to be an accurate representation of its location.

As it stands, this procedure is unacceptable, since there is no guarantee that the orienting system will not produce a trajectory that runs through a wall or other permanent obstacle. This can be addressed by adding terms to the error function that will penalize state vectors that represent world states close to walls, even if they are very similar to the desired perception. For example, in the cases being considered here, where the components of the sensation vectors indicate the distance from walls at various angles, the error for the ith output node could be defined as $(D(a)_i - d_i) + \frac{\alpha}{D(a)_i}$, since the reciprocal of $D(a)_i$ is an indication of how close one would be to a wall if one were in the location denoted by a. The constant α is used to scale the importance of the obstacle avoidance term: the higher α is, the further the robot will stay away from walls.

This procedure can be (loosely) called "content-addressable planning" since it generates a plan based upon a qualitative specification of a place, rather than a conventional address, a coordinate, of the place. As in state-based navigation, content-addressable planning can be used in conjunction with stored routes. The robot, while determining a route, can store various "scenes" or sensations along the way. At a later time, this route can be followed by inputting these stored sensations to the content-addressable planner, which will move the robot along the route. Depending on the nature of the robot's environment, this method might be more robust than state based route following. The absolute state vector values stored might change their meaning between storage and use, due to a variable topology's compensation for a systematic change in the world, such as tread wear on the robot's wheels. Since the CNM assumes that a given location's sensory input to the robot is more or less constant, it might be best to remember a route based on these actual sensations, rather than coordinates in a varying topological code. Of course, even with a sensation-based route, errors will still creep in, but measures directly analogous to the three mentioned at the end of section 5.1 can be taken to help ensure that the destination is reached.

6 THEORETICAL MOTIVATION[17]

The primary theoretical motivation for constructing the CNM system is to understand how a robot can make the the transition from pre-conceptual to conceptual representations of space. The underlying hypothesis is that map construction and use is a paradigmatic case of concept formation, that the computational means underlying spatial concept mastery will be of a type similar to the means used for concept mastery in other domains. The preceding discussion and simulations were a start at an answer to the question. how might we use PDP to understand cognitive map construction and use? Now it is time to address the question: what could understanding cognitive map construction and use tell us about representation and the mind in general?

A complete account of cognition, natural or artificial, will have to be based on a theory of representation.

[17] Those readers of a less philosophical nature might want to skip this section and proceed directly to the conclusions in section 7.

Our theories will have to explicitly characterize the representations that play a role in cognitive systems as well as the content, or information, that is carried by these representations. Traditional (conceptual) ways of specifying content (such as in the ascription "The content of Leslie's belief is that this is a brown building") do so in such a way that requires that the agent to which the ascription is being made (Leslie) have the concepts used in the specification (e.g., brown and building). This means is inappropriate, then, for ascribing contents to simple systems that do not possess (many) concepts, and for many of our representational contents (such as perceptual and indexical ones) that are not conceptually mediated. We need a means (a non-conceptual means) of specifying contents that does not imply that the agent in question possesses the concepts used in the ascription. We also need a related account of the conditions under which it is appropriate to ascribe a particular content, so non-conceptually specified, to a particular physical system.

Consider the domain of spatial navigation, the ability to find one's way in the world. A plausible idea is that the first need, the need for a non-conceptual specification of the contents involved in navigation, can be answered by specifying contents in terms of an organism's abilities. For example, one might characterize the content of a particular computational state of a rat's brain in terms of the abilities it engenders, such as the ability to keep the end of the corridor in the center of its visual array. This would be a *non-conceptual* specification since, e.g., a rat doesn't need the concepts corridor or visual array in order to possess this ability or the contents which are specified in terms of it, and it would be a specification of *content*, since the rat certainly represents the world when it navigates. if the rat were exercising this ability in front of a life-size painting of a corridor, as opposed to an actual corridor, it would in some sense be wrong about the way it is taking the world to be, and being wrong about something is a good indication of the presence of representation.

Given this, one might think that the second need, the need for content ascription conditions, could be provided by any computational architecture that explained why a system possessed the abilities used in the content specifications. If, as in some cases seems likely, rats do not navigate by employing some concept of absolute location, but rather merely memorize the sequence of turns necessary to get from one place to another, then an account of how those abilities can be provided by a particular computational system might also provide the ascription conditions for the contents involved in rat navigation. But rats aren't the only animals that represent the world; we do, and oftentimes we do so conceptually, in terms of objective contents such as that carried by the concept location. Thus, there is a further requirement on our account of content, the requirement for an explanation of concept possession and acquisition. the transition from non-objective to objective ways of representing the world. This requirement imposes a constraint that restricts the class of explanatory computational architectures, a constraint that suggests that a clear member of the class would be a type of PDP network.

The reasoning behind this claim is as follows. Traditional, conceptual specifications of content are an attempt to characterize contents that are objective, contents that correspond to (could be specified in terms of) abilities that are perspective-independent, in that the abilities are appropriate in almost any context. An example of such an ability is using a conventional, paper-in-hand map: no matter where one is, or where one's destination, such a map indicates, at least roughly, in what direction one should go. But as we have seen, there are many contents that aren't objective or context-independent. These could be characterized in terms of abilities that are perspective-dependent, such as the ability to use a route-based map. Such an ability will only be of use if one is on the route in question, so unlike a conventional road map, its utility will depend on where one actually is. If we can provide a computational architecture that doesn't merely explain why a system has the abilities it does, but also explains why a system can move, via learning, from perspective-dependent abilities (such as route-based navigation) to perspective-independent abilities (such as full-blooded map-based navigation), then we can provide an explanation of how a non-conceptually characterized system can possess concepts. Analyzing a system non-conceptually allows us, unlike a purely conceptual analysis, to explain concept acquisition, since both the starting point (context-dependent contents) and ideal end (objective contents) of that process, as well as the interesting area in between, can be so analyzed.

We need, therefore, an architecture that renders explicit the non-conceptual contents in a system. PDP is a promising candidate for this because:

- PDP representations are of varying context-sensitivity and perspective-dependence,[18] thus making them more amenable to non-conceptual content specifications than are the representations in logical or symbolic architectures, which encourage a conceptualist, strictly objective interpretation.
- Conversely, conceptual specifications of content seem inappropriate for PDP representations, since it is typically difficult (hidden-unit analyses not withstanding) to isolate an aspect of a PDP network that corresponds to some objective feature of the task domain as registered by the theorist. In fact (to paraphrase Cussins), it may be that

[18] See, for example, Smolensky's "coffee" example in (Smolensky '88) and the discussion of it in (Cussins '90)

PDP needs non-conceptual analysis more than non-conceptual analysis needs PDP (Cussins '90, p. 431).

- PDP networks emphasize incremental learning, which is the kind of transformation on representations that would appear to be useful for a system to make the transition from perspective-dependence to perspective-independence, from context-sensitivity to objectivity, from mere representation to conceptual thought.
- Finally, the continuous mathematics and non-linearity of PDP networks provide for a flexibility in expressing relations and transitions between non-conceptual representations that would be lacking in a discrete, classical architecture that has been developed to express the relations between conceptual representations only.

These philosophical motivations depend heavily on the discussion in (Cussins '90).

7 CONCLUSIONS

It has been shown how a robot might, through interaction with its environment, acquire and maintain a map that allows it to predict what sensations it might have if it moved in a particular way.[19] Preliminary simulations indicate that the generalization properties of standard feed-forward networks trained with back-propagation are well suited to this spatial learning task, since a training set of only 200 points produced a network that could provide a reasonable estimate of the sensations to be found at an indefinite number of locations.

Work that remains to be done includes further simulations that vary orientation as well as location; simulations involving the chaining of constraints algorithm; simulations that test the relative feasibility of variable *a priori* vs. empirical topologies, and applications of all of these ideas to the actual Hero 2000 robot.

Acknowledgements

Special thanks to David Gurr, who played a major role in the birth of many of the ideas in this paper, to Mike Jordan, for valuable suggestions and encouragement, and to Dave Touretzky, for comments on a draft. Also thanks to Dave Chalmers, Adrian Cussins, Peter Dayan, Rainer Goebel, David Kirsh, John Woodfill, and Ramin Zabih for discussion and suggestions. This work was made possible by support from the Center for the Study of Linguistics and Information at Stanford University, the Systems Sciences Laboratory at Xerox PARC, and by grants from the San Francisco branch of the English-Speaking Union, the Chancellors of the UK Universities, and the Oxford University Overseas Student Support Scheme.

References

Carpenter, G. A., and Grossberg, S. (1987) A massively parallel architecture for a self-organizing neural pattern recognition machine. In *Computer Vision, Graphics, Image Processing* 37, pp. 54-116.

Chalmers, D. (1990) Syntactic transformations on distributed representations. To appear in *Connection Science.*

Chen, J. R., Belew, R. K., and Salomon, G. B. (1990) A connectionist network for color selection. In *The Proceedings of the International Joint Conference on Neural Networks, January 1990*, pp. 467-470. San Diego: IEEE.

Cussins, A. (1990) The connectionist construction of concepts. In M. Boden (editor), *The Philosophy of Artificial Intelligence*, pp. 368-440. Oxford: Oxford University Press.

Elman, J. (1989) Finding structure in time. Center for Research in Language technical report 8801. La Jolla: UCSD.

Fahlman, S. E. (1988) Faster-learning variations on back-propagation: an empirical study. In D. Touretzky, G. Hinton, and T. Sejnowski (editors) *The Proceedings of the 1988 Connectionist Models Summer School*, pp. 11-20. San Mateo: Morgan Kaufmann.

Hancock, P. J. B. (1988) Data representation in neural nets. an empirical study. In D. Touretzky, G. Hinton, and T. Sejnowski (editors) *The Proceedings of the 1988 Connectionist Models Summer School*, pp. 11-20. San Mateo: Morgan Kaufmann.

Jordan, M. (1986) Serial order: a parallel distributed processing approach. Institute for Cognitive Science report 8604. La Jolla: UCSD.

Jordan, M. and Rumelhart, D. (1990) Forward models. supervised learning with a distal teacher. Submitted to *Cognitive Science*. Hillsdale, NJ. Lawrence Erlbaum.

Kindermann, J., and Linden, A. (1990) Inversion of neural networks by gradient descent. To appear in *Parallel Computing.*

Linden, A., and Kindermann, J. (1989) Inversion of multilayer nets. In *The Proceedings of the First International Joint Conference on Neural Networks.* San

[19] (Mozer and Bachrach '89) present an alternative way of achieving this kind of functionality. However, their approach differs notably from the CNM: the action space is small and discrete (there is a weight matrix for each action); there is a prior, localist encoding of world states (one unit per state); and the back-propagation "unfolding in time" learning procedure (Rumelhart, Hinton, and Williams '86) is used.

Diego: IEEE.

Mozer, M. C., and Bachrach, J. (1990) Discovering the structure of a reactive environment by exploration. In D. Touretzky (editor) *Advances in Neural Information Processing Systems 2*, pp. 439-446. San Mateo: Morgan Kaufmann.

Mikkulainen, R., and Dyer, M.G. (1988) Encoding input/output representations in connectionist cognitive systems. In D. Touretzky, G. Hinton, and T. Sejnowski (editors) *The Proceedings of the 1988 Connectionist Models Summer School*, pp. 347-356. San Mateo: Morgan Kaufmann.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986) Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland (editors) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, pp. 318-362. Cambridge: MIT Press.

Servan-Schreiber, D., Cleeremans, A., and McClelland, J. (1989) Learning sequential structure in simple recurrent networks. In D. Touretzky (editor) *Advances in Neural Information Processing Systems I*, pp. 643-652. San Mateo: Morgan Kaufmann.

Smolensky, P. (1988) On the proper treatment of connectionism. In *Behavioral and Brain Sciences* **11**, pp. 1-74.

Thrun, S. B., Möller, K., and Linden, A. (1990) Planning with an adaptive world model. Internal report. German National Research Center for Computer Science, D-5205 St. Augustin, Postfach 1240, F.R.G.

Williams, R. (1986) Inverting a connectionist network mapping by back-propagation of error. In *The Proceedings of the 8th Annual Conference of the Cognitive Science Society*, pp. 859-865. Hillsdale, NJ: Lawrence Erlbaum.

Part VIII

Speech and Vision

UNSUPERVISED DISCOVERY OF SPEECH SEGMENTS USING RECURRENT NETWORKS

Antoine Doutriaux* David Zipser
Cognitive Science Department
University of California, San Diego
LA JOLLA, CA 92093

Abstract

The purpose of the project described here is to find what segments a network can discover by itself, using only prediction as a teacher. A recurrent network was trained to do a prediction task, using a speech spectrogram as both input and teacher signals. The error vector and hidden unit activation transitions were used to extract segments from the multi-speaker, continuous speech TIMIT database.

The network was analysed to see what speech segments it discovered. Many of the segments the network found correspond to TIMIT phones and are very well segmented, but some TIMIT phones are not extracted.

We are examining the use of these segments to drive a second network to label speech. Early results are encouraging because the system gives the same error rate whether network segments or the segments provided with the TIMIT database are used.

As Klatt remarks in [4], the "recognition of a small set of words would not be difficult were it not for the remarkable variability seen in the pronunciation of any given word". When we deal with many speakers, this variability is even more important. Variability may be due to:

- when dealing with one speaker, emotional state, speaking rate, accent, ...all have a great influence on the speech spectrum and on the articulation between successive units.
- when dealing with different speakers, anatomical differences due to age, sex, ...result in many acoustical differences for the same word or sentence;

Variability may also be due to context:

- different speaking rates result in different articulatory movements, that cause both temporal and spectral variations, as explained in [10];
- a worse context variation is due to coarticulation, where features of one phoneme may be spread over the previous and next phonemes.

These effects make speech segmentation, the dividing of continuous speech into elementary units known as segments, a hard task. Segmentation is an impor ant issue in many fields. In psycholinguistics and phonology, researchers have done a great deal of work attempting to discover the speech units. This is of importance in understanding the nature of language, and to answer questions such as: how do infants acquire their lexicon, and how do adults do the lexical access. There is no real consensus on this issue but, as Mehler *et al.* note in [6], psycholinguists try to represent words in terms of linguistic units, such as syllables.

In the field of Continuous Speech Recognizer Systems (CSRS), the problem facing researchers is to develop robust systems that translate a speech signal into its phonetic or alphabetic equivalent. This task can be done with very good accuracy when the number of speakers is limited, or when the vocabulary is restricted to a few words, syllables or phonemes. But error rates in the general case, that is with continuous speech in a multispeaker environment, are generally poor (of the order of 60 to 70 % correct recognition). Nevertheless, the number of applications of this type of system is enormous.

Generally, speech learning based recognizers have two stages:

1. first, the system is provided with a speech signal and the corresponding desired output and tries to learn the correspondence between input and output;
2. then, the trained system should give the desired output for any other input.

*The first author can now be reached at: D.C.A.N., Sous Direction Etudes, 83000 TOULON CEDEX, FRANCE.

A problem :vith this approach is that the subunits of speech useu in the training stage are generally chosen a priori, considering the complexity of the speech signal, better results might be obtained if the speec:ı subunits themselves could be learned.

At the moment, the best techniques for CSRS use Hidden Markov Models [5]. Recently, the emerging Parallel Distributed Processing approach [9] has also achieved good results in speech recognition [3, 8]. But both methods use a training period, where a priori information is given to the system. In this project, a form of unsupervised learning, prediction, was used to automatically find speech segments. The hope was that the recurrent network used would be able to find an appropriate set of speech segments in continuous, multispeaker speech. Segmenting speech using prediction is based on the supposition that prediction is possible within speech segments but not possible across segment boundaries. This suggests the error will be low within a segment and high at the very beginning of a new segment. It also suggests that hidden unit activations will be stable within a segment but change rapidly at segment boundaries. Both of these effects were observed, but hidden unit state change was found to be the better way to identify speech segments.

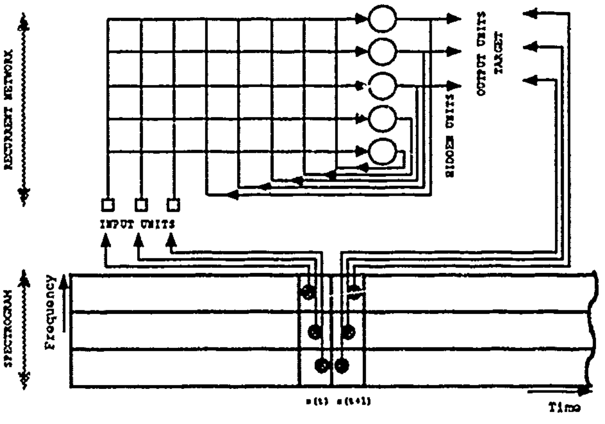


Figure 1: *Architecture of the network. An input unit and an output unit is dedıcated to each of the spectrogram channels. In thıs example, 2 addıtıonal hidden unıts are shown. Each unıt ıs connected to every other unit by a changeable weight. Sound ınput was generated from the TIMIT database which contaıns speech dıgıtalızed at 16-kHz. An FFT over 16-ms frames, advancıng 8-ms per frame was reformatted using logscale ranges to give the spectral magnitudes over 16 frequency ranges. These spectral magnitudes were normalızed usıng a 3-s wındow average and squashed using the logistic function $f(x) = \frac{1}{1+exp(-x)}$. Each tıme slice of thıs spectrogram was ınput to the network as explained in the text. The network was started with random weights in the range* $[-1, 1]$.

A continuous, unsupervised prediction task was used to train the network. A timeslice at time t from the continuous speech spectrogram was input to the network with the timeslice at t+1 as the teacher signal. This is the natural task given the dynamical nature of speech and corresponds to the human ability to guess the end of a segment from the beginning of the segment. The architecture of the network used is shown in Figure 1. A spectrogram was used as input because this is a low-level, continuous representation close to what is perceived by the ear, and has no a priori segmentation imposed upon it. Using such a spectrogram we hoped the network would find low-level acoustical units, but did not expect it to discover temporally-extended segments such as words.

A r...urrent network architecture was used because prediction requires memory. The algorithm used was Back Propagation Through Time (BPTT), introduced by Rumelhart *et al.* in [9] and described precisely by Williams and Zipser in [11]. Figure 2 shows the schematic representation of the algorithm used.

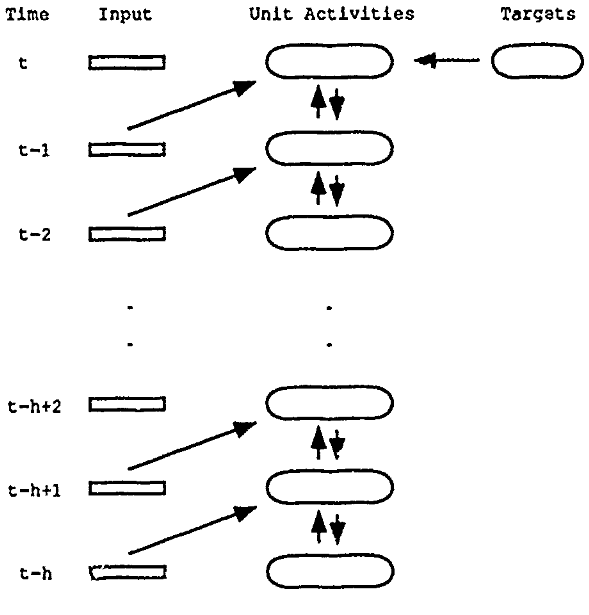


Figure 2: *Schematic representation of Backpropagation Through Time: this network consists of a stack of h copies of a recurrent network. At time t, the network of time t-1 is copied and the oldest copy from time t-h is thrown away; the input of time t is then propagated through this new copy; the error is propagated through the stack and the weight changes due to every level are accumulated in the new network.*

In the results presented below, the network used had 8 hidden units, and 5 copies of the recurrent network were kept, which seems to be about optimal.

Some preliminary studies were carried out to see if the error could be used for segmentation in a simplified version of the task. The database used by Elman and Zipser in [1] was used. This database consists of 520 instances of the 9 different "consonant followed by vowel" (CV) tokens from the set of consonants /b d g/ and set of vowels /a i u/. All tokens were extracted from real speech, trimmed to a uniform duration of 64-ms and appear in random order. The results obtained are described below and in Figure 3:

- in the first part of the training, the error became high when a new token arrived then relaxed relatively quickly. This can be seen in the top row of Figure 3. The peaks of the error vector give the position of the tokens in the spectrogram with very good accuracy.
- after a while, the network learned the length of the tokens and took the average value of the incoming time slice when it expected a new token. This minimizes global error, as the algorithm is intended to do, but means that the error vector is much less segmented or much more noisy. This can be seen in the bottom two rows of Figure 3.

Even when randomly shortening the tokens to their first n-ms, n being picked between 38- and 64-ms for every token, the network computed the average length of the tokens. So after learning to recognize the different tokens, the network continued to reduce global error by taking the expected value at the beginning of a new token, the beginning being known from the average length. This result means that error does not appear to be a good way to detect segement boundaries.

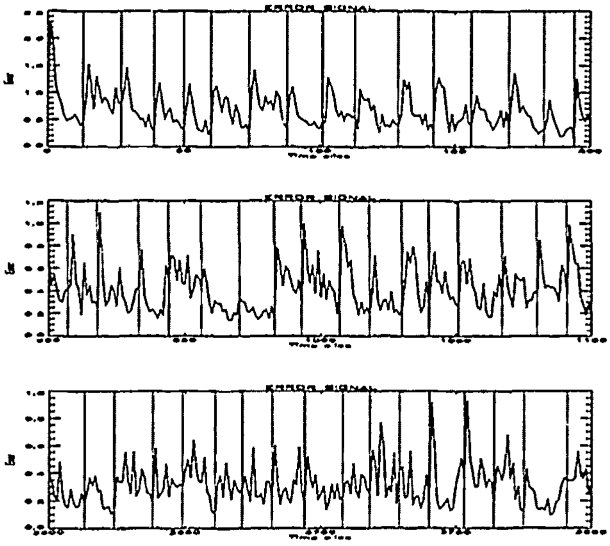


Figure 3: *Error signal after 0, 900, 3600 time steps. The dashed lines show the position of the successive tokens.*

Having found that the error vector is not rich enough to do segmentation, we looked at changes in hidden unit activations to detect a switch from one network state to another. Such a switch would indicate that the network had detected a new segment in the input. For this purpose, the TIMIT database [2] was used. TIMIT is a large multispeaker database, designed to train and evaluate speech recognizers. Each of 420 speakers says 10 sentences, 2 of which are the same for every speaker. The sentences have been segmented and labeled by a human expert into 62 previously defined TIMIT phones. In the work reported here, only a part of TIMIT was used, taking a training set of 84 speakers and a testing set of 27 different speakers. The 2 sentences common to all speakers were removed from both sets to prevent bias in segment contexts. The network used had 8 hidden units. Figure 4 shows an example run on a part of the TIMIT database. Note that the hidden unit activations show sudden swings in magnitude, often at segment boundaries.

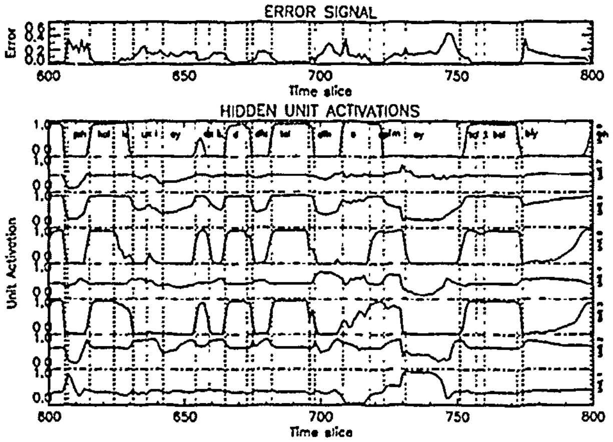


Figure 4: *Example of a run on a part of the sentence: "Some observers s/peculated that this might be h/is revenge on his home town". The graph shows the 8 hidden unit activations versus time. The upper graph is the error signal versus time. The dashed lines show the position of TIMIT phones.*

To use these swings for segmentation, we define a tick as a jump from one state to another in the hidden unit activation space. These ticks delineate the speech segments found by the network. Ticks are extracted from the hidden unit activations using the following transformation: let $\vec{h}(t)$ be the hidden unit activations vector at time t: $\vec{h}(t) = [h_1(t), h_2(t), \ldots, h_n(t)]$, where n is the number of hidden units, and t is the time step. The first derivative, $\vec{d}(t)$, is computed using:

$$\vec{d}(t) = \vec{h}(t) - \vec{h}(t-1)$$

We need to remove noise from $\vec{d}(t)$ (which corresponds to successive positive and negative derivatives), and also need to identify state changes which occur slowly over time. Thus we sum $\vec{d}(t)$, with a small coefficient, over p time slices. So let's define:

$$\vec{s}(t) = \vec{d}(t) + \alpha\vec{d}(t-1) + \alpha^2\vec{d}(t-2) + \ldots + \alpha^p\vec{d}(t-p)$$

Typically, $\alpha = 0.7$ and $p = 4$. The maximas of the magnitude of the vector $\vec{s}(t)$ give us the ticks extracted from the hidden unit activations.

$$tick(t) = \begin{cases} \|\vec{s}(t)\| & \begin{cases} if & \|\vec{s}(t-1)\| < \|\vec{s}(t)\| \\ and & \|\vec{s}(t+1)\| < \|\vec{s}(t)\| \end{cases} \\ 0 & \text{otherwise} \end{cases}$$

Figure 5 shows extracted ticks obtained using this transformation. After this transformation, any hidden unit that swings from 0 to 1 over 1,2,...or p timeslices will produce an extracted tick of the same magnitude. By summing over p timeslices, the noisy units do not contribute to the magnitude of the extracted ticks. Note that applying the transformation to the input spectrogram itself does not give good results - the spectrogram is too smooth.

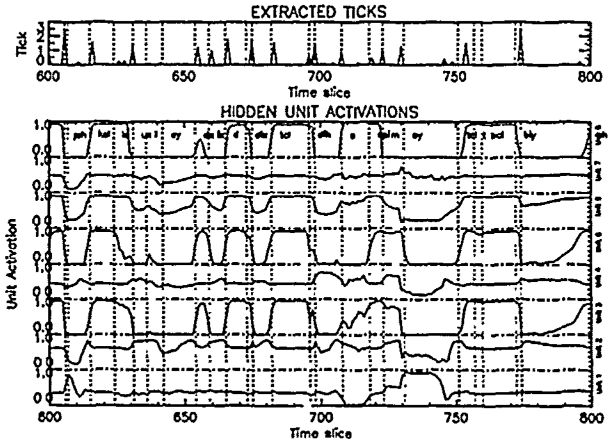


Figure 5: *Ticks extracted from the hidden unit activations. Note the correspondence between extracted ticks and TIMIT phones.*

The ticks extracted in this way were compared to those of the TIMIT database. Figure 6 shows the possible configurations of extracted ticks with respect to the TIMIT segments. The tolerance of one time slice to the left and two time slices to the right is not symmetric because of the small time decay introduced by the transformation. We define the categories *correct* where an extracted tick matches a TIMIT phone, *insertion* where an extracted tick occurs without any TIMIT phone, and *deletion* where a TIMIT phone occurs without an extracted tick. If we regard the magnitude of a tick as a measure of the likelihood of a new segment, we can define a threshold value below which ticks will be ignored and set the threshold to maximize the number of *correct* ticks and minimize the number of *insertions* and *deletions*. Figure 6 plots the number of *correct, insertion* and *deletion* ticks above threshold against the possible range of threshold values. This shows that threshold = 0.2 is the most appropriate value.

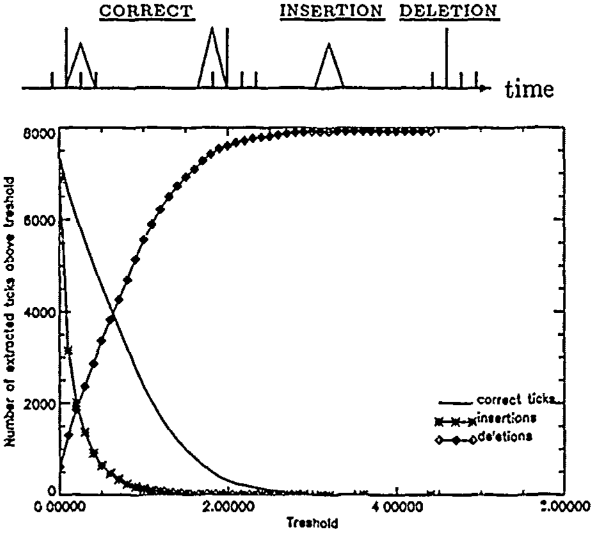
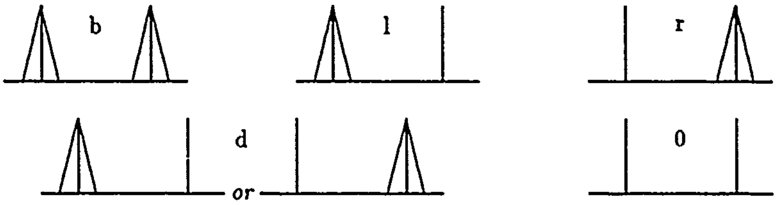


Figure 6: *The top part of the figure shows the possible configurations of extracted ticks with respect to TIMIT phones. The vertical solid line indicates the position of a TIMIT phone, the triangle indicates the extracted tick. In the bottom part of the figure, we plot the number of correct, insertion and deletion ticks above threshold against threshold, as described in the text. In this case, threshold would be set to 0.2.*

A key point is that since we have used unsupervised learning, we can't expect to compare the network's extracted ticks and TIMIT's phones directly:

1. TIMIT segmentation was done by human experts bringing to bear all the theory and tools of classical segmentation. Note that human experts may disagree over segmentation, also that TIMIT may have its own error rate in segmentation;
2. the network began with no a priori knowledge nor did it have an intelligent teacher, so it has learned autonomously the segmentation it performs;
3. there are systematic differences between TIMIT and the network. For example the network never segments /b/ and /a/ in *ba*, so it puts only one tick where TIMIT puts two.

To analyse the composition of the extracted segments, we have looked at all of the possible pairs of TIMIT phones appearing above a minimal frequency. The probability with which the left and the right phone of the pair is extracted was computed over all the instances of a given pair of TIMIT phones. Six configurations are possible as shown in the top part of Figure 7. A square matrix was constructed



```
                                    RIGHT PHONE

      | i i e a u i a a u u a a e a o a o l r y w e a e e e e m n n c j d b d d n g p t k q z z v f t s s h h p t k q b d g
      | y h h e x x x h w h o a y y y w w         r x l m n n     g h h h     x x             h     h   h h v c c c c c c c
      |                                             r       g                                                 l l l l l l l
  iy  |           0                       1                     1 0                       0 b   b b           b r r     r
  ih  |                                   1                   r r d           1             r         b b       r b     r r
  eh  |                                   1 1                 r b             b                 r     r           b
  ae  |                                                         r             r                       1         b b
  ux  |                                                       r
  ix  |                                                       b r r                         r   r     r       b b r     r b
  ax  |                                   1                   d d                               b r   r       r       b
  ah  |                                                       b r             b                 b             b r
  uw  |
  uh  |
  ao  |                                   1 0                                                         r
  aa  |                                   1 1                   b                                     r         b
  ey  |                                                         r             r                         r       r r     b
  ay  |                                                         b                                               b b     r
  oy  |
  aw  |                                                         l
  ow  |                                                         d                                     r
  l   | d d r r   0 d b     d 0 d d     d                                                           r
  r   | 1 0 1 1   0   1       d 1 0     0                     r                                                         r
  y   |         0
  w   | d   1     d d 1         1
  er  |           0                                                                           d         r
  axr |                                                       r                             d         r                 r
  el  |
  em  |
  en  |
L eng |
E m   | b         b d b         r r                                                                           1
F n   | b b r     r                 b                               1   1             1 r         1         1       1
T ng  |
  ch  |
P jh  |           r
H dh  | r r r r   r r           r
O b   | r r           r         r       r r
N d   | r r       r                         d
E dx  | b 1       1 1
  nx  |
  g   |                                     r
  p   |     r                 r           r r     r r
  t   | r b b b b b     b       b b         b                                                         r
  k   |       r   b   b       b             r   r     r                                               r
  q   |   1 r r   r   b     1 d
  z   |   r       b b                                                                                     1     b
  zh  |
  v   |           1 1
  f   |                     r               r       r
  th  |
  s   | b b b     b b b           b                                                                           b b b
  sh  |           b                                       b
  hh  | r     d
  hv  |
  pcl |                                                                             1
  tcl |                                                             b   1   1         b               1
  kcl |                                                                               b 1
  qcl |
  bcl |                                                                   1
  dcl |                                                               1   1 1               1
  gcl |                                                                           1
```

Figure 7. *The top part of the figure shows 5 of the 6 possible extracted tick configurations. the vertical lines indicate the position of the TIMIT phones, and the triangles indicate the position of the extracted ticks. In case b, both TIMIT phones are extracted, in case l, only the left phone is extracted, in case m, either the left or the right phones are extracted unpredictably. The sixth configuration happens when the number of instances of this pair is too small ie. less than or equal to 8. The bottom part is the matrix of the phone pairs types. The empty elements correspond to too small a number of instances of the given pair. Note the columns and rows filled with the same type, which give interesting information about the corresponding phone.*

such that, for each pair $/i/$ and $/j/$ of TIMIT phones, the element m_{ij} of this matrix represents the way the network segmented the pair of phones $/i/$ $/j/$. This matrix is shown in the bottom part of Figure 7. The first thing to note about this matrix is that the element density of the array is proportional to the frequency of the corresponding pairs of phones: CV and VC syllables are more frequent than CC and especially VV syllables. In the case of the CV and VC syllables, two types of consonants can be distinguished:

1. Easily extracted consonants: the corresponding column is filled with b's or r's and the corresponding row is filled with b's or l's. For example /s/, /t/, /m/, /n/, /dx/, /k/, /v/. Within this category, some like /s/, /t/ and /k/ prevent their neighbor from being extracted while others "help" their neighbor to be extracted.
2. Never extracted consonants. this is the case with /l/, /dh/, /b/, /d/, /p/, /q/. Their neighbor is generally extracted. This result was foreseeable because the network was unable to segment the consonant sound in the set /b d g/ x /a u i/.

Note that the closures are always segmented when following a vowel. The points made above are still true in the case of the CC consonants. /n/, /s/,... are always segmented whereas /b/, /g/, /p/, /f/ are hardly ever extracted. The closures are always segmented. In [7], Ohala *et al.* insists on an asymmetry between CV and VC syllables: the asymmetrical effects of coarticulation in these syllables mean that the features of the vowel are spread more into the preceding consonant in a CV sequence than the features of the consonant are spread into the preceding vowel in a VC sequence. Given this asymmetry, we might expect our network to have different segmentation rates for the different cases. The segmentation done by our network for the different sequences is shown in Table 1, which shows that there is some difference in segmenting CV and VC pairs, as expected from O'Hara's observation. The easiest phone for the network to extract is the consonant when preceded by a vowel. This means that the hidden unit activations swing more strongly when switching from a vowel to a consonant, presumably due to the lesser coarticulation effect in the VC case.

CV	*C*	70%
	V	74%
	both	52%
VC	*V*	75%
	C	83%
	both	63%
CC	C_1	72%
	C_2	75%
	both	51%

Table 1. *Segmentation done by the network on the VC, CV and CC syllables. the figures indicate the percentage of segmentation on the left, right and both phones of the pair, for the pairs appearing with a frequency greater than 8.*

We wanted to see if our extracted ticks could be used to label the spectrogram. We trained a classic feed-forward algorithm to attempt this. After training the recurrent net as described above on 88 speakers, we tested it on 80 other speakers and used the extracted ticks obtained on this set to train our recognition system. The recognition system was tested on another 20 speakers.

In order to reduce the problem to a reasonable size, we limited ourselves to the 16 vowels used by Waibel *et al.* in [3]. These vowels are listed in Table 2. After testing the segmentation network on the set of 80 speakers, each extracted tick corresponding to one of the 16 TIMIT vowels of Table 2 gives one input pattern for our database: this pattern corresponds to a window of the spectrogram extending a given width to the right and a given width to the left of our extracted tick. Let w be the total width of the window and f the number of frequency channels, this means that our input patterns are of dimension $w * f$. The output corresponding to this pattern is distributed over 16 bits, each representing one of our vowels. Either 4, 7, 10 or 20 hidden units were used. The architecture of this system is described in Figure 8.

Phone	Example	Phone	Example
/iy/	beat	/eh/	bet
/ix/	roses	/ah/	butt
/uh/	book	/aa/	cot
/ay/	bite	/aw/	about
/ih/	bit	/ae/	bat
/ax/	the	/uw/	boot
/ao/	bought	/ey/	bait
/oy/	boy	/ow/	boat

Table 2: *List of the 16 vowels used in our recognition task.*

Because of the high variability in the length of the vowels used (from 60- to 180-ms), the best results were obtained when training two different networks separately. For the first network, the total width of the window equaled 104-ms and only the 7 shortest syllables were presented; for the second network, the window was 152-ms wide and the network was trained only on the 9 remaining syllables. After training, the networks were tested on 20 other speakers. in this case, the recognition rates were 52% for the short vowels and 49% for the long vowels. The goal of this part of the project was to see to which extent our ticks could be used by a labeling system to label the spectrogram. so we repeated the above using the TIMIT segments instead

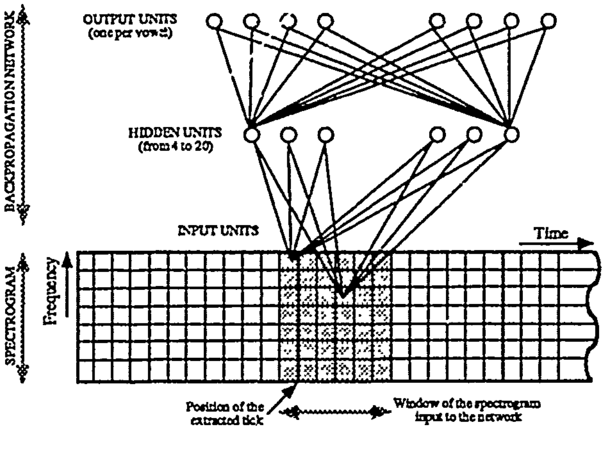

BACKPROPAGATION NETWORK
OUTPUT UNITS (one per vowel)
HIDDEN UNITS (from 4 to 20)
INPUT UNITS
Time
SPECTROGRAM
Frequency
Position of the extracted tick
Window of the spectrogram input to the network

Figure 8: *Architecture of the labeling system.*

of our ticks. The results in this case were 54% for the short vowels and 51% for the long vowels. These results have to be compared with the 56 % achieved by Waibel *et al.* in [3], for the case of a single TDNN architecture. Moreover, the fact that the results are the same whether using our ticks or the TIMIT segments is interesting because it implies that our system could be used to do the preprocessing for a labeling system that needs presegmentation.

Our initial goal was to discover what types of speech segments a recurrent network could find by itself. We have produced a set of segments that consist of phones or of pairs of phones that could be used by a recognition system. It is of some interest to see what the individual hidden units do in the network. Simple spectrograms were presented to the network and the subsequent activity of the hidden units was analysed. First, a dirac function was presented to the network on one of the frequency channels individually. For each input line, there is at least one HU that computes the first derivative of this input signal. This seems relevant when trying to predict the next time slice. In addition each of the input lines was activated one at a time, for 150-ms, every 300-ms. Of the 8 hidden units, 3 were selectively sensitive to a given range of frequency. The others compute different quantities, whose significance is not known.

We wish to thank Jeff Elman, Tony Smith, Mark Dolson and David Corina for many useful discussions in the development and writing of this project.

References

[1] J. L. Elman and D. Zipser. Learning the Hidden Structure of Speech. *Journal of the Acoustical Society of America*, 83, 1988.

[2] John S. Garofolo. Getting Started with the DARPA TIMIT CD-ROM: An Acoustic Phonetic Continous Speech Database. Technical report, National Institute of Standards and Technology (NIST), Gaithersburgh, MD, 1988.

[3] Nobuo Hataoka and Alex H. Waibel. Speaker Independent Phoneme Recognition on TIMIT Database using Integrated Time-Delay Neural Networks. Technical Report CMU-CS-89-190, School of Computer Science, Carnegie Mellon Un., 1989.

[4] K.H. Klatt. The Problem of Variability in Speech Recognition and Models of Speech Recognition. In J.S. Perkell and D.H.Klatt, editors, *Invariance and Variability in Speech Processes*, pages 300–324. New Jersey: Lawrence Erlbaum, 1986.

[5] K.F. Lee and H.W. Hon. Speaker Independant Phone Recognition using Hidden Markov Models. Technical Report CMU-CS-88-121, Carnegie Mellon Un., March 1988.

[6] J. Mehler, E. Dupoux, and J. Segui. Acquisition Constraints on Speech Recognition Models. In G. Altman, editor, *Cognitive Models of Speech Processing (to appear)*. MIT Press, 1990.

[7] John J. Ohala and Haruko Kawasaki. Prosodic Phonology and Phonetics. In *Phonology Yearbook 1*. University of Cambridge Press, 1984.

[8] A.J. Robinson and F. Fallside. Phoneme Recognition from the TIMIT Database using Recurrent Error Propagation Networks. Technical Report CUED/F-INFENG/TR.42, Cambridge University Engineering Department, Cambridge, England, 1990.

[9] D.E. Rumelhart, G.E.Hinton, and R.J.Williams. Learning representations by back-propagating errors. In D.E.Rumelhart and J.L.McClelland, editors, *Parallel Distributed Processing. Explorations in the Microstructure of Cognition.*, volume 1. MIT Press, Cambridge, MA, 1986.

[10] J. Vaissiere. Speech Recognition: a Tutorial. In Frank Fallside and William A. Woods, editors, *Computer Speech Processing*, pages 191–242. Prentice/Hall International, 1985.

[11] R.J. Williams and D. Zipser. Gradient-Based Learning Algorithms for Recurrent Connectionist Networks. Technical Report NU-CCS-90-9, Northeastern University, 1990.

Feature Extraction using an Unsupervised Neural Network

Nathan Intrator
Div. of Applied Mathematics, and Center for Neural Science
Brown University
Providence, RI 02912

Abstract

A novel unsupervised neural network for dimensionality reduction which seeks directions emphasizing distinguishing features in the data is presented. A statistical framework for the parameter estimation problem associated with this neural network is given and its connection to exploratory projection pursuit methods is established. The network is shown to minimize a loss function (projection index) over a set of parameters, yielding an optimal decision rule under some norm. A specific projection index that favors directions possessing multimodality is presented. This leads to a similar form to the synaptic modification equations governing learning in Bienenstock, Cooper, and Munro (BCM) neurons (1982).

The importance of a dimensionality reduction principal based, solely on distinguishing features, is demonstrated using a linguistically motivated phoneme recognition experiment, and compared with feature extraction using principal components and back-propagation network.

1 How to construct optimal unsupervised feature extraction

When a classification of high dimensional vectors is sought, the *curse of dimensionality* (Bellman, 1961) becomes the main factor affecting the classification performance. The curse of dimensionality problem is due to the inherent sparsity of high dimensional spaces, implying that the amount of training data needed to get reasonably low variance estimators is ridiculously high. One approach to the problem is to assume that important structure in the data actually lies in a much smaller dimensional space, and therefore try to reduce the dimensionality before attempting the classification. Hence, the desired property of a dimensionality reduction/feature extraction method is to loose as little information as possible after the transformation from the high dimensional space to the low dimensional one. This motivation underlies methods such as principal components (PC), mutual information maximization (Linsker, 1986), and self supervised form of back-propagation.

At a first glance, it seems that a supervised feature extraction method will always be superior to an unsupervised one, because if one has more information about the problem, it is natural to suppose that finding the solution is easier. However, unsupervised methods use a local measure to optimally estimate single dimensional functions of projections instead of functions of the full dimensionality of the space, and therefore tend to be less sensitive to the curse of dimensionality problem (Huber, 1985).

One way to reduce the curse of dimensionality is to look for lower dimensional structures (features) by using a localized and smooth objective function that directly measures the importance of the extracted features.

A useful class of features to explore is defined by some linear projections of the high dimensional data. This class is used in projection pursuit methods (PP) originally introduced by Kruskal (1969 1972). Switzer (1970, 1971), and later implemented by Friedman and Tukey (1974). These methods are reviewed in Huber (1985).

It is still difficult to characterize what interesting projections are although, it is easy to point at projections that are uninteresting. To motivate this discussion, consider the following example in which two data clusters lie in a two dimensional space. If we are interested in reducing the dimensionality of the data, and still retaining an indication on the structure, it is best to project the data onto the x axis, even though the variance of the projection to the y axis is larger.

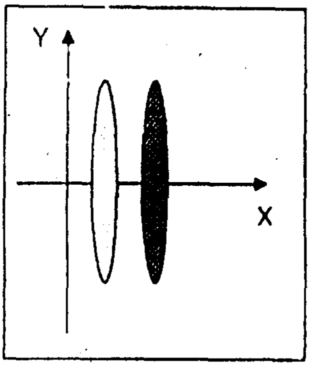


Figure 1: In this dimensionality reduction problem the interesting direction is not the one that maximizes the variance: Two data clusters which can be separated by projecting to the x axis, can not be separated by projecting to the y axis, although the variance in the y axis is larger.

Notice that in the above example, the projection onto the x axis will give a two hump distribution, while the projection onto the y axis will give a normal distribution. It turns out that this is not a coincidence. A statement that has recently been made precise by Diaconis and Freedman (1984) says that for most high-dimensional clouds, most low-dimensional projections are approximately normal. This finding suggests that the important information in the data is conveyed in those directions whose single dimensional projected distribution is far from Gaussian. Friedman (1987) argues that the most computationally attractive measures for deviation from normality (projection indices) are based on polynomial moments. For example, principal components extraction uses a projection index which is based on polynomials of the second moment of the projections (maximizing the projected variance). In some special cases where the data is known in advance to be bi-modal, it is relatively straightforward to define a good projection index (Hinton & Nowlan, 1990).

Despite their computational attractiveness, projection indices based on polynomial moments are not directly applicable, since they very heavily emphasize departure from normality in the tails of the distribution (Huber, 1985). Friedman (1987) addresses this issue by introducing a nonlinear transformation that squashes the projected data from R to $[-1,1]$ using a normal distribution function. We address the problem by applying a sigmoidal squashing function to the projections, and then applying an objective function based on polynomial moments.

2 Feature Extraction using ANN

In this section, the intuitive idea presented above is used to form a statistically plausible objective function whose minimization will find those projections having a single dimensional projected distribution that is far from Gaussian.

We first informally describe the statistical formulation that leads to this objective function (the mathematical details are left to the appendix). Based on statistical decision theory, a neuron is considered as capable of making decisions. The most intuitive decision for a neuron is whether to fire or not for a given input and vector of synaptic weights. To aid the neuron in making the decision, a loss function is attached to each decision, namely a function that measures the loss from making each decision. The neurons task is then to choose the decision that minimizes the loss. Since the loss function depends on the synaptic weights vector in addition to the input vector, it is natural to seek a synaptic weight vector that will minimize the sum of the losses associated with every input, or more precisely, the average loss (also called the risk). The search for such vector, which yields an optimal synaptic weight vector under this formulation, can be viewed as learning or parameter estimation. In those cases where the risk is a smooth function, its minimization can be done using gradient descent.

The ideas presented so far make no specific assumptions regarding the loss function, and it is clear that different loss functions will yield different learning procedures. For example, if the loss function is related to the inverse of the projection variance (including some normalization) then minimizing the risk will yield directions that maximize the variance of the projections, i.e. will find the principal components.

Before presenting our version of the loss function, let us review some necessary notations and assumptions. Consider a neuron with input vector $x = (x_1, \ldots, x_N)$, synaptic weights vector $m = (m_1, \ldots, m_N)$, both in R^N, and activity (in the linear region) $c = x \cdot m$. Define the threshold $\Theta_m = E[(x \cdot m)^2]$, and the functions $\hat{\phi}(c, \Theta_m) = c^2 - \frac{2}{3}c\Theta_m$, $\phi(c, \Theta_m) = c^2 - \frac{4}{3}c\Theta_m$. The ϕ function have been suggested as a biologically plausible synaptic modification function to explain visual cortical plasticity (Bienenstock, Cooper and Munro, 1982). The main features of BCM theory will be discussed below. Θ_m is a dynamic threshold which will be shown later to have an affect on the sign of the synaptic modification. The input x, which is a stochastic process, is assumed to be of Type II φ mixing[1], bounded, and piecewise constant. These assumptions are plausible, since they represent the closest continuous approximation to the usual training algorithms, in which training patterns are presented at random. The φ mixing property allows for some time dependency in the presentation of the training patterns. The assumption are needed for the approximation of the resulting

[1] The φ mixing property specifies the dependency of the future of the process on its past.

deterministic gradient descent by a stochastic one (Intrator, 1990b). For this reason we use a *learning rate* μ that has to decay in time so that this approximation is valid. Note that at this point c represents the linear projection of x onto m, and we seek an optimal projection in some sense.

Our projection index is aimed at finding directions for which the projected distribution is far from Gaussian, more specifically, we are interested in finding clusters in a high dimensional data. Since high dimensional clusters have a multimodal projected distribution, our aim is to find a projection index (loss function) that emphasizes multimodality. For computational efficiency, we would like to base the projection index on polynomial moments of low degree. Using second degree polynomials, one can get measures of the mean and variance of the distribution, which do not give information on multimodality, therefore, higher order polynomials are necessary. Furthermore, the projection index should exhibit the fact that bimodal distribution is already interesting, and any additional mode should make the distribution even more interesting.

With this in mind, consider the following family of loss functions which depend on the synaptic weight vector and on the input x (the derivation based on decision theory appears in the appendix).

$$\begin{aligned} L_m(x) &= -\mu \int_{\Theta_m}^{(x \cdot m)} \hat{\phi}(s, \Theta_m) ds \\ &= -\frac{\mu}{3}\{(x \cdot m)^3 - E[(x \cdot m)^2](x \cdot m)^2\} \end{aligned}$$

The motivation for this loss function can be seen in the following graph, which represents the ϕ function and the associated loss function $L_m(x)$. For simplicity the loss for a fixed threshold Θ_m and synaptic vector m can be written as $L_m(c) = -\frac{\mu}{3}c^2(c - \Theta_m)$, where $c = (x \cdot m)$.

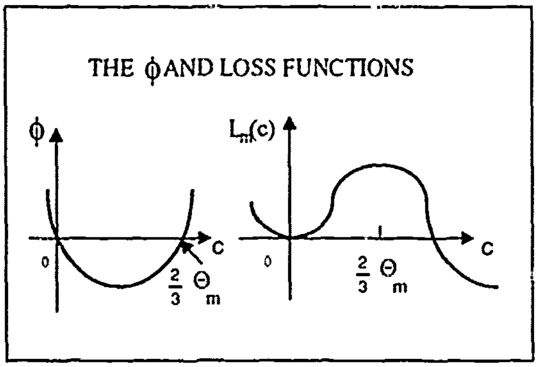


Figure 2: The function ϕ and the loss functions for a fixed m and Θ_m.

The graph of the loss function shows that for any fixed m and Θ_m, the loss is small for a given input x, when either $c = x \cdot m$ is close to zero, or when $x \cdot m$ is larger than $\frac{4}{3}\Theta_m$. Moreover, the loss function remains negative for $(x \cdot m) > \frac{4}{3}\Theta_m$, therefore any kind of distribution at the right hand side of $\frac{4}{3}\Theta_m$ is possible, and the preferred ones are those which are concentrated further from $\frac{4}{3}\Theta_m$.

It remains to show why it is not possible that a minimizer of the average loss will be such that all the mass of the distribution will be concentrated in one of the regions. Roughly speaking, this can not happen because the threshold Θ_m is dynamic and depends on the projections in a nonlinear way, namely, $\Theta_m = E(x \cdot m)^2$. This implies that Θ_m will always move itself to a position such that the distribution will never be concentrated at only one of its sides. This yield that the part of the distribution for $c < \frac{4}{3}\Theta_m$ has high loss, making those distributions in which the distribution for $c < \frac{4}{3}\Theta_m$ has its mode at zero, more plausible.

The fact that the distribution has part of its mass on both sides of $\frac{4}{3}\Theta_m$ makes it already a plausible projection index that seeks multi- modalities. However, this projection index will be more general, if in addition, the loss will be insensitive to outliers, if we allow any projected distribution to be shifted so that the part of the distribution that satisfies $c < \frac{4}{3}\Theta_m$ will have its mode at zero. These points will be discussed below.

The risk (expected value of the loss) is given by:

$$\begin{aligned} R_m = & -\tfrac{\mu}{3}E\{(x \cdot m)^3 - E[(x \cdot m)^2](x \cdot m)^2\} \\ = & -\tfrac{\mu}{3}\{E[(x \cdot m)^3] - E^2[(x \cdot m)^2]\}. \end{aligned}$$

Since the risk is continuously differentiable, its minimization can be achieved via a gradient descent method with respect to m, namely:

$$\begin{aligned} \frac{dm_i}{dt} = -\frac{\partial}{\partial m_i}R_m &= \mu\ \{E[(x \cdot m)^2 x_i] \\ &\quad -\frac{4}{3}E[(x \cdot m)^2]E[(x \cdot m)x_i]\} \\ &= \mu\ E[\phi(x \cdot m, \Theta_m)x_i]. \end{aligned}$$

The resulting differential equations suggest a modified version of the law governing synaptic weight modification in the BCM theory for learning and memory (Bienenstock, Cooper and Munro, 1982). This theory was presented to account for various experimental results in visual cortical plasticity. According to this theory, the synaptic efficacy of active inputs increases when the postsynaptic target is concurrently depolarized beyond a *modification threshold*, Θ_m. However, when the level of postsynaptic activity falls below Θ_m, then the strength of active synapses decreases. An important feature of this theory is that the value of the modification threshold is not fixed, but instead varies as a nonlinear function of the average output of the postsynaptic neuron. This feature provides the stability properties of the model, for positive or mean positive inputs, and is necessary in order to explain, for example, why the low level of postsynaptic activity caused by binocular deprivation does not drive the strengths of all cortical synapses to zero. Mean field theory for a network based on these neurons is presented in (Scofield and Cooper, 1985; Cooper and Scofield, 1988), statistical analysis is given in Intrator (1990c) computer simulations and biological relevance are discussed in (Soul et al., 1986; Bear et al., 1987; Cooper et al., 1987; Bear et al., 1S88; Clothioux, 1990).

Up to this point we have presented an unsupervised (exploratory) method for feature extraction that seeks projections in which the single dimensional distribution is multi-modal, namely we have presented an exploratory projection pursuit method. This method uses polynomial moments as a projection index and therefore suffers from over-sensitivity to outliers (Freidman, 1987). We address this problem by considering a nonlinear neuron in which the neuron's activity is defined to be $c = \sigma(x \cdot m)$, where σ usually represents a smooth sigmoidal function. A more general definition that would allow symmetry breaking of the projected distributions, will provide solution to the second problem raised above, and is still consistent with the statistical formulation is $c = \sigma(x \cdot m - \alpha)$, for an arbitrary threshold α which can be found by using gradient descent as well. For the nonlinear neuron Θ_m is defined to be $\Theta_m = E[\sigma^2(x \cdot m)]$. The loss function is given by:

$$\begin{aligned} L_m(x) &= -\mu \int_{\Theta_m}^{\sigma(x \cdot m)} \hat{\phi}(s, \Theta_m) ds \\ &= -\frac{\mu}{3} \{\sigma^3(x \cdot m) - E[\sigma^2(x \cdot m)]\sigma^2(x \cdot m)\} \end{aligned}$$

The gradient of the risk becomes:

$$\begin{aligned} -\nabla_m R_m &= \mu \{E[\sigma^2(x \cdot m)\sigma' x] \\ &\quad -\frac{4}{3} E[\sigma^2(x \cdot m)] E[\sigma(x \cdot m)\sigma' x]\} \\ &= \mu \, E[\phi\Big(\sigma(x \cdot m), \Theta_m\Big)\sigma' x], \end{aligned}$$

where σ' represents the derivative of σ at the point $(x \cdot m)$. Note that the multiplication by σ' reduces sensitivity to outliers of the differential equation since for outliers σ' is close to zero.

Based on this formulation, a network of Q identical nodes, which receive the same input and inhibit each other, may be constructed in order to extract several features at once. A similar network has been studied by Scofield and Cooper (1985). The activity of neuron k in the network is defined as $c_k = x \cdot m_k$, where m_k is the synaptic weight vector of neuron k. The *inhibited* activity and threshold of the k'th neuron are given by

$$\bar{c}_k = c_k - \eta \sum_{j \neq k} c_j, \qquad \bar{\Theta}_m^k = E[\bar{c}_k^2].$$

Schematic structure of the network is given in Figure 3.

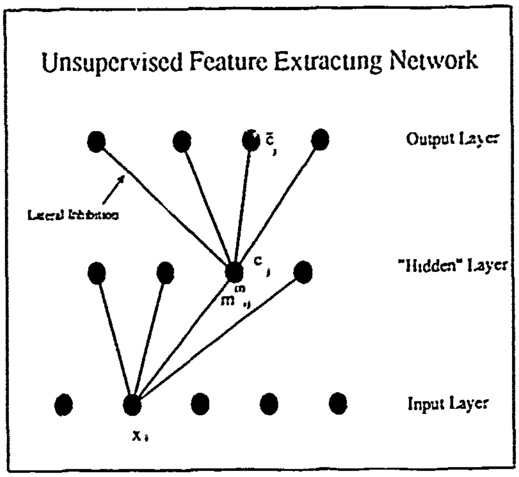


Figure 3: The activity of a nonlinear neuron j is given by $c_j = \sigma(x \cdot m_j)$, the inhibited activity is given by $\bar{c}_j = c_j - \eta \sum_{k \neq j} c_k$.

We omit the derivation of the synaptic modification equations which is similar to the one for a single neuron, and present only the resulting modification equations for a synaptic vector m_k in a lateral inhibition network of nonlinear neurons:

$$\begin{aligned} \dot{m}_k &= -\mu \, E\{\phi(\bar{c}_k, \bar{\Theta}_m^k)\Big(\sigma'(x \cdot m_k) \\ &\quad -\eta \sum_{j \neq k} \sigma'(x \cdot m_j)\Big)x\} \end{aligned}$$

The full derivation can be found in Intrator (1990a). The lateral inhibition network performs a direct search of k-dimensional projections together, which may find a richer structure that a stepwise approach may miss, e.g. see example 14.1 Huber (1985).

3 Comparison with other feature extraction methods

The problem of feature extraction for classification is in some sense easier than that of feature extraction for density or function estimation. This is because the only interesting features in such case are those that distinguish *between* a finite set of classes. The common features, namely those features that do not help in making the distinction between classes are uninteresting, even though they may be very important for data compression, e.g. the self supervised back-propagation network in which the number of hidden units is smaller than the number of input and output units (Elman & Zipser, 1989). The network presented in the previous sections has been shown to seek multimodality in the projected distributions, which translates to clusters in the original space, and therefore to find those directions that make a distinction between different sets in the training data.

In this section we explore the differences in classification performance between a network that performs dimensionality reduction (before the classification) based upon distinguishing features, and a network that performs dimensionality reduction based upon minimization of misclassification error. The performance of the different methods will be compared on a specific classification task: a phoneme classification experiment whose linguistic motivation is described below.

We looked at the six stop consonants [p,k,t,b,g,d] which have been a subject of recent research in evaluating neural networks for phoneme recognition (see review in Lippmann, 1989). These stops posses several common features, but only two distinguishing phonetic features, place of articulation and voicing (table 1) (see Blumstein & Lieberman for a review and related references on phonetic feature theory).

	Place of Articulation		
	Velar	Alveolar	Labial
Voiced	[g]	[d]	[b]
Unvoiced	[k]	[t]	[p]

Table 1: The two distinguishing phonetic features between the six stop consonants.

The Linguistic information in the table suggests the following experiment: A network is to be trained to reduce dimensionality from the unvoiced stops [p,k,t]. In order to reduce variability in the data, only a single speaker and a single vowel context is used. Therefore, the only distinguishing features in the training data are associated with place of articulation, since the features that are speaker dependent, voicing dependent, or context dependent belong to the set of common features in the training data. A dimensionality reduction method that concentrates mainly on distinguishing features should find only the features associated with place of articulation, and therefore become insensitive to voicing dependent and speaker dependent features, which are the common features in the training data. This can easily be tested by evaluating the performance on place of articulation classification of voiced stops and data from other speakers.

For comparison, we have attempted to extract features using three methods. principal components, backpropagation, and the above unsupervised network, all trained and tested on the same data. In backpropagation, the only supervised method, the place of articulation phonetic feature was used as a supervisor.

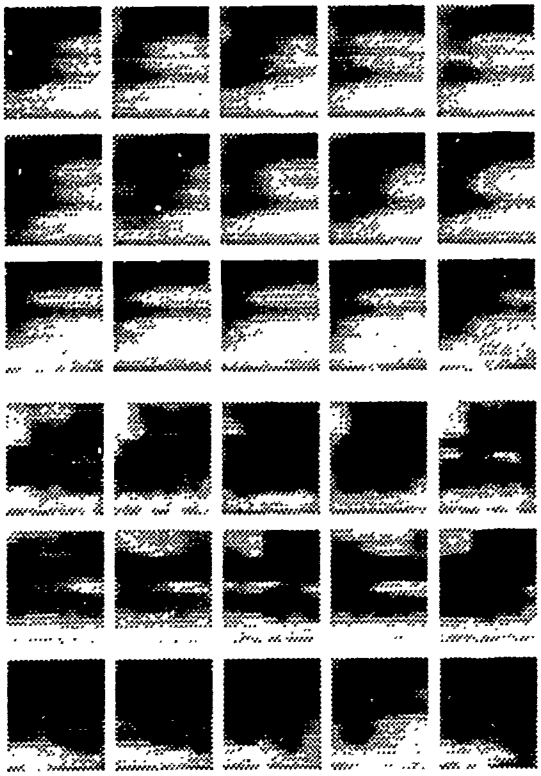


Figure 4: The six stop consonants followed by the vowel [a] for male speaker BSS. Their order from bottom to top is [pa] [ka] [ta] [ba] [ga] [da]. Each token is represented by a 20 consecutive time windows of 32msec with 30msec overlap. In each time frame a set of 22 energy levels in Zwicker critical band filters are computed. Notice the significant difference between the voiced and the unvoiced images.

The speech data consists of 20 consecutive time windows of 32msec with 30mSec overlap, aligned to the beginning of the burst. In each time window, a set of 22 energy levels is computed. These energy levels correspond to Zwicker critical band filters (Zwicker, 1961).

The consonant-vowel (CV) pairs were pronounced in isolation by native American speakers (two male BSS and LTN, and one female JES.) Five tokens of each of the CV pairs used for training are presented in Figure 4. Additional details on biological motivation for the preprocessing, and linguistic motivation related to child language acquisition can be found in Seebach (1990), and Seebach and Intrator (1990).

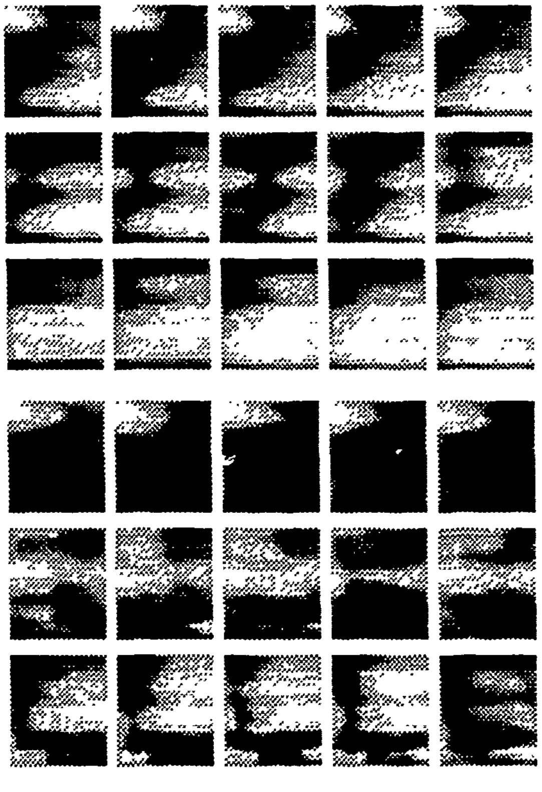


Figure 5: The six stop consonants followed by the vowel [a] for female speaker JES. Their order from bottom to top is [pa] [ka] [ta] [ba] [ga] [da]. Pre-processing is the same as above. Notice that the same burst that appear in [ta] is clear in the [da] as well.

Figure 5 presents five tokens of each of the CV pairs pronounced by the female speaker JES. The classification results obtained using BCM network and principal components methods, were better on this speaker, than on those obtained when testing the performance on the speaker that was used in the training. This is due to the very 'clean' sound that corresponds closely to the acoustic features that are known (Blumstein & Lieberman, 1984) to exist in these sounds. For example, this was the only speaker out of several that we tested, in which the high frequency burst (top left corner) is clear for the voiced stop as it is clear for the unvoiced stops.

The unsupervised feature extraction/classification method is presented in Figure 6. Similar approach using the RCE and back-propagation network have been carried out by several researchers (Rimey et al., 1986, Reilly et al., 1987, 1988; Zemani et al., 1989), and using the unsupervised charge clustering network by Scofield (1988)

Five features/directions were extracted from the 440 dimensional preprocessed speech vectors. These features were the activation of five neurons in the unsupervised network, the five principal components in the PC method, and the five hidden unit activations in back-propagation. The extracted features were used to train a k-NN classifier (with $k = 3$) to classify place of articulation. Although the three dimensionality reduction methods were trained only with the unvoiced tokens of a single speaker, the five dimensional k-NN classifier was trained on voiced and unvoiced data from the other speakers as well.

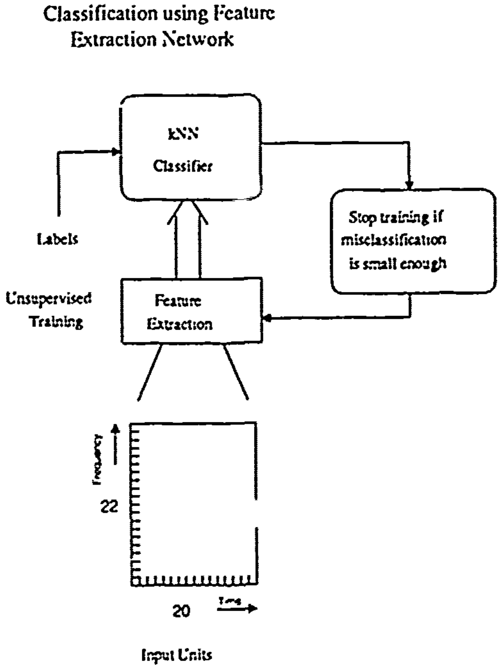

Classification using Feature Extraction Network
kNN Classifier
Labels
Stop training if misclassification is small enough
Unsupervised Training
Feature Extraction
Frequency
22
20
Time
Input Units

Figure 6: Low dimensional k-NN classifier is trained on the features extracted from the high dimensional data. Training of the feature extraction network stops, when misclassification rate drops below a predetermined threshold on either the same training data (cross validatory test) or on a different testing data.

The classification results are summarized in table 2. Several observations can be made from the results; First, the principal components dimensionality reduction is clearly not sufficient in discovering structure for this kind of data, suggesting that the structure is highly non linear. Second, the back-propagation network is doing well in finding structure useful for classification of the trained data, but this structure does not concentrates on distinctive features solely, it also contains speaker dependent and voicing dependent features, and therefore has degraded classification performance when tested on voiced data, or data from other speakers. This can also be viewed as a generalization problem, in which case one can say that the network is overfitting to the training data. Third, classification results using the BCM network for dimensionality reduction suggest that for this specific task, structure that is less sensitive to voicing features can be extracted, even though the network was trained on the unvoiced data only and voicing has significant effects on the speech signal itself.

Place of Articulation Classification			
	P-C	B-P	BCM
BSS /p,k,t/	66.0	100.0	98.6
BSS /b,g,d/	57.4	73.3	94.0
LTN /p,k,t/	60.0	95.8	98.9
LTN /b,g,d/	46.6	66.7	90.0
JES (Both)	70.6	83.7	99.4

Table 2: Percentage of correct classification of place of articulation in voiced and unvoiced stops using principal components, back-propagation, and the BCM network. Training for dimensionality reduction was done on unvoiced stops of male speaker BSS in all three experiments. LTN is a male speaker aswell. The result in the last column represents testing on both the voiced and unvoiced stops of a female speaker (JES). The results represent an average result of several trials, which differ only in the initial conditions of the networks.

4 Discussion

It has been shown that the BCM neuron is capable of effectively discovering nonlinear structures in high dimensional spaces. When compared with other projection indices, the highlights of the presented method are *i*) the projection index concentrates on directions where the separability property as well as the non-normality of the data is large, thus giving rise to better classification properties; *ii*) the degree of correlation between the directions (features) extracted by the network can be regulated via the global inhibition, allowing some tuning of the network to different types of data for optimal results; *iii*) the pursuit is done on all the directions at once thus leading to the capability of finding more interesting structures than methods that find only one projection direction at a time.

Regarding the speech experiment, the network and its training paradigm present a different approach to speaker independent speech recognition. In this approach the speaker variability problem is addressed by training a network that concentrates mainly on the distinguishing features, on a single speaker, as opposed to training a network that concentrates on both the distinguishing and common features, on multi-speaker data.

Acknowledgements

I wish to thank Leon N Cooper for suggesting the problem and for providing many helpful hints and insights. Geoff Hinton made invaluable comments that made this manuscript much more readable. The application of BCM to speech is discussed in more detail in Seebach (1990) and in a forthcoming article (Seebach and Intrator, 1990). The back-propagation experiments were done by Charles M. Bachmann.

Research was supported by the Office of Naval Research, the National Science Foundation, and the Army Research Office.

References

Bellman, R. E. (1961) Adaptive Control Processes, Princton, NJ, Princton University Press.

Bienenstock, E. L. (1980) A theory of the development of neuronal selectivity. Doctoral dissertation, Brown University, Providence, RI

Bienenstock, E. L., L. N Cooper, and P.W. Munro (1982) Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. *J.Neurosci.* 2:32-48

Bear, M. F., L. N Cooper, and F. F. Ebner (1987) A Physiological Basis for a Theory of Synapse Modification. *Science* 237:42-48

Bear, M. F., L. N Cooper, and F. F. Ebner (1988) Synaptic Modification Model of Learning and Memory. In *Encyclopedia of Neuroscience.*

Cooper, L.N. and C. L. Scofield (1988) Mean-field theory of a neural network. *Proc. Natl. Acad. Sci. USA* 85:1973-1977

Devijver P. A., and J. Kittler (1982) Pattern Recognition: A Statistical Approach. Prentice Hall London

Diaconis, P, and D. Freedman (1984) Asymptotics of Graphical Projection Pursuit. *The Annals of Statistics*, 12 793-815.

Friedman, J. H. and J. W. Tukey (1974) A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.* C-23:881-889

Friedman, J. H. and W. Stuetzle (1981) Projection pursuit regression. *J. Amer. Statist. Assoc.* 76:817-

823

Friedman, J. H., W. Stuetzle and A. Schroeder (1984) Projection pursuit density estimation. *J. Amer. Statist. Assoc.* 79:599-608

Friedman, J.H. (1987) Exploratory Projection Pursuit. *Journal of the American Statistical Association* 82-397:249-266

Hall, P. (1988) Estimating the Direction in which Data set is Most Interesting. *Probab. Theory Rel. Fields* 80, 51-78

Hall, P. (1989) On Polynomial-Based Projection Indices for Exploratory Projection Pursuit. *The Annals of Statistics*, 17, 589-605.

Hinton, G.E. and S.J. Nowlan (1990) *Neural Computation*, (in press).

Huber P. J. (1981) Projection Pursuit. Research Report PJH-6, Harvard University, Dept. of Statistics.

Huber P. J. (1985) Projection Pursuit. *The Annal. of Stat.* 13:435-475

Intrator N. (1990) A Neural Network For Feature Extraction. In D. S. Touretzky (ed.), *Advances in Neural Information Processing Systems 2.* San Mateo, CA. Morgan Kaufmann.

Intrator N. (1990) An Averaging Result for Random Differential Equations. Technical report CNS-54, Brown University.

Intrator N. (1990) Feature Extraction using an Exploratory Projection Pursuit Neural Network. Ph.D. Dissertation Brown University.

Jones, M. C. (1983) The Projection Pursuit Algorithm for Exploratory Data Analysis. Unpublished Ph.D. dissertation, University of Bath, School of Mathematics.

Kruskal, J.B. (1969) Toward a practical method which helps uncover the structure of the set of multivariate observations by finding the linear transformation which optimizes a new 'index of condensation'. In *Statistical Computation.* (R.C. Milton and J. A. Nelder, eds.)

Kruskal, J.B. (1972) Linear Transformation of multivariate data to reveal clustering. In *Multidimensional Scaling: Theory and Application in the Behavioral Sciences, I, Theory.* Seminar Press, New York and London.

Linsker R. (1988) Self-Organization in a Perceptual Network. *IEEE. Computer* March 88:105-117

Lippmann, R. P. (1989) Review of Neural Networks for Speech Recognition. *Neural Computation* 1, 1-38.

von der Malsburg, C. (1973) Self-organization of orientation sensitivity cells in the striate cortex. *Kybernetik* 14:85-100

Reilly, D.L., C.L. Scofield, C. Elbaum and L. N Cooper (1987) Learning system architectures composed of multiple learning modules *Proc. First International Conference on Neural Networks.*

Reilly, D.L., C.L. Scofield, L. N Cooper and C. Elbaum (1988) GENSEP: a multiple neural network with modifiable network topology. *INNS Conference on Neural Networks.*

Rimey, R., P. Gouin, C.L. Scofield and D.L. Reilly (1986) Real-time 3-D object classification using a learning system. *SPIE, Intelligent Robots and Computer Vision*, October 1986.

Saul, A. and E. E. Clothiaux, 1986) Modeling and Simulation II: Simulation of a Model for Development of Visual Cortical specificity. *J. of Electrophysiological Techniques*, 13:279-306

Scofield, C. L. and L. N Cooper (1985) Development and properties of neural networks. *Contemp. Phys.* 26:125-145

Scofield, C. L. (1988) Unsupervised learning in the N dimensional Coulomb net. *Abstracts in the first annual international nerual network soc. meeting.* Vol. 1 Supl. 1, 1988 p.129.

D. P. Morgan, C.L. Scofield. T.M. Lorenzo, E.C Real and D.P Loconto (1990) A key word spotter which incorporates neural network for secondary processing. *Proc. ICCASSP*, Alberquerque New Mexico 1990 pp.113-116.

Seebach, B.S. (1990) Evidence for the Development of Phonetic Property Detectors in a Neural Net without Innate Knowledge of Linguistic Structure. Ph.D. Dissertation Brown University.

Seebach, B.S. and N. Intrator (1990) To appear.

Switzer, P. (1970) Numerical classification. IN *Geostatistics.* Plenum, New York.

Switzer, P. and R. M. Wright (1971) Numerical classification applied to certain Jamaican eocene nummulitids. *Math. Geol.* 3:297-311

Zemani P.D., D.P. Morgan, D.L. Reilly, C.L. Scofield et al. (1989) Experiments in discrete utterance recognition using neural network. *Proc. of the Boston ASSP mini-conference, Weston Massachusetts* May, 1989

Zwicker E. (1961) Subdivision of the audible frequency range into critical bands (Frequenzgruppen) *Journal of the Acoustical Society of America* 33:248

Mathematical Appendix

In this section we develop the statistical formulation that yields the loss function presented in section 2.

Let $(\Omega, \mathcal{F}_\Omega, P)$ be a probability space on the space of inputs Ω with probability law P. Let $\mathcal{A} = \{0, 1\}$ be a decision space, in the case of a single neuron a zero decision means that the neuron does not fire. Let m be a vector of parameters such as the one described above, and assume that it lies in a compact space B^M. This parameter space defines a family of loss functions, $\{L_m\}_{m \in B^M}$, $L_m : \Omega \times \mathcal{A} \mapsto R$. Let $\mathcal{D}$ be the space of all decision rules. The empirical risk (average loss) $R_m : \mathcal{D} \mapsto R$, is given by:

$$R_m(\delta) = \sum_{i=1}^{n} P(x^{(i)}) L_m(x^{(i)}, \delta(x^{(i)}))$$

For a fixed m, the optimal decision δ_m is chosen so that:

$$R_m(\delta_m) = \min_{\delta \in \mathcal{D}} \{R_m(\delta)\}$$

Since this minimization takes place over a finite set, the minimizer exists. In particular, for a given $x^{(i)}$ the decision $\delta_m(x^{(i)})$ is chosen so that

$$L_m(x^{(i)}, \delta_m(x^{(i)})) \leq L_m(x^{(i)}, 1 - \delta_m(x^{(i)})).$$

At this point $R_m(\delta_m)$ is a risk function that depends only on the vector of parameters m, and assuming R_m is bounded, it is natural to seek a parameter $\tilde{m}$ that minimizes R_m, namely,

$$R_{\tilde{m}}(\delta_{\tilde{m}}) = \min_{m \in B^M} \{R_m(\delta_m)\}.$$

The minimum with respect to m exists since B^M is compact, and R_m is bounded. When m represents a vector in R^N, R_m can be viewed as a projection index.

Based on the above, let m, the synaptic weight vector, be the parameter to be esimated, and consider the following family of loss functions. The loss functions depend on the cell's decision whether to fire or not, and they represent the intuitive idea that the neuron will fire when its activity is greater than some threshold, and will not otherwise. We denote the firing of the neuron by $a = 1$. Define $K = -\mu \int_0^{\frac{2}{3}\Theta_m} \hat{\phi}(s, \Theta_m) ds$. The loss function for a decision to fire is given by:

$$L_m(x, 1) = \begin{cases} -\mu \int_{\Theta_m}^{(x \cdot m)} \hat{\phi}(s, \Theta_m) ds, & (x \cdot m) \geq \Theta_m \\ K - \mu \int_{\Theta_m}^{(x \cdot m)} \hat{\phi}(s, \Theta_m) ds, & (x \cdot m) < \Theta_m, \end{cases}$$

and for the decision not to fire by:

$$L_m(x, 0) = \begin{cases} -\mu \int_{\Theta_m}^{(x \cdot m)} \hat{\phi}(s, \Theta_m) ds, & (x \cdot m) \leq \Theta_m \\ K - \mu \int_{\Theta_m}^{(x \cdot m)} \hat{\phi}(s, \Theta_m) ds, & (x \cdot m) > \Theta_m. \end{cases}$$

It follows from the definition of L_m and from the definition of δ_m that

$$\begin{aligned} L_m(x, \delta_m) &= -\mu \int_{\Theta_m}^{(x \cdot m)} \hat{\phi}(s, \Theta_m) ds \\ &= -\frac{\mu}{3}\{(x \cdot m)^3 - E[(x \cdot m)^2](x \cdot m)^2\}. \end{aligned}$$

We can write $L_m(x)$ instead of $L_m(x, \delta_m)$ when there is no confusion.

The risk is given by:

$$R_\theta(\delta_\theta) = -\frac{\mu}{3}\{E[(x \cdot m)^3] - E^2[(x \cdot m)^2]\}.$$

Since the risk is continuously differentiable, its minimization can be done via the gradient descent method with respect to m, namely:

$$\frac{\partial m_i}{\partial t} = -\frac{\partial}{\partial m_i} R_\theta(\delta_\theta) = \mu \; E[\phi(x \cdot m, \Theta_m) x_i].$$

Notice that the resulting equation represents an averaged deterministic equation of the stochastic BCM modification equations. It turns out that under suitable conditions on the mixing of the input x and the global function μ, this equation is a good approximation of its stochastic version (Intrator, 1990b), namely:

$$\frac{\partial r}{\partial i} \qquad \mu \; \phi(x \cdot m, \Theta_m) x_i.$$

Motor Control for Speech Skills: a Connectionist Approach

Rafael Laboissière, Jean-Luc Schwartz, and Gérard Bailly
Institut de la Communication Parlée
I.N.P.G. - U.R.A. C.N.R.S. N° 368
46, av. Félix Viallet
F-38.031 Grenoble CEDEX France
e-mail: rafael@icp.imag.fr

Abstract

The acoustic-phonetic decoding of speech is particularly difficult because of the signal variability with phonetic context (coarticulation mechanisms, linked with economy principles which rule the functioning of the speech motor system). We try here to apply the basic principles introduced by Jordan – driving motor systems with excess degrees of freedom by sequential networks – in modeling coarticulation. We use an articulatory model – the "forward model" – elaborated by Maeda, allowing the passage from five articulatory parameters (jaw, lips, tongue body, tongue dorsum, tongue tip) to formants (acoustic resonances) in the speech signal. Our work proposes an analytic progressive implementation of Jordan's concepts: we attempt to isolate in the sequential network various functionalities, and to see in each case what is necessary, and what is problematic. Finally, we show that such a structure is indeed able to predict plausible coarticulation effects.

1 INTRODUCTION

Acoustic-phonetic decoding belongs to the classical set of problems where a bridge has to be designed between the physical world and the mental world – or between signals and symbols, or between continuous and discrete objects. However, the passage of this gap is particularly difficult in the case of speech, because of the specificity and the complexity of signal variability, that has to be in some way "filtered out" in the search for invariance. A key problem is variability due to context: a phoneme is not realized in the signal independantly of the neighbour phonemes, and the contextual effects – coarticulation in general – can expend on quite long durations.

This is of course linked to the fact that the speech acoustic signal is a temporal signal, which means that time is crucial in speech processing: hence the success of statistical models where time relationships are taken into account, such as HMMs or, more recently, the MLP-based Time-Delayed Neural Networks.

More deeply, the problem is that the speech signal is a *biological* signal, *produced* by a system acting with its own rules, one major rule being the respect of economy principles (smoothness of gestures, whatever the content of this concept).

In this respect, Michael Jordan's work on sequential networks and their use to pilot motor systems with excess degrees of freedom thanks to constraints specified in very general terms is quite influential. The basic aim of the present work is to apply these principles to modeling of coarticulation phenomena in speech. More generally, it will be able to introduce general constraints within the framework of a connectionist control system driving an articulatory model, in order to produce increasingly complex speech sequences.

2 GENERAL PRINCIPLES

The general framework of our present research direction is depicted in Fig.1. Through the solide lines in this figure, we have the flow of commands and signals during speech production. The Mental Targets are some kind of abstract representation of the utterance (or parts of it). They specificy the phonological task for the Motor Planning feedforward controller which, in turn, is responsible for the generation of the complex spatio-temporal pattern of muscle activation. These commands will ultimately produce an acoustical signal.

The dotted paths in Fig.1 stand for the flow of information during the learning phase. We hypothesize the existence of a Forward Model somewhere in the Central Nervous System. This Forward Model is an internal mapping between the articulator activations and the final acoustical results, taking into account both kinematics and dynamics of the physical system. The learning of this mental representation is done through babbling, thanks to the auditory feedback.

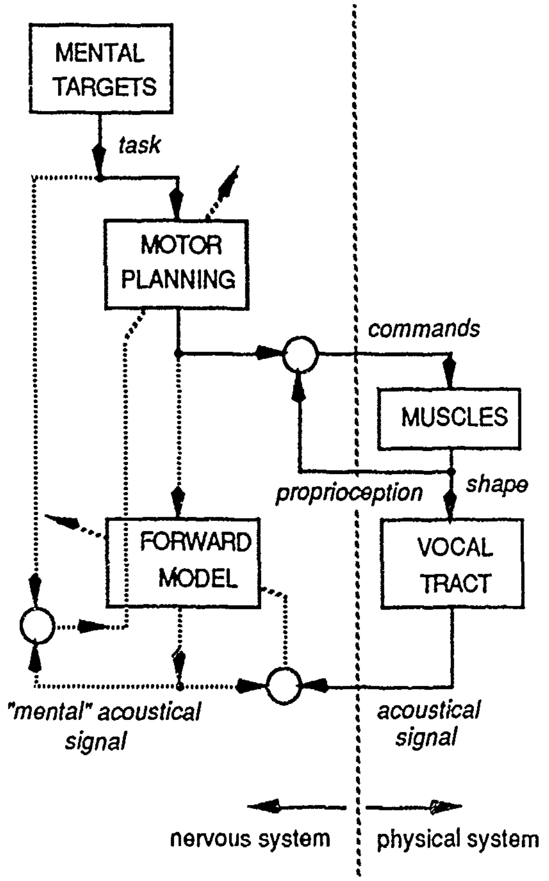


Figure 1: General Framework

We assume, in a first approximation, that this feedback through hearing is too slow to be used in the control of ongoing utterances. However, the auditory information will be helpful for the latter learning of the Motor Planning. Indeed, once the Forward Model achieves learning, the essential characteristics of the physical system are modelled. This has the additional advantage that it is possible to infer the correction of errors in the Motor Planning based on the knowledge of how far the actual output (speech signal) is from the specifications (Mental Targets). This is essentially the backpropagation paradigm.

The problem of learning the Motor Planning is ill-posed, due to the many-to-one mapping accomplished by the Forward Model. It can be regularized by the introduction of constraints which enable the selection of one solution. The constraints precisely consist of economy costs which orientate the learning of Motor Planning towards smooth trajectories.

Although proprioceptive feedback appears to play an important role in articulatory compensation during external perturbation, we do not take it into account in this paper. Our prime interest is just the study of coordinative structures invoked in the realization of phonological segments. These coordinative structures are pre-stored stereotype· of complex spatio-temporal muscle activities in the vocal apparatus. In the action theory, these two kinds of movement control, namely *pre-programmed trajectories* and *afferent feedback control*, are concurrently used to explain how animals produce motion.

Three well-known connectionist principles induce the choice of the system architecture and the learning procedures to be used. First of all, the paradigm of learning-by-examples is useful to train the Forward Model and the Motor Planning. Second, the principle of error back-propagation allows the computation of inverse kinematics and inverse mechanics, needed for solving the indeterminate excess degrees of freedom problem. Hence we use a class of back-propagation neural networks as the basic architecture. The total error function will involve both configurational terms and additional economy terms for selection of smooth trajectories. Finally, recurrent neural network architectures will be helpful to obtain the proper treatment of time in the sequence generation and in generalization.

The basic principles of this approach for solving motor control problems have been introduced by Jordan (1988, 1989, 1990). The present work intends to apply these principles to speech production. Nonetheless, we propose a slightly different methodological program, somewhat more analytical. We shall try to break down the whole problem in a number of more tractable ones. Besides the straightforward advantages of this approach, we will be able to test the limits and potentialities of each basic assumption of our work. In the following sections we deal with each of these sub-problems separately:

- In Section 3, we quickly present the model of the vocal tract that we use in this study and its approximation by neural networks for realizing the Forward Model, and we show how smooth gestures can be learnt on this model, by optimizing cost functions on trajectories of articulatory parameters with a back-propagation algorithm;
- In Section 4, we show how time could be controlled in the system, by describing an original experiment on learning and generalization of oscillators at controlled frequencies;
- In Section 5, we show how a simple non-recurrent MLP can learn to transform a stereotypic oscillation trajectory into a set of complex articulatory and acoustic gestures – conversion of simple into complex movements – depending on an input coding the sequence;
- Finally, we show in Section 6 how a complete sequential network driving the articulatory model can indeed learn to produce vowel-to-vowel sequences and exhibit plausible coarticulation effects.

3 FROM ACOUSTIC SIGNAL TO ARTICULATORY GESTURES: THE ROLE OF CONSTRAINTS

Like in robotics, the inversion in speech (i.e. recovery of the articulatory gesture that generated a given acoustic trajectory) is an ill-posed problem. Indeed, Atal *et al.* (1978) showed the many-to-one property of the articulatory-to-acoustic transformation, through the concept of "articulatory fiber", a path in the space of vocal tract configurations along which the acoustic result (formants) is constant. Notice however that some key characteristics of all vocal tract shapes within a fiber can be more or less invariant, namely position and area of the main constriction, and lip area (Boë & Perrier, 1988). Larar *et al.* (1988) used dynamic programming techniques to solve the inversion problem. They added costs to transitions in the articulatory space, allowing the selection of an "optimal" path in this space. In this section we describe an error back-propagation approach to the inversion task and we show how general temporal constraints can be introduced in the formalism of neural network training.

3.1 FORWARD MODEL

One way to back-propagate errors in the speech inversion task is to find an analytic description of the articulatory-to-acoustic transform, in order to calculate derivatives. It is however impossible to do it directly, because the second step of this transform (see below) is not defined by an input-output function, but by solving complex systems of non-linear equations. Instead of using a code-book as complete as possible to describe the transform extensively, as Larar *et al.* do in their simulations, it is somewhat more straightforward to use an analytic interpolation as can be learnt by classic MLPs (the typical association problem). This is the forward model introduced in Section 2.

The input-output pairs used to train the forward model come from a two-step procedure. First, vocal-tract shapes are computed by a realistic model developed by Maeda (1979). This model was designed from a knowledge-driven principal-component analysis of X-ray sagittal cuts of a given speaker. It realizes the passage from five articulatory parameters (jaw, tongue body, tongue dorsum, tongue tip, lips closure/protrusion) to the whole sagittal cut. The second step consists in the computation, by means of an electrical-line analogy, of the acoustic transfer function of a multi-tube model whose area function is computed from the sagittal cut (Perrier *et al.*, 1990). The first three resonance frequencies (formants) of the transfer function are then estimated through all-pole modeling. Hence, the articulatory-to-acoustic transform to be learned associates five articulatory parameters (commands of Maeda's model) to three acoustic formants.

With this whole model, we prepared a 1500-large training database covering all the eleven French vowels and their neighbourhood in the three-formant space, and we attempted to design a neural-like forward model fitting these data. Previous work (Bailly *et al.*, 1990a) showed that a polynomial approximation can yield quite a good fitting, and a second-order polynomial represents the best trade-off between simplicity, stability and accuracy. The forward model we used in this study contains three output units directly receiving signals from the five input units, the net input of these output units being a linear combination of the first and second-order terms of the input signals ("quadratic units"). This is a special case of the sigma-pi units introduced by Rumelhart *et al.* (1986).

3.2. TECHNICAL IMPLEMENTATION OF CONSTRAINTS

First of all, it is important to note that the forward model we use is purely geometrical and that the concept of *force* appears nowhere. Therefore, we are at the present state unable to apply constraints related to the *dynamics* of the speech articulators, restricting ourselves to a class of *kinematic* constraints. They can be either *spatial* (targets in distal coordinates, rest position attractors), or *temporal* (minimal kinetic energy, minimal jerk, among others).

A connectionist model suitable for optimizing articulatory trajectories is shown in Fig.2. The trained forward model is included in the top of the network and its weights do not change during optimization. To simulate an articulatory trajectory of length N, we connect each articulatory unit to an input layer with N units which are sequentially activated, taking the value one when active, and zero otherwise (for sake of clarity, we display the connections for only one articulatory unit in Fig.2). Each unit performs a linear combination of its inputs. Hence, the weights of connections, which are optimized by back-propagation, represent the articulatory trajectory itself.

The first class of error terms to be back-propagated concerns the configurational task to be accomplished at the output of the forward model. An example of acoustic specification through time in the three-formant space is depicted on Fig.3. The horizontal straight lines stand for precise target value specification, representing steady portions of the vowels. Filled rectangles appear at each vowel transition and represent intervals, inside which error is zero and outside which error grows in a quadratic way.

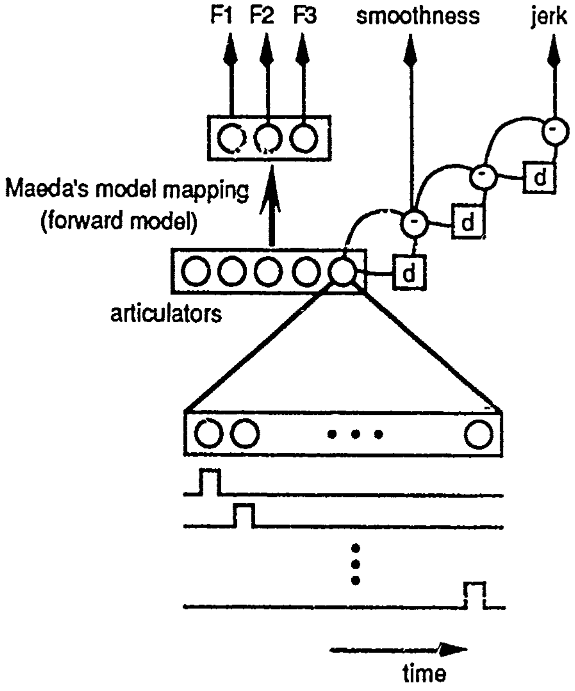


Figure 2: Network for Optimization of Articulatory Trajectories

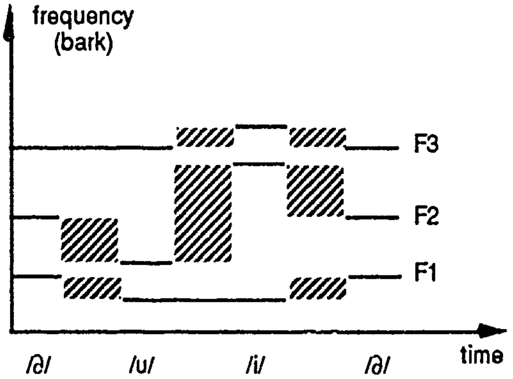


Figure 3: Configurational Specifications
Phonological sequence /əuiə/

Due to excess degrees of freedom in the forward transformation, optimization carried out with only configurational specifications can yield an infinity of solutions. Temporal constraints help us find a "good" one. The optimizing criterion is the minimization of some physical quantities over the trajectories in the articulatory space, like kinetic energy – integral of the squared velocity over the whole trajectory – or jerk – idem, replacing velocity by time derivative of the acceleration. The ultimate result of the optimization process will be "smooth" articulatory trajectories producing an acoustic result fitting the configurational specifications.

The way we can deal with temporal constraints in the back-propagation formalism is also depicted in Fig.2. In order to obtain the time derivative of an articulator displacement we simply connect a fixed linear unit to the corresponding articulator unit, the first one computing the difference between two consecutive values of the second. Higher-order time derivatives can also be obtained by means of the repetition of this procedure. These units are considered as outputs and errors are back-propagated through them. For instance, minimization of the jerk is realized by specifying a target value equal to zero to the jerk units. Temporal constraints are indistinguishable from configurational ones, in the point of view of the back-propagation technique.

The two kinds of constraints, namely configurational and temporal, are inherently contradictory, as noticed by Jordan (1990). Because the temporal constraints must not prevent the accomplishment of the configurational task, we must weaken the influence of the smoothness costs as learning progresses. One way to do this is to multiply each smoothness cost by a coefficient proportional to the configurational cost. The total cost is a weighted sum of several costs and, as the configurational cost vanishes, the total error goes down to zero. Fig.4 shows the evolution of these different costs during learning. Notice the long and slow training phase, after which the total error suddenly goes to zero and smoothness is set to a final, possibly optimal, value.

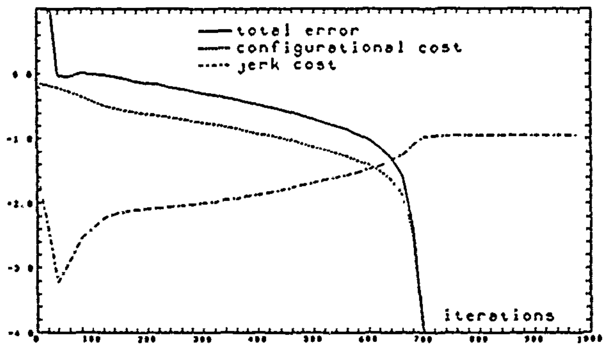


Figure 4: Typical Cost Evolution
During Learning

3.3 TEMPORAL CONSTRAINTS AND COARTICULATED TRAJECTORIES

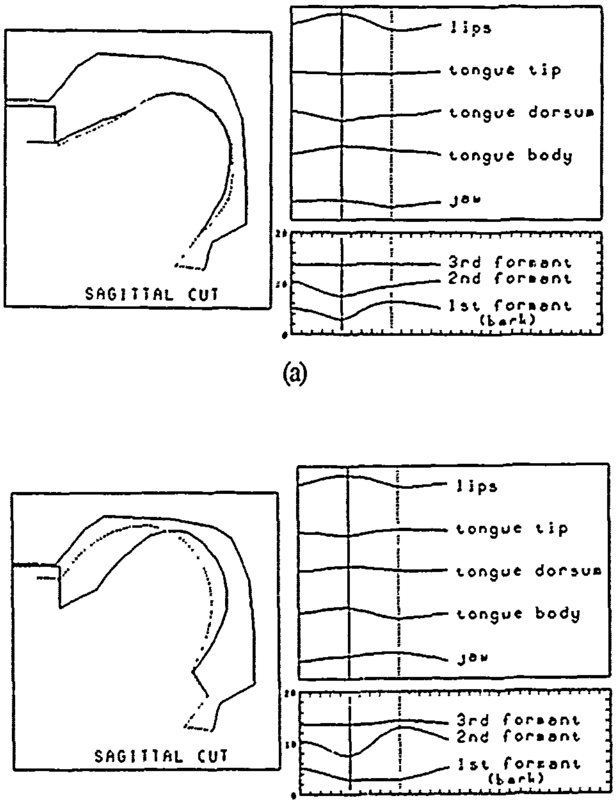
All the simulations presented in this paper concern acoustic trajectories going from a neutral configuration corresponding to the vowel /ə/ (schwa), towards the back high rounded /u/, then either the front high unrounded /i/ or the low /a/, and back to the schwa.

Using a temporal constraint based on kinetic energy only generally leads to trajectories made of portions of movement with a constant speed from one target to the next, and sudden changes of direction between these portions. This is quite understandable, since it is easy to demonstrate that the trajectory connecting two points and minimizing the kinetic energy is indeed a constant-speed one. However, our complete trajectories are then characterized by infinite accelerations when a target is reached, in order to travel to the next one, and this is of course not acceptable.

Hence, we used a blend of jerk and kinetic energy temporal constraints. This led to the results presented in Fig.5. On the left half of Figs. 5a and 5b, vocal tract shapes are displayed. The upper solid line of the sagittal cut represents the motionless upper boundary of the vocal tract. The two lower lines (solid and dotted) stand for the lower boundary of the vocal tract sampled at two different instants. These sample instants are showed with corresponding line types on the right half of Figs. 5a and 5b where the evolution of the articulatory (top) and acoustic (bottom) parameters are depicted.

During the production of the vowel /u/, characterized by a strong closure/protrusion of the lips, the tongue anticipates – as much as allowed by /u/ – the following vowel, either /a/ or /i/, but the acoustic configuration reached for /u/ is the same in both cases. This shows that general temporal constraints can be useful to explain coarticulation, at least in our limited geometrical framework.

In conclusion, it is worth saying that although this network can produce only a single sequence, it provides a valuable way to test some hypotheses about constraints and coarticulated articulatory trajectories. On the other hand, we feel that direct approaches to optimize and store trajectories at the same time in connectionist networks (as is done in Sections 5 and 6) do not completely clarify the effects of smoothness constraints. Indeed, the architecture of the network used to store learned trajectories is itself a constraint to the set of possible solutions we can find. This final remark validates the analytic approach we are pursuing.



(a)

(b)

Figure 5: Simulated
Articulatory/Acoustic Trajectories
(a) Sequence /əuaə/ – (b) Sequence /əuiə/

4 GENERATION OF CONTROLLED OSCILLATORS BY SEQUENTIAL NETWORKS

4.1 THE NEED FOR OSCILLATORS

A key problem in Jordan's sequential networks lies in the control of time. Indeed, Jordan's single proposal in this respect is just an over- or under- sampling of a learned trajectory. This is of course not at all satisfying, the more so since modifications of the time-scale of a given gesture in speech production lead to non-linear modifications of the articulatory-acoustic trajectory because of mechanical constraints (see the so-called "non-reached targets" as first described by Lindblom, 1963) and even to non-linear reorganizations of the planning itself, in order to achieve perceptual targets (Abry *et al.*, 1990).

Without addressing this problem in general, we propose that a first step lies in the possibility to explicitely control time during learning, and that oscillators controlled in frequency could provide a good basis for so doing. This strategy may be confirmed on physiological grounds by the ability of neurons in the inferior olive and the thalamus to oscillate solely or through inter-neuron coupling at a frequency which may be controlled by transmitters (Llinas, 1989; Adams & Benson, 1989).

4.2 THE VCO EXPERIMENT

In this section, we show how to train a recurrent neural network to behave like a Voltage Controlled Oscillator (VCO). Basically, a VCO is a system that can oscillate at different frequencies tuned by an input signal. This input signal is assumed to be held constant during the oscillations. The aim here is to test the capabilities of this kind of network to learn and generalize complex temporal behaviors.

The structure of our network is depicted in Fig.6. The hidden layers each possess three sigmoidal units. The arrows in the figure represent full-connectivity among units in different layers. There are no connections between two units belonging to the same layer. The links going from the righthand hidden layer to the other one pass through a delay, set to one time step. This architecture allows the mapping between states of the running network (taken by definition as the activations on the lefthand hidden layer) to be equivalent to the mapping of a two-layer perceptron.

The five units on the input layer are clamped with a coarse-coded value representing the frequency at which the network must oscillate. The smearing function is triangular. The output layer is made of only one unit which is also a sigmoidal cell with saturation values at -1 and +1. Notice the two-layer passage from the input units and state units to the output unit.

The chosen architecture precludes the use of simplified learning procedures like teacher-forcing since the desired activation values of the hidden units are not *a priori* known. We used a version of the back-propagation through time (BPTT) algorithm to train this network (Rumelhart *et al.*, 1986; Williams & Zipser, 1990).

The simulation results are shown in Fig.7. The network was trained to produce three different frequencies of oscillation during the 12 initial time steps following the application of the input. The actual output is plotted with solid lines in Fig.7. The periods of these training oscillations were 2, 4 and 6 time steps.

The first nice property of our network is its capability to maintain relatively stable oscillations beyond time step 12 (see the dashed lines in the 1st, 3rd and 5th frames in Fig.7). The most astounding result regards the generalization ability. For instance, with an input value corresponding to the middle between periods 2 and 4, we have an oscillation with a period of exactly 3 time steps.

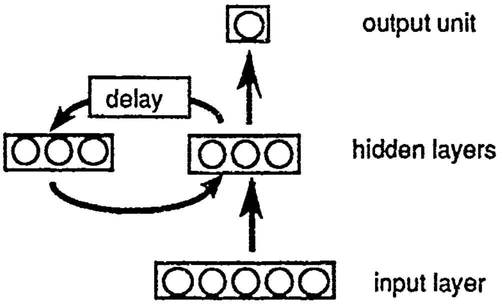


Figure 6: Network for VCO Generation

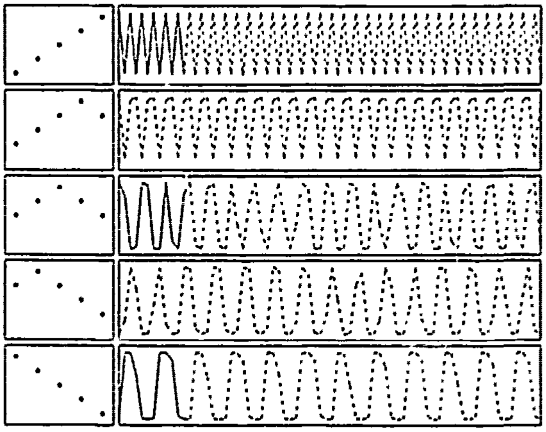


Figure 7: VCO Simulations
Targets in solid lines, generalization in dotted lines

During the simulations, the smooth interpolation behavior appeared to be a rule, but the exact linear relationship between the input value and the period of oscillation did not. Experimentations with a greater number of units in the hidden layers showed that learning is speeded up at the expense of a degraded generalization capability. In all cases the number of iterations to convergence remains quite high, typically around a thousand, due to the complete gradient calculation.

5 NON-LINEAR TRANSFORMATION OF A SIMPLE OSCILLATOR TRAJECTORY INTO COMPLEX GESTURES

An oscillator was in fact already introduced in the first version of Jordan's sequential network (Jordan, 1988). Indeed, his work on realization by an arm model of a trajectory linking the four corners of a rectangle shows that learning is much easier when the network involves two state units connected between each other to produce a resonant second-order filter, hence an oscillator. However, this filter is fed with the output of the forward model, so that the excitation trajectory in the state units can be very complicated, which generally leads to long and difficult learning phases.

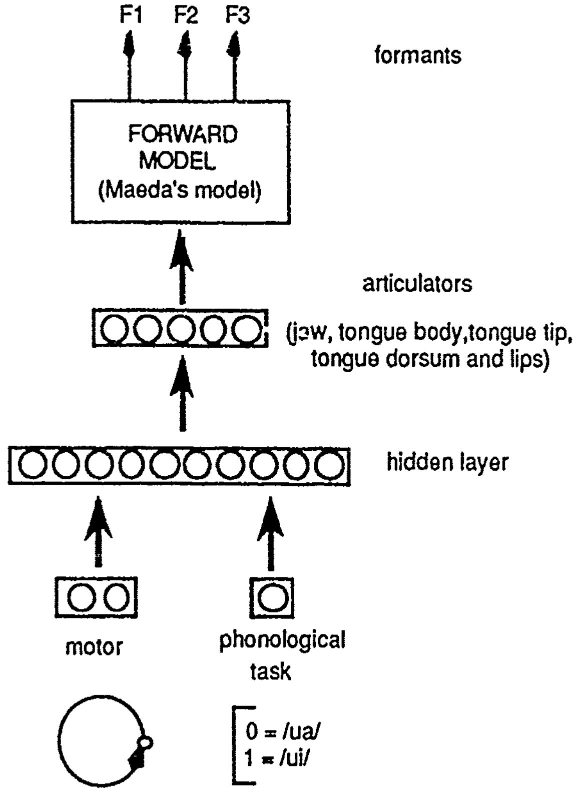

F1 F2 F3
formants
FORWARD MODEL (Maeda's model)
articulators
(jaw, tongue body,tongue tip, tongue dorsum and lips)
hidden layer
motor
phonological task
0 = /ua/
1 = /ui/

Figure 8: Network for Conversion of Simple Motor Cycles into Complex Trajectories

This led us test a simplified structure in which an explicit 2D-oscillator – the "motor" – was introduced in the system (see Fig.8). This oscillator provides the internal representation of time, just as the forward model provides an internal representation of space. It could of course be generated by the model presented in the previous section, but it simply consists here of two cells which respectively deliver a cosine x(t) and a sine y(t) function, so that the point M(x,y) which represents the motor moves along a fixed circle in the (x,y) plane.

The corresponding simulation is based on the assumption that producing vocalic gestures is seen as a highly non-linear transformation of the motor signals delivered by the simple oscillator, the frequency of which controls the speech rate. This approach is consistent with Fowler's model of overlapping gestures (Fowler, 1980) and the hypothesis of a relative invariance of the vocalic gestures to consonantal perturbations as put forward by Öhman (1966).

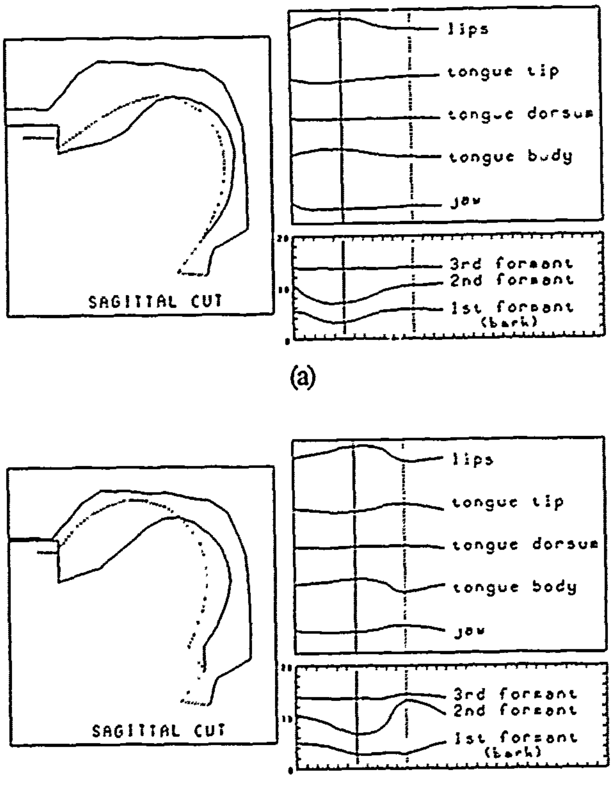
In this model, the control layer consists of three inputs, two for the oscillator and one that codes an entire vocalic gesture. The model appears to be able to learn two vocalic gestures (/ua/ and /ui/) with the same structure and two different input codes. However, it is at the present state of our work relatively hard to observe coarticulation phenomena like those discussed in Section 3. The network tends to find the same articulatory realizations for /u/ in the two different vowel contexts in spite of the additional smoothness constraints. This is due to the explicit time representation both in the motor and in the configurational task, and we are currently investigating how to train this network in a suitable way.

However, we think that this approach is very promising, because the rate of convergence is largely higher than that obtained with the complete model described in the next section. Notice finally that such a structure could provide a basis for a kind of "modular central pattern generator" (see Grillner, 1985; Grillner *et al.*, 1989), able to produce patterns of increasing complexity from simple ones.

6 COMPLETE MODEL

Having separately tested all the main components of the system (the forward model, the use of generalized cost functions for production of smooth gestures and hence coarticulation, the concept of motor oscillator, the possibility to convert a simple pattern into a complex one), we finally tested a complete – and classical – sequential network connected to Maeda's articulatory model, for generation of vowel-to-vowel sequences. The structure of the model is like the structure of the VCO net presented in Section 4. An important point is that the recurrent connections to the state units are fed with the output of cells of the first hidden layer, and not the articulators themselves: this has the interest, between others, to create an internal state representation independent of the distal coordinates. This is useful in the case where several trajectories share a common piece of trajectory and the identification of the trajectory being produced cannot be obtained from the piece of trajectory alone.

The final results are presented on Fig.9 (same presentation as in Fig.5). We observe that it was possible to generate two sequences with the same structure and two different inputs, and that this model does generate coarticulated gestures (compare the shape of the vocal tract for /u/ in the first and the second case). This constitutes the major achievement of this work: it appears that the sequential model structure is indeed able to drive an articulatory model for speech production, and to exhibit coarticulation phenomena compatible with the classical experimental data in this field.



(a)

(b)

Figure 9: Simulated Articulatory/Acoustic Trajectories
(a) Sequence /əuaə/ - (b) Sequence /əuiə/

7 CONCLUSION

This work only constitutes the preliminary phase of long-term research into modeling speech dynamics with reliable models of the vocal tract, general principles inspired by theories of motor control and technical tools available in the field of connectionism. In its present state, it contains three main achievements: (i) show the *feasibility* of the approach in the field of speech production, and the possibility to produce coarticulation phenomena by driving Maeda's model with a sequential network trained with BP algorithms; (ii) show the potential interest of explicit concepts that were only implicit in the original structure, such as those of *oscillators, motors* and *movement conversion*; (iii) give the possibility to separately test the role of *temporal constraints* and the choice of *architectures* in simulation of coarticulation.

There are now a number of perspectives for continuation of this work. First we have to think more about the coding of time in this kind of architecture. Second, we shall try to introduce mechanical constraints, such as inertia, forces, losses, (see Perrier *et al.*, 1988) to produce other kinds of coarticulation effects (non-reached targets for example). Third, we must more clearly define the phonological entities at the input, and estimate the storage and generalization abilities of the network. Fourth, we shall extend our results on vowel-to-vowel gesture to more difficult cases such as superposition of vocalic and consonantal gestures (Öhman, 1966). Finally, we want to introduce proprioceptive and possibly auditory feedback for on-line control and readjustments in reaction to perturbations (see Bailly *et al.*, 1990b, for a first try).

Acknowledgements

The first author was supported by a scholarship of the National Research and Development Council (CNPq), Brazil, under FIAS-CNPq agreement file number 92.0208/88.6. He is also with the Aeronautics Technology Institute (ITA-CTA), Sao José dos Campos, Brazil.

We acknowlegde the IMAG's Neural Nets Group for the financial support to the participation in the Summer School. This work was also supported by LASCO 3 (Rhône-Alpes Pluriannual Program in Human Sciences).

References

C. Abry, J.P. Orliaguet & R. Sock (1990) Patterns of speech phasing. Their robustness in the production of a timed linguistic task: single versus double (abutted) consonants in French. *European Bull. of Cogn. Psych.* 10.

W.B. Adams & J.A. Benson (1989) Rhythmic neuronal burst generation: experiment and theory. In A. Goldbeter (ed), *Cell to Cell Signaling: from Experiments to Theoretical Models*, 29-45. Academic Press.

B.S. Atal, J.J. Chang, M.V. Mathews & J.W. Tukey (1978) Inversion of Articulatory-to-Acoustic Transformation in the vocal tract by a computer sorting technique. *J. Acoust. Soc. Am.* **63**, 1535-1555.

G. Bailly, M. Jordan, M. Mantakas, J.L. Schwartz, M. Bach & M. Olesen (1990a) Simulation of vocalic gestures using an articulatory model driven by a sequential neural network. *J. Acoust. Soc. Am.* **87, S 1**, S105.

G. Bailly, R. Laboissière & J.L. Schwartz (1990b) Formant trajectories as audible gestures: an alternative for speech synthesis. *Special issue of Journal of Phonetics on Speech Synthesis* (forthcoming).

L.J. Boë & P. Perrier (1988) C.F. Hellwag 200 ans après, ou les éléments d'une fibre conductrice. *17th JEPs, Société Française d'Acoustique*, 200-205.

C.A. Fowler (1980) Coarticulation and theories of extrinsic timing control. *Journal of Phonetics* **8**, 113-133.

M. Jordan (1988) Supervised learning and systems with degrees of freedom. *COINS Technical report 88-27.* University of Massachusetts, Amherst, MA.

M. Jordan (1989) Serial order: A parallel, distributed processing approach. In J.L. Elman & D.E. Rumelhart (eds) *Advances in Connexionist Theory: Speech.* Hillsdale, NJ: Erlbaum.

M. Jordan (1990) Indeterminate motor skill learning problems. In M. Jeannerod (ed) *Attention and Performance, XIII.* MIT Press.

R. Laboissière (1990) RBP: a simulator for recurrent back-propagation neural networks. *Technical Report.* ICP (in French).

J.N. Larar, J. Schroeter & M.M. Sondhi (1988) Vector quantization of the articulatory space. *IEEE Trans. on Acoust., Speech and Sig. Proc.* **36**, 1812-1818.

B. Lindblom (1963) Spectrographic study of vowel reduction. *J. Acoust. Soc. Am.* **35**, 1773-1781.

R.R. Llinas (1989) The role of the intrinsic electrophysiological properties of central neurones in oscillation and resonance. In A. Goldbeter (ed) *Cell to Cell Signaling: from Experiments to Theoretical Models*, 3-16. Academic Press.

S. Maeda (1979) An articulatory model of the tongue based on a statistical analysis. *J. Acoust. Soc. Am.* **65**, S22.

S.E.G. Öhman (1966) Coarticulation in VCV utterances: spectrographic measurements. *J. Acoust. Soc. Am.* **39**, 224-230.

P. Perrier, C. Abry & E. Keller E. (1988) Vers une modélisation des mouvements du dos de la langue. *J. d'Acoustique* **2**, 69-77.

P. Perrier, L.J. Boë & R. Sock (1990) Using tomography scanners and a vocal tract cast to determine the transition from midsagittal view to area functions. *J. Speech Hear. Res.* (forthcoming).

D.E. Rumelhart, G.E. Hinton & J.L. McClelland (1986) A general framework for parallel distributed processing. In D.E. Rumelhart & J.L. McClelland (eds), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition 1*, 45-76. Cambridge, MA, Mit Press.

R.J. Williams & D. Zipser D. (1990) Gradient-based learning algorithms for recurrent connectionist networks. In Y. Chauvin & D. E. Rumelhart (eds) *Back-propagation: Theory, Architectures and Applications.* Hillsdale, NJ: Erlbaum (forthcoming).

Extracting features from faces using compression networks: Face, identity, emotion, and gender recognition using holons

Garrison W. Cottrell
Computer Science & Engineering Dept.
University of California, San Diego
La Jolla, CA 92093

Abstract

We summarize a series of experiments using simple autoencoder networks to extract features from 64×64 face images. For our data sets, these networks are capable of extracting features that are linearly separable with respect to several classification tasks. We demonstrate the ability of these networks to extract information relevant to the identification of faces from non-faces, identity of the faces, s^x and feigned emotional state. Also, we show how such networks can be used to form a basis for association of codes from different modalities, forming the basis for meaning representation in terms of association between mental imagery and linguistic codes. Finally, while the mechanisms and architecture are only distantly related to cortical mechanisms, they show how a representational element could be a "distributed grandmother cell", i.e., how a distributed representation may appear localist when only one cell is examined at a time.

1 INTRODUCTION

In this paper we review results from several experiments that share the feature of using an encoder network - a network that does an identity mapping through a narrow channel of hidden units - to extract features from images of human faces. The features are thus extracted in an unsupervised manner, since the "teacher" for the network is the input itself. This shows how much can be done by a network that simply tries to preserve as much information (in the mean squared error sense) as possible in a narrow channel. Such networks have been shown empirically to span the principal subspace of the images (Cottrell & Munro, 1988; cf. Sanger, 1989) when the hidden units stay in their linear range, but differ from Principal Components Analysis (PCA) in that they are capable of using the nonlinearities of the hidden units to produce a compact code even when the PCA solution is degenerate. An example is the original version of the encoder problem (Rumelhart, Hinton & Williams, 1986), where all input vectors are orthogonal.

The recognition of each other as individuals is probably one of the most complex tasks that we perform, and one that is still poorly understood. While our ability to recognize friends seems effortless, as we leave the ground of highly familiar, frequently observed faces, we find a process that is prone to error and highly context-dependent. Neurophysiological studies on monkeys (Desimone et al., 1984). have found cells that are probably involved in this task in the Superior Temporal Sulcus. The findings support the notion of specialized "face cells", that respond best to faces presented frontally, and respond well to human and monkey faces, faces lacking color, and faces with the eyes blanked out. They do not respond to just the outline of the face, or the face outline filled with scrambled features or monkey hair. The response of these cells drops gradually as the face is rotated away from the frontal view. The question is, are these the proverbial grandmother cells? Counting arguments suggest that they could not be, or they would not be discovered so easily. Hence they must be part of some distributed representation.

The work presented here describes a model that suggests a resolution of this dilemma by describing how a neural representation could have face-like receptive fields, while still being distributed across a population of cells. Since the model is not faithful to the architecture of the visual system, no claims are made that this is how it is actually done by the brain - it is simply a demonstration of how a distributed representation may appear localist when only one cell is examined at a time.

2 COMPRESSION NETWORKS

Cottrell, Munro & Zipser (1987) showed that a back propagation network could be used for image compression. The network is trained to simply reproduce its input, and so can be seen as a non-linear version of Kohonen's (1977) auto-associator. The input must be reproduced through a narrow channel of hidden units, so the network learning algorithm must extract regularities from the input vectors. Empirical analysis of the trained compression network shows that the hidden units span the principal subspace of the image vectors, with some noise on the first principal component due to network nonlinearity (Cottrell & Munro, 1988). When linear networks are used, perfect spanning of the subspace is achieved.

Although the network uses error-correction learning, no teacher other than the input is provided, so the learning can be regarded as *unsupervised*. Cottrell et al. suggested that this network could be used for automatic feature extraction in a pattern recognition system. This is the approach taken in the experiments reviewed here.

3 FACE RECOGNITION USING COMPRESSION NETWORKS

The model is shown in Figure 1.

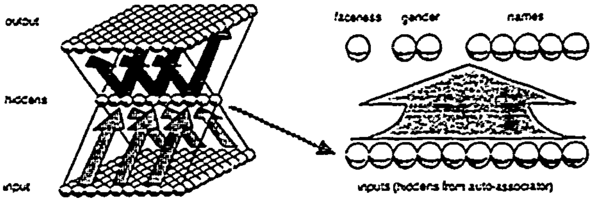


Figure 1: Basic model. The compression network (left) is 64×64. Hidden unit outputs are given as input to single layer classification network, (right).

The image compression network extracts the features, and then the hidden unit representation so developed is given as input to a single layer network which is trained to extract the desired classification depending on the experiment. If it can achieve this, then the representation developed by the compression network must necessarily be linearly separable with respect to the classification desired. We start with a 512×512 pixel image, using 256 gray levels, and reduce via averaging to a 64×64 pixel image. Unlike the original image compression network, the whole image is input at once to our network, so the input layer is 64×64. (For mundane reasons, the inputs in the first experiment described below were actually 61×63.) Depending on the experiment, various training sets, learning rates, and numbers of hidden units were used.

3.1 EXPERIMENT 1: FACENESS, GENDER AND IDENTITY

3.1.1 Data Set

In the initial experiment, performed with Michael Fleming (Cottrell & Fleming, 1990; Fleming & Cottrell, 1990), we obtained 5-20 images each of 17 people, and several non-face images. The face images were taken such that brightness, position of the eyes, and size of the face were held roughly constant. This was done by asking subjects to maintain a position such that their eyes and noses lined up with markers on the monitor, and adjusting the camera so that their chins were approximately at the bottom of the screen. Subjects were asked to make different expressions, and if they wore glasses, images were taken with glasses on and off. If they had long hair, images were taken with this pulled back and let down. This resulted in a set of 204 face images. 27 non-face images were obtained by taking random shots around the lab. Some examples are shown in Figure 2.

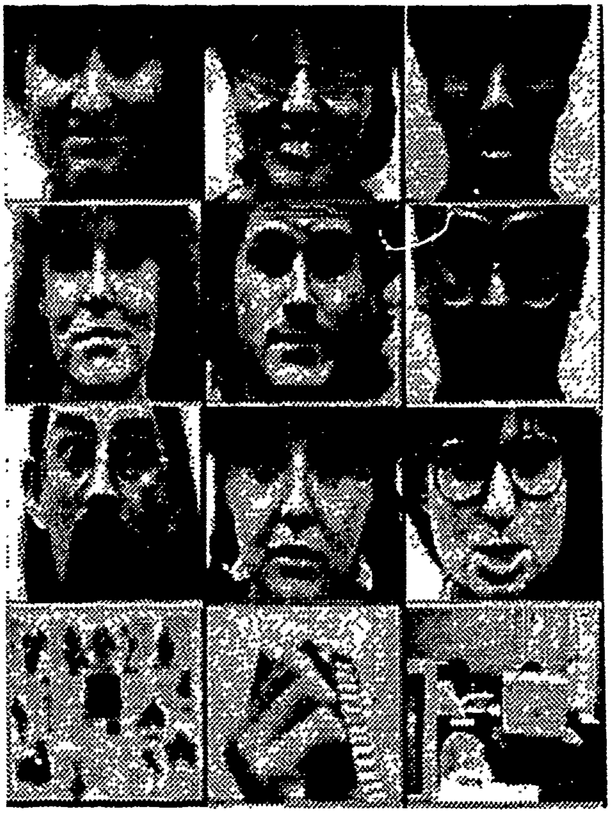


Figure 2: Some elements of the training set for Experiment 1.

A set of 11 people were chosen to be the "familiar" set, with 6 subjects reserved for a "novel" set. For the training set, half of the images of the familiars were selected randomly (64 in all), along with 13 of the non-face images, resulting in a training set of 77 images. No prenormalization other than the initial capture procedure was used on the images. The compression network used [0,1] sigmoidal units at the hidden and output levels, and the 256 gray scales of the original images were linearly scaled into the [.2,.8] range.

3.1.2 Training

In the experiment reported here, we used 80 hidden units. Thus there were about as many hidden units as patterns, and the network could have formed a localist representation. The networks were trained in two stages. The compression network was trained for approximately 200 passes through the entire training set. Then the weights in this network were given tenure (Fahlman's term), i.e., no more learning occurred. At this point the error averaged 225 squared gray levels. The classification network used [-1,1] units, and was trained to classify the hidden unit representations of these images from the compression network into face/non-face, male/female, and identity, using localist output units for each feature.

3.1.3 Results

The model may now be tested in several ways. The important first test is whether the compression network can form linearly separable classes with respect to the different classifications attempted. This is important with respect to the training set, since the features so derived were not derived for classification purposes, but simply to extract as much information as possible with respect to mean square error in the output. Our criteria here is that if a classification unit is over 0 in activation, it is on, otherwise it is off (these units were [-1,1] units). For the training set, the network achieved 100% performance by this criteria.

The second test is whether the network can recognize novel images of familiar faces. The network achieved over 98% performance on 65 familiar (but never trained) faces in both name and gender classification using The network misclassified one female face's name and gender. A second generalization test is whether the network can recognize completely novel faces as faces, and classify them according to gender. There were no errors with respect to faceness, but 26 out of 70 female images were classified incorrectly as male.

Finally, the classification network can be tested for its ability to recognize faces with bars through them and other kinds of distortions. Using the untrained-on familiar set, and the same criterion as before, we obscured one fifth of the image with a gray bar at five locations, one for each fifth of the face. This had little effect on recognition, except at the top of the image, where a 29% error rate on identity was obtained, suggesting that the network is using the forehead of the subject as a discriminator in many cases. Subjects differed considerably here with respect to how much of their forehead was obscured by their hair, whether the boundary of ـıe hair was smooth or ragged, and and whether their hair was straight or curly.

The compression network may be tested as a memory and representation network for faces by giving it novel faces and observing how well it represents them by looking at the output of the network. This ability is highly dependent in this experiment on the training set, because the number of subjects is small. We show an example in Figure 3. The space spanned by the hidden units does not capture subject TA very well. We expect that a network with many more hidden units and a more diverse training set would be able to represent a much larger set than the training subjects.

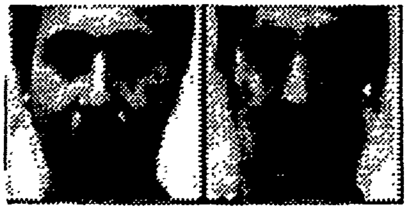


Figure 3: Example of the reproduction of a novel face.

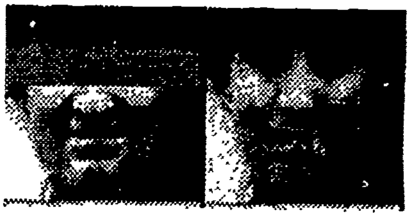


Figure 4: An example of redintegrative memory: Recovering the whole from a partial input.

The network may also be tested as a redintegrative memory by distorting familiar patterns (using the gray bars as above) and observing how well it recovers the originals. It should do well at this task, given the recognition rates for such images, and it does. An example is given in Figure 4. Again, this is an image that was not trained on, although other images of this subject were. It is completing the pattern based on the stored information about this person's face.

3.1.4 Internal Representation

What is the hidden unit representation? First, the hidden unit activation patterns are essentially binary: A histogram of the hidden unit representations shows that almost all are either above 0.75 or below 0.25 for all presentations (in fact most are within .1 of full off or on). Second, even though a localist representation would suffice, the representation is distributed. Histograms of the hidden unit activations for individual subjects show that roughly half of the units are on for most faces.

We can convert the receptive fields of the hidden units to gray scale (with multiplicative enhancement) for viewing purposes. Figure 5 shows several of the hidden unit receptive fields. As can be clearly seen from this figure, the "features" discovered by this network are not simple discrete features such as ratios of distances. Nor are they simply templates of a single person's face. Thus the term *holon* (suggested by Janet Metcalfe) seems appropriate. As a first attempt at definition, we will call any representational element a holon if its receptive field subtends the whole object whose representation it is participating in. A preferable additional constraint is that the information in a set of holons in the ideal case be maximally distributed: I.e., the entropy of any unit is maximized across the training set, although we have not demonstrated that here. The latter restriction eliminates grandmother cells, ensures that the representation is noise resistant, and also distributes the processing load evenly. A weak point of our definition is the difficulty of defining precisely the notion of a "whole object".

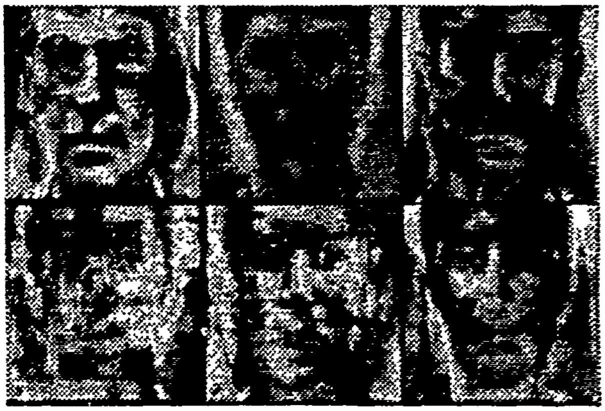


Figure 5: Holons from experiment 1.

This definition applies to many distributed representational schemes, but does not apply to articulated ones such as the Wickelfeatures used by Rumelhart and McClelland (1987) in their past tense model as these only represent portions of the verb. The whole object assumption prevents having holons for a "room", simply because we can not get a room to fill but not extend beyond our sensory surface at once. Given this meaning for the term, the units of area 17 are not holons, but the units in Superior Temporal Sulcus (STS) are. The main motivation for this definition is to give an alternative notion to the grandmother cell one for face cells in STS (Desimone et al., 1984).

In any event, the hidden unit representations developed by our network are full-face features, yet, they are not person-specific. The network does a kind of distributed template matching process. We can test the specificity of the representation by driving the recognition network with each hidden unit, or holon, separately. For example, holon 75, when used to drive the recognition network, generates a response that is about equally divided between two subjects, one male and the other female, who have similar facial characteristics. Also, holon 70 most resembles Mike Fleming, yet when this unit is used to drive the recognition layer, the strongest response is from a female's name unit.

3.2 EXPERIMENT 2: EXTRACTING FEIGNED EMOTIONAL STATE

3.2.1 Data Set

In this experiment, performed with Janet Metcalfe (Cottrell & Metcalfe, forthcoming) a network as above was trained using 160 64×64 images of 20 undergraduate subjects who where asked to display (in randomized order) facial expressions corresponding to the adjectives: Happy, sad, astonished, pleased, sleepy, angry, content, bored. In this experiment, the images were normalized to have equal average brightness and variance. The network had 40 hidden units, and used sigmoids in the [-1,1] range. The gray levels were linearly scaled to the range [0,.8].

3.2.2 Training

The compression network was trained for 50 epochs. The speed-up was attained by using a very low learning rate on the hidden units, .0001 vs .1 on the output units. This prevented early "cell-death" resulting from changes to the 4096 input weights forcing the hiddens into the flat areas of the sigmoid, where learning is extremely slow.

3.2.3 Results and Discussion

Part of the training set and its reproduction by the network is shown in Figure 6. The columns correspond to three of the eight different feigned emotional expressions. It is clear from this figure that even humans cannot reliably distinguish between some of the feigned emotions displayed. The classification network can't either (discussed below), but it can distinguish some of the positive ones, and treats as similar nearby emotions in an emotional space defined by Russell (1980), based on multidimensional scaling of subjects ratings of the adjectives we used.

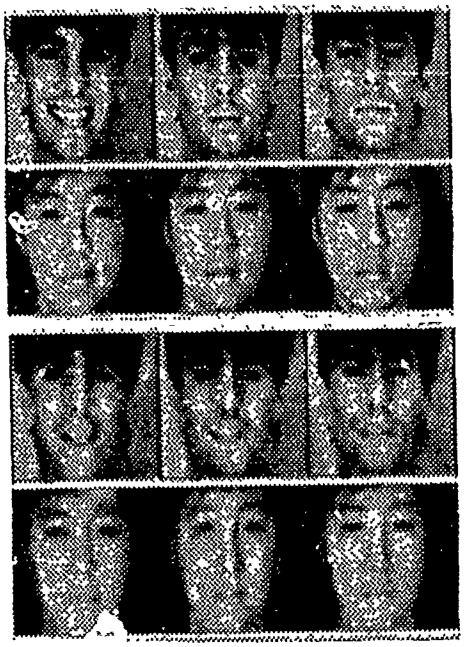


Figure 6: Part of the training set and its reproduction by the network.

The network learned to discriminate 94% of the training set for identity. One woman was taken for another. Sex discrimination was perfect. However, the observation during data acquisition that negative emotions are poorly portrayed was confirmed by the network's performance. It could not reliably distinguish the four negative emotions. The network was much better at detecting positive states than negative ones (Figure 7). Further, for these emotions, the network tended to blend two states that were nearby one another in Russell's emotion space. This is evidenced by the shoulders in these graphs.

It is interesting that the compression network obtains a very different representation in this study compared to the previous one. First the units operate in the linear range of the squashing function, so the first network is basically extracting the principal subspace of the image vectors (Cottrell & Munro, 1988; Sanger, 1989). Viewing the hidden unit receptive fields as before results in white noise. The representational elements may be converted into a more interesting form by taking the principal components of the hidden unit activations treated as 160 vectors, and running these through the decompression half of the network for viewing. The final images are normalized for brightness and variance in the same way as the training set. The first 6 are displayed in Figure 8. These are the underlying features used by this distributed system, in order of importance (decreasing eigenvalues).

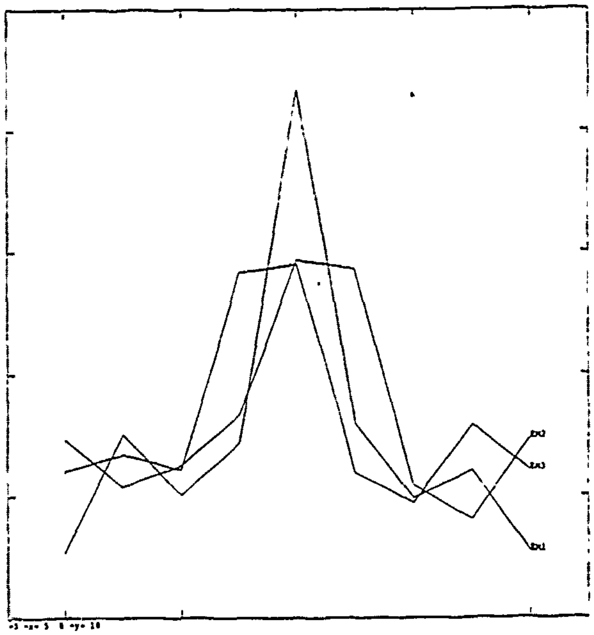


Figure 7: Total response of 8 output units over all faces for three positive feigned emotions. Each graph is centered on correct response.

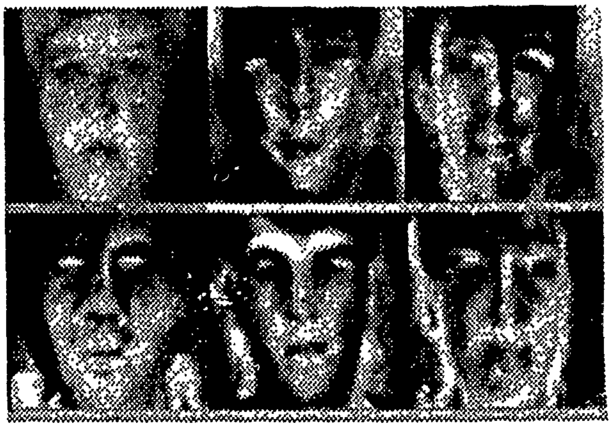


Figure 8: Six holons from experiment 2.

3.3 GRANDMOTHER CELLS OR DISTRIBUTED REPRESENTATION?

The difference in the representation obtained in experiments 1 and 2 is interesting. In the first experiment, where the learning rate was much higher, the representation was basically binary, and the hidden unit receptive fields were "face-like". In the second network, the hidden units were basically linear, and because of that an uninterpretable (to the human eye, without analysis) linear transform of the principal components was obtained. A single cell recording of one of these units would vary smoothly across the range for different faces. For the nonlinear representation of the first experiment, the cell would either fire or not for a given face.

This suggests that the cells of the STS represent faces in a manner similar to the representation obtained in the first experiment, rather than the second. I.e., in Feldman and Ballard's (1982) terms, they will be more like value units than variable units. However the values represented are not those of an individual face, but a *blend* of many faces. That is, they are coarse-coded.

4 GROUNDING MEANING IN PERCEPTION

In this experiment, conducted with the help of Chris Haupt, we explored the ability of networks to classify inputs without an external teacher, in a variation of the experiment suggested in (Cottrell, 1987), and similar to Chauvin (1988). The idea here is to explore the possibility of networks to form associations between differing modalities, in an effort to provide a computational basis for grounding meaning in perception.

The basic idea of our model is that the learner is processing input from several modalities, extracting regularities from each modality, and forming associations between elements in different modalities that co-occur often in time. We will assume that one of the modalities is auditory, and that linguistically sensitive processes have already segmented the speech signal into tokens which we will represent simply as strings of letters for convenience. Self-supervised connectionist models have been developed that can segment streams of letters into words (Elman, 1990); similar results can be obtained for speech (Doutriaux & Zipser, this volume).

The other modality will be visual, and we will use the compression network as our model of the visual input. A more realistic network would do preprocessing on these images in the manner that the visual system does it, but for now, it matters little what the actual inputs to our networks are - whether their inputs are preprocessed or not, the point of this exercise is to demonstrate the idea that automatically extracted internal representations from different modalities can be trained to evoke one another, becoming "prototype" attractors that can be considered the basis of "meaning". However, the actual internal representations formed, the speed of learning, etc., would change with a different input encoding. These are details that are not of import to the current simple demonstrations.

In this experiment, we show that a network can be presented with inputs representing names and faces of a series of 10 subjects, and can internally generate representations that can be used to associate the two inputs. This corresponds, in our view, to the learning of a grounded predicate such as **Name(OBJ-33, "Fred")**, but it has the added advantage of being able to generalize to new instances of Fred (e.g., **OBJ-436** and forms a prototype of what Fred "looks like".

4.1 ARCHITECTURE

The associative network consists of four sub-networks as detailed in Figure 9.

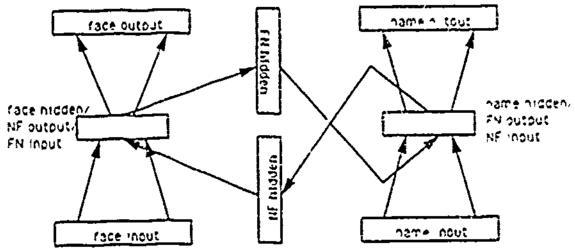


Figure 9. The associative network. On the left, the face compression net. It has 4096 inputs and 40 hidden units. On the right, the name compression net. It has 130 inputs and 10 hidden units. In between, the hidden units of the subnetworks that learn the mappings between the hidden units of the compression networks. They use 5 hidden units each.

Each sub-network is trained using the back-propagation algorithm. The face network, on the left of Figure 9, learns a compact encoding of the faces as in previous experiments. The name network, on the right of Figure 9, is also a compression network, and can conceptually be treated as learning the names in parallel with the face network learning the faces. The face-to-name network (FN-net) consists of the hidden units from the face network, an internal set of hidden units, and the hidden units from the name network. It is trained to map hidden unit activations of the face network to hidden unit activations of the name network. The name-to-face network (NF-net) works in a similar fashion. The NF and the FN networks comprise the associative memory of this model.

4.2 DATA SET

The faces used for this experiment were eight faces of ten individuals drawn from experiment 2 above. Half of the faces (a total of 40) for each individual were used for training, and the other half for testing. A name consisting of five letters was assigned to each individual. Twenty-six processing units (one for each letter of the alphabet) were employed for each letter of the name. A localist encoding was used which consisted of turning on the unit corresponding to the letter in that position, and turning off all of the other units. There were a total of 10 names corresponding to the 10 individuals whose faces appeared in the images.

4.3 TRAINING

The training process was as follows: First the name and face compression networks were trained separately. The face compression network was trained with a learning rate of 0.01 for 1000 epochs which resulted in an average root mean square pixel intensity error less than 13 gray scale levels (the original images are 255 gray level images). The name network was trained in a similar fashion.

After the face and name networks were trained they were given tenure. Then the FN and NF networks were trained. For each face in the training set, the hidden unit activations of the face network are obtained, and the target for this learning trial is the hidden unit activations of the corresponding name in the name network. For the name to face mapping, on each trial, for a particular name, one of the four faces of that person is randomly chosen and its compressed representation in the face network is used as the target. As this is a one to many mapping (one name to four faces for that name) the network learns to produce the average of the faces in the training set for the named individual.

We developed two training methods. In the first, we simply trained pairs of name and face representations in one pass from one net to the other using back propagation. We termed this "unidirectional training. In the second, the activations were cycled back and forth 1 or more times, using the original compressed representation on each side as the target, and changing weights on each forward pass from one side to the other. The first method can be seen as a special case of the second, using 0 cycles (or half cycles, if you prefer). The learning rate was decreased during the cycling process. This is in response to the fact that as the networks cycle back and forth, before the pairs have been adequately learned, the error in the outputs gets progressively worse. These outputs are the inputs for the next cycle. Thus, the network would be learning improper input-target associations if the weights were changed with the same rate at later stages as they are in earlier stages when the inputs are closer to the target values.

Using the half cycle method, the face-to-name network achieved a total sum squared error of 0.27 after being trained for 500 epochs with a learning rate of 0.2. The name-to-face network stabilized at a total sum squared error of around 65 (an average error of 0.04) after 9000 iterations with a learning rate of 0.05.

For the cyclic training, the network was trained until the total sum squared error at each stage of the cycle dropped to at least the level of the network with only unidirectional training. An in depth study of the optimal epochs and learning rates was not performed. In these experiments a learning rate of 0.01 was used for both the name-to-face and the face-to-name networks. One thousand epochs were found to be sufficient when training with 1, 2, or 5 cycles per face-name pair. Five times as many epochs were needed for 10 and 20 training cycles.

4.4 TESTING

The trained network is tested by presenting a face or a name to the appropriate side of the network and cycling the activation. At any point during this process, the actual face or name represented can be "read out" from the decompression network. The name that is closest in Euclidean distance to that produced by the network is considered to be the name chosen by the network. If this is the name associated with the face, then it is considered a correct choice by the network. Similarly, the closest face in Euclidean distance to that produced by the network is considered to be the face chosen by the network. (This may either be a test face or a training face). If this is one of the faces associated with the input name then it is considered a correct choice by the network.

4.5 PERFORMANCE

First we tested the trained network by presenting faces to the face net and cycling for 0 or more times. Table 1 shows the results of this test for unidirectionally and bidirectionally trained associative networks. The table entries are the number of correctly chosen names (out of 40 trials). The name is the result of presenting a face and cycling back through the face net the specified number of times before reading the output.

For the unidirectional version, when the activation is propagated directly to the output of the name network without any cycling, the network chooses the correct name 100% of the time. As the activation is cycled through the network, however, the accuracy degrades until it eventually stabilizes at 60% after 50 cycles. This means that not all of the pairs of faces and names form attractors when treated dynamically. This behavior is mimicked on the test faces.

Networks with bidirectional training perform substantially better after several cycles than the network trained unidirectionally. A bidirectionally trained network with 10 cycles of training maintained perfect recall of the appropriate name for each face even after 50 cycles. Hence each pair is near a stable attractor. On unseen faces of individuals in the training set the bidirectionally trained networks again exceeded the performance of the unidirectionally trained network.

Table 1: Table 1. Face to name association. A. Training faces. B. Test faces.

# cycles in training	# cycles in testing			
	0	2	5	50
0	40	33	28	24
1	40	40	33	29
2	40	40	33	28
5	40	40	35	29
10	40	40	40	40
20	40	40	40	40

(a) Training faces

# cycles in training	# cycles in testing			
	0	2	5	50
0	37	30	25	22
1	37	35	30	27
2	37	36	31	25
5	37	36	32	26
10	38	36	35	35
20	37	36	36	36

(b) Testing faces

Table 2: Table 2. Name to face association. A. Training faces. B. Test faces.

# cycles in training	# cycles in testing			
	0	2	5	50
0	10	9	8	6
1	10	10	8	7
2	10	9	8	7
5	10	9	9	7
10	10	10	10	10
20	10	10	10	10

(a) Training faces

# cycles in training	# cycles in testing			
	0	2	5	50
0	10	9	7	6
1	10	10	8	7
2	10	9	8	7
5	10	9	9	7
10	10	10	10	10
20	10	10	10	10

(b) Testing faces

The second test of the associative network presents names to the name network, activating the face network hidden units through the name-to-face network. As in the previous test this activation is circulated back and forth for any number of cycles after which the resulting face is read out from the output layer of the face network. This output is compared via Euclidean distance to the training faces or the testing faces. Table 2 shows the results of this test for unidirectional and bidirectional training of the associative networks. The table entries are the number of correctly chosen faces (out of 10 trials). The face is the result of presenting a name and cycling back through the name network the specified number of times before reading the output.

As in the first test, the network trained unidirectionally achieves 100% correctness on the training faces after 0 cycles through the associative networks. This degrades to 60% after 50 cycles. The network also achieves 100% correctness on the untrained familiar faces after 0 cycles which then again degrades to 60 % after 50 cycles. As in the first test, the accuracy of the network when the activation is cycled through the associative networks improves when it is trained bidirectionally.

For both the training and test faces a bidirectionally trained network with at least 10 cycles through the associative nets on each training pass produced an appropriate face for every name, even after 50 cycles through the associative networks.

The reason the bidirectionally trained network performs better than the unidirectional version is that given a name, the network produces the average of the compressed representation of the training faces for the individual with that name. The bidirectionally-trained network has been trained not only with face/name pairs, but with average-face/name pairs as well. Since the unidirectional network was never trained with the average face it produces more errors in the name network when mapping from the average face. This "slightly off" representation has not been trained to produce the average face, and the system does not form an attractor.

Figure 10 shows an example of the unidirectional network producing the incorrect face/name after several cycles through the associative network. This is the readout from the face compression network as the hidden units cycle starting with the name JANEL. It eventually shifts to JAMES. (Note the similarity of these strings). Initially an appropriate face was produced by the network as was the correct name JANEL. Each cycle through the network resulted in a face that looked less like JANEL and more like JAMES. Eventually the network produced the average face for JAMES and also produced JAMES' name in the name network. All subsequent cycles continued to produce JAMES. The bidirectional network produced the average face for JANEL, and the name JANEL, on every cycle through the associative network. This behavior was typical for all of the face/name pairs presented that lost track. This is very interesting because it is indicative of how chains of association can occur. JANEL has "reminded" the network of JAMES. Our set of examples was small enough that the network can be trained to hold onto the faces, but in a realistically sized network, and with noise in the system, we can expect networks like this to "free-associate".

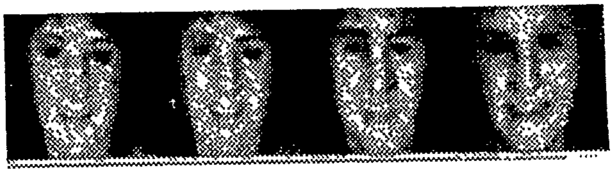


Figure 10: Losing track of a face.

5 CONCLUSIONS

We summarized a set of recent experiments that use compression networks to extract features from environmental stimuli. These networks were shown to often produce representations that were linearly separable with respect to various classification tasks. What remains to be done is a crucial control experiment: It may be the case that the inputs themselves are linearly separable for the classifications attempted in the first place. However, there are two reasons for continuing interest in these models should that test prove successful. First, they produce a compressed representation which is useful for many tasks, and practical applications need smaller input sets for fast learning and smaller VC dimension in the classifying network. Second, as the last experiment shows, these networks can provide the basis for constructing automatically internal representations that may then be used as inputs or targets for other networks, allowing associations between differing modalities to operate on smaller codes.

One interesting insight provided by these tests has been that representations must be considered as a whole before one can determine whether they are distributed or localist. A scientist considering only the receptive field properties of one or a small number of the hidden units in the first experiment may conclude that there is a localist representation, since the receptive field is similar to whole faces. However, inspection of the network as a whole reveals that the representation is distributed - each cell participates in many face representations, and each face is represented by many cells.

References

Chauvin, Yves (1988) Symbol acquisition in humans and neural (PDP) networks. Unpublished Ph.D. thesis, Department of Psychology, University of California, San Diego.

Cottrell, G. (1987) Toward connectionist semantics. Invited paper in *Theoretical Issues in Natural Language Processing-3: Position Papers,* Las Cruces, New Mexico. Reprinted by LEA (1989).

Cottrell, G.W. & Fleming, M.K. (1990) Face recognition using unsupervised feature extraction (1990) In *Proceedings of the International Neural Network Conference.* Paris, France.

Cottrell, G. & J. Metcalfe (to appear) EMPATH: Face, emotion & gender recognition using holons. In *Advances in Neural Information Processing,* D. Touretzky, (Ed.), San Mateo: Morgan Kaufmann.

Cottrell, G.W. and Munro, P. (1988) Principal components analysis of images via back propagation. Invited paper in *Proceedings of the Society of Photo-Optical Instrumentation Engineers,* Cambridge, MA.

Cottrell, G., Munro, P. and Zipser D. (1987) Learning internal representations from gray-scale images: An example of extensional programming. In *Proceedings of the Ninth Annual Cognitive Science Society Conference,* Seattle, Wa.

Desimone, Robert, Albright, Thomas, Gross, Charles, and Bruce, Charles. (1984) Stimulus-selective properties of inferior temporal neurons in the Macaque. *J. Neuroscience, 4,* 2051-2062.

Doutriaux, A., & Zipser, D. (1990) Unsupervised discovery of speech segments using recurrent networks. In Touretzky, D., Elman, J., Sejnowski, T. and Hinton G.E. (Eds.) *Proceedings of the 1990 Connectionist Models Summer School.* San Mateo: Morgan Kaufman.

Elman, J. (1990) Finding structure in time. *Cognitive Science, 14.*

Feldman, Jerome A. and Dana Ballard (1982) Connectionist models and their properties. *Cognitive Science, 6,* 205-254.

Fleming, M.K. & Cottrell, G.W. (1990) Categorization of faces using unsupervised feature extraction. In *Proceedings of the International Joint Conference on Neural Networks,* San Diego, CA, June, 1990.

Kohonen, T. Lehtio, P., Oja, E., Kortekangas, A., & Makisara, K. (1977) Demonstration of pattern processing properties of the optimal associative mappings. In *Proc Intl. Conf. on Cybernetics and Society,* Wash., D.C.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986) Learning representations by back-propagating errors. *Nature, 323,* 533-536.

Rumelhart, D. & McClelland, J. (1986) On learning the past tenses of English verbs. In J.L. McClelland & D.E. Rumelhart (Eds.), *Parallel Distributed Processing, Vol* 2., Cambridge, MA: MIT Press.

Russell, J. A. (1980) A circomplex model of affect. Journal of Personality and Social Psychology, 39,1161-1178.

Sanger, T. D. (1989) Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks, 2*, pp. 459-473.

The Development of Topography and Ocular Dominance

Geoffrey J. Goodhill
School of Cognitive and Computing Sciences
University of Sussex
Falmer
Brighton, BN1 9QN
GREAT BRITAIN

Abstract

The experimental evidence regarding the development of topography and ocular dominance in the visual system is reviewed, and previously proposed models addressing one or both phenomena are discussed. Two new models, each addressing both problems simultaneously, are then introduced, and their strengths and weaknesses compared.

1 INTRODUCTION

1.1 THE BIOLOGICAL MODELING PROBLEM

The problem of modeling the nervous system can be divided into two separate but related parts. The first is to understand how the complex but extremely regular patterns of connections between neural structures come about. The second is to understand how the resulting neural hardware supports the computations underlying learning, memory and higher-level cognitive functions. This research addresses the first problem.

One very prominent feature of many mappings in the brain is that they are topographic (i.e. neighbourhood preserving): this applies to mappings from high dimensional spaces (e.g. space and orientation) to the 2-D cortex as well as to mappings from 2-D spaces (e.g. one retina) to the cortex. It is clear that there is not enough genetic information to specify each connection explicitly as an entry in a weight matrix, therefore the brain must be using more general mechanisms. The task is thus to understand what guiding principles the brain could be using to set up these regular patterns.

One of the best studied examples of a topographic mapping in the brain is the retinotopic projection from retina to cortex (higher vertebrates) or optic tectum (lower vertebrates). Many experimental manipulations are possible that allow stu'ly of the formation of this map under abnormal conditions, and thus shed light on the mechanisms involved. Some of the experiments that have been performed on frogs and goldfish include rotating, removing or transplanting fragments of retina or tectum (for review see Udin and Fawcett (1988)), silencing electrical activity using chemicals such as Tetrodotoxin (e.g. Schmidt and Edwards, 1983; Meyer, 1983) and changing the properties of the visual environment (Schmidt and Eisele, 1985; Cook and Rankin, 1986).

A particularly interesting case of a retinotopic mapping is the pattern of ocular dominance stripes seen in mammalian striate cortex. Fibres from the 2 eyes cross at the optic chiasm and separate so that the left visual field (fibres from the right half of each eye) maps to the right half of the brain, and conversely for the left visual field. After synapsing at the lateral geniculate nucleus (LGN), where fibres from the two eyes are kept completely separate in different layers, the fibres terminate in the striate cortex, mostly contacting cells in layer IVc (for more details see Lund (1988)). In this layer, connections are laid out topographically, but in a pattern of interdigitating stripes. In monkeys, cortical cells are almost entirely monocular in this layer, so that an electrode traversing this region would be driven by one eye and then the other in an alternating fashion (e.g. Hubel and Wiesel, 1977; LeVay *et al*, 1985). Information from the two eyes is thus kept separate at this stage, and is only brought together at later stages of processing. Many other examples can be found where fibres from related sensory structures map to the same part of brain in a segregated pattern, usually either stripes or blobs.

Amphibians and fish do not naturally possess a region of brain innervated by both eyes. However, experiments can be performed which lead to fibres from one retina invading the space normally occupied solely by fibres from the other retina, and in these cases stripes can be formed. Such experiments include deflection of the optic nerve (Cronly-Dillon and Glaizner, 1974), implant of a third eye (Constantine-Paton and Law, 1978), tectal ablation (Law ar.d Constantine-Paton, 1980), and the formation of compound eyes (Fawcett and Willshaw, 1982).

There is evidence that both chemical markers and electrical activity in nerve fibres play a role in the development of topographic mappings. Even though crude topography between sheets of nerve cells can be set up in the absence of electrical activity, a refined map will not form if activity is blocked (e.g. Archer, Dubin and Stark, 1982; Schmidt and Edwards, 1983). Blocking activity also prevents segregation into dominance stripes (Meyer, 1982; Boss and Schmidt, 1984; Reh and Constantine-Paton, 1985; Stryker and Harris, 1986). For reviews of the role of activity see Fawcett and O'Leary (1985), Udin and Fawcett (1988), and Constantine-Paton, Cline and Debski (1990).

The fact that stripes form in the unnaturally produced cases mentioned above suggests that ocular dominance may be a side-effect of the application of more general rules of cortical plasticity. It is argued that ocular dominance thus provides a key test for models of development, in particular those concerning topography and orientation selectivity.

1.2 DEPRIVATION EXPERIMENTS

One property of ocular dominance columns that makes them especially useful for constraining models of development is that in cats and monkeys their structure can be changed by altering the visual experience of the animal. This provides a powerful tool for investigating the role of experience in cortical development. The basic experiment is to change the visual input to one or both eyes for a period of time and then observe the effect of this on both the pattern of stripes in the cortex and the effectiveness with which layer IV cortical cells can be driven by each eye. It should be noted that in all the experiments, effects in the cortex can only be seen when the deprivation occurs during the so-called *critical period*. This period is between about 3 and 6 weeks after birth in kittens, and the first 6 weeks in monkeys (Hubel, Wiesel and LeVay, 1977).[1] A sample of the procedures used and their effects are as follows.

- **Monocular suture.** Here one eye is occluded or sewn shut during the critical period. After a few weeks it is found that substantially more of the cells in layer IV can be driven by the normal eye as compared to the deprived eye, and the ocular dominance stripes are now of different thicknesses. The stripes representing input from the normal eye expand at the expense of the stripes from the deprived eye (e.g. Hubel, Wiesel and LeVay, 1977, Shatz and Stryker, 1978).
- **Binocular suture.** Here both eyes are sewn shut. Surprisingly, stripes appear to form roughly as normal (e.g. LeVay, Wiesel and Hubel, 1980).[2]
- **Reverse suture.** Here one eye is shut for a time, then opened again and the other eye is shut. If this reversal takes place during the critical period, the impoverished ocular dominance stripes from the eye first deprived are able to expand and eventually (given sufficient time within the critical period) end up wider than the stripes from the eye that was initially open (e.g. LeVay, Wiesel and Hubel, 1980).
- **Impulse blockade.** Even if an eye is shut, there is still some spontaneous activity in retinal ganglion cells. Stryker and Harris (1986) investigated the effect on the pattern of stripes by blocking activity totally with Tetrodotoxin (TTX). When both eyes were thus treated, stripes failed to form.
- **Artificial strabismus.** Here the correlational structure of the inputs from the two eyes is disrupted by making each eye view different parts of the world (Hubel and Wiesel, 1965). In this case, stripes form that are sharper than normal: virtually no cells in the cortex can be binocularly driven. However, these changes occur more slowly than the changes following monocular deprivation (Freeman, Sclar and Ohzawa, 1982).

2 COMPUTATIONAL MODELS

These models fall into three classes: those attempting to explain topography, those attempting to describe ocular dominance, and those attempting to describe both processes simultaneously. They will be discussed in this order. It should be noted that another way of categorizing the models is in terms of whether they refer to chemical mechanisms, electrical mechanisms, or both, and this division is orthogonal to the above categorization. In addition, although understanding topography and ocular dominance is related to both the development of orientation selectivity (e.g. Barrow, 1987) and the pattern of orientation columns across the cortex (e.g. Durbin and Mitchison, 1990), for simplicity orientation will not be considered in this paper.

2.1 TOPOGRAPHIC MAPPINGS

"Chemoaffinity", the idea that chemical specificities can be used to guide axons growing from one sheet of nerve cells to their correct topographic sites of termination on another sheet of nerve cells, was first proposed by Sperry (1963), and has been the subject of much experimental investigation. Mathematical models using such affinities were first investigated by Prestige and Willshaw (1975), who showed that a topographic map could be formed by

[1] However, there is evidence that the critical period can be delayed almost indefinitely in the absence of any visual experience. For discussion see Munro (1986).

[2] Some controversy exists regarding whether this is true in cats see for example Swindale (1981).

matching *gradients* of chemical substance existing separately in the retina and tectum, without requiring each retinal axon to be specific for only one site in the tectum. Various models have subsequently been investigated using the idea of gradients (for example Fraser (1980)). The basic proposal is that there exist at least 2 (one for each spatial dimension) chemical markers in the pre-synaptic sheet and at least 2 markers in the post-synaptic sheet, distributed so that there are smooth gradients of marker in each direction in both retina and tectum. It is assumed that an axon arriving from the pre-synaptic sheet has an affinity for post-synaptic sites that is proportional to both the amount of chemical marker at its point of origin in the pre-synaptic sheet and the amount of chemical marker at its post-synaptic site of termination. The added assumption of a normalization rule on synaptic strengths provides enough machinery for a topographic map to form (Prestige and Willshaw, 1975). Direct evidence for the existence of gradients of marker chemicals in retina and tectum comes from cell adhesion studies (for review see Gottlieb and Glaser (1981)). Although able to account for some experimental results, there are other experiments which such theories have difficulty explaining (for some examples see Willshaw and von der Malsburg (1979))

A modification of this idea that has wider explanatory power is the Tea Trade model (von der Malsburg and Willshaw, 1977; Willshaw and von der Malsburg, 1979). Here there is just one set of markers, in the pre-synaptic sheet, and these are induced into the post-synaptic sheet where they diffuse, so setting up a distribution of chemicals in the post-synaptic sheet. Willshaw and von der Malsburg used a molecular version of the Hebb rule to modify synaptic strengths between pre- and post-synaptic cells. They showed that simultaneously maximizing chemical similarities between both pre-synaptic axons and their post-synaptic site of termination, and neighbouring post-synaptic cells, leads to topographic projections.[3] Experimental results supporting the idea that there may not be intrinsic markers in the tectum include Jacobson and Levine (1975), who showed that an ordered map can form when frog retinal projections are forced to regenerate to an ectopic piece of tectum. The results of Schmidt (1978) provide positive evidence for the induction of pre-synaptic markers into the post-synaptic sheet. Here, goldfish tectum exhibited a "memory" for a previously created distorted map when the optic nerve was forced to regenerate normally.

Models of retinotopic map formation based on activity-dependent processes have also been proposed. It is clear that, when presented with patterned input, there will be correlations in the activity of neighbouring ganglion cells. There is also evidence that the purely spontaneous firing of neighbouring ganglion cells is temporally correlated (Arnett, 1978, Mastronade, 1989), and this may provide a means of giving the post-synaptic sheet information about the topography of the pre-synaptic sheet. A model of retinotopic map formation based on this idea was proposed by Willshaw and von der Malsburg (1976). the so-called neural activity model. At each time step, a pre-synaptic cell fires along with its neighbours, exciting cells in a post-synaptic sheet (which also have fixed lateral connections of a centre-surround form). Changes in synaptic efficacy are calculated according to a Hebb rule, and Willshaw and von der Malsburg showed that such a mechanism is capable of forming retinotopic maps.

However, evidence from the blocking experiments described earlier shows that both chemical and activity-based processes must play a role in map formation. So far the only serious attempt to integrate these two factors has been Whitelaw and Cowan (1981), who proposed an activity mechanism similar to the neural activity model, with the assumption that there are chemical gradients in both retina and tectum which define an intrinsic specificity between pairs of retinal and tectal cells (similar to Fraser (1980)), and that this specificity acts as a scaling factor for the magnitude of the synaptic modification induced by correlated activity. However, there is no direct evidence for this assumption. This model has recently been updated (Cowan and Friedman, 1989) to include the effects of fibre-fibre interactions between retinal axons. Such interactions will be discussed in more detail later.

2.2 OCULAR DOMINANCE

All the models described in this section do not address topographic map formation: they assume a pre-existing retinotopy. The neural activity model described above was adapted to the ocular dominance case by von der Malsburg and Willshaw (1976), who showed that the assumption of correlated activity within each eye, but with no correlation between the two eyes, was sufficient to cause monocular cells to form. The centre-surround excitation/inhibition in the cortex then ensures that these monocular cells are laid out in a pattern of stripes. Swindale (1980) proposed a theory of the formation of ocular dominance stripes in terms of the effect exerted by synapses on the growth of other synapses. He assumed mutually reinforcing interactions of a circularly symmetric form, between synapses from the same eye, and similar inhibitory interactions between synapses from different eyes. Computer simulations of the development of this system showed that a wide variety of conditions incorporating these assumptions leads to the formation of striped projections.

An attempt to provide a more general framework expressed in terms of biologically plausible parameters that subsumes both these approaches was made by Miller, Keller and Stryker (1989); hereafter MKS. They proposed a model based on spatial correlations of activity within

[3] Note the similarity between this procedure and the later self-organizing algorithm due to Kohonen (1982).

each eye, along with an effective cortical interaction (local excitation with inhibition at longer range, as in the neural activity model). Rather than presenting patterns of activity to the retina as in the neural activity model, in MKS correlations between the activities of retinal cells are given explicitly in terms of a function that decreases with distance in the retina. Simulations and theoretical analysis showed the formation of ocular dominance stripes closely resembling those seen biologically.

2.3 TOPOGRAPHY AND OCULAR DOMINANCE

Constantine-Paton (1983) argues that the experimental evidence described earlier supports the notion that the mechanisms that normally produce retinotopic maps can also produce striped projections under certain conditions. However, others disagree: Udin and Fawcett (1988) state that "A single unifying mechanism cannot explain both topographic mapping and ocular dominance stripe formation". From a modeling point of view, the question is whether an account of topography can be extended to simultaneously explain ocular dominance, without requiring additional machinery. So far this goal has remained elusive. The problem appears to be how to provide information to the cortex about the eye of origin of each fibre.

One obvious possibility is that of a global chemical difference between the two eyes. von der Malsburg (1979) investigated this in an extension to the Tea Trade model, by assuming an extra chemical marker present in one eye but not the other. Although stripes were formed in this case, there is evidence against the existence of such global markers. Firstly, in the three-eyed frog experiments (Constantine-Paton and Law, 1978), stripes formed equally well from two equivalent eyes as from one right and one left eye. Secondly, in the isogenic eye experiments of Ide, Fraser and Meyer (1983), stripes were formed between two equivalent half eyes grown from a single eye fragment. However, it should be noted that these results rule out only genetic global chemical differences: the possibility still remains of such differences arising during development (G. Hinton, personal communication).

Local chemical differences could also provide the information necessary for stripe formation. Such an idea has been investigated by me in an extension to the Tea Trade model, where the same markers are present in both eyes, but with sources at slightly different locations in the two eyes. The idea was that fibres from the two eyes would then be "interleaved" in marker space, and that this pattern would be reflected in the cortex in the form of ocular dominance stripes. However, as yet I have not succeeded in producing stripes with this model.

Constantine-Paton (1983) discusses a model of both topography and stripe formation using a "biphasic" process. It is assumed that there are two competing interactions at work. Firstly, a weak graded affinity between retinal cells and tectal cells, perhaps implemented by shallow chemical gradients, serves to set up an initial crude order to the map. The second process is a fibre-fibre interaction between axons from neighbouring retinal cells that attempts to keep these fibres together. This causes the later refinement of the initial map. In the binocular case, it is supposed that stripes arise in this model as a compromise between both retinae attempting to map in a continuous fashion across the whole tectum (retino-tectal affinities), and each retina trying to maximize the degree to which its fibres terminate next to other fibres from the same eye (fibre-fibre affinities). The source of fibre-fibre interactions may come directly from chemical adhesion, but it could also be caused indirectly by activity-based processes. Since neighbouring retinal cells have strongly correlated activities, their synapses in the tectum stand most chance of being reinforced when fibres from both cells terminate in the same position (assuming a Hebbian updating rule). This has the effect of tending to cause fibres from neighbouring cells to terminate together. However, if there is at least some degree of binocular convergence, there will also be correlations in the activity of cells in corresponding positions in the two retinae. We will return to this point later.

A model of the "balancing" type described above has been investigated by Fraser (1985), in an extension to Fraser (1980). This is expressed in terms of parameters representing the strengths of the various interactions involved. Simultaneous topography and ocular dominance stripes were obtained by assuming a "Hebbian" attraction between fibres from neighbouring tectal cells. However, the nature of the correlated activity was not examined explicitly: its effect was simply modeled as a decrease in the parameter representing the strength of competition between axons for post-synaptic sites.

Two new approaches to the topography/ocular dominance problem will now be discussed, and their strengths and weaknesses compared. These are an extension to the neural activity model (Willshaw and von der Malsburg, 1976), and the elastic net approach (Goodhill and Willshaw, 1990).

3 THE BINOCULAR NEURAL ACTIVITY MODEL

I have investigated an extension of the neural activity model to the case of two eyes mapping to the same part of cortex, without assuming a pre-existing retinotopy as in von der Malsburg and Willshaw (1976). It is assumed that a local unstructured pattern of activity appears briefly in one retina or the other, but not both simultaneously.

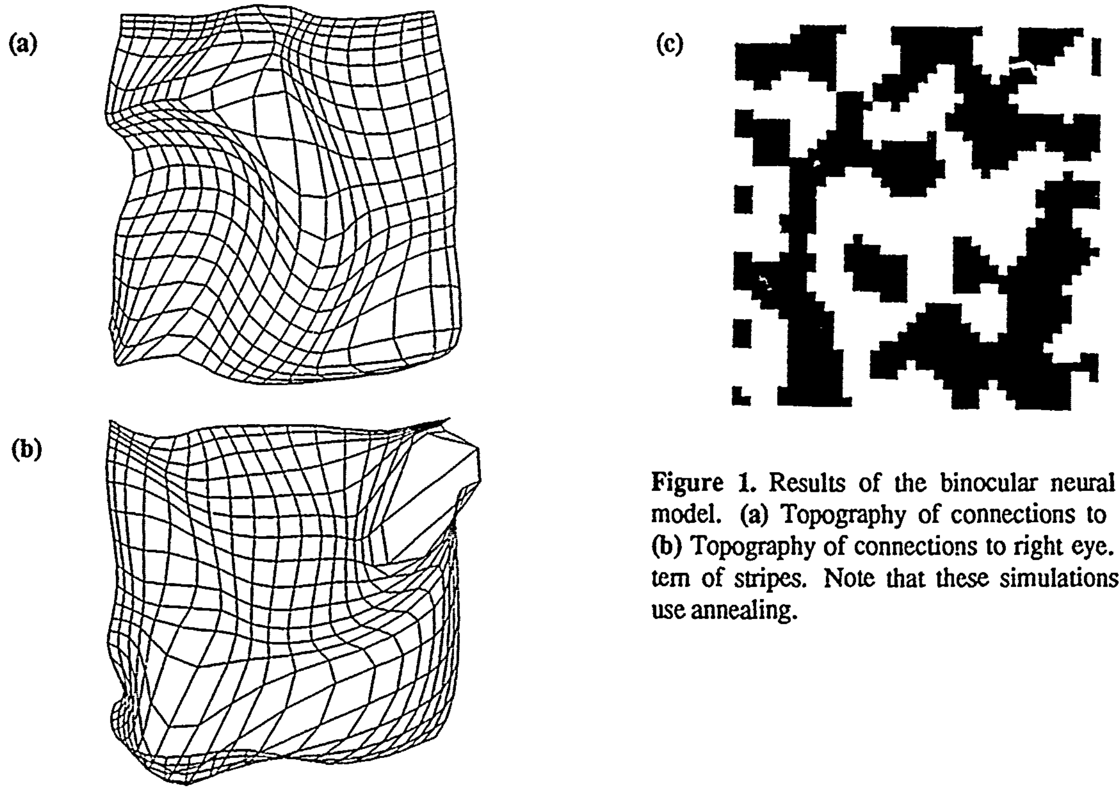


Figure 1. Results of the binocular neural activity model. (a) Topography of connections to left eye. (b) Topography of connections to right eye. (c) Pattern of stripes. Note that these simulations did not use annealing.

Such a regime is plausible before eye-opening, when only spontaneous activity is present (M. Stryker, personal communication). Here information about eye of origin is being supplied by activity, which is correlated within an eye but not between eyes. Initial simulations show that this model can produce both stripes and a topographic map simultaneously (figure 1).

I recently discovered that a similar model is being investigated on a larger scale and in more detail. Obermayer, Ritter and Schulten (1990) discuss a version of the Kohonen algorithm (Kohonen, 1982), or a "principle of continuous mapping", that is quite similar to the generalization of the neural activity model. In addition, they consider the simultaneous development of orientation columns, and present impressive results showing final patterns of orientation, topography and ocular dominance bearing close similarity to those seen biologically. However, an important difference between the two models is that Obermayer *et al* "anneal": that is, they gradually reduce the range of the intracortical interaction from an initially high value, whereas in the binocular neural activity model the intracortical interaction is a constant (low) value throughout a simulation. There is no doubt that annealing improves the quality of the results: however, it is difficult to interpret annealing biologically (see Discussion).

The main limitation of such models as discussed is that they only consider presentation of a single local pattern of activity at each time, whereas in the biological case topography and ocular dominance stripes can form in response to input across the whole retina and in both eyes simultaneously. What is lacking is a general framework for expressing the amount of correlation between corresponding cells in the two eyes, as compared to the amount of correlation between neighbouring cells in the same eye. Such a framework is present in the MKS model of ocular dominance, in the form of a function C^{same} expressing how correlation within a retina falls off with distance, and a function C^{diff} expressing how correlation between corresponding cells in the two retinae falls off with distance. However, this mcdel does not consider the origin of retinotopy, and results are only presented for the "strabismatic" case of zero or a slight negative correlation between the two eyes.[4] The only model to date which simultaneously addresses both topography and ocular dominance and which has been shown to work in the case of positive correlations between the two eyes is the elastic net approach proposed by Goodhill and Willshaw (1990); hereafter GW. This will now be described, and results presented which extend the original work to the case of monocular deprivation for this and a related algorithm.

[4] However, the fact that the MKS model uses additive, rather than multiplicative, normalization constraints means that positive correlations should not hinder stripe development (K. Miller, personal communication).

4 THE ELASTIC NET APPROACH

4.1 THE ELASTIC NET ALGORITHM

The elastic net algorithm is a recently-proposed technique for the approximate solution of the Travelling Salesman Problem (TSP) and other combinatorial optimization problems that have a geometric interpretation (Durbin and Willshaw, 1987; Durbin, Szeliski and Yuille, 1989). This has been shown to be related to the Hopfield and Tank (1985) approach to optimization (Simic, 1990; Yuille, 1990). It was inspired by the equations of the Tea Trade model described earlier. In Goodhill and Willshaw (1990) it was applied to the topography/ocular dominance problem and shown to be capable of solving both simultaneously.

The arrangement used is as shown in figure 2. This abstract geometrical layout of cells represents the correlational structure of retinal activity. The parameter d (see figure 2) measures the correlation between cells within a retina, while l measures the correlation between corresponding cells in the two retinae. The elastic "rope" moving in the plane of the cells is a representation of the cortex (output space) mapped onto the retinae (input space). In the limit the rope is continuous, but here is represented as a string of "beads". Each cortical cell (bead) $\mathbf{y}_j$ is assumed to have a gaussian receptive field: distances between retinal cells $\mathbf{x}_i$ and beads represent connection strengths. Beads obey the following equations of motion:

$$\Delta \mathbf{y}_j = \alpha \sum_i w_{ij} (\mathbf{x}_i - \mathbf{y}_j) + \beta k (\mathbf{y}_{j+1} - 2\mathbf{y}_j + \mathbf{y}_{j-1})$$

where $\mathbf{x}_i$ is the position of cell i, and α and β are scaling constants for the contributions from retinal cells and from neighbouring beads respectively. w_{ij} represents the normalized force of cell i on bead j:

$$w_{ij} = \frac{\Phi(|\mathbf{x}_i - \mathbf{y}_j|, k)}{\sum_p \Phi(|\mathbf{x}_i - \mathbf{y}_p|, k)}$$

where

$$\Phi(|\mathbf{x}_i - \mathbf{y}_j|, k) = \exp\left(\frac{-|\mathbf{x}_i - \mathbf{y}_j|^2}{2k^2}\right)$$

Note that the influence between each cell and each bead is modulated by an "annealing" parameter k, which starts at a high value and is gradually reduced over the course of a simulation. This corresponds to systematically shrinking the size of the receptive field for each cortical cell until finally each has a strong connection with only one retinal cell, and simultaneously reducing the lateral interaction between neighbouring points in the cortex. These equations can be integrated to produce an "energy function":

$$E = -\alpha k \sum_i \log \sum_j \Phi(|\mathbf{x}_i - \mathbf{y}_j|, k) + \tfrac{1}{2}\beta \sum_j |\mathbf{y}_{j+1} - \mathbf{y}_j|^2$$

It can be shown (Durbin, Szeliski and Yuille, 1989) that the first term is a maximum likelihood estimate of how well the model ($\mathbf{y}_j$'s) fits the data ($\mathbf{x}_i$'s). The second term is a "regularization" term that imposes prior assumptions on the model: in this case that in visiting all the cells the rope length is kept short.

This analogy with the TSP was used in Goodhill and Willshaw (1990) to calculate the stripe width n in terms of the parameters l and d that minimizes the total distance along the rope for a striped retinotopic 1-D map. This was found to be

$$n = 1 + \frac{l^2}{d^2}$$

It should be noted however that the algorithm actually minimizes the sum of *squares* of the distances between points (for discussion see Simic (1990)). Using this objective a similar analysis produces an optimum of

$$n = \frac{2l}{d}$$

In the sum-of-squares case n increases more slowly with $l \,/\, d$, which is more in keeping with simulation results.

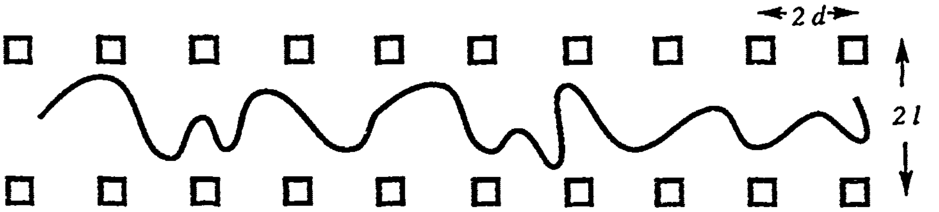


Figure 2. The arrangement of cells in the 1-D elastic net model. The two retinae are represented by two rows of squares (retinal cells) with a horizontal separation of 2d. The retinae are separated by a vertical distance of 2l. The rope represents the mapping to the cortex: the position of each point on the rope is related to the strength of its connection with each retinal cell.

4.2 RELATED ALGORITHMS

The arrangement of cells in figure 2 as a way of understanding the correlational structure of activity in the two retinae is independent of whether the elastic net or a similar algorithm is actually used to produce a mapping. Recently, the self-organizing algorithm of Kohonen (Kohonen, 1982) for mapping between spaces of different dimensionalities has been applied to the TSP (Angeniol *et al*, 1988; Fort 1988). In these algorithms, each retinal cell attracts only those points on the rope closest to it, rather than all the points as in the elastic net. These algorithms are appealing for application to the case of ocular dominance stripes because they are generally computationally cheaper. The following slightly modified version of the algorithm of (Angeniol *et al*, 1988), developed by this author in conjunction with Harry Barrow, was used in the current work. For the purposes of this paper I will refer to this as the BG (Barrow-Goodhill) algorithm.

$$\Delta y_j = \alpha \sum_i w_{ij} (x_i - y_j)$$

where

$$w_{ij} = \frac{\Phi(|j - j^*_i|, k)}{\sum_q \Phi(|j - j^*_q|, k)}$$

where j^*_i is the point on the rope that is closest to x_i, and $\Phi(|j - j^*_i|, k)$ is given by

$$\Phi(|j - j^*_i|, k) = \exp\left(\frac{|j - j^*_i|^2}{2k^2}\right)$$

Note that in this algorithm normalization of weights is over pre-synaptic cells, whereas in the elastic net it is over post-synaptic cells. Thus in BG the amount to which a cortical cell is adapted towards a retinal cell is modulated by the amount that cortical cell is being pulled towards other retinal cells, whereas in GW the modulation is determined by the amount that retinal cell is attracting other cortical cells.

4.3 RESULTS

Results are presented for the 2-D case, which is a straightforward generalization of the 1-D case shown in figure 2. The elastic rope becomes an elastic sheet lying between two sheets of retinal cells. As described in Goodhill and Willshaw (1990), initially the cortical sheet flattens between the two retinae. With no deprivation, this equilibrium position is halfway between the two retina, corresponding to points in the cortex receiving equal innervation from the two eyes. As k is reduced the sheet expands until it is a square the same size as a retina. At a certain critical value of k stripes form (see Goodhill and Willshaw (1990) for analysis).

The space parameters used for all simulations described here were d=0.02 and l=0.05. The cortical sheet contained 50×50 beads. There were 25×25 cells in each retina, as used in MKS.[5] For the GW case, the values of α , β and the initial value of k were as in Durbin and Willshaw (1987). Initial conditions of the sheet were random z position between the two retinae, and the x and y components random within a wide band, which biases the sheet to a particular orientation. The width of this band was here set to half the width of the array of cells. Periodic boundary conditions on the z component of the sheet were imposed for the GW case. The factor by which k is reduced at each step was set to 0.95 in both the GW and BG cases. The version of the elastic net algorithm used was that recently derived by Richard Durbin (personal communication). This is much faster than the version described above since it employs techniques from the EM algorithm (Dempster, Laird and Rubin, 1976).

4.3.1 Normal Development

Ocular dominance pictures for the normal case are shown in figures 3(a) and 3(d). Since in both the GW and BG cases each retinal cell can capture at most one point, and there are twice as many points in the cortical sheet as there are cells in both retinae, approximately half the points do not eventually lie atop a retinal cell. However, it was observed that when stripes form all points move a substantial distance away from the mid-plane towards one retina or the other, and thus the pictures in figure 3 were obtained by thresholding at half the separation between the two retinae. I will refer to the black stripes as those coming from the right eye.

4.3.2 Deprivation

As in MKS, monocular deprivation was modeled as a decrease in strength of one eye relative to the other (Chapman *et al*, 1986). This was achieved in both cases by defining two α parameters α_L and α_R, one for each eye. Results are shown in figure 3. As in the biological situation, there is a marked shift towards the open (in this case left) eye: a greater number of cortical cells can be driven by the left eye than the right eye.

There is also some biological evidence that disrupting the correlational structure of one eye can produce results similar to monocular deprivation. This could be simulated in the model being discussed here by increasing d for one retina relative to the other. Trial simulations of this case suggest that this indeed leads to stripes of different thicknesses for the two eyes.

[5] It should however be noted that the results presented in MKS show a grid of 40×40 cells, with 15 cells repeated in each dimension. This was to demonstrate that the periodic boundary conditions they employed do indeed lead to continuity across the borders.

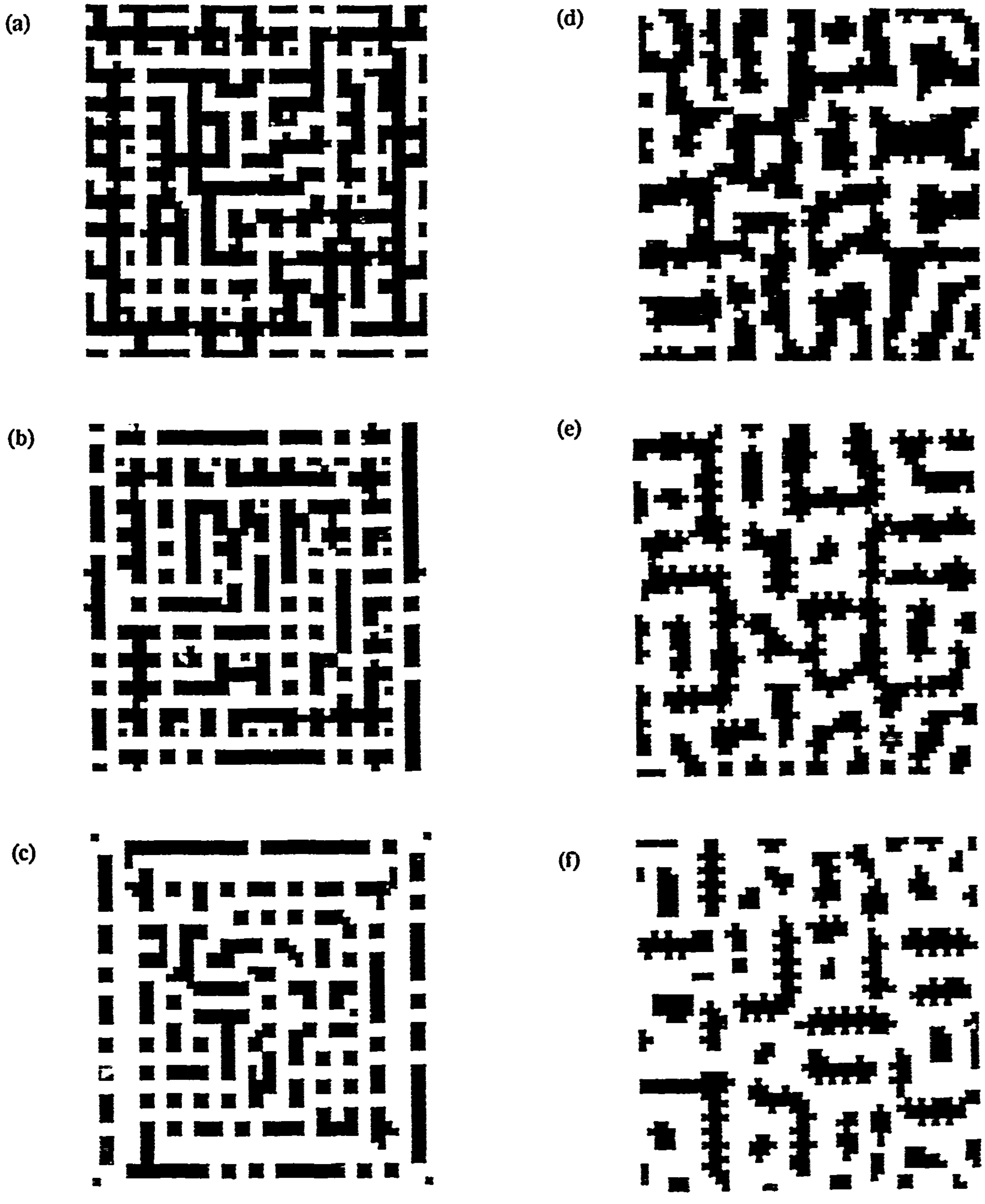


Figure 3. Normal and monocular deprivation results for the GW and BG models. (a) GW, normal development, $\alpha = 0.2$. (b) GW, slight deprivation, $\alpha_L = 1.2\,\alpha$, $\alpha_R = 0.8\,\alpha$. (c) GW, severe deprivation, $\alpha_L = 1.4\,\alpha$, $\alpha_R = 0.6\,\alpha$. (d) BG, normal development, $\alpha = 0.2$. (e) BG, slight deprivation, $\alpha_L = 1.2\,\alpha$, $\alpha_R = 0.8\,\alpha$. (f) BG, severe deprivation, $\alpha_L = 1.4\,\alpha$, $\alpha_R = 0.6\,\alpha$. For discussion see text.

It is interesting to compare results from the two algorithms (see figure 3). In the case of normal development, BG produces a pattern more closely resembling that seen biologically. GW appears more geometrical, which may be because it produces a better solution to the optimization problem than BG. This raises the issue of whether the biological problem can be understood purely as an abstract optimization without also considering the algorithm actually used to perform that optimization. For the deprivation results, note that in GW the stripes from the deprived eye become more patchy while not decreasing appreciably in width. In BG, the stripes do become thinner and remain fairly continuous. However, much larger scale simulations are required before firm conclusions can be drawn about the qualitative differences between GW, BG, and the biological case. In addition, it should be noted that both GW and BG are discrete versions of continuous cases. With the very geometrical layout of cells used in these experiments such discretization may be causing unwanted effects.

5 DISCUSSION

A difference between the GW/BG approach and the binocular neural activity model is that the former is an abstract optimization model and is hard to compare directly with biological reality. Although the parameters of the binocular neural activity model have more biological relevance, this model has as yet only been shown to work in a highly specialised case. However, it has the advantage of not requiring annealing.

A problem encountered by GW and any model involving annealing (as in Obermayer, Ritter and Schulten (1990)) is in explaining the reverse suturing experiments described earlier. Biologically, it is in fact possible to change which set of stripes is thicker several times during the critical period, by appropriate opening and closing of each eye. In addition, due to different modes of growth of retina and tectum in amphibians and fish connections are being continually shifted during development. However, in annealing-based models a pattern of stripes only forms at the end of the process, and this pattern is not then susceptible to further change. Thus there are problems in naively interpreting the annealing period as corresponding to the critical period. Annealing certainly improves the quality of the solution found to the optimization problem, but it may be the case that the brain employs other "good enough" techniques that do not use annealing (G. Hinton, personal communication).

It provides useful insights into the GW approach to compare it with the MKS model (even though this model does not address the development of topography). Although a detailed mathematical comparison has not yet been performed, some general comments can be made (see also Goodhill and Willshaw (1990)). The most important thing to note is that the two models make radically different predictions regarding which parameters determine the width of the stripes. In MKS, it is the peak of the Fourier transform of the cortical interaction function, i.e. a property only of the cortex (not, as incorrectly stated in Goodhill and Willshaw (1990), a function of the retinal correlations). However in GW, it is the correlational structure of retinal activity that determines stripe width. Two intuitive arguments can be given regarding the latter view, one in agreement and one in contradiction.

Firstly, although the stripe width is uniform in the cortex, when this pattern is mapped onto the retina the width becomes highly non-uniform (for a picture of this see LeVay *et al*, 1985). The stripes are narrow in the fovea and wide at the periphery, with a smooth variation of widths in between. Since in general an animal's gaze is focussed at the fovea in both eyes, this is where there will be most correlation between the two sets of retinal cells. In GW terms this means l is small, predicting thin stripes as seen biologically. At the periphery, the 2 images will be less in registration, i.e. l is large and GW predicts wide stripes as seen biologically. Care must be taken with this argument, since the density of receptors in the retina is highly non-uniform.

However, the argument against this view comes from the artificial strabismus experiments. Decorrelating the 2 images in GW terms corresponds to making l very large, hence GW predicts very wide stripes. However, in the biological case the width of the stripes is unaffected.

It is worth mentioning that there is some evidence that in reality in cats and monkeys, the development of a retinotopic map, orientation columns and ocular dominance stripes is a 3-stage process (M. Stryker, personal communication, C. von der Malsburg, personal communication). A crude topographic map forms very early, before birth, followed by rough orientation columns. Ocular dominance does not form until much later, during which time the map and orientation columns sharpen and become more precise. However, not enough is known of the detailed time course of these developments (due largely to the difficulty of performing experiments *in utero*) to assess the implications for the type of unified model being advocated in this paper.

6 CONCLUSIONS

The problem of the simultaneous development of topography and ocular dominance stripes has not been entirely solved. Although the models discussed in this paper appear promising, they all have limitations that I have attempted to outline. More work needs to be done on the mathematical relationship between the different approaches, and on how the parameters of each relate to biological parameters. Finally, more biological data needs to be obtained in order to further constrain computational models. Useful domains for such experimental investigations include the form of lateral interactions in the cortex, and the actual correlations in nerve firing in normal illumination.

Acknowledgements

I thank both Harry Barrow and David Willshaw for the many discussions that inspired this work, and for helpful comments on an earlier draft of this paper. I am very grateful to Geoff Hinton for several discussions regarding the problem of ocular dominance, and for his perceptive editing of this paper. I am also grateful to everyone at the Summer School for creating the stimulating atmosphere in which these ideas were refined and developed: in particular Richard Durbin and Jack Cowan. Finally, thanks to Ken Miller for many useful conversations.

References

Angeniol, B., de la Croix Vaubois, G. and Le Texier, J. (1988). 'Self-organizing feature maps and the traveling salesman problem'. *Neural Networks*, **1**, 289-293.

Archer, S.M., Dubin, M.W. and Stark, L.A. (1982). 'Abnormal development of kitten retino-geniculate connectivity in the absence of action potentials'. *Science*, **217**, 743-745.

Arnett, D.W. (1978). 'Statistical dependence between neighbouring retinal ganglion cells in goldfish'. *Exp. Brain. Res.*, **32**, 49-53.

Barrow, H.G. (1987). 'Learning Receptive Fields' *Proc. IEEE First Annual Conference on Neural Networks.*, **IV**, 115-121.

Boss, V.C. and Schmidt, J.T. (1984). 'Activity and the formation of ocular dominance patches in dually innervated tectum of goldfish'. *Journal of Neuroscience*, **4**, 2891-2905.

Chapman, B., Jacobson, M.D., Reiter, H.O. and Stryker, M.P. (1986). 'Ocular dominance shift in kitten visual cortex caused by imbalance in retinal electrical activity'. *Nature*, **324**, 154-157.

Constantine-Paton, M. (1983). 'Position and proximity in the development of maps and stripes'. *Trends Neurosci.*, **6**, 32-36.

Constantine-Paton, M., Cline, H.T. and Debski, E. (1990). 'Patterned activity, synaptic convergence, and the NMDA receptor in developing visual pathways'. *Annu. Rev. Neurosci.*, 13, 129-154.

Constantine-Paton, M. and Law, M.I. (1978). 'Eye-specific termination bands in tecta of three-eyed frogs'. *Science*, 202, 639-641.

Cook, J.E. and Rankin, E.C.C. (1986). 'Impaired refinement of the regenerated retinotectal projection of the goldfish in stroboscopic light: a quantitative WGA-HRP study'. *Exp. Brain. Res.*, **63**, 421-430.

Cowan, JD and Friedman, AE (1990). 'Development and regeneration of eye-brain maps: A computational model'. In D.S. Touretzky, ed, *Advances in Neural Information Processing Systems II*, 92-99.

Cronly-Dillon, J.R. and Glaizner, B. (1974). 'Specificity of regenerating optic fibres for left and right optic tecta in goldfish'. *Nature*, **251**, 505-507.

Dempster, A.P., Laird, N.M. and Rubin, D.B. (1977). 'Maximum likelihood from incomplete data using the EM algorithm'. *Proc. Roy. Statistical. Soc. B*, **39**, 1-37.

Durbin, R. and Mitchison, G. (1990). 'A dimension reduction framework for understanding cortical maps'. *Nature*, **343**, 644-647.

Durbin, R., Szeliski, R. and Yuille, A. (1989). 'An analysis of the elastic net approach to the traveling salesman problem'. *Neural Computation*, **1**, 348-358.

Durbin, R. and Willshaw, D.J. (1987). 'An analogue approach to the traveling salesman problem using an elastic net method'. *Nature*, **326**, 689-691.

Fawcett, J.W. and O'Leary, D.D.M. (1985). 'The role of electrical activity in the formation of topographic maps in the nervous system'. *Trends Neurosci.*, **8**, 201-206

Fawcett, J.W. and Willshaw, D.J. (1982). 'Compound eyes project stripes on the optic tectum in Xenopus'. *Nature*, **296**, 350-352.

Fort, J.C. (1988). 'Solving a combinatorial problem via self-organizing process: an application of the Kohonen algorithm to the traveling salesman problem'. *Biol. Cybern.*, **59**, 33-40.

Fraser, S.E. (1980). 'A differential adhesion approach to the patterning of neural connections'. *Dev. Biol.*, **79**, 453-464.

Fraser, S.E. (1985). 'Cell interactions involved in neural patterning'. In *Molecular Bases Of Neural Development*, eds. G.M. Edelman, W.E. Gall and W.M. Cowan, 481-507, Wiley, New York.

Freeman, R.D., Sclar, G. and Ohzawa, I. (1982). 'Cortical binocularity is disrupted by strabismus more slowly than by monocular deprivation'. *Dev. Br. Res.*, **3**, 311-316.

Goodhill, G.J. and Willshaw, D.J. (1990). 'Application of the elastic net algorithm to the formation of ocular dominance stripes'. *Network*, **1**, 41-59.

Gottlieb, D.I. and Glaser, L. (1981). 'Cellular recognition during neural development'. In *Studies in Developmental Neurobiology*, ed. W.M. Cowan, 243-260, Oxford University Press, New York.

Hopfield, J.J. and Tank, D.W. (1985). 'Neural computation of decisions in optimization problems'. *Biol. Cybern.*, **52**, 141-152.

Hubel, D.H. and Wiesel, T.N. (1965). 'Binocular interaction in striate cortex of kittens reared with artificial squint'. *Journal of Neurophysiology*, **28**, 1041-1059.

Hubel, D.H. and Wiesel, T.N. (1977). 'Functional architecture of the macaque monkey visual cortex'. *Proc. R. Soc. Lond. B*, **198**, 1-59.

Hubel, D.H., Wiesel, T.N. and LeVay, S. (1977). 'Plasticity of ocular dominance columns in monkey striate cortex'. *Phil. Trans. R. Soc. Lond. B*, **278**, 377-409.

Ide, C.F., Fraser, S.E. and Meyer, R.L. (1983). 'Eye dominance columns formed by an isogenic double-nasal frog eye'. *Science*, **221**, 293-295.

Jacobson, M. and Levine, R.L. (1975). 'Plasticity in the adult frog brain: filling the visual scotoma after excision or translocation of parts of the optic tectum'. *Brain Res.*, **88**, 339-345.

Kohonen, T. (1982). 'Self-organized formation of topclogically correct feature maps'. *Biol. Cybern.*, **43**

Law, M.I. and Constantine-Paton, M. (1980). 'Right and left eye bands in frogs with unilateral tectal ablations'. *Proc. Natl. Acad. Sci., U.S.A.*, **77**, 2314-2318.

LeVay, S., Connolly, M., Houde, J. and Van Essen, D.C. (1985). 'The complete pattern of ocular dominance stripes in the striate cortex and visual field of the macaque monkey'. *Journal of Neuroscience*, **5**, 486-501.

LeVay, S., Wiesel, T.N. and Hubel, D.H. (1980). 'The development of ocular dominance columns in normal and visually deprived monkeys'. *Jou. Comp. Neurol.*, **191**, 1-51.

Lund, J.S. (1988). 'Anatomical organization of macaque monkey striate visual cortex'. *Ann. Rev. Neurosci.*, **11**, 253-288.

Malsburg, C. von der (1979). 'Development of ocularity domains and growth behaviour of axon terminals'. *Biol. Cybern.*, **32**, 49-62.

Malsburg, C. von der and Willshaw, D.J. (1976). 'A mechanism for producing continuous neural mappings: ocularity dominance stripes and ordered retino-tectal projections'. *Exp. brain. Res. Supplementum 1*. 463-469.

Malsburg, C. von der and Willshaw, D.J. (1977). 'How to label nerve cells so that they can interconnect in an ordered fashion'. *Proc. Nat. Acad. Sci., U.S.A.*, **74**, 5176-5178.

Mastronade, D.N. (1989). 'Correlated firing of retinal ganglion cells'. *Trends Neurosci.*, **12**, 2, 75-80.

Meyer, R.L. (1982). 'Tetrodotoxin blocks the formation of ocular dominance columns in goldfish'. *Science*, **218**, 589-591.

Meyer, R.L. (1983). 'Tetrodotoxin inhibits the formation of refined retinotopography in goldfish'. *Dev. Brain. Res*, **6**, 293-298.

Miller, K.D., Keller, J.B. and Stryker, M.P. (1989). 'Ocular dominance column development: analysis and simulation'. *Science*, **245**, 605-615.

Munro, P.W. (1986). 'State-dependent factors affecting neural plasticity: a partial account of the critical period'. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.*, eds. J.L. McClelland and D.E. Rumelhart, 471-503, MIT Press, Cambridge, M.A.

Obermayer, K., Ritter, H. and Schulten, K. (1990). 'A principle for the formation of the spatial structure of cortical feature maps'. *Proc. Nat. Acad. Sci., U.S.A.*, In Press.

Prestige, M.C. and Willshaw, D.J. (1975). 'On a role for competition in the formation of patterned neural connexions'. *Proc. R. Soc. Lond. B*, **190**, 77-98.

Reh, T.A. and Constantine-Paton, M. (1985). 'Eye-specific segregation requires neural activity in three-eyed Rana pipiens'. *Journal of Neuroscience*, **5**, 1132-1143.

Schmidt, J.T. (1978). 'Retinal fibres alter positional markers during the expansion of the half retinal projection in goldfish'. *Jou. Comp. Neurol.*, **177**, 279-295.

Schmidt, J.T. and Edwards, D.L. (1983). 'Activity sharpens the map during the regeneration of the retinotectal projection in goldfish'. *Brain Research*, **269**, 29-39.

Schmidt, J.T. and Eisele, L.E. (1985). 'Stroboscopic illumination and dark-rearing block the sharpening of the regenerated retinotectal map in goldfish'. *Neuroscience*, **14**, 535-546.

Shatz, C.J. and Stryker, M.P. (1978). 'Ocular dominance in layer IV of the cat's visual cortex and the effects of monocular deprivation'. *Jou. Physiol.*, **281**, 267-283.

Simic, P.D. (1990). 'Statistical mechanics as the underlying theory of 'elastic' and 'neural' optimizations'. *Network*, **1**, 89-103.

Sperry, R.W. (1963). 'Chemoaffinity in the orderly growth of nerve fiber patterns and connections'. *Proc. Nat. Acad. Sci., U.S.A.*, **50**, 703-710.

Stryker, M.P. and Harris, W. (1986). 'Binocular impulse blockade prevents the formation of ocular dominance columns in cat visual cortex'. *Journal of Neuroscience*, **6**, 2117-2133.

Swindale, N.V. (1980). 'A model for the formation of ocular dominance stripes'. *Proc. R. Soc. Lond. B*, **208**, 243-264.

Swindale, N.V. (1981). 'Absence of ocular dominance patches in dark-reared cats'. *Nature*, **290**, 332-333.

Udin, S.B. and Fawcett, J.W. (1988). 'Formation of topographic maps'. *Ann. Rev. Neurosci.*, **11**, 289-327.

Whitelaw, V.A. and Cowan, J.D. (1981). 'Specificity and plasticity of retinotectal connections: a computational model'. *Journal of Neuroscience*, **1**, 1369-1387.

Willshaw, D.J. and Malsburg, C. von der (1976). 'How patterned neural connections can be set up by self-organization'. *Proc. R. Soc. Lond. B*, **194**, 431-445.

Willshaw, D.J. and Malsburg, C. von der (1979). 'A marker induction mechanism for the establishment of ordered neural mappings: its application to the retinotectal problem'. *Phil. Trans. Roy. Soc. B*, **287**, 203-243.

Yuille, A.L. (1990). 'Generalized deformable models, statistical physics, and matching problems'. *Neural Computation*, **2**, 1-24.

On Modeling Some Aspects of Higher Level Vision

David Bennett
Cognitive and Linguistic Sciences
Brown University
Providence, R.I. 02912
Bennett@Browncog.Bitnet

Abstract

This paper outlines several intertwined modeling projects concerning various aspects of higher level vision. A model of visual priming is presented, followed by an account of "top down" visual expectation or hypothesis testing as a kind of self-priming, closely associated with the calling up of visual imagery. Several preprocessing mechanisms are then introduced that might account for the relative ease with which we can identify objects and patterns independently of viewpoint.

The following is a sketch of several ongoing, intertwined modeling projects concerning various aspects of higher level vision. All of the work referred to must be considered preliminary, and I only provide general characterizations of the results obtained thus far. But the basic, qualitative behavior of the nets described seems both psychologically interesting and–as the simple consequence of simple architectures–quite robust. And there may as well be an advantage in here taking the long view of several complimentary projects, for this will allow me to outline, illustrate, and begin to argue for an approach to modeling higher-level vision, and to cognitive modeling in general, that is different in important respects from the approach taken in most contemporary modeling (one exception is James Anderson, whose modeling work and theoretical musings have exerted a great deal of influence over what follows).

The modeling reported assumes that fast, spontaneous, "basic level" recognition is mediated by a global, direct-matching process, where the input is compared to a viewer-centered representation that is continually evolving with experience. The primary aim, thus far, has been to model various pre and postprocessing mechanisms designed to guide and facilitate this matching process. But I will begin by both briefly defending the plausibility of the foregoing central assumption and by explaining how I have thus far been implementing the matching process. This will also help to specify the precise psychological capacity that is being modeled. Following this, I discuss a model of visual priming. It is then shown how "top down" visual expectation or hypothesis testing can be modeled as a kind of self-priming, closely associated with the calling up of visual imagery. Next, several preprocessing mechanisms are introduced that might account for the relative (though by no means uniform) ease with which we can identify objects and patterns which are at differing locations in our visual field, which have different orientations, and which project different-sized retinal images. Finally, I close with a brief remark on the problem of image segmentation, suggesting that a useful lesson might be drawn from the discussion of the foregoing higher-level preprocessing mechanisms.

1. BACKGROUND VIEWS AND ASSUMPTIONS

It appears that first, fast, visually prompted contact with memory generally occurs at a level of categorization that is, by a number of different measures, psychologically "basic". Thus, basic level categorization is said to constitute the "entry point" (Jolicoeur, Gluck, and Kosslyn, 1984) or "first access" (Biederman, 1987) to memory. (The details here are complex and the literature enormous; in addition to the foregoing, see Rosch et al., 1976; Smith and Medin, 1981; Murphy and Brownell, 1985, Lakoff, 1987.) Such categories are psychologically closely associated with the characteristic outline or shape of the object when seen from a preferred or "canonical" point of view. And it appears that initial identification is shape-based (in addition to the previously cited works, see Palmer et al., 1981, Biederman and Ju, 1988). The number of basic level categories is unclear, but after considering various estimates and their rationale, Biederman (1987) settles on 1600 as a rough estimate.

A widely held view is that the image of the object is first parsed into its parts in the course of forming a representation in object centered coordinates (Biederman, 1987; Hoffman and Richards, 1984; Marr, 1982). But there is evidence that human subjects do not proceed in this way. Edelman and Bülthoff (1990) examined how subjects who were exposed, in a carefully controlled way, to different views of novel items (shaded wire figures) generalized to new presentations. They found that subjects appeared to compare the input to *viewer-centered,* largely 2-D representations which evolved in a way that closely corresponded to the history of their experiences. In closely related work with similar stimuli, Edelman, Bülthoff, and Weinshall (1989) found that while subjects began with certain preferred or canonical views–as reflected in both subjective judgments and reaction times--this effect diminished with experience in a way that suggested a decreasing reliance on preprocessing mechanisms which serve to align the image with a memory representation, and which are at least closely akin to mechanisms underlying mental imagery (see also Tarr and Pinker, 1989).

The idea of *first* access should be stressed. It is, typically, perhaps less recognition, than the fast generation of an initial hypothesis–a hypothesis which is associated with certain background knowledge and search routines which drive further exploration, and perhaps the subsequent application of more fine grained templates associated with the perceptually salient and functionally important parts or features of objects (compare Nakayama, 1987). It also can't be emphasized too strongly that humans use a variety of perceptual strategies depending upon the task at hand, and the route to memory that I describe is, while preferred when the conditions permit, still only one of several (for a discussion of some of the strategies used, see Ullman, 1986).

Finally, the system described in the following sections is to be considered to take deep, cortical input, carrying fairly refined shape and surface information. This, of course, assumes that an independent account can be given of how to get to the required point, and there are many active research programs engaged in various aspects of the problem (sometimes unabashed buck-passing is the rational thing to do). The work cited earlier suggests that it is edge and contour information which is of primary importance in the first pass through the recognition system. But it is pointed out in the next paragraph that there is reason to think that depth and motion information is used in coming to represent and select edges and contours for purposes of higher level matching.

Keeping in mind the refined nature of the information available for higher level matching is important in evaluating some of the standard, textbook examples alleged to show the hopelessness of the current approach. Thus, e.g., while the letters "O" and a "Q" don't differ much in appearance, their deep neural coding is dramatically different (with all of the corners and end-stops in the latter). Conversely, Nakayama et al. (1989) argue that some apparently quite severe occlusions of objects need not be especially disruptive of the deep neural coding available for later template matching because of the action of long-range filling–which is largely data driven and spontaneous, the result of the operation of fairly peripheral neural mechanisms in which the reliable coding of depth is crucial. More generally, it appears that, in segregating candidate, meaningful objects from the background clutter, the organism uses a complex bag of tricks that operate on fairly refined information about the disposition of the local surfaces and contours (see Nakayama et al., 1989, as well as the passage from Livingston and Hubel, 1988, presented in section 5 below). I will, in any case, assume that the recognition system receives segregated "neural images"–though possibly displaced, rotated, distorted, and of different sizes–corresponding to distinct objects. I'll return briefly to the problem of image segmentation in closing.

It must be conceded that the foregoing constitutes less an argument than an outline of a view and a brief sketch of how it might be defended. But if the general conclusions turn out to be correct, then the task facing the recognition module is, while still difficult, more manageable than it may at first have appeared: there are a restricted number of categories to consider, the image is already segmented, and only 2-D representations must be stored and compared.

In what follows I will work through the construction of a recognition module by, first, starting with a simple matching mechanism, and then adding various pre and postprocessing mechanisms–each corresponding to various capacities revealed in the experimental literature, and each adding to the power of the system as a whole. (A schematic diagram of the essentials of the whole architecture is presented at the end of the paper.) For a matching mechanism, the simplest of associative nets has been used. the linear associator, or slight variants, with hebbian learning (error correction, in the form of the delta rule, was only used, occasionally, for purposes of comparison). As indicated earlier, I am not presently mainly concerned with the issue of how to implement the matching process. (And, indeed, various alternatives will soon be added and explored.) But it may well be that the reports of the inadequacy of such simple associative nets have–at least for some tasks–been premature, or at least inflated (for a discussion see Anderson et al., 1990). Such nets have, in any case, certain virtues that other candidates are fruitfully measured against. Thus, in addition to the obvious appeal of their simplicity and (at least compared with many other nets currently in service) biological plausibility[1], they also scale fairly well, and support one-trial learning. And humans do exhibit one-trial learning, and they do have very, very big brains (a fact often alluded to, but rarely taken seriously in actual modeling of cognitive function). Moreover the behavior of such simple nets is largely shaped by input coding–the richer, more fine-grained, and appropriate the input representations, the finer and more psychologically realistic the computations performed (Anderson et al., 1990). In this vein, it is worth noting that the matching process, as here conceived, is set very deep and fed by a number of simple but (at least collectively) powerful preprocessing mechanisms, which serve to sharpen, refine, and prepare the input coding--both those discussed explicitly in sections 2-4, as well as more peripheral mechanisms only briefly alluded to in passing.

2. PERCEPTUAL PRIMING

It is well known that prior exposure to an object or pattern can lead to faster and more accurate identification of the same, or similar, object or pattern at a later time. Several general conclusions can be drawn from the many studies of such perceptual priming (cf. Bartram, 1974; Biederman and Cooper, 1990; for a comprehensive review, see Schacter et al., 1990). To begin, some priming is nonverbal–it results from the action of the visual stimulus, and not simply from the elicitation of the name of the object, covertly or otherwise. However, it does appear that, at least if the object has a name and the subject is encouraged to apply it, some of the effects of priming are, in the foregoing sense, verbal. Further, though this is more controversial, it appears that it is possible to produce priming effects by simply providing the name of an object prior to its visual presentation. Finally, the strength of the priming effect is primarily determined by two factors. First, the more similar the test to the prime, the greater the effect. Second, the shorter the delay, the stronger the prime–at least up to a point, when the entire effect seems to be the result of long term learning.

These central effects can be accounted for if we change and add to our core, linear-associator in accordance with the following three assumptions–which will also play a role in the modeling of further effects discussed in the next section. (The model presented here is indebted to McClelland and Rumelhart, 1986, though it differs form their account of priming in important respects–e.g., they pair the label with the image in a *single* vector which is then autoassociated.)

The first assumption is that the priming effects result from well-known transient and long-term changes in synaptic efficacy. (For reviews, see Brown et al., 1989; Sejnowski et. al., 1989.) In the simulations, the nets were trained, the weights were normalized, the prime was presented, and the connections were strengthened by an amount proportional to pre-and-post synaptic activity. A priming constant, which could vary, controlled the amount of change. This constitutes only a fairly coarse approximation to the biology; the modest aim, here and in the next section, was simply to illustrate the possible connection between certain basic features of our mental life and certain general assumptions about high-level, biological properties.

The second assumption is that the output of the two-layer associative net is fed into an auto-associative net (in the simulations I used Anderson's BSB auto-associative net, or slight variants; cf., Anderson et al., 1977). During training, the pattern corresponding to the symbolic tag or label is impressed upon both the output units of the two-layer net–and so directly associated with the input pattern–as well as autoassociated. The idea is that this would typically correspond to an episode of ostensive learning, but it might also result from the "internal" generation of a label applied to help provide later access, in the way described in section 3 below, by higher level control structures to the visual characteristics of an object or kind of object.[2] Then, later, after the learning or training period has ended, the output of the two-layer net is fed directly to the autoassociative net. Why might such an arrangement have evolved? In part, to help clean-up, clarify, and standardize the often imperfect output of the feedforward net (an effect easily demonstrated in simulations). Perhaps even more important, adding the autoassociator allows the system to hold on to the output–here, in order to continue to feed activation into semantic memory. Our thoughts do not disappear with the withdrawal of the stimulus.

[1] For ease of exposition, I will sometimes blur the distinction between model and reality by using terms like "synapse" to refer to both real wetware as well to elements of the models presented. But difficult conceptual and philosophical problems lurk here and need to be addressed (Bennett, 1990).

[2] I think that it may be possible to expand this latter suggestion into a scheme that allows for periods of unsupervised learning (not just in the trivial sense of involving the self-generation of a symbolic tag, but also in allowing for the capacity to categorize to develop in the absence of a tutor, partly driven by the structure of the input). Or one could change the matching mechanism. The general point is that there are a number of moves that might be made here to allow for some unsupervised learning, all of which appear to be relatively independent of the discussion of pre and postprocessing mechanisms to follow.

The final assumption is that a natural interpretation of reaction times in systems that realize dynamical systems is the number of iterations for an input vector to migrate into a point attractor (for a fuller discussion and somewhat different examples, see Anderson, to appear 1991). In addition to giving a simple measure, looking to the time to settle provides an explanation of *what* intervening process takes the time and *why*.

The central, qualitative features of visual priming noted above fall out of the foregoing scheme, and have been illustrated in a number of simulations. Thus, there are two sets of synapses, with changes in synaptic strength on the first corresponding to visually driven priming, while activity on the second leads to the separate priming of the tag or label (which can be manipulated independently–giving an account of how the presentation of the name alone might lead to faster response times). Manipulating the amount of change in the weights leads to differences in the time to settle–corresponding, perhaps, to the changes in reaction time exhibited by human subjects as the time between the prime and the test stimulus increases. And, as in the experiments with human subjects, if different input vectors are associated with the same name vector, the degree to which the presentation of one primes the second is determined by their similarity. There is, further, an interesting suppression in the response to other, unprimed categories. And when multiple categories are primed there is an attenuation of the effects of the priming on responses to exemplars of specific categories.

3. EXPECTATION AS IMAGINATIVE SELF-PRIMING

The foregoing pertains to facilitation that results from *external* stimulation. But it is a small step from these observations to an account of top-down visual expectation or hypothesis testing in terms of a kind of imaginative *self*-priming–a step that only requires making a simple, further addition to our modest network.

Though the details are both controversial and differ from case to case, it is well known that background knowledge can influence perceptual judgment. Bruner (1957), in a famous article, surveys a mass of experimental data and summarizes the basic phenomena that we would like to see mirrored and explained by a model of perceptual expectation: "To put it in ...ordinary language. apples [i.e., the expected object] will be more swiftly and easily recognized, a wider range of things will be identified or misidentified as apples, and in consequence the correct or best fitting identity of these other inputs will be masked." (And he later adds that "perceptual readiness or accessibility serves two functions. to minimize the surprise value of the environment by [flexibly] matching category accessibility to the probabilities of events in the world about one, and to maximize the attainment of sought-after objects and events.") These effects can be modeled by making the core, associative net bidirectional by simply adding a second, reversed, linear-associator which, during training, pairs the label as input with the incoming image as output. The label-vector can then be used to "call up" the image vector, which can be fed through the front end of the second associative net, thereby priming the system in the way discussed above.[1]

It is true that the foregoing scheme does not, as it stands, provide a close fit to the details of the human data on mental imagery (cf. Kosslyn et al., 1984; for evidence supporting the assumption that mental imagery exploits the same biological machinery as is employed during regular visual perception, see Kosslyn, 1987). However, first, it should be pointed out that an account of the flexible manipulation of images will be offered in section 4, where the manipulations are held to be carried out on the same preprocessing mechanisms that perform image alignment during recognition. And there may as well be a way of expanding the present system of storing and generating images to make it more psychologically realistic. The basic idea would be to model all of the more fine-grained icons–which detail parts and the like and which are associatively linked to both the basic level representation as well as to each other–with bidirectional associative nets of the sort discussed above. We might then construct a hybrid system, with the basic principles governing the retrieval, construction, and manipulation of images embodied in rules that access the images by their labels. But I have not yet attempted to implement this scheme.

In any case, the central results of such imaginative self-priming, as well as their qualitative similarity to Bruner's characterization of the effects of expectation, are obvious from the previous discussion of the simulations of visual

[1] Here and elsewhere it was necessary to both normalize the weights and , often, to scale the output , so that the different modules could "communicate" properly. A biologically more realistic way to achieve the same ends would be by using, in the two-layer associators, some sort of nonlinear activation function that incorporated a kind of automatic gain control. The relative sizes of the nets as well as the density of the connections could also be manipulated. The resulting nets would be non-linear, but presumably their general, qualitative behavior would mirror that of the current nets in the relevant respects, at least while they were operating in the (roughly) linear range.
Further, all of the detailed simulations of self-priming have, thus far, used random vectors for "images". However, the programs are written so that the simple, but coherent and recognizable images used in the image-normalization simulations discussed in section 4 could be used here as well.

priming and won't be belabored. However, it is worth pointing out a couple of especially interesting features of the self-priming simulations. First, the enhancement and suppression of reaction times noted previously show up as well in measures of the accuracy of the response. Most interesting, for present purposes, is the increase in the kind of over-generalization described by Bruner that accompanies stronger self-priming: that is, if the category structure is kept constant (i.e., the exemplars are formed by the same kind and degree of distortion around a central prototype), and the strength of expectation/priming is increased, then there is an increasing tendency for members of other categories to be sucked into the attractor associated with the expected category-while at the same time reaction time goes down. This illustrates that there are costs as well as benefits of relying upon expectation. Further, the generation of the image-vector exhibits a prototype effect: under certain conditions (see Knapp and Anderson, 1984 for a relevant general discussion), the image-vector that is "called up" is closer to an unseen protoype than to the exemplars that the system was trained on (which were formed by distorting the prototype). And it might be conjectured that something like this process underlies the impression that our mental images of often encountered, familiar *kinds* of things don't seem to correspond to any *particular* thing--while still having the right, general "look" and "feel".

It is important to appreciate that as long as the input is well-behaved and the memory demands are modest, the present system will work fine standing alone, without the benefit of priming or of higher level expectation. And, indeed, it may well be a general principle of the architecture of the mind that there are a number of specialized, peripheral mechanisms that work fine without intervention, but which are periodically "taken over" by higher level control centers by adapting the same mechanisms to the special needs of the moment. If so, then our models should specify precisely where, how, and with what effect such intervention might take place. In the foregoing simulations, the intervention serves to bias the recognition process. But such imaginative self-priming might also be held to reach back to influence the formation of the perceptual representation itself (perhaps exploiting well known feedback connections found throughout the visual system).

4. THE IMAGE NORMALIZATION PROBLEM

Projected images of objects-and so, their refined, neural representations-can be distorted, displaced, rotated, or of different sizes. I will speak first, and in the most detail, about a preprocessing mechanism to help correct for differences in orientation. But I will also add a few remarks about related schemes to handle spatial displacement and changes of size. Distortion, on the other hand, is chiefly the concern of the central, matching mechanism, and I will have little to say about it here.

Recall the work mentioned in section 1 which uncovered what appeared to be a trade-off between image-alignment preprocessing mechanisms and the storage of representations corresponding to multiple points of view. This suggests that the organism employs at least two strategies which mesh in a complex way that we will want to capture in our model. The first strategy is simply the dumb, brute force one of using the enormous memory capacity of the brain to store a variety of representations corresponding to different points of view. But this clearly won't do as a complete account, especially of how we respond to relatively unfamiliar items. Further, the way in which the need for preprocessing mechanisms is often put suggests paradox, as typically pointed out in the textbooks: it would appear that the only way in which a mechanism could "correct" for orientation is by first *recognizing* the object and then righting it based upon what is known about its correct orientation. But this is the w.ong way to look at matters: what we want is to add a mechanism that serves to make the *over-all system* respond *as if* the input was normalized. And a key to seeing how this might be done, without requiring prior classification of the type of object present, is provided by the following observation.

Suppose that the system described above is fed either an image that is very different from any it has been trained on, or a familiar image at an unfamiliar orientation. Then the system can be set up so that the output almost invariably either comes to rest in a spurious attractor or is otherwise frozen in a state very different from anything it has been trained on.[1] From the point of view of the

[1] I don't have an analysis of exactly how and when this can be assured. But it appears from simulations that the scheme I outline here can be made to work. In addition to taking advantage of the spurious attractors that arise with normal training, one can manipulate both the number of iterations allowed, as well as manipulating where activity is clipped as it cycles through the BSB net. (Thus, in most of the simulations, I stopped execution after a fairly small number of iterations, usually 31, on the assumption that there was some sort of impatient neural mechanism which had the same effect-at least in the absence of higher-level feedback.) One can also do some thresholding of the output of the 2-layer net to shut the door on what would otherwise be especially weak ouput provoked by unfamiliar or tilted images. These manipulations will affect the reconstructive capacity of BSB, but, though the matter needs more careful study, it appears that values can be found for the foregoing parameters that leave the reconstructive properties of BSB largely intact for vectors that the system knows about, while still helping to freeze vectors corresponding to mis-oriented or otherwise unfamiliar input in meaningless states.

organism, such resulting states are just inert, meaningless noise, if we view the regular, trained outputs of the net as constituents in a physical symbol system–signaling cats, dogs, and the like–then we might say that the states associated with, say, spurious attractors are just not in the organism's symbolic repertoire–there are just no physical processes to detect them and, say, drive motor responses. This suggests that we devise a mechanism that systematically tries out different possible orientations of the input image very fast, for it won't matter if a number of the trials are wrong so long as the procedure ensures that a manageable orientation will be fed in within a reasonable period of time. If only 2-D rotations need be tried, and each candidate can be matched in parallel against all of the categories and some range of associated pre-stored viewpoints (from which the net, of course, generalizes), then the search is feasible and can be accomplished by adding the following 2-layer recurrent net (essentially a recurrent version of the "phase space sandwiches," proposed for somewhat different purposes, by the Churchlands, see Churchland 1986).

It is possible to construct a 2-layer net in which the projection of one layer to the next realizes a rotation by some angle, say, 20 degrees. If the activity on the second layer is then fed back through the first layer, the image will be spun about by (here) 20 degree jumps. Output at each pass can then be sampled and fed into the pattern-matching system. Simulations reveal a trade-off that roughly mirrors that found in the experimental literature. as more, distinct rotations of the same image are paired with the same label (or as the rotations are spread out), the amount of rotation needed to find an input orientation that the system responds to quickly and accurately decreases. This was checked, for various training regimens, by starting the input images at the same far leftward opening orientation and then rotating them rightward by small increments, and recording the response of both the BSB net (accuracy and response time) as well as the 2-layer net (length and accuracy)--where it was important to look at the latter as well, since this gave a deeper insight into the sort of stored representation that was driving the over-all response of the system. The behavior of the system is, of course, sensitive to such things as the relative number of exemplars paired with each name, the degree of distortion by rotation or otherwise–of the different exemplars, and the similarity of the different categories, some care was, however, taken to try to isolate for the effects of differences in orientation. Finally, since relatively small (13 x 13) input fields were used, only two coherent, recognizable images were learned, though the system was trained on anywhere from 5-13 additional associations consisting entirely of random vectors.[1]

The foregoing net just dumbly rotates every image in the same direction, but humans clearly do not take this strategy (Neisser 1966; Ullman 1986; Tarr and Pinker 1989). A slight modification is to, in effect, layer two such nets on top of each other, each constructed to rotate input in opposite directions. To rotate in one direction, one set of synapses is opened and the other closed. Neurally implemented, this could be accomplished either through regular inhibition or excitation, or perhaps through the use of global neuromodulators, much as such modulators are used to change the topology of invertebrate nervous systems on the fly (Selverston and Mazzoni 1989; Marder 1987). The idea would be to add mechanisms–that one hopes would be fairly simple–to detect certain very general features of the object, like the axis of elongation, and then use this information to choose the direction of rotation (notice that, on the present scheme, this is all that is necessary–one doesn't need to determine when to arrest the rotation). This issue is currently under investigation. Finally, it must be conceded that–even if they need not do so in most, ordinary perceptual situations–subjects clearly are both capable of rotations in depth of 3-D forms and they do this spontaneously under certain circumstances (see Shephard and Metzer, 1971, and the discussion in Tarr and Pinker, 1989). Yet the model in its present form cannot account for this.

It is also possible to build similar two-layer recurrent nets to either shrink or expand images--together constituting a kind of neural "zoom lens". Presumably, the image is either expanded or shrunk to a roughly standard size, thus, e.g., when an expanding image hits an "edge," an inhibitory gate–acting on units into which the output, at each cycle, is fed–might be swung open. It would also be possible to allow for some sampling at different sizes, perhaps triggered when enough of the "mass" of an image neared an edge. And it would as well be interesting to explore the possibility of allowing for more flexible deformations of the image at this stage, this would amount to "spreading" the capacity to handle object distortion throughout much of the system–much as the capacity to handle differences in orientation is, in the manner explained above, distributed across several, interacting modules. I have only run a few, simplified simulations of an "expander" net. Here, too, refinements will no doubt have to be added if the model is to be

[1] There are some subtle but important problems that remain to be studied. Exactly how finely do we sample? Can the system be arranged so that this can be changed flexibly? Is there perhaps some relatively safe strategy that involves choosing a sampling size that corresponds to assuming that the present object is of a kind that the organism has a fair amount of experience with--and then sampling more finely if this strategy fails? Moreover, if, as briefly alluded to below, more flexible deformations of the input need to be added to the system, then the structure and feasibility of the search becomes correspondingly less clear.

brought into close agreement with experimental results (cf., Larsen and Bundesen, 1978).

Finally, I have run some simulations of a net that implements a parallel center of mass computation. This net is somewhat different from the preceding two, and I won't go through it in detail. Put roughly, it works by repeatedly adding up activity over the whole of a map and then opening or facilitating a pattern of connections, again over the whole map, which results in the pattern being "slid" into place; as the net is currently configured, vertical and horizontal center of mass computations are actually computed separately , one after the other. It is true that this sort of center of mass computation is a bit messy and imprecise, but it may still remain useful and effective when combined with the other mechanisms described above. For it is not the performance of the simple, constituent mechanisms taken by themselves that is important. What matters is the performance of the system as a whole. Anu an obvious next step is to systematically explore the interactive effects of the different modules and processes that have been discussed.

I would propose that spatial displacement and size are dealt with first and in that order, and that the neural image is passed through the orientation net only optionally–perhaps when the subject encounters special difficulty in recognizing an object, or has information that it is likely to be misoriented (see the discussion in Tarr and Pinker, 1989, on this). I'm not aware of experimental evidence that bears on the order of such processes. However, in line with the Kosslyn-inspired story of mental imagery endorsed above, these perceptual preprocessing mechanisms would also be the same mechanisms used to manipulate mental images (accessed and fed backward in the way discussed earlier). And it is, first, interesting to note that, introspectively, it feels much more natural to perform the operations of translating, rotating, and expanding images separately. In keeping with this impression, the effects of irrelevant differences of size and orientation on the time needed to decide whether or not two visually presented forms are the same, are additive (Sekular and Nash, 1972).

Psychological data aside, how does the present scheme compare with other models of object recognition in practical terms? Without attempting a systematic review, I tentatively suggest the following. It is true that the model in its current form–with a small, simple, coarsely coded associative net at its core–cannot compete with other contemporary accounts in the ability to handle object distortions. But the relative ease with which it can handle extreme variations in displacement, size, and orientation–without special, extra training–does, I think, compare favorably with the performance of many other contemporary models. It may, in short, prove computationally practical and efficient to harness whatever matching mechanism one favors to the sort of simple pre and post-processing mechanisms discussed here.

5. IMAGE SEGMENTATION AGAIN

Finally, as promised, a closing note on the problem of image segmentation. I must say, to begin, that I have no precise and concrete proposals to offer regarding this difficult problem. But we may be in a position to view the issue in a useful light. Livingston and Hubel (1988) summarize certain features and strategies, first noted by the gestalt psychologists, used in order to segregate objects from each other and from the background (Livingston and Hubel associate these functions with the older and more primitive magnocellular system):

> ..common movement (objects move against a common stationary background; contours moving in the same direction and velocity are likely to belong to the same object, even if they are different in orientation or not contiguous); common depth (contours at different distances from the observer are unlikely to belong to the same object); collinearity (if a straight or continuously curved contour is interrupted by being occluded by another object, it is still seen as a single object); and common color or lightness...[though experiments suggest] that only luminance contrast, and not color differences, is used to link parts together.

And other features and strategies can be added. Thus, e.g., in a recent lecture at Brown, Dana Ballard suggested that with the eye/camera attached to a body/robot, a useful strategy is to fixate a point on a surface, move back in forth, and then use the blur information to peel out a candidate coherent, significant object.

The term "candidate" may be especially appropriate here. The problem of segmenting an image may seem especially daunting if the task is seen to require applying a procedure that will draw out a significant object for further processing *without fail*, or at least with high reliability. But our earlier discussion suggests that this may be the wrong way to look at things: there is no harm in using a strategy that *frequently fails* to isolate a part of the environment that elicits a meaningful response--so long as the broader strategy employed insures that a meaningful candidate will be tried out within a reasonable period of time. Thus, while no one strategy may be especially foolproof or even reliable, the rapid, repeated, experimental, and perhaps somewhat messy application of them all may prove quite sufficient.

Acknowledgements

The paper benefited from comments from Jim Anderson, Heinrich Bülthoff, Gregory Murphy, and David Sheinberg. Kathy Spoehr helped me find my way through the relevant literature. I was also helped by a discussion, at the summer school, with my editor, Geoff Hinton (who takes a very different approach to object recognition; see Zemel, Mozer, and Hinton, 1990).

References

Anderson, J.A., Silverstein, J.W., Ritz, S.A., and Jones, R. (1977). Distinctive features, categorical perception, and probability learning: some applications of a neural model. Psychological Review 84:413-451.

Anderson, J.A., Rossen, M.L., Viscuso, S.R., M.E, Sereno. (to appear, 1990). Experiments with representation in neural networks: Object motion, speech, and arithmetic. In S. Haken and M. Stadler (eds.) Synergetics of Cognition. New York: Springer-Verlag.

Anderson, J.A. (to appear, 1991). Why, having so many neurons, do we have so few thoughts? In S. Lewandowsky and W. Hockley, (eds.) Relating Theory to Data. LEA: Hillsdale.

Bartram, D. (1974). The role of visual and semantic codes in object naming. Cognitive Psychology, 6, 325-356.

Biederman, I. (1987). Recognition-by-components. A theory of image understanding. Psychological Review, 94, 115-147.

Biederman, I. and Ju, G. (1988). Surface versus edge-based determinants of visual recognition. Cognitive Psychology 20, 38-64.

Biederman, I. and Cooper, E.E. (1990). Evidence for complete translational and reflectional invariance in visual object priming. Unpublished ms. University of Minnesota.

Bennett, D.B. (1990). Psychophysical laws: A Framework. Ph.D. dissertation in Philosophy, Brown University.

Brown, T.H., Ganong, A.H., Kairiss, E.W., and Keenan, C.L. (1989). Hebbian synapses: Biophysical mechanisms and algorothms. Annual Review of Neuroscience, Vol 12.

Bruner, J. (1957). On perceptual readiness. Psychological Review, 64, 2, 123-152.

Churchland, P. M. (1986). Neurophilosophy: Toward a unified theory of the mind-brain. M.I.T.: Bradford.

Edelman, S., and Bülthoff, S. (1990). Viewpoint-specific representations in three-dimensional object recognition. A.I. Memo No. 1239. Artificial Intelligence Laboratory, M.I.T.

Edelman, S., Bülfhoff, H.H., Weinshall, D. (1989) Stimulus familiarity determines recognition strategy for novel 3D objects. A.I. Memo No. 1138. Artificial Intelligence Laboratory, M.I.T.

Hoffman, D.D., and Richards, W.A. (1984). Parts of recognition. Cognition, 18, 65-96.

Jolicoeur, P., Gluck, M.A., and Kosslyn, S.M. (1984). Pictures and names: Making the connection. Cognitive Psychology, 16, 243-275.

Kosslyn, S.M. (1987). Seeing and imagining in the cerebral hemispheres. A computational approach. Psychological Review, 94, 148-175.

Knapp, A. G., and Anderson, J. A. (1984). Theory of categorization based on distributed memory storage. Journal of Experimental Psychology: Learning, Memory, and Cognition, 10, 616-637.

Lakoff, G. (1987). Women, Fire, and Dangerous Things. Chicago: University of Chicago Press.

Livingston, D.H, and Hubel, D.H. (1988). Segregation of form, color, movement, and depth: Anatomy, physiology, and perception. Science, 240, 740-479.

Marder, E. (1987). Neurotransmitters and modulators. In A.I. Selverston and M. Moulins (eds.) The Crustacean Stomatogastric System. N.Y.: Springer-Verlag.

Marr, D. (1982). Vision. San Francisco: Freeman.

McClelland, J.L., and Rumelhart, D.E. (1986). A distributed model of human learning and memory. In McClelland, J.L., and Rumelhart, D.E. (eds.) Parallel Distributed Processing Processing, Vol. II. Cambridge· Bradford.

Murphy, G.L., and Brownell, H.H. (1985). Category differentiation in object recognition: Typicality constraints on the basic level category advantage. Journal of Experimental Psychology: Learning, Memory, and Cognition, 11, 1, 70-84.

Nakayama, K. (1987). The iconic bottleneck and the tenous link between early visual processing and perception. In C. Blakemore (ed.), Vision: Coding and Efficiency, Cambridge, England: Cambridge University Press.

Nakayama, K., Shimojo, S., and Silverman, G.H. (1989). Stereoscopic depth. Its relation to image segmentation, grouping, and the recognition of occluded objects. Perception, 18, 55-68.

Neisser, U. (1966). Cognitive Psychology. N.Y.. Appelton Century-Crofts.

Palmer, S.E., Rosch, E. and P. Chase (1981). Canonical perspective and the perception of objects. In Long J., and Baddeley, A. (eds.) Attention and Performance IX, 135-151. Hillsdale, N.J.: Erlbaum.

Rosch, E., Mervis, C.B., Gray, W., Johnson, D., and Boyes-Braem, P. (1976). Basic objects in natural categories. Cognitive Psychology, 8, 382-439.

Schacter, D.L., Cooper, L. A., and Delaney, S.M. (in press). Priming of nonverbal memory and the nature of implicit memory. In G.H. Bower (ed.) The psychology of of learning and motivation, Vol 26. New York: Academic Press.

Sejnowski, T., Chattarji, S., and Stanton, P. (1989). Induction of synaptic plasticity by hebbian covariance in the hippocampus. In Durbin, R, Miall, C., Mitchison, G. (eds.) The Computing Neuron. New York: Addison-Wesley.

Sekular and Nash (1972). Speed of size scaling in human vision. Psychonomic Science, 27, 93-94.

Selverston, A., and Mazzoni, P. (1989). Flexibility of computational units in invertebrate CPGs. In Durbin, R, Miall, C., Mitchison, G. (eds.) The Computing Neuron. New York: Addison-Wesley.

Shephard, R.N., and Metzer, J. (1971). Mental rotation of three dimensional objects. Science, 171, 701-703.

Smith, E.E., and Medin, D. (1981). Categories and Concepts. Cambridge: Harvard.

Tarr, M.J., and Pinker, S. (1989). Mental rotation and orientation-dependence in shape recognition. Cognitive Psychology, 21, 2, 233-282.

Tversky, B. and Hemenway, K. (1984). Objects, parts, and categories. Journal of Experimental Psychology: General, 113, 169-193.

Ullman, S. (1986). An approach to object recognition: Aligning pictorial descriptions. A.I. Memo No. 931, Artificial Intelligence Laboratory, M.I.T.

Viscuso, S. R., Anderson, J.A., Spoehr, K. T. (1989). Representing simple arithmetic in simple neural networks. G. Tiberghien (Ed.) Advances in Cognitive Science. Theory and Application. New York: Wiley.

Zemel, R.S., Mozer, M., Hinton, G. (1990). TRAFFIC: Recognizing Objects Using Hierarchical Reference Frame Transformations. In D.S.Touretzky (ed.), Advances in Neural Information Processing Systems 2. San Mateo, Ca.: Morgan Kaufman.

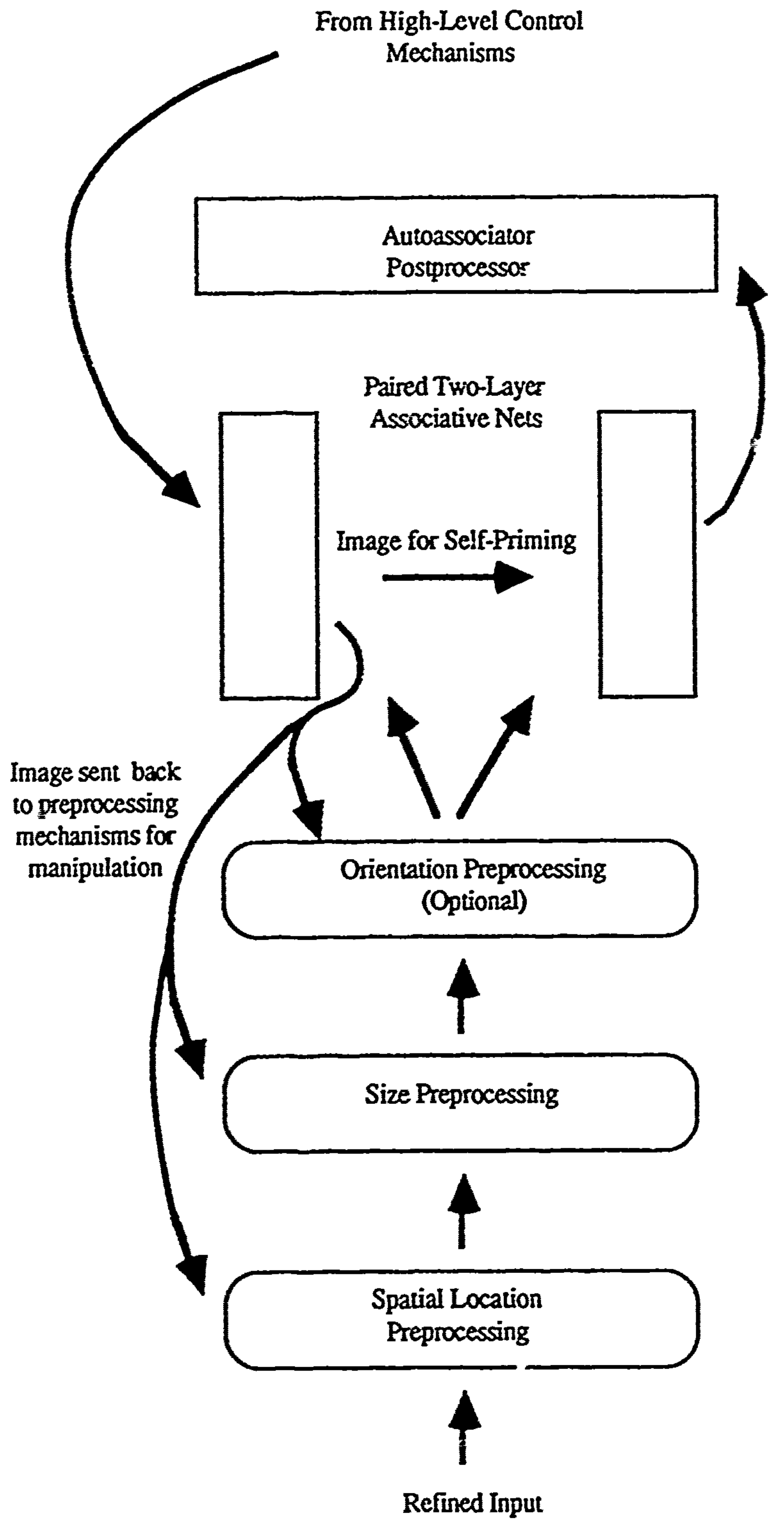

From High-Level Control Mechanisms
Autoassociator Postprocessor
Paired Two-Layer Associative Nets
Image for Self-Priming
Image sent back to preprocessing mechanisms for manipulation
Orientation Preprocessing (Optional)
Size Preprocessing
Spatial Location Preprocessing
Refined Input

Figure 1: Main flow of activity and information through the system. This oversimplifies the discussion in the text and should thus be viewed with some caution.

Part IX

Biology

Modeling cortical area 7a using Stochastic Real-Valued (SRV) units

Vijaykumar Gullapalli
Computer and Information Sciences Department
University of Massachusetts
Amherst, MA 01003

Abstract

Area 7a of primate cortex is thought to transform visual stimuli from retinotopic coordinates into a head-centered coordinate system by combining retinal stimuli with proprioceptive feedback of eye position (Andersen, 1989). Two previous neural network models of area 7a (Zipser and Andersen, 1988; Andersen and Zipser, 1988; Mazzoni, Andersen and Jordan, 1990), trained using back-propagation and a variant of the associative reward-penalty (A_{R-P}) learning rule, have been shown to replicate the overall function as well as the unique response properties of neurons in area 7a. Validity of neural network models of area 7a remains questionable, however, on the grounds of biological plausibility of the learning mechanisms employed, oı because of computational limitations of the network units used in these models. This paper presents a neural network with stochastic real-valued (SRV) units and with architecture similar to that of the previous models, trained on the same coordinate transformation task. It is shown that while capturing the performance features of the previous models, this network is also free from their limitations. This reaffirms the validity of the basic neural network model of the coordinate transformation circuitry of area 7a proposed by Andersen and Zipser.

1 INTRODUCTION

Neural networks are so called because their architecture and learning mechanisms are often inspired by those observed in the neural circuits of the brain. It therefore seems natural to use such networks to model subsystems in the brain. Andersen and Zipser (Andersen and Zipser, 1988, Zipser and Andersen, 1988) proposed one such neural network model of area 7a of the posterior parietal cortex in monkeys.

Area 7a of primate cortex is thought to transform visual stimuli from retinotopic coordinates into a head-centered coordinate system by combining retinal stimuli with proprioceptive feedback of eye position (Andersen, 1989). In recordings from area 7a, Andersen and Zipser found neurons whose responses depend on both the retinal location of visual stimuli and the position of the eyes in the orbits. Their experiments suggest that locations in craniotopic space are encoded in two stages. First an eye-position-dependent coding is formed by cells that respond over limited ranges of eye positions. The outputs of groups of such cells are used in the second stage to produce an eye-position-independent encoding of the stimulus in craniotopic space.

Recordings from individual neurons also showed that neurons in area 7a have unique response properties which are called *spatial gain fields* and *visual receptive fields*. Spatial gain field is a term for the modulating effect of the eye position on a neuron's response to a fixed retinal stimulus. The visual receptive field of a neuron determines the magnitude of its response to stimuli at different retinal locations when the eye position is held fixed. Andersen and Zipser (Andersen and Zipser, 1988; Zipser and Andersen, 1988) proposed a neural network model of this coordinate transformation system that was trained using the back-propagation algorithm (Rumelhart et al., 1986). Units of the trained network were shown to have spatial and eye position dependent response properties that were qualitatively similar to those observed in area 7a neurons of monkeys.

Although the Andersen and Zipser model gives good qualitative results, doubts regarding the biological plausibility of the back-propagation algorithm used for training the network hinder the acceptance of their model as a valid model of area 7a. For this reason, Mazzoni, Andersen, and Jordan (1990) decided to examine the necessity of back-propagation for developing appropriate response properties in the units of the network. Using a modified version of a more biologically plausible learning algorithm called the associative reward-penalty (A_{R-P}) algorithm (Barto and Anandan, 1985), they were able to show that essentially identical response topographies were learned by units in networks using the A_{R-P} algorithm. Thus, the neural network architecture and learning principles of the Andersen-Zipser model of area 7a appear to be physiologically viable.

Unfortunately, the A_{R-P} units used by Mazzoni et al. have the drawback of being binary units, which necessitates interpreting their outputs as spikes of neuronal activity over time. Since the measure of activity of interest in this task is the spiking *rate* of the neurons, these units do not appear to be well suited to the task requirements. In fact, Mazzoni et al. point out the desirability of replacing the A_{R-P} units by continuous-valued units that can learn using algorithms similar to the A_{R-P} algorithm. My previous work led to the development of one such unit, called the stochastic real-valued (SRV) unit (Gullapalli, 1990). In this paper, I present the performance of a network with SRV units trained on the coordinate transformation task postulated for area 7a neurons. The architecture of the network is similar to that of Andersen and Zipser's model, and I compare the results of my network with their results as well as those of Mazzoni et al. This work demonstrates once again the validity of the general neural network architecture and learning principles of the Andersen-Zipser model. It also serves to verify the prediction in Mazzoni et al.'s paper that the use of continuous-valued units that learn using reinforcement feedback should not drastically change the type of solution found by these networks.

2 NETWORK STRUCTURE AND TRAINING

The network used for the simulations had a three layer, fully connected, feed-forward architecture, and is most similar to the "Mixed A_{R-P}" network of Mazzoni et al. (1990). The architecture is depicted in Figure 1. Inputs to the network were split into two groups consisting of the retinal inputs and the eye position inputs, both of which were modeled according to the known characteristics of the corresponding neuron groups in area 7a (e.g., Andersen and Zipser, 1988). Retinal inputs were encoded as an 8 × 8 array of units with Gaussian receptive fields, while the eye position was represented by 32 units with linear activation functions, half of which were used to encode the x position of the eye, and the other half to encode the y position.

Hidden units in the network were SRV units producing a random output between 0 and 1. Networks with between two and eight hidden units were tried in the simulations, although all the results reported in this paper are for networks with three hidden units. Since

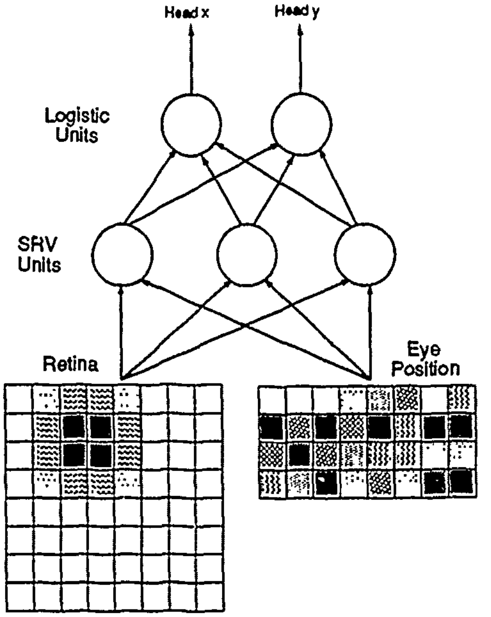


Figure 1: Architecture of the Network Used.

the output of the network was the craniotopic location of the retinal stimulus, the representation used to encode this location determined the structure of the output layer of the network. Possible encodings include a Gaussian coarse-coding array like the retinal input, a distributed representation using linear units like the eye position input, or a scalar encoding of the x and y coordinates of the craniotopic location. In our experiments, we used the first and the third types of encodings, but all the simulations reported in this paper used only the third type of encoding. Hence the network had two output units, which are logistic units producing outputs between 0 and 1. These were scaled to the appropriate ranges for the x and y coordinates of the craniotopic locations.

The training data consisted of eight training patterns, or stimulus-response pairs, that were chosen randomly. Training the network involved repeated execution of a fixed sequence of operations. Each repetition is called a *trial* and a set of eight consecutive trials, one for each training pattern, is called an *epoch*. Each attempt to train the network, called a *training run*, involved the execution of a fixed large number of epochs. A trial consists of the following operations. Each trial begins with the presentation of the stimulus vector of a training pattern as input to the network. The hidden units compute their stochastic outputs based on these inputs, following which the output units compute their activations to form the network's output vector. This output vector is compared with the desired response vector corresponding to the training pattern, yielding an error vector used by the network in two ways. First, the error vector is used directly by the units in the output layer to update their weights using the standard back-propagation rule.[1] Second, the magnitude of the error vector is computed and used to generate a reinforcement or reward signal that is broadcast to all the units in the hidden layer. These units use the SRV unit learning algorithm (Gullapalli, 1990) to update their weights so as to increase the probability of higher reward in the future. The updating of the weights is the last operation performed in each trial.

The equations for computing the outputs and the learning rules used for updating the weights of units in the hidden and output layers are summarized below. Let **x** denote the vector of inputs to a unit in the network. SRV units compute their stochastic outputs by first computing the mean, μ, and the standard deviation, σ, of the standard normal distribution, using these parameters to generate a random activation, and then computing the output as a function of the activation.

Two weight vectors, **w** and **v**, are used for computing the two parameters μ and σ. The mean is computed as

$$\mu(t) = \sum_{i=1}^{n} w_i(t)x_i(t) + w_{thres}(t). \qquad (1)$$

The standard deviation σ is computed in two stages. First, an expected reinforcement, $\hat{r}$, is computed using the weight vector **v** as

$$\hat{r}(t) = \sum_{i=1}^{n} v_i(t)x_i(t) + v_{thres}(t). \qquad (2)$$

This expected reinforcement is then used to compute the standard deviation as

$$\sigma(t) = s(\hat{r}(t)), \qquad (3)$$

where $s(.)$ is a monotonically decreasing, nonnegative function of $\hat{r}(t)$. Moreover, $s(1.0) = 0.0$, so that when the maximum reinforcement is expected, the standard deviation is zero. In the simulations reported in this paper, $s(.)$ was defined as

$$s(\hat{r}(t)) = \max\left(\left(\frac{1.0 - \hat{r}(t)}{5.0}\right), 0.0\right). \qquad (4)$$

Based on $\mu(t)$ and $\sigma(t)$, the unit computes its activation, $a(t)$, which is a normally distributed random variable:

$$a(t) \sim N(\mu(t), \sigma(t)). \qquad (5)$$

Finally, the activation $a(t)$ is transformed into the output of the unit $y(t)$ using the output function $f(.)$, so that

$$y(t) = f(a(t)). \qquad (6)$$

In the simulations reported in this paper, the logistic function

$$f(z) = \frac{1}{1 + e^{-z}} \qquad (7)$$

was used as the output function of the unit.

Equations (1)–(7) describe how the SRV unit uses its inputs to compute its output at a given time step. I now give the equations describing how the unit updates its weights. The environment provides these units with a reinforcement signal $r(t)$ at time t, which is a function of the output error $e(t)$:

$$r(t) = 1.0 - e(t), \qquad (8)$$

where

$$e(t) = \left(\frac{1}{2}\sum_{i \in O} |d_i(t) - x_i(t)|\right)^{\frac{1}{2}}. \qquad (9)$$

Here O is the index set of the output units and d_i is the desired output of the i^{th} output unit.

The weights, **w**, determining μ are updated at each time step as follows:

$$w_i(t+1) = w_i(t) + \alpha\Delta_{\mathbf{w}}(t)x_i(t), \qquad (10)$$

$$w_{thres}(t+1) = w_{thres}(t) + \alpha\Delta_{\mathbf{w}}(t), \qquad (11)$$

where α is the learning rate parameter and

$$\Delta_{\mathbf{w}}(t) = (r(t) - \hat{r}(t))\left(\frac{a(t) - \mu(t)}{\sigma(t)}\right). \qquad (12)$$

For updating of the weights, μ, determining the expected reinforcement, the LMS rule of Widrow and Hoff (1960) is used, as shown in the following equations:

$$v_i(t+1) = v_i(t) + \beta\Delta_{\mathbf{v}}(t)x_i(t), \qquad (13)$$

$$v_{thres}(t+1) = v_{thres}(t) + \beta\Delta_{\mathbf{v}}(t), \qquad (14)$$

where β is the learning rate parameter and

$$\Delta_{\mathbf{v}}(t) = r(t) - \hat{r}(t). \qquad (15)$$

Typical values for the learning rates used in the simulations were 0.2 for α and 0.02 for β.

The logistic units in the output layer of the network use the following equations to compute their outputs and to update their weights. Each output is computed as a function of the weighted sum

$$s(t) = \sum_{i=1}^{n} w_i(t)x_i(t) + w_{thres}(t) \qquad (16)$$

as

$$y(t) = f(s(t)), \qquad (17)$$

where $f(.)$ is the logistic function given by Equation (7). The weights are adjusted using the backpropagation or delta rule

$$w_i(t+1) = w_i(t) + \epsilon(d(t) - y(t))f'(s(t))x_i(t), \qquad (18)$$

where $f'(.)$ is the derivative of the logistic function, ϵ is the learning rate parameter, and $d(t)$ is the desired response for the unit.

[1] Since the back-propagation rule is being used only in the output layer, doubts about the biological plausibility of the retrograde propagation of error signals through synapses are irrelevant here. (See Discussion).

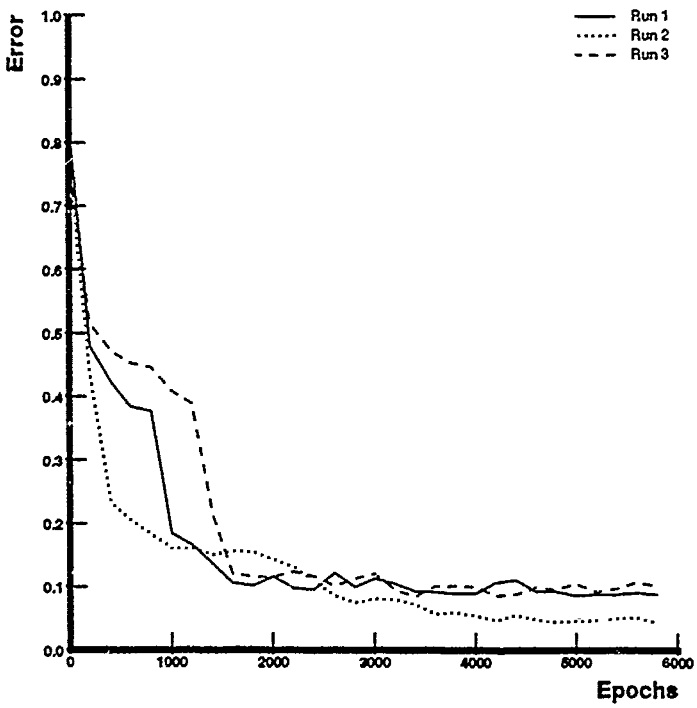


Figure 2: Learning Curves For 3 Training Runs On The Coordinate Transformation Task.

3 SIMULATION RESULTS

3.1 LEARNING

Statistics of the learning performance of the network were collected over twenty training runs. The network learned to perform the coordinate transformation task reasonably quickly. After an average of about 1200 epochs, output accuracy equivalent to the resolution of the visual input was achieved, with continued reduction in error, though at a lower rate, with further training. Local minima were not a serious problem for the network, because the use of stochastic units enabled the eventual escape from such minima. So the main effect of local minima on training was to increase the number of epochs to convergence on some of the training runs.

Sample learning curves for 3 training runs are presented in Figure 2. The error plotted at each time step in the figure is the average of output error $e(t)$ (Equation 9) over the preceeding 200 time steps. It must noted here that when compared to a back-propagation network of the same architecture, the network with SRV units takes about 5 times as many epochs to attain accuracy equivalent to the resolution of the visual input. This can be attributed to the stochastic nature of the SRV units and to the non-specificity of the training signal used by them.

3.2 RESPONSE PROPERTIES

Using the methodology of Andersen and Zipser, I analyzed the response properties acquired by the hidden units during training. Both the spatial gain fields and the visual receptive fields of the hidden layers were computed and compared with those observed for neurons in area 7a. Andersen and Zipser showed (Andersen and Zipser, 1988; Zipser and Andersen, 1988) that the majority of spatial gain fields of area 7a neurons are roughly planar. Moreover, both the visual and the eye position components of the gain field can vary simultaneously, either in the same, or in opposite directions, and to different degrees with eye position.

Figure 3 shows the spatial gain fields of the hidden units of the network for different retinal stimuli. Each set of nine circles comprises a single gain field obtained for a fixed retinal input by setting the eye position to one of nine fixed locations and measuring the output of the hidden unit with and without the retinal input. The difference between these two measurements is taken to be the contribution due to the retinal input and is shown as filled circles in the figure. The outer circles represent the magnitude of the output with the retinal inputs present, whereas the annulus between the outer circle and the filled circle represents the contribution of the eye position inputs.

Different hidden units and different retinal inputs produce a wide variation in the gain fields, as can be seen in Figure 3. The planar nature of the gain fields is easily seen, as are examples in which the the contribution of one input component is fairly constant whereas that due to the other varies roughly linearly in each direction. Also shown is a case in which the two components increase or decrease together with changes in the eye position. These spatial gain fields are very similar to those observed by Andersen and Zipser (1988) in recordings from monkeys and to those developed by hidden units of the previous models of Andersen and Zipser (1988), and Mazzoni et al. (1990).

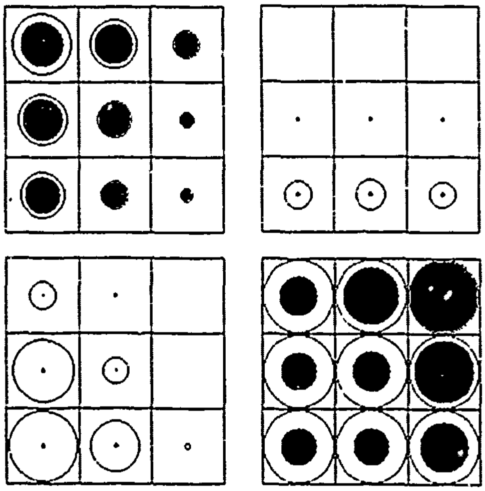


Figure 3: Examples Of Spatial Gain Fields Of The Hidden Units.

Andersen and Zipser's studies of the area 7a neurons in monkeys included measurement of the visual receptive field of a neuron to determine the magnitude of its response to stimuli at different retinal locations when the eye position is held fixed. A similar study of the receptive fields of the hidden units of the network with SRV units yielded the fields shown in Figure 4. Each field was obtained by recording the response of the hidden unit to retinal stimuli at each of seventeen locations with the eye position input held constant, and fitting a fourth order quadratic surface to the data. These receptive fields are again qualitatively very similar to those obtained by Andersen and Zipser in recordings from monkeys, and to those developed by the hidden units of the Andersen-Zipser and the Mazzoni et al. models.

4 DISCUSSION AND CONCLUSIONS

The back-propagation algorithm, the A_{R-P} algorithm, and the SRV unit algorithm all use the gradient of the performance measure to update the weights of the network. The main difference between the first of these and the latter two is that back-propagation computes the exact value of the sample gradient for each training pattern, whereas the latter two rely on an estimate of the gradient obtained by a stochastic search in the output space. The computation of the exact sample gradient by back-propagation, however, would require the retrograde propagation of error signals through synapses, which many have pointed out to be biologically implausible. This was seen as a limitation of the original neural network model of area 7a proposed by Andersen and Zipser (1988), which led Mazzoni et al. (1990) to present an alternative model based on A_{R-P} units.

Mazzoni et al. (1990) present detailed reasons for regarding the learning mechanisms used by A_{R-P} units as more biologically plausible than back-propagation. Chief among these are the use of a diffuse scalar reinforcement signal that gets broadcast to all the units in the network, the use of only information local to a synapse for adjusting the strength of that synapse, and the stochastic nature of the outputs of the units. Due to the common underlying framework on which the two algorithms are based, the SRV unit algorithm also has the same desirable characteristics as the A_{R-P} algorithm. Furthermore, SRV units have the advantage of being able to produce real-valued outputs which can be directly interpreted as the firing rates of neurons. Thus the network presented in this paper does not have the computational limitations of the networks used by Mazzoni et al.

Neural network models of area 7a trained using each of these three learning algorithms show similar performance characteristics. Each network can learn the coordinate transformation task and each network develops hidden units with response properties similar to those of area 7a neurons. Computational performance, therefore, does not appear to depend very strongly on the choice of the learning procedure. These results provide support for the basic neural network model of area 7a proposed by Andersen and Zipser. Furthermore, comparisons between the networks with A_{R-P} units and SRV units indicate that switching from binary to continuous-valued reinforcement learning units does not drastically alter the type of solutions found by these networks for the coordinate transformation task.

Acknowledgements

Mike Jordan was the motivator of the work reported in this paper. I would like to acknowledge his help and also that of Dave Zipser and Pietro Mazzoni in clarifying some of the methodology used in their work on modeling cortical area 7a. This material is based upon work supported by the Air Force Office of Scientific Research, Bolling AFB, under Grant AFOSR-89-0526 and by the National Science Foundation under Grant ECS-8912623.

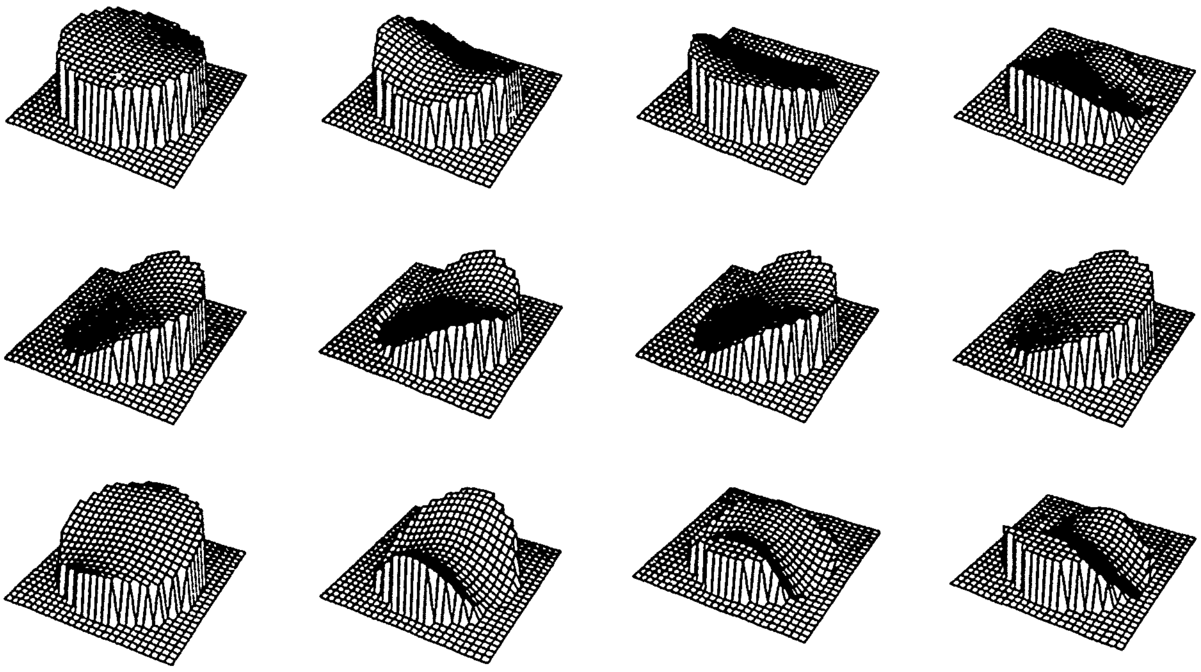


Figure 4: Examples Of The Visual Receptive Fields Of The Hidden Units.

References

Andersen, R. A. (1989) Visual and eye movement functions of the posterior parietal cortex. *Ann. Rev. Neuroscience*, **12**: 377-403.

Andersen, R. A., and Zipser, D. (1988) The role of the posterior parietal cortex in coordinate transformations for visual-motor integration. *Can. J. Physiol. Pharm*, **66**(4):488-501.

Barto, A. G., and Anandan, P. (1985) Pattern recognizing stochastic learning automata. *IEEE Transactions on Systems, Man, and Cybernetics*, 15:360-375.

Gullapalli, V. (1990) A stochastic learning algorithm for learning real-valued functions via reinforcement feedback. To appear in *Neural Networks.*

Mazzoni, P., Andersen, R. A., and Jordan, M. I. (1990) A more biologically plausible learning network model of area 7a of macaque cortex. Submitted to *J. Neuroscience.*

Mazzoni, P., Andersen, R. A., and Jordan, M. I. (1990) A_{R-P} learning applied to a network model of cortical area 7a. In *Proc. of IJCNN-90*, Vol.II, 373-379.

Widrow, B. and Hoff, M. E. (1960). Adaptive switching circuits. *1960 WESCON Convention Record Part IV*, 96-104.

Zipser, D., and Andersen, R. A. (1988) A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. *Nature*, **331**:679-684.

Neuronal signal strength is enhanced by rhythmic firing

Alan Heirich & Christof Koch
Computation & Neural Systems Program
California Institute of Technology, 216-76
Pasadena, CA 91125

Abstract

Recent discoveries of rhythmic firing patterns in cortex suggest the brain might use phase locked oscillations to coordinate activity in separate areas (Gray & Singer, 1989). Theoretical studies have suggested mechanisms by which these rhythms might propagate through cortex (Lytton & Sejnowski, 1990). This paper presents preliminary results from a study of the biophysical effects of rhythmic signals at the single cell level. The work presented here considers the transmissibility of rhythmic signals and suggests their efficacy is increased by their rhythmicity. This is demonstrated by simulations in which a train of regularly spaced synaptic inputs at an appropriate frequency causes a neuron to fire more often than an irregularly spaced train does. The reason is the arhythmic signal is diminished by transient after-hyperpolarizations. Some recent attentional phenomena (Spitzer, Desimone & Moran, 1988, Wise & Desimone, 1988) are discussed in light of these results.

1 INTRODUCTION

Our brains process information in many forms: visual, auditory, olfactory, and somatic, to name a few. Distinctly different regions of the brain receive these different forms of information. How does the brain know that these separate pieces of information are related? Conversely, how is the brain able to represent several pieces of information simultaneously without confusing them? In the sensory domain this first question has come to be known as the "binding problem" (Stryker, 1989). In the cognitive domain Norman has identified the second as the "type-token problem" (Norman, 1986). A possible solution to both problems is for the brain to use timing relationships to organize pieces of information that are represented in physically distinct areas. In particular, frequency and phase of neural firing have been proposed as candidate codes for this purpose (Perkel & Bullock, 1968; Malsburg, 1981; Damasio, 1989).

Recent discoveries of phase locked oscillations in visual cortex suggest that these speculations may indeed be true (Gray & Singer, 1989; Gray, König, Engel & Singer, 1989). If they are true a number of questions come to mind. First, what advantages does this coding mechanism offer over other coding mechanisms? Another question might be how this mechanism relates to attention. (Crick, 1984) suggests the thalamus might be responsible for guiding an inner "attentional searchlight" (Treisman, 1982). (Crick & Koch, 1990a) suggest that this locus of attention might correlate with the presence of the phase locked oscillations.

In this paper we consider the advantages rhythmic firing patterns might have for an attentional mechanism. We present preliminary results from a biophysical study of cortical pyramidal neurons which suggest that a rhythmic signal induces a more powerful cellular response than does an arhythmic signal. We relate this to attention by suggesting that this enhanced signal strength might account for competitive effects observed in visual area V4.

2 SIMULATION OF A CORTICAL PYRAMIDAL NEURON

We modeled the soma of a realistic pyramidal neuron using data from (Lytton & Sejnowski, 1990). We simulated the three major voltage gated channels for sodium (I_{Na}), delayed rectifying potassium (I_K) and a transient non-inactivating potassium (I_A). The soma was a cylindrical compartment 10 μm in radius by 22 μm in length with a total surface area of 1382.3 μm^2. The maximal conductance of sodium was 0.1 S/cm^2 with a reversal potential of 45 mV. The maximal conductance of the potassium rectifier was 0.12 S/cm^2 with a reversal potential of -90 mV The maximal conductance of the non-inactivating potas sium was 0.2 S/cm^2 with a reversal potential of $-77\ mV$. (The different reversal potentials for the two potassium currents is a result of the different selective permeabilities of the two types of channels). The membrane capacitance was 1 $\mu F/cm^2$. Synaptic conductance $g_{synapse}$ was modeled by an alpha function with a peak at 2.5 ms and reversal potential 45 mV At peak conductance $g_{synapse}$ was 9 nS. The leak conductance was 0.79 × 10^{-4} S/cm^2 with a reversal potential of $-64.72\ mV$. The input impedance was 16 $M\Omega$ measured with current injections of 0.01 nA and 0.05nA for 100 ms.

We stimulated the soma for 1000 ms with a signal whose average frequency was 40 Hz. For the rhythmic case we scheduled excitatory synaptic potentials every 25 ms. For the arhythmic case we generated four uniformly distributed random schedules of 40 EPSPs within 1000 ms. Table 1 presents one of the four arhythmic schedules. We simulated all four schedules and counted the resulting action potentials. Simulations used 2nd order Runge-Kutta integration with a fixed step size of $\Delta t = 0.01\ ms$. Abbreviated simulations were run with smaller step sizes with no discernible differences. Simulations were programmed in *Dynamica*, a dynamical system modeling environment developed by the first author.

Figures 1 & 2 tell the story. the rhythmic EPSP schedule produced 40 action potentials, 1 for each EPSP, while the arhythmic schedules produced an average of 28.25 spikes, a ratio of 1 to 1.42. Upon closer examination we concluded that the I_A current was responsible for this effect. The A channels closed slowly due to the lower bound on τ_a. When an initial EPSP was followed too closely by another, the resulting depolarizing synaptic current was overwhelmed by the residual I_A.

In order to verify this conclusion we ran a control experiment in which we slowed the A channel by increasing the lower bound on τ_a by approximately 50% (from 0.28 to 0.45). This had the paradoxical effect of retarding g_A and thus reducing the A current. An arhythmic schedule of 40 EPSPs in 1000 ms produced 37 action potentials.

3 COMPETITIVE EFFECTS IN VISUAL ATTENTION

How do these results relate to attention? (Spitzer, Desimone & Moran, 1988, Wise & Desimone, 1988) have demonstrated that attention can enhance competitive effects in area V4 of visual cortex. V4 contains cells with very large receptive fields. The receptive fields of cells overlap considerably and thus a given stimulus might excite multiple cells with the same receptive field. When this happens attention appears to create a competition which is won by the cell representing the attended feature. For example, a given receptive field might be stimulated by a red dot and by a green dot, in different spatial locations. Under normal circumstances the neurons for both of these features will fire. But if attention is paid to the red dot the corresponding neuron fires at a higher rate, while the neuron for the green dot stops firing.

This phenomenon could be explained if attention were able to boost a driving signal that led to a competition among all of the stimulated neurons. If we posit that attention correlates highly with rhythmic firing, then the advantages of rhythmic coding should accrue to attended information (Crick & Koch, 1990b). The work cited in the introduction suggests that rhythmic firing patterns might originate in primary visual cortex, and that these rhythmic patterns can propagate through cortical tissue to higher areas such as V4. Our simulations have shown that these rhythmic signals might reach V4 more forcefully than would unattended arhythmic signals. If attention were responsible for creating the rhythmic firing patterns the signals from the attended part of the visual field would propagate forcefully through cortex. This might then account for the ability of attention to determine the winner of the competition with receptive fields of V4.

In conclusion, our simulations with a biophysically realistic model of a cortical neuron shows that regularly spaced input can be more effective than irregular input. In this study the effect was due to cellular adaptation leading to prolonged afterhyperpolarization. Other mechanisms which limit the efficacy of irregularly spaced synaptic input are also possible. For example, a series of very closely spaced presynaptic action potentials would induce a sublinear postsynaptic conductance response, due to saturation of the synaptic conductance Further detailed simulations of cortical neurons are needed to bear out the robustness of these and similar phenomena.

4 APPENDIX: SIMULATION

Figure 3 shows the trajectories of the currents and the membrane voltage during the first 10 ms following an EPSP at resting potential. The dynamics of the model follow a standard Hodgkin-Huxley formalism with three types of voltage gated channels: fast sodium (Na); rectifying potassium (K); and non-inactivating potassium (A). The channel parameters for sodium and delayed rectifying potassium were taken from (Lytton & Sejnowski, 1990) modified from (Traub, 1982). The parameters for the non-inactivating potassium channel were from (Lytton & Sejnowski, 1990) modified from (Borg-Graham, 1987) to give a monotonically increasing current-frequency relationship These parameters are non-standard. The channel dynamics are governed by the gating variables m, h, n and a and corresponding time constants τ. The total membrane capacitance C_N is 0.0138 nF.

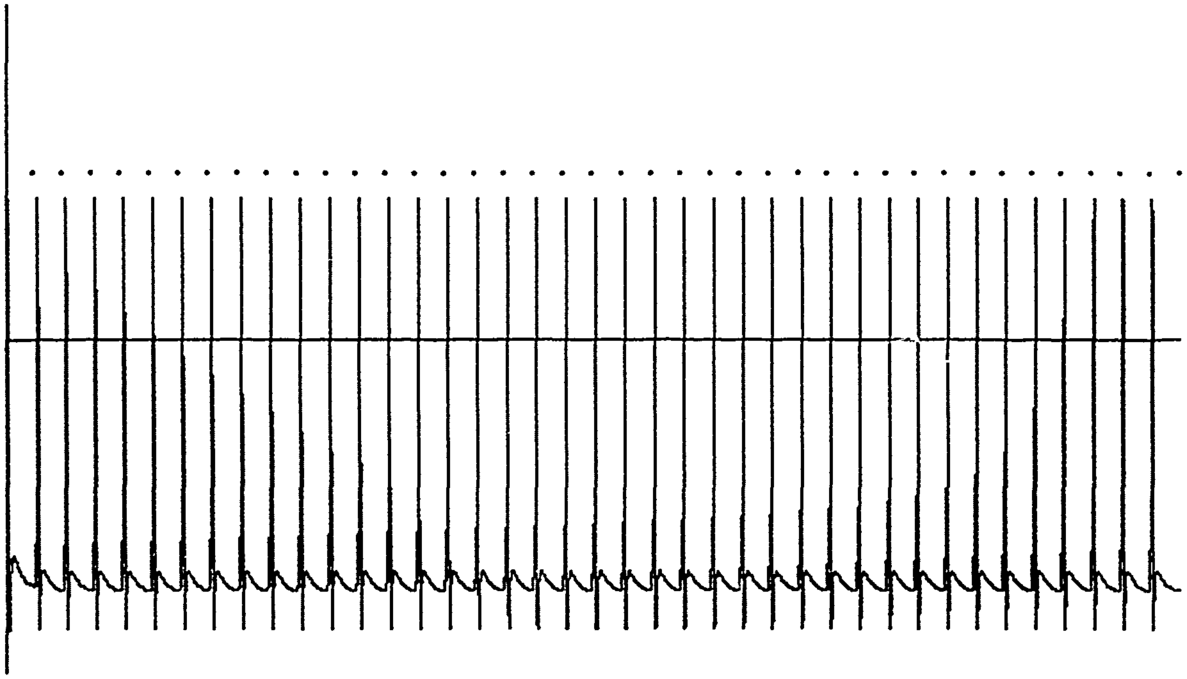


Figure 1: V vs. t at 40 Hz (regular). Horizontal axis shows time from 0 to 1000 ms. Vertical axis represents -100 to $+100$ mV. A dot overhead indicates an EPSP.

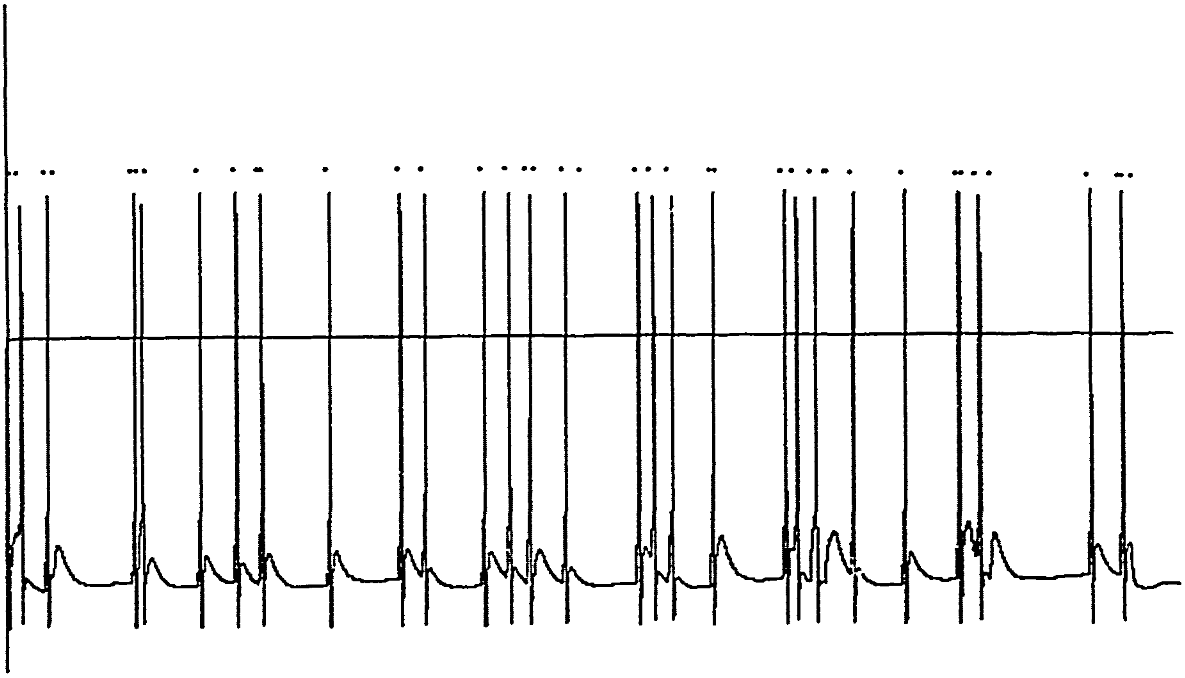


Figure 2. V vs. t at 40 Hz (arhythmic). Horizontal axis shows time from 0 to 1000 ms. Vertical axis represents -100 to $+100$ mV. A dot overhead indicates an EPSP.

0	3.97	10.56	33.27	41.51	108.98	113.96	121.43
165.17	195.78	217.20	219.43	274.79	335.15	355.72	406.20
426.80	444.29	451.03	474.90	536.97	549.18	564.35	599.72
604.76	661.67	671.26	685.54	698.23	700.34	720.02	764.40
810.91	815.49	827.63	839.53	923.25	949.92	952.61	959.53

Table 1: Arhythmic EPSP schedule used in figure 2

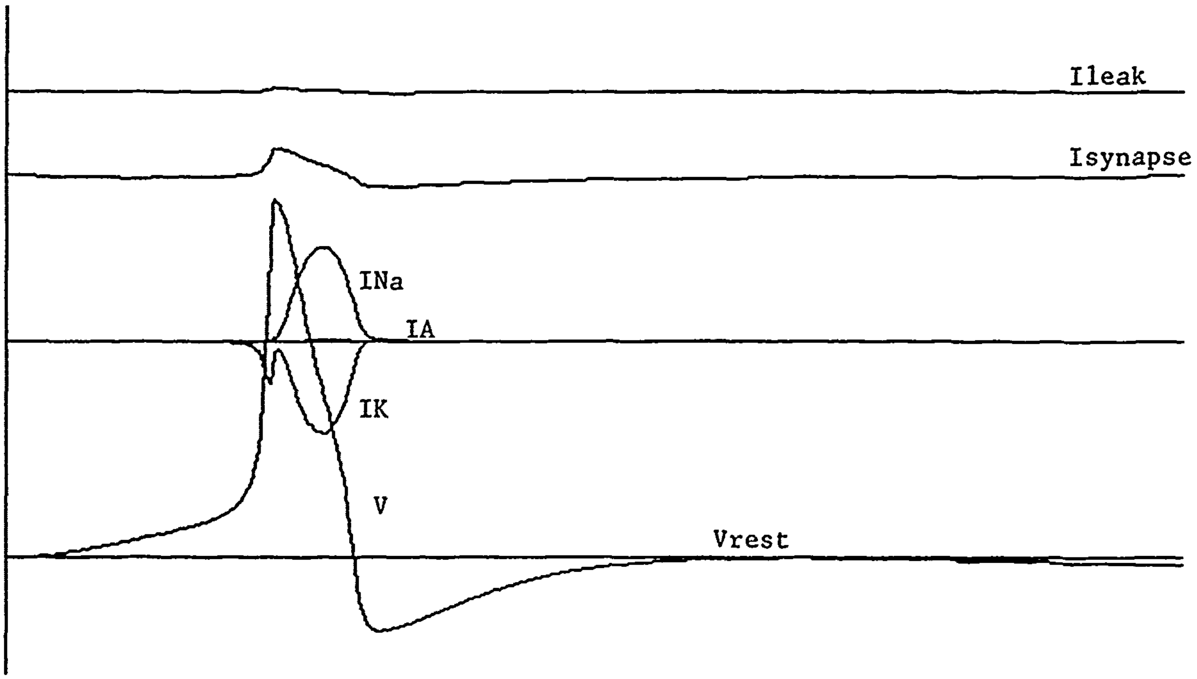


Figure 3: from top: currents I_{leak}; $I_{synapse}$; I_{Na} (downward), I_K (upward), I_A (very small upward); and membrane voltage V during first 10 ms following an EPSP. Horizontal axis shows time from 0 to 10 ms. Lower horizontal line shows resting potential $V_{rest} = -65\ mV$. I_{leak} and $I_{synapse}$ are offset from zero and multiplied by 10 for visibility.

The resting membrane potential is $-65\ mV$.

$$\frac{DV}{Dt} = \frac{-1}{C_N}(I_{Na} + I_K + I_A + I_{synapse} + I_{leak}) \quad (1)$$

$$I_{Na} = m^3 h g_{Na}(V - E_{Na}) \quad (2)$$

$$I_K = n^4 g_K(V - E_K) \quad (3)$$

$$I_A = a^4 g_A(V - E_A) \quad (4)$$

$$I_{synapse} = g_{synapse}(\Delta t)(V - E_{synapse}) \quad (5)$$

$$I_{leak} = g_{leak}(V - E_{leak}) \quad (6)$$

Figure 4 shows the channel gating variables m, h, n, a versus voltage. The horizontal axis shows V from -100 to $+100\ mV$. The vertical axis runs from 0.0 to 1.0. Figures 5–8 show the τ versus voltage. The horizontal axis shows V from -100 to $+100\ mV$. The vertical axes differ and are indicated in the captions for each figure. The gating variables were updated in the standard way, illustrated below for m. Lower bounds were placed on the τ equal to approximately 10% of their peak values.

$$m_\infty = \frac{\alpha_m}{\alpha_m + \beta_m} \quad (7)$$

$$\tau_m = \frac{m_\infty}{\alpha_m} \quad (8)$$

$$\frac{Dm}{Dt} = \frac{m_\infty - m}{\tau_m} \quad (9)$$

The dynamics of the gating variables and the τ are determined by the α and β variables defined below.

$$\alpha_m = \frac{-320(V+52)}{e^{\frac{V+52}{-4}} - 1} \quad (10)$$

$$\beta_m = \frac{-260(V+25)}{e^{\frac{V+25}{5}} - 1} \quad (11)$$

$$\alpha_h = 128e^{\frac{V+48}{-18}} \quad (12)$$

$$\beta_h = \frac{4000}{e^{\frac{V+25}{-6}} + 1} \quad (13)$$

$$\alpha_n = \frac{-32(V+50)}{e^{\frac{V+50}{-5}} - 1} \quad (14)$$

x	$g_x(\mu S)$	$E_x(mV)$
Na	13.823	45
K	16.58	-90
A	27.646	-77
$leak$	1.0915110^{-3}	-64.72
$synapse$	0.009	45

Table 2: maximal conductances and equilibria

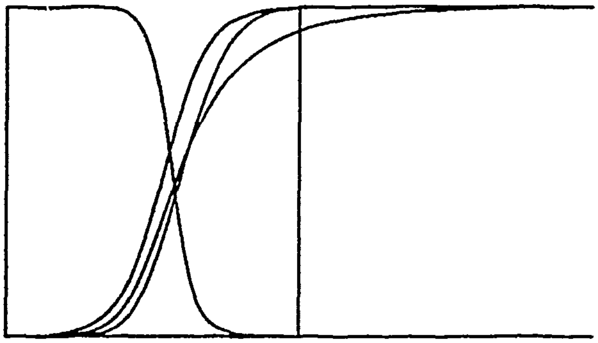


Figure 4: $m_\infty, h_\infty, n_\infty, a_\infty$ vs. V.

$$\beta_n = 500e^{\frac{V+55}{-40}} \quad (15)$$

$$\alpha_a = 0.2e^{(V+46)0.1085} \quad (16)$$

$$\beta_a = 0.2e^{-(V+46)0.0271} \quad (17)$$

Synaptic conductance is modeled by an alpha function: .

$$g_{synapse}(t) = 0.009te^{\frac{-t}{2.5}} \quad (18)$$

Acknowledgements

Thanks to Bill Lytton for providing the data on the pyramidal neuron and for his help implementing the

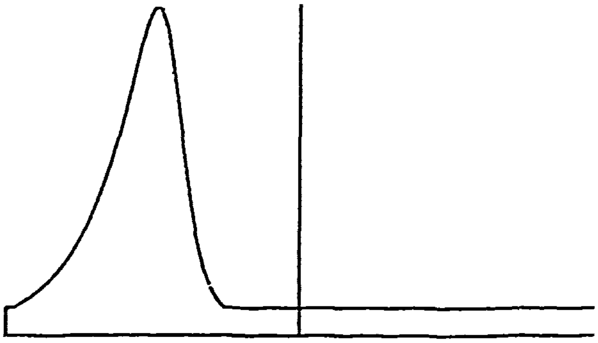


Figure 5: τ_m vs. V. Vertical axis runs from 0.0 to 6.0.

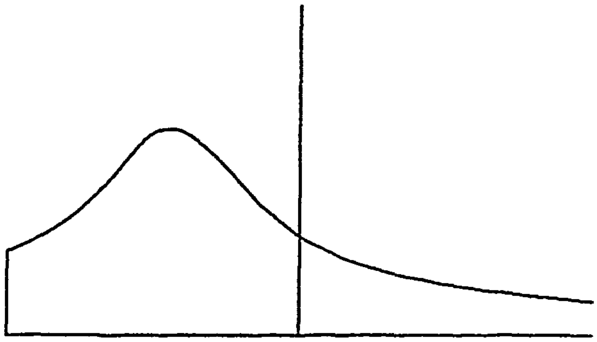


Figure 6: τ_h vs. V Vertical axis runs from 0.0 to 0.2.

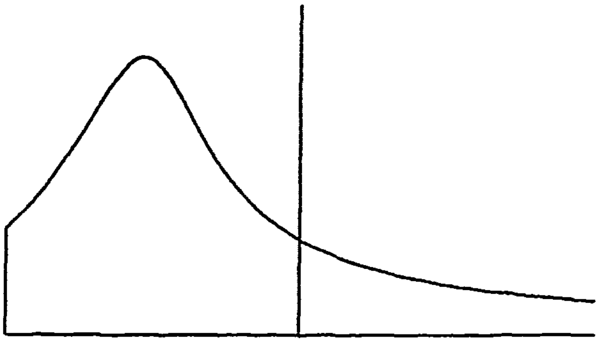


Figure 7: τ_n vs. V Vertical axis runs from 0.0 to 2.0.

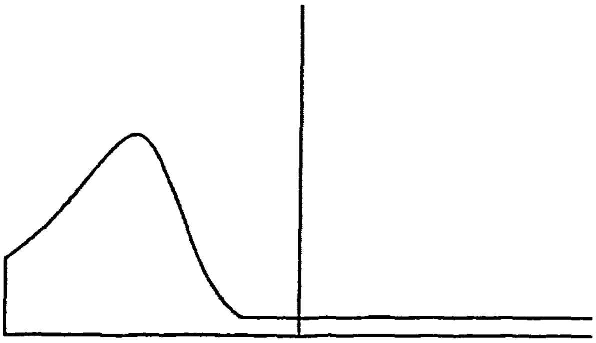


Figure 8: τ_a vs. V Vertical axis runs from 0.0 to 50.0.

simulation. Alan Heirich was supported by an NSF graduate fellowship. Christof Koch acknowledges support by the Air Force Office of Scientific Research. The authors can be reached by electronic mail at *X@aurel.cns.caltech.edu* where X is in the set { *koch, heirich* }.

References

L.J. Borg-Graham. (1987) Simulations suggest information processing roles for the diverse currents in hippocampal neurons. In D. Anderson (ed.), *Neural Information Processing Systems.* New York: American Institute of Physics.

F. Crick. (1984) Function of the thalamic reticular complex: the searchlight hypothesis. *Proceedings of the National Academy of Sciences USA* 81: 4586-4590.

F. Crick & C. Koch. (1990a) Toward a neurobiological theory of consciousness. *Seminar in the Neurosciences* (in press).

F. Crick & C. Koch. (1990b) Some reflections on visual awareness. In *Symposia on Quantitative Biology* 55. Cold Spring Harbor Press.

A. Damasio. (1989) The brain binds entities and events by multiregional activation from convergence zones. *Neural Computation* 1: 123-132.

C. Gray, P. König, A. Engel & W. Singer. (1989) Oscillatory responses in cat visual cortex exhibit intercolumnar synchronization which reflects global stimulus properties. *Nature* 338: 334-337.

C. Gray & W. Singer. (1989) Stimulus-specific neuronal oscillations in orientation columns of cat visual cortex. *Proc. Natl. Acad. Sci. USA* 86:1698-1702.

K. Kuba & S. Nishi. (1979) Characteristics of fast excitatory postsynaptic current in bullfrog sympathetic ganglion cells: effects of membrane potential, temperature and *Ca* ions. *Pflüger's Arch.* 378:205-212.

W. Lytton & T. Sejnowski. (1990) Simulations of cortical pyramidal neurons synchronized by inhibitory interneurons. *Salk Institute for Biological Studies Technical Report.*

C. von der Malsburg. (1981) The correlation theory of brain function. *Internal report 81-2, department of neurobiology, Max-Planck institute for biophysical chemistry, Göttingen, FRG.*

D. Norman. (1986) Reflections on cognition and parallel distributed processing. In J. McClleland & D. Rumelhart (eds.), *Parallel Distributed Processing* 2, 531-546. Cambridge, MA: The MIT Press.

D. Perkel & T. Bullock. (1968) Neural Coding: a report based on an NRP work session. *Neurosciences Research Program Bulletin* 6:221-348.

H. Spitzer, R. Desimone & J. Moran. (1988) Increased attention enhances both behavioral and neuronal performance. *Science* 240:338-340.

M. Stryker. (1989) Is grandmother an oscillation? *Nature* 338: 297-298.

R. Traub. (1982) Simulation of Intrinsic Bursting in CA3 Hippocampal Neurons. *J. Neurosc.* 7: 1233-1242.

A. Treisman. (1982) Perceptual grouping and attention in visual search for features and for objects. *J. Exp. Psych. Human Percept.* 8:194-214.

S. Wise & R. Desimone. (1988) Behavioral neurophysiology: insights into seeing and grasping. *Science* 242:736-741.

W. Yamada, C. Koch & P. Adams. (1989) Multiple channels and calcium dynamics. In C. Koch & I. Segev (eds.), *Methods in Neuronal Modeling*, 97-134. Cambridge, MA: The MIT Press.

Part X

VLSI Implementation

An Analog VLSI Neural Network Cocktail Party Processor

Alan Heirich *
Comp. Sci. & Eng.
Cognitive Science
Univ. Calif., San Diego
La Jolla, CA 92037

Steven Watkins
Electr. & Comp. Eng.
Univ. Calif., San Diego
La Jolla, CA 92037

Michael [illegible]ton
Electr. & Comp. Eng.
Univ. Calif., San Diego
La Jolla, CA 92037

Paul Chau
Electr. & Comp. Eng.
Univ. Calif., San Diego
La Jolla, CA 92037

Abstract

This paper presents the design of an analog CMOS VLSI chip we have fabricated to implement a "neural cocktail party processor" (Malsburg & Schneider, 1986). The job of this processor is to bind related components of an auditory stream for later selection by an attentional mechanism. The binding is represented by phase correlation: feature neurons which are deemed 'related' are forced to fire at common phase and frequency. The model is based on realistic anatomical features of the auditory system with the addition of hypothetical "Malsburg synapses". The chip contains elements to represent feature neurons, a common inhibitor pool, and Malsburg synapses with real time Hebbian learning. This paper explains the model, presents the chip design, and discusses some recent theoretical results (Kammen, Holmes & Koch, 1989) which suggest refinements to the model and to our next iteration of the chip.

1 INTRODUCTION

Imagine you are in a crowded room at a noisy cocktail party. Dozens of people are speaking simultaneously and you have to strain to hear the words of the person standing across from you. Because you want to hear her words you are able to pay attention to them and ignore the other speakers around you. This is what psychologists have called the "cocktail party problem": the ability for an attentional mechanism to discriminate among the multitude of sensory inputs with which it is bombarded, and to track certain inputs to the exclusion of others.

(Malsburg & Schneider, 1986) present a model for a biological mechanism that helps solve the cocktail party problem. Their model segments features of the sensory input and groups together those features which are related to each other. These feature groups are then available as input to higher levels of the system which select the group(s) to attend to. Their model accomplishes the segmentation task based on information present in the auditory signal, preattentively, without any direction by higher level processes.

The model works by "learning" on very short time scales of milliseconds to seconds. The model's inputs are feature neurons which represent the presence of specific components of the auditory signal. Hebbian synapses, known as "Malsburg synapses" (Crick, 1984) learn to bind together feature neurons which are active at similar times. When the synapse between a pair of feature neurons is strengthened the neurons will tend to fire in common phase. Higher levels of the nervous system can discriminate among different sets of feature neurons by using phase as a label. Given enough time the entire set of feature neurons will partition themselves into a handful of differently labelled groups. Each of these groups can then be perceived as a unitary object by an attentional mechanism.

In the remainder of this paper we describe recent evidence about neural coding by phase correlation, some anatomical features of the auditory system and the Malsburg & Schneider model. We then present our design for an analog CMOS VLSI chip which implements the qualitative features of the model. Finally we discuss recent theoretical results (Kammen, Holmes & Koch, 1989) which explain the dynamics of the model and suggest refinements to it.

2 BINDING BY PHASE CORRELATION

Electrical oscillations in cortex have been reported for several years (Freeman, 1978; Eckhorn et. al, 1988) as well as reports of correlated neural firing (Schneider, Eckhorn & Reitböck, 1983). Most recently this

*Present address: Computation and Neural Systems Program, California Institute of Technology, 216-76, Pasadena, CA 91125

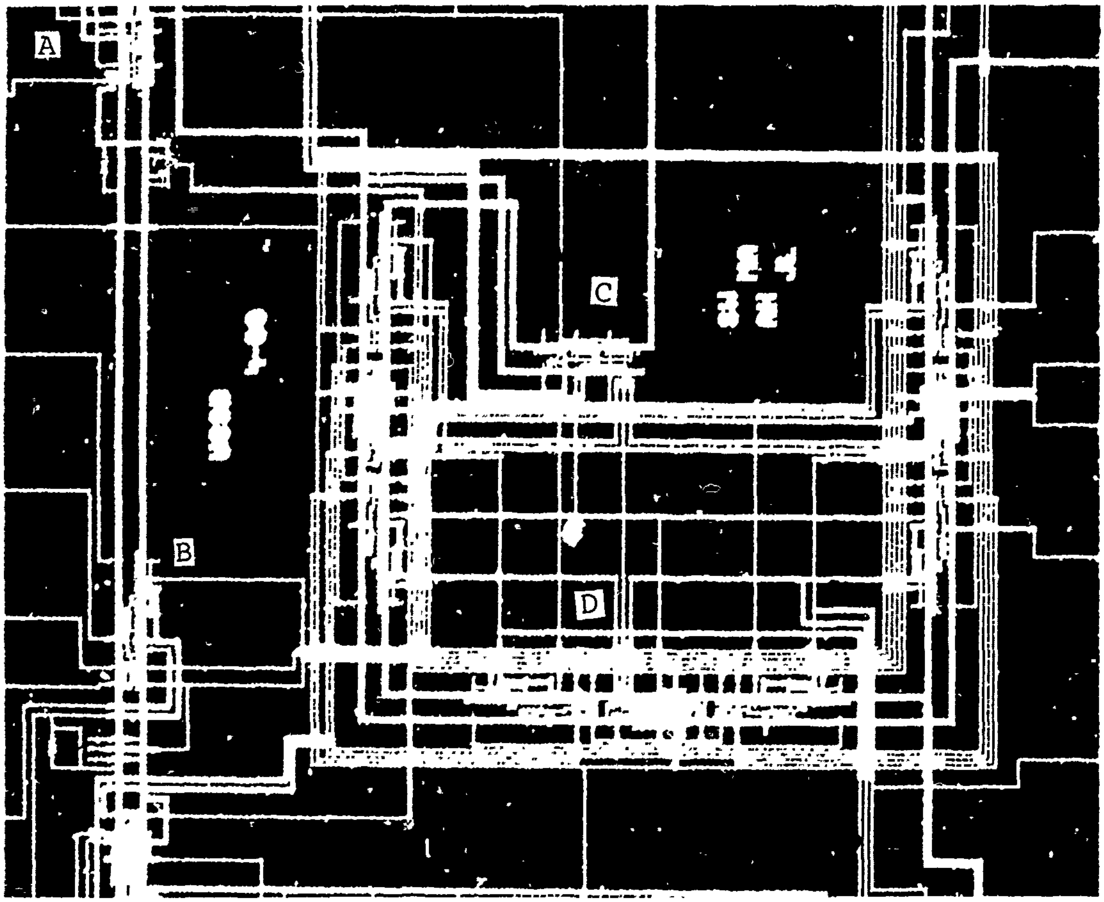


Figure 1: Our first implementation fabricated on a MOSIS "tiny chip". Labels identify. (A) a feature neuron; (B) a Malsburg synapse; (C) the inhibitory pool, (D) a network of 3 feature neurons and Malsburg synapses.

correlation has been shown in primary visual cortex to correspond to the presence of a visual stimulus and further, the correlations depend on global properties of the stimulus (Gray et al, 1989) These developments are consistent with speculations about the representation of information in the brain (von der Malsburg, 1981, Crick, 1984; Damasio, 1989). These speculations suggest that correlation of phase is used by the nervous system to bind related pieces of information. According to this "labeling hypothesis" (Stryker, 1989; Baldi & Atiya, 1989), for example, representations of color and texture for a common object are linked in separate areas of visual cortex by common phase of neural firing. A similar principle might operate within the auditory system. Most recently (Crick & Koch, 1990a, Crick & Koch, 1990b) have suggested that attention may be related to the presence of phase locked oscillations.

3 THE MALSBURG & SCHNEIDER MODEL

The Malsburg & Schneider model uses phase correlation to bind related features of the auditory signal. The mechanism by which this binding occurs uses Malsburg synapses and a feedback pathway in the neural network. Their original model considered only the frequency inputs from the cochlea. We have chosen to include directional information based on psychophysical evidence that it is important for speech segmentation (Dannenbring & Bregman, 1978, Darwin & Gardner, 1986; Darwin, 1990).

From animal studies we know the auditory system performs a considerable amount of feature extraction on the auditory signal between the ear and the auditory cortex The ear receives the physical sound waves and transforms them into neural pulses. In the ear the cochlea (inner ear) produces an instantaneous spectral decomposition of the frequencies present in the auditory signal It is this decomposition which is processed by later stages of the auditory system. The cochlear nucleus performs further feature extraction on the auditory signal, for example, detecting stimulus onset and measuring stimulus duration. Most interestingly, two substructures, the medial and lateral superior olives, detect the spatial location of the origin of each spectral component of the auditory signal, in the horizontal and vertical planes with respect to the two ears. The information from these structures is passed to the acoustic thalamus and from there to the

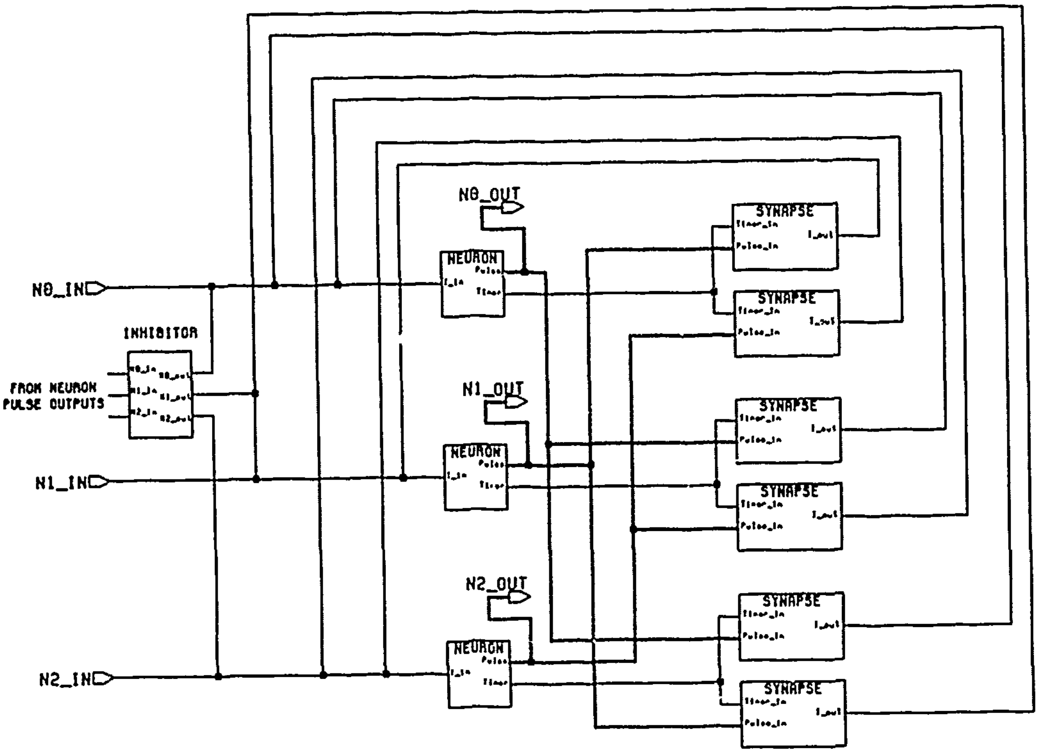


Figure 2: Elements of the network and their interconnections. From left. 3 analog (dendritic) input lines N0_IN, N1_IN, N2_IN; the inhibitory pool (figure 6), 3 feature neurons (figure 4),,3 analog (axonal) output lines N0_OUT, N1_OUT, N2_OUT; 6 unidirectional Malsburg synapses (figure 5).

auditory cortex.

In addition to this feedforward path into cortex there are feedback paths in three locations. First, within the path from ear to thalamus there is feedback from the inferior colliculus back to the dorsal cochlear nucleus. Second, the auditory cortex sends projections back to the acoustic thalamus, which is believed to be concerned with attention. Finally, there is extensive recurrence within the auditory cortex itself. These feedback paths play a crucial role in the model.

To summarize, the auditory system extracts frequency information, direction of sound source, and time of stimulus onset. It passes this information to the auditory cortex through a pathway that includes at least three major avenues for recurrent feedback. Our instantiation of the Malsburg & Schneider model will use frequency and direction information flowing through similar pathways in a recurrent network.

3.1 MALSBURG SYNAPSES

Crick proposed the term "Malsburg synapse" (Crick, 1984) to refer to synapses that perform Hebbian learning on very short time scales of milliseconds to seconds. Malsburg synapses are hypothetical entities and have never been investigated experimentally. Although their existence cannot be assumed they are not inconsistent with known physiology. In the Malsburg & Schneider model the feature neurons are completely interconnected by Malsburg synapses. In the auditory system these synapses might be assumed to exist in the dorsal cochlear nucleus or the auditory cortex. (Malsburg & Schneider, 1986) suggest that the synapses should learn on the order of a few milliseconds roughly the time period of a single action potential, and that this learning should decay over a period of seconds.

3.2 NETWORK ARCHITECTURE

Figure 2 is a diagram of the network architecture. In addition to the recurrent network of feature neurons there is a pool of inhibitory neurons which are excited by the feature neurons. This inhibitory pool feeds back an excitation level which reflects the total number of feature neurons that are active within the past few milliseconds and which inhibits all of the feature neurons equally. The synapses between the feature neurons and the inhibitory pool are fixed in strength. If the Malsburg synapses exist in the cochlear nucleus then this inhibitory pool might exist in the inferior collicu-

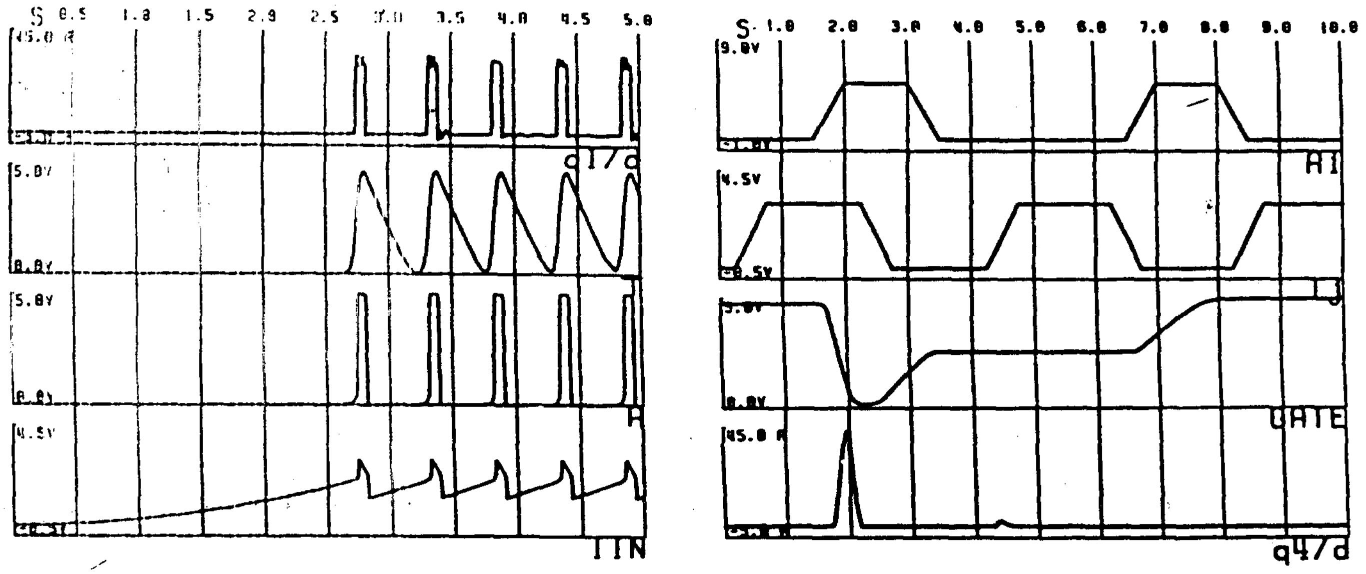


Figure 3: Timing diagrams for (left) **feature neuron circuit (figure 4) and (right) Malsburg synapse circuit (figure** 5). Neuron circuit, from bottom left: **charge builds up (bottom) at axon hillock capacitor C1 until threshold is** reached; action potential output **A is triggered by positive feedback from sodium channels; timer output T marks** time from most **recent action potential; charge is drained by potassium channels Q1, Q2. Malsburg synapse** circuit, from top right: **curves show (top) action potential input Ai, timer input Tj, synaptic strength at GATE,** and propagated **excitatory synaptic potential through Q4. Sequence of events: (1.5–2.0 ms) an action potential** arrives from neuron i **while the timer of neuron j is still active, meaning neuron j fired recently; (2.0–2.5 ms)** charge on synaptic **weight capacitor C1 falls creating current flow to neuron i; (2.5–3.5 ms, 6.75–8.25 ms) action** potential arrives from neuron i **when timer from j has expired, causing increase in charge at C1 thus decreasing** synaptic weight **at GATE.** No **current flows to neuron i.**

lus or the auditory cortex. If the **Malsburg synapses** exist in cortex then the **inhibitory pool must also be** there.

3.3 ORIGIN OF THE CORRELATIONS

An intuitive **explanation for the origin of the correlations was provided by (Malsburg, 1989). The Malsburg** synapses **and the inhibition play opposing roles. The synapses due to their Hebbian nature tend to draw** together **neurons that fire close together in time. The** inhibitory **pool, since it reflects the combined activity** level **of all of the feature neurons, reflects a bias** in favor of the **first neuron in a set that fires within** a temporal window. This is **because as each successive** neuron fires the **level of inhibition rises to reflect** the total activity level. If two **neurons fire once with** a time delay Δt **between them then they will subsequently** fire with $\Delta t + \epsilon_1$, $\Delta t + \epsilon_2$, ... **as the trailing** neuron is **repeatedly inhibited more than the leading** neuron.

4 AN ANALOG VLSI IMPLEMENTATION

Our implementation of the model has three circuit elements: a self-resetting neuron with timer (figure 4); a Malsburg synapse (figure 5); and an inhibitory pool (figure 6). The self-resetting neurons represent the feature neurons in the model. They are completely interconnected by Malsburg synapses. Since each Malsburg synapse is unidirectional there will be $n(n-1)$ Malsburg synapses for n feature neurons. A single large capacitance represents the inhibitor pool. The output of every feature neuron connects to the input of the inhibitory pool and the output of the inhibitory pool connects to the input of every feature neuron (see figure 2). These connections have fixed strength.

Inputs to the network are digital pulses from two sources. One source represents frequency input and the other source represents spatial location (azimuth). In a typical situation several neurons representing frequency would be excited by a signal containing multiple frequencies but only one neuron representing spatial location would be excited.

The digital pulses represent excitatory postsynaptic potentials coming from the cochlea and the olivary structures. At every instant in time every feature neuron either receives a pulse or receives no input. It receives a pulse if the corresponding feature is present in the auditory stream. For example, whenever the cochlea detects a 120 Hz pitch the feature neuron that represents 120 Hz receives a pulse. A feature neuron

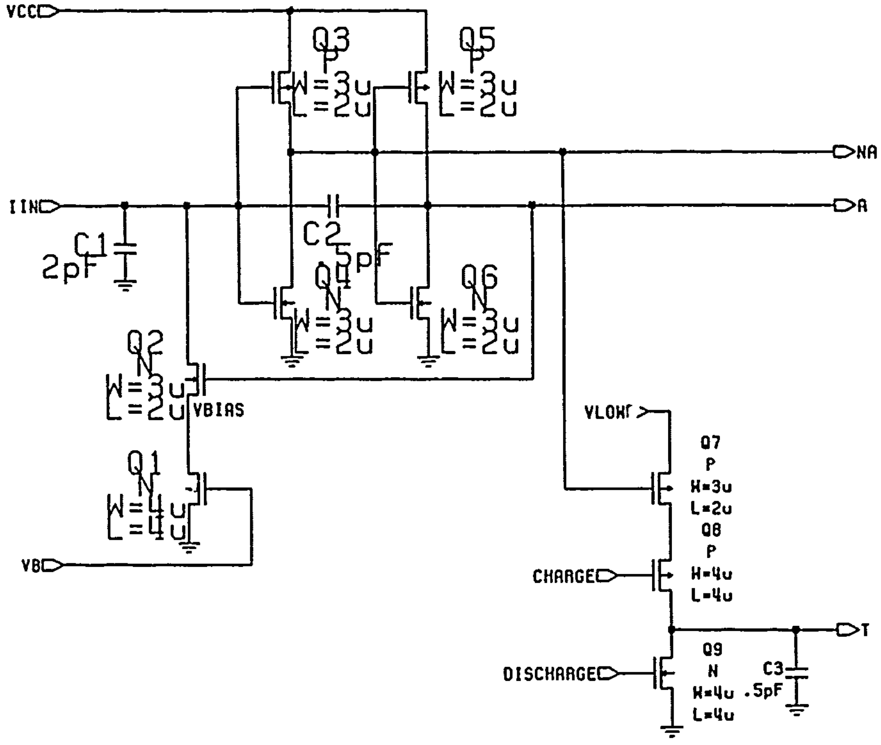


Figure 4. Integrate-and-fire neuron soma with added timer circuit, adapted from (Mead, 1989). From left. dendritic input IIN, axon hillock C1 and discharge path Q1, Q2, sodium channels Q3, Q4, Q5, Q6, C2 create feedback which drives the neuron to fire once above threshold, inhibitory drain path NA and action potential output A, decaying timer subcircuit Q7, Q8, Q9, C3; timer output T. Also pictured. power sources VCC, VLOW, control points VB, CHARGE, DISCHARGE. See figure 3 for timing diagrams.

receives no current if the corresponding feature is absent.

Each neuron sums the current flowing into its input, and when a threshold is reached it fires both a digital pulse and a decaying timing pulse. The digital pulse represents an action potential, while the decaying timing pulse marks a refractory period during which Hebbian learning is possible. When two neurons fire closely together in time, the effect is to strengthen the synapse between them. When two neurons fire farther apart in time the effect is to weaken the synapse. The single inhibitor neuron will affect each neuron equally by subtracting charge from the inputs of all of them.

4.1 DIGITAL AND ANALOG TRANSISTORS

This VLSI device employs both digital and analog transistors. The digital transistors act as switches: they operate either totally "off" allowing essentially no current flow, or totally "on" allowing maximum current flow. The analog transistors act as valves, operating mostly in the subthreshold region and allowing a variable amount of current to flow depending on the gate voltage. In the digital world this region would be considered "off", but in reality some current still flows. The advantages of using a transistor in this manner are threefold: power dissipation is extremely low; the transistor can act as a current source over a large voltage range; and the exponential nonlinearity of current flow versus gate voltage is an ideal computational primitive.

4.2 THE SELF RESETTING NEURON

The feature neuron (figure 4) is a basic integrate-and fire neuron, and the core circuitry for it was taken directly from the self-resetting neuron of (Mead, 1989). The primary output of this circuit is a digital pulse which represents an action potential. We extended the circuit by adding three transistors (Q7, Q8 and Q9) and a capacitor (C3) to create a timer output which is a slowly decaying pulse used in Hebbian learning.

Current from the other neurons and the external input is summed at capacitor C1 which represents the axon hillock. When enough charge has accumulated at this capacitor to create a voltage above threshold,

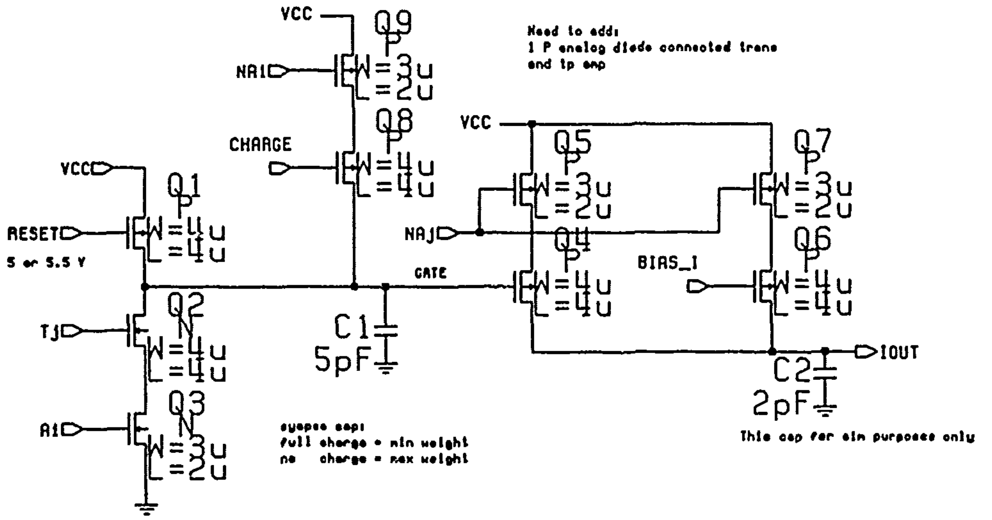


Figure 5. Malsburg synapse with Hebbian learning. From left. timer Tj and action potential Ai inputs, analog "AND" subcircuit Q2, Q3 controls Hebbian learning; synapse weight C1 (weight $\propto \frac{1}{charge}$), forgetting circuit Q8, Q9, NAi; synaptic potential output NAj and generator Q4, Q5, Q6, Q7. Also pictured. power source VCC, control points RESET, CHARGE, BIAS_I.

the neuron will fire. Capacitor C2 is used to create positive feedback which helps the voltage at C1 build up faster once the neuron begins to fire. (Capacitor C2 resembles the voltage activated sodium channels in the axon initial segment.) The VB input controls how quickly charge will leak away from C1 once the neuron fires. (This leaking resembles the inactivating potassium current, and controls how quickly the neuron will inactive once it fires.) The CHARGE and DISCHARGE inputs control how quickly the timer output rises and falls once the neuron fires.

4.3 THE MALSBURG SYNAPSE

The Malsburg synapse circuit (figure 5) stores the value of the synaptic weight and provides the means to dynamically increase or decrease the synaptic weight between neuron *i* and neuron *j* based on how closely neurons *i* and *j* fire together in time

The most important parts of the synapse circuitry are transistors Q2 and Q3. These two transistors essentially form an AND gate. When a timer output pulse from neuron *j* is present at the gate of Q2 AND a neuron pulse from neuron *i* is present at the gate of Q3, the effect is to strengthen the synapse between neuron *j* and neuron *i*. The synaptic strength is represented by the voltage due to the charge stored at capacitor C1. the lower the voltage, the higher the synaptic weight value and vice versa. When both transistors Q2 and Q3 are turned on (due to timer and neuron pulses at their gates), a discharge path from the weight capacitor C1 to ground is created, charge leaks off, the voltage is lowered and, therefore, the weight increases.

Transistors Q8 and Q9 form a path to charge C1. This path decreases the synaptic weight when neurons *i* and *j* do not fire closely together in time. Transistors Q4, Q5, Q6 and Q7 form current sources to supply current to the input of neuron *i* whenever neuron *j* fires. The amount of current supplied is determined by the value on the synaptic weight capacitor C1 at the time that neuron *j* fires, and also by the *BIAS_I* input. The *BIAS_I* input allows a constant bias current to flow to the input of neuron *i* every time neuron *j* fires regardless of the synaptic weight value. This bias circuitry may or may not be used; it was included to make the synapse as flexible as possible.

4.4 THE INHIBITOR POOL

The inhibitor circuit (figure 6) feeds back a function of the total excitation level to every neuron. It inhibits all of the neurons proportional to this excitation level. When any combination of the three self-resetting neurons in the network are on, transistors Q1, Q2, Q3, Q4, Q5 and Q6 form a path to charge capacitor C1. The IKNOB input provides a means to limit the amount of current charging C1 whenever output pulses from the self-resetting neurons are present. C1 in turn controls transistor Q7 in the subthreshold region. When Q7 is on, a leakage path is created from the inputs of the self-resetting neurons to the LRAIL. The more strongly Q7 turns on, the more current is shunted away from the self-resetting neuron inputs which inhibits them from turning on.

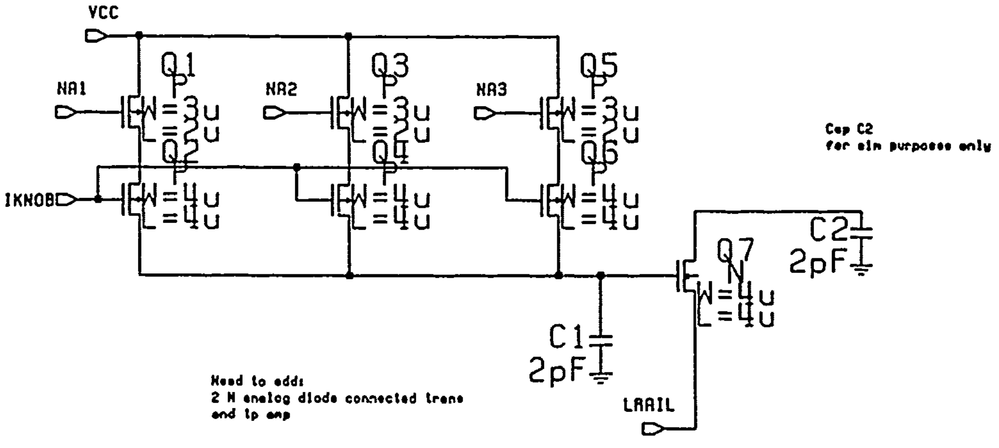


Figure 6. Inhibitory pool circuit. From top. neuron action potentials NA1, NA2, NA3 and summing network Q1, Q2, Q3, Q4, Q5, Q6, inhibitory capacitance C1 and drain path Q7. Also pictured: power source VCC, control point IKNOB.

5 PRESENT STATUS AND FUTURE DIRECTIONS

We have fabricated our first iteration of the chip (figure 1). It contains an interconnected network of three neurons with Malsburg synapses and an inhibitor pool, as well as isolated instances of each of the components. This will allow experimentation with each individual circuit element. As many test points as the 40 pin package would allow are connected to device pins. These test points represent critical nodes in the circuits. for example, the values of the synaptic weights, the digital pulse and timer outputs of the neurons, etc. We will view the neuron's digital outputs on an oscilliscope. We are considering more complex representations of the internal state of the network.

Our purpose in fabricating this first chip was to test our circuit designs and the operation of the model. Subsequent iterations will approach a full scale working system. We have calculated that a MOSIS "big chip" has enough space for approximately 50 neurons, their associated synapses, and the inhibitor. The working system is intended as a companion to the silicon cochlea (Lyon & Mead, 1988) and the auditory localizer (Lazzaro & Mead, 1989), which perform the tasks of the cochlea and the superior olives respectively. The major difficulty that we forsee in integrating these devices is the chip-to-chip interconnect. We have not yet addressed this problem.

Software simulations of the model have shown a problem with using Malsburg synapses as presented in (Malsburg & Schneider, 1986). Real speech waveforms contain transients that are too brief to be tracked by synapses that decay on the order of seconds. This model may be useful for other signal processing tasks, for example, musical perception, where time constants are lower.

Recent theoretical work (Kammen, Holmes & Koch, 1989) suggests that a simpler network architecture, without Malsburg synapses, can produce correlated firing solely as a result of the feedback pathway. In this simpler architecture an ensemble average activity level is fed back to every feature neuron. Each feature neuron locally compares this quantity x to its own activity level, and adds some function $f(\Delta x)$ to its excitation. For appropriate classes of f the governing rate equations have locally stable phase locked solutions.

These results suggest that we may be able to simplify our design by removing the Malsburg synapses, at the expense of slightly more complexity in the individual neurons. Since the number of synapses grows quadratically with respect to the number of neurons this will save a tremendous amount of space on the chip and allow us to roughly square the number of neurons. This possibility is particularly interesting because it might allow us to support devices like the silicon retina (Mead, 1989) if the problem of chip-to-chip interconnect can be solved. Alternatively, it might make our circuit small enough to fit onto an existing chip.

Acknowledgements

Special thanks to Terry Sejnowski, Gary Cottrell and John Lazzaro. Terry and Gary encouraged us in this project and Terry put us in touch with John Lazzaro. John provided circuit design assistance and acted as our design and layout reviewer. Without his help we could never have completed this project. Thanks to Claudia Farb and Joe LeDoux for producing the photograph in figure 1. Alan Heirich was supported by an NSF graduate fellowship. Michael Alston was supported by a National Defense Science & Engineering fellowship.

All except the first'author can be reached by electronic mail at "X@ece.ucsd.edu" where X is in the set { watkins, malston, chau }. The first author can be reached at "heirich@aurel.cns.caltech.edu".

References

P. Baldi & A. Atiya. (1989) Oscillations and synchronizations in neural networks: an exploration of the labeling hypothesis. In D.S. Touretzky, (ed.), *Advances in Neural Information Processing Systems 2*, San Mateo, CA: Morgan Kaufman.

F. Crick. (1984) Function of the thalamic reticular complex: the searchlight hypothesis. *Proceedings of the National Academy of Sciences USA* 81: 4586-4590.

F. Crick & C. Koch. (1990a) Toward a neurobiological theory of consciousness. *Seminar in the Neurosciences* (in press).

F. Crick & C. Koch. (1990b) Some reflections on visual awareness. In *Symposia on Quantitative Biology* 55. Cold Spring Harbor Press.

A. Damasio. (1989) The brain binds entities and events by multiregional activation from convergence zones. *Neural Computation* 1: 123-132.

G. Dannenbring & A. Bregman. (1978) Streaming versus fusion of sinusoidal components of complex tones. *Perception & Psychophysics* 24: 369-376.

C. Darwin. (1990) The relationship between speech perception and the perception of other sounds. To appear in I. Mattingly & M. Studdert-Kennedy (eds.), *Modularity and Motor Theory*, Hillsdale, N.J.: Erlbaum.

C. Darwin & R. Gardner. (1986) Perceptual separation of speech from concurrent sounds. In M.E.H. Schouten (ed.), *The Psychophysics of Speech Perception*, Utrecht, Netherlands: NATO ASI Series.

R. Eckhorn, R. Bauer, Jordan, M. Brosch, W. Kruse, M. Munk, & H. Reitböck. (1988) *Biological Cybernetics* 60: 121-130.

W. Freeman. (1978) Spatial Properties of an EEG Event in the Olfactory Bulb and Cortex. *Electroencephalography and Clinical Neurophysiology* 44: 586-605.

C. Gray, P. König, A. Engel & W. Singer. (1989) Oscillatory responses in cat visual cortex exhibit intercolumnar synchronization which reflects global stimulus properties. *Nature* 338: 334-337.

D. Kammen, P. Holmes, & C. Koch. (1989) Cortical architecture and oscillations in neuronal networks: feedback versus local coupling, in R.M.J. Cotterill (ed.), *Models of Brain Function*, Cambridge. Cambridge University Press.

J. Kelly. (1985) The Auditory System. In E. Kandel & J. Schwartz (eds.), *Principles of Neural Science*. New York: Elsevier.

J. Lazzaro & C. Mead. (1989) A silicon model of auditory localization. *Neural Computation* 1: 47-57.

R. Lyon & C. Mead. (1988) An analog electronic cochlea. *IEEE Transactions on Acoustic, Speech & Signal Processing* 36: 1119-1134.

C. Mead. (1989) *Analog VLSI and Neural Systems*. New York: Addison-Wesley.

J. Schneider, R. Eckhorn & H. Reitböck. (1983) Evaluation of Neuronal Coupling Dynamics. *Biological Cybernetics* 46: 129-134.

M. Stryker. (1989) Is grandmother an oscillation? *Nature* 338: 297-298.

C. von der Malsburg. (1981) The correlation theory of brain function. *Internal report 81-2, department of neurobiology, Max-Planck institute for biophysical chemistry, Göttingen, FRG.*

C. von der Malsburg & W. Schneider. (1986) A Neural Cocktail Party Processor. *Biological Cybernetics* 54. 29-40.

C. von der Malsburg (1989). Personal communication.

A VLSI Neural Network with On-Chip Learning

Shawn P. Day
Dept. of Electrical Engineering
University of British Columbia
Vancouver, B. C., Canada V6T 1W5

Daniel S. Camporese
Dept. of Electrical Engineering
University of British Columbia
Vancouver, B. C., Canada V6T 1W5

Abstract

Feed-forward neural networks can be used to implement Boolean functions which are specified by a set of training examples. These networks are usually trained with the back propagation training technique (Rumelhart et al., 1986a; Rumelhart et al., 1986b). Implementing this training technique in hardware is difficult because of the need for sigmoidal neural transfer functions and real-valued weights. If step transfer functions and ternary-valued weights which take on the values $\{-1, 0, 1\}$ are used instead, hardware implementations become much simpler. This modified network can be trained using combinatorial optimization techniques such as simulated annealing (Kirkpatrick et al., 1983; van Laarhoven and Aarts, 1987), random local search (Aarts and Korst, 1989), and stochastic tunneling (Day and Camporese, 1990). Simulated annealing requires the difficult choice of an appropriate cooling schedule while the random local search can become trapped in local minima. Stochastic tunneling has the ability to escape from local minima and it is often faster than simulated annealing. In this paper we present an electronic VLSI implementation of the stochastic tunneling training algorithm.

1 THE NETWORK MODEL

Layered feed-forward networks with L input units and M output units can be used to map an L-dimensional Boolean space to an M-dimensional Boolean space. The output of unit j is connected to an input of unit i in the succeeding layer with a connection weight w_{ij}. Training such a network involves finding values of these connection weights which satisfy the input/output relationships to be learned. These networks are often trained with the supervised learning technique known as back propagation (Rumelhart et al., 1986a; Rumelhart et al., 1986b). Back propagation requires a differentiable transfer function so that the gradient of an error surface can be calculated. It also requires high precision weights which can be updated in small increments.

We wish to design a feed-forward network which can be easily implemented in VLSI hardware, including the learning circuitry. Since digital VLSI hardware is reliable and well understood, we restrict the weights to the values $\{-1, 0, 1\}$ so that they can be easily implemented as bit storage cells. Also, the differentiable transfer function is replaced with a step or hard-limiting transfer function. Thus, the individual units are familiar linear threshold elements (McCulloch and Pitts, 1943; Rosenblatt, 1962). Their behaviour can be described by the following equation:

$$o_i = \begin{cases} -1 & \text{if } \sum_{j \in S_i} w_{ij} o_j < 0 \\ 1 & \text{otherwise} \end{cases} \tag{1}$$

where o_j is the output from unit j and S_i is the set of all units which have forward connections to unit i. The thresholds can be adjusted by connections to other units whose outputs are always "1". Thus, the network output is a bipolar vector whose components have the values $+1$ or -1. Such linear threshold elements are easily implemented in VLSI hardware, but the step transfer function and restricted weights preclude the use of conventional back propagation.

2 TRAINING

A network with L inputs and M outputs is trained to respond to a set of P input vectors $\mathbf{i}^1, \mathbf{i}^2, ..., \mathbf{i}^P$ by producing the target output vectors $\mathbf{t}^1, \mathbf{t}^2, ..., \mathbf{t}^P$. Each training vector pair, k, has components i_j^k and t_i^k where $j = 1, 2, 3, ..., L$ and $i = 1, 2, 3, ..., M$. The network response to the input vector $\mathbf{i}^k$ is denoted by $\mathbf{o}^k$. Thus, an error function giving the number of erroneous bits in the network response to the input $\mathbf{i}^k$ is

$$E^k = \frac{1}{4}\|\mathbf{t}^k - \mathbf{o}^k\|^2 = \frac{1}{4}\sum_{i=1}^{M}(t_i^k - o_i^k)^2 \qquad (2)$$

and the total number of erroneous bits over all training patterns is

$$E = \sum_{k=1}^{P} E^k = \frac{1}{4}\sum_{k=1}^{P}\sum_{i=1}^{M}(t_i^k - o_i^k)^2 \qquad (3)$$

The error function measures how poorly the current network weights map the training input vectors to the target responses. This error measure is to be minimized by altering the values of the weights. Methods of combinatorial optimization must be used to search for the minimum since the domain and range of the error function are not continuous. Randomization algorithms such as simulated annealing (van Laarhoven and Aarts, 1987) and random local search (Aarts and Korst, 1989) have been applied to networks with real valued weights (Engel, 1988; Baba, 1989; Bremermann and Anderson, 1990). These techniques are appropriate because they are generally easy to implement in hardware and they don't require continuous error functions.

We have presented a randomization algorithm for performing this error minimization which is suitable for VLSI implementation (Day and Camporese, 1990). To begin, we define a training epoch as one complete cycle through the training data set. For each training epoch, all of the weights are altered with the same probability, p. The weight is changed to one of its two possible alternative values with equal probability. Each weight is updated independently so that no global information is required. If the resulting total error decreases or remains constant, the new weights are retained. If the error increases, the previous values of the weights are restored. Such a method is very inefficient when simulated using a serial computer since every weight must be updated during each training epoch. With parallel hardware, however, all of the weights can be updated simultaneously so that the time required for a training epoch becomes independent of the network size.

If $\mathbf{w}$ is a vector describing the current state of the network weights, the Hamming distance in weight space to the updated state $\mathbf{w}'$ follows a binomial distribution. The mean Hamming distance to the new state is given by

$$\mu_{\text{dist}}(p) = Wp \qquad (4)$$

where W is the total number of weights in the network. Increasing p causes the mean distance to the new state to increase. The variance of the new state distribution is given by

$$\sigma^2_{\text{dist}} = Wp(1-p) \qquad (5)$$

As the number of weights W in the network increases, the binomial state distribution approaches a gaussian distribution in Hamming space with mean $\mu_{\text{dist}} = Wp$ and variance $\sigma^2_{\text{dist}} = Wp(1-p)$ according to the Central Limit Theorem (Devore, 1982). The maximum variance occurs at $p = \frac{1}{2}$. Given the error of the current network configuration from equation (3), we wish to update the weights with an optimal value of p. Three criteria were used to determine an appropriate relationship between E and p.

When the error, E, is zero, the network has successfully learned all of the training patterns and p must be zero to preserve this condition. The maximum possible error is $E_{\text{max}} = MP$. We wish the new state distribution to have the greatest variance when the error is at its maximum, so we set $p = \frac{1}{2}$ when $E = E_{\text{max}}$. Thus, on average, half of the weights will be modified during each update when the system error is at a maximum. Finally, when a single bit is in error and $E = 1$, we set $p = 1/W$ so that an average of one weight will be modified during each training epoch. If we choose a simple quadratic relationship between p and E which satisfies these three conditions we obtain:

$$p = \left(\frac{E}{MP}\right)\left(\frac{2MP(MP-E)+W(E-1)}{2W(MP-1)}\right) \qquad (6)$$

The quantity p plays a role like the annealing parameter in simulated annealing, and equation (6) implements a type of annealing schedule. However, p is a function only of the current error and network configuration. It is not an explicit function of time as is usually the case for simulated annealing.

Since the entire perturbation of all the weights is retained only if the error decreases or remains constant, the error can never rise as it does in simulated annealing. Local minima in the error surface can be escaped by a tunneling process whereby the network can move to a distant state of lower error without having to travel through intermediate higher error states. The probability of tunneling long distances is greater in higher error configurations since the number of random weight changes depends on the current error of the system. Such a method allows the network to escape from any local minimum given a sufficient amount of time. The time can still be very long if the local minimum has a low error value and is far away from a region of lower error.

We call this training procedure *stochastic tunneling*. Stochastic tunneling succeeds at performing a nonlocal search of the weight space while maintaining a sense of direction during the entire search. We have shown that stochastic tunneling is often faster than simulated annealing when the weight updates are performed in parallel (Day and Camporese, 1990). Furthermore, unlike simulated annealing, the algorithm doesn't require the choice of a different cooling schedule for each task.

Another probabilistic learning technique which has been intensively studied is the Boltzmann learning algorithm (Ackley et al., 1985, Hinton and Sejnowski, 1986). Boltzmann learning is used with recurrent networks which have symmetric connectivity ($w_{ij} = w_{ji}$). The algorithm requires probabilistic neurons whose outputs can be observed and correlated over time, and many-valued weights which can be updated in small increments. The many-valued weights lead to a large chip area since the synapses dominate the layout of a fully connected network. The Boltzmann machine has been implemented using electronic VLSI circuitry (Alspector et al., 1989). Hardware implementations of both Boltzmann machines and stochastic tunneling require the generation of probabilistic signals.

2.1 ALTERNATIVE RELATIONSHIPS BETWEEN THE GLOBAL ERROR AND THE WEIGHT CHANGE PROBABILITY

The relationship between p and E given in equation (6) becomes non-monotonic if

$$W < \frac{2(MP)^2}{2MP-1}$$

An increase in the error can sometimes lead to a decrease in the weight change probability. A more reasonable approach is the exponential relationship:

$$p = \frac{1}{2} - \frac{1}{2}e^{-KE}, \qquad \text{where } K = -\ln(1-\frac{2}{W}) \quad (7)$$

The constant K is chosen so that $p = 1/W$ when a single bit of the training set is in error. Here the weight change probability asymptotically approaches $\frac{1}{2}$ and is a monotonic function of E. We can also choose a linear relationship between p and E:

$$p = \begin{cases} \frac{E}{W} & \text{if} \quad F \leq \frac{W}{2} \\ \frac{1}{2} & \text{if} \quad E > \frac{W}{2} \end{cases} \qquad (8)$$

Two network structures and sets of training data were chosen to investigate the behaviour of the three relationships (6), (7), and (8).

1. Parity

   The parity problem is a generalized version of exclusive-OR. The network contains four input units and a single output unit. The output is '1' if the number of '1' inputs is odd, and '-1' if the number of '1' inputs is even. The set of training data consists of sixteen four-bit input vectors and sixteen single-bit output vectors. The network topology contains two hidden layers consisting of five units each.

2. Local Minimum

   This network and set of training patterns was designed to have a local minimum in its error function. There are three input units, a single output unit, and two hidden layers consisting of a single unit each. The six training vectors and target responses are shown in Table 1.

Table 1: Training Vectors for the Network Containing a Local Minimum

Input			*Target*
-1	-1	1	1
-1	1	-1	1
-1	1	1	-1
1	-1	-1	1
1	-1	1	-1
1	1	-1	-1

Both networks were trained using stochastic tunneling with the three relationships (6), (7), and (8). These relationships completely characterize the behaviour of the learning algorithm, so there are no additional parameters to be determined. Figures 1 and 2 show the results from an average of 100 training runs for the two sets of test data. Learning curves for simulated annealing and a random local search are also plotted for comparison. The random local search alters a single weight chosen at random, and accepts this change only if the error remains the same or decreases. For simulated annealing a new network state is chosen in the same way as for the random local search, but states which increase the error are accepted with a probability dependent on a temperature parameter T (van Laarhoven and Aarts, 19[illegible] We chose an initial temperature of $T_0 = 1$ and [illegible] it in steps of 0.2. At each temperature $10W$ weight updates were performed, where W is the total number of weights in the network.

Figure 2 shows how the random local search gets caught in a local minimum of the error function. Simulated annealing also gets caught, but not as often. Increasing the length of the cooling schedule would further reduce this tendency at the expense of longer training times. Stochastic tunneling never gets caught in local minima since the probability of a weight changing is reduced to zero only when a global minimum has been found. For stochastic tunneling it is apparent that the precise form of the relationship between p and E has little effect on the required training time. For a hardware implementation it would be appropriate to use the most easily realizable function.

3 HARDWARE REALIZATION

A VLSI implementation of the network architecture has been designed which performs the learning directly on the chip. For a 20-neuron network with 100 training patterns the hardware will be approximately 20,000 times faster than the serial simulations, allowing for

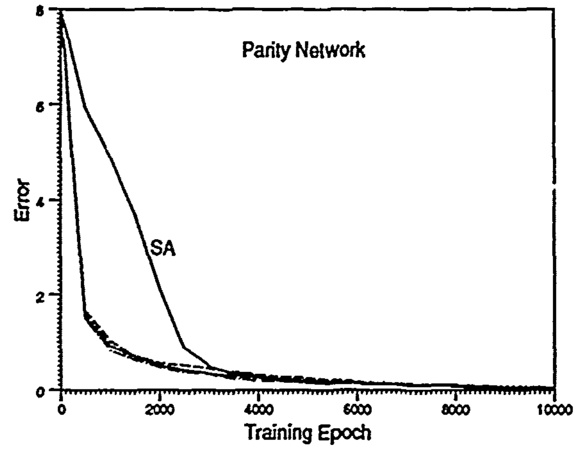


Figure 1: Learning Results for the Parity Network. The unlabeled cur/es are for quadratic (dot), exponential (dash), and linear (solid) relationships between the global error and the weight change probability. A further unlabeled curve (dot-dash) represents the results for a random local search. Finally, a curve for simulated annealing (SA) is included. All of the curves except for simulated annealing are virtually identical.

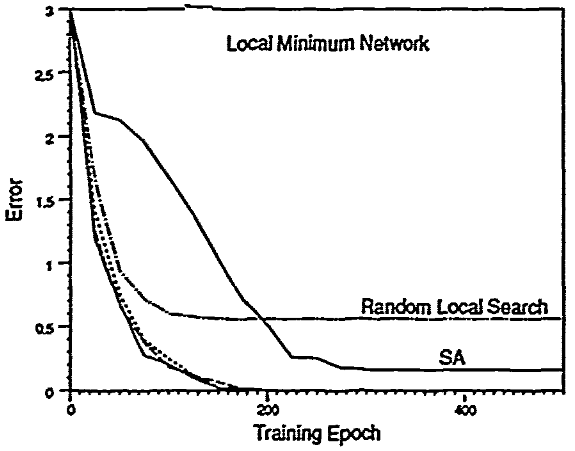


Figure 2: Learning Results for the Local Minimum Network. The unlabeled curves are for quadratic (dot), exponential (dash), and linear (solid) relationships between the global error and the weight change probability. Learning curves for the Random Local Search and Simulated Annealing (SA) are included for comparison.

a more extensive investigation of the network learning properties. For larger networks the speed-up will be even greater since the feed-forward evaluation time is independent of the network size (assuming a fixed depth of one or two hidden layers). A block diagram for one layer of the network is shown in Figure 3. The neurons implement the hard limiting transfer function and their outputs can be fed into the next layer. A single network layer is implemented on a single chip, so a multilayer network will be comprised of multiple chips connected in succession. The design is based on a 3 micron CMOS process. A neuronal output of "1" is represented by a potential of 5 volts and an output of "-1" is represented by a potential of 0 volts.

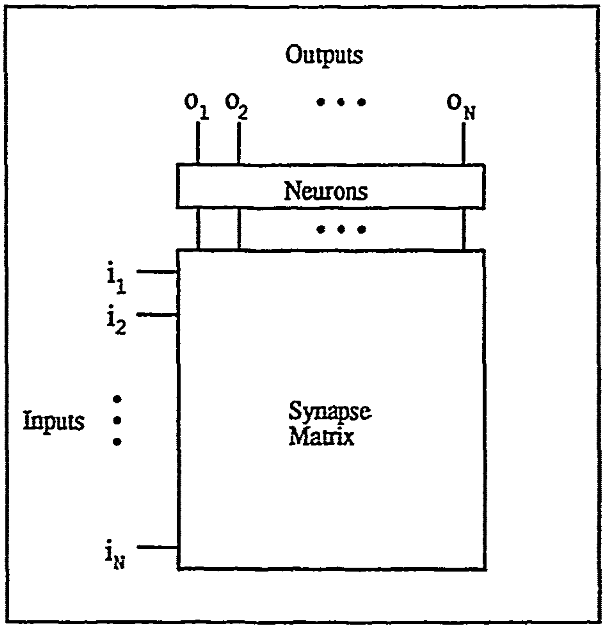


Figure 3: Circuit Block Diagram for the Neural Network Chip. The input lines feed into the synapse matrix horizontally. An analog summation is performed for the input of each neuron along a vertical bus in the synapse matrix.

A single synapse is shown in Figure 4. It is based on a previously published design using separate excitatory and inhibitory summation lines (Verleysen et al., 1989), but it uses fewer transistors to accomplish the same task. Also, our design uses p-channel rather than n-channel synaptic transistors. The p-channel transistors pass less current and decrease the synaptic power dissipation. The weight w_{ij} is stored in memory cells A and B. When the INPUT line is high, representing a value of 1, the $\overline{\text{INPUT}}$ line is low. Transistor T3 is turned on and the synapse can source current onto the INHIBIT and EXC TE lines depending on the states of transistors T1 and T2. If cell A is storing a "0" then transistor T1 is turned on, increasing the current in the EXCITE line. This condition represents an excitatory synaptic connection. An inhibitory synaptic connection is achieved when cell B is storing a "0", increasing the current in the INHIBIT line through transistor T2. If cell A and cell B both store a "1" then the synapse is in the disconnected state where $w_{ij} = 0$. The state where cell A and cell B both store a "0" is not used. When the input line is low, representing a value of -1, transistor T6 is turned on and the current is increased in the alternate summation lines. Thus, transistors T1

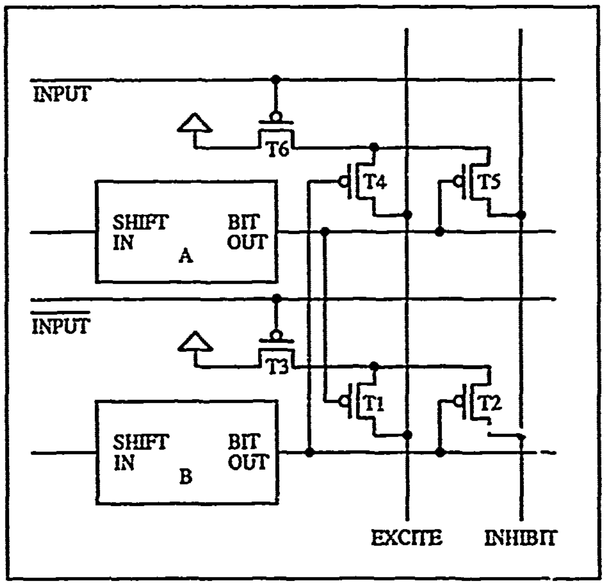


Figure 4: A Single Synaptic Connection. The weight w_{ij} is stored in registers A and B.

through T6 implement the multiplication of - weight with an input signal. The weight is storeـ in cells A and B according to Table 2.

Table 2: Weight Storage in Registers A and B

Weight	A	B
1	0	1
0	1	1
-1	1	0

Memory cells A and B are each implemented by the circuit of Figure 5. The circuit contains both a primary memory cell and a secondary or "backup" memory cell. In operation, the primary cells store the values of the synaptic connections. During the learning phase, the logic value stored in the primary cells is "backed up" into the secondary cells. The random updates are made to the primary cells in the circuit and if the total error of the network configuration increases, the primary cells are restored from the secondary cells. Two of these cell structures are required for each synapse in order to implement both inhibitory and excitatory connections as discussed above. All of the synaptic bit cells in the network are connected in the form of a serial shift register with the aid of transistor T3 so that the synaptic values can loaded onto or read from the chip serially. Figure 6 shows the control waveforms required for the quiescent, backup, restore, and shifting functions of the cell.

The neurons are implemented by the circuitry shown in Figure 7 which is similar to a circuit described by Verleysen (Verleysen et al., 1989), but with the polarity reversed. The output is 5V if the current in the EXCITE line is greater than the current in the INHIBIT line and 0V in the opposite case. Thus, if there are more active excitatory connections than inhibitory connections, the output of the neuron will be 5V. If there are more active inhibitory connections the output will be 0V. If equal numbers of excitatory and

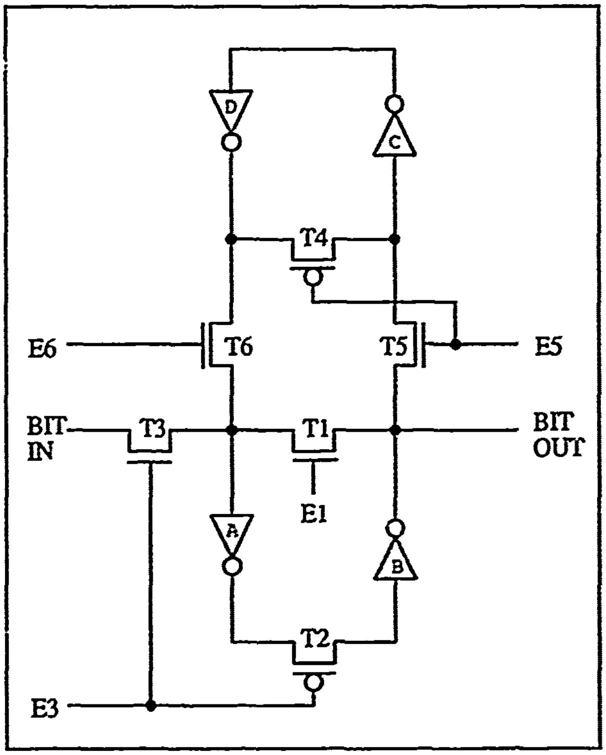


Figure 5: Synaptic Bit Cell. The primary cell consists of transistors T1 and T2 and inverters A and B. The secondary cell consists of transistor T4 and inverters C and D.

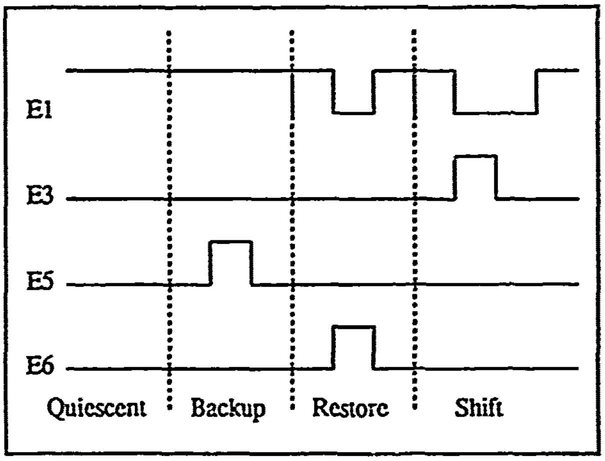


Figure 6. Control Waveforms for the Synaptic Bit Cell

inhibitory connections are active we want the output of the neuron to be 5V in order to satisfy equation (1). To ensure this condition, an extra synaptic transistor which is always on is connected to the EXCITE line. This transistor is half the size of the regular synaptic transistors so that it can be overridden by a single inhibitory connection.

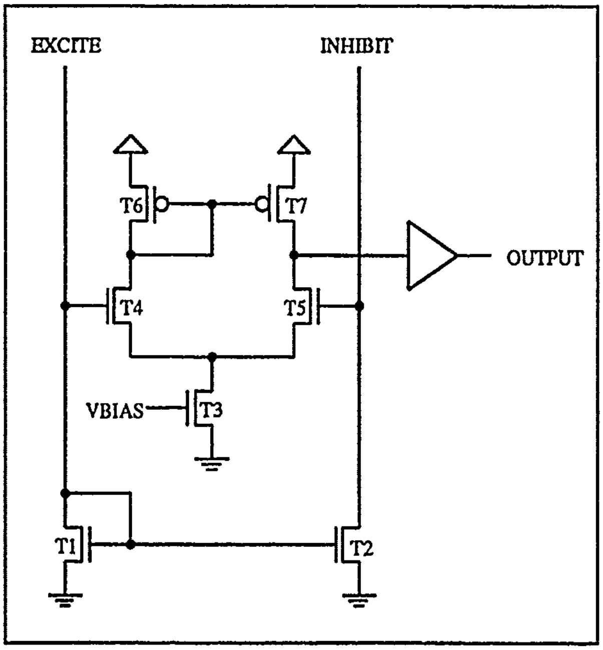


Figure 7: A Single Neuron. The currents in the EXCITE and INHIBIT lines are converted into voltages by transistors T1 and T2, and then compared by the transconductance amplifier (Mead, 1989) consisting of transistors T3 through T7.

During training with the stochastic tunneling algorithm, the current error is obtained by presenting all of the training patterns to the network and observing its outputs. The error signal controls a random bit generator which is used to alter the values of the synaptic connections. If there are N synaptic connections then the synaptic matrix is shifted circularly N times so that the synaptic values will be back in their correct locations when the shifting is complete. The random bit generator is introduced at one point in the circular shift chain and alters the value of the synaptic connection at that point with a probability dependent upon the total error of the current network configuration. Thus, a complete update of all of the weights requires on the order of W time steps.

The neural network chip is approximately 0.5 centimeters square. The size of a single synapse is 153 by 112 mic·ons and the chip contains 480 such synapses. There are 20 neurons fully connected to 24 input lines. The neurons occupy a space of 153 by 122 microns. Functional and electrical simulations of the final design have performed as expected.

Compared with an electronic implementation of the Boltzmann machine (Alspector et al., 1989), our chip requires much less area for a network of fixed size. This area reduction is due mainly to the ternary-valued weights which require only two binary storage cells. However, the ternary-valued weights will likely give our network less representational power than a Boltzmann machine of similar topology.

4 PARALLEL RANDOM NUMBER GENERATION

The hardware described above does not take full advantage of the parallel nature of the training algorithm. Since all of the weights are updated independently, it is possible to perform the updates in parallel using only local information. A complete random number generator must be placed at each synaptic site in order to supply update information to all of the synapses simultaneously. Alspector et al. have described an analog noise generator based on the amplification of thermal noise in the channel of a transistor (Alspector et al., 1989). When many such generators are placed on a single chip, they tend to influence one another due to induced fluctuations on the supply rails and other coupling effects. Thus, it is extremely difficult to generate uncorrelated noise signals. Also, Alspector's noise generator requires three stages of amplification which consume a large chip area. The large area is not a great problem for the Boltzmann machine since only one noise generator is required for each neuron. The size of these noise amplifiers makes them infeasible for our application since we require one noise generator for each synapse. Finally, the resolution of these noise generators is limited by the characteristics of the transistors which can be fabricated.

We have decided to use digital pseudorandom noise generators rather than analog devices in order to circumvent the correlation problem. Multi-bit numbers must be generated to obtain the required resolution. A single bit cell of a random number generator is placed at each synaptic site to satisfy the area constraint. The generated bit stream is observed over time in order to obtain a random number with any desired resolution. Increasing the resolution of the generator is accomplished by running it for a longer period of time.

A binary signal having any desired probability of being "1" can be generated from this random bit train as follows. For n-bit precision, let the random bit train r_i represent the binary expansion of a number N_r between 0 and 1:

$$N_r = \sum_{i=1}^{n} r_{-i} 2^{-i} \qquad 0 \leq N_r \leq 1 - 2^{-n}$$

The desired probability p can also be represented as an n-bit binary expansion of a number between 0 and 1:

$$p = \sum_{i=1}^{n} p_{-i} 2^{-i} \qquad 0 \leq p \leq 1 - 2^{-n}$$

Now define a binary variable Q with an initial value of $Q_0 = 0$. Q is updated iteratively on the bits of N_r and p according to:

$$Q_{i+1} = p_{i-n} \wedge (r_{i-n} \oplus p_{i-n}) \ \vee \ Q_i \wedge \overline{(r_{i-n} \oplus p_{i-n})} \quad (9)$$

where $i = 0, 1, ..., n-1$. After n updates, Q_n will satisfy:

$$Q_n = \begin{cases} 1 & \text{if } p > N_r \\ 0 & \text{if } p \leq N_r \end{cases}$$

Thus, if N_r is uniformly distributed, the expected value of Q_n will be $\langle Q_n \rangle = p$. Equation (9) is implemented with the circuitry of Figure 8. The bits p_i of the desired probability are supplied sequentially by external error measuring circuitry. The random bits r_i are generated by a pseudorandom number generator based on chaotic sequences in cellular automata (Wolfram, 1983; Wolfram, 1984; Wolfram, 1985; Wolfram, 1986). A single cell of the cellular automaton is placed at each synaptic site. Each of the cells is implemented by the circuitry of Figure 9. A clock pulse updates the state of the bit based on its current state and the states of its immediate neighbors to the left and right. Periodic boundary conditions are applied so that the cells effectively form a ring.

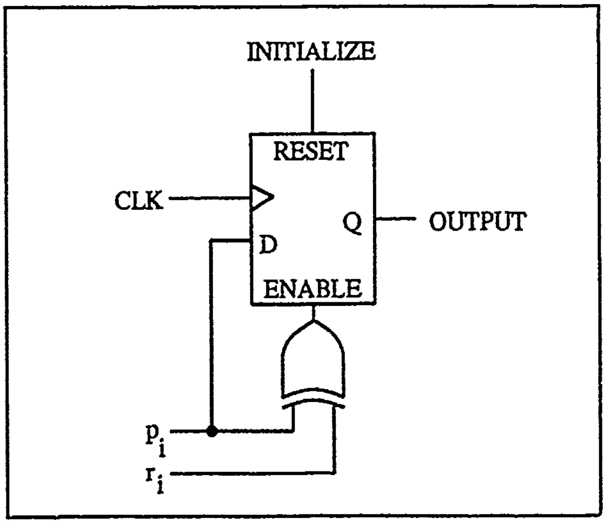


Figure 8: Synaptic Update Circuitry. The bits p_i represent the desired probability for a weight change and the bits r_i represent a locally generated uniform random number N_r.

The properties of this random number generator have been thoroughly investigated (Hortensius et al., 1989b; Hortensius et al., 1989a). It was shown that adjacent cells in the generator produce partially correlated bit sequences, but the correlation dies out over time. Thus, to avoid correlation the cellular automata can be updated several times between each set of random bits which is extracted. Hortensius showed that performing four updates between each extraction produces high quality uncorrelated random bit sequences. With this new circuitry the network can be completely updated in a constant time, independent of the number of weights. An order of magnitude increase in training speed should be realized for small networks such as the examples in section 2.1. The speed-up will be even greater as the sizes of the networks increase.

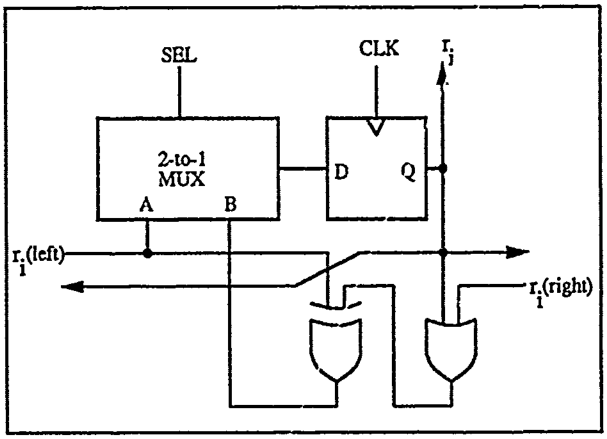


Figure 9: A Single Cell of the Random Number Generator. The SEL line to the multiplexor allows the cells to be connected as a serial shift register so that the initial random number can be seeded.

We plan to fabricate a second version of the chip which incorporates this parallel random number generation circuitry. Using the same 3 micron CMOS process we expect the resulting size of a synapse to double when compared with the original design. As a result, the size of network which can be placed on a single chip will be approximately halved. To get around this limitation we are investigating the possibility of fabricating separate synaptic matrix and neuron chips which can be used as building blocks for constructing arbitrary network topologies.

5 CONCLUDING REMARKS

The well known back propagation network can be modified to permit simple implementation using VLSI technology. The units with differentiable (usually sigmoidal) transfer functions are replaced with linear threshold elements and the weights are restricted to the values $\{-1, 0, 1\}$. Unfortunately, these modifications preclude the use of back propagation as a training technique, but methods of combinatorial optimization may be used instead.

Randomization algorithms such as simulated anneal ing, random local search, and stochastic tunneling can be used to train these networks. Stochastic tunneling is amenable to VLSI implementation and we have designed an integrated circuit which incorporates this learning technique directly on the chip. Simulations of the chip have performed as expected, and we are currently waiting for the silicon to come back from the foundry. We are also designing a second version of the chip which incorporates the parallel random number generation circuitry described in section 4. This new chip will allow the weight updates of the entire network to be performed in parallel, providing an order of magnitude increase in training speed for small networks. For large networks the increase in training speed will be even greater. We plan to fabricate this second chip after we have finished testing the first.

Acknowledgements

We wish to thank Terry Sejnowski for helpful comments concerning this paper. Shawn Day wishes to thank the Natural Sciences and Engineering Research Council of Canada for financial support. Both authors wish to acknowledge the support of the Canadian Microelectronics Corporation.

References

Aarts, E. and Korst, J. (1989). *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing.* Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons Ltd.

Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive Science*, 9:147–169.

Alspector, J., Gupta, B., and Allen, R. B. (1989). Performance of a stochastic learning microchip. In Touretzky, D. S., editor, *Advances in Neural Information Processing Systems 1.* Morgan-Kaufmann Publishers, Inc., San Mateo, CA.

Baba, N. (1989). A new approach for finding the global minimum of error function of neural networks. *Neural Networks*, 2(5):367–373.

Bremermann, H. J. and Anderson, R. W. (1990). An alternative to back-propagation: A simple rule of synaptic modification for neural net training and memory. Technical Report PAM-483, Center for Pure and Applied Mathematics, University of California, Berkeley.

Day, S. P. and Camporese, D. S. (1990). A stochastic training technique for feed-forward neural networks. In *International Joint Conference on Neural Networks, 1990*, volume 1, pages 607–612, San Diego, CA.

Devore, J. L. (1982). *Probability and Statistics for Engineering and the Sciences.* Brooks/Cole Publishing Company, Monterey, California.

Engel, J. (1988). Teaching feed-forward neural networks by simulated annealing. *Complex Systems*, 2(6):641–648.

Hinton, G. E. and Sejnowski, T. J. (1986). *Learning and Relearning in Boltzmann Machines*, volume 1 of *Parallel Distributed Processing. Explorations in the Microstructure of Cognition*, chapter 7, pages 282–317. MIT Press, Cambridge, MA.

Hortensius, P. D., McLeod, R. D., and Card, H. C. (1989a). Parallel random number generation for VLSI systems using cellular automata. *IEEE Transactions on Computers*, 38(10):1466–1473.

Hortensius, P. D., McLeod, R. D., Pries, W., Miller, D. M., and Card, H. C. (1989b). Cellular automata-based pseudorandom number generators for built-in self-test. *IEEE Transactions on Computer-Aided Design*, 8(8):842–859.

Kirkpatrick, S., Gelatt, Jr., C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133.

Mead, C. (1989). *Analog VLSI and Neural Systems.* Addison-Wesley.

Rosenblatt, F. (1962). *Principles of Neurodynamics.* Spartan Books, New York.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986a). *Learning Internal Representations by Error Propagation*, volume 1 of *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, chapter 8, pages 318–362. MIT Press, Cambridge, MA.

Rumelhart, D. E., Hinton, G E., and Williams, R. J. (1986b). Learning representations by back-propagating errors. *Nature*, 323(9):533–536.

van Laarhoven, P. J. M. and Aarts, E. H. L. (1987). *Simulated Annealing: Theory and Applications.* D. Reidel Publishing Company, Dordrecht, Holland.

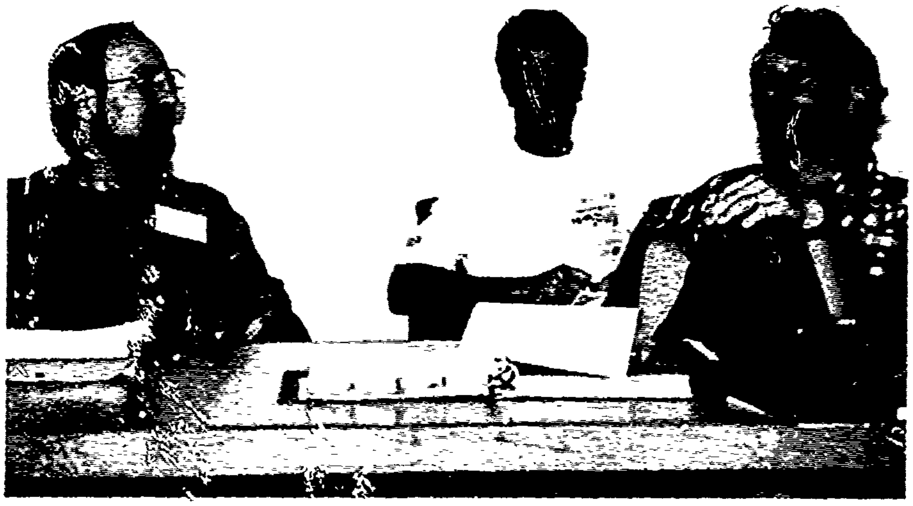
Verleysen, M., Sirletti, B., Vandemeulebroecke, A. M., and Jespers, P. G. A. (1989). Neural networks for high-storage content-addressable memory: VLSI circuit and learning algorithm. *IEEE Journal of Solid-State Circuits*, 24(3):562–568.

Wolfram, . (1983). Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55(3):601–644.

Wolfram, S. (1984). Universality and complexity in cellular automata. *Physica 10D*, pages 1–35.

Wolfram, S. (1985). Origins of randomness in physical systems. *Physical Review Letters*, 55(5):449–452.

Wolfram, S. (1986). Random sequence generation by cellular automata. *Advances in Applied Mathematics*, 7:123–169.



CONNECTIONIST MODELS

Edited by David S. Touretzky (Carnegie Mellon University), Jeffrey L. Elman (University of California, San Diego), Terrence J. Sejnowski (The Salk Institute, UC San Diego), and Geoffrey E. Hinton (University of Toronto)

This volume comprises a collection of original research papers that summarize research at international labs and universities at the forefront of neural network research. The Connectionist Models Summer Schools bring together distinguished researchers and outstanding graduate students to provide a forum for presenting and evaluating current research results on connectionist models in neural networks.

The papers, rigorously selected from faculty and student entries, have been updated and revised to incorporate the evaluations discussed at the workshop, as well as extensive interaction between the authors and the editors of this volume. The selections represent a wide variety of current research in VLSI design, optimization methods, learning theory, vision, speech, neuroscience, linguistics, and cognitive psychology. This collection, like its successful predecessor, will be a valuable reference for researchers and students in AI, neurobiology, cognitive science and other areas with an interest in neural networks.

MORGAN KAUFMANN PUBLISHERS
P.O. Box 50490
Palo Alto, CA 94303

Neural Networks,
Artificial Intelligence,
Cognitive Science

ISBN 1-55860-156-2
90000>
9 781558 601567