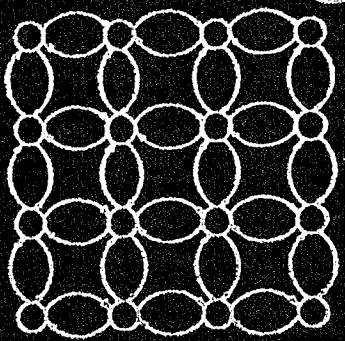
Neural Networks for Signal Processing V

PROCEEDINGS OF THE 1995 IEEE WORKSHOP

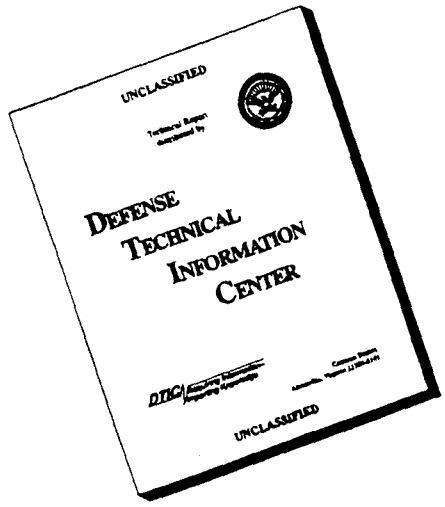


Edited by
Federico Girosi
John Makhoul
Elias Manolakos
Elizabeth Wilson

Approved for public release
Distribution unlimited

19960104 088

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

NEURAL NETWORKS FOR SIGNAL PROCESSING V

PROCEEDINGS
OF THE
1995
IEEE WORKSHOP

Fifth in a Series of Workshops
Organized by the IEEE Signal Processing Society
Neural Networks Technical Committee

Accesion For
NTIS CRA&I
DTIC TAB
Unannounced
Justification
By
Distribution /
Availability Co
Dist | Avail and / o Special
A-1

Edited by

Federico Girosi
John Makhoul
Elias Manolakos
Elizabeth Wilson

Published under the sponsorship of the
IEEE Signal Processing Society
in cooperation with the IEEE Neural Networks Council
and co-sponsored by ARPA/ONR and NSF/CBCL-MIT

The Institute of Electrical and Electronics Engineers, Inc.
New York, New York

DTIC QUALITY INSPECTED 1

1995 Neural Networks for Signal Processing V

Abstracting is permitted with credit to the source. Libraries are permitted to photocopy beyond the limits of U.S. copyright law for private use of patrons those articles in this volume that carry a code at the bottom of the first page, provided the per-copy fee indicated in the code is paid through the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923. Instructors are permitted to photocopy isolated articles for non commercial classroom use without fee. For other copying, reprint, or republication permission, write to the IEEE Copyright Manager, IEEE Service Center 445 Hoes Lane, P.O Box 1331, Piscataway, NJ 08855-1331. All rights reserved. Copyright 1995 by The Institute of Electrical and Electronics Engineers, Inc.

IEEE Catalog Number: 95TH8094
ISBN Softbound: 0-7803-2739-X
ISBN Microfiche: 0-7803-2740-3

Library of Congress: 95-77353

Additional copies of this publication are available from

IEEE Service Center
PO Box 1331
445 Hoes Lane
Piscataway, NJ 08855-1331

1-800-678-IEEE
1-908-981-1393
1-908-981-9667 (Fax)
833-233 (Telex)

Neural Networks for Signal Processing
Proceedings of the 1995 IEEE-SP Workshop

The chapters in this book are based on presentations given at the IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing.

NNSP'95 TECHNICAL PROGRAM COMMITTEE

Joshua Alspector	(Bellcore, USA)
Charles Bachmann	(Naval Research Lab., USA)
Alice Chiang	(MIT Lincoln Lab., USA)
A. Constantinides	(Imperial College, UK)
Lee Giles	(NEC Research, USA)
Federico Girosi	(CBCL, MIT, USA)
Lars Kai Hansen	(Tech. U. of Denmark, Denmark)
Yu-Hen Hu	(U. of Wisconsin, USA)
Jenq-Neng Hwang	(U. of Washington, USA)
Bing-Huang Juang	(AT&T Bell Lab., USA)
Shigeru Katagiri	(ATR Japan)
George Kechriotis	(Thinking Machines Inc., USA)
Stephanos Kollias	(National Tech. U. of Athens, Greece)
Sun-Yuan Kung	(Princeton U., USA)
Gary M. Kuhn	(Siemens Corp. Research, USA)
Richard Lippmann	(MIT Lincoln Lab., USA)
John Makhoul	(BBN Lab., USA)
Elias Manolakos	(CDSP, Northeastern U., USA)
P. Takis Mathiopoulos	(U. of British Columbia, Canada)
Mahesan Niranjan	(Cambridge U., UK)
Tomaso Poggio	(CBCL, MIT, USA)
Jose Principe	(U. of Florida, USA)
Wojtek Przytula	(Hughes Research Lab., USA)
John Sorensen	(Tech. U. of Denmark, Denmark)
Andreas Stafylopatis	(National Tech. U. of Athens, Greece)
John Vlontzos	(Intracom S.A., Greece)
Raymond Watrous	(Siemens Corp. Research, USA)
Christian Wellekens	(Eurecom, France)
Ron Williams	(Northeastern U., USA)
Barbara Yoon	(ARPA, USA)
Xinhua Zhuang	(U. of Missouri, USA)

PREFACE

This book contains refereed papers presented at the Fifth IEEE Workshop on Neural Networks for Signal Processing (NNSP'95) at the Royal Sonesta Hotel, Cambridge, MA, on August 31st - September 2nd, 1995.

NNSP'95 was sponsored by the Neural Networks Technical Committee of the IEEE Signal Processing Society, in cooperation with the IEEE Neural Network Council and with co-sponsorship from ONR/ARPA and NSF (through CBCL, the Center for Biological and Computational Learning at MIT). The Workshop is designed to serve as a regular forum for researchers from universities and industry who are interested in interdisciplinary research on neural networks for signal processing applications. NNSP'95 offers a showcase for current research results in key areas, including learning algorithms, network architectures, speech processing, image processing, computer vision, adaptive signal processing, medical signal processing, digital communications and other applications.

Our deep appreciation is extended to Prof. Abu-Mostafa of Caltech, Prof. John Moody of Oregon Graduate Institute, Prof. S.Y. Kung, of Princeton U., Prof. Michael I. Jordan of MIT and Dr. Vladimir Vapnik of AT&T Bell Labs, for their insightful plenary talks. Thanks to Dr. Gary Kuhn of Siemens Corporate Research for organizing a wonderful evening panel discussion on "Why Neural Networks are not Dead". Our sincere thanks go to all the authors for their timely contributions and to all the members of the Program Committee for the outstanding and high-quality program. We would like to thank the other members of the Organizing Committee: Finance Chair Dr. Judy Franklin of GTE Lab. Inc., and Local Arrangements Chair Mary Pat Fitzgerald of MIT, for the superb job they have done.

It is worth mentioning that this year we did not generate any post office mail and the whole NNSP'95 organization and review of papers were handled electronically. Special thanks are extended to the Publicity Chair Marney Smyth of MIT for maintaining the NNSP'95 WWW home page (URL: http://www.cdsp.neu.edu/info/ nnsp95.html) and running an effective publicity "campaign" on the Internet. Also to Anna Patch of the Communications and Digital Signal Processing (CDSP) Center for research and graduate studies at Northeastern U. for managing the database, and to Stylianos Markogiannakis of CDSP for writing the software that allowed us to handle very efficiently the review process. We plan to make this software available on ftp for future conference organizations.

Finally we would like to acknowledge Dr. Barbara Yoon of ARPA for her continued enthusiasm and support in this emerging cross-disciplinary field.

Elias S. Manolakos, *CDSP Center, Northeastern U.*
Federico Girosi, *CBCL, MIT*
John Makhoul, *BBN*
Beth Wilson, *Raytheon Company*

Table of Contents

Preface v

Part 1 Theory

Missing and Noisy Data in Nonlinear Time-Series Prediction 1
Volker Tresp

Non-Linear Time Series Modeling with Self-Organization Feature Maps 11
Jose C. Principe, Ludong Wang

Neural Networks for Function Approximation 21
H.N. Mhaskar, L.Khachikyan

Empirical Generalization Assessment of Neural Network Models 30
Jan Larsen, Lars Kai Hansen

Active Learning the Weights of a RBF Network 40
Kah-Kay Sung, Partha Niyogi

A Novel Approach to Pattern Recognition Based on Discriminative Metric Design 48
Hideyuki Watanabe, Tsuyoshi Yamaguchi, Shigeru Katagiri

A Maximum Entropy Approach for Optimal Statistical Classification 58
David Miller, Ajit Rao, Kenneth Rose, Allen Gersho

Simultaneous Design of Feature Extractor and Pattern Classifier Using the Minimum Classification Error Training Algorithm 67
K.K.Paliwal, M. Bacchiani, Y. Sagisaka

Discriminative Subspace Method for Minimum Error Pattern Recognition 77
Hideyuki Watanabe, Shigeru Katagiri

A unifying view of Stochastic Approximation Kalman Filter and Backpropagation 87
Enrico Capobianco

Globally-Ordered Topology-Preserving Maps Achieved with a Learning Rule Performing Local Weight Updates Only 95
Marc M. Van Hulle

A Self-Organizing System for the Development of Neural Network Parameter Estimators 105
Michael Manry

Recognition of Oscillatory Signals Using a Neural Network Oscillator 115
Masakazu Matsugu, Chi-Sang Poon

Principal Feature Classification 125
Donald W. Tufts, Qi Li

A Habituation Based Neural Network Structure for Classifying Spatio-Temporal Patterns 135
Bryan W. Stiles, Joydeep Ghosh

A Numerical Approach for Estimating Higher Order Spectra Using Neural Network Autoregressive Model 145
Naohiro Toda, Shiro Usui

Fuzzy Neural Network Approach Based on Dirichlet Tesselations for Nearest Neighbor Classification of Patterns 153
K. Blekas, A. Likas, A. Stafylopatis

The Dynamics of Associative Memory with a Self-Consistent Noise 162
Ioan Opris

Recursive Nonlinear Identification using Multiple Model Algorithm 171
Visakan Kadirkamanathan

Mutual Information in a Linear Noisy Network 181
Alessandro Campa, Paolo Del Giudice, Nestor Parga, Jean-Pierre Nadal

Constrained Pole-Zero Filters as Discrete-Time Operators for System Approximation 191
Andrew D. Back, Ah Chung Tsoi

Prior Knowledge and the Creation of "Virtual" Examples for RBF Networks 201
F. Girosi, N. Chan

Part 2 Speech Processing

Speaker Verification using Phoneme-Based Neural Tree Networks and Phonetic Weighting Scoring Method 213
Han-Sheng Liou, Richard J. Mammone

Scaling Down: Applying Large Vocabulary Hybrid HMM-MLP Methods to Telephone Recognition of Digits and Natural Numbers 223
Kristine Ma, Nelson Morgan

Combining Local PCA and Radial Basis Function Networks for Speaker Normalization 233
Cesare Furlanello, D. Giuliani

Discriminatory Measures for Speaker Recognition 243
Kevin R. Farrell

From Artificial Neural Network Inversion to Hidden Markov Model Inversion: Application to Robust Speech Recognition 253
Seokyong Moon, Jenq-Neng Hwang

Hierarchical Mixtures of Experts Methodology Applied to Continuous Speech Recognition 263
Ying Zhao, Richard Schwartz, Jason Sroka, John Makhoul

A Speech Recognizer with Low Complexity Based on RNN 272
Claus Kasper, Herbert Reininger, Dietrich Wolf, Harald Wust

Automatic Speech Segementation Using Neural Tree Network (NTN) 282
Manish Sharma, Richard Mammone

Part 3 Image Processing and Computer Vision

Motion Estimation and Segmentation using a Recurrent Mixture of Experts Architecture 293
Yair Weiss, Edward H. Adelson

Using perceptron-like algorithms for the analysis and parameterization of object motion 303
M. Mattavelli, E. Amaldi

A Multiple Scale Neural System for Boundary and Surface Representation of SAR Data 313
Stephen Grossberg, Ennio Mingolla, James Williamson

A Neural Network Approach to Face/Palm Recognition 323
S.Y. Kung, Shang-Hung Lin, Ming Fang

A Probabilistic DBNN with Applications to Sensor Fusion and Object Recognition 337
Shang-Hung Lin, S.Y. Kung, Long-Ji Lin

Sample Weighting when Training Self-Organizing Maps for Image Compression 343
Jari Kangas

Estimating Image Velocity with Convected Activation Profiles: Analysis and Improvements for Special Cases 351
Robert K. Cunningham, Allen M. Waxman

Pruning Projection Pursuit Models for Improved Cloud Detection in AVIRIS Imagery 361
Charles M. Bachmann, Eugene E. Clothiaux, John W. Moore, Dong Q. Luong

A New Learning Scheme for the Recognition of Dynamic Handwritten Characters 371
Fidimahery Andrianasy, Maurice Milgram

Velocity Measurement of Granular Flow with a Hopfield Network 380
Jingeol Lee, Jose C. Principe, Daniel M. Hanes

Neural Network Based Image Segmentation for Image Interpolation 388
Stefano Marsi, Sergio Carrato

Learning a Distribution-based Face Model for Human Face Detection 398
Kah-Kay Sung, Tomaso Poggio

Action-Based Neural Networks for Effective Recognition of Images 407
Vassilios N. Alexopoulos, Stefanos D. Kollias

Feature-Locked Loop and its Application to Image Databases 417
Alex Sherstinsky, Rosalind W. Picard

An Error Diffusion Neural Network for Digital Image Halftoning 427
Barry L. Shoop, Eugene K. Ressler

Part 4 Applications and Implementations

Estimation of the Glucose Metabolism from Dynamic PET-Scans Using Neural Networks 439
Claus Svarer, Soren Holm, Niels Morch, Olaf Paulson and L.K. Hansen

Nonlinear Echo Cancellation Using a Partial Adaptive Time Delay Neural Network 449
A.N. Birkett, R.A. Goubran

Customized ECG Beat Classifier Using Mixture of Experts 459
Yu Hen Hu, Surekha Palreddy, Willis J. Tompkins

Semiautomated Extraction of Decision Relevant Features from a Raw Data Based Artificial Neural network Demonstrated by the Problem of Saccade Detection in EOG Recordings of Smooth Pursuit Eye Movements 465
Peter K. Tigges, Norbert Kathmann, Rolf R. Engel

EEG Signal Classification with Different Signal Representations 475
Charles W. Anderson, Saikumar V. Devulapalli, Erik A. Stolc

Design and Evaluation of Neural Classifiers - Application to Skin Lesion Classification 484
Mads Hintz-Madsen, Lars Kai Hansen, Jan Larsen, Eric Olesen and Krzysztof T. Drzewiecki

A Study of the Application of the CMAC Artificial Neural Network to the Problem of Gas Sensor Array Calibration 494
Parag M. Bajaria, Bruce E. Segee

Classification of Gamma Ray Signals Using Neural Networks 504
N.G. Bourbakis, A. Tacsillo, M. Tacsillo

Adaptive Preprocessing for On-Line Learning with Adaptive Resonance Theory (ART) Networks 513
Harald Ruda, Magnus Snorasson

Intelligent Network Monitoring 521
Cynthia S. Hood, Chuanyi Ji

A Robust Backward Adaptive Quantizer 531
Dominique Martinez, Woodward Yang

A Maximum Partial Likelihood Framework for Channel Equalization by Distribution Learning 541
Tulay Adali, Xiao Liu, Kemal Sonmez

Constructive Neural Network Design for the Solution of Two State Classification: Problems with Application to Channel Equalization 551
Catherine Z.W. Hassell Sweatman, Gavin J. Gibson, Bernard Mulgrew

A Parallel Mapping of Backpropagation Algorithm for Mesh Signal Processor 561
Shoab A. Khan, Vijay K. Madisetti

Digital Neuroimplementations of Visual Motion-Tracking Systems 571
Anna Maria Colla, Luca Trogu, Rodolfo Zunino

Level Crossing Time Interval Circuit for Micropower Analog VLSI Auditory Processing 581
Nagendra Kumar, Gert Cauwenberghs, Andreas G. Andreou

Part 5 Communications

Optimum Lag and Subset Selection for Radial Basis Function Equaliser 593
Eng-Siong Chng, Bernard Mulgrew, Shen Chen, Gavin Gibson

Channel Equalization by Finite Mixtures and EM Algorithm 603
Lei Xu

Comparison of a Neural Network based Receiver to the Optimal and Multistage CDMA Multiuser Detectors 613
George Kechriotis, Elias S. Manolakos

Author Index 623

Theory

Missing and Noisy Data in Nonlinear Time-Series Prediction

Volker Tresp and Reimar Hofmann
Siemens AG, Central Research
81730 Munich, Germany*

Abstract

We discuss the issue of missing and noisy data in nonlinear time-series prediction. We derive fundamental equations both for prediction and for training. Our discussion shows that if measurements are noisy or missing, treating the time series as a static input/output mapping problem (the usual time-delay neural network approach) is suboptimal. We describe approximations of the solutions which are based on stochastic simulations. A special case is K-step prediction in which a one-step predictor is iterated K times. Our solutions provide error bars for prediction with missing or noisy data and for K-step prediction. Using the K-step iterated logistic map as an example, we show that the proposed solutions are a considerable improvement over simple heuristic solutions. Using our formalism we derive algorithms for training recurrent networks, for control of stochastic systems and for reinforcement learning problems.

1 Introduction

Missing data in time-series prediction are a common problem in many applications. The goal is to obtain valid predictions even if some measurements become unavailable or are not recorded. Similarly, training data are often incomplete. In this paper we analyze this problem from a probabilistic point of view. In previous publications the problem of learning and prediction with missing and noisy features in (static) estimation problems was examined (see, for example [2, 3, 4]). The solutions for both prediction and learning consisted of integrals over the unknown variable weighted by the conditional probability density of the unknown variable given the known variables. The basic idea is the same for missing data in time-series prediction, but here, we can exploit the fact that the missing measurement itself is part of the time series. Similar

*Volker.Tresp@zfe.siemens.de, Reimar.Hofmann@zfe.siemens.de

0-7803-2739-X/95 $4.00 © 1995 IEEE

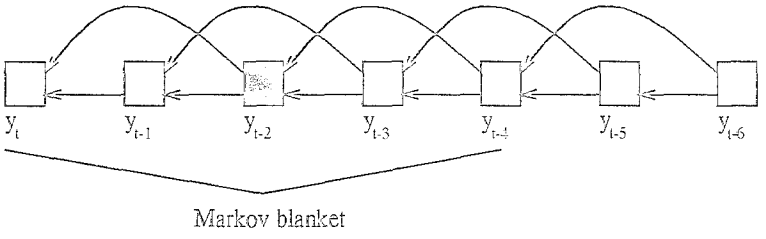


Figure 1: A time series unfolded in time. The arrows indicate that the next realization of the time series can be predicted from the two most recent values, $y_t = f(y_{t-1}, y_{t-2}) + \epsilon_t$. Here, y_{t-2} is assumed to be missing. The Markov blanket shows which variables are relevant for estimating y_{t-2}.

issues arise if we can only obtain noisy measurements of the underlying true time series.

In this paper, we provide solutions for the problem of prediction and training with missing or noisy data in time-series predictions. As a special case, we consider K-step prediction in which a one-step predictor is iterated K times. We show how error bars can be derived for prediction with missing and noisy inputs and for K-step prediction. Finally, we point out that the learning algorithms derived in this paper can be used to train recurrent neural networks, for stochastic control and for reinforcement learning problems.

2 Prediction with Missing Inputs

2.1 One Missing Realization

We assume that the underlying probabilistic model of the time series can be described by

$$y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-N}) + \epsilon_t \tag{1}$$

where $f()$ is either known or approximated sufficiently well by a function approximator such as a neural network. ϵ_t is assumed to be additive uncorrelated zero-mean noise with probability density $P_\epsilon(\epsilon)$ and typically represents unmodeled dynamics. The conditional probability density of the predicted instance of the time series is then

$$P(y_t|y_{t-1}, y_{t-2}, \ldots, y_{t-N}) = P_\epsilon(y_t - f(y_{t-1}, y_{t-2}, \ldots, y_{t-N})). \tag{2}$$

Often, Gaussian noise is assumed such that

$$P(y_t|y_{t-1}, y_{t-2}, \ldots, y_{t-N}) = G(y_t; f(y_{t-1}, \ldots, y_{t-N}), \sigma^2) \tag{3}$$

where $G(x; c, \sigma^2)$ is our notation for a normal density evaluated at x with center c and variance σ^2.

It is convenient to unfold the system in time which leads to the system shown in Figure 1. The realizations of the time series are now random variables in a probabilistic network. Our problem is to predict y_t using the available information. According to our assumptions, the joint probability density is

$$P(y_1, y_2, \ldots, y_t) = P(y_1, \ldots, y_N) \prod_{l=N+1}^{t} P(y_l | y_{l-1}, \ldots, y_{l-N}). \tag{4}$$

Let's now assume that y_{t-k} with $k \leq N$ is missing. Let $y^u = \{y_{t-k}\}$ and let $y^m = \{y_{t-1} \ldots y_{t-k-N}\} \setminus \{y_{t-k}\}$. We can calculate the expected value of the next realization of the time series as

$$E(y_t | M_{t-1}) = \int f(y_{t-1}, \ldots, y_{t-k}, \ldots, y_{t-N})\, P(y^u | y^m)\, dy^u \tag{5}$$

where M_{t-1} stands for all measurements up to $t-1$. The last equation is the fundamental equation for prediction with missing data. Note, that the unknown y_{t-k} is not only dependent on realizations of the time series *previous* to $t-k$ but also on measurements *after* $t-k$. The reason is that the variables in $y^m \cup y_t$ form a minimal Markov blanket of y_{t-k} in the Bayesian net in Figure 1. A minimal Markov blanket in a Bayesian network consists of the direct parents, the direct successors of a variable and all direct parents of a variables direct successor. In our case, the direct successors are $y_t \ldots y_{t-k+1}$, the direct parents are $y_{t-k-1} \ldots y_{t-k-N}$ and the direct parents of a variables direct successor are $y_{t-1} \ldots y_{t-k-N+1}$. The theory of Bayesian and Markov networks now tells us that a variable is independent of all other variables in the network if the variables in the Markov blanket are known (see Figure 1). This discussion shows that simply approximating $y_{t-k} \approx f(y_{t-k-1}, y_{t-k-2}, \ldots, y_{t-k-N})$ is suboptimal. The required conditional density in Equation 5 is (recall that $y^u = y_{t-k}$)

$$P(y^u | y^m) \propto P(y_{t-1} | y_{t-2}, \ldots, y_{t-k}, \ldots, y_{t-1-N})$$

$$\times P(y_{t-2} | y_{t-3}, \ldots, y_{t-k}, \ldots, y_{t-2-N}) \ldots P(y_{t-k} | y_{t-k-1}, \ldots, y_{t-k-N}).$$

This expression can be evaluated easily using Equation 1 or in the Gaussian noise case Equation 3.

2.2 Several Missing Realizations

From the preceding discussion it should be clear that nothing changes if the missing realizations are separated by more than N known realizations. Then the Markov blankets of the missing variable are still completely known. If this is not the case we obtain Equation 5 where $y^u \subseteq \{y_{t-1}, y_{t-2}, \ldots, y_{t-N}\}$ denote all missing instances between $t-1$ and $t-N$ of the time series and where $y^m \subseteq \{y_{t-1}, y_{t-2}, \ldots, y_1\}$ denote the set of all measurement up to $t-1$. Also $P(y^u | y^m) \propto P(y_{t-1}, \ldots, y_2, y_1)$ where the right-hand side is obtained from Equation 4.

2.3 Training with Missing Realizations

We consider the case that $y_1, \ldots, y_t$ are possible realizations. Let $y^m \subseteq \{y_1, \ldots, y_t\}$ denote the set of all measurements and $y^u = \{y_1, \ldots, y_t\} \setminus y^m$ the set of all unknowns. Our model is assumed to be a neural network parameterized by a set of weights w

$$f(y_{t-1}, \ldots, y_{t-N}) \approx NN_w(y_{t-1}, \ldots, y_{t-N})$$

or any other kind of parameterized function approximator. The log-likelihood function of the time series is $L = \log \int P^M(y_t, y_{t-1}, \ldots, y_2, y_1)\, dy^u$. Here

$$P^M(y_t, y_{t-1}, \ldots, y_2, y_1) = P^M(y_N, \ldots, y_1) \prod_{l=N+1}^{t} P^M(y_l | y_{l-1}, \ldots, y_{l-N}). \quad (6)$$

is an approximation to the joint density and

$$P^M(y_t | y_{t-1}, y_{t-2}, \ldots, y_{t-N}) = P_\epsilon(y_t - NN_w(y_{t-1}, y_{t-2}, \ldots, y_{t-N})). \quad (7)$$

For backpropagation learning or other gradient based learning algorithms we need the gradient of the log-likelihood with respect to the weights which is[1]

$$\frac{\partial L}{\partial w} = \sum_{l=N+1}^{t} \int \frac{\partial \log P^M(y_l | y_{l-1}, \ldots, y_{l-N})}{\partial w} P^M(y^{u(l)} | y^m)\, dy^{u(l)}. \quad (8)$$

In case of Gaussian noise, $\frac{\partial L}{\partial w} \propto$

$$\sum_{l=N+1}^{t} \int (y_l - NN_w(y_{l-1}, \ldots, y_{l-N})) \frac{\partial NN_w(y_{l-1}, \ldots, y_{l-N})}{\partial w} P^M(y^{u(l)} | y^m)\, dy^{u(l)}.$$

where $y^{u(l)} = y^u \cap \{y_l, \ldots y_{l-N}\}$ are the missing realizations in the input of the network. The last equation shows that if all $y_l \ldots y_{l-N}$ are known, the integral "disappears".

3 Prediction and Training with Noisy Measurements

Let again $y_t = f(y_{t-1}, y_{t-2}, \ldots, y_{t-N}) + \epsilon_t$ but now we assume that we have no access to y_t directly. Instead, we measure $z_t = y_t + \delta_t$ where δ_t is independent zero-mean noise. Let $z = \{z_1 \ldots z_{t-1}\}$ and $y = \{y_1 \ldots y_t\}$. The joint probability density is

$$P(y, z) = P(y_N, \ldots, y_1) \prod_{l=N+1}^{t} P(y_l | y_{l-1}, \ldots, y_{l-N}) \prod_{l=1}^{t} P(z_l | y_l).$$

[1] Assuming known initial conditions for $y_1, \ldots, y_N$. In this paper, we use repeatedly that if $f(x) > 0$, then $\frac{\partial f(x)}{\partial x} = \frac{\partial \log f(x)}{\partial x} f(x)$.

The expression for the expected value of the next instance of the time series (prediction) is

$$E(y_t|z) = \int f(y_{t-1}, \ldots, y_{t-N})\ P(y_{t-1}, \ldots, y_{t-N}|z)\ dy_{t-1} \ldots dy_{t-N}. \qquad (9)$$

Similarly the gradient of the likelihood for training can be calculated. For the special case of Gaussian noise, with $z = \{z_1 \ldots z_t\}$

$$\frac{\partial L}{\partial w} \propto \sum_{l=N+1}^{t} \int (y_l - NN_w(y_{l-1}, \ldots, y_{l-N}))\ \frac{\partial NN_w(y_{l-1}, \ldots, y_{l-N})}{\partial w}$$

$$\times P^M(y_l, \ldots, y_{l-N}|z)\ dy_l \ldots dy_{t-N}.$$

4 Approximations

4.1 Approximations of the Solution

In general, if $f()$ is a nonlinear function the equations we obtained for prediction and for calculating the gradient cannot be solved analytically and must be approximated numerically. We will discuss a solution based on Monte Carlo sampling. Note that all solutions have the general form $\int h(u,m)P(u|m)du$ where u is the set of unknown variables and m is the set of known variables. An integral of this form can be solved by drawing random samples of the unknown variables following $P(u|m)$. Let $u^1, \ldots, u^S$ denote these samples. Then we can approximate

$$\int h(u,m)P(u|m)du \approx \frac{1}{S}\sum_{s=1}^{S} h(u^s, m).$$

The problem now reduces to sampling from $P(u|m)$. Let's first assume that only one variable is missing. Then the problem reduces to sampling from a one-variate distribution which can be done using *sampling-importance-resampling* or other sampling techniques [1].

If more than one realization is missing the situation becomes more complicated. The reason is that the unknown variables are in general dependent and we have to draw from the distribution of all unknowns. A general solution is Gibbs sampling. In Gibbs sampling we initialize the unknown variables either randomly or better with reasonable initial values. Then we select one of the unknown variables u_i and pick a sample from $P(u_i|m, u \setminus u_i)$ and set u_i to that value. Then we repeat the procedure for the next unknown variables and so on. Discard the first samples. Then samples are produced with the correct distribution. This of course means that we might have to sample all unknowns which ever occurred in the time series. In practice, one would restrict the sampling to some reasonable chosen time window. Note, that in the missing data case, if N consecutive values are known the coupling is broken

and we do not need to consider missing values which lie further away. Also sampling is simple if only samples of *future* values are required as in K-step prediction (Section 5) and in control problems (Section 6.3). The reason is that we can sample forward in time. Note, that sampling does not work for deterministic systems. Finally, we want to point out the simplicity behind the complicated looking solutions. Both for prediction and training we draw samples of the unknown variables according to their probability density. In prediction we substitute those samples for the missing data and average the predictions. In training calculate the average of the error gradients using the substituted samples.

4.2 Maximum-Likelihood Substitution

Here, we do not sample but substitute the most likely values for u, that is $\int h(u,m)P(u|m)du \approx h(u^{ml},m)$ where $u^{ml} = \max_u \log P(u|m)$. For the prediction model in Equation 5 where y_{t-1} is missing and assuming Gaussians

$$y_{t-k}^{ml} = \min_{y_{t-k}} \sum_{l=t-k}^{t-1} (y_l - f(y_{l-1}, \ldots, y_{l-N}))^2$$

we simply find the substitution which minimizes the sum of the squared errors. In case of noisy measurements and Gaussian distributions

$$y^{ml} = \min_{y_1,\ldots,y_{t-1}} [-\log P(y_1, \ldots, y_N) + \frac{1}{2\sigma_\epsilon^2} \sum_{l=N+1}^{t-1} (y_l - f(y_{l-1}, \ldots, y_{l-N}))^2$$

$$+ \frac{1}{2\sigma_\delta^2} \sum_{l=1}^{t-1} (y_l - z_l)^2]$$

Where σ_ϵ^2 and σ_δ^2 are the variances of the two noise sources. Note, that this is a multidimensional optimization problem. In training, both the weights and the estimates of the missing or noisy realizations are adapted concurrently.

5 Experiments: K-step Prediction

We can use our preceding equations to predict K-steps into the future. In our framework, this is equivalent to the problem that $y_{t-1}, \ldots, y_{t-K+1}$ are missing and we want to predict y_t. In this case, Monte-Carlo sampling is very simple: generate a sample y_{t-K+1}^s of the first missing value using the distribution $P(y_{t-K+1}|y_{t-K}, \ldots, y_{t-K-N})$. Using that sample and the previous measurements, generate a sample of y_{t-K+2} following $P(y_{t-K+2}|y_{t-K+1}^s, \ldots, y_{t-K+1-N}))$ and so on until a sample of each unknown is produced. Repeat this procedure S times and approximate

$$E(y_t|M_{t-1}) \approx \frac{1}{S} \sum_{s=1}^{S} f(y_{t-1}^s, y_{t-1}^s, \ldots, y_{t-N}^s)$$

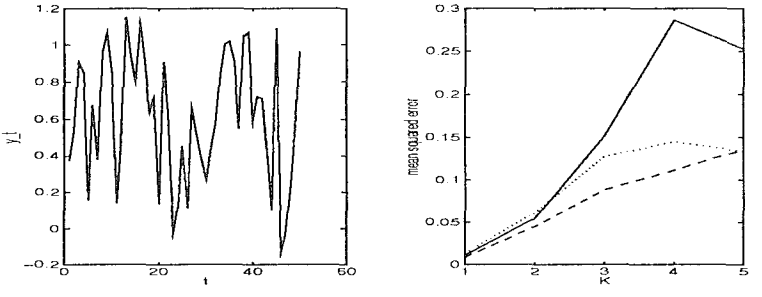


Figure 2: Left: Samples of the logistic map. Right: The mean squared error as a function of K in $K-$step prediction. The iterated solution (continuous) and the Monte-Carlo approximation with 3 (dotted) and 20 samples (dashed) are shown. Only for one-step prediction, the iterated model is optimal. Note, that by sampling we obtained an estimate of the prediction error of the iterated system (assuming a correct model).

where we have assumed that $K > N$. If $K \leq N$ substitute measured values for $k \geq K$. Note, that simply iterating the model K-times as it is usually done in $K-$step prediction is suboptimal in nonlinear time-series prediction if $K > 1$!

In our experiments, we wanted to find out to which degree our solutions are superior to simply iterating the time series in K-step prediction. We used the noisy logistic map $y_t = 4z_{t-1}(1 - z_{t-1}) + \epsilon_t$ where

$$z_t = \begin{cases} y_t & \text{if } 0 \leq y_t < 1 \\ y_t - 1 & \text{if } y_t \geq 1 \\ y_t + 1 & \text{if } y_t < 0 \end{cases}$$

where ϵ_t is uncorrelated Gaussian noise with a variance of $\sigma^2 = 0.01$. Figure 2 (left) shows the time series. Figure 2 (right) shows the mean squared error as a function of K. Shown are the iterated system (continuous line) and the solution following our sampling approach. As expected, for $K = 1$ the interated solution is optimal, but for $K > 1$, the Monte-Carlo approximation even with only few samples is far superior.

6 Extensions

6.1 Error Bars

Sampling provides much more information than just expected values. In all of the cases considered earlier — missing or noisy data, K-step prediction — we can also easily obtain error bars of the predicted value by calculating the variance (or the covariances) in the samples produced (Figure 2).

6.2 Recurrent Neural Networks for Stochastic Models

The approach can be applied to state space models of the form[2]

$$y_t = f(y_{t-1}) + \epsilon_t \quad z_t = y_t + \delta_t$$

where $y_t, \epsilon_t \in \Re^{D_y}$, $z_t, \delta_t \in \Re^{D_z}$. In the following discussion and for the rest of the paper we will assume independent Gaussian noise for all noise sources (here: for all components of ϵ_t and δ_t), although the results can be easily generalized to other distributions. We also assume that the components of ϵ_t have identical variances. Infinite variance in the components of δ_t indicates an unknown (hidden) variable, a variance of zero a certain measurement[3] and a finite variance a noisy measurement. $z = \{z_1, \ldots z_{t-1}\}$ is the set of measured values and $y = \{y_1, \ldots y_{t-1}\}$ are unknown. Let's assume that $f(y_{t-1}) \approx NN_w(y_{t-1})$ is approximated by a neural network. The gradient of the log-likelihood with respect to a weight w is

$$\frac{\partial L}{\partial w} \propto \sum_{l=2}^{t} \int \frac{\partial NN_w(y_{l-1})}{\partial w} \left(y_l - NN_w(y_{l-1})\right) P^M(y_l, y_{l-1}|z)\, dy_l dy_{l-1}. \quad (10)$$

Note, that $\frac{\partial NN_w(y_{l-1})}{\partial w}$ is a $D_w \times D_y$ dimensional matrix, where D_w are the number of weights in the network. In the Monte Carlo approximation, the main problem is the sampling from $P^M(y_l, y_{l-1}|z)$ which can be very time-consuming. We will discuss another learning algorithm for recurrent networks in the next section where the generation of samples is much easier.

6.3 Control of Stochastic Systems, Reinforcement Learning and Training of Recurrent Networks Revisited

Deterministic Control. Consider $y_t = f(y_{t-1}, u_{t-1}) + \epsilon_t$ where $u_t = NN_w(y_t)$ is a parameterized controller. The task is to minimize the expected cost up to a finite horizon T

$$E(cost) = \int \sum_{l=1}^{T} \gamma^{l-1} C(y_l) P(y_1, \ldots, y_T) dy_1, \ldots, dy_T$$

where $\gamma \leq 1$ is a discount factor and $P(y_1, \ldots, y_T) = P(y_1) \prod_{l=2}^{T} p(y_l|y_{l-1})$. To optimize the controller, we need the gradient of the expected cost with respect to w. We obtain[4]

$$\frac{\partial E(cost)}{\partial w} \propto \sum_{l=2}^{T} \int \gamma^{l-1} C(y_l) \times \quad (11)$$

[2] Introducing the more general assumption that $z_t = g(y_t) + \delta_t$ poses no additional difficulties.

[3] For the sampling procedure to work, a small error should always be assumed, such that the noise variance is always greater than zero.

[4] Recall that we assume Gaussian noise distributions.

$$[\sum_{m=2}^{l} \frac{\partial f(y_{m-1}, u_{m-1}))}{\partial u_{m-1}} \frac{\partial NN_w(y_{m-1})}{\partial w}(y_m - f(y_{m-1}, u_{m-1}))]$$

$$P(y_1, \ldots y_l)\, dy_1 \ldots dy_l$$

This solution can be approximated using stochastic sampling (see the following discussion). To avoid infinite control actions, it might be useful to introduce a cost which takes control actions into account or which adds a penalty for large weights in NN_w.

Stochastic Control. Now assume that the control action is stochastic $u_t = NN_w(y_t) + \delta_t$ and that we allow that the cost depends on the control action. Then,

$$\frac{\partial E(cost)}{\partial w} \propto \sum_{l=1}^{T} \int \gamma^{l-1} C(y_l, u_l)[\sum_{m=1}^{l} \frac{\partial NN_w(y_m)}{\partial w}(u_m - NN_w(y_m))] \qquad (12)$$

$$\times P(y_1, \ldots, y_l, u_1, \ldots, u_{l-1})\, dy_1 \ldots dy_l du_1 \ldots du_{l-1}.$$

Note, that we do not need a model of the process $f()$ any more! This is a result of the fact that we execute *stochastic* control. The system "tries" different actions and adapts the controller to favor actions which lead to low costs. We simply simulate the system (or collect data on the real process) and execute control actions. In the course of training we might want to reduce the noise variance on the control to eventually converge to deterministic controls. Let's assume that we generated S time series of the process by starting at $l = 1$ and iterating until T generating samples u_l^s and y_l^s. For each experiment s, we iterate for $l = 2, \ldots, T$ ($a_0^s = e_0^s = 0$)

$$a_l^s = \gamma a_{l-1}^s + \gamma^{l-1} \frac{\partial NN_w(y_l^s)}{\partial w}(u_l^s - NN_w(y_l^s))$$

and $e_l^s = e_{l-1}^s + C(y_l^s, u_l^s) a_{l-1}^s$. Then $\partial E(cost)/\partial w \approx 1/S \sum_{s=1}^{S} e_T^s$.

Recurrent Neural Networks. The previous equations also contain an algorithm for training recurrent neural networks. Assume $y_t = NN_w(y_{t-1}) + \epsilon_t$. Define $C(y_t) = ||b_t(y_t^d - y_t)||^2$, $\gamma = 1$. Here, y_t^d is a target at time t and b_t is a vector with $b_{ti} = 1$ if the i-th component of y_t is measured and zero otherwise (i. e. for the hidden variables). Then

$$a_l^s = a_{l-1}^s + \frac{\partial NN_w(y_{l-1}^s)}{\partial w}(y_l^s - NN_w(y_{l-1}^s))$$

and $e_l^s = e_{l-1}^s + ||b_l(y_t^d - y_t^s)||^2 a_l^s$. Finally, $\partial E(cost)/\partial w \approx \frac{1}{S} \sum_{s=1}^{S} e_T^s$.

On-line Adaptation. Consider stochastic control again. We let $T \to \infty$. We now assume that at every time-step, we start a new experiment s. Then let $a_l = \sum_{s=1}^{l} a_l^s$ and

$$a_l = \gamma a_{l-1} + [\sum_{m=1}^{l} \gamma^{l-m}] \frac{\partial NN_w(y_l^s)}{\partial w}(u_l^s - NN_w(y_l^s))$$

$$\approx \gamma a_l + \frac{1}{1-\gamma}\frac{\partial NN_w(y_l^s)}{\partial w}(u_l^s - NN_w(y_l^s)).$$

The last approximation is valid for large l. Then: $e_l = \rho e_{l-1} + C(y_l^s)a_{l-1}$ ($\rho = 1$). In practice, a small gradient descent learning step might be executed at every time-step $\Delta w(l) \propto e_l$ with an additional decay term on e_l ($\rho < 1$). Note that this is easily recognized as a variant of reinforcement learning. We can reduce the variance of the controller in the course of training and converge to a deterministic control law. The connection with reinforcement learning is even more obvious if we write for the right-hand side of Equation 12 (exchange the order of summations, $T \to \infty$)

$$\sum_{m=1}^{\infty} \int \frac{\partial NN_w(y_m)}{\partial w}(u_m - NN_w(y_m))\gamma^{m-1}(C(y_m, u_m) + \gamma V(y_{m+1}))$$

$$\times P(y_m, y_{m+1}, u_m)\, dy_m dy_{m+1} du_m.$$

$V(y_{m+1}) = \int C(y_{m+1}, u_{m+1}) du_{m+1} + \sum_{l=m+2}^{\infty} \gamma^{l-m-1} \int C(y_l, u_l) P(y_l, u_l) dy_l du_l$ is the expected cost if the current state is y_{m+1} (not including the factor γ^m).

7 Conclusions

We have shown how the problem of missing snd noisy data can be approached in a principled way in time-series prediction. In addition, we derived equations for training recurrent neural networks, for stochastic control and for reinforcement learning problems. The proposed approximations are based on stochastic simulations which, in general, are computationally expensive. Sampling is particularly simple if we can sample only forward in time as in K-step prediction and in the control laws and learning rules discussed in Section 6.3.

References

[1] Bernardo, J. M., Smith, A. F. M. (1994) *Bayesian Theory*. Wiley & Sons.

[2] Buntine, W. L. and Weigend, A. S. (1991). Bayesian Back-Propagation. *Complex systems*, Vol. 5, pp. 605-643.

[3] Ghahramani, Z. and Jordan, M. I. (1994). Supervised Learning from Incomplete Data via an EM approach. In: Cowan, J. D. *et al.*, eds., *Advances in Neural Information Processing Systems 6*. Morgan Kaufman.

[4] Tresp, V., Ahmad, S. and Neuneier, R. (1994). Training Neural Networks with Deficient Data. In: Cowan, J. D. *et al.*, eds., *Advances in Neural Information Processing Systems 6*. Morgan Kaufman.

Non-Linear Time Series Modeling with Self-Organization Feature Maps

Jose C. Principe, Ludong Wang

Computational NeuroEngineering Laboratory
University of Florida, Gainesville, FL32611
phone 904-392-2662/fax 904-392-0044
principe @synapse.ee.ufl.edu

1 Abstract

A locally linear approach based on Kohonen self-organizing feature mapping (SOFM) is proposed for the modeling of non-linear time series. This approach exploits the neighborhood preserving property of Kohonen feature maps. The key difference is that the local model fitting is performed directly over a matched neighborhood of the constructed SOFM neural field. The initial results show that this neural network scenario is an effective approach for local modeling of low dimensional non-linear processes.

2 INTRODUCTION

Farmer and Sidorowich have used a local approximation for the prediction of chaotic time series throughout state space. With that approach, the time series is first embedded in a state space using delay coordinates, and the underlying nonlinear mapping is inferred by a local approximation using only nearby states. This approach can be easily extended to higher order local polynomial approximations. The experiment by Farmer and Sidorowich shows that the linear model is an effective local approximation, while higher-order polynomials in higher dimensions are not significantly better than those obtained with first order. From the point of view of signal processing, the local linear approximation is derived as a state-dependent AR modeling, by Singer, et al. . This derivation shows that a single plane through the origin in state space is replaced with state-dependent approximation planes to account for the non-linear dynamic process. A good performance is attained at the cost of a large memory and inefficient computation for both the state space representation and local state search. This problem becomes worse with longer signal history, although the memory limitation can be alleviated with dynamical updating of data samples. The inefficient computation for nearby state search makes the implementation of this approach even harder. This is due to the fact that such search procedure is performed among the accumulated signal history without significant structuring.

0-7803-2739-X/95 $4.00 © 1995 IEEE

Our observation is that the computation in the above scenario can be facilitated in two ways. First, the signal representation can be streamlined with a vector quantization procedure. That is, the local model fitting is based on statistically averaged prototypes instead of the original state vector samples. Secondly, the nearby state search can be significantly simplified with all prototypes organized according to a certain metric such as pattern similarity. With a vector quantization procedure, the model estimation accuracy depends on the compactness of this prototype set. An optimal representation is found with respect to both the approximation performance and memory limitation. The beget advantages will make such local modelling more practical and feasible. Based on the above observation, we propose for local modeling a neural network vector quantization representation as an alternative to the state space representation.

3 METHODOLOGY

3.1 NonLinear Processes and Locally Linear Prediction

Different local prediction approaches were tested by Farmer and Sidorowich, and Casdagli for a variety of low-dimensional chaotic systems. In this work, we pursue the locally linear approach, but instead of using the data samples directly, the local fitting is done over the output of the SOFM neural field.

A given time series $x(t)$ can be embedded in a state space using Takens approach ,

$$\mathbf{x}(t) = [x(t), x(t-T), \ldots, x(t-(N-1)\tau)]$$

where τ is a delay time. For a given non-linear dynamical system of dimension D, a minimal requirement is $N \geq D$. The predictive relationship between the current state $\mathbf{x}(t)$ and the next value of the time series can then be expressed as [5]

$$x(t+T) = f_T(\mathbf{x}(t)) \tag{1}$$

The problem of predictive modeling is to find the mapping f_T: $\mathbf{R}^N$ to $\mathbf{R}^1$. A local predictor is constructed based on the nearby neighbors of $\mathbf{x}(t)$, that is, fitting a polynomials to the pairs $(\mathbf{x}(t_i), x(t_i+T))$ with $\mathbf{x}(t_i)$ being the nearest neighbors of $\mathbf{x}(t)$ for $t_i<t$. The original signal can also be viewed as an evolution of the state $\mathbf{x}(t)$ of a dynamical system in $\mathbf{R}^N$

$$\mathbf{x}(t+T) = f_T(\mathbf{x}(t)) \tag{2}$$

,

where f_T is the predictive mapping from $\mathbf{R}^N$ to $\mathbf{R}^N$. With simple matrix operations, f_T can be converted to f_T. Such predictive model concept is illustrated

in Fig. 1. The empty circles represent the current state $\mathbf{x}(t)$ and its evolution $\mathbf{x}(t+T)$, while the solid squares represent the nearby neighbors $\mathbf{x}(t_i)$ and future evolution $\mathbf{x}(t_i+T)$, where $t_i<t$. A simple model estimation is to fit a linear polynomial to pairs $(x(t_i), x(t_i+T))$

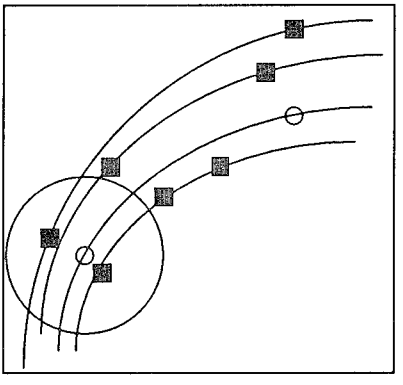


Fig. 1 Symbolic illustration of state prediction

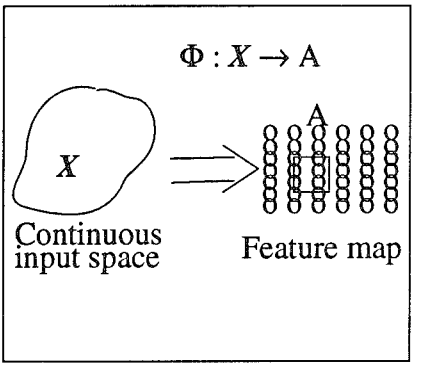


Fig. 2 Feature Mapping

From the point of view of signal processing, Singer, et al. [4] derived the locally linear prediction as an AR model generalization. In discrete time, a non-linear process can be described by a N^{th} order difference equations of the form

$$x(k+1) = f(\mathbf{x}(k)) + u(k) \tag{3}$$

where $\mathbf{x}(k) = [x(k), x(k-1), \ldots, x(k-N+1)]^T$, $f(\mathbf{x})$ represents the non-linear map from $\mathbf{R}^N$ to $\mathbf{R}^1$, and $u(k)$ is the white noise innovation term. Due to the statistical Markov structure of the nonlinear dynamics, we have

$$P(x(k+1) \mid x(i), 0<i<k) = P(x(k+1) \mid \mathbf{x}(k)) \tag{4}$$

Based on minimum mean square error criterion, the estimated value of $x(k+1)$ is

$$\hat{x}(k+1) = E[x(k+1) \mid \mathbf{x}(k)] = E[f(\mathbf{x}(k)) + u(k) \mid (\mathbf{x}(k))] = f(\mathbf{x}(k)) \tag{5}$$

Since the realization of the unknown dynamics $f(\mathbf{x})$ can be observed from

$$x(k+1) = f(\mathbf{x}(k)) + u(k) \tag{6}$$

that is, the signal history composes the map from state space of dimension N to a scalar space, the solution to the estimation of $x(k+1)$ can be solved by

interpolating $f(\mathbf{x})$ from noisy signal samples. Among several methods, as shown by Singer et al., the local modelling is superior and simpler under the condition that the given dynamics is locally smooth and a long enough signal history is available. Therefore the local linear prediction model fitting is implemented as follows. For the current state $\mathbf{x}(k)$, a set of pairs $(\mathbf{x}(i), x(i+1))$ is selected according to the similarity between $\mathbf{x}(k)$ and $\mathbf{x}(i)$ where $\mathbf{x}(i)$ is one of selected close state space neighbors and $x(i+1)$ its future value. Thus, the local model approximation becomes an interpolation problem which can be solved with polynomial fitting.

Following the approach by Singer et al, under the condition that $f(\mathbf{x})$ is smooth enough in the vicinity of $\mathbf{x}(k)$, $f(\mathbf{x})$ can be approximated by the first few terms of its multidimensional Taylor series expansion,

$$\tilde{f}(\mathbf{x}) = f(\mathbf{x}(k)) + \nabla F^{T}(\mathbf{x}(k))(\mathbf{x}-\mathbf{x}(k)) + \ldots \approx b + \mathbf{a}^{T}\mathbf{x} = \mathbf{a}^{T}\mathbf{x} + b \tag{7}$$

which is the local linear predictor. The vector and scalar quantities of $\mathbf{a}$ and b are estimated from the selected pairs $(\mathbf{x}(i), x(i+1))$ in the least square sense. To secure a stable solution, more than N pairs must be selected.

In general, the above local model fitting is composed of two steps: a set of nearby state searches over the signal history and model parameters fitting. For a given signal, this procedure results in a set of local model parameters which, when pieced together, provide a global modeling of the dynamics in state space. Since the state search is performed over the whole signal history a lot of redundant computation results which in turn hinders effective implementation of this approach.

3.2 Localized Signal Representation with SOFM modeling

Instead of direct sample collection from the signal history, we propose to alleviate these problems by the use of a Kohonen self-organizing feature map neural network [7]. The SOFM has very interesting properties for time series modelling. Let Φ, X, A denote the SOFM mapping, input sample space and the discrete output space respectively. When the network converges to its final stable state following a successful learning process, it displays four major remarkable properties:

1. The SOFM map Φ is a good approximation to the input space X. This property is important since it provide a compact representation of the given input space.

2. The feature map Φ naturally forms a topologically ordered output space such that the spatial location of a neuron in the lattice corresponds to a particular domain in input space. The advantage of this feature is that it can simplify local modeling of the input signal X embedded in the A space.

3. The feature map Φ embodies a statistical law. In other words, the input with more frequent occurrence occupies a larger output domain of the output space A. This property helps to make the SOFM an optimum codebook of the given input space.

4. A space dimension reduction is attained via the feature map Φ. That is, the continuous input space is mapped to a discrete output space with lower dimension. This property makes the simple architecture of codebook representation feasible.

The straightforward way to take advantage of the above properties for time series modelling is to create a SOFM from the input signal. Since such feature map Φ provides a faithful topologically organized output of the input vectors $\mathbf{x} \in X$, the local model fitting can then be performed over the compact codebook domain A.

The proposed non-linear modelling scenario follows three steps: *a.* Reconstruction of the state space from the input signal; *b.* Embedding the state space in the neural field; *c.* Estimation of the locally linear predictors.

a) *Reconstruction of the state space from the training signal.* Following the approach by Takens, a sequence of N+1 dimensional state vectors $[\mathbf{x}(n)^{\mathrm{T}}, x(n+\tau)]^{\mathrm{T}}$ is created from the given training time series, where $\mathbf{x}(n) = [x(n-(N-1)\tau), x(n-(N-2)\tau), \ldots, x(n)]^{T}$ and τ is the appropriate time delay where $N \geq D$ and D the dimension of the underlying dynamical process.

b) *Embedding the state space in the neural field.* This step is accomplished via the Kohonen learning process. With each vector-scalar pair $[\mathbf{x}(n), x(n+l)]$ presented as the input to the network, the learning process of Kohonen feature mapping algorithm adaptively discretizes the continuous input space $X \subset \mathbf{R}^{N+1}$ into a set of K disjoint cells A to construct the mapping Φ: $X \rightarrow A$. This process continues until the learning rate decreases close to zero and the neighborhood function covers about one output unit. After learning, a neural field representation A of the input space X via the constructed mapping relationship Φ is formed in terms of a set of disjoint units topologically organized in the output space (Figure 2).

c) *Estimation of the locally linear predictors.* For each element $u_i \in A$, its local linear predictor in terms of $[\mathbf{a}_i^{\mathrm{T}}, b_i]$ is estimated based on $\alpha_i \subset A$, which is a set of L elements in the neighborhood of u_i including u_i itself. See Figure 2. Each element u_i, has a corresponding weight vector $\left[\mathbf{w}_i^{T}, w_{i(N+1)}\right]^{T} \in \mathbf{R}^{N+1}$, where $\mathbf{w}_i^{T} = [w_{i(1)}, w_{i(2)}, \ldots, w_{i(N)}]$. The local prediction model $[\mathbf{a}_i^{\mathrm{T}}, b_i]$ is fitted in the least-square sense to the set of weights in α_i, i.e.

$$w_{j(N+1)} = b + a^T w_j \tag{8}$$

To ensure a stable solution of the above equations, α_i must have more than $N+1$ elements. Thereafter for each output unit $u_i \subset A$, there corresponds a unique linearly local model function $\tilde{f}(\circ)$ in terms of the vector-scalar parameter pair $[a_i, b_i]$.

The global dynamics of the given process can be described by the set of all the constructed local models pieced together. For an input state vector x(n)=[x(n-N+1), $x(n-N+2),..., x(n)]^T$, the matched prototype element $u_{i^\circ} \in A$ is found based on the SOFM competition among all elements in A. The predicted value x(n+1) is obtained by evaluating $\tilde{f}(\circ)$ at x(n)=[$x(n-N+1), x(n-N+2),..., x(n)]^T$

$$x(n+1) = \tilde{f}(a(u_{i^\circ}), b(u_{i^\circ}), x(n)) = b(u_{i^\circ}) + a^T(u_{i^\circ})\, x(n) \tag{9}$$

In a similar manner, a K-step prediction x(n+K) based on x(n) can also be obtained by iterative prediction, i.e. feeding the output back to the input,

$$x(n+K) = \tilde{f}_K(\tilde{f}_{K-1}(\ldots \tilde{f}_1(a(u_{i^\circ}), b(u_{i^\circ}), x(n)))) \tag{10}$$

where $\tilde{f}_j = \tilde{f}(\circ)$ is the prediction function at step j. That is, the first prediction generates a new state, which is used to find the new local model function. Evaluation of the new local model function at the new state produces in turn a new prediction until the final K-step prediction. Compared with the direct prediction, this recursive prediction has the advantage of higher accuracy [2], [12].

4 IMPLEMENTATION AND EXPERIMENTAL RESULTS

A non-linear time series from the Mackey-Glass system is modeled with the proposed scenario. A total of 2500 neurons, arranged on a 50 X 50 square lattice, constitute the SOFM output space. The dimension of the weight vectors $w_i(n)$ was chosen as 8, so the dimension of the state input during the training process is 9 (N+1). The learning rate and evolution of neighborhood function in eq. (14) and (15) were used with a_η=1, b_η=10^{-3}, a_σ=1/30, b_σ=.6x10^{-4}. A 10,000 samples Mackey-Glass time series is generated with d=30 and f_s=1/6 Hz. The Kohonen SOFM network is trained with this segment of time series for five epochs (50,000 samples). After training the weight vectors are frozen for local model estimation. A typical post-training output trajectory corresponding to 400 consecutive input samples is as shown in Figure 3.

As shown in section 2, to ensure a stable solution in the least square sense, the subset α_i must contain at least N+1 neighbors for stable model estimation. We take all 21 neurons surrounding the neuron u_i in the output space as its neighborhood subset $\alpha_i \subset A$ to estimate the corresponding local linear prediction function $\tilde{f}_i = \tilde{f}(\circ)$. Another different 5,000 sample Mackey-Glass time series is taken to test the prediction performance of the estimated local prediction model set. The testing is performed with multi-step prediction ranging from 1 up to 20 samples ahead. Iterative prediction (i.e. use the predicted values as new inputs) is applied for multiple-step prediction. The mean squared error normalized by the variance of the original signal is shown in Fig. 4, and it starts at .06 and increases to .4 for 30 step ahead prediction.

If the averaged Euclidean distance between weight vectors of two neighboring neurons is taken as the resolution of the neural field A, it is obvious that the larger its dimension the finer the resolution, which in turn provides more accurate local model estimation. With this notion, three SOFM networks with different lattice dimension (50 X 50, 60 X 60, 70 X 70) are compared in terms of the MSE for 20 step prediction steps and the result is consistent with the above observation as shown in Fig. 5. The MSE error decreases from 0.22 to 0.18.

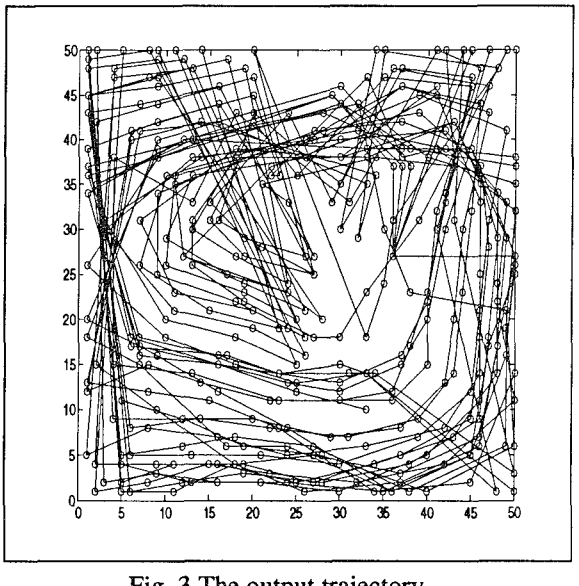


Fig. 3 The output trajectory

Finally, Figure 6 shows an autonomous (i.e. the predictor is seeded and then the output is fed back to its input) 500 point segment of the signal generated by the local models. This signal clearly shows that the dynamics of the system that produced the time series have been captured.

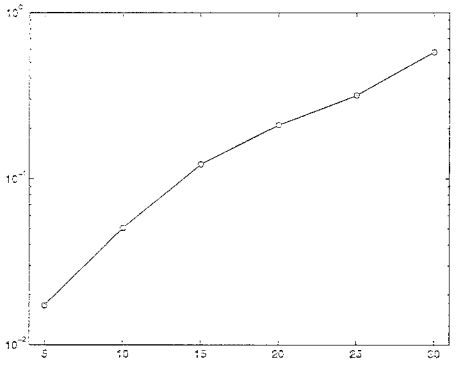


Fig. 4 MSE vs. Prediction Interval

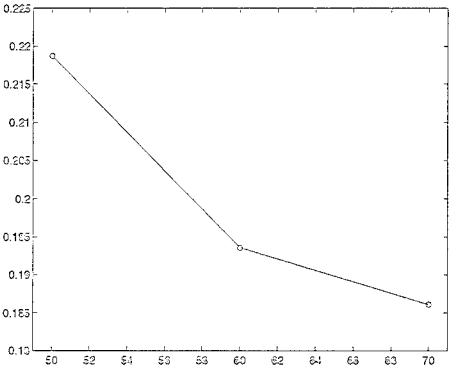


Fig. 5 MSE vs. Network Dimension

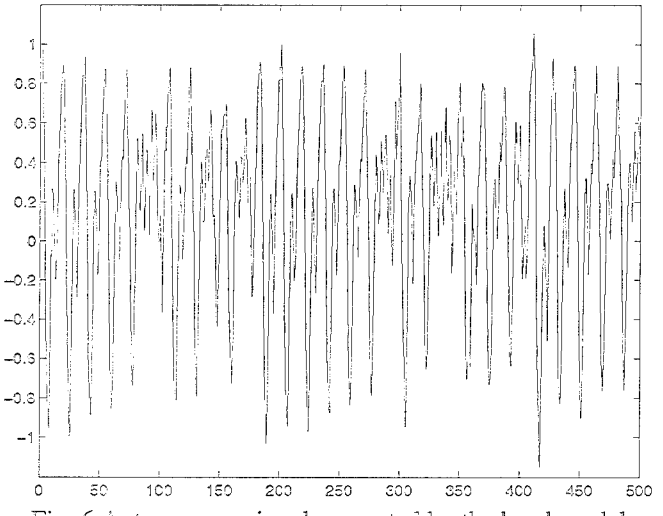


Fig. 6 Autonomous signal generated by the local models

5 CONCLUSIONS

In this work it is shown that the non-linear dynamics can be modeled by a set of local models, each of which is fitted to the matched subset of the quantized prototype state space. This quantized prototype space is built up with the Kohonen self-organizing feature map neural network. The SOFM has the advantages of compact state space representation of the original time series and simple nearby state selection. The latter feature is ensured by the neighborhood preserving property which is the direct consequence of the learning law. With this spatial advantage, the local model fitting can be performed directly over the matched response region in the neural field. Since the local information is extracted directly

from the neural field, this scenario represents a further in-depth exploration of the Kohonen feature mapping.

The Kohonen feature map has been used by Walter, et al. for similar purposes. However, the topologically ordering relationship intrinsic in the construction of the neural field was not explored in their approach. Instead the desired local models were adaptively constructed during the network training process. The topologically organized weight vectors only served for state searching.

The experimental results demonstrate that this Kohonen SOFM scenario is feasible as an effective approach for non-linear dynamical modeling. We are presently comparing this approach with others using dynamic neural networks (i.e. multilayer perceptrons extended with short term memory mechanisms) [12]. One advantage of the SOFM is the creation of states that can be explored for the prediction of time varying nonlinear signals.

Acknowledgments:

This work was partially supported by NSF grant ECS #9208789 and ARPA grant #N00014-94-1-0858.

6 REFERENCE

1. A.V. Oppenheim, et al., "Signal processing in the context of chaotic signals," IEEE ICASSP, 4, pp. 117, 1992.

2. H.D.I. Abarbanel, "The analysis of observed chaotic data in physical system," Reviews of Modern Physics, vol. 65, No. 4, pp. 1331, 1993.

3. J.D. Farmer, et al., "Predicting chaotic time series," Physical Review Letters, vol. 59, No. 8, pp. 845, 1987.

4. A.C. Singer, et al., "Codebook prediction: a nonlinear signal modeling paradigm," IEEE ICASSP, 5, pp. 325, 1992.

5. M. Casdagli, "Nonlinear prediction of chaotic time series," Physics D 35, pp. 335, 1989.

6. F. Takens, in Dynamical Systems and Turbulence, ed. by D.A. Rand and L.-S. Young, Springer-Verlag, Berlin, 1981.

7. T. Kohonen, Self-Organization and Associative Memory, 3rd ed., New York, Springer-Verlag, 1988.

8. S. Haykin, Neural Networks, A Comprehensive Foundation, Macmillan College Publishing, 1994.

9. J. Hertz, et al., Introduction to the Theory of Neural Computation, Adison-Wesley Publishing, 1991.

10. S.C. Ahalt, et al., "Competitive learning algorithms for vector quantization," Neural Networks 3, pp. 277, 1990.

11. J. Walter, et al. "Non-linear prediction with self-organization maps," IJCNN, I-589, 1990.

12. Principe J. and Kuo J-M., "Dynamic modelling of chaotic time series with neural networks", Proc. of Neural Infor. Proc. Sys, NIPS 7, in press, 1995.

NEURAL NETWORKS FOR FUNCTION APPROXIMATION

H. N. Mhaskar and L. Khachikyan
Department of Mathematics, California State University
Los Angeles, California, 90032, U.S.A.
email : hmhaska@calstatela.edu, lkhachi@calstatela.edu

ABSTRACT. We describe certain recent results of the first author concerning the approximation capabilities of neural networks with one hidden layer. In particular, these results demonstrate the construction of neural networks evaluating a squashing function or a radial basis function for *optimal approximation* of the Sobolev spaces. We also report on our joint work, where some of the first author's earlier ideas are applied to construct general purpose networks for the prediction of time series, when the number of independent variables is known in advance, such as the Mackey- Glass series or the flour data.

1. INTRODUCTION

One of the major applications of neural networks is to approximate a function of several variables. In fact, it is well known that any neural network training can be thought of as function approximation. A typical example is the prediction of time series, where it is desired to predict the observation x_t at time t as a function of the s previous observations $x_{t-1}, \cdots, x_{t-s}$, where s is a fixed positive integer.

There are two major problems which arise in this theory. One is to determine the number of neurons necessary to achieve the approximation of the target function within a given margin of tolerance. The other is to develop algorithms to actually construct the approximating networks.

Although the target function is usually unknown, it is customary to assume that it belongs to some known class of functions. A common assumption is that the function has a certain number, r, of continuous derivatives on the domain where the approximation is desired. The *complexity problem* is the problem of determining the number of neurons required to approximate any such function in terms of the desired accuracy, the number of independent variables on which the function depends, and the size of the derivatives as measured by a suitable norm. Equivalently, the problem is to determine how much accuracy one can achieve in the approximation of *any* function in this class with a given number of neurons. Section 2 of this paper describes some of the recent work of the first author in this direction.

In designing algorithms for the construction of networks, one may or may not be able to sample the function at prescribed points. The proofs of the results described in Section 2 also give a training method in the case the function may be sampled at prescribed points. In Section 3, we use some of the earlier ideas of the first author [10] for "universal" approximation of

0-7803-2739-X/95 $4.00 © 1995 IEEE

functions, where such a sampling is not possible. We do need to assume that the number of variables, on which the function depends, is known in advance.

Our research was supported, in part, by grants from AFOSR and NSF. We are also grateful to F. Girosi, j. Larsen, J. A. Sørensen and T. Poggio for their kind encouragement.

2. THEORETICAL RESULTS

In [8, 18], Girosi, Poggio and Jones have introduced the notion of a *generalized translation network* (*generalized regularization network* in their terminology). Let $1 \leq d \leq s$, $n \geq 1$ be integers, $\phi : \mathbb{R}^d \to \mathbb{R}$. A generalized translation network with n *neurons* (principal elements) evaluates a function of the form $\sum_{k=1}^{n} a_k \phi(A_k(\cdot) + \mathrm{b}_k)$ where the A_k's are $d \times s$ real matrices, $\mathrm{b}_k \in \mathbb{R}^d$ and $a_k \in \mathbb{R}$ $(1 \leq k \leq n)$. In the case $d = 1$, we have the usual neural networks with ϕ as the activation function. In the case $d = s$, we recover the traditional radial basis function (RBF) networks, where, in the most traditional setting, the matrices A_k are required to be all equal to the identity matrix. Girosi, Poggio and Jones have demonstrated in [8, 18] how the generalized translation networks arise naturally in applications such as image processing and graphics as solutions of certain extremal problems.

Motivated (in part) by this work, Mhaskar and Micchelli [14] carried out an in-depth investigation of the approximation capabilities of the generalized translation networks. Under very general conditions on ϕ, they have constructed networks that approximate an arbitrary function in the L^p or uniform norm, and illustrated how the smoothness and growth of ϕ affect the degree of approximation of the target function. The networks constructed in [14] are also capable of providing simultaneous approximation of the target function and its derivatives, under minimal conditions on ϕ. The main thrust of [14] is to study what properties of ϕ have what effect on the degree of approximation; in particular, to gain an insight on how to choose an activation function, rather than to obtain the best estimates. A preliminary announcement of some of the results in [14] was made in [15].

In order to discuss the complexity problem further, we need to introduce some notation. For the sake of simplicity of exposition, we restrict ourselves to uniform approximation on $[0, 1]^s$. Thus, for $f : [0, 1]^s \to \mathbb{R}$ we write

$$\|f\| := \sup_{\mathrm{x} \in [0,1]^s} |f(\mathrm{x})|. \tag{2.1}$$

The class of all the output functions of a generalized translation network with n neurons, each evaluating the activation function ϕ and receiving s inputs, will be denoted by $\Pi_{\phi;n,s}$. We measure the *degree of approximation* of f by the expression

$$E_{\phi;n}(f) := \inf\{\|f - P\| \ : \ P \in \Pi_{\phi;n,s}\}. \tag{2.2}$$

For integer $r \geq 1$, the class of all functions $f : [0, 1]^s \to \mathbb{R}$ having r continuous derivatives on $[0, 1]^s$ will be denoted by $W_{r,s}$. For a multi-integer

$\mathbf{k} = (k_1, \ldots, k_s) \in \mathbf{Z}^s$, the notation $0 \le \mathbf{k} \le r$ means that $0 \le k_j \le r$ for $1 \le j \le s$ and we write $|\mathbf{k}| = \sum_{j=1}^{s} |k_j|$. If $f \in W_{r,s}$, its Sobolev norm is defined by

$$||f||_{W_{r,s}} := \sum_{0 \le \mathbf{k} \le r} ||D^{\mathbf{k}} f|| \tag{2.3}$$

where the partial derivatives $D^{\mathbf{k}} f$ are defined by

$$D^{\mathbf{k}} f := \frac{\partial^{|\mathbf{k}|} f}{\partial x_1^{k_1} \cdots \partial x_s^{k_s}}.$$

Since the target function is usually unknown, the quantity of interest in this theory is

$$E_{\phi;n,r,s} := \sup\{E_{\phi;n}(f) \ : \ ||f||_{W_{r,s}} \le 1\}. \tag{2.4}$$

We observe that any function in $W_{r,s}$ can be normalized so that $||f||_{W_{r,s}} \le 1$. Hence, $E_{\phi;n,r,s}$ measures the "worst case" degree of approximation by generalized translation networks with n neurons under the assumption that $f \in W_{r,s}$ and is properly normalized.

There are general theorems in approximation theory due to DeVore, Howard and Micchelli [6] which indicate that $E_{\phi;n,r,s}$ must be at least of the order $n^{-s/r}$. This order was achieved in [10] using networks with multiple hidden layers. However, it is shown in [4] that these methods cannot work for networks with one hidden layer. It was conjectured in [11] that $E_{\phi;n,r,s}$ cannot be $\mathcal{O}(n^{-s/r})$ at least in the usual neural network setting, where $d = 1$ and ϕ is a sigmoidal activation function.

The following theorem due to the first author [12] disproves this conjecture, and describes certain conditions under which the optimal order of approximation can be realized, in fact, for generalized translation networks with a single hidden layer.

THEOREM 2.1. *Let $1 \le d \le s$, $r \ge 1$, $n \ge 1$ be integers, $\phi : \mathbf{R}^d \to \mathbf{R}$ be infinitely many times continuously differentiable in some open sphere in $\mathbf{R}^d$. We further assume that there exists $\mathbf{b}$ in this sphere such that*

$$D^{\mathbf{k}} \phi(\mathbf{b}) \ne 0, \qquad \mathbf{k} \in \mathbf{Z}^d, \ \mathbf{k} \ge 0. \tag{2.5}$$

Then there exist $d \times s$ matrices $\{A_j\}_{j=1}^n$ and a positive constant c depending at most on ϕ, r and s, but independent of n, with the following property. For any $f \in W_{r,s}$, there exist coefficients $a_j(f)$ such that

$$||f - \sum_{j=1}^{n} a_j(f) \phi(A_j(\cdot) + \mathbf{b})|| \le c n^{-r/s} ||f||_{W_{r,s}}. \tag{2.6}$$

The functionals a_j are continuous linear functionals on $W_{r,s}$.

Some of the important examples where (2.5) is satisfied are the following, where for $\mathbf{x} \in \mathbb{R}^d$, we write $|\mathbf{x}| := (\sum_{j=1}^{d} x_j^2)^{1/2}$: the squashing function, where $d = 1$, $\phi(x) := (1 + e^{-x})^{-1}$, the generalized multiquadrics, where $d \geq 1$ and $\phi(\mathbf{x}) := (1 + |\mathbf{x}|^2)^{\alpha}$, $(\alpha \notin \mathbb{Z})$, thin plate splines, where $d \geq 1$, and with $q \in \mathbb{Z}$, $q > d/2$,

$$\phi(\mathbf{x}) := \begin{cases} |\mathbf{x}|^{2q-d} \log |\mathbf{x}|, & d \text{ even}, \\ |\mathbf{x}|^{2q-d}, & d \text{ odd}, \end{cases}$$

and the Gaussian function, where $d \geq 1$, $\phi(\mathbf{x}) := \exp(-|\mathbf{x}|^2)$.

A remarkable feature of Theorem 2.1 is that the matrices A_j and the threshold $\mathbf{b}$ are determined independent of the target function. The determination of these quantities is typically a major problem in most network training algorithms such as backpropagation. In fact, the proof of Theorem 2.1 does not depend upon any optimization at all, so that none of the usual problems in network training, such as local minima, arise. The proof also gives an explicit formula for the functionals $a_j(f)$, thus reducing the "training" to a simple evaluation of linear functionals.

For functions which are analytic in a (complex) ellipsoid containing $[0, 1]^s$, the method of the proof gives *dimension independent bounds*, allowing a geometric rate of approximation. The analyticity condition is substantially stronger than the more well known conditions of Barron [1], but the geometric rate of convergence is also substantially stronger than that obtained in [1]. Moreover, it is a local condition.

Dimension independent bounds under a different set of local conditions are studied in [13]. It is worthwhile to remark in this connection that the paper [14] studies the construction of networks which provide an "optimal recovery" of a class of functions, based on the number of observations on the target function, rather than the number of neurons.

3. AN ALGORITHM FOR ADAPTIVE APPROXIMATION

The networks described in the proofs of the results in Section 2 require that one should be able to sample the target function at prescribed points, without noise. In this section, we describe an algorithm which does not require these assumptions. The basic idea is the fact (cf. [10]) that it is possible to approximate the characteristic function of an s-dimensional cube arbitrarily closely using a neural network with a fixed (dependent only on s) number of neurons, each evaluating a bounded sigmoidal activation function, arranged in two hidden layers. In this section, we describe an algorithm which will provide an adaptive approximation to a target function f on $[0, 1]^s$ by choosing a suitable partition of the cube.

The algorithm in Fig. 1 is a very simple adaptation of some of the ideas in R. DeVore's lecture [5]. The starting point of the algorithm is the *training data*, $\mathcal{T}$, which is organized as an array of (s+1)-tuples $(\mathbf{x}, y)$ where $\mathbf{x} \in [0, 1]^s$

and y is the value of the target function f. At the culmination of this algorithm, the cube $[0,1]^s$ is partitioned into subcubes and the approximation of the target function is the function g that takes a constant value on each of these subcubes. The accuracy of this approximation is measured by a suitable functional $E(T)$ of the training data, such as the root-mean squared error. During the algorithm, the subcubes are organized in a tree structure. The root of the tree is $[0,1]^s$, and the children of a node are the 2^s partitioning subcubes of that node obtained by halving each side of the node. For each node Q, $\mathcal{S}_Q$ denotes the set of observations $(\mathbf{x}, y)$ such that $\mathbf{x} \in Q$; $\hat{y}_Q$ is the value of g on Q.

```
1. Let 𝒩 := {[0,1]^s}, 𝒮 := {(x_1, y_1)}.
2. Q := [0,1]^s, ŷ_Q := y_1, E(𝒮) := 0.
3. while there is more training data, do begin
4.       while ((E(𝒮) ≤ ε) and (|𝒮_Q| ≤ P)) do begin
5.             Read next (x, y) and add it to the set 𝒮.
6.             Find the smallest subcube Q ∈ 𝒩 such that x ∈ Q.
7.             Set z_Q := (|𝒮_Q| ŷ_Q + y)/(|𝒮_Q| + 1).
8.             Calculate E(𝒮) using g(x) := z_Q for x ∈ Q.
9.       end {Straightforward processing, go to line 4.}
10.      while ((E(𝒮) > ε) or (|𝒮_Q| < P)) do begin
11.            Split Q into 2^s equal subcubes.
12.            For each subcube C of Q do
13.                If 𝒮_C is empty, set ŷ_C := ŷ_Q.
14.                Otherwise, recalculate ŷ_C.
15.                Add C to 𝒩.
16.            end {subcube processing}
17.            Rename the subcube containing x as Q.
18.            Calculate E(𝒮).
19.      end {Bringing the error within margin, go to 10}.
20.end { Outer while, go to 3 for more data}
```

Fig. 1.
Partitioning algorithm

To start with, the tree consists of the root alone. Given any tree, we keep on accumulating the data such that the $\mathbf{x}$ values fall on a leaf of this tree. The value of g on each leaf is simply the average of the y values corresponding to the $\mathbf{x}$ values falling on that leaf. This process continues until the error functional for the partition becomes unacceptably large, or until a single node accumulates too many $\mathbf{x}$ values; thus indicating that a more refined analysis is required. At this point, the node is split into its children and the data is

redistributed. If there are no $\mathbf{x}$ values on any of the new leaves, the value of g on these leaves is the same as that on their parent, before the parent was split. This splitting continues until the error functional for the training data encountered so far is satisfactory. The process stops when the training data is finished and the error functional $E(T)$ is satisfactory.

If the target function is continuous, then this process is guaranteed to stop after a sufficient amount of training data is analyzed. Thus, the algorithm provides *universal approximation.* It can be easily modified to incorporate spline functions of order higher than 0. This is expected to significantly enhance the performance of the algorithm, albeit at some computational cost.

There are many algorithms in the literature for adaptive approximation, which are perhaps better than our algorithm in many ways. The main interesting feature of our algorithm, in addition to its simplicity, is that it does not seek to solve *any minimization problem.* For example, at each stage of the recursive partitioning algorithm, (CART for regression) [7, p. 11], [9, p. 231], the domain and the sampling data are split so as to minimize a variance. In the MARS algorithm, [7, p. 17], the nodes and the coefficients of the approximating spline are chosen at each stage to minimize a "lack-of-fit" functional. In the k-NN algorithm, one has to search in the sample space to find the k argument values which are closest, i.e., minimize a distance functional, to the given argument value. Our algorithm does not require any such minimization. The resulting error is therefore not optimal, but is still under control. In fact, the algorithm guarantees that the error does not exceed a preset tolerance. We have not done any pruning as suggested in [2, 7]. The number of subcubes, and hence the number of neurons is typically substantially large. In the case of the algorithm of Fig. 1, pruning could have been accomplished easily. One just combines all the neighboring subcubes having the same predicted value into one subregion. However, since our algorithm is so simple, and the combined subcubes might not give a cube, it was not thought worthwhile to prune the tree. Finally, we observe that the resulting neural network provides localized approximation in the sense of [4].

Although our main interest was only to check how the ideas of [10] can be implemented in practice, rather than to solve any particular problem, we tried the algorithm to solve two time series prediction problems. The Mackey-Glass time series, studied recently by Plutowski, et. al. [17] and Platt [16], is a four variable problem. We set the training RMS at 0.001, trained on 400 samples and predicted the next 100 samples. The net RMS error (on the test data) was 0.0026 (Fig. 2). This required 279 cubes at the end of the training. The test phase required an addition of 40 cubes, which was done very easily. When we trained on 500 examples, and predicted the next 500 data, the net RMS (on the test data) was 0.0022 (Fig. 3). At the end of the training, there were 300 subcubes, the test phase required 131 new subcubes. We also analyzed the flour data studied recently by Chakraborty et. al. [3]. As in [3], we trained on 90 samples and predicted the next 10. The training RMS was preset at 0.25. In separate modelling (a two variable

problem), we obtained an RMS of 0.0028, 0.0028 and 0.0014 for the three cities (Fig. 4 shows the graph for Buffalo). In combined modelling (a six variable problem), the numbers were respectively 0.0043, 0.0038 and 0.0040. This is a substantial improvement on the results quoted in [3]; indicating that the target function must be very smooth.

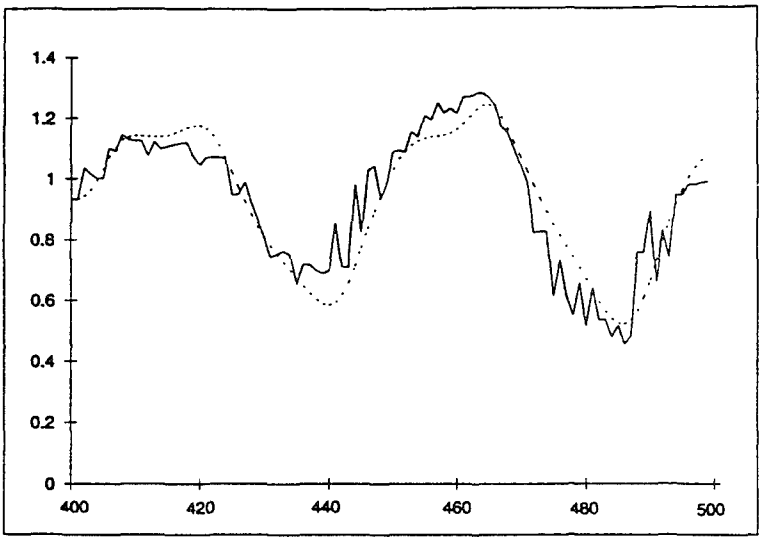


FIGURE 2. Mackey-Glass Series; – predicted, ... actual

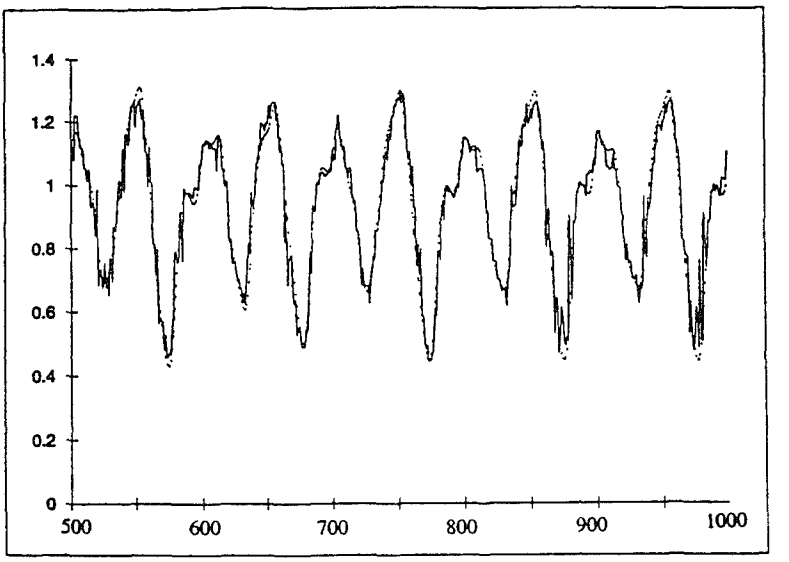


FIGURE 3. Mackey-Glass Series; – predicted, ... actual

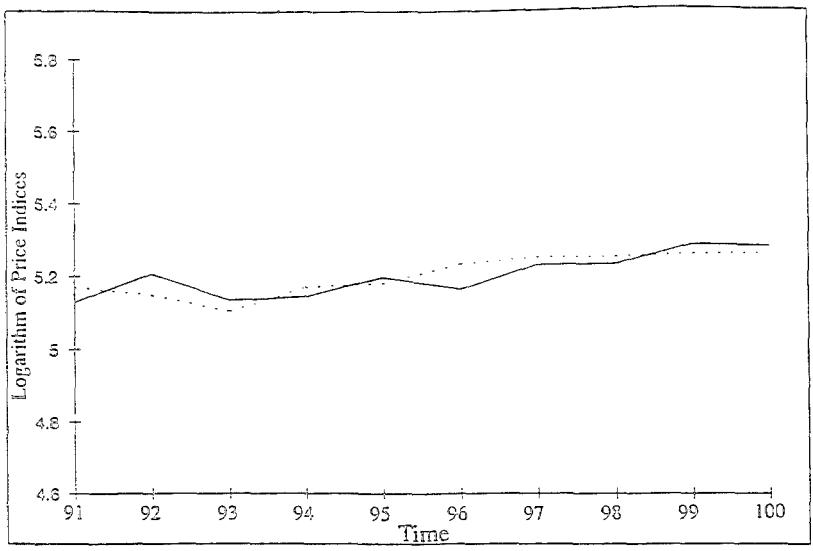


FIGURE 4. Flour Price Data; – predicted, ... actual: (Buffalo)

REFERENCES

1. A. R. Barron, *Universal approximation bounds for superposition of a sigmoidal function,* IEEE Trans. Information Theory, 39 (1993), 930-945.
2. L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and Regression Trees", Wadsworth and Brooks, Pacific Grove, California, 1984.
3. K. Chakraborty, K. Mehrotra, C. K. Mohan, and S. Ranka, *Forecasting the behavior of multivariate time series using neural networks,* Neural Networks, 5 (1992), 961-970.
4. C. K. Chui, X. Li and H. N. Mhaskar, *Neural networks for localized approximation,* Mathematics of Computation, 63 (1994), 607-623.
5. R. DeVore, *Degree of nonlinear approximation,* in "Approximation Theory, VI", (C. K. Chui, L. L. Schumaker and J. D. Ward Eds.), Academic Press, 1989, pp. 175-201.
6. R. DeVore, R. Howard and C. A. Micchelli, *Optimal nonlinear approximation,* Manuscripta Mathematica, 63 (1989), 469-478.
7. J. H. Friedman, *Multivariate adaptive regression splines,* Annals of Statistics, 19 (1991), 1-141.
8. F. Girosi, M. Jones and T. Poggio, *Regularization theory and neural networks architectures,* Neural Computation, 7 (1995), 219-269.

9. W. HÄRDLE, "Applied Nonparametric Regression", Cambridge University Press, Cambridge, 1990.
10. H. N. MHASKAR, *Approximation properties of a multilayered feedforward artificial neural network,* Advances in Computational Mathematics, **1** (1993), 61-80.
11. H. N. MHASKAR, *Approximation of real functions using neural networks,* in Proc. of Int. Conf. on Computational Mathematics, New Delhi, India, 1993, (H. P. Dikshit and C. A. Micchelli Eds.), World Scientific Press, 1994.
12. H. N. MHASKAR, *Neural networks for optimal approximation of smooth and analytic functions,* to appear in Neural Computation.
13. H. N. MHASKAR AND C. A. MICCHELLI, *Dimension independent bounds on the degree of approximation by neural networks,* IBM Journal of Research and Development, **38** (1994), 277-284.
14. H. N. MHASKAR AND C. A. MICCHELLI, *Degree of approximation by neural and translation networks with a single hidden layer,* to appear in Advances in Applied Mathematics.
15. H. N. MHASKAR AND C. A. MICCHELLI, *How to choose an activation function,* in NIPS*6 (J. D. Cowan, G. Tesauro, J. Alspector Eds.), Morgan Kaufmann, 1994, 319-326.
16. J. PLATT, *A resource-allocating network for function interpolation,* Neural Computation, **3** (1991), 213-225.
17. M. PLUTOWSKI, G. COTTRELL AND H. WHITE, *Learning Mackey-Glass from 25 examples, plus or minus 2,* in NIPS*6 (J. D. Cowan, G. Tesauro, J. Alspector Eds.), Morgan Kaufmann, 1994, 1135- 1142.
18. T. POGGIO AND F. GIROSI, *Networks for approximation and learning,* in Proceedings of the IEEE, **78**(9), (1990).
19. G. C. TIAO AND R. S. TSAY, *Model specification in multivariate time series,* J. R. Statist. Soc., **51** (1989), 157-213.

EMPIRICAL GENERALIZATION ASSESSMENT OF NEURAL NETWORK MODELS

Jan Larsen and Lars Kai Hansen
The Computational Neural Network Center (CONNECT)
Electronics Institute, Building 349
Technical University of Denmark
DK-2800 Lyngby, Denmark
emails: jlarsen,lkhansen@ei.dtu.dk

Abstract. This paper addresses the assessment of generalization performance of neural network models by use of empirical techniques.

We suggest to use the cross-validation scheme combined with a resampling technique to obtain an estimate of the generalization performance distribution of a specific model. This enables the formulation of a bulk of new generalization performance measures. Numerical results demonstrate the viability of the approach compared to the standard technique of using algebraic estimates like the FPE [1].

Moreover, we consider the problem of comparing the generalization performance of different competing models. Since all models are trained on the same data, a key issue is to take this dependency into account.

The optimal split of the data set of size N into a cross-validation set of size $N\gamma$ and a training set of size $N(1-\gamma)$ is discussed. Asymptotically (large data sets), $\gamma_{\text{opt}} \to 1$ such that a relatively larger amount is left for validation.

INTRODUCTION

Consider a tapped-delay neural network predicting a stochastic output signal[1] $y(k)$ from observations of the L-dimensional stochastic input vector signal $x(k) = [x(k), x(k-1), \cdots, x(k-L+1)]^\top$. Let $f(\cdot)$ denote the mapping of the neural network, and w the vector of network weights (parameters). Then the prediction is given as: $\widehat{y}(k) = f(x(k); w)$.

[1] For convenience, we focus on single output models.

0-7803-2739-X/95 $4.00 © 1995 IEEE

The basic object of interest in neural network modeling is the conditional input-output distribution $p(y|\boldsymbol{x})$, i.e., the probability distribution of the output conditioned on a test input vector, see e.g., [16]. Normally the network is trained to implement the conditional mean[2], $E\{y|\boldsymbol{x}\} = \int y \cdot p(y|\boldsymbol{x})\, dy$. The first source of uncertainty is the inherent prediction error $\varepsilon = y - E\{y|\boldsymbol{x}\}$ which – per definition – cannot be modeled. Another considerable source of uncertainty is the estimation of $E\{y|\boldsymbol{x}\}$ from a limited number of training data.

This paper deals with empirical assessment of model quality expressed in terms of generalization performance defined as prediction accuracy on future data. Reliable estimates of the generalization performance of a particular model is very important for practical applications. Moreover, in order to choose the best model from a pool of candidate model architectures[3], one requires a test which determines if a particular model has a significantly higher generalization performance than a competing model. The empirical framework enables both *absolute* and *comparative* generalization assessment.

The generalization performance can be decomposed into three components, see e.g., [3], [6]. The first term is due to the inherent prediction error, ε. The second term expresses the insufficiency of the neural architecture[4] to model the conditional mean, and is often referred to as the model bias. Finally, the third term reflects finite training set effects, also known as the model variance. While the first term - per definition - cannot be decreased, there will normally exist a trade off between bias and variance which is accomplished by optimizing the architecture, e.g., by using pruning techniques.

ON GENERALIZATION PERFORMANCE

Suppose the network is trained by minimizing a cost function, viz. the sum of a loss function, $S_N(\boldsymbol{w})$, and a regularization term $R(\boldsymbol{w})$, i.e.,

$$C_N(\boldsymbol{w}) = S_N(\boldsymbol{w}) + R(\boldsymbol{w}) = \frac{1}{N}\sum_{k=1}^{N} \ell\left(y(k), \widehat{y}(k); \boldsymbol{w}\right) + R(\boldsymbol{w}) \qquad (1)$$

where $\ell(\cdot)$ measures the distance between the output $y(k)$ and the network prediction $\widehat{y}(k) = f(\boldsymbol{x}(k); \boldsymbol{w})$. Even though much of the material in this paper applies for general loss functions, often the mean square error loss function, $\ell = (y - \widehat{y})^2 = e^2$ is considered. N defines the number of training examples, i.e., input-output pairs of the training set: $\mathcal{D} = \{(\boldsymbol{x}(k), y(k))\}_{k=1}^{N}$.

Training on the full set of examples provides the estimated weight vector $\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} C_N(\boldsymbol{w})$. The generalization error, G, is defined as the *expected*

[2]This is optimal when using a mean square error cost function, see e.g., [16].

[3]E.g., feed-forward neural nets with different input lag-space and number of hidden units.

[4]The architecture is presumed to be finite, i.e., the weight vector is finite dimensional.

loss of the estimated model on a test sample $(\boldsymbol{x}, y)$ independent of those in the training set,

$$G(\widehat{\boldsymbol{w}}) = E\left\{\ell(y, \widehat{y}; \widehat{\boldsymbol{w}})\right\} = \int \ell(y, \widehat{y}; \widehat{\boldsymbol{w}}) \cdot p(\boldsymbol{x}, y)\, d\boldsymbol{x}\, dy \tag{2}$$

where $E\{\cdot\}$ denotes expectation w.r.t. the unknown joint input-output probability density $p(\boldsymbol{x}, y)$. $G(\widehat{\boldsymbol{w}})$ depends on the actual training set $\mathcal{D}$ through the estimated weights $\widehat{\boldsymbol{w}}$ and has the lower bound $G_{\min} = G(\boldsymbol{w}^*)$. $\boldsymbol{w}^*$ denotes the optimal weight vector $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E\{C_N(\boldsymbol{w})\} = \arg\min_{\boldsymbol{w}} [G(\boldsymbol{w}) + R(\boldsymbol{w})]$ which corresponds to training on an infinite training set. Under fairly mild assumptions, it is possible to show $\lim_{N\to\infty} \widehat{\boldsymbol{w}} = \boldsymbol{w}^*$, see e.g., [10], [11], [17]. $G_{\min}$ expresses the fundamental uncertainty of y when $\boldsymbol{x}$ is known, and furthermore the potential lack of modeling capability, i.e., the network is incapable of implementing the optimal[5] function, $g(\boldsymbol{x}) = \arg\min_{\phi(\boldsymbol{x})} E\{\ell(y, \phi(\boldsymbol{x}))\}$, $\phi(\boldsymbol{x}) : \mathbb{R}^p \mapsto \mathbb{R}$. Insufficient modeling capability is due to two facts:

- In general, when using a finite architecture the model is incomplete, i.e., $f(\boldsymbol{x}; \boldsymbol{w}^\circ) \neq g(\boldsymbol{x})$ where $\boldsymbol{w}^\circ = \arg\min_{\boldsymbol{w}} G(\boldsymbol{w})$ is the weights minimizing the expected loss using the architecture embodied by $f(\cdot)$.
- Regularization implies that the optimal weight vector $\boldsymbol{w}^*$ does not equal $\boldsymbol{w}^\circ$; even when using a complete model.

Since the N samples in $\mathcal{D}$ are randomly selected from the joint density $p((\boldsymbol{x}(1), y(1)), \cdots, (\boldsymbol{x}(N), y(N)))$ the generalization error $G(\widehat{\boldsymbol{w}})$ is stochastic with a certain *generalization error probability distribution* $P(G) = \text{Prob}\{G(\widehat{\boldsymbol{w}}) < G)\}$ and associated density $p(G)$.

The object of interest for model design could be either the full generalization distribution or just the generalization error $G(\widehat{\boldsymbol{w}})$ on the particular training set available. These cases are treated separately in the following. If one has a strong belief in the training set (e.g., if it is large) one might address $G(\widehat{\boldsymbol{w}})$. Otherwise, it might be better to consider the training set as a typical set drawn randomly from the joint input-output distribution in order to reveal the generic characteristics of the employed model.

Since $p(G)$ depends on the true distribution of data, the model architecture, and the number of training data, it is impossible to fully characterize it. However, it is possible to give some general properties. Obviously, $p(G) = 0$ as $G < G_{\min}$. For finite training sets, $p(G)$ will have non-zero values for $G \geq G_{\min}$, and since $\lim_{N\to\infty} \widehat{\boldsymbol{w}} = \boldsymbol{w}^*$, $p(G)$ tends to a Dirac delta function $\delta(G - G_{\min})$ for $N \to \infty$. If the model is complete, the loss function is the mean square error and no regularization is employed, it is possible to show[6] asymptotically as $N \to \infty$, $G(\widehat{\boldsymbol{w}}) \sim \sigma_\varepsilon^2(1 + \chi^2(p)/N)$ where σ_ε^2 is the prediction error noise variance and $\chi^2(p)$ is the χ^2-distribution with $p = \dim(\boldsymbol{w})$ degrees of freedom.

[5] With respect to the employed loss function.

[6] This is done by using second order expansions of $G(\widehat{\boldsymbol{w}})$ around $\boldsymbol{w}^*$, and the fact that that the fluctuations $\Delta\boldsymbol{w} = \widehat{\boldsymbol{w}} - \boldsymbol{w}^*$ are asymptotically Gaussian distributed. See e.g., [6], [7], [8] and [16].

The literature on generalization theory and estimation of generalization error does not in general address the problem of characterizing the full prediction risk probability density. Most work has focused on simple measures of location such as the *average generalization error*

$$avr(G) = E_{\mathcal{D}}\{G(\widehat{\boldsymbol{w}})\} = \int G \cdot p(G)\, dG. \tag{3}$$

This includes algebraic estimators like FPE [1], FPER [7], GEN [5], GPE [9] and NIC [10] which are valid asymptotically $N \to \infty$ and make several assumptions on the model and statistics of the data. However, also algebraic estimates of fractiles of $p(G)$ have been developed, see e.g., [14], [15]. Thus the $1-\alpha$ fractile $G_{1-\alpha}$ defined by $\text{Prob}\{G \leq G_{1-\alpha}\} = 1-\alpha$ guarantees that the probability of G exceeding $G_{1-\alpha}$ is α, which can be set to some low percentage.

EMPIRICAL GENERALIZATION ERROR ESTIMATION

If the object of interest is the generalization error $G(\widehat{\boldsymbol{w}})$ for the particular training set available, we consider the hold-out cross-validation technique [13] for estimating $G(\widehat{\boldsymbol{w}})$. Suppose that a cross-validation set $\mathcal{C}$ of $N_c = \lceil N\gamma \rceil$,[7] $0 < \gamma < 1$, samples are hold out for cross-validation and denote by $\mathcal{T}$ the remaining $N_t = N - N_c$ data for training, i.e., let $\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} C_{N_t}(\boldsymbol{w})$. The cross-validation estimate of $G(\widehat{\boldsymbol{w}})$ then reads:

$$\widehat{G}(\widehat{\boldsymbol{w}}) = \frac{1}{N_c} \sum_{k \in \mathcal{C}} \ell(y(k), \widehat{y}(k); \widehat{\boldsymbol{w}}) \tag{4}$$

Under suitable regularity conditions, $\widehat{G}(\widehat{\boldsymbol{w}}) \to G(\widehat{\boldsymbol{w}})$ as $N_c \to \infty$. However, a very large cross-validation set leaves only few data for training thus increasing $G(\widehat{\boldsymbol{w}})$. Obviously, there exists an optimal fraction γ which trades off the conflicting aims. Assume that the quality of the cross-validation estimator is measured by

$$\text{MSE}(\gamma) = E_{\mathcal{D}}\left\{\left[\widehat{G}(\widehat{\boldsymbol{w}}) - G(\widehat{\boldsymbol{w}})\right]^2\right\} \tag{5}$$

where $E_{\mathcal{D}}\{\cdot\}$ is the expectation w.r.t. all training data. Further, assume that the loss is the mean square error and that the training data are independent. Since $E_{\mathcal{C}}\{\widehat{G}(\widehat{\boldsymbol{w}})\} = G(\widehat{\boldsymbol{w}})$ evaluating Eq. (5) gives

$$\text{MSE}(\gamma) = E_{\mathcal{T}}\left\{\frac{1}{N_c}\left[E_{\mathcal{C}}\left\{e^4(\widehat{\boldsymbol{w}})\right\} - G^2(\widehat{\boldsymbol{w}})\right]\right\} \tag{6}$$

Using asymptotic expansions (see e.g., [6],[7]) for the terms in Eq. (6) and considering the model to be complete, it is possible to show that the the

[7] $\lceil \cdot \rceil$ denotes rounding upwards to the nearest integer.

optimal fraction is given by $\gamma_{\text{opt}} = 1 - \sqrt{\beta/N}$ where $\beta = 4p\sigma_\varepsilon^2/(1-\xi)$, ξ is the kurtosis of the inherent noise (equal to 3 for Gaussian noise), and $p = \dim(\boldsymbol{w})$. That is, $\lim_{N\to\infty} \gamma_{\text{opt}} = 1$ while $N_t = O(\sqrt{N\beta})$ and $N_c = O(N - \sqrt{N\beta})$ asymptotically. It should be emphasized that the choice of γ for a finite small N still needs to be tuned by hand.

The hold-out cross-validation scheme can also be used for comparing generalization errors of different models. Consider the scenario of pruning a nested family of neural net models and suppose that two alternative models with weights $\widehat{\boldsymbol{w}}_1$, $\widehat{\boldsymbol{w}}_2$ both are estimated from $\mathcal{T}$. If we take $\widehat{\boldsymbol{w}}_2$ to be a subset of $\widehat{\boldsymbol{w}}_1$, i.e., $\dim(\boldsymbol{w}_2) < \dim(\boldsymbol{w}_1)$, the hypothesis to be tested is: $G(\widehat{\boldsymbol{w}}_2) > G(\widehat{\boldsymbol{w}}_1)$. Since the models are nested and estimated from the same training set, the corresponding generalization errors are highly dependent. A straight forward procedure which puts error bars on the individual generalization error estimates may fail to unveil the superiority of one model relative to another. The dependence is easily taken into account by analyzing the difference in generalization error, $\Delta\widehat{G} = \widehat{G}(\widehat{\boldsymbol{w}}_2) - \widehat{G}(\widehat{\boldsymbol{w}}_1)$. According to the central limit theorem[8] $\Delta\widehat{G}$ tends to a Gaussian distribution as $N_c \to \infty$. That is a standard t-test for the hypothesis can be used. If $\Delta\widehat{G}/std(\Delta\widehat{G}) < t_\alpha(N_c - 1)$ we reject the hypothesis on an α significance level. $t_\alpha(N_c - 1)$ is the α-fractile of the t-distribution with $N_c - 1$ degrees of freedom, and $std(\cdot)$ denotes the standard deviation. Define $\Delta e^2(k) = e^2(k, \widehat{\boldsymbol{w}}_2) - e^2(k, \widehat{\boldsymbol{w}}_1)$ then the standard deviation is estimated via $(\widehat{std}(\Delta\widehat{G}))^2 = (N_c - 1)^{-1} N_c^{-1} \sum_{k\in\mathcal{C}} (\Delta e^2(k) - \Delta\widehat{G})^2$.

EMPIRICAL GENERALIZATION ERROR DISTRIBUTIONS

We suggest to estimate the generalization error distribution by using leave-out cross-validation [12], [13] and resampling techniques. The basic algorithm is given by:

1. Specify the leave-out fraction γ and determine $N_c = \lceil N\gamma \rceil$. Further specify the number of resamplings $J \leq N!/N_c!(N - N_c)!$.
2. For $j = 1, 2, \cdots, J$ split the training set randomly into a cross-validation subset, $\mathcal{C}_j$, and a training set, $\mathcal{T}_j = \mathcal{D} \setminus \mathcal{C}_j$ not used previously[9].
3. Train on $\mathcal{T}_j$ with $N_t = N - N_c$ examples to obtain the weight estimate $\widehat{\boldsymbol{w}}_j$ and calculate the empirical mean of the loss on the samples $\mathcal{C}_j$, which yields the generalization error estimate:

$$\widehat{G}_j = \widehat{G}_j(\widehat{\boldsymbol{w}}_j) = \frac{1}{N_c} \sum_{k\in\mathcal{C}_j} \ell(y(k), \widehat{y}(k); \widehat{\boldsymbol{w}}_j). \tag{7}$$

The training in step 3 can be very time consuming and in [4] we developed an approximate technique for leave-one-out cross-validation.

[8] This also applies when the error signal is a strongly mixing sequence (time-dependent).

[9] Note that this is resampling without replacement, as opposed to the Bootstrap technique.

Ideally, when estimating Eq. (7) we should train and test on independent sets. Moreover, the training sets should be independent. These properties only hold approximately. First, it is very important to stress the significance leaving out a *fraction* γ compared to the standard approach of leaving out a *fixed* number. In the latter case, the different training sets will be too dependent even in the limit of $N \to \infty$[10]. However, as discussed in the previous section by letting $\gamma \to 1$ and if $N_t = O(\nu \log(N))$, $N_c = O(N - \nu \log(N))$, where ν is a constant, all moments of $\widehat{G}_j$ converges[11]. The number of resamplings J should also be allowed to increase towards infinity as N grows. Secondly, for most signal processing problems, time-dependence can especially for small N cause noise in the estimates. However, asymptotically this is no problem since we expect the input signal to be a strongly mixing sequence, i.e., the time-dependence vanishes for large lags.

From the estimates $\widehat{G}_j$ in Eq. (7) it is possible to form the *empirical generalization error distribution*

$$P_{\text{emp}}(G) = \frac{1}{J} \sum_{j=1}^{J} \mu(G - \widehat{G}_{(j)}) \tag{8}$$

where $\widehat{G}_{(1)} \leq \widehat{G}_{(2)} \cdots \leq \widehat{G}_{(J)}$ is the sample order statistics, and $\mu(G - G_{(j)}) = 1$ when $G \geq G_{(j)}$, and zero otherwise.

Since $p(G)$ is highly non-Gaussian and long tailed (which is demonstrated experimentally below), the mean and variance are not sufficient for characterizing the shape of $p(G)$. It may consequently be desirable to consider more robust location and dispersion measures which we are able to calculate with $P_{\text{emp}}(G)$ in hand. In general the location of $p(G)$ delivers an estimate of the level of generalization error. The dispersion conveys the fluctuation in generalization error and might suggest if the current number of examples is sufficient for learning the task properly. We consider the following quantities:

Location:

- The average $avr(G) = \int G p(G)\, dG$.
- The trimmed average $tavr(G) = \int_{G_{5\%}}^{G_{95\%}} G p(G)\, dG$ which reflects the average in which the highest and lowest 5% of the data are excluded.
- The median $med(G)$ equal to the $\alpha = 50\%$ fractile $G_{50\%}$.

Dispersion:

- The standard deviation $std(G) = (\int [G - avr(G)]^2\, dG)^{1/2}$.
- The median absolute deviation $mad(G) = med(\,|G - med(G)|\,)$.
- The interquartile range $iqr(G) = G_{75\%} - G_{25\%}$.

[10] This is discussed in the literature of the so-called Jackknife estimators, see e.g., [2], [11, Ch. 5.7]

[11] This is a generalization of what was stated in the previous section for convergence of the second order moment in Eq. (5).

Due to the fact that $p(G)$ follows a χ^2 like distribution, we might consider a transformation of G in order to make it more well behaved. In the general family of Box-Cox transformations (see e.g., [11, Ch. 2.8]) we found that a suitable transformation[12] is $\widetilde{G} = \log(1+G)$.

As in the previous section it is possible to compare the generalization ability e.g. by comparing estimated average generalization errors for two models described by w_2, w_1. Define the associated estimates $\widehat{avr}(G)_i = J^{-1}\sum_{j=1}^{J} G_{ij}(\widehat{w}_{ij})$, $i = 1, 2$, and the difference $\Delta\widehat{avr}(G) = \widehat{avr}(G_2) - \widehat{avr}(G_1)$. For J large $\Delta\widehat{avr}(G)$ tends to a Gaussian distribution by the central limit theorem with standard deviation given by

$$std(\Delta\widehat{avr}(G)) = \sqrt{\frac{1}{(J-1)J}\sum_{j=1}^{J}\left[G_{2j} - G_{1j} - \Delta\widehat{avr}(G)\right]^2} \qquad (9)$$

Here the individual differences are assumed to be independent. A standard t-test (as described previously) can then be applied.

NUMERICAL EXAMPLES

Consider the following data generating system: $y(k) = x^\top(k)w^\circ + \varepsilon(k)$. $x(k)$ follows a $L = 10$ variate Gaussian distribution $\mathcal{N}(0, H)$ with H chosen as a random positive definite symmetric matrix. $x(k)$ is time-dependent: each component is a first order AR-process with coefficient 0.6518 scaled to give unit variance; thus implementing a low-pass filter with memory length approx. equal to 7. The noise $\varepsilon(k) \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ is i.i.d. and independent of $x(k)$. The weights, w°, were chosen independently from a $\mathcal{N}(0,1)$ distribution.

We generated $Q = 30000$ independent training sets of size $N = 20$ and trained with a $p = 10$ dimensional linear model using the mean square error cost (without regularization) to obtain the estimates $\widehat{w}^{(i)}$, $i \in [1; Q]$. This enables a highly accurate estimate of the considered generalization performance measures. As an example, the "true" average generalization error is calculated by $avr(G) = Q^{-1}\sum_{i=1}^{Q} G(\widehat{w}^{(i)})$ where $G(\widehat{w}^{(i)}) = \sigma_\varepsilon^2 + (\widehat{w}^{(i)} - w^\circ)^\top H(\widehat{w}^{(i)} - w^\circ)$. For $q = 500$ of the $Q = 30000$ training sets we applied the leave-out procedure with $\gamma = 0.25$, $J = 500$ to obtain the weight estimates $\widehat{w}_j^{(i)}$, and corresponding generalization error estimates $\widehat{G}_j$, $j \in [1; J]$. For comparison we calculated the FPE [1] estimate of $avr(G)$ by $\mathrm{FPE}^{(i)} = S_N(\widehat{w}^{(i)}) \cdot (N+p)/(N-p)$.

Fig. 1 shows the obtained generalization error probability distributions. Table 1 shows a comparison of the suggested measures of location and dispersion. We consider the transformed variables $\widetilde{G}$ which experimentally showed to improve the performance significantly over G. The table indicates that the proposed leave-out technique is fairly accurate for estimating the location and

[12] This implies: $\widetilde{G} = 0$ for $G = 0$.

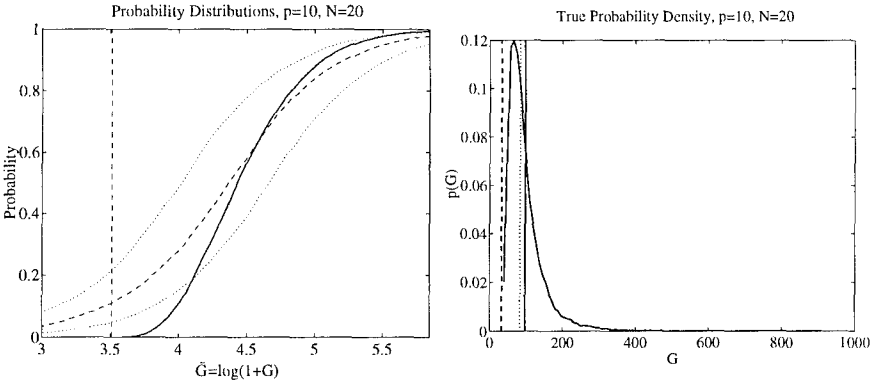


Figure 1: *Left panel*: True (solid) and empirical generalization error distributions as a function of $\widetilde{G} = \log(1+G)$. The dashed line indicates the median distribution of $q = 500$ obtained by leave-out cross-validation while the dotted lines denote the 25% and 75% fractiles. The vertical dashed line is the lower bound $\log(1+\sigma_\varepsilon^2)$. *Right panel*: True generalization error density $p(G)$ estimated from $Q = 30000$ replications. The vertical dashed line is the lower bound $\sigma_\varepsilon^2 = 32.45$. The vertical solid line denotes the average, and the vertical dotted the median. Note that $p(G)$ is highly non-Gaussian and long-tailed (ranges to $G = 1000$ approx.). This implies that the classical measure of location, viz. the average overestimates the typical (the mode) generalization error.

dispersion measures even though the number of training data is only twice as large as the number of weights. Definitely, the leave-out method outperforms the classical FPE estimate at the expense of increased computational complexity. However, the framework offers the possibility of estimating other quantities which are not possible in the asymptotic framework on which FPE relies.

We considered furthermore the comparison of two competing linear models: $\boldsymbol{w}_1$ with dimension $p_1 = 10$, and $\boldsymbol{w}_2$ with dimension $p_2 = 9$ which consequently is an incomplete model of the true data generating system. The true difference in average generalization ability $\Delta avr(G)$ is positive thus indicating that one should prefer model 1 over model 2. Using the same simulation setup as described above the t-test on a specified $\alpha = 5\%$ significance level resulted in that the hypothesis fails to be accepted in approx. 30% of the cases. More over we considered estimating $\text{Prob}(G_2 > G_1)$ from the empirical distributions. It turned out that the estimate tend to under estimate the probability by 20%. Further, it is somewhat more robust than the estimates of the location measures of the generalization error difference.

[14]When considering $\widetilde{G}$ the FPE estimate becomes: $\log(1+\sigma_\varepsilon^2) + \sigma_\varepsilon^2 p/(1+\sigma_\varepsilon^2)N$ with $\sigma_\varepsilon^2 = S_N(\widehat{\boldsymbol{w}})/(N-p)$.

Measure	Min.	25% fract.	Median	75% fract.	Max.
FPE	−62.0	−19.3	−12.3	−5.83	19.7
$\widehat{avr}(G)$	−24.0	−15.5	−7.00	2.27	6.72
$\widehat{tavr}(G)$	−23.5	−15.2	−6.47	2.38	7.05
$\widehat{med}(G)$	−23.2	−15.3	−6.47	3.36	9.31
$\widehat{std}(G)$	44.6	49.9	53.8	62.9	89.4
$\widehat{mad}(G)$	21.6	29.9	40.7	51.4	83.6
$\widehat{iqr}(G)$	20.1	28.6	40.3	53.2	89.4

Table 1: The values are deviations from the true measures in percent when considering the transformed value $\widetilde{G} = \log(1+G)$, e.g., $100\% \cdot (\widehat{avr}(\widetilde{G}) - avr(\widetilde{G}))/avr(\widetilde{G})$. The columns indicate the fluctuation in the deviations w.r.t. the $q = 500$ times the leave-out cross-validation procedure is replicated. As regards FPE^{14}, the fluctuations are based on $Q = 30000$ replications. In median the location measures $avr(G)$, $tavr(G)$, and $med(G)$ seem to underestimate but are still fairly close to zero, and closer than the estimate of $avr(\widetilde{G})$ obtained by FPE. Moreover, the fluctuations are much smaller when considering the fractiles. As regards the dispersion measures $std(G)$, $mad(G)$, and $iqr(G)$ we note that they overestimates by roughly 50%; however, with fairly small amount of fluctuation. The small fluctuation relates strongly to the fact that the transformed variable is used.

CONCLUSION

This paper reports on generalization performance measures which can be attained empirically by using the cross-validation technique in combination with resampling. The major advantage is that the framework provides insight into the shape of the generalization error probability distribution by considering different location and dispersion measures. Traditionally, only the average generalization error has been investigated; however a simple simulation study shows that this measure overestimates the typical generalization performance of a model estimated from a randomly selected set of N examples. Moreover, the assessment of dispersion measures allows for testing the hypothesis whether a model generalizes significantly better than a competitor.

ACKNOWLEDGMENTS

This research was supported by the Danish Natural Science and Technical Research Councils through the Computational Neural Network Center (CONNECT). JL furthermore acknowledge the Radio Parts Foundation for financial support.

REFERENCES

[1] H. Akaike, "Fitting Autoregressive Models for Prediction," Annals of the Institute of Statistical Mathematics, vol. 21, pp. 243–247, 1969.

[2] T. Fox, D. Hinkley & K. Larntz, "Jackknifing in Nonlinear Regression," Technometrics, vol. 22, pp. 29–33, 1980.

[3] S. Geman, E. Bienenstock & R. Doursat, "Neural Networks and the Bias/Variance Dilemma," Neural Computation, vol. 4, pp. 1–58, 1992.

[4] L.K. Hansen & J. Larsen, "Linear Unlearning for Cross-Validation," submitted for publication, 1995. harlar.luloo.ps.Z is retrieved by anonymous ftp at ei.dtu.dk.

[5] J. Larsen, "A Generalization Error Estimate for Nonlinear Systems," in S.Y. Kung, F. Fallside, J. Aa. Sørensen & C.A. Kamm (eds.) Neural Networks for Signal Processing 2: Proceedings of the 1992 IEEE-SP Workshop, Piscataway, New Jersey: IEEE, 1992, pp. 29–38.

[6] J. Larsen, Design of Neural Network Filters, Ph.D. Thesis, Electronics Institute, The Technical University of Denmark, March 1993.

[7] J. Larsen & L.K. Hansen, "Generalization Performance of Regularized Neural Network Models," in J. Vlontzos, J.-N. Hwang & E. Wilson (eds.), Proceedings of the IEEE Workshop on Neural Networks for Signal Processing IV, Piscataway, New Jersey: IEEE, pp. 42–51, 1994.

[8] L. Ljung, System Identification: Theory for the User, Englewood Cliffs, New Jersey: Prentice-Hall, 1987.

[9] J. Moody, "Note on Generalization, Regularization, and Architecture Selection in Nonlinear Learning Systems," in B.H. Juang, S.Y. Kung & C.A. Kamm (eds.) Proceedings of the first IEEE Workshop on Neural Networks for Signal Processing, Piscataway, New Jersey: IEEE, pp. 1–10, 1991.

[10] N. Murata, S. Yoshizawaand & S. Amari, "Network Information Criterion — Determining the Number of Hidden Units for an Artificial Neural Network Model," IEEE Transactions on Neural Networks, vol. 5, no. 6, pp. 865–872, Nov. 1994.

[11] G.A.F. Seber & C.J. Wild, Nonlinear Regression, New York, New York: John Wiley & Sons, 1989.

[12] M. Stone, "Cross-validatory Choice and Assessment of Statistical Predictors," Journal of the Royal Statistical Society B, vol. 36, no. 2, pp. 111–147, 1974.

[13] G.T. Toussaint, "Bibliography on Estimation of Misclassification," IEEE Transactions on Information Theory, vol. 20, no. 4, pp. 472–479, July 1974.

[14] V.N. Vapnik, Estimation of Dependences Based on Empirical Data, New York, New York: Springer-Verlag, 1982.

[15] V.N. Vapnik, "Principles of Risk Minimization for Learning Theory," in J.E. Moody, S.J. Hanson, R.P. Lippmann (eds.) Advances in Neural Information Processing Systems 4, Proceedings of the 1991 Conference, San Mateo, California: Morgan Kaufmann Publishers, 1992, pp. 831–838.

[16] H. White, "Learning in Artificial Neural Networks: A Statistical Perspective," Neural Computation, vol. 1, pp. 425–464, 1989.

[17] H. White, "Consequences and Detection of Misspecified Nonlinear Regression Models," Journal of the American Statistical Association, vol. 76, no. 374, pp. 419–433, June 1981.

ACTIVE LEARNING THE WEIGHTS OF A RBF NETWORK

Kah-Kay Sung and Partha Niyogi

Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Abstract

We describe a principled strategy to sample functions optimally for function approximation tasks. The strategy works within a Bayesian framework and uses ideas from optimal experiment design to evaluate the potential utility of new data points. We consider an application of this general framework for active learning the weight coefficients of a Gaussian Radial Basis Function (RBF) network. We also derive some sufficiency conditions on the learning problem for which there are analytical solutions to the data sampling procedure.

1 Introduction

In most classical formulations of learning from examples, the data (examples) are assumed to be randomly drawn and presented to the learner. This is the case for a variety of situations ranging from network models (Rumelhart et. al. [5], Poggio and Girosi [4]), PAC (Valiant [8]) frameworks, and classical pattern recognition. In this sense, the learner is a *passive* recipient of information about the target function. In contrast, one could consider a learner that plays a more *active* role in collecting its examples. By judiciously selecting examples instead of allowing for possible random sampling, *active learning* techniques can conceivably have faster learning rates and better approximation results than passive learning methods.

Using ideas from Optimal Experiment Design (Fedorov [2]), we have, in an earlier work (Sung and Niyogi [7]) formulated a general procedure for sampling an unknown target function at points in its domain. In this paper, we consider an application of this general framework to encompass the active learning of the coefficients of a Radial Basis Function (RBF) network (Poggio and Girosi [4]). More generally, our solution allows us to handle parameter estimation problems for function classes that are linear in their parameters. In this paper, our contributions are:

1. The application of an Optimal Experiment Design framework for estimating the weights of RBF-type function classes, and the analytical derivation of an optimal sampling strategy for parameter estimation of such classes from data. This represents a significant improvement over existing work on toy function classes, like step functions considered in

0-7803-2739-X/95 $4.00 © 1995 IEEE

(Sung and Niyogi [7]). Recently, Sollich [6] has applied a similar optimal experiment design framework for estimating the weights of a single perceptron unit. To the best of our knowledge, this analysis has not been done before on RBF-type networks.

2. Empirical comparisons of the data requirements for active and passive learning of RBF-type function classes.

3. Development of sufficiency conditions on the learning problem for which an active data selection sequence can be analytically derived and precomputed within our optimal experiment design framework.

We now provide a brief statement of our problem, our solution for the active sampling strategy, and a glimpse of the empirical simulations.

2 The General Formulation

We adopt a Bayesian formulation for active example selection. Specifically, we can pose the problem as follows: Let $\mathcal{D}_n = \{(x_i, y_i) | i = 1, \ldots, n\}$ be a set of n data points sampled from an unknown target function, $g_{\mathbf{t}}$, possibly in the presence of noise. Given a class of approximation functions, $\mathcal{F}$, where each $f \in \mathcal{F}$ has prior probability $P_{\mathcal{F}}(f)$, one can use regularization techniques to approximate $g_{\mathbf{t}}$ from $\mathcal{D}_n$ (in a MAP sense) by means of a function $\hat{g} \in \mathcal{F}$. We want a strategy to determine at what input location one should sample the next data point, (x_{n+1}, y_{n+1}), in order to obtain the "best" possible Bayes optimal approximation of the unknown target function $g_{\mathbf{t}}$ with our concept class $\mathcal{F}$.

We approach the active example selection problem in two stages:

1. **Define the notion of a "best" possible Maximum A-Posteriori (MAP) approximation for an unknown target function.** We propose an optimality criterion for evaluating how well a solution approximates an *unknown* target function. Our active learning goal is to find a solution $\hat{g} \in \mathcal{F}$ that best approximates the unknown target function in terms of the optimality criterion.

2. **Formalize mathematically the task of determining the best input location to sample next.** We express the active learning optimality criterion as a cost function to be minimized. The task of choosing the next data point becomes one of minimizing the cost function with respect to the next sample's input location.

Earlier work by Cohn [1] and MacKay [3] have tried using similar optimal experiment design techniques [2] to collect data that maximizes information about an unknown target function. Our work differs from theirs in two respects. First, we use a different, and perhaps more general, optimality

criterion for evaluating solutions to an unknown target function, based on a measure of function uncertainty that incorporates both bias and variance components of the total *output* generalization error. In contrast, MacKay and Cohn consider only the variance component in *model parameter* space. Second, we also show that our active example selection strategy requires fewer training examples than passive methods to learn a target function to a given degree of accuracy.

2.1 The Formulation

We begin by establishing an optimality criterion for measuring the quality of an approximation function with respect to an unknown target. Recall that our active learning goal is to find solutions that best approximate an unknown target according to our proposed optimality criterion. From the optimality criterion, we can derive a scheme for measuring a new example's utility in terms of how well the example steers the learner toward the goal, and an accompanying computational procedure for selecting the next training example.

Let $g_{\mathbf{t}}$ be a target function that we want to estimate by means of an approximation function $\hat{g} \in \mathcal{F}$. If the target function $g_{\mathbf{t}}$ were known, then one popular measure of how well (or badly) $\hat{g}$ approximates $g_{\mathbf{t}}$ is their *Integrated Squared Difference* (ISD) over an appropriate region of interest $\mathcal{X}$:

$$\delta(\hat{g}, g_{\mathbf{t}}) = \int_{x \in \mathcal{X}} (g_{\mathbf{t}}(x) - \hat{g}(x))^2 dx. \tag{1}$$

In most function approximation tasks, the target $g_{\mathbf{t}}$ is unknown, so we clearly cannot express the quality of a learning result, $\hat{g}$, in terms of $g_{\mathbf{t}}$. We can, however, compute an *expected* integrated squared difference (EISD) between $\hat{g}$ and its *unknown* target, $g_{\mathbf{t}}$, by treating the unknown target $g_{\mathbf{t}}$ as a random variable in the approximation function concept class $\mathcal{F}$. Notice that the distribution of $g_{\mathbf{t}}$ is simply its a-posteriori likelihood given $\mathcal{D}_n$, the n data points seen so far: i.e. $P(g_{\mathbf{t}}|\mathcal{D}_n) \propto P_{\mathcal{F}}(g_{\mathbf{t}})P(\mathcal{D}_n|g_{\mathbf{t}})$. We shall use the EISD as a criterion for evaluating the quality of an approximation result, $\hat{g}$, for an *unknown* target function $g_{\mathbf{t}}$:

$$\begin{aligned}
\texttt{EISD}(\hat{g}, g_{\mathbf{t}}) &= E_{g_{\mathbf{t}} \in \mathcal{F}}[\delta(\hat{g}, g_{\mathbf{t}})|\mathcal{D}_n] \\
&= \int_{g_{\mathbf{t}} \in \mathcal{F}} P(g_{\mathbf{t}}|\mathcal{D}_n)\delta(\hat{g}, g_{\mathbf{t}})dg_{\mathbf{t}} \\
&= \int_{g_{\mathbf{t}} \in \mathcal{F}} P_{\mathcal{F}}(g_{\mathbf{t}})P(\mathcal{D}_n|g_{\mathbf{t}})\delta(\hat{g}, g_{\mathbf{t}})dg_{\mathbf{t}}.
\end{aligned}$$

Let $\hat{g}_n$ be the approximation result given n data points, $\mathcal{D}_n$. Our learning goal is to minimize $EISD(\hat{g}_n, g_{\mathbf{t}})$ for each new n, and so a reasonable active

example selection strategy would be to choose the next input location, x_{n+1}, that minimizes $EISD(\hat{g}_{n+1}, g_t)$. For a given x_{n+1}, we can predict the new EISD that results from sampling our next data point there as follows:

Suppose we also know the target output value y_{n+1} at the given x_{n+1}. Our new data set would then be $\mathcal{D}_{n+1} = \mathcal{D}_n \cup \{(x_{n+1}, y_{n+1})\}$, and the new EISD between g_t and its new estimate $\hat{g}_{n+1}$ is given by $E_{\mathcal{F}}[\delta(\hat{g}_{n+1}, g_t)|\mathcal{D}_{n+1}]$, where the new estimate, $\hat{g}_{n+1}$, can be derived from the new data set, $\mathcal{D}_{n+1}$ via regularization. In reality, we do not know the actual value of y_{n+1}, but we can derive for it the following conditional probability distribution:

$$\begin{aligned} P(y_{n+1}|x_{n+1}, \mathcal{D}_n) &= \int_{f \in \mathcal{F}} P(y_{n+1}|x_{n+1}, f) P(f|\mathcal{D}_n) df \\ &\propto \int_{f \in \mathcal{F}} P(\mathcal{D}_n \cup \{(x_{n+1}, y_{n+1})\}|f) P_{\mathcal{F}}(f) df. \end{aligned}$$

So, for a given next location to sample, x_{n+1}, we can compute the *expected* value of the resulting EISD measure:

$$\begin{aligned} \mathcal{U}(\hat{g}_{n+1}|x_{n+1}, \mathcal{D}_n) &= \mathrm{E}_{y_{n+1}}[\mathrm{E}_{\mathcal{F}}[\delta(\hat{g}_{n+1}, g_t)|\mathcal{D}_{n+1} \cup (x_{n+1}, y_{n+1})]|x_{n+1}, \mathcal{D}_n] \\ &= \int_{y_{n+1} \in \Re} P(y_{n+1}|x_{n+1}, \mathcal{D}_n) \\ &\qquad E_{\mathcal{F}}[\delta(\hat{g}_{n+1}, g_t)|\mathcal{D}_n \cup (x_{n+1}, y_{n+1})] dy_{n+1}. \end{aligned} \quad (2)$$

Clearly, the optimal location to sample next is the location that minimizes the above cost function:

$$\hat{x}_{n+1} = \arg\min_{x_{n+1}} \mathcal{U}(\hat{g}_{n+1}|x_{n+1}, \mathcal{D}_n).$$

The important questions at this stage are:

1. Can the equations above be analytically solved to yield a specific sampling procedure?
2. Does the sampling procedure allow one to learn the target function with fewer examples? Does it reduce the sample complexity?

We answer these questions in the context of estimating the weight parameters of a Radial Basis Function network. In particular, we are able to derive analytical solutions to the equations above and compare the sample complexities of active and passive learning.

3 Radial Basis Function Networks

3.1 The Function Class $\mathcal{F}$

We consider a class of d-dimensional Gaussian Radial Basis Functions with K fixed centers where the ith basis function, $\mathcal{G}_i$, has a fixed covariance of $\mathcal{S}_i$, and

a center c_i. The coefficients to be learned are denoted by $\mathrm{a} = [a_1\ a_2\ \ldots\ a_K]^{\mathrm{T}}$. Specifically, an arbitrary function in this class can be represented as:

$$\begin{aligned} g_{\mathrm{a}}(\mathrm{x}) &= \sum_{i=1}^{K} a_i \mathcal{G}_i(\mathrm{x}) \\ &= \sum_{i=1}^{K} a_i \frac{1}{(2\pi)^{d/2}|\mathcal{S}_i|^{1/2}} \exp(-\frac{1}{2}(\mathrm{x}-\mathrm{c}_i)^{\mathrm{T}}\mathcal{S}_i^{-1}(\mathrm{x}-\mathrm{c}_i)) \end{aligned}$$

The priors on the class $\mathcal{F}$ are obtained by putting a normal distribution with covariance $\Sigma_{\mathcal{F}}$ on the coefficients a. Thus, for an arbitrary g_{a} :

$$P(g_{\mathrm{a}}) = P(\mathrm{a}) = \frac{1}{(2\pi)^{K/2}|\Sigma_{\mathcal{F}}|^{1/2}} \exp(-\frac{1}{2}\mathrm{a}^{\mathrm{T}}\Sigma_{\mathcal{F}}^{-1}\mathrm{a}).$$

Finally, the learner has access to noisy data of the form $\mathcal{D}_n = \{(\mathrm{x}_i, y_i = g_t(\mathrm{x}_i) + \eta) : i = 1, \ldots, n\}$, where g_t is the unknown target function, and η is zero-mean additive Gaussian noise with variance σ_η^2. This leads to an expression for $P(\mathcal{D}_n|g_{\mathrm{a}})$ for every candidate $g_{\mathrm{a}} \in \mathcal{F}$ that is of the form:

$$P(\mathcal{D}_n|g_{\mathrm{a}}) \propto \exp\left(-\sum_{i=1}^{n} \frac{(y_i - g_{\mathrm{a}}(\mathrm{x}_i))^2}{2\sigma_\eta^2}\right).$$

The weight parameters, $\hat{\mathrm{a}}$, of the MAP solution ($g_{\hat{\mathrm{a}}}$) to the learning problem can then be recovered as follows:

$$\hat{\mathrm{a}} = \mathcal{C}_n \mathrm{b}$$

where:

$$\begin{aligned} \mathrm{b} &= \frac{1}{\sigma_\eta^2}[\sum_{i=1}^{n} y_i \mathcal{G}_1(\mathrm{x}) \ \sum_{i=1}^{n} y_i \mathcal{G}_2(\mathrm{x}_i) \ \ldots \ \sum_{i=1}^{n} y_i \mathcal{G}_K(\mathrm{x}_i)]^{\mathrm{T}} \\ \mathcal{C}_n^{-1} &= \Sigma_{\mathcal{F}}^{-1} + \Sigma_n^{-1} \end{aligned} \tag{3}$$

and Σ_n^{-1} is a $K \times K$ matrix whose $(i,j)^{\mathrm{th}}$ element is: $\frac{1}{\sigma_\eta^2}\sum_{k=1}^{n} \mathcal{G}_i(\mathrm{x}_k)\mathcal{G}_j(\mathrm{x}_k)$.

This is the solution the learner proposes on the basis of the data set *irrespective* of how the data points are selected. We now provide an active strategy for selecting the data optimally.

3.2 Active Sampling

Recall that we are interested in answering the two questions at the end of Section 2.

1. For the RBF class considered here, it is possible to show that an analytical expression exists for Equation 2, the cost function to minimize that yields the optimal location to sample next:

$$U(\hat{g}_{n+1}|\mathbf{x}_{n+1}, \mathcal{D}_n) \propto |\mathcal{C}_{n+1}|$$

$\mathcal{C}_{n+1}$ has the same form as $\mathcal{C}_n$ of Equation 3, and depends on the previous data sample locations $\{\mathbf{x}_i : i = 1, \ldots, n\}$, the weight priors $\Sigma_{\mathcal{F}}$, σ_η and $\mathbf{x}_{n+1}$. When minimized over $\mathbf{x}_{n+1}$, we get $\hat{\mathbf{x}}_{n+1}$ as the maximum utility location where the active learner should next sample the unknown target function.

2. Is it indeed the case that the active strategy outperforms the passive one? To investigate this we randomly generated 5000 target RBF functions with 8 fixed, arbitrarily chosen centers. The priors on the weights $\mathbf{a}$ are provided by $\Sigma_{\mathcal{F}} = \mathbf{I}_K$. For each target RBF function, we collected data according to the active strategy as well as the passive (uniformly random) strategy for varying number of data points. Figure 1 shows the mean error rate (i.e., the integrated squared difference, Equation 1, between the *actual* target function and its MAP *estimate*, averaged over the 5000 different target functions) as a function of the number of data points. Notice that the active strategy has a lower mean error rate than the passive for the same number of examples and is particularly true for small number of data. The left graph shows a situation where the learner has knowledge of the true priors on $\mathcal{F}$. The right graph is for a case where the learner has the following incorrect prior assumptions: (a) the true centers of the target RBF's are slightly different from the values that the learner assumes them to be, and (b) the learner's priors on the weights ($\Sigma_{\mathcal{F}} = 0.9\mathbf{I}_K$) are different from that on the target class ($\Sigma_{\mathcal{F}} = \mathbf{I}_K$). Despite these incorrect prior assumptions, the active strategy still outperforms the passive case.

4 Sufficiency Conditions for Pre-Computing a Data Sampling Sequence

It is noteworthy that for learning RBF weights, the new optimal sample location, $\mathbf{x}_{n+1}$, does not depend on the y_i data values previously observed but only on the $\mathbf{x}_i$ values sampled. Thus, if the learner were to collect n data points, it can pre-compute the exact sequence of n points at which to sample from the start, even before receiving any data from the target function. This behavior has been observed by MacKay [3] for an active example selection strategy that minimizes only a model parameter variance cost function. For such cost functions, any class of approximation functions that are linear in their model parameters would exhibit such behavior.

In our framework, we minimize an output uncertainty cost function that includes both bias and variance terms. The following theorem provides sufficiency conditions on the learning problem for which our active learning for-

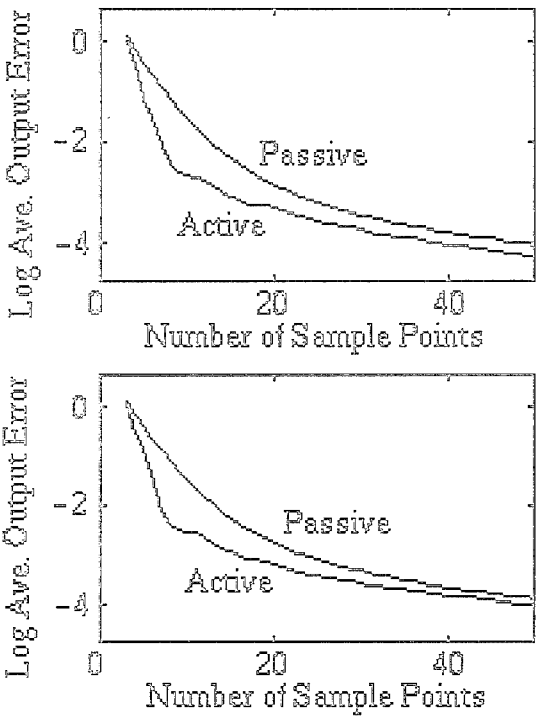


Figure 1: Comparing the active and passive mean error rates for learning an unknown RBF target function with 8 fixed centers. **Top:** The learner uses the same priors on model parameters as the process that randomly generates the unknown target functions. **Bottom:** The learner uses slightly different weight priors and has centers that are slightly displaced from the true centers.

mulation leads to a data selection strategy that does not depend on previously observed y_i data values.

Theorem 1 *Suppose $\mathcal{F}$ is a class of real-valued functions parameterized by $\mathrm{a} \in \Re^k$. On the basis of a data set $\mathcal{D}_n = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$, let the MAP solution to the learning problem be given by $\hat{\mathrm{a}} = \arg\min_{\mathrm{a} \in \Re^k} P(g(\mathrm{a}|\mathcal{D}_n))$. Then the following three conditions guarantee that the choice of $\hat{\mathbf{x}}_{n+1}$ will be independent of the previously observed y_i's in $\mathcal{D}_n$.*

1. *$P(g(\mathrm{a})|D_n)$ can be expressed as $Q((\mathrm{a} - \hat{\mathrm{a}}(\mathcal{D}_n)), \{\mathbf{x}_i : i = 1 \ldots n\})$ where Q is some arbitrary function that does not depend on the data, $\mathcal{D}_n$. In other words, the y_i terms of $\mathcal{D}_n$ do not appear anywhere else in $P(g(\mathrm{a})|D_n) = Q((\mathrm{a} - \hat{\mathrm{a}}(\mathcal{D}_n)), \{\mathbf{x}_i : i = 1 \ldots n\})$ other than in $\hat{\mathrm{a}}$.*

2. *$\mathcal{F}$ is linear in its parameters* $\mathbf{a}$*, i.e.:* $g_{\mathbf{a}_1+\mathbf{a}_2}(\mathbf{x}) = g_{\mathbf{a}_1}(\mathbf{x}) + g_{\mathbf{a}_2}(\mathbf{x})$.

3. *The prior distribution on model parameters* $\mathbf{a}$ *has support* $\Re^k$.

5 Remarks and Conclusions

We have extended a previously developed Bayesian framework for active learning to the case of learning the weights of an RBF network. We derive a specific sampling strategy, and show how this strategy allows us to learn with fewer examples (alternatively, make smaller error with the same number of examples) than random (*passive*) sampling. This is an application of the optimal experiment design paradigm to function approximation and seems to bear promise for the design of more efficient learning algorithms.

References

[1] D. Cohn. A Local Approach to Optimal Queries. In D. Touretzky, editor, *Proc. of 1990 Connectionist Summer School*, San Mateo, CA, 1991. Morgan Kaufmann Publishers.

[2] V. Fedorov. *Theory of Optimal Experiments*, page 35. Academic Press, New York, 1972.

[3] D. MacKay. *Bayesian Methods for Adaptive Models.* PhD thesis, California Institute of Technology, Pasadena, CA, 1992.

[4] T. Poggio and F. Girosi. A Theory of Networks for Approximation and Learning. Technical Report AIM–1140, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1989.

[5] D. Rumelhart and J. McClelland. *Parallel Distributed Processing*, volume 1. MIT Press, Cambridge, Massachusetts, 1986.

[6] P. Sollich. Query Construction, Entropy, Generalization in Neural Network Models. *Physical Review E*, 49:4637–4651, 1994.

[7] K. Sung and P. Niyogi. Active Learning for Function Approximation. In *Advances in Neural Information Processings Systems 7*, San Mateo, CA, 1995. Morgan Kaufman.

[8] L.G. Valiant. A theory of learnable. *Proc. of the 1984 STOC*, pages 436–445, 1984.

A Novel Approach to Pattern Recognition Based on Discriminative Metric Design

Hideyuki WATANABE, Tsuyoshi YAMAGUCHI, and Shigeru KATAGIRI
ATR Interpreting Telecommunications Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan
Tel.: +81-774-95-1383
Fax.: +81-774-95-1308
E-mail: watanabe@itl.atr.co.jp

Abstract

This paper proposes a novel approach, named *Discriminative Metric Design (DMD)*, to pattern recognition. DMD optimizes the whole metrics of discriminant functions with the Minimum Classification Error/Generalized Probabilistic Descent method (MCE/GPD) such that the intrinsic features of each pattern class can be represented efficiently. The resulting metrics lead accordingly to robust recognizers. DMD is quite general. Several existing methods, such as Learning Vector Quantization, Subspace Method, Discriminative Feature Extraction, Radial-Basis Function Network, and the Continuous Hidden Markov Model, are defined as its special cases. Among the many possibilities, this paper specifically elaborates the DMD formulation for recognizing fixed-dimensional patterns using quadratic discriminant functions, and clearly demonstrates its utility in a speaker-independent Japanese vowel recognition task.

1 Introduction

Over-learning is a long-standing problem in the statistical approach to pattern recognizer design. Because recognizers are inevitably trained using only finite resources, specially finite design samples, recognition accuracy over the finite design sample set is not necessarily equivalent to that over unknown samples. Actually, a recognizer achieving high accuracy over design data sometimes degrades in performance over unknown data. This is the so-called over-learning phenomenon, and generally exists in statistical estimation tasks.

0-7803-2739-X/95 $4.00 © 1995 IEEE

Pattern recognition usually consists of front-end feature extraction and back-end classification. In most cases, high recognition accuracies have been achieved by increasing the number of classifier parameters. For example, by using Learning Vector Quantization (LVQ), one can easily achieve a highly-performing distance classifier. This discriminative design method that enables one to handle multiple prototypes per class achieves more accurate classification over design data by using more prototypes, i.e., trainable classifier parameters. However, such high performance over design data does not necessarily guarantee high accuracy over unknown data (See the later section.). Obviously, here we are faced with the over-learning problem.

A fundamental solution to the above problem has been to increase the statistical stability of estimation by reducing the number of classifier parameters, or degrees of freedom of the model, relative to the number of available design samples. However, as still vigorously discussed in Artificial Neural Networks studies, satisfactory solutions to this problem have not obtained yet. Our purpose in this paper is therefore to develop a method of alleviating the over-learning through re-consideration about the other important factor of the recognition process: feature extraction.

It is naturally desirable that the feature extraction should appropriately interact with the classification: This interaction enables one to discover features useful for the classification. However, in practice, no recognizers yet include such interaction. Incorporating the interaction in the recognizer design would lead to features peculiar to each class, make easier the classification, reduce classifier parameters, and accordingly alleviate the over-learning phenomenon. In this light, we propose in this paper a new method, called Discriminative Metric Design (DMD), of achieving discriminant function metrics for minimizing recognition errors, in other words, misclassifications.

DMD is a general method based on the Minimum Classification Error/Generalized Probabilistic Descent method (MCE/GPD), and can be applied to various types of recognizers as well as a wide range of recognition tasks. By way of example, we focus in the paper on DMD implementation for a recognizer using the fundamental quadractic discriminant function to recognize *static* (fixed-dimensional) patterns, and on its evaluation in a speaker-independent Japanese five-vowel recognition task.

The relation has been clarified between MCE/GPD and LVQ [4]. The development of DMD greatly widens the scope of such relationship analysis. In the paper, we specially clarify the relationships between DMD and important design algorithms, i.e., LVQ, Subspace Method (SM) [6], and Discriminative Feature Extraction (DFE) [1]. We also describe that DMD is related to a segmental GPD-trained continuous Gaussian HMM recognizer [5] and an MCE/GPD-trained kernel function recognizer [3].

2 Discriminative Metric Design (DMD)

2.1 Statistical Pattern Recognition

We consider the problem of classifying an input pattern $x \in \mathcal{X}$, where $\mathcal{X}$ is the pattern space, into one of the K classes $\{C_s\}_{s=1}^{K}$. x may be either static or *dynamic* (variable-dimensional). Our decision rule is as follows:

$$C(x): \quad C(x) = C_i \quad \text{if} \quad i = \arg\min_s g_s(x), \tag{1}$$

where $C(x)$ is the recognition operation and $g_s(x)$ is the discriminant function that indicates the degree to which x belongs to C_s. The ultimate goal here is to achieve discriminant functions that can minimize the recognition error probability. In reality, however, despite many approaches, achieving this goal has been rather difficult due to the limited amount of available resources such as design samples.

2.2 General Formulations of DMD

In most cases, the discriminant function is simply based on heuristics and on some kind of scientific knowledge indirectly related to error minimization. However, such functions are never guaranteed to lead to a *robust* recognition. One way to remedy this inadequacy is to design each discriminant function so as to represent the salient, intrinsic features of its corresponding class efficiently.

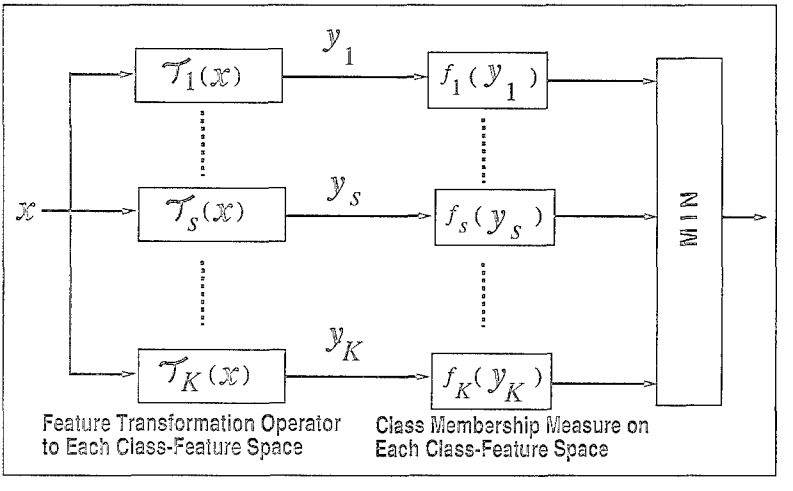


Figure 1: Pattern Recognizer Based on DMD

Our solution, DMD, is illustrated in Figure 1. Each discriminant function $g_s(x)$ $(s = 1, 2, ..., K)$ involves a *class-feature transformation*

operator $\mathcal{T}_s$ defined as a mapping from the original pattern space $\mathcal{X}$ to the s-th *class feature space* $\mathcal{Y}_s$ and a *class-membership measure* f_s defined on $\mathcal{Y}_s$. Hereafter we call the set $\{\mathcal{T}_s, f_s\}$ the *metric*. DMD optimizes each metric with MCE/GPD [2] so as to minimize the recognition error probability. This design can consequently increase the design robustness: Each class membership is evaluated in its corresponding class-feature space where features relevant to recognition are emphasized and information irrelevant to recognition is suppressed.

Note that $\boldsymbol{x}$ is not restricted to a feature vector but may be a pattern before feature extraction process; the latter case implies that $\mathcal{T}_s$ execute the individual feature extraction in each class.

3 An Exemplar Implementation of DMD

This paper specially elaborates the above formulation for the case of recognizing d-dimensional static patterns using a linear transformation $\mathcal{T}_s$ and a Euclidean distance-based measure f_s $(s = 1, 2, ..., K)$:

$$\boldsymbol{y}_s = \mathcal{T}_s(\boldsymbol{x}) = \boldsymbol{W}_s\boldsymbol{x}, \tag{2}$$

$$f_s = ||\mathcal{T}_s(\boldsymbol{x}) - \mathcal{T}_s(\boldsymbol{r}_s)||^2, \tag{3}$$

therefore

$$g_s(\boldsymbol{x}) = (\boldsymbol{x} - \boldsymbol{r}_s)^T \boldsymbol{W}_s^T \boldsymbol{W}_s (\boldsymbol{x} - \boldsymbol{r}_s), \tag{4}$$

where the superscript ^{T} denotes the matrix transpose, each $\boldsymbol{W}_s$ has a size of $d \times d$ and $\boldsymbol{r}_s$ denotes the reference vector of $\boldsymbol{C}_s$. Consequently, each discriminant function comes to quadratic-form function. The DMD (GPD) training is applied to $\{\boldsymbol{W}_s\}_{s=1}^K$ and $\{\boldsymbol{r}_s\}_{s=1}^K$. The updating rule for these parameters can be derived using the chain rule of differentiation and given as follows: for a sample $\boldsymbol{x}_t$ which is selected randomly at the updating step t from labeled design samples and belongs to $\boldsymbol{C}_k$,

$$\boldsymbol{r}_s^{(t)} = \boldsymbol{r}_s^{(t-1)} - \varepsilon_t \ell_k'(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}) \rho_{k,s}(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}) \nabla_{\boldsymbol{r}_s} g_s(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}), \tag{5}$$

$$\boldsymbol{W}_s^{(t)} = \boldsymbol{W}_s^{(t-1)} - \varepsilon_t \ell_k'(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}) \rho_{k,s}(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}) \nabla_{\boldsymbol{W}_s} g_s(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}), \tag{6}$$

where

$$\nabla_{\boldsymbol{r}_s} g_s(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}) = -2\boldsymbol{W}_s^{(t-1)^T} \boldsymbol{W}_s^{(t-1)} (\boldsymbol{x}_t - \boldsymbol{r}_s^{(t-1)}), \tag{7}$$

$$\nabla_{\boldsymbol{W}_s} g_s(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}) = 2\boldsymbol{W}_s^{(t-1)} (\boldsymbol{x}_t - \boldsymbol{r}_s^{(t-1)}) (\boldsymbol{x}_t - \boldsymbol{r}_s^{(t-1)})^T, \tag{8}$$

the suffix t denotes its corresponding parameter status at t $(t = 1, 2, ...)$, $\boldsymbol{\Lambda}$ represents $\left(\{\boldsymbol{r}_s\}_{s=1}^K, \{\boldsymbol{W}_s\}_{s=1}^K\right)$, ε_t (> 0) is the learning coefficient satisfying $\sum_{t=1}^{\infty} \varepsilon_t \to \infty$ and $\sum_{t=1}^{\infty} \varepsilon_t^2 < \infty$, $\ell_k'(\boldsymbol{x}; \boldsymbol{\Lambda})$ (> 0) denotes the derivative (with respect to the misclassification measure) of the smooth

loss function (see [2]), and $\rho_{k,s}(x;\Lambda)$ denotes the derivative (with respect to the s-th discriminant function) of the misclassification measure (see [2]) satisfying $\rho_{k,k} > 0$ and $\rho_{k,j} < 0$ $(j \neq k)$. Properties of the updating convergence are discussed in [2].

The effect of this exemplar DMD implementation can be elucidated by the singular value decomposition (SVD) of $\bar{W}_s$, which is given as

$$\bar{W}_s = \bar{U}_s \bar{\Phi}_s \bar{V}_s^T, \tag{9}$$

where $\bar{U}_s$ and $\bar{V}_s$ are the orthonormal matrices of size $d \times d$, and $\bar{\Phi}_s$ is the $d \times d$ diagonal matrix whose diagonal elements $\{\varphi_{s,i}\}_{i=1}^{d}$, called singular values, are non-negative. The above equation says that $\bar{W}_s$ projects the input vector into each column vector of $\bar{V}_s$, weights each component with $\varphi_{s,i}$, and assigns each to its corresponding base vector of $\mathcal{Y}_s$, i.e. the column vector of $\bar{U}_s$. Note that $\bar{U}_s$ is independent of the recognition decision in the case of the Euclidean distance measure f_s. It can be seen that the projected components with larger singular values, which contribute more to the class-membership evaluation, correspond to more relevant information for recognition decision. DMD finds the weighting parameter set $\bar{\Phi}_s$ and the orthonormal base $\bar{V}_s$ which are optimal for recognition decision.

4 Experimental Results

To evaluate DMD, we conducted a five-class, fixed-dimensional Japanese vowel pattern recognition experiment in a speaker-independent mode. Vowel tokens were extracted from 520 isolated words spoken by 70 speakers (36 males and 34 females), and digitized at a sampling rate of 12 kHz. The center fragment of each vowel segment was selected using a 20 ms Hamming window and converted into a recognizer input pattern consisting of 16 LPC cepstral coefficients without the 0-th coefficient. Note that each pattern sample was a static cepstral vector.

We demonstrate the recognition performance of the DMD-trained recognizers. For careful execution, we computed *averages* and *standard deviations* of recognition error rates with five trials as follows;

For $n = 1$ to 5 (the n-th trial) {

> Select 50 speakers randomly for *unknown set* $\Omega_u^{(n)}$ (about 7500 samples) from the whole set Ω of 70 speakers;
> Let $\bar{\Omega}_u^{(n)}$ be the remaining set of 20 speakers;
>
> For $m = 1$ to 5 {
>
> > Select 10 speakers randomly for *validation set* $\Omega_v^{(n,m)}$ (about 1500 samples) from $\bar{\Omega}_u^{(n)}$;
> > Let $\Omega_d^{(n,m)}$ be the remaining *design set* of 10 speakers (about 1500 samples);

Train the recognizer $\Lambda^{(n,m)}$ using $\Omega_d^{(n,m)}$;

Compute the error rate $P_{ev}^{(n,m)}$ of $\Lambda^{(n,m)}$ for the validation set $\Omega_v^{(n,m)}$;

}

Select the *best* recognizer $\Lambda^{(n,m^*)}$ where $P_{ev}^{(n,m^*)} = \min_m P_{ev}^{(n,m)}$;
Compute the error rate $P_{ed}^{(n)}$ and $P_{eu}^{(n)}$ of $\Lambda^{(n,m^*)}$ for the design set $\Omega_d^{(n,m^*)}$ and the unknown set $\Omega_u^{(n)}$, respectively;

}

Compute the averages and standard deviations of the error rates with the five trials $\bar{P}_{ed} = (1/5)\sum_{n=1}^{5} P_{ed}^{(n)}$ and $SD_{ed} = \sqrt{(1/5)\sum_{n=1}^{5} {P_{ed}^{(n)}}^2 - \bar{P}_{ed}^2}$ for the design set, and $\bar{P}_{eu} = (1/5)\sum_{n=1}^{5} P_{eu}^{(n)}$ and $SD_{eu} = \sqrt{(1/5)\sum_{n=1}^{5} {P_{eu}^{(n)}}^2 - \bar{P}_{eu}^2}$ for the unknown set.

For comparison purposes, we used four types of recognizers: 1) quadratic discriminant-based DMD recognizers, 2) a Euclidean distance recognizer, 3) a Mahalanobis distance recognizer, and 4) multi-template (reference) LVQ recognizers. Each LVQ system used the Euclidean distance for its discriminant function. Note that LVQ system was trained using MCE/GPD [4]. All of the parameters in the DMD-based recognizer were initialized using the Euclidean distance or the Mahalanobis distance. In both initialization schemes, each reference vector $\boldsymbol{r}_s$ was the sample mean vector of $\boldsymbol{C}_s$. As for the initial value of each transformation matrix $\boldsymbol{W}_s$, $\boldsymbol{W}_s^{(0)}$ was the identity matrix in the Euclidean distance-based initialization; in the Mahalanobis distance-based initialization,

$$\boldsymbol{W}_s^{(0)} = \boldsymbol{\Gamma}_s \boldsymbol{E}_s^T, \tag{10}$$

$$\boldsymbol{\Gamma}_s = \mathrm{diag}\left(\frac{1}{\sqrt{\gamma_{s,d}}} \frac{1}{\sqrt{\gamma_{s,d-1}}} \cdots \frac{1}{\sqrt{\gamma_{s,2}}} \frac{1}{\sqrt{\gamma_{s,1}}} \right), \tag{11}$$

$$\boldsymbol{E}_s = [\, \boldsymbol{e}_{s,d}\ \boldsymbol{e}_{s,d-1} \cdots \boldsymbol{e}_{s,2}\ \boldsymbol{e}_{s,1}\,], \tag{12}$$

where $\gamma_{s,i}$ and $\boldsymbol{e}_{s,i}$ represent the i-th eigenvalue and its corresponding eigenvector of the sample covariance matrix of $\boldsymbol{C}_s$, respectively.

Table 1 summarizes the averages of recognition error rates for these three systems. For reference, the standard deviations of error rates are also shown in Table 2. The DMD-based recognizer initialized by the Euclidean distance achieved the highest recognition performance for unknown patterns. Moreover, interestingly, the DMD-based system performed much better over the unknown sets than did the LVQ system with several templates, while the LVQ system performed best over the design set. This result demonstrates that DMD contributes toward increasing the robustness through a suitable design of the class-feature space rather than an assignment of many templates in each class.

Table 1: Averages of recognition error rates

	design set	unknown set
DMD (init:Euclid)	2.86%	7.60%
DMD (init:Mahalanobis)	3.27%	8.09%
Euclidean distance	13.54%	13.37%
Mahalanobis distance	4.41%	8.63%
LVQ (1 template)	5.47%	8.05%
LVQ (8 templates)	1.613%	9.79%
LVQ (16 templates)	0.726%	10.74%

Table 2: Standard deviations of recognition error rates

	design set	unknown set
DMD (init:Euclid)	0.766%	0.679%
DMD (init:Mahalanobis)	1.049%	0.353%
Euclidean distance	1.909%	0.741%
Mahalanobis distance	1.106%	0.775%
LVQ (1 template)	1.352%	0.250%
LVQ (8 templates)	0.634%	0.702%
LVQ (16 templates)	0.289%	0.586%

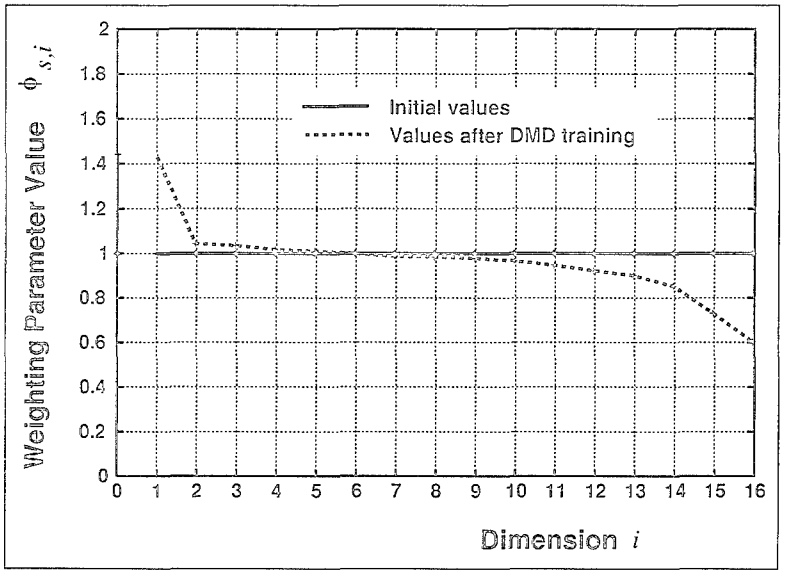


Figure 2: Values of the weighting parameters (singular values) for the cla ss /a/

Next we show how the values of weighting parameters $\{\varphi_{s,i}\}$ (i.e. the singular values of $\boldsymbol{W}_s$) were changed by the DMD. Figure 2 illustrates the initial values and the values after the DMD training for the vowel class /a/. The results were almost the same for the other vowel classes. In the graph, the horizontal axis denotes the dimension index i and the vertical axis stands for the weighting parameter value at each dimension. Here the metrics of all discriminant functions were initialized by the Euclidean distance.

This figure shows that DMD yields the large weighting values and the small ones (and their corresponding rotated feature axes) after training. DMD suggests that the axes with large weight values are related to the relevant features for recognition decision whereas those with small weight values correspond to the irrelevant features.

5 Relationships Between DMD and Other Techniques

The DMD implementation for the quadratic discriminant function has important implications for other recognizer design techniques.

Performing the well-known Principal Component Analysis (PCA) in each class can be a simple solution for finding each metric. In PCA, the eigenvectors associated with the large eigenvalues of the sample covariance matrix represent the intra-class statistical variation factors. To reduce the influence of such variation factors on recognition decisions, in other words, to normalize this type of variation, each weighting parameter $\varphi_{s,i}$ is usually set to the value of the parameter that is inversely proportional to the i-th eigenvalue; This leads to a commonly-used Mahalanobis distance classifier whose metric is specified as (10)-(12). This PCA-based design, however, estimates the parameters of each class independently and does not consider the influence of different classes; this does not necessarily reduce the recognition errors. This insufficiency has been demonstrated in the experimental results above.

Recently, demonstrations have been made of continuous Gaussian HMM speech recognizers based on MCE/GPD [5], which have achieved highly accurate recognition results. In most of these applications, diagonal covariance matrices were used: the original GPD adjustment rule was applicable only to this type of simple form matrix. In contrast, the DMD adjustment rule enables the full adaptation of full covariance matrices; this will improve the recognition performance compared to usual mixture Gaussian HMMs with diagonal matrices which essentially correspond to multi-template classifiers using a limited, simplified distance measure.

It is obvious that the continuous HMM recognizer is a general version of the RBF recognizer and the Likelihood Network recognizer [3]. Therefore, DMD also enables the full adjustment of these types of Neural

Network-based systems.

The feature transformation considered in this paper can be viewed as a feature extraction process. This point reminds us of the close relation between DMD and the DFE that jointly optimizes both the feature extraction and classification processes for the purpose of minimum error [1]. It is actually obvious that DFE can be considered to be a special case of DMD. The difference between these two is that DFE uses a common feature space over all of the classes while DMD designs an individual feature space for each class.

DMD is also related to the Subspace Method in the sense that each class possesses its own feature space [6]. Indeed, the subspace method can also be formalized as a special implementation of DMD. For example, the CLAFIC method [6], which designs each class subspace by Karhunen-Loève expansion, can be expressed through the linear class-feature transformation operator $\mathcal{T}_s$ and the Euclidean distance-based membership measure f_s $(s = 1, 2, ..., K)$ as

$$y_s = \Phi_s V_s^T x, \tag{13}$$

$$\Phi_s = \mathrm{diag}(\underbrace{1\ 1\ ...\ 1}_{d-p_s \text{ elements}}\ \ 0\ ...\ 0\ ), \tag{14}$$

$$V_s = [\ v_{s,d}\ v_{s,d-1}\ ...\ v_{s,2}\ v_{s,1}\ ], \tag{15}$$

$$f_s = ||y_s||^2, \tag{16}$$

where p_s denotes the dimensionality of the s-th class subspace and each $v_{s,i}$ stands for the eigenvector of the sample correlation (not covariance) matrix Q_s:

$$Q_s v_{s,i} = \lambda_{s,i} v_{s,i}\ (i = 1, 2, ..., d) \tag{17}$$

$$(\lambda_{s,1} \geq \lambda_{s,2} \geq ... \geq \lambda_{s,d}), \tag{18}$$

and each class-reference vector r_s is set to zero. Furthermore, the Learning Subspace Method (LSM) proposed by Kohonen [6], which improves the discrimination power of CLAFIC by learning, can also be viewed as a restricted form of DMD in the sense that only V_s is adjusted while Φ_s and r_s are fixed. Thus, our DMD comprehends the subspace method and accordingly provides more flexible ways to design robust discriminant functions for accurate recognition since not only subspaces V_s but also weighting parameters Φ_s (contributions to their corresponding feature axes) and class templates r_s can be adjusted for the minimum-error purpose. Moreover, DMD has the potential to discover the dimensionality of each class that is essential for recognition through the adjustment of the weighting parameters $\{\varphi_{s,i}\}_{i=1}^{d}$ while LSM determines each dimensionality at the initialization phase and fixes it during the learning phase.

6 Conclusion

This paper proposed a novel approach to pattern recognition, named Discriminative Metric Design (DMD), which fully designs the metric of each class discriminant function in a manner consistent with recognition error minimization. The experimental results in a vowel recognition task clearly demonstrated its high utility. Moreover, a comparison study of the relationships between DMD and several other recognition methods provided quite a useful basis for future theoretical analysis and a clear perspective on feature representation. It is lastly worth noting that one can easily apply the DMD formulation presented in this paper to *dynamic* (variable-durational) patterns by using a state transition structure like an HMM.

References

[1] A. Biem and S. Katagiri, "Feature extraction based on minimum classification error/ generalized probabilistic descent method", *Proc. ICASSP 93,* vol. 2, pp. 275-278, Apr., 1993.

[2] B.-H. Juang and S. Katagiri, "Discriminative learning for minimum error classification", *IEEE Trans. Signal Processing,* vol. 40, No. 12, pp. 3043-3054, Dec. 1992.

[3] S. Katagiri, C.-H. Lee, and B.-H. Juang, "Discriminative Multi-Layer Feed-Forward Networks", in *Proc. 1991 IEEE Workshop on Neural Networks for Signal Processing,* pp. 11-20, Princeton, NJ, Sept. 1991.

[4] S. Katagiri, C.-H. Lee, and B.-H. Juang, "New discriminative training algorithms based on the generalized probabilistic descent method", in *Proc. 1991 IEEE Workshop on Neural Networks for Signal Processing,* pp. 299-308, Princeton, NJ, Sept. 1991.

[5] W. Chou, B.-H. Juang, and C.-H. Lee, "Segmental GPD training of HMM based speech recognizer", *Proc. ICASSP 92,* vol. 1, pp. 473-476, 1992.

[6] E. Oja, *Subspace Methods of Pattern Recognition*, Research Studies Press, 1983.

A MAXIMUM ENTROPY APPROACH FOR OPTIMAL STATISTICAL CLASSIFICATION *

David Miller, Ajit Rao, Kenneth Rose, and Allen Gersho
Center for Information Processing Research
Department of Electrical and Computer Engineering
University of California, Santa Barbara, CA 93106
e-mail:rose@ece.ucsb.edu

Abstract

A global optimization technique is introduced for statistical classifier design to minimize the probability of classification error. The method, which is based on ideas from information theory and analogies to statistical physics, is inherently probabilistic. During the design phase, data are assigned to classes *in probability*, with the probability distributions chosen to maximize entropy subject to a constraint on the expected classification error. This entropy maximization problem is seen to be equivalent to a free energy minimization, motivating a deterministic annealing approach to minimize the misclassification cost. Our method is applicable to a variety of classifier structures, including nearest prototype, radial basis function, and multilayer perceptron-based classifiers. On standard benchmark examples, the method applied to nearest prototype classifier design achieves performance improvements over both the learning vector quantizer, as well as over multilayer perceptron classifiers designed by the standard back-propagation algorithm. Remarkably substantial performance gains over learning vector quantization are achieved for complicated mixture examples where there is significant class overlap.

*This work was supported in part by the National Science Foundation under grant no. NCR-9314335, the University of California MICRO program, Rockwell International Corporation, Hughes Aircraft Company, Echo Speech Corporation, Signal Technology Inc., Lockheed Missile and Space Company, and Qualcomm, Inc.

0-7803-2739-X/95 $4.00 © 1995 IEEE

1 Introduction

In recent years, the tremendous growth in neural networks research has stimulated renewed interest in statistical classification. Structures such as the multilayer perceptron (MLP) have the capability of implementing complex decision boundaries, and have been demonstrated to perform well in comparison with conventional classifiers, both for engineering applications such as speech recognition [8], as well as in the context of scientific inquiry [14]. However, several researchers have observed that MLPs and other structures trained to minimize the distance to output classification levels ($\{0, 1\}$ for the two-class case) do not directly minimize the classification error rate. Instead, these networks approximate the Bayes-optimal discriminant function, or equivalently the *a posteriori* probabilities that observations belong to a given class, e.g. [13]. (Similar observations have been made for linear classifiers [2]). Clearly, very large networks may be able, in principle, to accurately implement the Bayes rule, and thus provide minimum classification error. However, practical classifiers have restricted size to avoid high complexity and overfitting of limited training data. Thus, in practice, approximating the optimal discriminant function may result in significantly greater classification error than alternative solutions.

Rather than choosing to approximate the discriminant function, a number of researchers have proposed alternative cost objectives and learning algorithms which better match the goal of minimizing misclassification error (or minizing risk, if errors are not weighed equally), e.g. [7],[4],[6],[11]. Typically, these methods descend on an energy surface, using either a batch or a sequential optimization technique. While these approaches optimize MLPs and other network models to effectively minimize classification error, a legitimate concern is the potential to fall into poor local minimum traps, which often riddle the energy surface. In fact, the problem of local optima in neural networks has been acknowledged in a number of papers, e.g. [14]. While some smart heuristics have been employed for initializing parameters, typically one is forced to generate solutions based on a large number of random initializations, and then choose the best result.

We propose a new deterministic learning algorithm for statistical classifier design with a demonstrated potential for avoiding local optima of the cost. Several deterministic, annealing-based techniques have been proposed for avoiding nonglobal optima in computer vision [18],[3], combinatorial optimization [1], and elsewhere. Our approach is derived based on ideas from information theory and statistical physics, and builds on the framework of the deterministic annealing (DA) approach to clustering and related problems [16][15][17]. DA's probabilistic framework for clustering was derived by applying the maximum entropy principle to determine the underlying distributions. In recent work [9], we have shown that the maximum entropy approach unifies a larger class of optimization methods than was originally

conceived, and moreover, can be used to develop new, effective optimization methods for a number of challenging problems in source coding and statistics [12]. The maximum entropy formulation is useful because it precisely characterizes the annealing process in these methods as a gradual lowering of both the entropy and cost of the system with decreasing "temperature", where the temperature parameter is a Lagrange multiplier used to control the level of average cost.

Here, this DA approach is extended to minimize the cost of misclassification. We thus provide an approach for designing statistical classifiers based on training data which avoids many local minima that trap other known methods. In the next section, the method is derived and interpreted, and then we present some simulation comparisons with classifiers designed using conventional techniques. While in the sequel we assume a nearest prototype classifier structure for concreteness, we emphasize here that our approach is, in fact, generally applicable to optimize a variety of classification structures, including MLPs. A more general derivation and results for other structures can be found in [10].

2 Derivation and Algorithm

Let $\mathcal{T} = \{(\mathbf{x}, c)\}$ be a training set of N labelled vectors, where $\mathbf{x} \in \mathcal{R}^n$ is a feature vector and $c \in \mathcal{I}$ is its class label from an index set $\mathcal{I}$. A classifier is a mapping $C : \mathcal{R}^n \rightarrow \mathcal{I}$, which assigns a class label in $\mathcal{I}$ to each vector in $\mathcal{R}^n$. A training pair $(\mathbf{x}, c) \in \mathcal{T}$ is *misclassified* if $C(\mathbf{x}) \neq c$. The performance measure of the classifier is the probability of error, i.e. the fraction of the training set that it misclassifies. Our ultimate objective is to design a classifier to minimize this cost. In this paper, we restrict C to be a nearest prototype classifier, representable by a set of vectors $\{\mathbf{x}_{jk}\} \subset \mathcal{R}^n$, where $\mathbf{x}_{jk}$ is the kth prototype associated with class $j \in \mathcal{I}$. The classifier maps a vector in $\mathcal{R}^n$ to the class associated with the nearest prototype, defining a partition of $\mathcal{R}^n$ into regions $R_j \equiv \bigcup_k C_{jk}$ where $C_{jk} \equiv \{\mathbf{x} \in \mathcal{R}^n : d(\mathbf{x}, \mathbf{x}_{jk}) \leq d(\mathbf{x}, \mathbf{x}_{lm}) \quad \forall l, m\}$. Here, $d(\cdot, \cdot)$ is the "distance measure" used for classification.

Due to the challenging nature of the classifier design problem, we adopt a DA approach for its solution. Unlike simulated annealing, which implements a sequence of stochastic "moves" on the cost surface, DA is a deterministic learning algorithm that replaces stochastic operations with expectations over the distribution. Accordingly, we cast the problem in a probabilistic framework and consider a "random" classifier characterized by a probabilistic assignment of features to classes. We define the *probability of association* between a feature $\mathbf{x}$ and subregion C_{jk}, $P[\mathbf{x} \in C_{jk}]$, as well as the probability of association with a class region, $P[\mathbf{x} \in R_j] \equiv \sum_k P[\mathbf{x} \in C_{jk}]$. As our method will optimize over these probabilities in choosing the classifier,

the distributions must be consistent with the formation of a nearest prototype classification rule. This structural restriction may be enforced via a well-chosen parametrization of the distribution. An appropriate choice is the Gibbs distribution,

$$P[\mathbf{x} \in C_{jk}] = \frac{e^{-\gamma d(\mathbf{x}, \mathbf{x}_{jk})}}{\sum_{l,m} e^{-\gamma d(\mathbf{x}, \mathbf{x}_{lm})}}, \tag{1}$$

which depends on the prototype vectors and on a scale parameter γ which controls the fuzziness of the distribution. As $\gamma \rightarrow \infty$, the association probabilities revert to hard classifications equivalent to application of the nearest prototype rule. Note that this choice can be directly obtained using the principle of maximum entropy, which provides stronger justification for the resulting optimization method [10]. However, for conciseness we omit this derivation.

In our approach, we simultaneously control the probability of error and the randomness of the classifier. We start with a highly random classifier with a high expected classification error rate and then gradually reduce both the randomness and the expected probability of error. A natural measure of randomness is Shannon's entropy. In fact, Jaynes [5] showed that while there may be infinitely many distributions which satisfy expected value constraints, the least biased distribution is that which maximizes entropy. For the classification problem, the expected value of interest is the average classification error $< P_e >$. Thus, the maximum entropy distribution $\{P[\mathbf{x} \in C_{jk}]\}$ associated with the classification problem is obtained by solving

$$\max_{\{\mathbf{x}_{jk}\},\gamma} H \equiv \max_{\{\mathbf{x}_{jk}\},\gamma} \{-\frac{1}{N} \sum_{(\mathbf{x},c)\in\mathcal{T}} \sum_{j} P[\mathbf{x} \in R_j] \log P[\mathbf{x} \in R_j]\} \tag{2}$$

subject to

$$< P_e >= \frac{1}{N} \sum_{(\mathbf{x},c)\in\mathcal{T}} \sum_{j} P[\mathbf{x} \in R_j] \rho(c, j).$$

Here the cost of misclassification is $\rho(c, j) = 1$ if $c \neq j$ and 0 otherwise. Effectively, entropy maximization over the distribution is achieved through optimization over its parameter set. Solving this problem is equivalent to solving the unconstrained minimization of the Lagrangian:

$$\min_{\{\mathbf{x}_{jk}\},\gamma} L \equiv \min_{\{\mathbf{x}_{jk}\},\gamma} \beta < P_e > -H, \tag{3}$$

where β is the Lagrange multiplier used to enforce a constraint on $< P_e >$. For $\beta = 0$, the sole objective is entropy maximization, which is achieved by the uniform distribution, choosing the prototype vectors to be non-distinct. For $\beta \rightarrow \infty$, minimizing L is equivalent to minimizing the probability of error P_e,

leading to a non-random (i.e. $H \rightarrow 0$) classifier. This solution can be obtained within our probabilistic framework by choosing all prototype vectors to be distinct and sending $\gamma \rightarrow \infty$. Thus, we observe that an annealing approach is naturally obtained by minimizing the Lagrangian starting from $\beta = 0$ and tracking the solution while increasing β towards infinity. In this way, we obtain a sequence of solutions of decreasing entropy and cost, leading to a "hard" classifier at the limit. The annealing process can avoid local optima of the cost, and is motivated by the physical interpretation of the Lagrangian as a Helmholtz free energy [9]. We can rewrite the Lagrangian explicitly as:

$$L = \frac{1}{N} \sum_{(\mathbf{x},c)\in\mathcal{T}} \sum_{j} P[\mathbf{x} \in R_j] \left(\beta\rho(c,j) + \log P[\mathbf{x} \in R_j]\right) \tag{4}$$

$$\equiv \frac{1}{N} \sum_{(\mathbf{x},c)\in\mathcal{T}} \left(\sum_{j} P[\mathbf{x} \in R_j] L_{xj}\right) \equiv \frac{1}{N} \sum_{(\mathbf{x},c)\in\mathcal{T}} L_x.$$

Here, parentheses identify L_{xj}, the contribution to the cost when the feature $\mathbf{x}$ is assigned to class j, and L_x, the average contribution for this feature. The necessary conditions for minimizing L at any β are :

$$\frac{\partial L}{\partial \mathbf{x}_{jk}} = \frac{2\gamma}{N} \sum_{(\mathbf{x},c)\in\mathcal{T}} (L_{xj} - L_x) P[\mathbf{x} \in C_{jk}] \frac{\partial d(\mathbf{x}, \mathbf{x}_{jk})}{\partial \mathbf{x}_{jk}} = 0, \quad \forall j,k \tag{5}$$

and

$$\frac{\partial L}{\partial \gamma} = \frac{1}{N} \sum_{(\mathbf{x},c)\in\mathcal{T}} \sum_{j} L_{xj} (P[\mathbf{x} \in R_j] v_x - v_{xj}) = 0. \tag{6}$$

Here v_x is the average distance from $\mathbf{x}$ to a prototype, i.e. $v_x = \sum_j \sum_k P[\mathbf{x} \in C_{jk}] d(\mathbf{x}, \mathbf{x}_{jk})$, and v_{xj} is the contribution to this average from the prototypes of R_j, i.e. $v_{xj} = \sum_k P[\mathbf{x} \in C_{jk}] d(\mathbf{x}, \mathbf{x}_{jk})$.

These conditions can be interpreted, appropriately, within the context of supervised learning. The condition for a prototype vector suggests moving it away from (towards) vectors that it "owns" probabilistically through $P[\mathbf{x} \in C_{jk}]$ and for which the cost L_{xj} incurred by classifying to region R_j is greater than (less than) the average cost. The optimality condition for the scale parameter γ leads to a similar interpretation. Essentially, γ is either increased to solidify ownership of a point by a region if the cost is small, or is decreased to weaken ownership of a point if the cost is large. The optimization at each β can be implemented by gradient descent or any other function minimization technique. For $\beta \rightarrow \infty$, $H \rightarrow 0$ and $< P_e > \rightarrow P_e$. At this limit, our method terminates satisfying the necessary optimality conditions.

3 Results

We have performed experimental comparisons of our nearest-prototype method with the learning vector quantizer (LVQ) [7]. As an example, consider the two-class data of Figure 4. Each class consists of a Gaussian mixture with three components. We designed prototype-based classifiers with three prototypes per class, using both the LVQ and DA optimization methods. LVQ solutions were generated using the public domain LVQ-pak software, running both an optimized LVQ (OLVQ) learning phase, as well as a fine-tuning phase with 500,000 iterations. The learning parameter μ was set to 0.03. Ten LVQ solutions were generated based on random initialization and in all cases the method was unable to discriminate the class 0 "minority" component in the upper right of Figure 4a (which retains only 5 % of the training set mass). Apparently, the initialization did not select a prototype from the class 0 minority component, and LVQ is unable to move class 0 prototypes through the "wall" of class 1 data which separates them from this component. The best LVQ solution, which is shown in Figure 4a, achieved $P_e = 7.7\%$. Increasing the number of prototypes, we found that LVQ was only able to discriminate the minority component when 21 prototypes per class were introduced, and in this case the method achieved $P_e = 3.4\%$. The extremity of this suboptimality does suggest that the LVQ-pak initialization could be improved. For example, if an initialization of prototypes based on Isodata clustering followed by allocation of prototypes to the majority class of the cluster were used, much fewer than 42 prototypes (but greater than six) would suffice to find good solutions. However, this example does demonstrate LVQ's susceptibility to finding poor solutions. In fact, we also performed gradient descent on $< P_e >$ and found that poor solutions were obtained in this case as well – excepting omnisicient initialization in the vicinity of the optimal solution, the best performance obtained for six prototypes was $P_e = 7.0\%$. It thus appears that strict descent methods will fail on this example unless given an excellent initialization. By contrast, the DA method using only five prototypes achieved the solution shown in Figure 1b, with $P_e = 2.7\%$. Note that the DA method is independent of the initialization, placing all prototypes together at the global data centroid (marked by X) at $\beta = 0$ so as to maximize entropy. (Such an initialization is in fact "fatal" for a strict descent-based approach, as the associated learning rule will not permit a class 0 prototype to pass through the "wall" of class 1 data.) Then, as β is increased, the prototypes separate, reducing the entropy as well as $< P_e >$. This example demonstrates the ability of the method to avoid local optima, since the DA optimization does succeed in moving a class 0 prototype from X directly *through* the class 1 data "wall" to correctly classify the minority class 0 component and achieve what appears to be the optimal piece-wise linear result. (Here, two of the class 0 prototypes are non-distinct, so the solution effectively uses five prototypes.)

In addition to this example, we have tested our approach on the "syn-

thetic" example from [14], as well as on some other complicated synthetically generated mixture examples. On the example from [14], our approach achieved $P_e = 8.9\%$ on the test set using eight prototypes and $P_e = 8.6\%$ using twelve prototypes, in comparison to LVQ's $P_e = 9.5\%$ based on twelve prototypes. For general reference, an MLP with six hidden units achieved $P_e = 9.4\%$. For complicated mixture examples, with possibly twenty or more overlapping mixture components and multiple classes, we have found our method to consistently achieve substantial peformance gains over LVQ. As an example, we generated training data for a four-class problem involving twenty-four overlapping, non-isotropic mixture components in two dimensions. We designed nearest prototype classifiers with 16 prototypes (four per class) using both LVQ and DA. The best LVQ solution based on ten random initializations achieved $P_e = 31\%$. By contrast the single DA solution achieved $P_e = 23\%$. This comparison is typical of what we have seen through extensive experimentation. Similar performance gains are achieved for higher-dimensional data sets, but we have restriced these examples to two dimensions for visual illustration. While for certain problems other structures such as MLPs or RBFs may be inherently superior to the prototype-based structure discussed here, our results demonstrate the potential of the optimization technique. Moreover, as we describe in [10], our method achieves similar performance gains in optimizing the RBF and MLP classifier structures.

References

[1] G. L. Bilbro, W. E. Snyder, and R. C. Mann. Mean-field approximation minimizes relative entropy. *Journal of the Opt. Soc. of Amer.*, 8:290–294, 1991.

[2] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis.* Wiley-Interscience, New York, NY, 1974.

[3] D. Geiger and F. Girosi. Parallel and deterministic algorithms from MRFs: Surface reconstruction. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 13:401–412, 1991.

[4] J. B. Hampshire and A. H. Waibel. A novel objective function for improved phoneme recognition using time-delay neural networks. *IEEE Trans. on Neural Net.*, 1:216–228, 1990.

[5] E. T. Jaynes. Information theory and statistical mechanics. In R. D. Rosenkrantz, editor, *Papers on probability, statistics and statistical physics.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1989. (Reprint of the original 1957 papers in *Physical Review*).

[6] B.-H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Trans. on Sig. Proc.*, 40:3043–3054, 1992.

[7] T. Kohonen, G. Barna, and R. Chrisley. Statistical pattern recognition with neural networks: Benchmarking studies. In *IEEE Proc. ICNN*, volume 1, pages 61–68, 1988.

[8] R. Lippmann. Review of neural networks for speech recognition. *Neural Comp.*, 1:1–39, 1989.

[9] D. Miller, A. Rao, K. Rose, and A. Gersho. A maximum entropy framework for optimization with application to supervised learning. (Submitted for publication.), 1994.

[10] D. Miller, A. Rao, K. Rose, and A. Gersho. An information-theoretic framework for optimal statistical classification. (Submitted for publication.), 1995.

[11] V. Nedeljkovic. A novel multilayer neural networks training algorithm that minimizes the probability of classification error. *IEEE Trans. on Neural Net.*, 4:650–659, 1993.

[12] A. Rao, D. Miller, K. Rose, and A. Gersho. Generalized vector quantization. (In preparation.), 1994.

[13] M. D. Richard and R. P. Lippmann. Neural network classifiers estimate Bayesian a posteriori probabilities. *Neural Comp.*, 3:461–483, 1991.

[14] B. D. Ripley. Neural networks and related methods for classification. *Journal of the Royal Stat. Soc., Ser. B*, 56:409–456, 1994.

[15] K. Rose, E. Gurewitz, and G. C. Fox. Statistical mechanics and phase transitions in clustering. *Phys. Rev. Lett.*, 65:945–948, 1990.

[16] K. Rose, E. Gurewitz, and G. C. Fox. Vector quantization by deterministic annealing. *IEEE Trans. on Inform. Theory*, 38:1249–1258, 1992.

[17] K. Rose, E. Gurewitz, and G. C. Fox. Constrained clustering as an optimization method. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 15:785–794, 1993.

[18] A. L. Yuille. Generalized deformable models, statistical physics, and matching problems. *Neural Comp.*, 2:1–24, 1990.

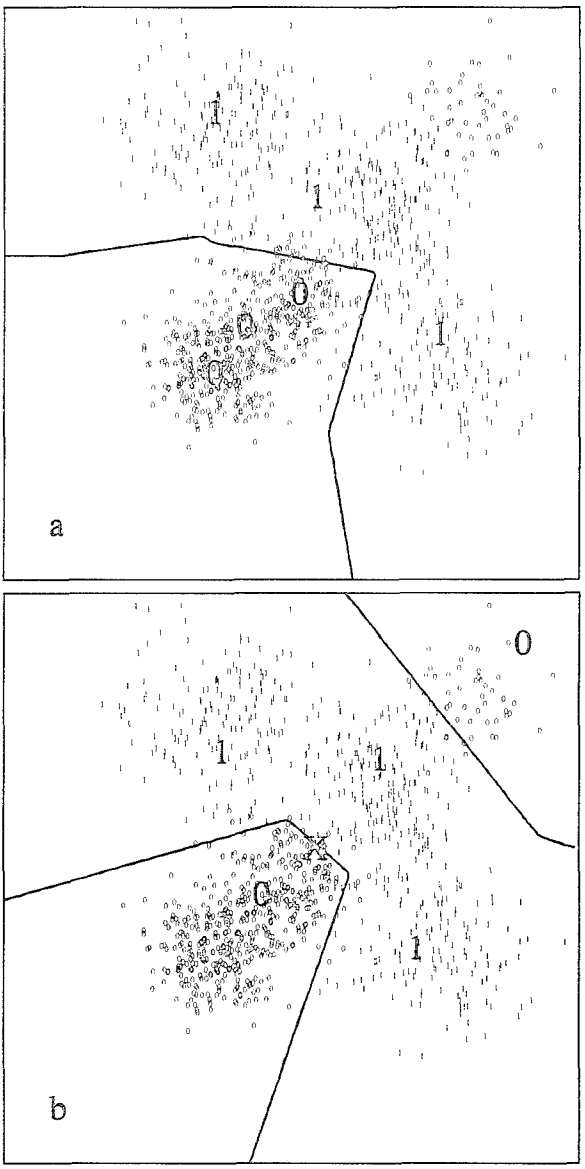


Figure 1: A two-class example, with a 3-component Gaussian mixture in each class: a) The LVQ solution, using three prototypes per class, with $P_e = 7.7\%$. b) The DA solution, using three prototypes per class, with $P_e = 2.7\%$. Note that since the solution at $\beta = 0$ placed all prototypes at the global centroid (X), the DA optimization has allowed a prototype for class 0 to "pass through a wall" of class 1 data in order to correctly classify the minority "0" mixture component.

SIMULTANEOUS DESIGN OF FEATURE EXTRACTOR AND PATTERN CLASSIFIER USING THE MINIMUM CLASSIFICATION ERROR TRAINING ALGORITHM

K.K. Paliwal [1], M. Bacchiani and Y. Sagisaka
ATR Interpreting Telecommunications Res. Labs.
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

ABSTRACT — Recently, a minimum classification error training algorithm has been proposed for minimizing the misclassification probability based on a given set of training samples using a generalized probabilistic descent method. This algorithm is a type of discriminative learning algorithm, but it approaches the objective of minimum classification error in a more direct manner than the conventional discriminative training algorithms. We apply this algorithm for simultaneous design of feature extractor and pattern classifier, and demonstrate some of its properties and advantages.

1. INTRODUCTION

Juang and Katagiri [1] have recently proposed a minimum classification error training algorithm which minimizes the misclassification probability based on a given set of training samples using a generalized probabilistic descent method. This algorithm is a type of discriminative learning algorithm, but it approaches the objective of minimum classification error in a more direct manner than the conventional discriminative training algorithms [2]. Because of this, it has been used in a number of pattern classification applications [3, 4, 5, 6, 7, 8]. For example, Chang et al. [3] and Komori and Katagiri [4] have used this algorithm for designing the pattern classifier for dynamic time-warping based speech recognition, Chou et al. [5] and Rainton and Sagayama [6] for hidden Markov model (HMM) based speech recognition, Sukkar and Wilpon [7] for word spotting, and Liu et al. [8] for HMM-based speaker recognition. More recently, the minimum classification error training algorithm has been used for feature extraction [9, 10, 11, 12]. For example, Biem and Katagiri have used it for determining the parameters of a cepstral lifter [9] and a filter bank [10]. They have found that the resulting parameters of cepstral lifter and filter bank have a good physical interpretation. Bacchiani and Aikawa [11] have used this algorithm for designing a dynamic cepstral filter. Watanabe and Katagiri [12] have used a class-dependent unitary transformation for feature extraction whose parameters are determined by the minimum classification error training algorithm.

[1]Present Address: School of Microelectronic Engineering, Griffith University, Brisbane, QLD 4111, Australia

0-7803-2739-X/95 $4.00 © 1995 IEEE

In the present paper, we use the minimum classification error (MCE) training algorithm to design the feature extractor as well as the classifier. Consider a canonic pattern recognizer shown in Fig. 1. The aim of the pattern recognizer is to classify an input signal as one of the K classes. This is done in two steps: a feature analysis step and a pattern classification step. In the feature analysis step, the input signal is analyzed and D parameters are measured. The D-dimensional parameter vector X is processed by the feature extractor which produces at its output a feature vector Y of dimensionality $\leq D$. In our study, we use a $D \times D$ linear transformation T for feature extraction: i.e., $Y = TX$. The transformation T is a general linear transformation; i.e., it is not restricted, for example, to be unitary as done by Watanabe and Katagiri [12]. The feature vector Y is a D-dimensional vector in our study. In the pattern classification step, the feature vector is compared with each of the K class-models and a dissimilarity (or, distance) measure is computed for each class. The class that gives the minimum distance is considered to be the recognized class.

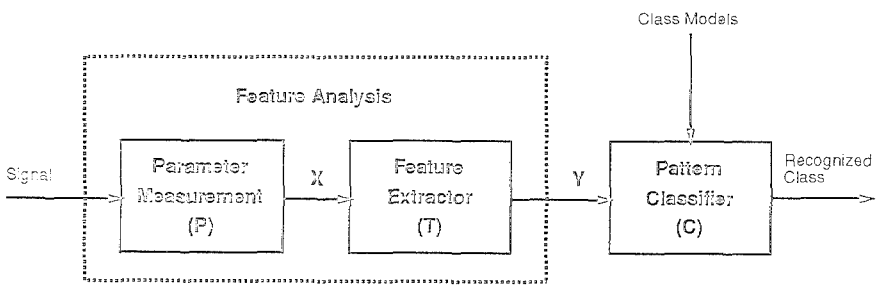


Figure 1: A pattern recognition system.

In the present paper, we study the minimum classification error training algorithm for the following configurations of feature extractor and classifier:

- **Configuration 1:** It is a baseline configuration where transformation T is unity and the class models are determined by the maximum-likelihood (ML) training algorithm (as the class-dependent means of the training vectors).
- **Configuration 2:** Here the transformation T is kept fixed to a unity matrix and the class models are computed using the MCE training algorithm.
- **Configuration 3:** Here the transformation T is computed by the MCE training algorithm and the class models are kept fixed to the baseline class models.
- **Configuration 4:** This configuration is similar to Configuration 3, except that transformation T is applied to the parameter vector as well

as to the class models of the baseline configuration prior to distance computation.

- **Configuration 5:** Here both the transformation T and the class models are computed independently using the MCE training algorithm.
- **Configuration 6:** This configuration is similar to Configuration 3, except that the transformation T is made class-dependent; i.e., we now have K different transformations, T_k, $k = 1, 2, \ldots, K$ for K different classes. We apply transformation T_k to parameter vector X before computing its distance from the k-th class.
- **Configuration 7:** This configuration is similar to Configuration 5, except that the transformation T is made class-dependent (similar to Configuration 6); i.e., we now have K different transformations, T_k, $k = 1, 2, \ldots, K$ for K different classes.

Note that Configurations 2,3,4 and 5 use a class-independent transformation as shown in Fig. 2; while Configurations 6 and 7 use a class-dependent transformation as shown in Fig. 3.

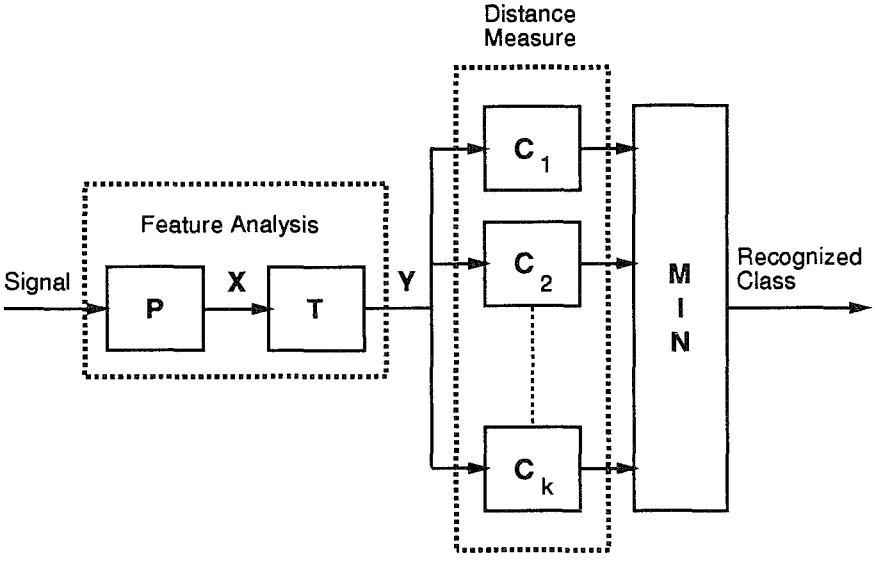


Figure 2: A pattern recognition system with class-independent transformation.

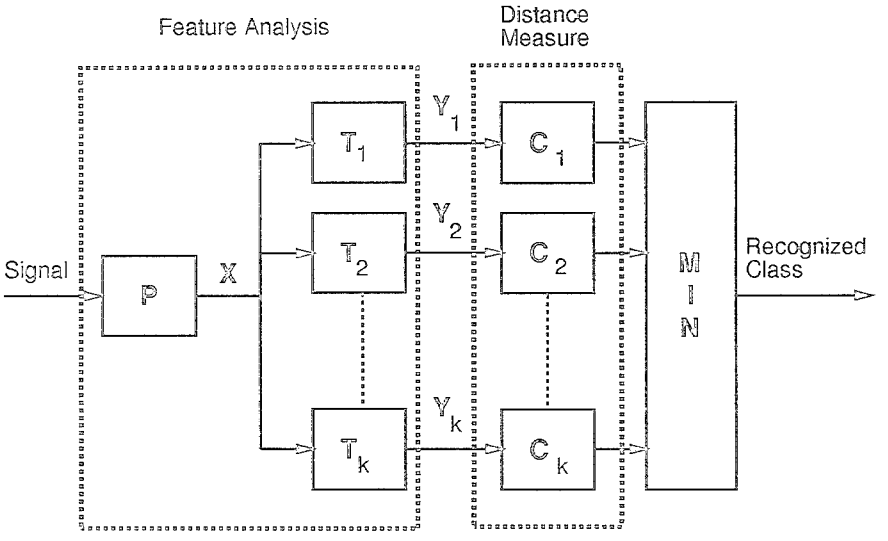


Figure 3: A pattern recognition system with class-dependent transformation.

2. MCE TRAINING ALGORITHM

In this section, we describe briefly the minimum classification error (MCE) training algorithm. For more details, see [1]. The MCE algorithm is described here only for Configuration 5. It can be extended to other configurations in a straightforward manner.

In the pattern recognition system shown in Fig. 1, the input parameter vector X is transformed to a feature vector Y $(= TX)$ and classified into class i if

$$D_i \leq D_j, \qquad \text{for all } j \neq i, \tag{1}$$

where D_i is the distance of feature vector Y from class i. In the present paper, we use a simple Euclidean distance measure to define this distance. It is given by

$$\begin{aligned} D_i &= \| Y - m^{(i)} \|^2 \\ &= \| TX - m^{(i)} \|^2, \end{aligned} \tag{2}$$

where $m^{(i)}$ is the prototype vector representing the class i.

Here, we are given a total of P labeled training vectors; i.e., we have P parameter vectors $X^{(1)}, X^{(2)}, \ldots, X^{(P)}$ available for training with corresponding classification $C^{(1)}, C^{(2)}, \ldots, C^{(P)}$ known to us. Our aim here is to use the MCE algorithm to estimate the transformation matrix T and class prototype vectors $m^{(1)}, m^{(2)}, \ldots, m^{(K)}$ using these labeled training vectors. The procedure for doing this is described below.

The distance of pth training vector from class i is given by

$$\begin{aligned} D_i^{(p)} &= \| Y^{(p)} - m^{(i)} \|^2 \\ &= \| TX^{(p)} - m^{(i)} \|^2 \\ &= \sum_{s=1}^{D} \left(\sum_{j=1}^{D} T_{sj} X_j^{(p)} - m_s^{(i)} \right)^2 . \end{aligned} \tag{3}$$

We use this distance to define the misclassification measure for the pth training vector as follows:

$$d^{(p)} = D_{C^{(p)}}^{(p)} - D_{N^{(p)}}^{(p)}, \tag{4}$$

where $D_{C^{(p)}}^{(p)}$ is the distance of pth training vector from its known class $C^{(p)}$ and the distance $D_{N^{(p)}}^{(p)}$ is computed from the relation

$$D_{N^{(p)}}^{(p)} = \underset{i, i \neq C^{(p)}}{\operatorname{argmin}} D_i^{(p)}. \tag{5}$$

The loss function $L^{(p)}$ for the pth training vector is then defined as the sigmoid of the misclassification measure as follows:

$$\begin{aligned} L^{(p)} &= f(d^{(p)}) \\ &= \frac{1}{1 + e^{-\alpha d^{(p)}}}, \end{aligned} \tag{6}$$

where α is a parameter defining the slope of the sigmoid function.

The total loss function L is defined as

$$L = \sum_{p=1}^{P} L^{(p)}. \tag{7}$$

In the MCE algorithm, the transformation matrix and class prototype vectors are obtained by minimizing this loss function through the steepest gradient descent algorithm. This is an iterative algorithm where parameters at the $(k+1)$th iteration are computed from the kth iteration results as follows:

$$T_{sj}(k+1) = T_{sj}(k) - \eta \frac{\partial L}{\partial T_{sj}}, \tag{8}$$

$$m_s^{(N^{(p)})}(k+1) = m_s^{N^{(p)}}(k) - \eta \frac{\partial L}{\partial m_s^{(N^{(p)})}}, \tag{9}$$

and

$$m_s^{(C^{(p)})}(k+1) = m_s^{C^{(p)}}(k) - \eta \frac{\partial L}{\partial m_s^{(C^{(p)})}}, \tag{10}$$

where η is a positive constant (known as the adaptation constant) and

$$\frac{\partial L}{\partial T_{sj}} = 2\alpha \sum_{p=1}^{P} f(d^{(p)})(1 - f(d^{(p)}))(m_s^{(N^{(p)})} - m_s^{(C^{(p)})})X_j^{(p)}. \tag{11}$$

$$\frac{\partial L}{\partial m_s^{(N^{(p)})}} = 2\alpha \sum_{p=1}^{P} f(d^{(p)})(1 - f(d^{(p)})) \left(\sum_{j=1}^{D} T_{sj} X_j^{(p)} - m_s^{(N^{(p)})} \right), \tag{12}$$

and

$$\frac{\partial L}{\partial m_s^{(C^{(p)})}} = -2\alpha \sum_{p=1}^{P} f(d^{(p)})(1 - f(d^{(p)})) \left(\sum_{j=1}^{D} T_{sj} X_j^{(p)} - m_s^{(C^{(p)})} \right). \tag{13}$$

For the initialization of the MCE algorithm, the transformation matrix T is taken to be a unity matrix. The prototype vectors for different classes are initialized by their maximum likelihood estimates (i.e., by their class-conditioned means).

3. RESULTS

The MCE algorithm is studied here on a multispeaker vowel recognition task. The Peterson-Barney vowel data base [13] is used for this purpose. Here each vowel is represented in terms of 4 parameters: fundamental frequency and frequencies of the first three formants. The data base consists of two repetitions of 10 vowels in /hVd/ context recorded from 76 speakers (33 men, 28 women and 15 children). Fundamental and formant frequencies were measured by Peterson and Barney from the central steady-state portions of the /hVd/ utterances. We use the first repetition for training and the second for testing. We use the Euclidean distance measure for classification of the 4-dimensional parameter vector into one of the 10 classes. The model for each class is determined as the mean vector of the training patterns of that class. In our implementation of the MCE training algorithm, we use the steepest gradient descent algorithm with the adaptation parameter updated every iteration using a fast-converging algorithm described by Lucke [14].

In order to study the convergence properties of this algorithm, we study it for Configuration 5. Figure 4 shows the the loss function, recognition error on training and test data as a function of iteration number. It can be seen from this figure that the loss function is decreasing with number of iterations and the algorithm is converging reasonably fast (within 500 iterations). Also, recognition results on test data are similar to those on training, showing the generalization property of the algorithm.

The MCE algorithm is studied for all the seven configurations. The results for different configurations are listed in Table 1. We list in column 2 of this table the total number of free parameters used in the transformation and the classifier. Column 3 of this table lists the number of parameters computed by the MCE training algorithm. The numbers shown within square brackets in columns 2 and 3 correspond to the vowel recognition task used in this study, where $K = 10$ and $D = 4$. From this table, we can make the following observations:

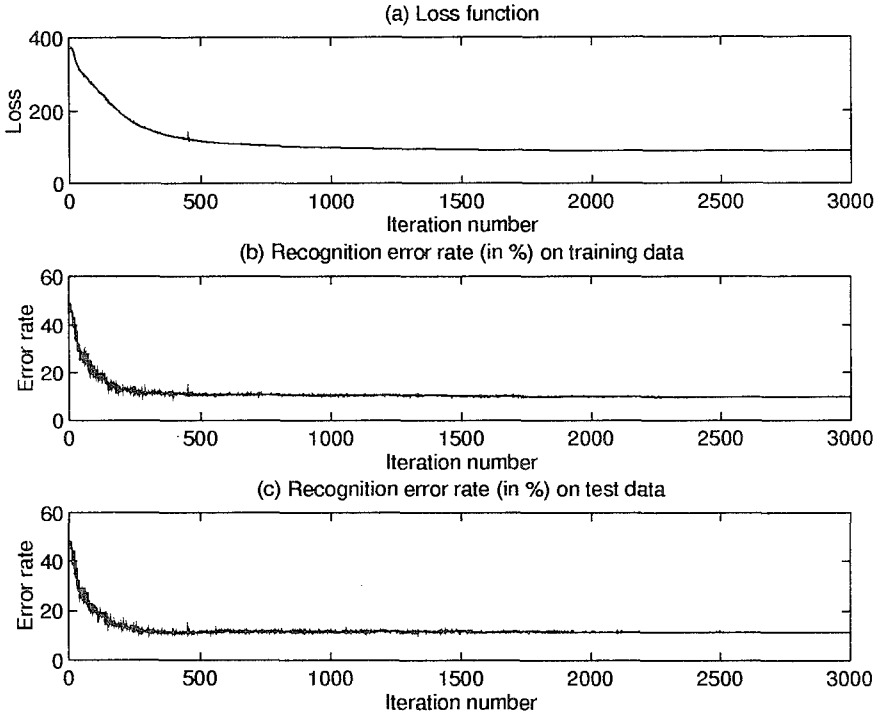


Figure 4: Results for Configuration 5 as a function of iteration number. (a) Loss function, (b) Recognition error rate (in %) on training data, and (c) Recognition error rate (in %) on test data.

1. The MCE training algorithm performs better than the ML algorithm (compare Configuration 1 with Configuration 2).
2. The recognition performance of the pattern recognizer improves with an increase in the total number of free parameters in the transformation and the classifier.
3. For a given number of free parameters, the recognition performance improves as the more number of parameters are updated by the MCE

Table 1: Recognition error rate (in %) for different configurations of feature extractors and classifiers studied using the minimum classification error (MCE) training algorithm.

Configuration	Total no. of free parameters	No. of parameters updated by MCE	Recognition error rate	
			Training	Test
Conf. 1	KD [40]	0 [0]	48.29	48.16
Conf. 2	KD [40]	KD [40]	34.47	36.18
Conf. 3	$D^2 + KD$ [56]	D^2 [16]	33.16	33.29
Conf. 4	$D^2 + KD$ [56]	D^2 [16]	13.95	16.32
Conf. 5	$D^2 + KD$ [56]	$D^2 + KD$ [56]	9.87	11.45
Conf. 6	$K(D^2 + D)$ [200]	KD^2 [160]	9.47	12.37
Conf. 7	$K(D^2 + D)$ [200]	$K(D^2 + D)$ [200]	9.34	12.76

training algorithm (compare Configuration 3 with Configuration 5 or Configuration 6 with Configuration 7).

4. Observe Configurations 3 and 4. Both of these configurations have same number of free parameters and same number of parameters are updated by the MCE training algorithm. But, Configuration 4 gives significantly better results than Configuration 3. This is because in Configuration 4 the transformation T is applied to the parameter vector X as well as the class models prior to distance computation.

5. Observe Configurations 5 and 7. Configuration 5 uses a class independent transformation, while Configuration 7 uses class-dependent transformations. Therefore, Configuration 7 shows better recognition performance than Configuration 5 on training data, though the difference in the recognition rates for the two configurations is small. However, note that recognition performance of Configuration 7 on test data is inferior to that of Configuration 5. This happens because the number of parameters updated by the MCE algorithm are too large for the limited amount of data available for training and, hence, the results do not generalize to test data properly.

4. SUMMARY

In this paper, we have studied the use of minimum classification error (MCE) training algorithm for the design of the feature extractor and the pattern classifier. We have investigated a number of configurations of the feature extractor and the pattern classifier, and have demonstrated a number of properties and advantages of the MCE training algorithm.

References

[1] B.H. Juang and S. Katagiri, "Discriminative learning for minimum error classification," *IEEE Trans. Signal Processing*, Vol. 40, No. 12, pp. 3043-3054, Dec. 1992.

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

[3] P.C. Chang, S.H. Chen and B.H. Juang, "Discriminative analysis of distortion sequences in speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1991, Vol. 1, pp. 549-552.

[4] T. Komori and S. Katagiri, "GPD training of dynamic programming-based speech recognizers," *Journal of Acoust. Soc. of Japan (E)*, Vol. 13, No. 6, pp. 341-349, Nov. 1992.

[5] W. Chou, B.H. Juang and C.H. Lee, "Segmental GPD training of HMM based speech recognizer," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, Mar. 1992, Vol. 1, pp. 473-476.

[6] D. Rainton and S. Sagayama, "Minimum error classification training of HMMs – Implementation details and experimental results," *Journal of Acoust. Soc. of Japan (E)*, Vol. 13, No. 6, pp. 379-387, Nov. 1992.

[7] R.A. Sukkar and J.G. Wilpon, "A two-pass classifier for utterance rejection in keyword spotting," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1993, Vol. 2, pp. 451-454.

[8] C.S. Liu, C.H. Lee, W. Chou, B.H. Juang and A.E. Rosenberg, "A study on minimum error discriminative training for speaker recognition," *Journal of Acoust. Soc. of Am.*, Vol. 97, No. 1, pp. 637-648, Jan. 1995.

[9] A. Biem and S. Katagiri, "Feature extraction based on minimum classification error/generalized probabilistic descent method," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1993, Vol. 2, pp. 275-278.

[10] A. Biem and S. Katagiri, "Filter bank design based on discriminative feature extraction," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1994, Vol. 1, pp. 485-488.

[11] M. Bacchiani and K. Aikawa, "Optimization of time-frequency masking filters using the minimum classification error criterion," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1994, Vol. 2, pp. 197-200.

[12] H. Watanabe, T. Yamaguchi and S. Katagiri, "Discriminative metric design for pattern recognition," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, 1995, Vol. 5, pp. 3439-3442.

[13] G. Peterson and H.L. Barney, "Control methods used in a study of the vowels," *Journal of Acoust. Soc. of Am.*, Vol. 24, pp. 175-184, 1952.

[14] H. Lucke, "On the representation of temporal data for connectionist word recognition," Ph.D. Thesis, Cambridge University, Nov. 1991.

Discriminative Subspace Method for Minimum Error Pattern Recognition

Hideyuki WATANABE and Shigeru KATAGIRI
ATR Interpreting Telecommunications Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan
Tel.: +81-774-95-1383
Fax.: +81-774-95-1308
E-mail: watanabe@itl.atr.co.jp

Abstract

Subspace Method (SM) is one of fundamental frameworks for pattern recognition. In particular, its discriminative learning version, called Learning Subspace Method (LSM), has been shown quite useful in various applications. However, this important design method leaves much room for further analysis due to the lack of a link between LSM and the ultimate goal of pattern recognition, i.e. the minimum error situation. In this light, we investigate in this paper SM from the viewpoint of the Minimum Classification Error/Generalized Probabilistic Descent method (MCE/GPD). Applying MCE/GPD to SM, we formalize a new discriminative subspace method, called the Minimum Error Learning Subspace method (MELS), which enables one to directly pursue the minimum error recognition. This paper also provides a rigorous analysis of the MELS's learning mechanism as well as a comparison between the conventional LSM and MELS.

1 Introduction

Subspace Method (SM), especially its discriminative learning version called Learning Subspace Method (LSM), has been shown quite useful in a wide range of pattern recognitions because of its computational simplicity and robustness to statistical pattern variations [1, 2, 3]. However, due to the lack of rigorous analysis from the viewpoint of the Bayes decision theory, this valuable recognizer design method still leaves much room for further mathematical investigation.

In the meantime, the Minimum Classification Error/Generalized Probabilistic Descent method (MCE/GPD) has provided new, general math-

0-7803-2739-X/95 $4.00 © 1995 IEEE

ematical bases for designing pattern recognizers aimed at minimizing recognition errors, or in other words, achieving the *optimal* minimum error situation [4]. Actually, it has been shown that the widely-used Learning Vector Quantization (LVQ) is a simple version of the MCE/GPD-based distance classifier [5]. The analogy between LVQ and LSM naturally suggests an analysis of LSM from the MCE/GPD viewpoint.

In this paper we investigate how the SM's discriminative power can be increased in the MCE/GPD framework. We also formalize a new discriminative subspace method, named *Minimum Error Learning Subspace method (MELS)*. This paper provides detailed analysis of the learning nature of MELS and proves that this proposed method leads to at least a locally optimal recognition situation. A comparison between the conventional LSM and MELS is also discussed.

2 Conventional Subspace Methods

2.1 Pattern Recognition by Subspace Method

We consider the problem of classifying a d-dimensional input pattern $x \in \mathcal{R}^d$ into one of the K classes $\{C_s\}_{s=1}^K$ using the following decision rule:

$$C(x) = C_i \quad \text{if} \quad i = \arg\max_s g_s(x), \tag{1}$$

where $g_s(x)$, called the *discriminant function*, represents the degree to which x belongs to C_s and $C(\cdot)$ is a recognition operation. In the SM framework, an individual *class subspace* in the d-dimensional pattern space is designed for each class, and each input pattern x is classified as the class giving the maximum value of the orthogonal projection of x. That is to say, the discriminant function of the s-th class ($s = 1, 2, ..., K$) is defined as

$$g_s(x; U_s) = ||P_{U_s} x||^2, \tag{2}$$

$$U_s = [\, u_{s,1}\ u_{s,2}\ ...\ u_{s,p_s}\,], \tag{3}$$

$$u_{s,i} = [\, u_{s,1,i}\ u_{s,2,i}\ ...\ u_{s,d,i}\,]^T \quad (i = 1, 2, ..., p_s), \tag{4}$$

where $||\cdot||$ denotes the Euclidean norm, U_s is the $d \times p_s$ ($p_s < d$) full-rank *base matrix* whose column vectors span the s-th class subspace $\mathcal{V}_s$, and the $d \times d$ matrix P_{U_s} is the *(orthogonal) projection matrix* onto $\mathcal{V}_s$ and is expressed as

$$P_{U_s} = U_s (U_s^T U_s)^{-1} U_s^T, \tag{5}$$

where the superscript ^{T} and $^{-1}$ denote the transpose and the inverse, respectively. Consequently, the discriminant function is rewritten as

$$g_s(x; U_s) = x^T U_s (U_s^T U_s)^{-1} U_s^T x, \tag{6}$$

and it turns out that the method of designing each class subspace $\{\mathcal{V}_s\}_{s=1}^K$, or base matrix $\{U_s\}_{s=1}^K$, determines the quality of recognition decision.

2.2 A Brief Review of LSM

The most fundamental algorithm for designing subspaces is the *CLAFIC* method, which designs each class subspace by using Karhunen-Loève expansion [1]. Since it designs each class subspace independently without considering the influences of other competing classes, it is not necessarily the case that the CLAFIC method can reduce the recognition errors efficiently. To alleviate this inadequacy, the *Learning Subspace Method (LSM)* was developed by Kohonen [1]. In LSM, each subspace is trained according to the recognition result of each design pattern so that the projection onto its true class gets larger while that onto each different competing class decreases.

Let us consider the subspace design problem using a labeled training sample set $\boldsymbol{\Omega} = \{\boldsymbol{x}_n\}$. Suppose that an input pattern $\boldsymbol{x}_t$ is selected randomly from $\boldsymbol{\Omega}$ at the t-th iteration. In LSM, if $\boldsymbol{x}_t$ belongs to the k-th class, each orthonormalized base matrix $\boldsymbol{U}_s$ is adjusted according to the following rule:

$$\tilde{\boldsymbol{U}}_k^{(t)} = (\boldsymbol{I} + \mu_k \boldsymbol{x}_t \boldsymbol{x}_t^T)\boldsymbol{U}_k^{(t-1)} \quad \text{(the true class)}, \tag{7}$$

$$\tilde{\boldsymbol{U}}_j^{(t)} = (\boldsymbol{I} - \mu_j \boldsymbol{x}_t \boldsymbol{x}_t^T)\boldsymbol{U}_j^{(t-1)} \quad \text{(the competing class)} \tag{8}$$

$$\left(j = \arg\max_{s, s \neq k} g_s(\boldsymbol{x}_t; \boldsymbol{U}_s^{(t-1)})\right),$$

$$\boldsymbol{U}_s^{(t)} = \tilde{\boldsymbol{U}}_s^{(t)} \boldsymbol{T}_s^{(t)} \quad (s = k, j), \tag{9}$$

$$\boldsymbol{U}_i^{(t)} = \boldsymbol{U}_i^{(t-1)} \quad (i = 1, 2, ..., K; i \neq k, j), \tag{10}$$

where $\boldsymbol{I}$ denotes the identity matrix, μ_k and μ_j are the positive real numbers called the learning coefficients, and the $p_s \times p_s$ matrix $\boldsymbol{T}_s^{(t)}$ represents the orthonormalization operator for the column vectors of $\tilde{\boldsymbol{U}}_s^{(t)}$. Detailed discussions of the properties of LSM are made in [1].

As mentioned in [1], the optimal (in the sense of minimum recognition errors) values of μ_k and μ_j are very difficult to find. Actually, LSM was vigorously investigated with regard to convergence properties. However, its capability to attain the minimum error situation has not yet been sufficiently analyzed. These values are thus usually specified heuristically, e.g. by performing certain preliminary experiments. Obviously, a mathematical method is needed to determine these learning coefficients so that one can directly persue the minimum recognition error probability by using the above formula.

3 Discriminative Subspace Method for Minimum Error Recognition

3.1 Formalization of MELS

The learning mechanism of MCE/GPD is based on gradient search optimization. The correction vector at each iteration step can be derived by differentiating the loss function in the recognizer parameters. Detailed discussions on MCE/GPD are made in [4, 6].

The first step in the MELS formulation is to derive the first-order derivative of each discriminant function. Its result is summarized in the following theorem.

Theorem 1 *The first-order derivative of* $g_s(\boldsymbol{x}; \boldsymbol{U}_s)$ *with respect to* $\boldsymbol{U}_s$ *is given as*

$$\nabla_{U_s} g_s(\boldsymbol{x}; \boldsymbol{U}_s) = 2P^{\perp}_{U_s} \boldsymbol{x}\boldsymbol{x}^T \boldsymbol{U}_s (\boldsymbol{U}_s^T \boldsymbol{U}_s)^{-1}, \tag{11}$$

where the matrix differentiation is defined as $\nabla_A = (\nabla_{a_{i,j}})$ $(A = (a_{i,j}))$, *and*

$$P^{\perp}_{U_s} = I - \boldsymbol{U}_s (\boldsymbol{U}_s^T \boldsymbol{U}_s)^{-1} \boldsymbol{U}_s^T, \tag{12}$$

which is an orthogonal projection operator onto the orthogonal complement of $\mathcal{V}_s = \text{range } \boldsymbol{U}_s$.

Proof. See Appendix A. □

Next, by applying the MCE/GPD adjustment rule to the above derivatives and operating the orthonormalization, an MCE/GPD-based subspace design algorithm, i.e. the MELS method, is formulated as follows:

The MELS method

At the t-th iteration, if a randomly-selected training pattern $\boldsymbol{x}_t$ belongs to the k-th class, then

$$\tilde{\boldsymbol{U}}_s^{(t)} = \boldsymbol{U}_s^{(t-1)} - \varepsilon_t \nabla_{U_s} \ell_k(\boldsymbol{x}_t; \Lambda^{(t-1)}), \tag{13}$$

$$\boldsymbol{U}_s^{(t)} = \tilde{\boldsymbol{U}}_s^{(t)} T_s^{(t)} \quad (s = 1, 2, ..., K), \tag{14}$$

where

$$\nabla_{U_s} \ell_k(\boldsymbol{x}_t; \Lambda^{(t-1)}) = \ell_k'(\boldsymbol{x}_t; \Lambda^{(t-1)}) \rho_{k,s}(\boldsymbol{x}_t; \Lambda^{(t-1)}) \nabla_{U_s} g_s(\boldsymbol{x}_t; \boldsymbol{U}_s^{(t-1)}), \tag{15}$$

$$\nabla_{U_s} g_s(\boldsymbol{x}_t; \boldsymbol{U}_s^{(t-1)}) = 2\mathcal{P}^{\perp}_{U_s^{(t-1)}} \boldsymbol{x}_t \boldsymbol{x}_t^T \boldsymbol{U}_s^{(t-1)}, \tag{16}$$

$$\Lambda^{(t)} = \left[\boldsymbol{U}_1^{(t)} \; \boldsymbol{U}_2^{(t)} \; ... \; \boldsymbol{U}_K^{(t)} \right] \in \mathcal{R}^{d \times (\sum_{s=1}^{K} p_s)}, \tag{17}$$

$\varepsilon_t > 0$, $\ell_k'(\boldsymbol{x}_t; \Lambda^{(t-1)})$ (> 0) denotes the derivative of the *loss function* [4], $\rho_{k,s}(\boldsymbol{x}_t; \Lambda^{(t-1)})$ denotes the derivative of the *misclassification*

measure [4] ($\rho_{k,k} < 0,\ \rho_{k,s} > 0\ (s \neq k)$), and each $\boldsymbol{T}_s^{(t)}$ denotes the orthonormalization matrix applied to $\tilde{\boldsymbol{U}}_s^{(t)}$.

If we take the L_∞ norm in the misclassification measure (see [4]), the MELS algorithm leads to its special form named MELS*:

The MELS* method

At the t-th iteration, if a randomly-selected training pattern $\boldsymbol{x}_t$ belongs to the k-th class, then

$$\tilde{\boldsymbol{U}}_k^{(t)} = \left(\boldsymbol{I} + 2\varepsilon_t \ell_k'(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}) \boldsymbol{P}^{\perp}_{\boldsymbol{U}_k^{(t-1)}} \boldsymbol{x}_t \boldsymbol{x}_t^T\right) \boldsymbol{U}_k^{(t-1)}, \tag{18}$$

$$\tilde{\boldsymbol{U}}_j^{(t)} = \left(\boldsymbol{I} - 2\varepsilon_t \ell_k'(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}) \boldsymbol{P}^{\perp}_{\boldsymbol{U}_j^{(t-1)}} \boldsymbol{x}_t \boldsymbol{x}_t^T\right) \boldsymbol{U}_j^{(t-1)} \tag{19}$$

$$\left(j = \arg\max_{s, s \neq k}\ g_s(\boldsymbol{x}_t; \boldsymbol{U}_s^{(t-1)})\right),$$

$$\boldsymbol{U}_s^{(t)} = \tilde{\boldsymbol{U}}_s^{(t)} \boldsymbol{T}_s^{(t)}\ (s = k, j), \tag{20}$$

$$\boldsymbol{U}_i^{(t)} = \boldsymbol{U}_i^{(t-1)}\ (i = 1, 2, ..., K; i \neq k, j). \tag{21}$$

It can be seen that this MELS* algorithm closely resembles Kohonen's LSM. The difference in their forms is that MELS*, unlike LSM, includes the projection matrix $\boldsymbol{P}^{\perp}_{\boldsymbol{U}_s}$.

3.2 Convergence Property of MELS

One may think it trivial that MELS can achieve the minimum recognition error situation in the same way as MCE/GPD because it is simply derived by applying MCE/GPD to the SM framework. However, the fact that MELS includes the orthonormalization operation, which was not used in the original differentiation-based GPD adjustment, seems to require a further investigation of the learning convergence.

We first introduce the following theorem.

Theorem 2 *Assume that* $|\nabla_{u_{s,j,i}} \ell_k(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)})| < \infty$ *(*$s = 1, 2, ..., K; j = 1, 2, ..., d; i = 1, 2, ..., p_s$*). Then, if the Gram-Schmidt orthonormalization (GSO) is applied to the MELS parameter adjustment, the parameter sequence* $\{\boldsymbol{U}_s^{(t)}\ (t = 1, 2, ...)\}\ (s = 1, 2, ..., K)$ *obeys the following rule:*

$$\boldsymbol{U}_s^{(t)} = \boldsymbol{U}_s^{(t-1)} - \varepsilon_t \nabla_{\boldsymbol{U}_s} \ell_k(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}) + O(\varepsilon_t^2). \tag{22}$$

Proof. See Appendix B. □

With this theorem and the theorems described in [6], we next verify that MELS can attain the minimum recognition error situation. Define

the following vector which corresponds to $\Lambda = [\ \boldsymbol{U}_1\ \boldsymbol{U}_2\ ...\ \boldsymbol{U}_K\ ]$ one to one:

$$\boldsymbol{\lambda} = [\ u_{1,1,1}\ .....\ u_{s,j,i}\ .....\ u_{K,d,p_K}\ ]^T \quad \in \mathcal{R}^{d(\sum_{s=1}^{K} p_s)}. \tag{23}$$

Accordingly, the rule (22) can be expressed in terms of $\boldsymbol{\lambda}$ as

$$\boldsymbol{\lambda}^{(t)} = \boldsymbol{\lambda}^{(t-1)} - \varepsilon_t \nabla_{\boldsymbol{\lambda}} \ell_k(\boldsymbol{x}_t; \boldsymbol{\lambda}^{(t-1)}) + O(\varepsilon_t^2). \tag{24}$$

There exist a $d(\sum_{s=1}^{K} p_s)$-dimensional vector $\boldsymbol{d}_t$ ($||\boldsymbol{d}_t|| < \infty$) such that

$$\boldsymbol{\lambda}^{(t)} = \boldsymbol{\lambda}^{(t-1)} - \varepsilon_t \left(\nabla_{\boldsymbol{\lambda}} \ell_k(\boldsymbol{x}_t; \boldsymbol{\lambda}^{(t-1)}) - \varepsilon_t \boldsymbol{d}_t \right). \tag{25}$$

For simplicity, we define

$$\Delta \boldsymbol{\lambda}^{(t)} = \nabla_{\boldsymbol{\lambda}} \ell_k(\boldsymbol{x}_t; \boldsymbol{\lambda}^{(t-1)}) - \varepsilon_t \boldsymbol{d}_t. \tag{26}$$

Furthermore, it can be seen that the Hessian matrix exists,

$$\boldsymbol{H}_k(\boldsymbol{x}; \boldsymbol{\lambda}) = \nabla_{\boldsymbol{\lambda}}^2 \ell_k(\boldsymbol{x}; \boldsymbol{\lambda}), \tag{27}$$

since each orthonormalized matrix $\boldsymbol{U}_s^{(t)}$ is full-rank. Then, we reach the following theorem guaranteeing the MELS convergence:

Theorem 3 *Make the following assumptions: for all* $t \in \{1, 2, ...\}$,

a1) $|\nabla_{u_{s,j,i}} \ell_k(\boldsymbol{x}_t; \Lambda^{(t-1)})| < \infty$
$(s = 1, 2, ..., K; j = 1, 2, ..., d; i = 1, 2, ..., p_s)$,

a2) $< \Delta \boldsymbol{\lambda}^{(t)} | \boldsymbol{H}_k(\boldsymbol{x}_t; \boldsymbol{\lambda}^{(t-1)} - \theta_t \varepsilon_t \Delta \boldsymbol{\lambda}^{(t)}) \Delta \boldsymbol{\lambda}^{(t)} > < \infty$,
where $< \cdot | \cdot >$ *denotes the inner product and* $0 \leq \theta_t \leq 1$.

Then, if an infinite sequence of random observations $\boldsymbol{x}_t$ *is presented for training and the parameter adjustment rule of* (13,14) *is utilized with a sequence* ε_t *that satisfies* $\sum_{t=1}^{\infty} \varepsilon_t \to \infty$ *and* $\sum_{t=1}^{\infty} \varepsilon_t^2 < \infty$, *then the parameter sequence* $\{\Lambda^{(t)}; t = 1, 2, ...\}$ *according to MELS converges with probability one to a* Λ^* *which results in a local minimum of* $L(\Lambda)$ *under the orthonormality constraints, where* $L(\Lambda)$, *called the expected loss, is defined as the expectation of* $\ell_k(\boldsymbol{x}; \Lambda)$ *over the whole pattern space* $\mathcal{X}$ *and approximates (arbitrarily closely) the recognition error probability in the case of the smoothed 0-1 loss function (see* [4, 6]*).*

Proof. Taking Taylor expansion,

$$\begin{aligned} \ell_k(\boldsymbol{x}_t; \boldsymbol{\lambda}^{(t)}) \quad = \quad & \ell_k(\boldsymbol{x}_t; \boldsymbol{\lambda}^{(t-1)}) - \varepsilon_t ||\nabla_{\boldsymbol{\lambda}} \ell_k(\boldsymbol{x}_t; \boldsymbol{\lambda}^{(t-1)})||^2 \\ & + \varepsilon_t^2 < \boldsymbol{d}_t | \nabla_{\boldsymbol{\lambda}} \ell_k(\boldsymbol{x}_t; \boldsymbol{\lambda}^{(t-1)}) > \\ & + \frac{1}{2} \varepsilon_t^2 < \Delta \boldsymbol{\lambda}^{(t)} | \boldsymbol{H}_k(\boldsymbol{x}_t; \boldsymbol{\lambda}^{(t-1)} - \theta_t \varepsilon_t \Delta \boldsymbol{\lambda}^{(t)}) \Delta \boldsymbol{\lambda}^{(t)} >, \end{aligned} \tag{28}$$

where the last term denotes the Lagrange's remainder term. By the Cauchy-Schwarz inequality,

$$| < \boldsymbol{d}_t | \nabla_{\boldsymbol{\lambda}} \ell_k(\boldsymbol{x}_t; \boldsymbol{\lambda}^{(t-1)}) > | \leq ||\boldsymbol{d}_t|| \cdot ||\nabla_{\boldsymbol{\lambda}} \ell_k(\boldsymbol{x}_t; \boldsymbol{\lambda}^{(t-1)})|| < \infty. \tag{29}$$

Then, the same proof as in [6] can be applied. □

4 Conclusions

We have studied the discriminative learning nature of SM from the viewpoint of minimizing recognition errors. Based on MCE/GPD, we have also formalized a new discriminative SM algorithm called the Minimum Error Learning Subspace method (MELS). MELS is a slightly different version of the conventional LSM and has been proven to achieve the minimum error situation. This result significantly increases the SM's applicability by providing the mathematical guarantee of training optimality.

Appendix

A Proof of Theorem 1

First we show the following formulae related to matrix differentiation without proofs: with proper-sized matrices $\boldsymbol{C}$ and $\boldsymbol{X}$,

$$\nabla_X \mathrm{tr}(\boldsymbol{C}\boldsymbol{X}) = \nabla_X \mathrm{tr}(\boldsymbol{X}\boldsymbol{C}) = \boldsymbol{C}^T, \tag{30}$$

$$\nabla_X \mathrm{tr}(\boldsymbol{C}\boldsymbol{X}^T) = \nabla_X \mathrm{tr}(\boldsymbol{X}^T\boldsymbol{C}) = \boldsymbol{C}, \tag{31}$$

where tr denotes the trace operator; if a matrix $\boldsymbol{C}(t)$ is non-singular and is parametrized by a scalar t,

$$\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{C}(t)^{-1} = -\boldsymbol{C}(t)^{-1}\frac{\mathrm{d}\boldsymbol{C}(t)}{\mathrm{d}t}\boldsymbol{C}(t)^{-1}. \tag{32}$$

The orthogonal projection defined in (6) can be rewritten as

$$g(\boldsymbol{x};\boldsymbol{U}) = \mathrm{tr}\left[\boldsymbol{U}(\boldsymbol{U}^T\boldsymbol{U})^{-1}\boldsymbol{U}^T\boldsymbol{x}\boldsymbol{x}^T\right]. \tag{33}$$

Hearafter we omit the suffix s for simplicity. Then, using the above-listed formulae, it can be easily shown that the derivative of $g(\boldsymbol{x};\boldsymbol{U})$ with respect to $\boldsymbol{U}$ is expressed as

$$\nabla_U g(\boldsymbol{x};\boldsymbol{U}) = 2\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{U}(\boldsymbol{U}^T\boldsymbol{U})^{-1} + \{\nabla_V F(\boldsymbol{V})\}_{V=U}, \tag{34}$$

$$F(\boldsymbol{V}) = \mathrm{tr}(\boldsymbol{A}\boldsymbol{Y}), \tag{35}$$

$$\boldsymbol{A} = (\boldsymbol{V}^T\boldsymbol{V})^{-1}, \tag{36}$$

$$\boldsymbol{Y} = \boldsymbol{U}^T\boldsymbol{x}\boldsymbol{x}^T\boldsymbol{U}. \tag{37}$$

Let $v_{i,j}$ be the (i,j) entry of the matrix $\boldsymbol{V}$ and $a_{m,n}$ be the (m,n) entry of $\boldsymbol{A}$. Considering that $\boldsymbol{A}$ is a function of $\boldsymbol{V}$, the partial derivative of F with respect to each entry $v_{i,j}$ is derived using a chain rule as

$$\frac{\partial F(\boldsymbol{V})}{\partial v_{i,j}} = \sum_{m=1}^{p}\sum_{n=1}^{p}\frac{\partial F}{\partial a_{m,n}}\frac{\partial a_{m,n}}{\partial v_{i,j}} = \mathrm{tr}\left[\left(\frac{\partial F}{\partial \boldsymbol{A}}\right)^T\frac{\partial \boldsymbol{A}}{\partial v_{i,j}}\right]. \tag{38}$$

Using the formulae (30) and (32),

$$\frac{\partial F}{\partial A} = \mathbb{V}^T, \tag{39}$$

$$\frac{\partial A}{\partial v_{i,j}} = -(\mathbb{V}^T\mathbb{V})^{-1}\left(\frac{\partial(\mathbb{V}^T\mathbb{V})}{\partial v_{i,j}}\right)(\mathbb{V}^T\mathbb{V})^{-1}. \tag{40}$$

Thus, using above equations, the following equation can be derived:

$$\begin{aligned}\frac{\partial F(\mathbb{V})}{\partial v_{i,j}} &= -\mathrm{tr}\left[\mathbb{V}(\mathbb{V}^T\mathbb{V})^{-1}\left(\frac{\partial(\mathbb{V}^T\mathbb{V})}{\partial v_{i,j}}\right)(\mathbb{V}^T\mathbb{V})^{-1}\right] \\ &= -\left\{\frac{\partial}{\partial w_{i,j}}\mathrm{tr}\left[(\mathbb{W}^T\mathbb{W})(\mathbb{V}^T\mathbb{V})^{-1}\mathbb{V}(\mathbb{V}^T\mathbb{V})^{-1}\right]\right\}_{w_{i,j}=v_{i,j}} \end{aligned} \tag{41}$$

where $w_{i,j}$ is the (i,j) entry of $\mathbb{W}$. Expressing the above equation in a matrix form and using the formulae (30,31), we can get

$$\begin{aligned}\nabla_{\mathbb{V}}F(\mathbb{V}) &= -\left\{\frac{\partial}{\partial \mathbb{W}}\mathrm{tr}\left[(\mathbb{W}^T\mathbb{W})(\mathbb{V}^T\mathbb{V})^{-1}\mathbb{V}(\mathbb{V}^T\mathbb{V})^{-1}\right]\right\}_{W=V} \\ &= -2\mathbb{V}(\mathbb{V}^T\mathbb{V})^{-1}\mathbb{V}(\mathbb{V}^T\mathbb{V})^{-1}. \end{aligned} \tag{42}$$

Consequently, substituting the above equation into (34), we can get

$$\begin{aligned}\nabla_U g(x;U) &= 2xx^TU(U^TU)^{-1} - 2U(U^TU)^{-1}U^Txx^TU(U^TU)^{-1} \\ &= 2P_U^{\perp}xx^TU(U^TU)^{-1}. \end{aligned} \tag{43}$$

This completes the proof.

B Proof of Theorem 2

The equation (22) is rewritten in terms of column vectors as

$$\begin{aligned}u_{s,i}^{(t)} &= u_{s,i}^{(t-1)} - \varepsilon_t\nabla_{s,i}\ell_k(x_t;A^{(t-1)}) + O(\varepsilon_t^2) \\ &\quad (s=1,2,...,K;i=1,2,...,p_s), \end{aligned} \tag{44}$$

where

$$\nabla_{s,i}\ell_k(x;A) = \nabla_{u_{s,i}}\ell_k(x;A) \tag{45}$$

$$= P_{U_s}^{\perp}\left(2\ell_k'(x;A)\rho_{k,s}(x;A)xx^T\right)u_{s,i}. \tag{46}$$

Before getting into the proof, let us state the following lemma.

Lemma 1 *In each class C_s $(s=1,2,...,K)$, every base vector $\{u_{s,i}\}_{i=1}^{p_s}$ is orthogonal to every gradient vector $\{\nabla_{s,i}\ell_k(x;A)\}_{i=1}^{p_s}$.*

Proof. It can be easily verified by (46). □

The proof of Theorem 2 (or (44)) follows the mathematical induction. First we give the proof in the case $i=1$. Since the first base vector $\tilde{u}_{s,1}$ is merely normalized by the GSO,

$$\tilde{\boldsymbol{u}}_{s,1}^{(t)} = \boldsymbol{u}_{s,1}^{(t-1)} - \varepsilon_t \nabla_{s,1} \ell_k(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}), \tag{47}$$

$$\boldsymbol{u}_{s,1}^{(t)} = \frac{\tilde{\boldsymbol{u}}_{s,1}^{(t)}}{||\tilde{\boldsymbol{u}}_{s,1}^{(t)}||}. \tag{48}$$

From (47) we can get

$$||\tilde{\boldsymbol{u}}_{s,1}^{(t)}||^2 = 1 + \varepsilon_t^2 ||\nabla_{s,1} \ell_k(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)})||^2 \tag{49}$$

because of the orthonormality of $\boldsymbol{U}_s^{(t-1)}$ and Lemma 1. Therefore,

$$\frac{1}{||\tilde{\boldsymbol{u}}_{s,1}^{(t)}||} = 1 - \frac{1}{2}\varepsilon_t^2 M + \left(1 + \varepsilon_t^2 M\right)^{-\frac{1}{2}} - \left(1 - \frac{1}{2}\varepsilon_t^2 M\right), \tag{50}$$

where $M = ||\nabla_{s,1} \ell_k(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)})||^2$. Then, by (48) and (50), and because of the boundedness of $||\nabla_{s,1} \ell_k(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)})||$, $\boldsymbol{u}_{s,1}^{(t)}$ becomes

$$\boldsymbol{u}_{s,1}^{(t)} = \tilde{\boldsymbol{u}}_{s,1}^{(t)} + O(\varepsilon_t^2) + \varepsilon_t^2 \gamma_{s,1}^{(t)} \tilde{\boldsymbol{u}}_{s,1}^{(t)}, \tag{51}$$

$$\gamma_{s,1}^{(t)} = \frac{F(\varepsilon_t^2)}{\varepsilon_t^2}, \tag{52}$$

$$F(\varepsilon_t^2) = \left(1 + \varepsilon_t^2 M\right)^{-\frac{1}{2}} - \left(1 - \frac{1}{2}\varepsilon_t^2 M\right). \tag{53}$$

Lemma 2 $\lim_{\varepsilon_t^2 \to 0} \gamma_{s,1}^{(t)} = 0.$

Proof. Since $\lim_{\varepsilon_t^2 \to 0} F(\varepsilon_t^2) = 0$, and by de l'Hospital's theorem,

$$\begin{aligned} \lim_{\varepsilon_t^2 \to 0} \gamma_{s,1}^{(t)} &= \lim_{\varepsilon_t^2 \to 0} \frac{\mathrm{d}F(\varepsilon_t^2)/\mathrm{d}(\varepsilon_t^2)}{\mathrm{d}(\varepsilon_t^2)/\mathrm{d}(\varepsilon_t^2)} \\ &= \lim_{\varepsilon_t^2 \to 0} \left(-\frac{1}{2}\left(1 + \varepsilon_t^2 M\right)^{-\frac{3}{2}} + \frac{1}{2}\right) M = 0. \end{aligned}$$ □

Accordingly, considering Lemma 2, we can say that

$$\boldsymbol{u}_{s,1}^{(t)} = \boldsymbol{u}_{s,1}^{(t-1)} - \varepsilon_t \nabla_{s,1} \ell_k(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}) + O(\varepsilon_t^2). \tag{54}$$

Next, *suppose* that in each case $i = 1, 2, ..., j$ the following equation holds:

$$\boldsymbol{u}_{s,i}^{(t)} = \boldsymbol{u}_{s,i}^{(t-1)} - \varepsilon_t \nabla_{s,i} \ell_k(\boldsymbol{x}_t; \boldsymbol{\Lambda}^{(t-1)}) + O(\varepsilon_t^2), \tag{55}$$

and consider the case $i = j + 1$. According to the GSO, the $(j+1)$-th normalized base vector $\boldsymbol{u}_{s,j+1}^{(t)}$ is derived from $\left\{\boldsymbol{u}_{s,i}^{(t)}\right\}_{i=1}^{j}$ as

$$\boldsymbol{v}_{s,j+1}^{(t)} = \left(\boldsymbol{I} - \sum_{i=1}^{j} \boldsymbol{u}_{s,i}^{(t)} \boldsymbol{u}_{s,i}^{(t)T}\right) \tilde{\boldsymbol{u}}_{s,j+1}^{(t)}, \tag{56}$$

$$\boldsymbol{u}_{s,j+1}^{(t)} = \frac{\boldsymbol{v}_{s,j+1}^{(t)}}{||\boldsymbol{v}_{s,j+1}^{(t)}||}. \tag{57}$$

Note that

$$\tilde{u}_{s,j+1}^{(t)} = u_{s,j+1}^{(t-1)} - \varepsilon_t \nabla_{s,j+1} \ell_k(x_t; A^{(t-1)}). \tag{58}$$

From (55) and (58) we can get $(i = 1, 2, ..., j)$

$$u_{s,i}^{(t)T} \tilde{u}_{s,j+1}^{(t)} = O(\varepsilon_t^2) \tag{59}$$

because of the orthonormality of $U_s^{(t-1)}$ and Lemma 1. Therefore, from (56), (58) and (59),

$$v_{s,j+1}^{(t)} = u_{s,j+1}^{(t-1)} - \varepsilon_t \nabla_{s,j+1} \ell_k(x_t; A^{(t-1)}) + O(\varepsilon_t^2), \tag{60}$$

and accordingly we can get

$$||v_{s,j+1}^{(t)}||^2 = 1 + \varepsilon_t^2 ||\nabla_{s,j+1} \ell_k(x_t; A^{(t-1)})||^2 + O(\varepsilon_t^2) \tag{61}$$

because of the orthonormality of $U_s^{(t-1)}$ and Lemma 1. Then, according to the logic analogous to the case $i = 1$, we reach

$$u_{s,j+1}^{(t)} = u_{s,j+1}^{(t-1)} - \varepsilon_t \nabla_{s,j+1} \ell_k(x_t; A^{(t-1)}) + O(\varepsilon_t^2). \tag{62}$$

This completes the proof according to the mathematical induction.

References

[1] E. Oja, *Subspace Methods of Pattern Recognition*, Research Studies Press, 1983.

[2] Y. Ariki and K. Doi, "Speaker recognition based on subspace methods", *Proc. ICSLP 94*, Yokohama, pp.1859-1862, Sep. 1994.

[3] Y. Takebayashi, "Speech recognition based on the subspace method: AI class-description learning viewpoint", *J. Acoust. Soc. Jpn. (E)*, vol. 13, No. 6, pp.429-439, Nov. 1992.

[4] B.-H. Juang and S. Katagiri, "Discriminative learning for minimum error classification", *IEEE Trans. Signal Processing*, vol. 40, No. 12, pp. 3043-3054, Dec. 1992.

[5] S. Katagiri, C.-H. Lee, and B.-H. Juang, "New discriminative training algorithms based on the generalized probabilistic descent method", in *Proc. 1991 IEEE Workshop on Neural Networks for Signal Processing*, pp. 299-308, Princeton, NJ, Sept. 1991.

[6] W. Chou and B.-H. Juang, "Adaptive discriminative learning in pattern recognition", Technical Report of AT&T Bell Laboratory.

A UNIFYING VIEW OF STOCHASTIC APPROXIMATION, KALMAN FILTER AND BACKPROPAGATION

*Enrico Capobianco**
University of Padua, Statistics Department

Abstract

In this paper the relationships between the Stochastic Approximation, the Kalman Filter and the Backpropagation algorithms are investigated. We show that when the Neural Network architecture at hand can be formalized such that the approximation of the optimum for a nonlinear objective function is the problem for which we seek a solution, then both Stochastic Approximation techniques and appropriate Kalman Filters can be employed in order to reach the goal but the latter can also handle various structural characteristics of the stochastic processes involved and suggest a more efficient two-step estimator.

1 Introduction

Recent developments of neural networks have required comparisons with statistical and control-systems approaches in order to verify the potential gains from using neural nets for statistical parameter estimation, nonlinear dynamic system modelling and identification, time series prediction, optimal and suboptimal filtering etc. Various algorithms have been proposed and tested on a real or simulated basis and they showed interesting results. Other studies have considered the opportunity of unifying the theoretical concepts behind the construction of the practical algorithms; this paper belongs to this second category and is devoted to synthesizing the most important learning procedure, i.e. the backpropagation algorithm, with stochastic approximation techniques and Kalman Filter algorithms. The paper proceeds as follows. Section 2 describes how the backpropagation algorithm is easily embedded in the stochastic approximation framework. Section 3 introduces the extended

*This paper was prepared at Stanford University while the author was a visiting research scholar with the PDP research group.

0-7803-2739-X/95 $4.00 © 1995 IEEE

and iterated versions of the Kalman Filter and shows the relationship existing with the Gauss-Newton method. Section 4 points out the basic advantages that can derive from working with a state space representation of the neural network and introduces a two-step parameter estimation procedure. Section 5 is for the conclusions.

2 Backpropagation and Stochastic Approximation algorithms

Following [8], suppose we have a nonlinear objective function $f(X_t, \theta)$ where $f : R^k \times \Theta \rightarrow R$, X_t is a $k \times 1$ random input vector and $\theta \in \Theta \subset R^p$ represents the vector of unknown parameters. We want to use this function for forecasting the random variable y_t and to do this we allow the following single hidden layer feedforward network structure for $f(X_t, \theta)$:

$$f(X_t, \theta) = x'\phi + \psi + \sum_{j=1}^{r} \psi_j F(\bar{x}'\gamma_j) \tag{1}$$

where $\bar{x} = (1, x')'$, $\theta = (\phi', \psi', \gamma')'$, $\psi = (\psi_0, \ldots, \psi_r)'$, $\gamma = (\gamma_1, \ldots, \gamma_r)'$, $r \in N$ and $F : R \rightarrow R$ (a bounded and continuosly differentiable function). Consider that $f(X_t, \theta)$ is an approximation of the objective function $g(X_t) = E(y_t/X_t)$. In this nonlinear least square set-up we seek a solution θ^* to $min_\theta[E([y_t - f(X_t, \theta)]^2)]$, or equivalently to $E(\nabla_\theta f(X_t, \theta)[y_t - f(X_t, \theta)]) = 0$, with ∇_θ representing the gradient $k \times 1$ vector calculated w.r.t. θ. A very straightforward way to solve this problem is to employ the Robbins-Monro (RM) Stochastic Approximation algorithm, whose structure is given by:

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \delta_t \nabla_\theta f(X_t, \hat{\theta})[y_t - f(X_t, \hat{\theta})] \tag{2}$$

This recursion is that of a Stochastic Gradient method and therefore generalizes the Backpropagation (BP) algorithm [12] for neural network learning in allowing for a time varying learning rate. In [8] some modifications to the RM algorithm are presented in order to speed up the convergence rate and thus include a Gauss-Newton step at each updating stage obtaining a Modified Robbins-Monro algorithm:

$$\hat{\theta}_{t+1} = \hat{\theta}_t + \delta_t \hat{L}_{t+1}^{-1} \nabla_\theta f(X_t, \hat{\theta}_t)[y_t - f(X_t, \hat{\theta}_t)] \tag{3}$$

$$\hat{L}_{t+1} = \hat{L}_t + \delta_t [\nabla_\theta f(X_t, \hat{\theta}_t) \nabla_\theta f(X_t, \hat{\theta}_t) - \hat{L}_t] \tag{4}$$

where now $\Xi = ((vecL)', \theta')'$ is the new augmented parameter vector. As a result of the above modification and other technical devices employed in order to avoid numerical problems and deal with the computational burden involved, the new algorithm and therefore the correspondent generalized

backpropagation scheme are able to perform with a better convergence rate than the simple RM scheme also when moderate dependence is present in the data. Of course the approximation to $E(y_t/X_t)$ is only locally optimal, but it's nevertheless important to relax the usually retained " i.i.d." assumption about the stochastic process generating the data. Then, the final important contribution given in [8] is that of showing the consistency and asymptotic normality of the designed estimator, under appropriate conditions on the learning rate (i.e. $\delta_t = (t+1)^{-1}$).

3 Extended and Iterated Kalman Filter algorithms

Artificial Neural Networks can be conveniently cast in a state space representation. A general nonlinear state space model is given by a system equation $x_{t+1} = g_t(x_t) + w_t$ and a measurement equation $y_t = h_t(x_t) + v_t$ where $g_t(.)$ and $h_t(.)$ are nonlinear functions. Consider now a k-layered feedforward network structure described as in [3]:

$$i_j^k = \sum_{l=1}^{N_{k-1}} w_{lj}^{k-1,k} o_l^{k-1} + \beta_j^k \tag{5}$$

where the input of the j^{th} node in the k^{th} layer is given by the sum of the product of the connection weight w with the output and a bias parameter, and

$$o_j^k = F(i_j^k) \tag{6}$$

where the output as a function of the input through $F : R \rightarrow R$. Given the standard BP procedure:

$$w_{lj}^{k-1,k}(t+1) = w_{lj}^{k-1,k}(t) - \delta \epsilon_j^k(t) o_l^{k-1}(t) \tag{7}$$

where δ is the learning rate and $\epsilon_j^k = (o_j^k - y_j^k)F^1(i_j^k)$, we can rewrite the network in state space (and usually compact notation) form as:

$$W_{t+1} = W_t + G\xi_t \tag{8}$$

$$y_t = O_t + \eta_t \tag{9}$$

where η_t is the output error, $G\xi_t$ is equal to the correction term[1] of the BP recursion (7), but with only ξ_t characterized by pure erratic behavior, and where the signal in the measurement equation is actually $O_t(W_t)$. As a matter of fact in this framework the Extended Kalman Filter (EKF) and/or the Iterated Kalman Filter (IKF) represent the master estimation scheme,

[1]The matrix G separates in a convenient way the deterministic components from the purely random ones; see [3] for details.

since standard approximations are introduced in order to derive a suboptimal filter for the signal or to forecast the observed random variable. The general nonlinear functions $g_t(.)$ and $h_t(.)$, when sufficiently smooth, are expanded in Taylor series about the conditional means $\hat{x}_{t/t}$ and $\hat{x}_{t/t-1}$, thus obtaining $g_t(x_t) = g_t(\hat{x}_{t/t}) + G_t(x_t - \hat{x}_{t/t}) + \ldots$ and $h_t(x_t) = h_t(\hat{x}_{t/t-1}) + H_t(x_t - \hat{x}_{t/t-1}) + \ldots$. A formal way to operate with a more accurate algorithm is the following one[2]; starting from the EKF algorithm:

$$\hat{x}^+ = \hat{x}^- + K(y - h(\hat{x}^-)) \tag{10}$$

$$P^+ = (I - KH)P \tag{11}$$

$$H = h^1(\hat{x}^-) \tag{12}$$

$$K = PH'(HPH' + R)^{-1} \tag{13}$$

and given $\hat{x} = \hat{x}^-$, we can obtain the Iterated Kalman Filter as follows:

$$x_{it+1} = \hat{x} + K_{it}(y - h(x_{it}) - H_{it}(\hat{x} - x_{it})) \tag{14}$$

$$P_{it+1} = (I - K_{it}H_{it})P_{it} \tag{15}$$

$$H_{it} = h^1(x_{it}) \tag{16}$$

$$K_{it} = P_{it}H'_{it}(H_{it}P_{it}H'_{it} + R)^{-1} \tag{17}$$

Recently [2] have shown that the IKF algorithm is an application of the Gauss-Newton (GN) method. It's common in statistics and econometrics to work with estimators that aim at minimizing some sum-of-squares functions like $S(\theta) = \sum_t \epsilon_t^2$, where the vector ϵ represents a residual term from an estimated linear/nonlinear regression or time series model. The vector of first derivatives, or Gradient, in this case is $v(\theta) = \frac{\partial S(\theta)}{\partial \theta} = 2\sum \frac{\partial \epsilon_t}{\partial \theta}\epsilon_t$ and the Hessian is given by $V(\theta) = \frac{\partial^2 S(\theta)}{\partial \theta \partial \theta'} = 2\sum[\frac{\partial \epsilon_t}{\partial \theta}\frac{\partial \epsilon_t}{\partial \theta'} - \frac{\partial^2 \epsilon_t}{\partial \theta \partial \theta'}\epsilon_t]$. Several schemes are able of iteratively finding a solution to the initial minimization problem. The most general one is the Newton-Raphson (NR) method, which is given by:

$$\theta^* = \hat{\theta} + [\sum(\frac{\partial \epsilon_t}{\partial \theta}\frac{\partial \epsilon_t}{\partial \theta'} - \frac{\partial^2 \epsilon_t}{\partial \theta \partial \theta'}\epsilon_t)]^{-1}\sum \frac{\partial \epsilon_t}{\partial \theta}\epsilon_t \tag{18}$$

Since the term involving second derivatives is usually small when compared to the first derivatives product term, the GN scheme approximates the above iterative solution and presents a formula that is identical to NR, apart from the term with second derivatives. In [2] this last approximate scheme is

[2]A different filter can be derived by including more terms in the Taylor series expansions, thus obtaining second order Extended Kalman Filters or, when $h_t(x_t)$ can be linearized about the updated conditional mean estimate $\hat{x}_{t/t}$, it should be possible to improve the linearization and thus the final estimate of the state variable (see [1]).

applied to the log-likelihood criterion function $l(\epsilon) = \frac{1}{2}(\epsilon' Q^{-1} \epsilon)$ derived from a state space model whose components are $z = [y, \ \hat{x}]'$, $m(x) = [h(x), \ x]'$, $z \sim N(m(x), Q)$, with the 2×2 matrix Q having zero off-diagonal elements and the variances R and P in the main diagonal. Given that in this case $S(\theta) = \mid \epsilon \mid^2 = (\epsilon' Q^{-1} \epsilon)$ and given the factorization $B'B = Q^{-1}$, we have $\epsilon(.) = B(z - m(x))$ and thus $\frac{\partial \epsilon(.)}{\partial \theta} = -Bm^1(x)$; replacing this last term in the GN formula, it is shown in [2] that, after some algebra, the same identical updating equation employed by the IKF to estimate the state variable can be obtained. Thus, by induction the iterates from the GN method correspond to those from the IKF algorithm.

4 Some extensions and generalizations

Since we showed how to cast a neural network architecture in a state space representation, we should try to exploit the properties of it. The most important fact is that from the Kalman Filter algorithm and its variants we obtain, in a very elegant and immediate way, the likelihood function through the well-known prediction error decomposition device [13]. The likelihood function for the whole set of observations is obtained by the joint PDF, i.e. $L(y, \theta) = \Pi_{t=1}^{N} p(y_t / Y_{t-1})$, considering Y_{t-1} the set of observations up to and including y_{t-1}; since the innovation or prediction error computed by the filter is $\eta_t = y_t - E(y_t / Y_{t-1})$ and $var(\eta_t) = var(y_t / Y_{t-1}) = D_t$, when the observations are normally distributed the likelihood function can be expressed in terms of the innovations. Therefore the likelihood function in prediction error decomposition form is:

$$\log L = -\frac{KN}{2} \log 2\pi - \frac{1}{2} \sum_{t=1}^{N} \log \mid D_t \mid - \frac{1}{2} \sum_{t=1} \eta_t' D_t^{-1} \eta_t \qquad (19)$$

considering a $k \times 1$ η_t vector and N observations[3]. For the case we study, where a nonlinear model is the object of investigation and some approximations were used, the solution is of course suboptimal and close to the optimal one according to the accuracy of the approximation involved. But there are other aspects which deserve to be mentioned, like:

- the state space set-up can deal with data from stochastic processes which are stationary or not and with parameters (those which build up the functional form of the neural network, for instance) that are fixed or time-varying

[3] Under Gaussianity the filter delivers an optimal Minimum Mean Squared solution for the estimation problem; under hypotheses different from the Gaussian, the filter gives only a Minimum Mean Square Linear solution and the values which are computed are Quasi Maximum Likelihood estimates, less efficient but consistent (and therefore useful to start a recursive procedure).

- apart from the typical markovian structure of the system equation which characterizes an autoregressive dynamic evolution for the state variable to be estimated, various other aspects can be considered by this model representation: fixed and random exogenous effects, switching regimes, missing data, stochastic variance models, qualitative observations etc.

- the disturbance terms allow for the analysis of real input data, usually noisy: when their distribution is not Gaussian various nonlinear filtering techniques are available (see [7], for instance)

Returning to the issue concerning the implementation of recursive schemes, consider what we have from KF running over the state space model: we obtain the likelihood function as given by (19) and we can calculate its derivatives either analytically or numerically. In this latter case through the same filter. For instance the i^{th} element of the Score vector is given by (see [5]):

$$\frac{\partial \log L}{\partial \theta_i} = -\frac{1}{2}\sum_t [tr[(D_t^{-1}\frac{\partial D_t}{\partial \theta_i})(I - D_t^{-1}\eta_t\eta_t')] - \frac{\partial \eta_t'}{\partial \theta_i}D_t^{-1}\eta_t] \qquad (20)$$

therefore requiring the evaluation of the $k \times k$ matrices of derivatives $\frac{\partial D_t}{\partial \theta_i}$ and the $k \times 1$ vector of derivatives $\frac{\partial \eta_t}{\partial \theta_i}$, for $i = 1, \ldots, p$ and $t = 1, \ldots, N$. These derivatives may be computed through p additional passes of the KF; if we consider a new run of the filter with $\theta = [\theta_1 \ldots\ldots \theta_i + \delta_i \ldots\ldots \theta_p]$ we obtain a new set of innovations $\eta_t^{(i)}$ and variances $D_t^{(i)}$ and the numerical approximations of the derivatives[4] are $\delta_i^{-1}[\eta_t^{(i)} - \eta_t]$ and $\delta_i^{-1}[D_t^{(i)} - D_t]$.

Then, the Berndt, Hall, Hall and Hausman (BHHH) algorithm which suggests the approximation of the Hessian by the well-known outer-product formula, i.e. $\sum_{t=1}^{N} \frac{\partial \log L}{\partial \theta}\frac{\partial \log L}{\partial \theta'}$, is the most indicated choice we have in order to improve, at least asymptotically, the efficiency of the initial GN or equivalently KF estimator $\hat{\theta}$ according to the recursion:

$$\theta^* = \hat{\theta} + \lambda\,[\sum_{t=1}^{N} \frac{\partial \log L}{\partial \theta}\frac{\partial \log L}{\partial \theta'}]^{-1}\sum_{t=1}^{N}\frac{\partial \log L}{\partial \theta} \qquad (21)$$

where λ is a variable step length chosen so that the likelihood function is maximized in a given direction.

5 Conclusions

In this paper we showed that the Backpropagation formula, the Stochastic Approximation method and the Extended and Iterated Kalman Filter algorithms have many common properties and we underlined the fact that most

[4] The same derivatives can be used to compute the Information Matrix (see [5]).

of the powerful state space machinery for the neural network representation has not been exploited. Then, from the inferential side it should be important the possibility of employing two-step estimators that use likelihood information (through its functionals involved) in order to reach better asymptotically efficient final estimates.

References

[1] B.D.O. Anderson, J.B. Moore: *Optimal Filtering.* Prentice Hall, Englewood Cliffs, NJ (1979).

[2] B.M Bell, F.W. Cathery: " The Iterated Kalman Filter update is a Gaussian Newton method ". *IEEE Transactions on Automatic Control,* 38, 294-297 (1993).

[3] G. Chen, H. Ogmen: " Modified extended Kalman filtering for supervised learning". *International Journal of Systems Science,* 24, 1207-1214 (1993).

[4] S. Chen, S.A. Billings: " Neural Networks for nonlinear dynamic system modelling and identification ". *International Journal of Control,* 56, 319-346 (1992).

[5] A.C. Harvey: *Forecasting, Structural Time Series Models and the Kalman Filter.* Cambridge University Press, Cambridge (1989).

[6] K.J. Hunt, D. Sbarbaro, R. Zbikowski, P.J. Gawthrop: " Neural Networks for Control Systems-A Survey ". *Automatica,* 28, 1083-1112 (1992).

[7] G. Kitagawa: " Nongaussian state space modelling of nonstationary time series ". *Journal of the American Statistical Association,* 82, 1032-1063 (1987).

[8] C.M. Kuan, H. White: " Artificial Neural Networks: an econometric perspective ". *Econometric Reviews,* 13, 1-92 (1994).

[9] O. Nerrand, P. Roussel-Ragot, L. Personnaz, G. Dreyfus: " Neural Networks and Nonlinear Adaptive Filtering: unifying concepts and new algorithms ". *Neural Computation,* 5, 165-199 (1993).

[10] I. Poli, R.D. Jones: " A neural net model for prediction ". *Journal of the American Statistical Association,* 89, 117-121 (1994).

[11] H. Robbins, S. Monro: " A Stochastic Approximation Method ". *Annals of Mathematical Statistics,* 22, 400-407 (1951).

[12] D.E. Rumelhart, G.E. Hinton, R.J. Williams: " Learning internal representations by error propagation ". In D.E. Rumelhart and J.L. McClelland eds: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Cambridge: MIT Press,1, 318-362 (1986).

[13] F.C. Schweppe:"Evaluation of likelihood functions for gaussian signals". *IEEE Transactions on Information Theory*, IT-11, 61-70 (1965).

GLOBALLY-ORDERED TOPOLOGY-PRESERVING MAPS ACHIEVED WITH A LEARNING RULE PERFORMING LOCAL WEIGHT UPDATES ONLY

Marc M. Van Hulle
Laboratorium voor Neuro- en Psychofysiologie
K.U.Leuven
Campus Gasthuisberg
Herestraat
B-3000 Leuven, BELGIUM
Tel.: + 32 16 34 59 61 Fax: + 32 16 34 59 93
E-mail: marc@neuro.kuleuven.ac.be

Abstract— A new unsupervised competitive learning rule is introduced for topology-preserving map formation and vector quantization. The rule, called Maximum Entropy learning Rule (MER), achieves a globally-ordered map by performing local weight updates only. Hence, contrary to Kohonen's self-organizing map algorithm and its many variations, no neighborhood function is needed. The rule yields an equiprobable quantization of a d-dimensional input *p.d.f.* Simulations are performed to show that the dynamical- and convergence properties of MER are essentially different from those of Kohonen's algorithm.

INTRODUCTION

One of the most striking features of the sensory cortex is the topographical organization of its areas. As a result of this organization, neighboring neurons code for neighboring positions in sensory space. Models accounting for the formation of these topology-preserving maps from simple principles of self-organization have been proposed by several authors [1-5]. Due to its inherent simplicity, the Kohonen self-organizing map is the most successful model in this series. It has enjoyed a wide range of applications (for partial overviews, see [5,6]), and its computational capabilities [7] and dynamics are thoroughly studied and understood, at least for the scalar case [6,8-11]. Furthermore, it is believed to offer a rather detailed physiological interpretation of topology-preserving map formation in cortical sensory areas [12].

The aim of Kohonen's algorithm is to establish, in an unsupervised way, a mapping from a higher d-dimensional space V of input signals onto an equal or lower-dimensional discrete lattice A of N formal neurons. To each formal neuron $i \in A$ corresponds a unique weight vector $\mathbf{w}_i = [w_{i1}, ..., w_{id}]$. The map assigns to each input $v = [v_1, ..., v_d] \in V$ a unique neuron in A and this is accomplished by searching for neuron i^* whose weight vector is closest to the current input vector v:

$$\|\mathbf{w}_{i^*} - v\| \leq \|\mathbf{w}_i - v\|, \quad \forall i \in A. \tag{1}$$

0-7803-2739-X/95 $4.00 © 1995 IEEE

To achieve a topology-preserving mapping, incremental weight adjustments are performed not only of the winning neuron but also of its neighboring neurons, using a neighborhood function Λ:

$$\Delta \mathbf{w}_i = \eta \Lambda(i, i^*, t)(v - \mathbf{w}_i), \quad \forall i \in A. \tag{2}$$

The learning rate η is usually gradually decreased over time t to zero. The neighborhood function Λ is 1 for $i = i^*$ and falls off with distance between i and i^* in lattice coordinates. The range spanned by the neighborhood function decreases over time until only the weight vector of the winner is updated (Winner-Take-All, WTA), and the Kohonen rule is identical to the standard unsupervised competitive learning rule (standard UCL).

There are a number of problems with the Kohonen algorithm, however. Firstly, a too fast decrease of the range spanned by the neighborhood function leads to topological defects such as kinks in the one-dimensional case and twists in the two-dimensional case [11] which are difficult to iron out, if at all. Secondly, due to this range, the error function which is minimized is no longer quadratic since, *e.g.* in the one-dimensional case, the exponent 2 is replaced by $r = \frac{1}{2} + \frac{3}{2(2n+1)^2}$ with n the number of neighbor neurons that are taken into account on each side of the winner [13]. Thirdly, Ritter and Schulten [14] have shown that, also for the one-dimensional case, the weight density is proportional to $p^{\frac{2}{3}}$, with p the input probability density function (*p.d.f.*). An ideal map, ideal in terms of resource usage, would have the weight density proportional to p (equiprobable map) and the Kohonen algorithm is unable to achieve this.

In this article, a new unsupervised competitive learning rule is introduced for topology-preserving map formation. The aim is to achieve with this map an equiprobable quantization of the input space. The rule does not require a neighborhood function as does the Kohonen rule and its many variations. Rather, it achieves a **globally-ordered** map by performing **local weight updates** only. We will compare its dynamics –which is completely different– with that of the Kohonen algorithm. The physiological interpretation and the complete formalization of the rule will be addressed elsewhere.

EQUIPROBABLE QUANTIZATION

Consider a lattice A of N formal neurons. The formal neurons quantize the input space V into partition cells or quantization regions. In case of the Kohonen algorithm, the quantization regions are defined by eq. (1) and thus are always disjoint. In our case, quantization regions are defined in a different way and are not necessarily disjoint when the map is not globally ordered (tangled). Assume that V and A have the same dimensionality d and that A comprises a regular d-dimensional grid with a rectangular topology. Since the topology is rectangular, the grid consists of a number of d-dimensional hypercubes (in topological sense in terms of grid coordinates) and these hypercubes define the quantization regions. The hypercubes $H_a, H_b, ..., H_i$ in Fig. 1B represent the quantization regions of the lattice portion shown in Fig. 1A. We also consider the "imaginary" hypercubes (unbounded quantization regions) adjacent to the outer border and vertices of A: the hypercubes

H_a, H_b, H_c, H_d, H_g are constructed by extending the lattice towards infinity. The links (full lines) separating the "real" hypercubes (bounded quantization regions) that share a common vertex on the border of A, are extended towards infinity (stippled lines). In this way, every neuron j of A is a common vertex to 4 adjacent hypercubes.

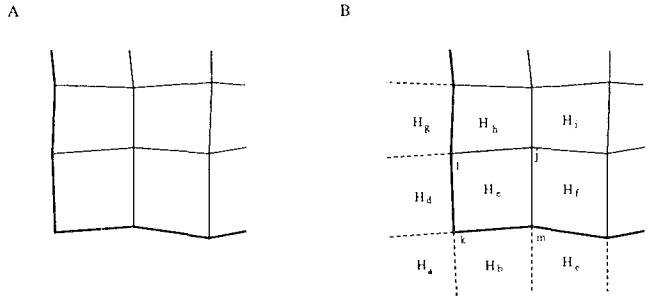


Figure 1: Definition of quantization region. (A) Portion of a lattice A with a rectangular topology, represented in V space. The bold line shows the outer border of the lattice. (B) The same portion of the lattice but with some of the neurons and hypercubes labeled. See text.

Following our reasoning, every hypercube defines a quantization region and a neuron will be *activated* if the input v activates one of its 2^d adjacent d-dimensional hypercubes. We will assume that $p(v)$ is a continuous $p.d.f.$: in this way, the boundaries of the hypercubus will have zero area and hence, there will be a zero probability that a single v will activate two or more adjacent hypercubes unless these hypercubes overlap.

Given that we have N neurons in the grid, we want that every neuron is activated with the same probability $\frac{1}{N}$. In other words, we want to establish an equiprobable quantization of the (joint) input $p.d.f.$ (In fact this is an approximation since such a partitioning may not exist for a given input $p.d.f.$) This is achieved by performing iterated changes in the weight vectors $\mathbf{w}_j$ corresponding to the vertices of the active hypercubes. An equiprobable quantization is then achieved when each neuron is *updated*, and thus *activated*, with equal probability. Finally, since information-theoretic entropy maximization and equiprobable quantization are equivalent, our rule is called Maximum Entropy learning Rule (MER).

MAXIMUM ENTROPY LEARNING RULE

For the sake of exposition, consider again the 2-dimensional case (Fig. 1). There are two ways to update the vertices of an active hypercube. Firstly, we randomly select only one of them, say i^*. The weight vector of neuron i^* is then updated as follows:

$$\Delta \mathbf{w}_{i^*} = \eta Sign(v - \mathbf{w}_{i^*}), \tag{3}$$

with $Sign(.)$ the sign function. Secondly, a much faster method is to update all the vertices of an active hypercube, scaled according to the number of

vertices. Assume that the input sample $v \in H_e$ (Fig. 2A). The weight vectors of neurons j, k, l, m are updated as follows:

$$\Delta \mathbf{w}_{i^*} = \frac{\eta}{n_{H_e}} Sign(v - \mathbf{w}_{i^*}), \forall i^* \in \{j, k, l, m\}, \tag{4}$$

with n_{H_e} the number of vertices of H_e, *i.e.* 4. As a result of this, neurons j, k, l, m will move with a fixed step size of $\frac{\eta}{4}$ in both V-dimensions towards v as shown in Fig. 2A (stippled lines). Similarly, in case v lies outside the lattice A, 2 neurons will be updated with step size $\frac{\eta}{2}$ if v faces the border of A (Fig. 2B) and 1 neuron with step size η if it faces one of the four corners of A (Fig. 2C).

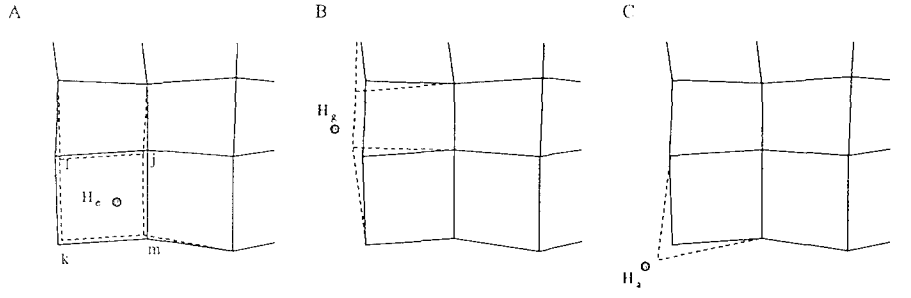


Figure 2: Update of neuron weight vectors as a function of active hypercubes. The full and stippled lines represent the lattice before and after the update (not to scale). The current input (black dot) activates hypercube H_e inside the lattice (A), or imaginary hypercubes H_g (B) and H_a (C) outside the lattice. The construction of the imaginary hypercubes is explained in Fig. 1.

In order to formalize MER for the general d-dimensional case where we update every vertex of an active hypercube, we define $\mathbb{1}_{H_j}(v)$, $j = 1, ..., Q$, as the code membership functions of the Q hypercubes of A:

$$\mathbb{1}_{H_j}(v) = \begin{cases} \frac{1}{n_{H_j}} & \text{if } v \in H_j \\ 0 & \text{if } v \notin H_j. \end{cases} \tag{5}$$

We assume that $p(v)$ is a continuous $p.d.f.$ hence, the probability that v falls on one of the links equals zero. Define S_i as the set of 2^d hypercubes that have neuron i as a common vertex. The d-dimensional MER rule then becomes:

$$\Delta \mathbf{w}_i = \eta \sum_{j \in S_i} \mathbb{1}_{H_j}(v) Sign(v - \mathbf{w}_i), \quad \forall i \in A. \tag{6}$$

Proposition 1: The average of eq. (6), with the average taken over the environment V, performs (stochastic) gradient descent on the cost function:

$$E = \sum_{i=1}^{N} \sum_{j \in S_i} \mathbb{1}_{H_j}(v) \mid v - \mathbf{w}_i \mid \tag{7}$$

with $\mid v - \mathbf{w}_i \mid$ denoting the (per letter) absolute value of $v - \mathbf{w}_i$.

Proof: Following the definition of gradient descent, the weights are updated so as to reduce the cost term introduced by the active hypercube:

$$< \Delta \mathbf{w}_i >_V = -\eta \frac{\partial E}{\partial \mathbf{w}_i} = \sum_{j \in S_i} \mathbb{1}_{H_j}(v) Sign(v - \mathbf{w}_i), \quad \forall i \in A, \tag{8}$$

and the latter exactly corresponds to the right hand side of eq. (6). QED.

Due to the latter proposition, we have that eq. (7) is a Liapunov function. Furthermore since E is radially unbounded, convergence will hold "in the large": when $w_{ij} \rightarrow \infty \Rightarrow E \rightarrow \infty$, $i = 1, ..., N$, $j = 1, ..., d$.

Proposition 2: In the one-dimensional case, MER is guaranteed to converge on average to an equiprobable quantization.

Proof: Assume a 1-dimensional grid with quantization intervals $H_1, ..., H_{N+1}$ separated by the weights $w_1, ..., w_N$. Due to the existence of a Liapunov function, we know that MER will converge on average. We have that:

$$< \Delta w_i >_V = 0 =< H_i - H_{i+1} >_V = p(H_i) + p(H_{i+1}), \quad i = 1, ..., N, \tag{9}$$

at convergence. If we substitute backwards the H_is in the equations, then we observe that the latter all equal the same value. Now since by definition we have that neuron i is activated when either H_i or H_{i+1} or both are activated, the probability that neuron i is activated is simply: $p(i) = p(H_i) + p(H_{i+1})$. If we now substitute for the H_is, we obtain that: $p(1) = ... = p(i) = ...p(N)$, and thus an equiprobable quantization. QED.

We now show that the grid cannot be folded at convergence (kinks).

Proposition 3: In the one-dimensional case, MER is guaranteed to converge on average to a 1-dimensional grid without kinks.

Proof: Assume the inverse that a 1-dimensional grid with at least one kink is a stable solution. Assume that there is a kink at neuron i, hence, the intervals H_i and H_{i+1} overlap and are located at the same side of i. This means that neuron i will only get weight updates in the direction of both overlapping intervals, and never in the opposite direction. This means that neuron i will keep on shifting until the overlap is removed. Hence, the kink at neuron i is not a stable solution. QED.

Since kinks are the only topological defects of 1-dimensional grids, the latter two propositions also signify that MER converges to a unique equiprobable quantization with $p(1) = ... = p(i) = ...p(N) = \frac{1}{N}$.

Finally, it is noted that MER in the one-dimensional case resembles our previously introduced Boundary Adaptation Rule (BAR) [15,16,17]. However, there is one crucial difference: BAR achieves an equiprobable quantization in terms of the quantization regions and MER achieves this in terms of the neurons themselves. Hence both rules are complementary.

CONVERGENCE- AND DYNAMICAL PROPERTIES

Up to now we have considered globally-ordered lattices to explain our rule. What happens in case the lattice is tangled and V is quantized by possibly

overlapping hypercubes? In other words, how does it come that by performing local updates only, MER is able to generate globally-ordered lattices while standard UCL is not? This point is most easily explained graphically.

Consider the lattice of Fig. 3A: hypercube *befw* overlaps with hypercube *abcd* and hence, the lattice is not locally ordered. Since neuron w is closest to the input (black dot), it wins the competition. As a result, neuron w moves towards the input and the overlap between hypercubes *abcd* and *befw* increases. Consider now the case of MER in Fig. 3B. The input activates hypercube *abcd*, but not hypercube *befw*. As a result, neurons a, b, c, d move towards the input, but not neuron w and, in this way, the overlap between hypercubes *abcd* and *befw* decreases. Hence, MER is able to untangle the lattice locally.

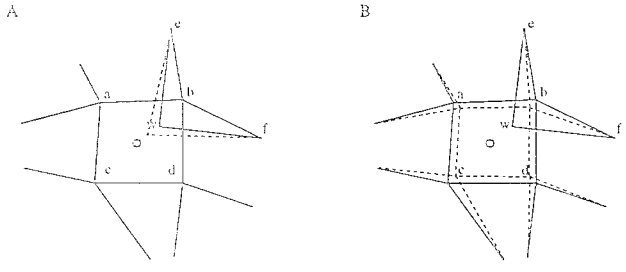


Figure 3: Untangling a locally-distorted lattice in case of standard UCL (A), and MER (B). Shown is a portion of the lattice (full lines) with vertices a, b, c, d, e, f, w. The present input sample is indicated by the black dot. The stippled lines show the lattice after the weight vectors are updated.

There are three dynamical properties that distinguish MER from the Kohonen rule. Firstly, in MER, the magnitude of the weight update vectors Δw_i is fixed. Secondly, MER does not need a shrinking neighborhood function to achieve globally-ordered lattices. In fact, purely local weight updates suffice. This will be shown in the next section. Thirdly, MER allows several quantization regions to be activated at the same time. This will happen when the quantization regions overlap and, thus, the lattice is tangled. Therefore, in case we have unique winners during an increasing number of subsequent time steps, confidence grows that the lattice is completely untangled and thus globally ordered. Hence, by monitoring the number of active quantization regions per time step we can decide when the untangling phase has ended.

SIMULATION RESULTS

In this section, we will first show the aforementioned dynamical properties of MER, and compare them with those of Kohonen's rule. We will consider the standard case of mapping a 2-dimensional input space onto a 24×24 planar lattice. The input samples $v = (v_1, v_2)$ are chosen randomly from a 2-dimensional uniform *p.d.f.* $p(v)$ within the square $\{0 \leq v_1 < 1, 0 \leq v_2 < 1\}$. The initial weight vectors are also randomly drawn from this distribution.

Since MER and the Kohonen rule are different, we have to devise a way to

compare them on an equal basis. First we will use a fixed but small learning rate η. For MER we take $\eta = 0.001$ and for the Kohonen rule $\eta = 0.015$ so that the variance on the weight vectors is of comparable magnitude at convergence. (This is a compromise since slightly lower η values for the Kohonen rule lead to topological defects.) The following neighborhood function is used for the Kohonen rule:

$$\Lambda(i, i^*, t) = \exp(-\frac{(r_i - r_{i^*})^2}{2\sigma(t)^2}), \text{ with } \sigma(\mathrm{t}) = \sigma_0 \exp(-2\sigma_0 \frac{\mathrm{t}}{\mathrm{t_{max}}}), \qquad (10)$$

where r_i and r_{i^*} represent the lattice coordinates of i and i^*, t the present time step, $t_{max} = 4,000,000$ the maximum number of time steps, and σ_0 the range spanned by the neighborhood function at $t = 0$; $\sigma_0 = 5$. Due to this neighborhood function, all neurons are updated at each time step, albeit with a smaller value as time progresses. In case of MER, only 1, 2 or 4 neurons are updated per active hypercube, and several hypercubes may be active at the same time.

The simulation results are shown in Figs. 4 and 5. We observe that the dynamical behavior of MER is completely different. The Kohonen rule first leads to a contraction of the lattice due to the initially large range spanned by the neighborhood function. The lattice rapidly untangles and becomes fairly evenly distributed and then adapts in detail to the input *p.d.f.* In case of MER, the lattice gradually adapts in detail to the input *p.d.f.* as it becomes untangled. We do not observe a comparable initial contraction of the lattice. Furthermore, we have that several hypercubes (*i.e.* quantization regions) are activated at the same time and that this number decreases as the lattice converges to a globally-ordered one (Fig. 6). Notice that for the Kohonen rule, only a single quantization region wins the competition at each time step.

The Kohonen rule definitely is faster, however if we compare the sum of all weight update vector magnitudes performed in t_{max} time steps, then the picture is different: the sum equals 4181 for MER and 27555 for the Kohonen rule. A fair comparison would be to shorten t_{max} for the Kohonen rule so that it yields a weight update sum similar to MER's (*i.e.* $t_{max} = 275,000$). The results for the Kohonen rule are given in Fig. 7. However, we now observe that the lattice is twisted. Hence, the range of the neighborhood function is too rapidly decreased, given η.

Finally, we will consider another type of input distribution. In the previous case the distribution was uniform within a square, hence, the type of solution found by both learning rules is the same. We will use a radial distribution of input samples $v = (r, \theta)$ with r and θ randomly and independently chosen within the intervals $[0, 0.5)$ and $[0, 2\pi)$. The resulting *p.d.f.* has a radial distribution $\propto \frac{1}{r}$. We will display the results for the converged lattices (Fig. 8A,B) in terms of the average probabilities that the neurons are updated over a period of 100,000 iterations. The average probabilities are represented as gray scales: black denotes zero probability and white denotes the maximum average probability found in both lattices (Fig. 8C,D). We immediately observe that the Kohonen rule leads to a lattice which undersamples the high probability region at the center (high update probability)

and oversamples the low probability region away from the center (low update probability) (Fig. 8C), as expected. Since the input space is almost uniform gray for MER (Fig. 8D), we conclude that MER leads to an almost equiprobable quantization.

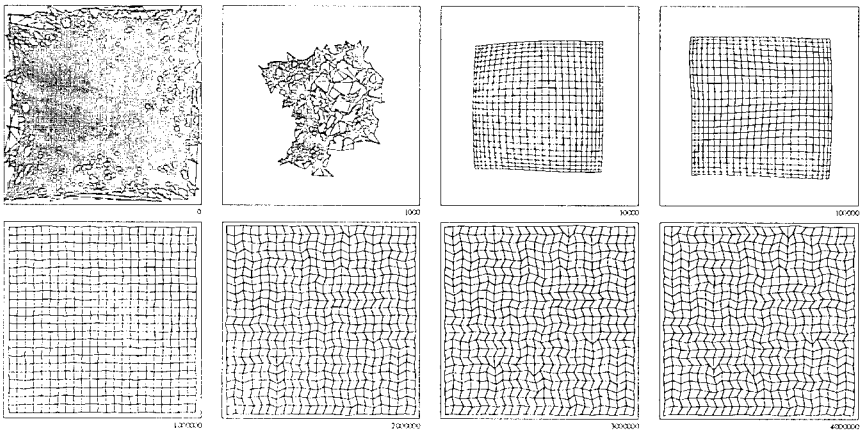


Figure 4: Evolution of a 24 × 24 lattice with a rectangular topology as a function of time in case of the Kohonen rule with $t_{max} = 4,000,000$. The outer squares outline the uniform input *p.d.f.* The values given below the squares represent time. Notice that at convergence, the weight vectors stabilize into a noisy state as a result of the vanished neighborhood function and the fixed but small learning rate η.

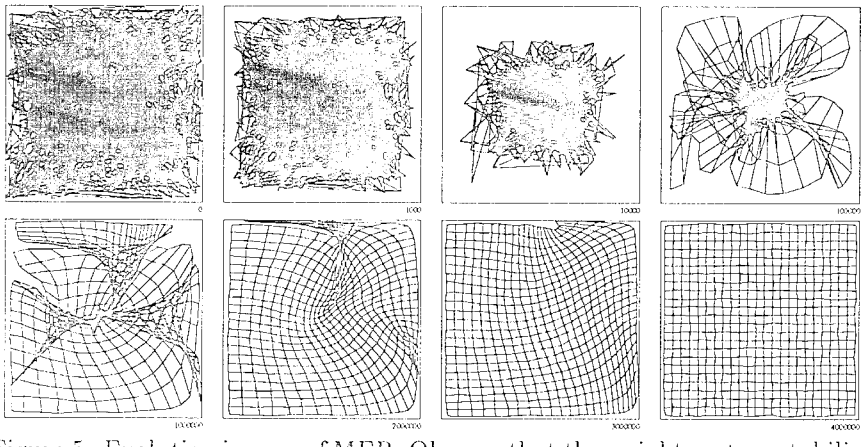


Figure 5: Evolution in case of MER. Observe that the weight vectors stabilize into a more favorable, more regular state (*cf.* Fig. 4).

DISCUSSION

We have introduced a new unsupervised competitive learning rule for topology-preserving map formation and vector quantization. The rule, called MER, yields an equiprobable quantization of the input space. Its main fea-

ture is that it is able to achieve a **globally-ordered** map by performing **local weight updates** only. Contrary to the Kohonen rule, we do not need a neighborhood function. Hence, MER is an even simpler rule than Kohonen's: 1) we only need to update locally, hence only local neural communications are needed, and 2) only one parameter, the learning rate, has to be chosen. The effect of the latter is much clearer understood than that of the parameters needed in Kohonen's rule to specify the neighborhood function. Especially the choice of the rate at which the range of the neighborhood function decreases is often a critical element for convergence [6,11].

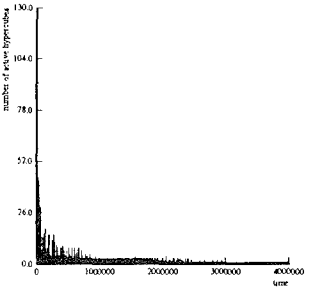


Figure 6: Number of active hypercubes (quantization regions) as a function of time. Notice the initially large number of active hypercubes (up to 130!).

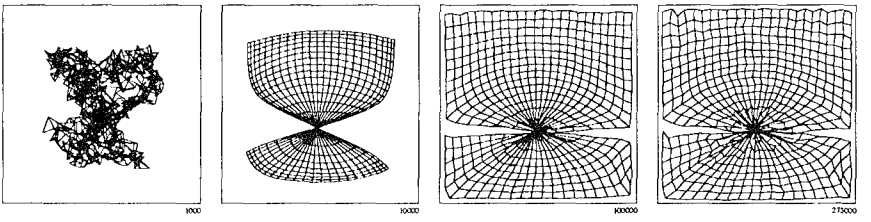


Figure 7: Evolution in case of the Kohonen rule with $t_{max} = 275,000$.

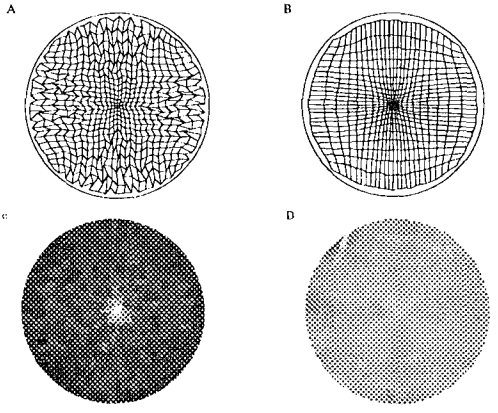


Figure 8: Radial input distribution. Converged lattices obtained with the Kohonen rule (A) and MER (B). Graphical rendition of the corresponding average update probabilities for the Kohonen rule (C) and MER (D).

ACKNOWLEDGEMENT

The author is a senior research associate of the National Fund for Scientific Research (Belgium).

REFERENCES

[1] S. Grossberg, "Adaptive pattern classification and universal recoding: I Parallel development and coding of neural feature detectors," Biol. Cybern., vol. 23, pp. 121-134, 1976.

[2] C. von der Malsburg and D.J. Willshaw, "How to label nerve cells so that they can interconnect in an orderly fashion?," Proc. Natl. Acad. Sci. USA, vol. 74, pp. 5176-5178, 1977.

[3] K.J. Overton and M.A. Arbib, "The branch arrow model of the formation of retino-tectal connections," Biol. Cybern., vol. 45, pp. 157-175, 1982.

[4] T. Kohonen, "Self-organized formation of topologically correct feature maps," Biol. Cybern., vol. 43, 59-69, 1982.

[5] T. Kohonen, Self-organization and associative memory, Berlin: Springer, 1989.

[6] H. Ritter, T. Martinetz and K. Schulten, Neural computation and self-organizing maps: An introduction, Reading, Mass: Addison-Wesley, 1992.

[7] H. Ritter and K. Schulten, "Kohonen's self-organizing maps: Exploring their computational capabilities," Proc. IEEE Intl. Conf. on Neural Networks, San Diego, 1988, vol. I, pp. 109-116.

[8] H. Ritter and K. Schulten, "Convergence properties of Kohonen's topology conserving maps: Fluctuations, stability and dimension selection," Biol. Cybern., vol. 60, pp. 59-71, 1988.

[9] M. Cottrell and J.C. Fort, "A stochastic model of retinotopy: A self-organizing process," Biol. Cybern., vol. 53, pp. 405-411, 1986.

[10] T.M. Heskes and B. Kappen, "Error potentials for self-organization," Proc. IEEE Intl. Conf. on Neural Networks, San Francisco, 1993, pp. 1219-1223.

[11] T. Geszti, Physical models of neural networks. Singapore: World Scientific Press, 1990.

[12] T. Kohonen, "Physiological interpretation of the self-organizing map algorithm," Neural Networks, vol. 6, pp. 895-905, 1993.

[13] H. Ritter, "Asymptotic level density for a class of vector quantization processes," IEEE Trans. on Neural Networks, vol. 2, no. 1, pp. 173-175, 1991.

[14] H. Ritter and K. Schulten, "On the stationary state of Kohonen's self-organizing sensory mapping," Biol. Cybern., vol. 54, pp. 99-106, 1986.

[15] M. M. Van Hulle and D. Martinez, "On a novel unsupervised competitive learning algorithm for scalar quantization," IEEE Trans. on Neural Networks, vol. 5 (3), pp. 498-501, 1994

[16] M.M. Van Hulle and D. Martinez, "On an unsupervised learning rule for scalar quantization following the maximum entropy principle," Neural Computation, vol. 5, pp. 939-953, 1993.

[17] D. Martinez and M.M. Van Hulle, "Generalized Boundary Adaptation Rule for minimizing r-th power law distortion in the high resolution case," Neural Networks, 1995, in press.

A Self-Organizing System for the Development of Neural Network Parameter Estimators

M.T. Manry
Department of Electrical Engineering
University of Texas at Arlington
Arlington, Texas 76019

I. Introduction

The design an optimal neural network estimator from training data is difficult because (1) the required complexity of the estimation network is unknown, (2) existing training algorithms for multilayer perceptrons (MLPs) are inefficient, in terms of training time and use of free parameters, (3) existing bounds on neural network estimation error assume noiseless inputs and are not practical to calculate, (4) there is no generally accepted procedure for finding the best subset of input features to be used in optimal estimation, and (5) a method for automatically developing optimal estimators from training data is not available.

In this paper, we describe a methodology for attacking these problems. In section II, we describe three separate processing blocks which attempt to solve problems (1), (2), and (3). In section III, these blocks are then assembled into larger compound systems or blocks which attempt to solve the remaining problems. Examples of multilayer perceptron (MLP) estimators, designed using the proposed system, are given in section IV.

II. Algorithmic Building Blocks

Building block algorithms have been developed for (1) determining an estimator's required complexity or size from training data, (2) efficient training, and (3) calculation of Cramer Rao lower bounds on estimation variance from training data and a signal model. These blocks are discussed in the remainder of this section.

A. Complexity Estimation

Several facts make complexity estimation possible and worthwhile; (1) training of a nearest neighbor estimator (NNE) is almost an order of magnitude faster than MLP training algorithms, (2) once trained, the MLP can be applied to data one or more orders of magnitude faster than the NNE, (3) an MLP can closely approximate the performance of a NNE if it can memorize the NNE's

0-7803-2739-X/95 $4.00 © 1995 IEEE

cluster vectors, and (4) a one-output MLP can memorize as many patterns as it has free parameters [1] (its complexity). This result is a great improvement over the much lower bounds, related to the number of hidden units, that are given in [2].

As a first step in the complexity estimation algorithm, we iteratively train an NNE through clustering of the input vectors. These clusters, and their associated outputs, constitute a reduced-size training set similar to that described in [3]. In [3], however, the input vectors are required to be noiseless, useless inputs are not rejected, and there is no theoretical relationship between number of chosen patterns and MLP network size. Second, the required complexity of the neural net to memorize the clusters is determined. As the number of clusters increases, and the performance of the NNE improves, the predicted complexity of the MLP, required to attain the error performance of the NNE, also increases. Picking a suggested network configuration from complexity estimation is one or more orders of magnitude faster than actually training multiple MLPs. Details of our algorithm can be found in [4].

B. Efficient Training of Estimators

1. Motivation

Current MLP training algorithms, such as output weight optimization (OWO) [5], conjugate gradient (CG) training [6], and backpropagation (BP) are not capable of training the MLP up to its maximum potential in a reasonable amount of time [1]. Consider a MLP with structure *8-36-1* (*8* inputs, *36* hidden units, and *1* output), which is trained to memorize varying numbers of random patterns, using BP, OWO, CG, and a method in which an *8*-variable *3*-rd degree polynomial or Volterra filter is trained and then efficiently mapped to the sigmoidal MLP. This mapping procedure first maps the Volterra filter to a polynomial network in which most of the units have third degree polynomial activations of the form $a \cdot x^2 + b \cdot x^3$ where x denotes the net function. Second, each polynomial unit is replaced by a single sigmoidal unit, with some changes to the weights of course. This second step is possible because the ratio $S''(x)/S'''(x)$, of the

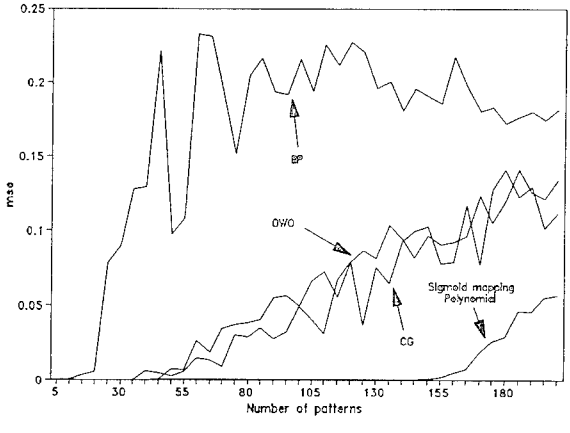


Figure 1. Training Error versus Number of Patterns for Four Training Algorithms

sigmoid's second to third derivatives, is continuously variable between $-\infty$ and $+\infty$. In Fig. 1, the networks' mean square training errors are plotted versus the number of training patterns used. Here, OWO and CG perform better than BP but lag far behind the mapping method, which leads to very parsimonious networks. Clearly, existing training algorithms can be improved if they can be given the capability of precisely adjusting $S''(x)/S'''(x)$.

2. Training Algorithm

Motivated by the previous example, our basic training approach consists of (1) training a polynomial-activation MLP via OWO [5], (2) converting it to a sigmoidal-activation MLP, and then (3) training the sigmoidal-activation MLP via OWO. The conversion to sigmoid activations is done because the sigmoid function is bounded, sigmoid activations can mimic polynomial activations of degree much higher than three, and the performance of the converted network is theoretically as good as or better than that of the polynomial network. Consider the polynomial activation function,

$$P(x) = a \cdot x^2 + (1-a) \cdot x^3 \tag{1}$$

where x denotes the net function. We want to approximate a polynomial unit, having activation P(x), by a sigmoid-activation sub-net having the activation

$$P_s(x) = d + w_b \cdot x + w_o \cdot S(c + w_i \cdot x) \tag{2}$$

This sub-net is shown in Fig. 2.

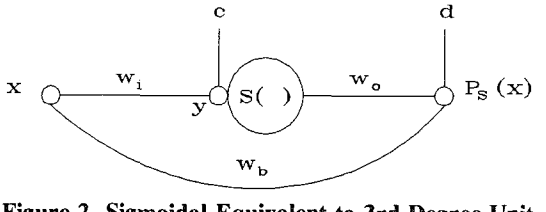


Figure 2. Sigmoidal Equivalent to 3rd-Degree Unit

Let m_x and σ_x respectively denote the mean and standard deviation of the net function x. Let y_o and σ_y respectively denote the mean and the desired standard deviation of the sigmoid's net function y. The weight w_i is found from $w_i = \sigma_y/\sigma_x$. Then $y_o = w_i m_x + c$. Equating $P''(x)/P'''(x)$ to $P_s''(x)/P_s'''(x)$ at $x = m_x$, we get

$$\frac{a}{1-a} = -3m + \frac{3S''(y_o)}{w_i \cdot S'''(y_o)} \tag{3}$$

We can solve for y_o by the Newton Raphson method. Then c can be obtained from $c = y_o - w_i m_x$. The remaining weights and threshold are easily found.

C. Bounding Estimation Training Error

The *Bound Calculation* block calculates the Cramer-Rao maximum a-posteriori (MAP) lower bounds on the estimation training error variance. Given the bounds, we know (1) how close our estimator is to being optimal [7,8], and (2) whether or not training should be stopped. The bounds are not widely used in the design of estimation algorithms because (1) an accurate, differentiable statistical signal model is often unavailable, and (2) the bounds both exist and are tight only for Gaussian parameter vectors. The signal modeling problem is addressed in section III. The second problem has been addressed in [9].

Let (x_p,θ_p) p = 1, 2, ..., N_v represent the training set for a MLP. Here, x_p represents the pth example of the random input vector x and θ_p represents the pth example of the random parameter vector θ. Our ultimate goal is to design an MLP which almost optimally estimates θ from x. In [7], we motivated minimum mean square estimation via the MLP by showing that the training error for the MLP is minimized when the MLP outputs equal $E[\theta|x]$. This quantity, which is denoted by θ_{MMS}, is the minimum mean-square estimate of θ.

Cramer-Rao MAP bounds on the error variance of θ_{MMS} are found from the Fisher information matrix (FIM), which is found from the log likelihood functions [10], $\Lambda^{MAP} = \ell n(p_{\theta|x}(\theta|x))$, $\Lambda^{MLE} = \ell n(p_{x|\theta}(x|\theta)$, and $\Lambda^{AP} = \ell n(p_\theta(\theta))$. Elements of the FIM, J_θ^{MAP}, are defined as [10]

$$J_{ij}^{MAP} = E_\theta[E_x[\frac{\partial\Lambda^{MLE}}{\partial\theta_i}\frac{\partial\Lambda^{MLE}}{\partial\theta_j}]] + E_\theta[\frac{\partial\Lambda^{AP}}{\partial\theta_i}\frac{\partial\Lambda^{AP}}{\partial\theta_j}] \tag{4}$$

Let x_p be modelled as $s_p + n_p$ where s_p denotes the pth noiseless signal component of x_p. Let $s_p(m)$ denote the mth element of s_p. Assuming that the noise covariance matrix is diagonal, we numerically evaluate elements of the FIM in (4) as

$$J_{ij}^{MAP}(\theta) = \frac{1}{N_v}\sum_{p=1}^{N_v}\left[\sum_{l=1}^{k}\frac{\partial s_p(l)}{\partial\theta_i}\cdot d(l,l)\cdot\frac{\partial s_p(l)}{\partial\theta_j}\right] + b(i,j)$$

where b(i,j) denotes an element of the inverse parameter vector covariance matrix C_θ and d(i,j) denotes an element of the inverse noise covariance matrix C'^{-1}_{nn}.

In order to calculate the log-likelihood functions and the Cramer-Rao lower bounds, a statistical model of the input vector x is required which consists of an expression s in terms of θ and the joint probability densities of θ and n.

III. Assembly of the Building Blocks

In this section we use the three building blocks from section II to construct a signal modelling block. Using this block and the ones described in section II, we construct a feature selection system for estimators and a self-

organizing estimator.

A. Signal Modelling

It is possible to find a statistical signal models from training data, given that the mapping or estimator is injective and that the inputs and outputs have the appropriate statistical variation [8]. A mapping is injective if only one input vector maps to each possible output vector. Given the training patterns we want to find the signal component model and noise pdf. We make the following assumptions.

(A1) The exact signal model is $\mathbf{x}_p = \mathbf{s}_p + \mathbf{n}_p$.

(A2) The elements $\theta_p(k)$ of $\boldsymbol{\theta}_p$ are statistically independent.

(A3) The noise vector $\mathbf{n}$ has independent elements with a jointly Gaussian pdf.

(A4) An expression exists for $\mathbf{s}$ in terms of $\boldsymbol{\theta}$.

The signal model of (A1) can be rewritten as $\mathbf{x}_p = \mathbf{s}'_p + \mathbf{n}'_p$ where $\mathbf{s}_p'$ and $\mathbf{n}_p'$ denote approximations to $\mathbf{s}_p$ and $\mathbf{n}_p$ respectively. The calculation of $\mathbf{s}_p'$ and $\mathbf{n}_p'$ are described separately. Assume that the nth element of the approximate model $\mathbf{s}_p'$ is represented by

$$s_p'(n) = \sum_{k=1}^{L} a_{nk} \cdot T_p(k) \tag{5}$$

where a_{nk} denotes the coefficient of $T_p(k)$ in the approximation to s(n), and where $T_p(k)$ is the kth basis function calculated from the desired pth output vector $\boldsymbol{\theta}_p$. $T_p(k)$ can represent a multinomial function of parameter vector $\boldsymbol{\theta}$ in a functional link network, or a hidden unit output in a MLP. In practice, because of the capabilities of the MLP for approximating derivatives [11,12], a neural network would be the first choice for $\mathbf{s}_p'$. The error between $x_p(n)$ and its model is measured as

$$E_p(n) = \frac{1}{N_v} \sum_{p=1}^{N_v} [x_p(n) - s_p'(n)]^2$$

The model is determined from the noisy data by setting the partial derivative

$$\frac{\partial E_p(n)}{\partial a_{nk}} = \frac{-2}{N_v} \sum_{p=1}^{N_v} [s_p(n) - s_p'(n)] \frac{\partial s_p'(n)}{\partial a_{nk}} + e_{nk} ,$$

$$e_{nk} = \frac{2}{N_v} \sum_{p=1}^{N_v} n_p(n) \frac{\partial s_p'(n)}{\partial a_{nk}}$$

equal to zero. Using the facts that

$$\frac{\partial s_p'(n)}{\partial a_{nk}} = T_p(k), \qquad E[n_p(n) n_q(n)] = \sigma_n^2 \cdot \delta(p-q)$$

the mean-square of the noise term e_{nk} is evaluated as

$$E[e_{nk}^2] = \frac{4}{N_v^2} \sum_{p}^{N_v} \sum_{q=1}^{N_v} E[n_p(n)n_q(n)]T_p(k)T_q(k)$$

$$= \frac{4\sigma_n^2}{N_v^2} \sum_{p=1}^{N_v} (T_p(k))^2 = \frac{4\sigma_n^2 \circ E_T(k)}{N_v}$$

where $E_T(k)$ is the average energy of the kth basis function. Note that the mean-square error goes to zero in the limit as the number of training vectors increases.

Given a model s_p' for the signal component, we model the mean vector and covariance matrices of the noise component as

$$m_n' = \frac{1}{N_v} \sum_{p=1}^{N_v} x_p - s_p' ,$$

$$C_{nn}' = \frac{1}{N_v} \sum_{p=1}^{N_v} (x_p - s_p' - m_n')(x_p - s_p' - m_n')^T$$

Next, a reasonable pdf for the parameter vector θ is determined. We determine an approximate Gaussian pdf for θ by estimating its mean vector and covariance matrix from the desired outputs, θ_p, in the training data file. Since equation (5) represents a MLP or Volterra filter, our *Signal Modelling* block uses the complexity estimator and efficient training.

B. Feature Selection

In theory, estimation performance improves as the size of the observation vector x increases. In practice however, larger observation vectors are not desirable because of (1) the longer training times required for the estimation algorithm, (2) the increased time necessary to apply the estimator to data, (3) instabilities or poor performance of the estimator because of linear dependence of elements in the observation vector, and (4) increased expense required when more observations are taken (extra weight of the sensors, extra money required for development). Therefore, it is useful perform feature selection, which is the process of finding useful subsets of the available observation vector x which lead to good estimator performance.

A methodology for comparing inputs, in order to determine their optimality relative to each other, is given in [7,8]. Feature selection requires three blocks. These include the signal modelling and bound calculation blocks. In addition we need a conventional subsetting block. The subsetting block will merely add features to the subset which most reduce a weighted sum of the MAP bounds on parameter estimation error variance. Features at the top of the list are the most important ones.

C. Self-Organizing Estimator

The *Self-Organizing Estimator* that we have developed requires three blocks. These are feature selection to find a good feature subset, complexity estimation to determine which size MLPs to use in signal modelling and in the final estimator, and finally efficient training. The feature selection calculates the Cramer-Rao MAP lower bounds on estimation error variance, which can be summed to form a lower bound on the estimator's training error. Our efficient training algorithm halts when the training error approaches the bound.

IV. Examples

As a first example, we chose the task of inverting the surface scattering parameters from an inhomogeneous layer above a homogeneous half-space, where both interfaces are randomly rough. The parameters to be inverted are the effective permittivity of the surface ε, the normalized rms height $k\sigma$ (upper surface $k\sigma_1$, lower surface $k\sigma_2$), the normalized surface correlation length kL (upper surface kL_1, lower surface kL_2), where k is the wavenumber, the optical depth τ, and single scattering albedo ω of an inhomogeneous irregular layer above a homogeneous half space from backscattering measurements [13,14].

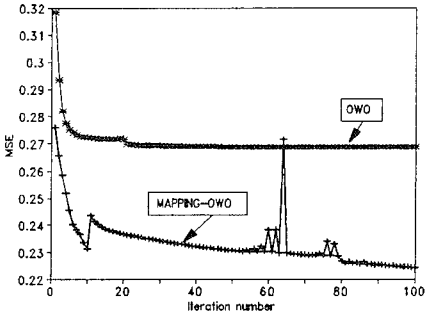


Figure 3. Training Results for Example 1.

Table 1. Bounds and Training MSE for Example 1.

Par.	ε	$k\sigma_1$	$k\sigma_2$	kL_1	kL_2	τ	ω
Bound	9.85 x 10^{-3}	8.52 x 10^{-5}	2.43 x 10^{-1}	4.02 x 10^{-5}	1.97 x 10^{-2}	8.85 x 10^{-3}	6.24 x 10^{-7}
MSE	1.125 x 10^{-2}	8.7 x 10^{-5}	1.65 x 10^{-1}	1.08 x 10^{-4}	2.19 x 10^{-2}	2.48 x 10^{-2}	8.01 x 10^{-8}

The training data for the MLP network contained 1768 patterns. The inputs consisted of eight theoretical values of backscattering coefficient parameters σ^o at V and H polarizations and four incident angles (10°, 30°, 50°, 70°). The outputs were the corresponding values of ε, $k\sigma_1$, $k\sigma_2$, kL_1, kL_2, τ, and ω, which had a jointly uniform probability density. The self-organizing estimator (1) chose a signal modeling MLP with the structure 7-24-12-8, (2) generated the seven

bounds on the estimation error variances shown in Table 1, (3) chose the MLP structure 8-25-7 for the estimation MLP, and (4) had the training errors shown in Fig. 3 and Table 1. In the figure, training errors are plotted for the new procedure for $\sigma_y = 1.0$ and for a sigmoid network initialized with random initial weights. The discontinuity in the new procedure's error curve dissapears when σ_y approaches 0, but training is difficult for that case. From the table, most of the parameters' training errors are slightly greater than the corresponding bounds, as one would expect.

The second data set has 16 inputs and 3 outputs and represents the training set for inversion of surface permittivity ε, the normalized surface rms roughness $k\sigma$, and the surface correlation length kL found in backscattering models from randomly rough dielectric surfaces [15,16]. In contrast to the first data set, no volume scattering related parameters are considered. The first eight of the sixteen inputs represent the simulated backscattering coefficient measured at 10, 30, 50 and 70 degrees at both vertical and horizontal polarizations. The remaining eight are various combinations of ratios of the original eight values. These ratios correspond to those used in several empirical retrieval algorithms.

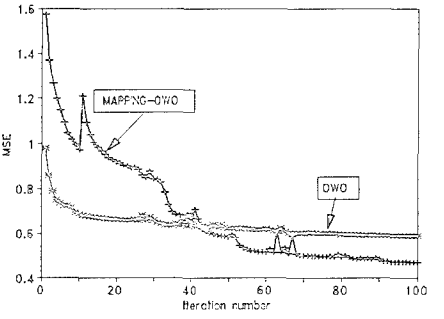


Figure 4. Training Results for Example 2.

The training data for the MLP network contained 10,000 patterns. The self-organizing estimator (1) chose a signal modeling MLP with the structure 3-15-8-16, (2) generated the three bounds on the estimation error variances shown in Table 2, (3) chose the MLP structure 16-40-3 for the estimation MLP, and (4) had the training errors shown in Fig. 4 and Table 2. In the figure, training errors are plotted for the new procedure for $\sigma_y = 1.0$ and for sigmoid networks initialized with random initial weights. The use of the polynomial network gave no initial advantage for this data set, unlike in the first example. From the table, all of the parameters' training errors are greater than the corresponding bounds, as one would expect. Clearly, the MLP has problems estimating $k\sigma$.

Table 2. Bounds and Training MSE for Example 2.

Parameter	ε	$k\sigma$	kL
Bound	2.32×10^{-6}	5.77×10^{-3}	6.53×10^{-6}
MSE	4.34×10^{-6}	4.68×10^{-1}	7.34×10^{-6}

V. Conclusions

In this paper, we have descibed a self-organizing system for generating parameter estimation networks. Given a set of training data, the system produces a statistical signal model for the input patterns, finds bounds on the training error for each desired output, determines a good network structure using a complexity estimation algorithm, and trains the estimator. The system was demonstrated on two remote sensing data sets. In both cases, most of the final estimation network's outputs had training errors greater than the corresponding bounds.

Much work remains before the system described here can be made widely applicable. The theory behind the complexity estimation algorithm needs to be fully developed. The efficient training procedure is not always better than other available training approaches, and must be improved. The extension of the bounds to the case where the desired mapping is non-injective will be necessary before the system can be applied to prediction and control problems.

Acknowledgements

This work was funded by NASA under grant NAGW-3091, by the NSF under grant IRI-9216545, by EPRI under grant RP 8030-09, and by the Advanced Technology Program of the state of Texas.

VI. References

[1] A. Gopalakrishnan, X. Jiang, M-S Chen, and M.T. Manry, "Constructive Proof of Efficient Pattern storage in the Multilayer Perceptron," *Conference Record of the Twenty-Seventh Annual Asilomar Conference on Signals, Systems, and Computers*, Nov. 1993.

[2] M.A. Sartori and P.J. Antsaklis, "A Simple Method to Derive Bounds on the Size and to Train Multilayer Neural Networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 4, July 1991, pp. 467-471.

[3] M. Plutowski and H. White, "Selecting Concise Training Sets from Clean Data," *IEEE Trans. on Neural Networks*, vol. 4, no. 2, March 1993, pp. 305-318.

[4] K. Kim and M.T. Manry, "Complexity Estimation for the Multilayer Perceptron", submitted to *The 29th Asilomar Conference on Signals, Systems, and Computers*.

[5] M.T. Manry, S.J. Apollo, L.S. Allen, W.D. Lyle, W. Gong, M.S. Dawson, and A.K. Fung, "Fast Training of Neural Networks for Remote Sensing," *Remote Sensing Reviews*, vol. 9, pp. 77-96, 1994.

[6] J.P. Fitch, S.K. Lehman, F.U. Dowla, S.Y. Lu, E.M. Johansson, and D.M. Goodman, "Ship Wake-Detection Procedure Using Conjugate Gradient Trained Artificial Neural Networks," *IEEE Trans. on Geoscience and Remote Sensing*, Vol. 29, No. 5, Sept 1991, pp. 718-726.

[7] Q. Yu, S.J. Apollo, and M.T. Manry, "MAP Estimation and the Multilayer Perceptron," *Proceedings of the 1993 IEEE Workshop on Neural Networks for Signal Processing*, Linthicum Heights, Maryland, Sept. 6-9, 1993, pp. 30-39.

[8] W. Liang, M.T. Manry, Q. Yu, S.J. Apollo, M.S. Dawson, and A.K. Fung, "Bounding the Performance of Neural Network Estimators, Given Only a Set of Training Data," *Conference Record of the Twenty-Eighth Annual Asilomar Conference on Signals, Systems, and Computers*, vol. 2, Nov. 1994, pp.912-916.

[9] W. Liang, M.T. Manry, S.J. Apollo, M.S. Dawson, and A.K. Fung, "Stochastic Cramer Rao Bounds for Non-Gaussian Signals and Parameters," accepted by *ICASSP-95*, May 1995.

[10] H. L. Van Trees, *Detection, Estimation, and Modulation Theory - Part I*, New York, NY: John Wiley and Sons, 1968.

[11] K. Hornik, M. Stinchcombe, and H. White, "Universal Approximation of an Unknown Mapping and its Derivatives Using Multilayer Feedforward Networks," *Neural Networks*, Vol. 3, 1990, pp. 551-560.

[12] Y. Ito, "Approximations of Differentiable Functions and Their Derivatives on Compact Sets by Neural Networks," *Math. Scientist*, Vol. 18, 1993, pp. 11-19.

[13] M.S. Dawson, A.K. Fung and M.T. Manry, "Surface parameter retrieval using fast learning neural networks," *Remote Sensing Reviews*, 1993, Vol. 7(1), pp. 1-18.

[14] M.S. Dawson, J. Olvera, A.K. Fung, M.T. Manry, "Inversion of surface parameters using fast learning neural networks," *Proc. of IGARSS'92*, Houston, Texas, May 1992, vol. II, pp 910-912.

[15] Fung, A.K., Z. Li, and K.S. Chen, "Backscattering from a Randomly Rough Dielectric Surface," *IEEE Trans. Geo. and Remote Sensing*, Vol. 30, no. 2, March 1992.

[16] Fung, A.K., *Microwave Scattering and Emission Models and Their Applications*, Arctec House, 1994.

Recognition of Oscillatory Signals Using a Neural Network Oscillator

Masakazu Matsugu
Imaging Research Center, Canon Inc., Shimomaruko, Ohta-ku, Tokyo, 146, Japan
Email: matsg@cam.canon.co.jp, Fax: +81(3)3757-8841

Chi-Sang Poon
Harvard-MIT Division of Health Sciences and Technology, Rm20A-126, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
Email: cpoon@cybernet.mit.edu, Tel: (617)258-5405, Fax: (617)253-2514

Abstract

We studied a possible role of a simple neural network oscillator as a neuronal classification unit of oscillatory signals. The neural oscillator, composed of two mutually inhibiting types of neuron with adaptive property in one neuron, is fed by periodic inputs of varying amplitudes, frequencies and phases. Using a performance measure defined in the frequency domain, we showed that the neural oscillator was able to accurately recognize the spatiotemporal content of the oscillatory input without any information loss. The simulation results demonstrated that such a neural oscillator may exhibit marked changes in its spatiotemporal pattern (e.g., trajectories of neuronal activities) in response to changes in the oscillatory input. Under a strong entrainment condition, the network could differentiate small changes in the frequency of in-phase inputs by displaying profound changes in both the waveforms and relative phases of the neuronal activities. Similarly, changes in the phase relationship of the oscillatory inputs are manifested as significant changes in the amplitude of neuronal activities. In this manner, the frequency, amplitude and phase of the oscillatory inputs may be represented (and possibly classified) in terms of the corresponding spatiotemporal pattern of the neural oscillator. The results suggest a plausible mechanism for the classification of oscillatory signals in biological neurons without the need for any quantitative measurements, and the feasibility of such a simple neural network architecture as a building block for oscillatory information processing.

1 Introduction

Stroage, retrieval and recognition of spatiotemporal patterns in oscillatory neural networks have been studied extensively under the Hebbian learning scheme (Amit et al., 1990; Wang et al., 1990; Schomaker, 1992;

0-7803-2739-X/95 $4.00 © 1995 IEEE

Poon, 1993; Matsugu & Yuille, 1994), the extentions of backpropagation scheme (Williams & Zipser, 1989; Doya & Yoshizawa, 1989 ; Rowat & Selverston, 1991; Toomarian & Barhen, 1992 ; Lin et al., 1992; Sun et al., 1992; Simard & Le Cun, 1992), the variational scheme (Pearlmutter, 1989; Bersini et al., 1994; Sotelino et al., 1994), and generalized matched filtering scheme (Hecht-Nielsen, 1991).

However, it is still largely unclear how biological systems effectively process oscillatory signals, and how information encoded in the spatiotemporal activities of neurons may be decoded in the brain.

It has been suggested that a periodically driven neural oscillator is functionally equivalent to a heteroassociative memory in that the unique spatiotemporal activity pattern induced by the periodic source represents a form of information storage (Matsugu & Poon, 1993).

In this paper, we propose another useful function of neural network oscillators: as a classifier of unknown periodic inputs (e.g., detecting frequency, phase, and amplitude), a function that cannot be subserved by conventional (non-oscillatory) neural networks. The emergent neurodynamics in response to sinusoidal inputs revealed that the trajectory in the state space of neuronal activities could serve as an indicator of spatiotemporal information content (i.e., frequency, phase, and amplitude). We propose a performance measure that approximately gives the information loss during the classification process. The results demonstrate the feasibility of a novel scheme for oscillatory pattern recognition using oscillatory neural networks.

2 The Network Model

A minimal network model of physiologically plausible rhythmogenesis (Fig. 1) is composed of two mutually inhibiting neurons: I and E, (Duffin, 1991). A virtual interneuron F, which provides negative recurrent feedback for the I neuron, accounts for the adaptation properties of the I neuron which lead to its decrementing activity during a burst (Matsuoka, 1985). The inputs driving the I and E neurons originate from a source D.

The feedback neuron F can simulate the adaptation effect in neuron I without causing self-induced oscillations provided the following inequality is met (Matsuoka, 1985):

$$(T_F - T_{I,E})^2 \geq 4T_F T_{I,E} C_F$$

where T_F is the time constant for neuron F (1.5 s), $T_{I,E}$ is the time constant for the I and E neurons (0.08 s) and C_F is the connection strength from neuron F to neuron I.

The equation describing the activity of any neuron i in the model is

(Duffin, 1991):

$$T_i \frac{dx_i}{dt} + x_i = R_i + \sum_j C_{ij} g(x_j - H_j) + S_i \tag{1}$$

where x_i is the current activity of the ith neuron; R_i is its resting activity (-10 for the I and E neurons, 0 for neuron F); H_i is its threshold (0 for all neurons); C_{ij} is the strength of the connection from the jth to the ith neuron; and $g(z)$ is a nonlinear function (e.g., $g(z) = max(0, z)$) used to model the activation threshold for the model neurons.

For simplicity, the inputs to oscillator neurons I and E from driving source D (denoted S_I and S_E, respectively) are assumed to be of the form:

$$S_{I,E} = A_{I,E} \sin(2\pi f_{I,E} t + \phi_{I,E}) + DC_{I,E} \tag{2}$$

where DC is the steady component and A, f, ϕ are respectively the amplitude, frequency and phase angle of the sinusoidal component of the signal to the I or E neurons (with corresponding subscripts).

Sinusoidal inputs to the I and E neurons are said to be *in-phase* or *anti-phase* if $f_I = f_E$ and $\phi_I - \phi_E = 0$ or 180 deg respectively (Cohen et al., 1992; Cymbalyuk et al., 1994).

3 Classification of spatiotemporal contents

In order to gain some insight about the ability of the simple neuronal classifier, we propose here a provisional measure of its discriminatory power, given the data set of input and output (i.e., induced activities of neurons) patterns.

To be specific and for practical reasons, we will measure the spatiotemporal contents of total neuronal activities and oscillatory inputs given by the state vectors $\mathbf{Q}_{in}$, $\mathbf{Q}_{out}$ respectively,

$$\mathbf{Q}_{in,out} = (b_1^{in,out}, b_2^{in,out}, ..., b_n^{in,out})^T,$$

where $n = N * M$, and N is the number of component neurons (mutually inhibitory neurons; $N = 2$ in a minimal model), M is the number of bins, and b_i^{in}, b_i^{out} denote frequency bins, for inputs and outputs respectively, in the frequency domain partitioned from $(m-1)\delta$ to $m\delta$ with $m = [(i-1)/N] + 1$ ($[x]$ designates taking the integer part of x),

$$b_i^{in,out} = \int_{(m-1)\delta}^{m\delta} G_j^{in,out}(f)\, df, \tag{3}$$

where $G_j^{in,out}(f)$ represents the spectrum (e.g., amplitude, phase spectrum, or else) of the input and output at the j th neuron ($j = [(i-1)/M] + 1$), respectively.

We define a state vector, $(b_1^{in,out}, b_2^{in,out}, \ldots, b_n^{in,out})^T$, which is, for activity of neurons (i.e., mutually inhibitory neurons, I and E), approximately unique to the spatiotemporal contents of the neural network. Thus, with appropriate dimensionality M, it can be used to measure the volume of oscillatory patterns realized by the system. The performance measure C_I (Matsugu & Poon, 1993) is then,

$$\begin{aligned} C_I &= \frac{V_{out}}{V_{in}} \\ &= \frac{\int \ldots \int_V \mathbf{Q}_{out}\, db_1 \ldots db_n}{\int \ldots \int_V \mathbf{Q}_{in}\, db_1 \ldots db_n} \end{aligned} \tag{4}$$

where V_{in}, V_{out} are the volume for input and output realizations respectively, defined in the multidimensional hyperspace that specifies the state vector $\mathbf{Q}$.

It should be noted that each volume is measured *only for* stable oscillation patterns. In practice, it is very difficult to perform the integration to obtain the estimate of volume. Instead, we approximate it by first remapping each state vector into a lattice space, which is simply given by discretizing the original hyperspace in units of certain normalization factor, L. That is, we divide each real valued component b_i by L and take a nearest integer value, which yields a lattice point. Thus in the lattice space, state vectors $(B_1, B_2, \ldots, B_n)$ are given by $B_i = [b_i/L]$, where $[\ X\ ]$ denotes the nearest integer part of X. Obviously, the larger L comes with less accurate estimate of the volume occupied by given realizations.

The number of remapped points in the state space gives a rough measure of total different realizations in the system's response. Thus the performance measure, the ratio of input and output volumes in the state space of spectrum domain, is intended to give a loose measure of discrimination power for the oscillatory neural system by comparing the number of different points for input signals and output activities.

4 Results

4.1 Emergent neurodynamics in a network oscillator

Using a sinusoidally varying driving source D ($A_I = A_E = 50$) with the drives to the I and E neurons in-phase ($\phi_I - \phi_E = 0; f_I = f_E$), and $DC_E/DC_I = 0.8 (DC_I = 50)$, we examined the oscillation patterns of the model at varying sinusoidal driving frequencies. The patterns of oscillation are displayed as limit-cycle trajectories by plotting E neuron activity versus I neuron activity in the phase plane (Fig. 2). For driving frequencies from 0.15 to 2.0 Hz, each neuron of the model was totally (1:1) entrained to the driving frequency but the limit-cycle trajectories exhibited a progressive transformation with drastic changes in character as the driving frequency was varied (Fig. 2 (a)). At driving frequencies

between 1.5 and 2.0 Hz (i.e., approximately 5 times the spontaneous frequency) the limit-cycle trajectory was a simple loop with the major and minor axes pointing in the horizontal and vertical directions, respectively, indicating that the I and E neurons were nearly 90 deg out of phase (i.e., somewhat between in-phase and anti-phase). Thus, in the in-phase input configuration the I and E neurons were entrained to the periodic input with differing phase shifts. As the driving frequency was decreased toward the spontaneous frequency, the limit-cycle trajectory deformed continuously until, at a frequency of $\sim$1.2 Hz, a cusp was formed which gradually turned into a twist when the driving frequency was decreased to 0.8 Hz (i.e., approximately 2 times the spontaneous frequency).

At lower driving frequencies, the limit-cycle trajectory again deformed continually until at a frequency of $\sim$0.43 Hz (i.e., near the spontaneous frequency) the twist disappeared and a simple loop returned. As shown in Fig. 2 (a) at $f \leq 0.5$ Hz the major axis of the loop was tilting to the left indicating that the oscillations of the I and E neurons were anti-phase, unlike the in-phase input. As the driving frequency was decreased still further, a cusp re-appeared at $\sim$0.33 Hz and was fully developed as the driving frequency fell below the spontaneous frequency. The amplitude of the entrained oscillation remained relatively stable at low driving frequencies. However, at driving frequencies above $\sim$1.0 Hz the amplitude decreased progressively with increasing drive frequency.

For stronger periodic inputs, the entrained rhythm was less susceptible to magnitude attenuation and/or harmonic and phase distortions (phase angle between I and E neuron activity < 180 deg) when the inputs were anti-phase than when they were in-phase.

Using the same sinusoidally varying driving source D except that the inputs to the I and E neurons were anti-phase ($\phi_I - \phi_E = 180\,\mathrm{deg}$), we examined the oscillation patterns of the model at varying sinusoidal driving frequencies. Over the same range of driving frequencies (0.15 to 2.0 Hz) the amplitude of the entrained oscillation was significantly greater than that resulting from the in-phase inputs.

The maximum response occurred at a driving frequency of $\sim$0.6 Hz and the response decreased rapidly with increasing departure of the driving frequency from the optimal frequency. At all driving frequencies the limit-cycle trajectory remained relatively stable without appreciable changes in character (i.e., appearance of twists and cusps and changes in phase angles between the I and E neuron activities) as observed for the in-phase inputs (Fig. 2 (b)).

4.2 Estimation of performance measure

The proposed perfomance measure for decoding of oscillatory signals (eq. 4) is dependent on the dimensionality of the hyperspace which is determined by the lattice size and the width of the partitioned frequency

range.

To investigate the effect of the dimensionality on the resulting estimate, we calcultated the performance measure using varying values of L and δ. A limited frequency range (from 0.15 to 2.0 Hz) of in-phase inputs with sufficient amplitude was used to ensure 1:1 entrainment of the oscillator output.

Figure 3 shows the performance measure estimated from the amplitude spectra for varying lattice sizes, L, (δ: 0.2 Hz, unbroken line; 0.4 Hz, broken line). It indicates an asymptotic performance measure of approximately one. Two points should be noted. First, the information content of the input signal is reflected in the output spatiotemporal pattern. Second, the performance measure is not critically dependent on δ or the lattice size used in the simulation.

However, it was found that if the amplitude of the input signal is less than a certain level, qualitatively different behaviors emerge: the network neurons oscillate with an intermediate frequency between the input frequency and the intrinsic frequency. If the input amplitude or coupling strengths to the inputs are too small compared with the intrinsic oscillation (i.e., spontaneous oscillation under DC inputs), the frequency contents of the resulting oscillation would be very similar to the intrinsic ones. In this case, regardless of any changes in the input, the output will remain close to the intrinsic oscillation and the performance measure as defined in Section 3 will be quite different from that resulting from the system with strong inputs.

Obviously for the latter case, the estimated preformance measure will be far less than unity.

5 Discussion

The primary purpose of this study was to explore the possibility of using a simple neural network oscillator to recognize and classify periodic signals. Despite the simplicity of the network architecture, the resulting performance measure (Fig. 3) approached unity for small lattice size suggesting the possibility of classifying oscillatory patterns without significant information loss even with a simple oscillatory neural network.

The present study suggests a novel scheme of classifying oscillatory signals that might be employed in biological neural networks and may also be useful for similar signal processing tasks with artificial neural networks.

Our results showed that anti-phase inputs generally resulted in stronger and more stable outputs than in-phase inputs with similar driving conditions. This behavior can be intuitively deduced by considering that the agonist-antagonist pattern of anti-phase inputs is intrinsically more compatible with network oscillation in reciprocally inhibiting neurons, and is suggested by the fact that the critical amplitude for entrainment

is lower for anti-phase inputs than for in-phase inputs (Matsugu, Duffin, and Poon, 1995).

The entrained oscillations induced by periodic inputs with varying information contents (i.e., frequencies and phase relationships) exhibited a wide variety of emergent neural activities (Fig. 2), characterized by dramatic deformations of output trajectories resulting in marked changes in the frequency contents of the output.

In the present neural oscillator model, the region of highest sensitivity (where profound changes in the trajectory occur) is centered around 1.0 Hz for oscillatory in-phase inputs (i.e., a frequency range from 0.7 Hz to 1.2 Hz). By appropriate scaling of the network time constants and connectivities, one may move this active region to anywhere within the spatiotemporal hyperspace.

It is interesting to note that the amplitude of the entrained oscillation may serve as an indicator of the input frequency as well as the phase relationship of the oscillatory input to individual neurons.

Similarly, the presently demonstrated complex modulations of the entrained oscillation waveform by varying frequency contents of the input may be useful for certain oscillatory information coding/decoding tasks. Such dynamic transformation of changes in oscillatory pattern of the input into changes in output trajectories may suggest another potential use of the oscillatory unit in the generation of complex motor patterns in locomotory or kinematic systems, in contrast to previous work (Ermentrout & Koppel, 1994) which focused on obtaining a fixed desired phase difference.

It is also tempting to combine the present oscillatory unit with a neural system that could classify distinct trajectory patterns (Sotelino et al., 1994; Sun et al., 1992; Hecht-Nielsen, 1991) so that the composite system would recognize the temporal information content of the oscillatory inputs. Another way of constructing a similar system is to use a group of such oscillatory units, each being tuned to differing frequency regions of interest (where trajectories deform maximally in response to changes in input frequencies or phases) in the spatiotemporal hyperspace so that the population dynamics of the unit oscillators would encode certain features of the temporal input patterns, somewhat like those in population coding schemes in the motor cortex (see for example, Lukashin & Georgopoulos, 1994; Sanger, 1994). In this regard, the present study suggests that a wide spectrum of trajectories may be realized by a small number of neuronal groups which are excited by inputs with differing spatiotemporal patterns.

A possible future direction of oscillatory pattern classification based upon such neurodynamics properties is to explore the abrupt transition between distinct entrainment conditions (i.e., from $n : m$ to $n' : m'$ phase-locking mode) in response to changes in the input. The idea is to construct a system composed of neuronal oscillators like the one in Fig. 1, but having different resonance structures (e.g., different structure

of Arnold tongues) where each phase-locking zone would correspond to different input categories.

Acknowledgements This work was supported in part by Office of Naval Research contract N00014-95-0414, National Science Foundation grant BES-9216419, and National Institutes of Health grants HL-45261, HL-50614.

References

[1] Amit, D.J., Evans, M.R., and Abeles, M.A. (1990) Attractor neural networks with biological probe records. Network, 1: 381-405.

[2] Bersini, H., Saerens, M., and Sotelino, L.G. (1994) Hopfield net generation, encoding and classification of temporal patterns. IEEE Trans. on Neural Networks, 5: 945-953.

[3] Cohen, M., Grossberg, S., and Pribe, C. (1992) A neural pattern generator that exhibits frequency-dependent in-phase and anti- phase oscillations. in *Conference on Neural Networks for Learning, Recognition and Control*, Boston Univ. May 14-16, pp. 32.

[4] Cymbalyuk, G.S., Nikolaev, R.M. and Borisyuk, R.M. (1994) In-phase and anti-phase self-oscillation in a model of two electrically coupled pacemakers. Biological Cybernetics 71: 153-160.

[5] Doya, K. and Yoshizawa, S. (1989) A neural network model of temporal pattern memory: Adaptive neural oscillator using continuous-time backpropagation learning. Neural Networks 2: 375-385.

[6] Duffin, J. (1991) A model of respiratory rhythm generation. Neuroreport 2: 623-626.

[7] Ermentrout, B. and Koppel, N. (1994) Learning of phase lags in coupled neural oscillators. Neural Computation 6: 225-241.

[8] Hecht-Nielsen, R. (1991) Towards hierarchical matched filtering. in *Neural Networks Theory and Applications*, Mammone, R.J., and Zeevi, Y. eds., Academic Press, pp. 217-232.

[9] Lin, D.-T., Dayhoff, J.E. and Ligomenides, P.A. (1992) Trajectory recognition with a time-delay neural network. in *Proc. Int. Joint Conf. on Neural Networks*, Baltimore, MD, pp. III-197-202.

[10] Lukashin, A. and Georgopoulos, A.P. (1994) A neural network for coding of trajectories by time series of neuronal population vectors. Neural Computation 6: 19-28.

[11] Matsugu, M. and Poon, C.S. (1993) Estimation of information capacity in oscillatory neural networks. in *Proc. Int. Joint Conf. on Neural Networks*, Nagoya, Japan, pp. 425-428.

[12] Matsugu, M. and Yuille, A.L. (1994) Spatiotemporal information storage in a content addressable memory using realistic neurons. Neural Networks, 7: 419-439.

[13] Matsugu, M., Duffin, J. and Poon, C.S. (1995) Entrainment, quasi-periodicity and chaos in a compound neural oscillator. (submitted).

[14] Matsuoka, K. (1985) Sustained oscillations generated by mutually inhibiting neurons with adaptation. Biological Cybernetics 52: 367-376.

[15] Pearlmutter, B. (1989) Learning state space trajectories in recurrent neural networks. Neural Computation 5: 263-269.
[16] Poon, C.S. (1993) Adaptive neural network that subserves optimal homeostatic control of breathing. Annals of Biomedical Engineering, 21: 501-508.
[17] Rowat, P.F. and Selverston, A.I. (1991) Learning algorithms for oscillatory networks with gap junctions and membrane currents. Network 2: 17-41.
[18] Sanger, T. (1994) Theoretical considerations for the analysis of population coding in motor cortex. Neural Computation 7: 29-37.
[19] Schomaker, L.R.B. (1992) A neural oscillator-network model of temporal pattern recognition. Human Movement Science 11: 181-192.
[20] Simard, P. and Le Cun, Y. (1992) Reverse TDNN: An architecture for trajectory generation. in *Advances in Neural Information Processing 4*, Morgan Kaufman, pp. 579-588.
[21] Sotelino, L.G., Saerens, M., and Bersini, H. (1994) Classification of temporal trajectories continuous-time recurrent nets. Neural Networks 7:767-776.
[22] Sun, G.-Z., Chen, H.-H., Lee, Y.-C., and Liu, Y.D. (1992) Time warping recurrent neural networks and trajectory classification. in *Proc. Int.Joint Conf. on Neural Networks*, Baltimore, MD, pp. I-431-436.
[23] Toomarian, N. and Barhen, J. (1992) Learning a trajectory using adjoint functions and teacher forcing. Neural Networks, 5: 473-484.
[24] Wang, D.L., Buhmann, J. and von der Malsburg, C. (1990) Segmentation in associative memory. Neural Computation, 2: 94-106.
[25] Williams, R. & Zipser, D. (1989) A learning algorithm for continually running fully recurrent neural networks. Neural Computation, 1: 270-280.

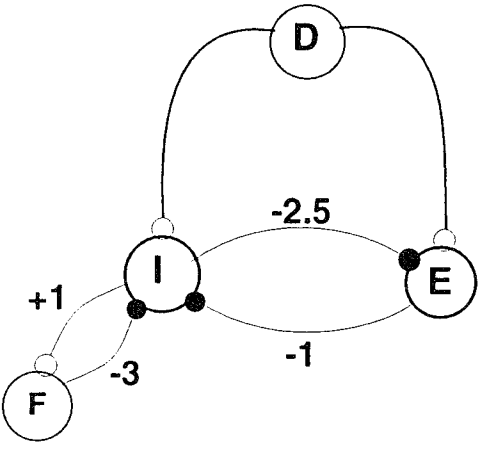


Fig. 1 **The oscillatory network model. I and E denote mutually inhibiting neurons, D is the oscillatory source, F designates a virtual neuron which accounts for adaptation of I neuron.**

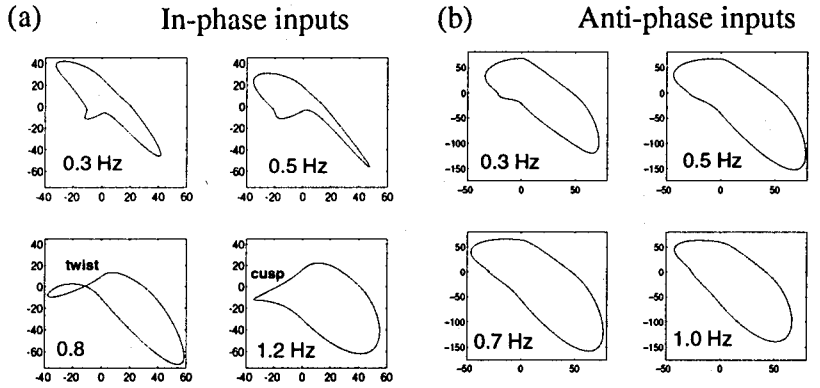


Fig. 2 **The phase trajectories of the activities of I (abscissa) and E (ordinate) neurons. (a) The trajectories corresponding to in-phase inputs vary significantly with input frequncy. (b) For anti-phase inputs, there is no appreciable change in the trajectory.**

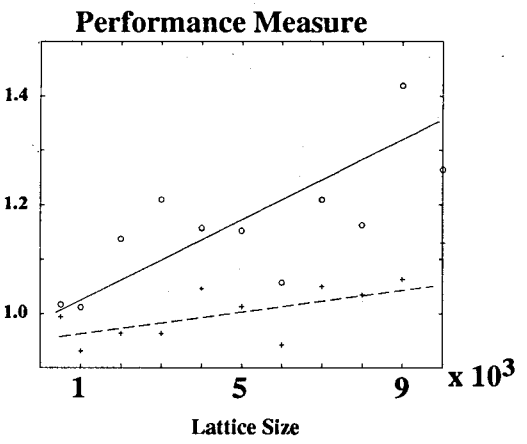


Fig. 3 **The estimated performance measure of the model network under in-phase inputs with entrainment condition. With decreasing lattice size (see section 4.2), the measure approaches to a unit value. The result suggests that the network could recognize the oscillatory signals without critical information loss.**

PRINCIPAL-FEATURE CLASSIFICATION

Donald W. Tufts and Qi Li
Department of Electrical and Computer Engineering
University of Rhode Island, Kingston, RI 02881
Email: tufts@ele.uri.edu and qli@research.att.com

Abstract – The concept of classification using principal features is presented. The principal features defined in this paper are analogous to principal components in statistics and linear algebra. Neural network training can be done by sequential identification of principal features and corresponding pruning of the training data. Two neural network simplification algorithms, lossless and lossy simplifications, make the the classifier design more efficient. The design procedure is compared with other classifier design algorithms.

1 INTRODUCTION

Principal Feature Classification is based on a sequential procedure for finding principal features. This is analogous to a method for sequentially finding principal-component basis vectors. One can first find the principal-component vector which provides the best single vector for use in least-squares approximation of the set of training vectors. Then one removes the contribution of this principal component from each training vector to force a modified ("pruned") set of training data. The procedure is then repeated to obtain the second-best principal component and so on, from the sequentially pruned training data.

Successive determination of principal features and the associated, successive pruning of the training data are naturally different than the analogous steps for principal components because the criterion, namely improvement in classification performance, is different.

In each stage, motivated by a multiple-Gaussian-component model for the probability density of a vector observation from any class, linear and nonlinear discriminant analysis is applied to find current principal features. The training vectors which are sufficiently well classified using these features are pruned. In the next stage, the design again applies linear and nonlinear discriminant analysis to the residual, unclassified training data set to find new features until the training vectors are classified at the target level of performance, which is chosen to permit good generalization to the test data.

0-7803-2739-X/95 $4.00 © 1995 IEEE

Example 1. We use the two classes of data in Figure 1 (a) to show the procedure of finding principal features. The design starts by first finding two hyperplanes in the input space associated with the first principal feature and the first hidden node. Fisher's method is used to find a weight vector as the first principal features with all the training data in the input space [6, 1]. The hyperplanes in perpendicular to the vector are show in Figure 1 (a). Then, the classified data are pruned off and only the unclassified data in between of the two hyperplanes are used to train the second hidden node. The residual data set from Figure 1 (a) is shown in Figure 1 (b). Since the mean vectors of the two classes are very close now, Fisher's method does not give the best principal features. In the second hidden node design, we use a principal component analysis [4] to find the second principal features and associated two hyperplanes. The network structure and associated training algorithm is summarized in the Section 2.

For this classification problem, the Backpropagation (BP) training method takes hundreds of seconds to hours, and one still does not get satisfactory classification. The Radial Basis Network (RBF) can converge to an acceptable performance in 35 seconds, but it needs 56 nodes. On the same problem, a design based on the principal feature classification only takes 0.2 seconds and needs only two hidden nodes with a better performance than both BP and RBF.

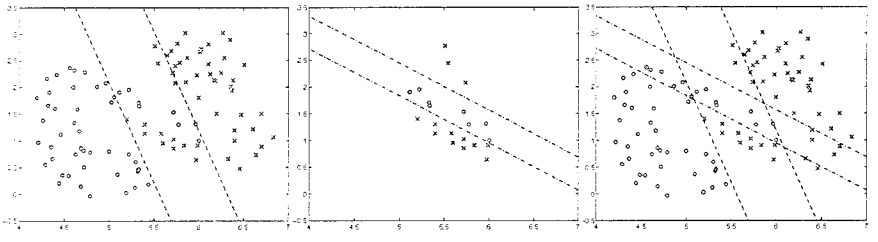


Fig. 1 (a). The original data and the hyperplanes of the first hidden node (Fisher's node). (b). The residual data set and the hyperplanes of the second hidden node (Principal Component Discriminant node). (c). The partitioned input space by two hidden nodes and four thresholds designed by the Principal Feature Classification.

2 PRINCIPAL-FEATURE NETWORKS

Principal Feature Networks (PFN) are a class of neural networks based on the principal feature concept. An implementation of the PFN is shown in Figure 2. It was called a *Discriminant Neural Network* (DNN) in [1]-[4]. Similar name is also used for other methods [18]. So we have now changed the name from DNN to PFN to be more specific and to concentrate on the new design principles.

The hidden nodes are the building blocks of PFN. The single hidden node design algorithm is motivated by multiple-multivariate-Gaussian component

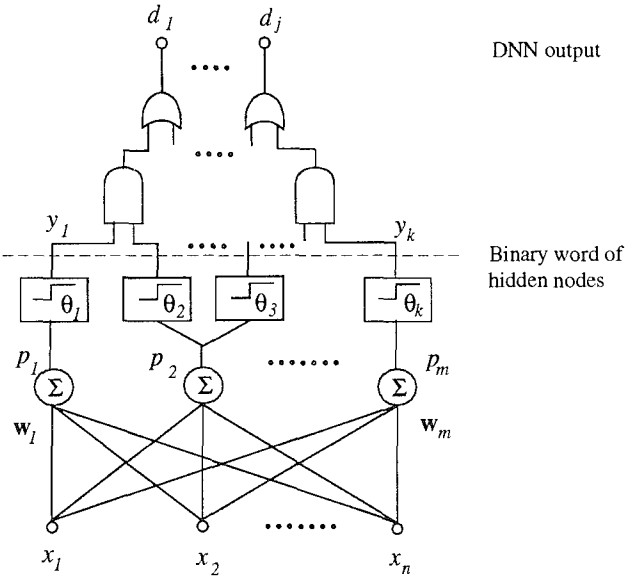


Fig. 2. An implementation for Principal Feature Networks (PFN).

classification [19]. Two kinds of practical hidden nodes, a Fisher's Node, for training classes with separable mean vectors, and a Principal Component Discriminant Node, for training classes with common mean vectors, are used to separate classes. They are designed for non-Gaussian and not linearly separable cases [4].

When components of two training data populations Class 1 and Class 2 are described as having multivariate Gaussian distributions with sample mean vectors and covariance matrices μ_1, Σ_1 and μ_2, Σ_2 respectively, the minimum-cost classification rule is given by:

$$Class1 : L(\mathbf{x}) \geq \theta; \quad Class2 : L(\mathbf{x}) < \theta; \tag{1}$$

where $\mathbf{x}$ is an observed data vector or feature vector of N components and θ is a threshold determined by the cost ratio, the prior probability ratio, and the determinants of the covariance matrices, and

$$\begin{aligned} L(\mathbf{x}) &= \mathbf{x}^t(\Sigma_1^{-1} - \Sigma_2^{-1})\mathbf{x} - 2(\mu_1^t\Sigma_1^{-1} - \mu_2^t\Sigma_2^{-1})\mathbf{x} \\ &= \sum_{i=1}^{N} \lambda_i |\mathbf{x}^t W_i|^2 - 2W_0\mathbf{x}, \end{aligned} \tag{2}$$

where $W_0 = (\mu_1^t\Sigma_1^{-1} - \mu_2^t\Sigma_2^{-1})$ and for $i > 0$, λ_i and W_i are the i'th eigenvalue and eigenvector for matrix $\Sigma_1^{-1} - \Sigma_2^{-1}$. Formula (2) is implemented as a *Gaussian Discriminant Node* in Figure 3(a).

The purpose of such a node is to provide a feature which permits an approximately or locally Gaussian component of a class to be separated from

other classes. It is not necessary at any stage to separate the whole class, but simply to isolate the next separable component of the class.

When the covariance matrices in (2) are the same, the first, quadratic term is zero, and it computes Fisher's linear discriminant. The general node becomes a Fisher's node as in Figure 3(b). When the second term can be ignored, the above formulas only have the first quadratic term. If we only use the first eigenvalue, the Gaussian node becomes a quadratic node as shown in Figure 3(c). The thresholded squaring function can be further approximated by two thresholds as in Figure 3(d).

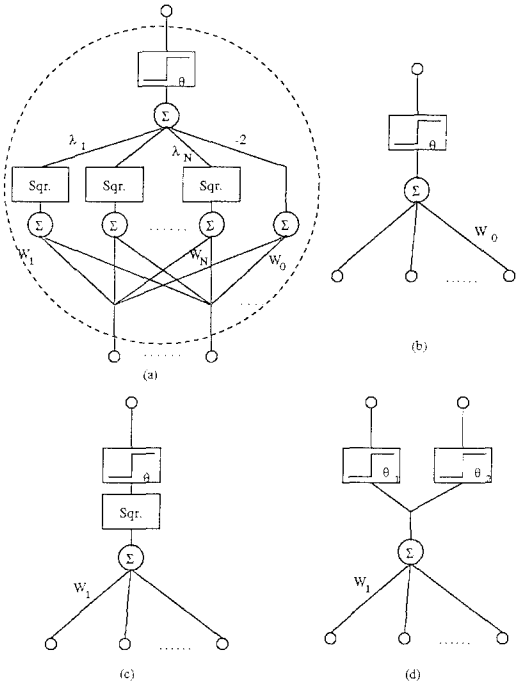


Fig. 3. (a) A single Gaussian discriminant node. (b) A Fisher's node. (c) A quadratic node. (d) An approximation of the quadratic node.

To design a Fisher's node, we can use Fisher's linear discriminant analysis [1, 6]. To design a quadratic node directly for non-Gaussian data, we use the following criterion for an alternate discriminant: We choose a weight vector w to maximize a discriminant signal-to-noise ratio J.

$$J = \frac{E\{(X_l \mathbf{w})^t (X_l \mathbf{w})\}}{E\{(X_e \mathbf{w})^t (X_e \mathbf{w})\}} = \frac{\mathbf{w}^t \Sigma_l \mathbf{w}}{\mathbf{w}^t \Sigma_e \mathbf{w}}, \tag{3}$$

where X_l is a matrix of row vectors of training data from class l. X_e is the matrix of training data from all classes except X_l. The class l is the class which has the largest eigenvalue among the eigenvalues calculated from the data matrices of each class respectively. Σ_l and Σ_e are the estimated

covariance matrices and $\mathbf{w}$ is the weight vector [4]. The sequential hidden node design and data pruning procedure has been introduced in Section 1. See [1] - [4] for the details.

Generally speaking, other single-node (perceptron) training algorithms can also be applied in PFN training. We prefer the above algorithms motivated from multivariate statistical analysis because they can solve large application problems much faster than gradient-descent or iterative algorithms without worrying about local-minimum in a signal-node training.

3 HIDDEN NODE SIMPLIFICATION

We present two kinds of simplification algorithms for different applications, *Lossless Simplification* for minimal implementation and *Lossy Simplification* for improving the ability of the network for generalization. Depending on applications, the lossless and lossy pruning algorithms can be applied individually or together.

3.1 Lossless Simplification

After the hidden node design, the input space is partitioned by hyperplanes associated with their thresholds and hidden nodes. Some of the hyperplanes may not be necessary for a minimal implementation. The hidden node simplification and the output node design can be treated as a Boolean minimization problem. The function of the output nodes is to group the partitioned regions of same class into one output binary word, i.e. to generate a Boolean function F, such as $d = F(y)$, where d represents the output binary words, one word for one class, y is the binary words of the hidden node outputs, one word for one region. Usually, in a multi-dimensional input data space, some of the partitioned regions may not have data vectors. The regions without data vectors can be used in Boolean minimization as don't care items to simplify the Boolean function F. The Boolean minimization can be done by logic-minimization algorithms implemented in computer software [15].

The above simplification algorithm will not change the logical representation of the Boolean function F. No region with data vectors is ignored and nothing is changed on the network accuracy on the training data set, so it is lossless pruning. The output nodes are designed during the pruning procedure for a minimal implementation. However, it is not designed for improving the network generality.

3.2 Lossy Simplification

The *Lossy Simplification* is developed for improving the ability of the network to generalize to new data. The simplify algorithm is based on the performance analysis of each threshold as well as hidden node. We call it lossy pruning

because a subset of partitioned regions and associated training vectors will be ignored and the network accuracy will be reduced on the training data set. Compared to the lossless simplification, the lossy Simplification is much faster and more practical for real applications.

During the PFN training, each threshold is labeled with the class partitioned by that threshold. Also, the contribution of each threshold for each class are saved in an array, called *contribution array*. The array is used for pruning analysis. We use the following example to illustrate the details of the simplification algorithm.

4 A DESIGN EXAMPLE

Example 2: A principal feature network was designed to recognize 10 classes of signals in a real application. Each of training and test data sets has about 3,000 examples and each example is a 24 dimensional vector. In the design specifications, the expected network accuracy is 95%, which will be used to determine the necessary number of hidden node, and the allowed miss-classification rate is 20% for all 10 classes, which is used in determining the thresholds to avoid over-fitting.

Using the sequential partition-pruning design procedure, all training examples were partitioned by 49 hidden nodes and 98 thresholds. The 49 hidden nodes included 36 Fisher nodes and 13 principal component nodes. Each node has 2 thresholds.

The contribution of each threshold was saved in a contribution array. The array was sorted and plotted in Figure 4(a). From the Figure 4(a), we can see that few of the thresholds have significant contribution to some of the classes, but many thresholds have too little contribution in partitioning the input space. The accumulated network performance in the order of the sorted thresholds is shown in Figure 4(b). The more the thresholds we keep, the higher the network accuracy we can obtain on the training data set, but to keep too many thresholds which have too little contribution can affect the generality of a designed network. In other word, to use more thresholds may not give a higher accuracy on the test data set.

For this example, the desired network performance is 95%. A horizontal dash-dot line in Figure 4(b) marked the desired 95% accuracy. The line has an intersection with the curve of the accumulated network performance. We projected the intersection onto the Figure 4(a) as the vertical broken line in both Figure 4 (a) and (b). Then a necessary number of thresholds to meet the desired network performance can be determined. For this example, the first 38 thresholds in Figure 4(a) can meet the 95% network accuracy. Thus the thresholds from the 39 to 98 can be pruned.

Once the thresholds are pruned, the hidden nodes which need to be pruned can be further determined. If all of the thresholds associated with one hidden node are pruned, the hidden nodes should be pruned. In this example, after

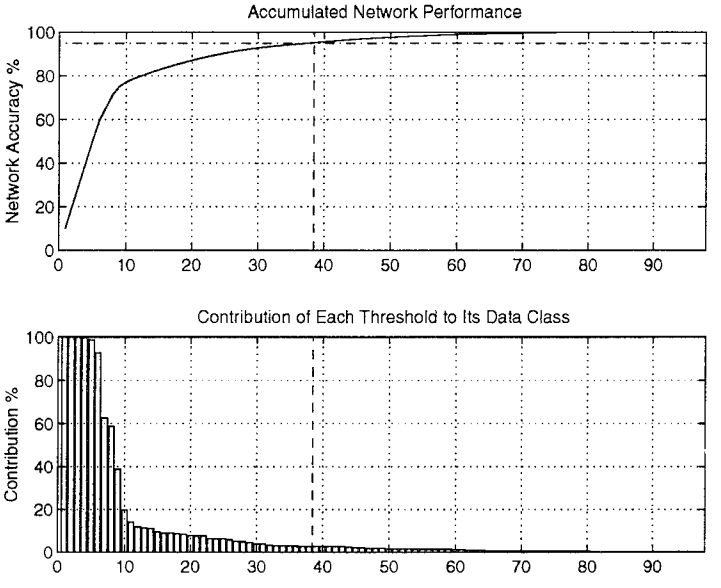


Fig. 4. (a) (bottom) The sorted contribution of each threshold in the order of its contribution to the class separated by the threshold. (b) (top) Accumulated network performance in the order of the sorted thresholds.

threshold pruning, 31 out of 49 hidden nodes have at least one associated threshold, thus these 31 hidden nodes are kept and other 18 hidden nodes are pruned. After the simplification, the actual design performance on the training set is 91.44%. On the test set, the simplified network has a performance of 87.68%.

5 COMPARISON AND CONCLUSIONS

Principal feature networks (PFN) have been compared in experiments with the most popular neural networks, such as backpropagation (BP), and radial basis function (RBF) network in term of performance, complexity of structure (number of hidden nodes), training time, and million floating-point operations (Mflops). One comparison was given in Example 1. In [3], the PFN was compared with BP, RBF, and linear discriminant analysis (LDA) in a multispectral image recognition problem. Due to the very large data set, both BP and RBF failed to train a classifier in a reasonable amount of time. For that problem, the LDA gave a classification rate of 55%; a modified RBF reached a rate of 60% with 490 hidden nodes in 221 Mflops; the PFN reached a rate of 72% with 77 hidden nodes in 38 Mflops.

The training procedure of sequential addition of hidden nodes in PFN looks similar to several constructive algorithms which have the capability to

add hidden nodes while the training in progress. These constructive algorithms include *Decision Tree Algorithms* (DTA) [12], *Neural Tree Network* (NTN) [10, 11], *Fisher Tree Networks* (FTN) [7, 8] *Cascade-Correlation Architecture* (CCA) [9], *Piecewise-linear discrimination* (PLD) [13], *Tiling Algorithm* (TA)[16], etc. For those algorithms and network architectures, since they are easy to be compared, we focus the comparison on theory and conceptual levels in the following list. Generally speaking, the PFN has the advantages of these networks or training algorithms. It can get 100 % accuracy on training set when it is necessary. It also has many new functions and advantages.

(1) Except PFN, TA, and PLD, all other algorithms are for tree structures and not for parallel implementation. In [8], the author give a algorithm to convert FTN to parallel architecture, however, the tree algorithms are not naturally for parallel implementation which should be a major advantage of neural networks. The PFN can be implemented in a tree structure for software or in a parallel structure, such as a processor array, for VLSI design when it is necessary [5].

(2) Except PFN and TA, in general, none of the other algorithms consider to pruning training data immediately after each hidden node design. The pruning can reduce the training data set, release memory space, and make the next hidden node design more efficient. Normally after first few hidden node design, most of the training data are pruned, so the PFN training in each additional hidden node will use less and less time and memory space. The TA repeats the pruning procedure in every layer of the multi-layers training while the PFN only needs to train one layer. The DTN and NTN can only prune training data when they reach the leaf nodes. The dimension of the input space of the CCA hidden node gets larger and larger during the training, so the CCA will be slowed down after adding more hidden nodes.

(3) Most of the algorithms, such as DTA, NTN, CCA, and TA, did not apply statistical method in training which can speed up the training significantly.

(4) The NTN, CCA, and TA are based on gradient-descent or iterative methods which can slow down the training and there is no guarantee for a global-minimum.

(5) The DTA can only construct hyperplanes in perpendicular to prescribed axes which either reduces the performance or needs more nodes.

(6) The PFN offers the algorithm to deal with the case in which two or more classes are over-lapped on one another and proved the optimal node design algorithm [4].

(7) The PFN allow quadratic nodes for a better performance.

(8) PFN and DTA allow multiple thresholds on each hidden nodes. This can prune more data at each hidden node without using additional weight vectors. It can also approximate the optimal quadratic nodes.

(9) The DTA and NTN have tree node pruning algorithms for a better performance on the test sets. The lossy simplification algorithm for PFN is

simpler than the tree pruning algorithms.

(10) The PFN has the lossless simplification for a minimal implementation as presented in this paper.

In conclusion, principal feature classification is a concept for designing constructive neural networks. By applying multivariate statistical analysis in defining and training hidden nodes, the principal feature networks can be trained much faster than gradient-descent or other iterative algorithms. The over-fitting problem as in most neural network training can be avoided in determining thresholds, and the generalization can be realized in the lossless simplification. The principal feature network has been used in solving real-world classification problems with large data sets. It gave better performance, less CPU time in training, and simpler network structures than other compared networks.

ACKNOWLEDGEMENT

The authors wish to thank Paul Zemany for providing the original data sets for the PFN design in the Example 2.

References

[1] Q. Li and D.W. Tufts, "Synthesizing neural networks by sequential addition of hidden nodes," **Proc. IEEE International Conference on Neural Networks**, pp. 708-713, Orlando, Florida, June 1994.

[2] Q. Li and D.W. Tufts, "Discriminant networks: a simple, effective, and rapidly trainable class of neural networks," Submitted to the **IEEE Trans. on Neural Networks**, February 1994.

[3] Q. Li, D.W. Tufts, R.J. Duhaime, and P.V. AugustFast, "Training Algorithms for Large Data Sets with Application to Classification of Multispectral Images," **Proc. IEEE 28th Asilomar Conference**, Pacific Grove, CA, October 1994.

[4] Q. Li and D.W. Tufts, "Improving discriminant neural network (DNN) design by the use of principal component analysis," **Proc. ICASSP**, Detroit, Michigan 1995.

[5] Q. Li, **Classification using principal features with application in speaker verification**, Ph.D. Thesis, University of Rhode Island, to be finished.

[6] R.A. Johnson and D.W. Wichern, **Applied multivariate statistical analysis**, pp. 470-530, New Jersey: Prentice Hall, 1988.

[7] J.C. Lee, Y.H. Kim, W.D. Lee, and S.H. Lee, "Pattern classifying neural network based on Fisher's linear discriminant function," **Proc. IJCNN**, pp. I-743 to I-748, 1992.

[8] J.C. Lee, Y.H. Kim, W.D. Lee, and S.H. Lee, "A method to find the structure and weights of layered neural networks," **Proc. world congress on neural networks**, pp. III-552 to III-555, Portland, Oregon, 1993.

[9] S.E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," In **Advances in neural information processing systems**, vol. 2, pp. 524-532, edited by D. Touretzky, San Mateo, CA : M. Kaufmann Publishers.

[10] A. Sankar and R.J. Mammone, "Growing and pruning neural tree networks," **IEEE Trans. Computers**, vol. C-42, pp. 221-229, March 1993.

[11] A. Sankar and R.J. Mammone, "Neural tree networks" in **Neural networks: theory and applications** (R.J. Mammone, Ed.). San Diego, CA: Academic, 1991.

[12] L. Breiman, J.H. Friedman, R.A. Olshen and C.J. Stone, **Classification and regression trees**, Belmont, CA: Wadsworth International Group, 1984.

[13] M. Nadler and E. Smith, **Pattern recognition engineering**, New York:J. Wiley & Sons, 1992.

[14] B. Flury, **Common principal components and related multivariate models**, New York: J. Wiley & Cons, 1988.

[15] R. Brayton, G. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, **Logic Minimization Algorithms for VLSI Synthesis**, Kluwer Academic Publishers, 1984.

[16] M. Mézard and J. P. Nada, "Learning in feedforward layered networks: the tiling algorithm", **J. Phys. A: Math Gen.** vol 22, no. 12, pp. 2129-2203, 1989.

[17] S.I. Gallant, **Neural network learning and expert systems**, Cambridge, MA: The MIT Press, 1993.

[18] B.H. Juang and S. Katagiri, "Discriminative learning for minimum error classification," **IEEE Transactions on signal processing**, vol. 40, no. 12, December 1992.

[19] R.L. Streit and T.E. Luginbuhl, "Maximum likelihood training of probabilistic neural networks," **IEEE Transactions on neural networks**, vol. 5, no. 5, September 1994.

A Habituation Based Neural Network for Spatio-Temporal Classification

Bryan W. Stiles and Joydeep Ghosh *
Department of Electrical and Computer Engineering
The University of Texas at Austin

Abstract

A novel neural network is proposed for the dynamic classification of spatio-temporal signals. The network is designed to classify signals of different durations, taking into account correlations among different signal segments. Such a network is applicable to SONAR and speech signal classification problems, among others. Network parameters are adapted based on the biologically observed habituation mechanism. This allows the storage of contextual information, without a substantial increase in network complexity. Experiments on classification of high dimensional feature vectors obtained from Banzhaf sonograms, demonstrate that the proposed network performs better than time delay neural networks while using a substantially simpler structure. The mathematical power of the network is discussed, including its ability to realize any function realizable by a TDNN. Additionally, principal component analysis is used to introduce a further improvement to the network design by reducing the dimensionality of the encoded temporal information.

Keywords: dynamic neural networks, habituation, classification, spatio-temporal signals, recurrent networks

1 INTRODUCTION

Among biological mechanisms that can encode temporal information, is a particularly simple and well understood phenomenon known as *habituation* [1], [3], [8], [13]. Primarily, habituation is a means by which biological neural systems vary their synaptic strengths in order to ignore repetitive, irrelevant stimuli. Habituation serves as a novelty filter. If the presynaptic neuron is active for a short period of time, habituation tends to decrease the synaptic strength which then recovers only after the period of activity is over. The longer the presynaptic neuron is active the slower it recovers. It is important to note that habituation does not act in a

*This work was supported in part by an NSF grant ECS 9307632 and ONR contract N00014-92C-0232. Bryan Stiles was also supported by the Du Pont Graduate Fellowship in Electrical Engineering. We thank Prof. I. Sandberg for several fruitful discussions.

0-7803-2739-X/95 $4.00 © 1995 IEEE

vacuum. Other learning mechanisms, such as sensitization and Hebbian learning, may also be operating concurrently to alter synaptic strengths based on the utility of the information from presynaptic neurons.

Several researchers in neurophysiology have developed mathematical models of habituation [1], [3], [14]. A discrete time version of the Wang-Arbib [14] habituation model for varying the strength, $W(t)$, of a single synapse is summarized by:

$$W(t+1) = W(t) + \tau(\alpha z(t)(W(0) - W(t)) - W(t)I(t)); \quad (1)$$

$$z(t+1) = z(t) + \gamma z(t)(z(t) - 1)I(t). \quad (2)$$

In this model, I(t) is the activation of the presynaptic neuron at time t, τ is a constant used to vary the habituation rate and α is a constant used to vary the ratio between the rate of habituation and the rate of recovery from habituation. The function $z(t)$ monotonically decreases with each activation of the presynaptic neuron. This function is used to model long term habituation. Due to the effect of $z(t)$, after a large number of activations of the presynaptic neuron, the synapse recovers from habituation more slowly.

Aside from its primary function, habituation has also been suggested to be a means of encoding short term temporal information [8]. In this paper, we introduce a mechanism for using habituation to encode temporal information in an artificial neural network. In Section 2 we describe the general structure of our design. In Section 3 we describe the mathematical properties of our mechanism. We demonstrate that our mechanism is a special case of a general neural network structure which is capable of approximating arbitrarily well any continuous, causal, time-invariant, mapping from one discrete time sequence to another. Finally we explain how it fundamentally differs from the gamma network model of de Vries and Principe [5]. In Section 4, we discuss experimental results for our network on the classification of artificial Banzhaf sonograms. We demonstrate that our network is more efficient than TDNNs for a number of classification problems involving long term temporal information. Finally in Section 5, we draw conclusions based upon our theoretical and experimental results.

2 GENERAL STRUCTURE

We have designed short term habituation units based upon the Wang and Arbib model of habituation [14] and used them in a spatio-temporal classification network. A set of habituated weights is first obtained from the input $I(t)$. If the input is multi-dimensional, one set is extracted for each component. These weights are affected by the past values of the input, and implicitily encode temporal information. Spatio-temporal classification can thus be achieved by using such habituated weights as inputs to a static classifier. For example, if a multilayered perceptron (MLP) (alt. radial basis function network) is used, the overall network is a habituated MLP (alt. habituated RBF) that can be applied for spatio-temporal classification. The model equation is shown as follows.

$$W_k(t+1) = W_k(t) + \tau_k(\alpha_k(1 - W_k(t)) - W_k(t)I(t)) \quad (3)$$

This equation is derived from Equation 1 by setting $z(t) = 1$ to eliminate long term habituation effects, and letting $W_k(t)$ rebound to 1 instead of $W_k(0)$. Long term

habituation is eliminated so that the ability of $W_k(t)$ to recover from habituation does not vary over time. Otherwise the $W_k(t)$ values would eventually decrease to zero for all but the most infrequent of inputs. The k index is used to indicate that multiple values $W_k(t+1)$ are determined for an input signal $I(t)$. It was found mathematically, that multiple habituation values are better able to encode temporal information. This fact may also have biological context, because it is known that a given pair of neurons often have multiple synapses between them.

In this paper, dynamic classification is achieved by training a suitable nonlinear feedforward network, whose inputs are a set of m habituated values, $W_k(t+1)$, $1 \leq k \leq m$, that are extracted from the raw input $I(t)$. Figure 1 shows the generic structure of such a classifier. In [14] $W_k(t)$ represents a synaptic strength, and $I(t)$ the activity of the presynaptic neuron, but because our designs use habituated values as network inputs rather than weights, the variables are redefined accordingly. We do not mean to imply that this network construction is either the most biologically feasible or the only method in which habituation might be used. A more biologically inspired approach would be to reflect $W_k(t)$ as modulating weights of the inputs. We found by experiment, however, that this approach, although more biologically feasible, does not encode temporal information as well for the classification problems which we studied. Moreover, the structure of Figure 1 can be shown mathematically to be very powerful.

The parameters, τ_k and α_k affect the rate at which habituation occurs, thereby determining the temporal resolution and range of the information obtained. The issues and tradeoffs involved are akin to memory depth versus resolution in dispersive delay line based models [5], [6]. We set $W_k(0)$ to zero for all k, employ positive values of α_k and τ_k such that $\alpha_k \tau_k + \tau_k < 1$ and normalize the input such that $I(t) \in [0,1]$. With these specifications, we can guarantee that the habituation process is stable. In fact we can guarantee that $W_k(t) \in [0,1]$ for all values of k and t.

3 MATHEMATICAL PROPERTIES

In this section we present theorems regarding the ability of a general category of neural networks, including habituation based networks, to approximate arbitrarily well any continuous, causal, time-invariant mapping f from one discrete sequence to another. Since all functions realized by TDNNs with arbitrarily large but finite input window size are continuous, causal, and time-invariant, the proofs of the theorems also imply that habituation based networks can realize any function which can be realized by a TDNN [10]. The key to the proof is to show that the memory structure realized by the habituated weights is a complete memory. Then so long as the feedforward stage is capable of uniformly approximating continuous functions, the overall network will be capable of mapping one sequence to another.

Due to space limitations, the theorems are stated without proof. An interested reader may obtain a copy of the complete proof by anonymous ftp to `ftp.lans.ece.utexas.edu`. The file in question is `/pub/hab_proof.ps`. The proof is related to previous work by Dr. I. W. Sandberg. In [9] he demonstrates a method for determining necessary and sufficient conditions for universal approximation of dynamic input-output mappings. Also in [10] he demonstrated a universal approximation proof for structures similar to that of Figure 1, with the exception that the

temporal encoding is performed with linear functionals.

Let X be the set of discrete time sequences for which $I \in X$ implies $I(t) \in [0,1]$. Let R be the set of all discrete time sequences. We are attempting to approximate a continuous, time-invariant, causal function, f, from X to R. We know additionally that any TDNN can be represented by such a function. As illustrated by Figure 1, we suggest transforming the inputs with m habituation units. After habituating, it is our conjecture that f can be approximated arbitrarily well by an MLP or RBF which takes the habituated values, $W_k(t+1)$, as inputs. If our conjecture is true, then a habituation based network is able to realize any function realizable by a TDNN.

Theorem 1 states that a two layer neural network with an exponential activation function and a particular structure for processing the inputs can universally approximate f. Theorem 2 states that habituation based networks are a specific case of the generalized structure exhibited in Theorem 1.

The results of the two theorems can be readily extended to include habituated MLP and RBF networks and to include multiple ($d > 1$) spatial input dimensions, $I_h(t)$, $1 \leq h \leq d$. In order to show that habituated MLPs and RBFs can perform the same approximations it is sufficient to show that the exponential function can be approximated arbitrarily well by a summation of sigmoids or gaussian functions. This is a special case of theorems which have already been proven for sigmoids by Cybenko [4] among others and for gaussian functions by Park and Sandberg [7]. The expansion of the result to multiple spatial dimensions follows directly from the proof of Theorem 1.

Before we state the theorems it is necessary to make a couple of definitions. First we will define the delay operator, T_β.

$$(T_\beta x)(t) = \begin{cases} 0 & \text{if } t < \beta \\ x(t-\beta) & \text{otherwise} \end{cases}$$

Next we define the concept of a *complete memory*. Let B be a set of mappings from X to R. B is a complete memory if it has the following four properties. First, there exist real numbers a and c such that $(bx)(t) \in (a,c)$ for all $t \in \mathcal{Z}^+$, $x \in X$, and $b \in B$. Second, for any $t \in \mathcal{Z}^+$ and any t_0 such that $0 < t_0 \leq t$, the following is true. If x and y are elements of X and $x(t_0) \neq y(t_0)$, then there exists some $b \in B$ such that $bx(t) \neq by(t)$. Third, if $b \in B$ then $(bT_\beta x)(t) = (bx)(t-\beta)$ for all $t \in \mathcal{Z}^+$, all $x \in X$ and any β such that $0 \leq \beta \leq t$. Fourth, every $b \in B$ is causal.

Theorem 1 *Let f be a continuous, causal, time-invariant function from X to R. If B is a complete memory then the following is true. Given any $\epsilon > 0$ and any arbitrarily large positive integer, t_0, there exist real numbers, a_j and c_{jk}, elements of B, b_{jk}, and natural numbers, p and m, such that the following inequality holds for all $x \in X$ and all $t < t_0$.*

$$\left| (fx)(t) - \sum_{j=1}^{p} a_j \exp\left(\sum_{k=1}^{m} c_{jk}(b_{jk}x)(t) \right) \right| < \epsilon \tag{4}$$

By proving Theorem 1 we demonstrate that a two layer static neural network with an exponential activation function and inputs operated on by elements

of a complete memory, B, can perform the same function as any TDNN. In order to show that a habituation based network is a special case of this type of generalized structure we state the following theorem.

Theorem 2 *Let $b_0 = W(0) = 1$. A prototypical habituation function is defined recursively as follows.*

$$bx(0) = W(1) = b_0 + \alpha\tau(1 - b_0) - \tau b_0 x(0) \tag{5}$$

$$bx(t) = W(t+1) = bx(t-1) + \alpha\tau(1 - bx(t-1)) - \tau bx(t-1)x(t) \tag{6}$$

Let B be the set of all such functions for all α and $\tau \in \mathcal{R}$ such that $\alpha > 0$, $\tau > 0$, $\tau < 1$, and $\alpha\tau + \tau < 1$. B is a complete memory.

It is important to notice that the input processing functions, b_{jk} used in Theorem 1 depend on j and thus the habituation parameters used also depend on j. This means that different hidden units in the feedforward network may have different input values. This dependency is not present in the structure illustrated in Figure 1. However, we can show that for any approximation g of the form discussed in Theorem 1, there is an equivalent network without this dependency. Let g be an approximation function of the form $\sum_{j=1}^{p} a_j \exp\left(\sum_{k=1}^{m} c_{jk} b_{jk} x\right)$. It is easy to see that given any such g one can find an h of the following form such that $g(x) = h(x)$ for all $x \in X$.

$$h(x) = \sum_{j=1}^{p} a_j \exp\left(\sum_{i=1}^{M} w_{ji} s_i(x)\right) \tag{7}$$

Here w_{ji} are real numbers which serve as weights to the hidden units and s_i are elements of a complete memory B.

Simply choose M to be the number of distinguishable functions b_{jk} used in g and let the sequence $\{s_i\}$ be the list of these distinguishable functions. For a particular s_i and a particular hidden node j, set w_{ji} to zero if the original b_{jk} corresponding to s_i was not present at hidden node j, otherwise set w_{ji} to the appropriate c_{jk}. An approximation of the form given by h has the same structure as that given in Figure 1, so the structure illustrated in Figure 1 is adequate.

Now we have demonstrated that habituated MLPs and RBFs are satisfactory substitutes for TDNNs. The question that remains is which are more efficient. The answer depends on the nature of the function that is being realized. The complexity of TDNNs depends on n, the input window size. The number of weighted inputs to each hidden unit in a TDNN is nd. For functions which only depend on recent values of the inputs, TDNNs can be quite efficient; but for functions which depend on long term temporal information or variable amounts of temporal information, TDNNs are not efficient solutions. For habituated networks, the required memory depth and resolution affects the choice of α and τ in Equation 3, and the number of habituated weights. Differents weights can have different values of α and τ, and the number of weights used can vary in different dimensions. Thus the memory structure can be optimized for a given mapping. The number of inputs to each hidden unit is $\sum_{i=1}^{d} m_i$, where m_i is the number of habituation values used to encode the ith component of $I(t)$.

A parallel can be drawn between finding a suitable m and finding a suitable number of hidden units in an MLP or RBF. In both cases there is no guarantee that the number required will not be inordinately large. However, in the case of

the MLP, a large set of problems have been found for which a small number of hidden units is suitable. The same is true for finding m. There may be many simple problems which are unsuitable for TDNNs because they require long term temporal information, but which can be solved with habituated networks with small values of m.

Since the output of the short-term memory stage is different for TDNNs and habituated networks, the complexity (number of hidden units) of the feedforward network needed at the output stage may also differ. For certain problems, habituated networks require a smaller feedforward output stage as compared to TDNNs for a given level of approximation. We have previously performed experiments using habituated MLPs to classify real SONAR data and have found that small habituated networks outperformed larger TDNNs. In fact we found that even $m = 1$ networks dramatically outperformed TDNNs with time window length of 5 or more [11]. Unfortunately due to the proprietary nature of the real SONAR data sets, they cannot be made public. Therefore, in the next section, we discuss experimental results on artificial Banzhaf sonograms, which can be easily generated and verified by other researchers.

The Gamma network proposed by Principe [5] has a structure which is similar to ours, but there are significant differences. For one, unlike habituation, the input transformation used in the Gamma network is linear. The nonlinearity of habituation can be readily seen by expanding $W(t+1)$ to obtain:

$$W(t+1) = \alpha\tau + \alpha\tau \sum_{j=1}^{t} \prod_{h=j}^{t} (1 - \alpha\tau - \tau I(h)) + W(0) \prod_{i=0}^{t} (1 - \alpha\tau - \tau I(i)) \quad (8)$$

Secondly, in the Gamma network each transformed input, W_i, depends on the previous transformed input, W_{i-1}, but in a habituated network each W_i is generated independently from the raw inputs.

4 EXPERIMENTAL RESULTS

The networks were trained on data sets consisting of Banzhaf sonograms, superpositions of 2-D gaussians in time and feature space [2]. The signals constructed are variable length (30-45 samples) sequences of 30 dimensional feature vectors. The reasons for choosing Banzhaf sonograms and specific details about the design of the data sets used are discussed in [12].

Three data sets were constructed with 7 classes each including a "noise only" class. The signals in the data sets were Banzhaf sonograms which were rotated, scaled, warped, and combined with additive noise. Figure 2 shows prototypical examples of each signal class in data set one (DS1).

Classification of DS1 is a problem which requires relatively long term temporal information. It is impossible to uniquely classify any signal based on only a short temporal window of inputs. For example, consider the prototypical signals of classes A, B, and C as illustrated in Figure 2. The signals in classes A and B are identical for the first twenty time samples, while classes A and C are identical for the last twenty time samples. Additionally, there is no time window of less than ten samples in any of the three signals that is not identical to a time window in one of the other two signals. This classification problem is obviously difficult for

short window TDNNs. In order to demonstrate the effectiveness of habituation for problems with a range of difficulties we have constructed two other data sets which do not depend as severely on long term temporal information. Data set 2 (DS2) and data set 3 (DS3) were generated using the same parameters as DS1 except that the centers of the component gaussians were shifted to reduce the overlap among the classes. In DS2, the component gaussians of samples of a particular class were all shifted uniformly, whereas in DS3, individual component gaussians were shifted so that a particular gaussian component might act as a tag for identifying the class membership of the signal. For this reason DS3, is the most local temporal information rich of the three data sets, and one would expect TDNNs to perform relatively better on DS3 than on either of the other data sets.

For our first experiment, we trained habituated MLPs and TDNNs on DS1 and DS2. The patterns in both data sets were randomly shuffled so that the classification of each pattern was uncorrelated with the classification of nearby patterns in the sequence. When the habituated MLP was tested, the habituation values, $W_{hk}(t+1)$, were calculated at each instance in time and then fed into the feedforward portion of the network. At each instance in time an output vector was computed and then used in classification. As an optimization, only the habituation values computed for the last ten samples in each signal were used to train the habituated MLP. This reduced training set method was used because habituation gradually builds up information about a signal as the signal is presented. During the first few samples of a signal, a habituated MLP does not have enough information to classify a signal. By the end of the signal, however, the network should have accumulated enough information to perform the classification. The reduced training set method was not used for TDNNs, however, because they exhibit no similar dichotomy in the way they store information over time.

For the first set of experiments we used habituated MLPs with random values of α_{hk} and τ_{hk} in the range $[0, 0.5]$. The α_{hk} and τ_{hk} parameters were not modified during training. The number of habituation units per input, m, was set to one. We found that for DS1 the habituated MLP, (HMLP), greatly outperformed a 5 sample time window TDNN and an MLP. All three networks utilized 10 hidden units. Increasing the number of hidden units was not found to greatly effect the performance.

Classification and detection of signals is accomplished using two thresholds, H and L. Detection occurs whenever a single output node has an output value, O_{max}, larger than all other output nodes, $O_{max} > H$, and all other output values are less than $1 - H$, for L consecutive input presentations. Classification is considered to be correct for a given signal if the only class detected within the length of the signal is the desired class. The best values of H and L may vary from network to network. For a fair comparison, for each network one should select the L for which the network achieved its highest classification rate for some H.

Figure 3 illustrates performance on DS1 in terms of the classification rate, i. e. the percentage of signals detected as well as correctly classified. Labels such as "L10" are included in the figures to denote the particular value of L used. As mentioned earlier, the best value of L was chosen for each classifier in order to make a fair comparison.

Results for DS2 similar to those found for DS1 are illustrated in Figure 4. For conciseness, this time the results are given in terms of the classification rate only. One notices in Figure 4, that although the HMLP has achieved a greater

maximum result than the TDNN, it does not have a better result at every value of H. This is an artifact of the particular L values used. The greater the value of L the steeper the decrease in performance for increasing H.

The HMLP was able to outperform the TDNN, because unlike the TDNN, the HMLP is capable of encoding long term temporal information. On DS1 this is particularly important, because classification is impossible without information from a large portion of the entire signal length.

One method for improving HMLP performance and reducing the complexity of the static classifier stage is to perform principal component analysis on the habituated values. The sequence of habituated values generated for a single pass through the training set is stored and the covariance matrix, M, is determined. Next, the eigenvalues and eigenvectors of M are computed. Finally the set of eigenvectors, λ_i, corresponding to the largest few eigenvalues are selected. Each vector of habituated weights, $W(t)$, is then replaced by the sequence of dot products, $W(t)^T\lambda_i$. These dot products are presented to the static classifier instead of the habituated values themselves.

By applying principal component analysis one can decrease the correlation among inputs to the static classifier, as well as, decreasing the number of inputs. For an HMLP with m=1 the number of inputs to the static classifier was reduced by a factor of 3, while simultaneously improving the classification rate on DS1 from 55 to 68 percent.

So far all the experiments discussed have focused on HMLPs with m=1, and randomly assigned habituation parameter values. Experiments which examine the effect of varying the m, α_{jk}, and τ_{jk} can be found in [13].

5 CONCLUSIONS

A multiply habituated MLP (MHMLP) can realize any function realizable by a TDNN with an arbitrarily large but finite input window. The relevant issues for comparing the two thus are which generates a simpler structure for a given accuracy level, and which network is easier and quicker to train. From the empirical results determined so far, MHMLPs are consistently more efficient than TDNNs. In fact for DS1 and DS2, a single habituation unit per input HMLP outperforms a TDNN with 5 sample time windows. Even on DS3, an MHMLP with m=2 was able to perform as well as the more complex 5 time sample TDNN. Note that because of the difficult nature of the data sets, the classification rates are low for both networks. Of course they perform much better than static classifiers (MLP, RBF) which fail because of substantial overlap among different classes at any time instant.

Simple MHMLPs seem to do particularly well on data sets which require long term temporal information for classification. In such cases, TDNNs need long time windows in order to perform well. Such TDNNs tend to be overly complex, leading to slow training and poor generalization.

Additionally it was found that large improvements in the complexity and performance of HMLPs can be obtained, by performing principal component analysis (PCA) on the habituated values. When PCA was used with DS1, a 13 percent improvement in performance was obtained, despite the fact that the number of parameters trained was reduced by more than twofold.

The performance of MHMLPs are quite robust with respect to parameter

values. The networks do well even when the habituation parameters are randomly assigned and never optimized. The next step in this research is to develop means of optimizing the habituation parameters, α and τ. Another topic for further study is to investigate other input transformation mechanisms, since a result similar to Theorem 1 can be proven for any set B which is a complete memory.

References

[1] Bailey, C. and Kandel, E. "Molecular Approaches to the Study of Short-Term and Long-Term Memory," *Functions of the Brain*. pp 98-129, Clarendon Press, Oxford, 1985.

[2] Banzhaf, W., and Kyuma, K. "The Time-into-Intensity-Mapping- Network" *Biological Cybernetics*, Vol. 66, pp 115-121, 1991.

[3] Byrne, J. H. and Gingrich, K. J. "Mathematical Model of Cellular and Molecular Processes Contributing to Associative and Nonassociative Learning in *Aplysia*," *Neural Models of Plasticity*, pp 58-70, 1989.

[4] Cybenko, G., "Approximation by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, Vol. 2, pp 303-314, 1989.

[5] de Vries, B. and Principe, J. C. "The Gamma Model - A New Neural Model for Temporal Processing," *Neural Networks*, Vol. 5, pp565-576, 1992.

[6] Mozer, M. C. Neural network architectures for temporal sequence processing. In Weigend, A. S. and Gershenfeld, N., editors, *Time Series Prediction*, pages 243–264. Addison Wesley, 1993.

[7] Park, J., and Sandberg, I. W. "Universal Approximation Using Radial-Basis-Function Networks," *Neural Computation*, Vol. 3, pp 246-257, 1991.

[8] Robin, D., Abbas, P., and Hug, L. "Neural Responses to Auditory Patterns," *J. Acoust. Soc. Am.* Vol. 87 part 4, pp 1673-1682, 1990.

[9] Sandberg, I. W. "Approximately-Finite Memory and the Theory of Representations," *International Journal of Electronics and Communications*, Vol. 46, No. 4, pp. 191-199, 1992.

[10] Sandberg, I. W. Structure theorems for nonlinear systems. *Multidimensional Systems and Signal Processing*, 2:267–286, 1991.

[11] Stiles, B. W., "Dynamic Neural Networks for the Classification of Oceanographic Data," Master's thesis, Department of Elec. Eng., Univ. of Texas at Austin, 1994.

[12] Stiles, B. W., and Ghosh, J. "A Habituation Based Mechanism for Encoding Temporal Information in Artificial Neural Networks", (invited paper) Proc. SPIE Conf. on Applications and Science of Artificial Neural Networks IV, SPIE Proc. Vol. 2492, pp. 404-4515, 1995.

[13] Wang, D. and Arbib, M. "Modeling the Dishabituation Hierarchy: the Role of the Primordial Hippocampus," *Biol. Cybern.* Vol. 67, pp 535-544, 1992.

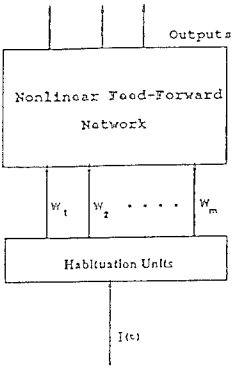


Figure 1: Structure of Habituated Neural Networks

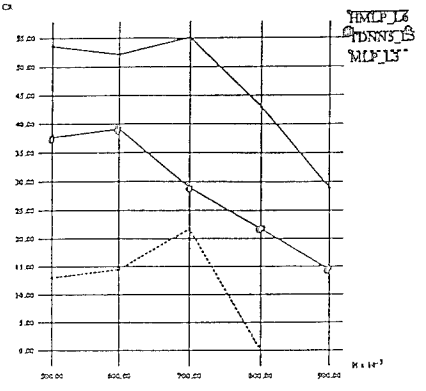


Figure 3: Classification Rate (CR) on Data Set 1

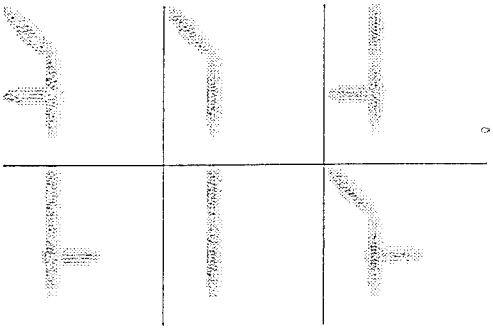


Figure 2: Banzhaf Mother Signals for classes A-F(clockwise from the top left) of Data Set 1. The vertical and horizontal axes depict time(increasing top to bttom) and frequency(increasing left to right) respectively.

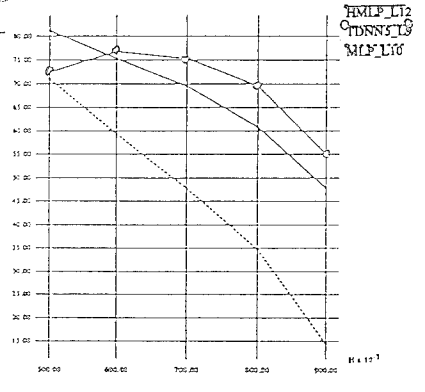


Figure 4: Classification Rate on Data Set 2

A Numerical Approach for Estimating Higher Order Spectra Using Neural Network Autoregressive Model

Naohiro TODA and Shiro USUI
Information and Computer Sciences, Toyohashi University of Technology,
Tempaku, Toyohashi, 441, JAPAN
fax:+81-532-46-7806, e-mail: usui@bpel.tutics.tut.ac.jp

ABSTRACT

A method for parametric estimation of higher order spectra of time series using a nonlinear autoregressive model based on multi-layered neural networks (NNAR model) is presented. In the real world problems, there exist signals that can not be described sufficiently by linear time series models such as AR or ARMA models. In order to characterize such signals, several nonlinear time series models have been investigated in recent years. However, in contrast with the case of linear models, there are a few parametric approaches that estimate the higher order statistical characteristics of observed time series using such nonlinear time series models. It is very difficult to derive analytically explicit formulations of higher order spectra from the expressions of such nonlinear time series models. In this study, employing numerical techniques, we constructed a parametric estimator of higher order spectra. It consists of following steps: 1. training an NNAR model on the given time series, 2. iteration of numerical integrals for solving the joint probability density function, 3. calculation of higher order cumulant functions by renewal equations based on the joint probability density function solved in 2., 4. multidimensional discrete Fourier transforms of higher order cumulant functions calculated in 3. We also show that any NNAR model with finite valued weights satisfies a sufficient condition of convergence.

1 Introduction

In the case of Gaussian time series, we can obtain a sufficient information about time series from the power spectra as the statistical characteristics of the time series can be described completely by the first and second order moments. Specifically, the linear AR model has been used widely in many areas of engineering as a conventional parametric estimator of power spectrum [1].

However, there exist many signals that have non-Gaussian nature. Since the nonlinear time series models can generate non-Gaussian time series, several authors have proposed them to characterize such non-Gaussian signals [2][3]. However, in contrast with the case of linear models, there are a few parametric approaches estimating the higher order spectra [4] of the observed time series using nonlinear time series models, except for some special cases [5]. In this study, we constructed a parametric estimator of higher order spectra using a nonlinear au-

0-7803-2739-X/95 $4.00 © 1995 IEEE

toregressive model based on the multilayer neural networks. It is very difficult to analytically derive the explicit forms of the joint probability density function from the expressions of such nonlinear time series models. Although the analytical solutions are not available, we can construct a parametric estimator of higher order spectrum using some numerical solutions. In general, if there exists a stationary joint probability density function of a nonlinear autoregressive model, this function satisfies a certain integral equation. Here we take an approach to solve the integral equation with an iterative formula using numerical integrals.

2 Nonlinear Autoregressive Model

A general form of the nonlinear autoregressive model is given as follows:

$$x_k = F[\boldsymbol{x}_{k-1,q}] + e_k \tag{1}$$

where, $\boldsymbol{x}_{k-1,q} = (x_{k-1}, x_{k-2}, \cdots, x_{k-q})^T$, e_k is an i.i.d of Gaussian $(\mathcal{N}(\mu_e, \sigma_e^2),\ \mu_e < \infty, \sigma_e^2 < \infty)$, $F[\ ]$ is a continuous function:$\mathbb{R}^q \rightarrow \mathbb{R}$, and $(\cdot)^T$ means the transpose of a vector.

We can rewrite (1) as a state space representation:

$$\boldsymbol{x}_{k,q} = f(\boldsymbol{x}_{k-1,q}, e_k), \tag{2}$$

where $f(\boldsymbol{x}_{k-1,q}, e_k) = (F[\boldsymbol{x}_{k-1,q}] + e_k, x_{k-1}, x_{k-2}, \cdots, x_{k-q+1})^T$. Equation (2) describes $\boldsymbol{x}_{k,q}$ as a real-valued Markov chain on the state space $\mathbb{R}^q$. The behavior of $\boldsymbol{x}_{k,q}$ depends on the shape of the continuous function F.

It has been shown that multilayer neural networks can avoid the difficulty called "curse of dimensionality" from which ordinary methods for function approximation suffer [8]. We construct the continuous function F by multilayered neural network as follows:

$$F[\boldsymbol{x}_{k-1,q}] = \boldsymbol{V}\Phi \tag{3}$$

where $\Phi = (o_1, \cdots, o_m)^T$, $o_i = \phi(\boldsymbol{W}_i \boldsymbol{x}_{k-1,q} - \theta_i)$ is an output of ith hidden unit, $\phi(x) = \dfrac{1}{1 + exp(-x)}$ is a sigmoid function, $\boldsymbol{V} \in \mathbb{R}^m$ is an output weight vector, $\boldsymbol{W}_i \in \mathbb{R}^q$ is an input weight vector connected to the ith hidden unit, θ_i is a bias for ith hidden unit, b is a bias for output unit, q is a total number of input units, and m is a total number of hidden units.

The above model is called a neural network autoregressive model (NNAR). This type of time series model has been applied to behavior prediction of various systems [6]. Fitting the NNAR model to the given time series is achieved by minimizing predictive error using conventional learning algorithms.

3 Stationarity of NNAR model

A sufficient condition for existence of stationary joint probability density function of Markov chain (2) was given by Chan [7]. In this section we show that NNAR model satisfies this condition.

If f in (2) is assumed to be decomposable as follows:

$$f(\boldsymbol{x}_{k-1,q}, e_k) = T(\boldsymbol{x}_{k-1,q}) + S(\boldsymbol{x}_{k-1,q}, e_k) \tag{4}$$

where $T : \mathbf{R}^q \to \mathbf{R}^q$, $S : \mathbf{R}^{q+1} \to \mathbf{R}^q$ and an extra sequence $\boldsymbol{z}_k$ is defined as $\boldsymbol{z}_n = T(\boldsymbol{z}_{n-1})$, $\boldsymbol{z}_n \in \mathbf{R}^q$. Furthermore, we assume S can be written as:

$$S(\boldsymbol{x}_{k-1,q}, e_k) = F[\boldsymbol{x}_{k-1,q}] + e_k. \tag{5}$$

Under the following conditions **A1.**$\sim$**A5.**, Markov chain defined by (2) is geometrically ergodic, provided that f is continuous everywhere and continuously differentiable [7]. If Markov chain is geometrically ergodic, it always has a stationary joint probability density function.

A1. $\mathbf{0} = T(\mathbf{0})$, and $\exists K, c > 0$ such that $\forall n \geq 0$; starting with $\boldsymbol{z}_0 \in \mathbf{R}^q$, $\| \boldsymbol{z}_n \| \leq Ke^{-cn} \| \boldsymbol{z}_0 \|$, where $\| \cdot \|$ denotes the Euclidian norm in $\mathbf{R}^q$. **A2.** e_k has probability density function $\pi(\cdot)$ that is continuous and positive everywhere. **A3.** $F[\cdot]$ is bounded over bounded sets. **A4.** $\exists M > 0$, such that $\forall \boldsymbol{x}, \boldsymbol{y} \in \mathbf{R}^q$, $\| T(\boldsymbol{x}) - T(\boldsymbol{y}) \| \leq M \| \boldsymbol{x} - \boldsymbol{y} \|$. **A5.** For some $\tau > 0$, $\mathbf{E}[\| S(\boldsymbol{x}_{k-1,q}, e_k) \| \text{ given } \boldsymbol{x}_{k-1,q} = \boldsymbol{x}] \leq \tau$, $\forall \boldsymbol{x} \in \mathbf{R}^q$.

f in (2) can be rewritten as:

$$\begin{aligned} f(\boldsymbol{x}_{k-1,q}, e_k) = {} & (0, x_{k-1}, x_{k-2}, \cdots, x_{k-q+1})^T \\ & + (F[\boldsymbol{x}_{k-1,q}] + e_k, 0, \cdots, 0)^T. \end{aligned} \tag{6}$$

Then, we can obtain a form (4) by defining T and S as:

$$\begin{aligned} T(\boldsymbol{x}_{k-1,q}) &= (0, x_{k-1}, x_{k-2}, \cdots, x_{k-q+1})^T, \\ S(\boldsymbol{x}_{k-1,q}, e_k) &= (F[\boldsymbol{x}_{k-1,q}] + e_k, 0, 0, \cdots, 0)^T. \end{aligned} \tag{7}$$

T satisfies clearly **A1.** and **A4.** Because e_k is Gaussian, **A2.** is also satisfied. The sigmoid function $\phi(\cdot)$ is bounded over $\mathbf{R}$, consequently, for all m, V_i $(i = 1, \cdots, m)$, $b(< \infty)$, there exists $c < \infty$ such that $|F[\boldsymbol{x}_{k-1,q}]| < c$, then **A3.** is satisfied. From this and (5), and properties of e_k, we have

$$\mathbf{E}[\| S(\boldsymbol{x}_{k-1,q}, e_k) \| \, | \boldsymbol{x}_{k-1,q} = \boldsymbol{x}] = F[\boldsymbol{x}] + \mathbf{E}[e_k] < c + \mu_e. \tag{8}$$

Substituting $c + \mu_e$ for τ, we can show that F given by (3) satisfies **A5.** Hence, we established geometric ergodicity of NNAR model.

4 Numerical Solution of Stationary Joint Probability Density Function

Suppose the following renewal equation of joint probability density function $p^{(n)}(\boldsymbol{x})$:

$$p^{(n)}(\boldsymbol{x}_{k,q}) = \int_{-\infty}^{\infty} \pi(x_k - F[\boldsymbol{x}_{k-1,q}]) p^{(n-1)}(\boldsymbol{x}_{k-1,q}) dx_{k-q}, \qquad (9)$$

where π denotes probability density function of e_k. If $\boldsymbol{x}_{k,q}$ satisfies a sufficient condition of geometrical ergodicity, starting with $\forall p^{(0)}(\boldsymbol{x})$, then there exists a unique limit: $\lim_{n\to\infty} p^{(n)}(\boldsymbol{x}) = p(\boldsymbol{x})$. The joint probability density function $p(\boldsymbol{x})$ satisfies following integral equation:

$$p(\boldsymbol{x}_{k,q}) = \int_{-\infty}^{\infty} \pi(x_k - F[\boldsymbol{x}_{k-1,q}]) p(\boldsymbol{x}_{k-1,q}) dx_{k-q}. \qquad (10)$$

Clearly, $p(\boldsymbol{x})$ is a stationary joint probability density function. If we know the explicit form of stationary joint probability density function, we can derive the higher order spectra such as power spectrum or bispectrum. However, it is very difficult to solve integral equation (10). For this reason, we adopted a numerical version of renewal equation (9), using a numerical integral with finite duration (x_{max}, x_{min}). By this way we can obtain the stationary joint probability density function after a sufficient number of numerical iterations. Moeanaddin et.al. [9] also tried to solve the stationary joint probability density function of self-existing threshold autoregressive model (SETAR) using similar numerical technique.

5 Power Spectrum and Bispectrum

There are a few parametric approaches that estimate higher order spectra using universal nonlinear autoregressive models. Here we present a numerical method of computation of power spectrum and bispectrum based on the stationary joint probability density function solved in 4. Other higher order spectra can be calculated by employing similar procedures.

5.1 Power Spectrum

Power spectrum (spectral density function) is defined as discrete Fourier transform of second order central moment function (cumulant). Second order central moment function ${}^2R(-r)$ is given as follows:

$${}^2R(-r) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x_k - \mu)(x_{k-r} - \mu) p(x_k, x_{k-r}) dx_k dx_{k-r}. \qquad (11)$$

Where, μ is a first order moment (average).

$$\mu = \int_{-\infty}^{\infty} x_k p(x_k) dx_k \qquad (12)$$

and $p(x_k)$is a marginal density function of joint probability density function $p(\boldsymbol{x}_{k,q})$:

$$p(x_k) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(\boldsymbol{x}_{k,q}) dx_{k-1} \cdots dx_{k-q+1}. \tag{13}$$

We must know $p(x_k, x_{k-r})$ in order to execute numerical integration of (11). In the case of $r < q$, the marginal density function of $p(\boldsymbol{x}_{k,q})$ can be computed directly as follows:

$$p(x_k, x_{k-r}) = \int_{-\infty}^{\infty} \cdots \int_{\infty}^{\infty} p(\boldsymbol{x}_{k,q}) dx_{k-1} \cdots dx_{k-r+1}, \\ dx_{k-r-1}, \cdots, dx_{k-q+1}. \tag{14}$$

In the case of $r \geq q$, we can construct a renewal formula using a property of the Markov chain. Namely, from the $(q+1)$th order joint probability density function $p(\boldsymbol{x}_{k-1,q}, x_{k-r})$ we can derive $p(\boldsymbol{x}_{k,q}, x_{k-r})$ that has a lag interval with one lag step longer than $p(\boldsymbol{x}_{k-1,q}, x_{k-r})$:

$$p(\boldsymbol{x}_{k,q}, x_{k-r}) = \int_{-\infty}^{\infty} \pi(x_k - F[\boldsymbol{x}_{k-1,q}]) p(\boldsymbol{x}_{k-1,q}, x_{k-r}) dx_{k-q}. \tag{15}$$

Then, setting initial joint probability density function $p(\boldsymbol{x}_{k,q}, x_{k-q})$ as:

$$p(\boldsymbol{x}_{k,q}, x_{k-q}) = \pi(x_k - F[\boldsymbol{x}_{k-1,q}]) p(\boldsymbol{x}_{k-1,q}), \tag{16}$$

we can compute $p(\boldsymbol{x}_{k,q}, x_{k-r})$ for all $r(\geq q)$. $p(x_k, x_{k-r})$ is given as a marginal density function of $p(\boldsymbol{x}_{k,q}, x_{k-r})$:

$$p(x_k, x_{k-r}) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} p(\boldsymbol{x}_{k,q}, x_{k-r}) dx_{k-1} \cdots dx_{k-r+1} \\ dx_{k-r-1} \cdots dx_{k-q}. \tag{17}$$

Finally, power spectrum can be represented in a form of discrete Fourier transform of ${}^2R(r)$:

$$P(\omega) = \sum_{r=-\infty}^{\infty} {}^2R(r) \exp(-j\omega r). \tag{18}$$

5.2 Bispectrum

Bispectrum $B(\omega_1, \omega_2)$ is defineded as two-dimensional discrete Fourier transform of third order central moment function ${}^3R(r, s)$:

$$B(\omega_1, \omega_2) = \sum_{r=-\infty}^{\infty} \sum_{s=-\infty}^{\infty} {}^3R(r, s) \exp(-j(\omega_1 r + \omega_2 s)), \tag{19}$$

where,

$$^3R(-r,-s) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}(x_k-\mu)(x_{k-r}-\mu)(x_{k-s}-\mu) \\ p(x_k,x_{k-r},x_{k-s})dx_k dx_{k-r}dx_{k-s}. \quad (20)$$

The third order joint probability density function $p(x_k, x_{k-r}, x_{k-s})$ in (20) can be calculated using following renewal procedure:

In the case of $r \leq s < q$, $p(x_k, x_{k-r}, x_{k-s})$ is given as a marginal density function of $p(\boldsymbol{x}_{k,q})$:

$$p(x_k,x_{k-r},x_{k-s}) = \int_{-\infty}^{\infty}\cdots\int_{\infty}^{\infty}p(\boldsymbol{x}_{k,q})dx_{k-1}\cdots \\ dx_{k-r+1}dx_{k-r-1}\cdots dx_{k-s+1}dx_{k-s-1}\cdots dx_{k-q+1}. \quad (21)$$

In the case of $q \leq r \leq s$, the $(p+2)$th order joint probability density function $p(\boldsymbol{x}_{k,q}, x_{k-r}, x_{k-s})$ is calculated as follows:

$$p(\boldsymbol{x}_{k,q},x_{k-r},x_{k-s}) = \int_{-\infty}^{\infty}\pi(x_k - F[\boldsymbol{x}_{k-1,q}]) \\ p(\boldsymbol{x}_{k-1,q},x_{k-r},x_{k-s})dx_{k-q}. \quad (22)$$

To set the interval between r and s an arbitrary length, we use the following formula:

$$p(\boldsymbol{x}_{k,q},x_{k-s}) = \int_{-\infty}^{\infty}\pi(x_k - F[\boldsymbol{x}_{k-1,q}])p(\boldsymbol{x}_{k-1,q},x_{k-s})dx_{k-q}, \quad (23)$$

where $r = q$. Then, for all r and s, we can obtain $p(\boldsymbol{x}_{k,q}, x_{k-r}, x_{k-s})$ by taking marginal density function as:

$$p(x_k,x_{k-r},x_{k-s}) = \int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}p(\boldsymbol{x}_{k,q},x_{k-r},x_{k-s}) \\ dx_{k-1}\cdots dx_{k-q}. \quad (24)$$

6 Numerical Example

In this section, we present a numerical example of parametric estimation of power spectrum and bispectrum. As an example of the time series we generated following signal:

$$x_k = \sin(\omega_0 k) + 0.5\sin(2\omega_0 k) + e_k \quad (k = 0, 1, \cdots, 2047) \quad (25)$$

where, e_k is a Gaussian white noise: $\mathcal{N}(0, 0.0025)$ and $\omega_0 = 3.14/5 \simeq \pi/5$. Figure 1 shows the distribution of the data on the state space (x_k, x_{k-1}). We trained the NNAR model with the structure 2-input(lag order), 7-hidden, and 1-output. Figure 2 shows the joint probability

density function $p(x_k, x_{k-1})$ computed by the procedure described in **4**. It can be seen that $p(x_k, x_{k-1})$ well represents the complex structure of the distribution of data (Figure 1).

Figure 3 shows the estimated power spectrum. The estimated power spectrum has two clear peaks at $\omega = (\omega_0,\ 2\omega_0)$. This means that the NNAR model can acquire the second order stochastic characteristics of the signal correctly, while the linear AR model with the same lag order 2 can not detect two spectral peaks.

Furthermore, we estimated the bispectrum by the procedure mentioned in **5.2**. Figure 4 shows the estimated $|B(\omega_1, \omega_2)|^2, (\omega_1 \geq 0, \omega_2 \geq 0)$. The true bispectrum of the signal (25) has a peak at (ω_0, ω_0). It can be seen that the estimated bispectrum approximates well the true structure.

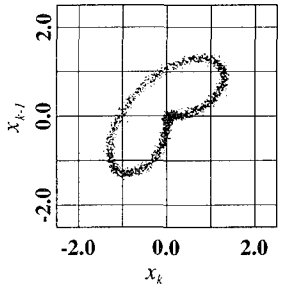


Fig.1 The state space plots of data

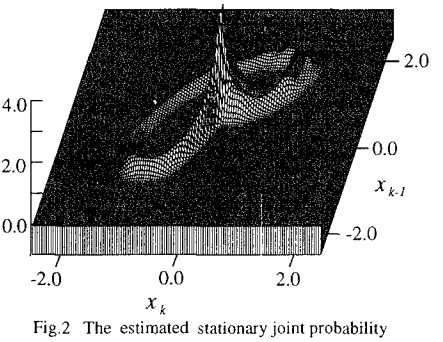


Fig.2 The estimated stationary joint probability density function

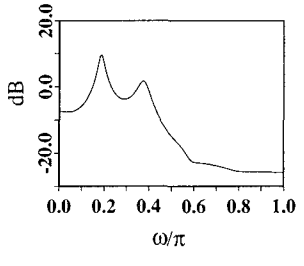


Fig. 3 The estimated power spectrum

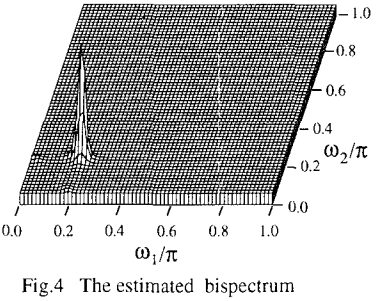


Fig.4 The estimated bispectrum

7 Conclusion

A method for parametric estimation of higher order spectra of time series using a nonlinear autoregressive model based on NNAR model is presented. Advantages of the NNAR model as a universal time series model are follows. As clarified in **3** for any finite weight values, the NNAR model has always stationary joint probability density function. Consequently, no special modifications of learning algorithm are required in order to keep the model stationary. Arbitrary continuous function F

with several variables in the difference equation (1) can be approximated by multilayer neural networks. Furthermore, multilayer neural networks can avoid so called difficulty "curse of dimensionality" [8].

Currently we are investigating the evaluation of the computational error and speeding up the calculations. Bearing in mind the enormous power of future computers, the presented method can be one of the standard signal processing techniques.

References

[1] Usui S. and Toda N.:"An overview of biological signal processing: non-linear and nonstationary aspects", Frontiers Med. Biol. Eng., 3, 2, pp.125-129, 1991.

[2] Tong H.:"Nonlinear Time Series" , Oxford Science Publications, 1990.

[3] Chen S. and Billings S.A.:"Modelling and analysis of non-linear time series", Int. J. Control, 50, 6, pp.2151-2171, 1989.

[4] Brillinger D.R.:"An introduction to polyspectra", Ann. Math. Statist., 36, pp.1351-1374, 1965.

[5] Rao T.S.: "Spectral and bispectral methods for the analysis of non-linear (non Gaussian) time series signals", Lecture Notes Control Inf. Sci., 106, pp.18-42, 1988.

[6] Weigend A.S., Huberman B.A. and Rumelhart D.E.: "Predicting the future: a connectionist approach", International Journal of Neural Systems, 1, 3, pp.193-209, 1990.

[7] Chan K.S.:"Deterministic stability, stochastic stability, and ergodicity", in H.Tongs book [2], Appendix 1, 1990.

[8] Barron A.R.:"Universal approximation bounds for superpositions of a sigmoidal function", IEEE, Trans. Inf. Theory, 39, 3, pp.930-945, 1993.

[9] Moeanaddin R., Tong H.:"Numerical evaluation of distributions in non-linear autoregression", J. Time Series Analysis, 11, 1, pp.33-48, 1990.

A FUZZY NEURAL NETWORK APPROACH BASED ON DIRICHLET TESSELATIONS FOR NEAREST NEIGHBOR CLASSIFICATION OF PATTERNS

K. Blekas, A. Likas and A. Stafylopatis
Department of Electrical and Computer Engineering
National Technical University of Athens
157 73 Zographou, Athens, Greece
Tel.: +301 7485056, Fax: +301 7784578
e-mail: andreas@theseas.ntua.gr

Abstract A neural network classifier using fuzzy set representation of pattern classes is presented. Network construction and learning is performed incrementally in a single pass by building an aggregate of space-filling regions that constitutes a simplified variant of the construction known as Dirichlet tesselation (or Voronoi diagram). Each region is delimited by a set of hyperplanes and is endowed by a fuzzy membership function that forms the basis of learning and recall. Experimental results concerning difficult recognition problems show that the proposed approach is very successful in applying fuzzy sets to pattern classification.

1 INTRODUCTION

Several models have been developed during the last years in an attempt to combine fuzzy systems and neural networks. Some of them focus on applying this synergistic combination to building efficient pattern classifiers [5, 7, 9], as the application of fuzzy sets to pattern classification has been considered for many years.

The fuzzy neural network presented here is an example of neural network classifier that builds decision boundaries by creating subsets of the pattern space. The creation of fuzzy subsets is based on the partition of the n-dimensional space in a way that constitutes a direct adaptation of the notion of Dirichlet tesselations, also known as Voronoi diagrams

0-7803-2739-X/95 $4.00 © 1995 IEEE

or Thiessen polygons [1].

A Dirichlet tesselation of a set S of points (called *sites*) is a partition of the n-dimensional space into convex polytopes. Each polytope which is also called 'cell' or 'tile' belongs to one site of the set S and contains all points of the space for which this site is the closest, or the one with the dominant influence. Each cell is defined with respect to an arrangement of halfspaces as the intersection of a finite number of hypeplanes, which are the perpendicular bisectors of the segments joining pairs of sites. From a given set of n-dimensional points classical Dirichlet tesselation can be constructed by obtaining the convex hull of these points [1] or by incremental insertion of the regions of these sites [3, 4]. Dirichlet tesselations express the proximity information of a set of given points in a very explicit and computationally useful manner that makes it applicable in many diverse areas among which are biology, visual perception, crystallography and archeology.

The application of Dirichlet tesselations to the design of neural networks has been considered recently. In [8] two neural network construction algorithms for pattern classification are proposed that rely directly or indirectly on the Dirichlet tesselation of the space based on the given training patterns. An efficient adaptation of the above algorithms is presented in [6], whereas a systematic procedure for designing neural networks following the same principle is formulated in [2]. In this paper we develop an analogous construction approach which incorporates the idea of fuzzy set classes by defining fuzzy decision boundaries for the regions of the tesselation. The proposed scheme allows for efficient online supervised learning using appropriately defined fuzzy membership functions during both learning and recall.

A description of the proposed fuzzy classification network is provided in the next section, while the network construction algorithm is presented in Section 3. Section 4 concerns experimental results from the application of the approach to difficult classification problems. Section 5 briefly describes the extension of the model to the case of both continuous and discrete attributes, and finally Section 6 summarizes the main conclusions.

2 FUZZY SET CLASSES AND NETWORK TOPOLOGY

Consider a classification problem with n continuous attributes, such that the n-dimensional patterns belong to p distinct classes. By means of the proposed construction scheme, we shall define a set of *regions* filling the feature space such that each region is associated with exactly one from the pattern classes. A properly computed fuzzy membership function (taking values in $[0, 1]$) indicates the degree to which a pattern is contained within each of the regions. During operation, the region with the

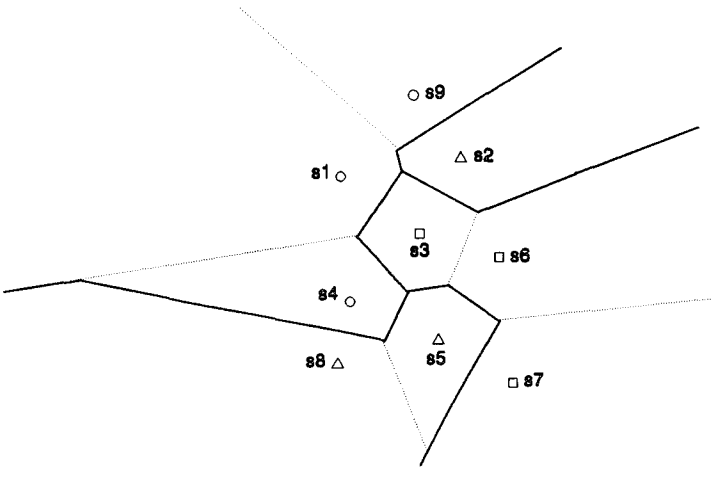


Figure 1: A partition of the plane

maximum membership value is selected and the class associated with the winning region is considered as the desicion of the network.

Learning in the fuzzy classification network consists of creating and adjusting regions and associating a class label to each of them. Each region is characterized by a point, which will be called the site of the region, and can be expressed as the intersection of a finite number of closed half-spaces defined by hyperplanes that separate regions of different classes. Regions corresponding to the same class can be overlapping. In general, not all training patterns constitute sites of regions. Following the principle of Dirichlet tesselations, the points of a region are closer to the site of the region than to all other sites belonging to different classes. This feature constitutes a relaxation with respect to the strict definition of Dirichlet tesselations and implies a construction scheme that prescribes no separating hyperplane between regions of the same class. Figure 1 represents such a convex construction on the 2-dimensional space, based on the Dirichlet tesselation principle, for a set of 9 input points with three seperated classes (dotted lines would be present in a classical Voronoi diagram).

When an input pattern $a = (a_1, \ldots, a_n)$ is presented to the network during operation, the corresponding membership function for each region is computed. The membership function $b_i(a)$ for the ith region must measure the degree to which the given pattern falls inside or outside the region. This can be considered as a measurement of how far is situated the pattern from all the hyperplanes which define the region. When the pattern a is in the interior of the region and far from the hyperplanes then $b_i(a)$ approaches 1, the value 1 meaning that the point is very

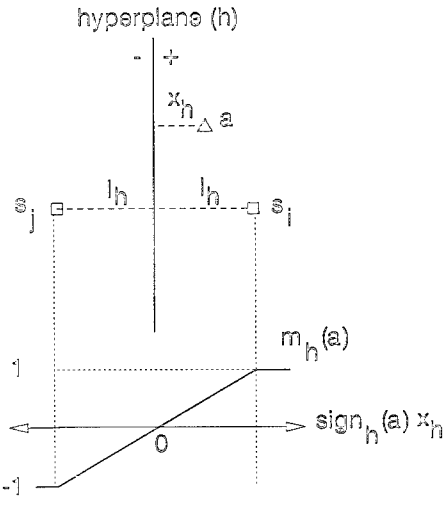


Figure 2: Fuzzy decision boundaries

close to the site of the region. When the pattern falls outside the region then the membership value approaches zero, the value 0 meaning that this point is close to some other site. A function following the above guidelines is the average value of the normalized vertical distances x_h of the pattern from all hyperplanes h supporting the region. Each distance x_h is normalized with respect to the distance l_h of the site of the region from hyperplane h.

Consider the function $\text{sign}_h(a)$ which describes on which side of the hyperplane h lies the pattern a. If it lies in the positive half space h^+ we have $\text{sign}_h(a) = 1$, else if it lies in the negative half space h^- then $\text{sign}_h(a) = -1$. Also consider the quantities v_{ih} which take the values 1 or -1 depending on whether the site i is situated in the positive or negative half-space defined by the hyperplane h, respectively.

The membership function taking values in $[0, 1]$ can be computed as follows:

$$b_i(a) = \frac{1}{2|H_i|} \sum_{h \in H_i} v_{ih} m_h(a) + \frac{1}{2} \tag{1}$$

where H_i is the set of hyperplanes defining the region i (having cardinality $|H_i|$) and m_h has the following form (Figure 2):

$$m_h(a) = \begin{cases} 1 & \text{if } x_h > l_h \text{ and } \text{sign}_h(a) = 1 \\ -1 & \text{if } x_h > l_h \text{ and } \text{sign}_h(a) = -1 \\ \text{sign}_h(a) x_h / l_h & \text{otherwise} \end{cases} \tag{2}$$

Other choices can be made for the computation of the membership functions, e.g. the form adopted in [5].

The fuzzy classifier can be implemented as a neural network that exploits the fuzzy set structure and allows for efficient implementation.

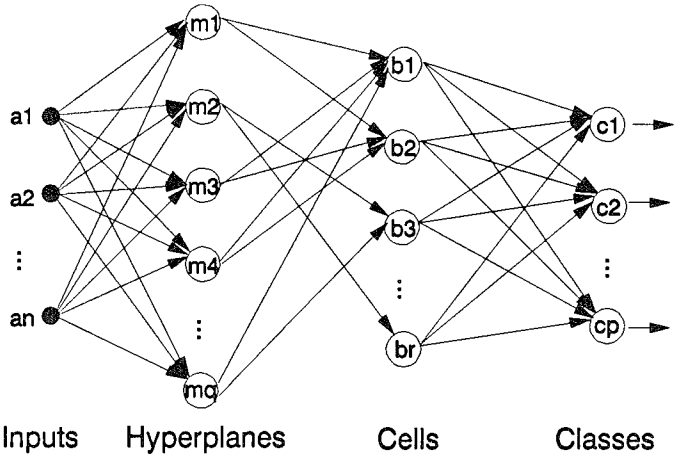


Figure 3: The Fuzzy Neural Network Classifier

Figure 3 illustrates the neural network that implements this approach. It consists of three layers such that connections exist between successive layers. The number of nodes in the first layer is equal to the number q of hyperplanes that define regions. Each first layer node computes the value of the function m_h for every input pattern using equation (2). The second layer contains as many nodes as the number r of regions. The output of each node of this layer represents the membership value of the pattern for the corresponding region as computed in equation (1). The connections between nodes of the first and second layer associate regions with their supporting hyperplanes and assume the values v_{ih} defined above. The last layer embodies nodes which correspond to the set of p classes. The connections u_{ji} between the second and third layer take binary values, such that $u_{ji} = 1$ if i is a region of class j and $u_{ji} = 0$ otherwise. Each node of the third layer computes the degree to which the input pattern fits within class j. The function that performs this computation is the fuzzy union of the appropriate region fuzzy set values. This operation is defined for each of the p classes as

$$c_j = \max_{i=1}^{r}[u_{ji}b_i(a)] \tag{3}$$

3 LEARNING AND CONSTRUCTION

Consider a set A of training patterns. The learning algorithm creates a division of the feature space by appropriately constructing regions. Each region is defined by hyperplanes that are successively created to separate neighboring regions of different classes. Implementation of the

below scheme requires the definition of the appropriate data structures for holding all the information necessary during the construction.

At an initialization step, the first two training patterns considered (which should be of different classes) become the sites of the first regions which are originally separated by a hyperplane (the perpendicular bisector of the segment joining the two sites). These regions will be restricted in the sequel as new sites are created.

During learning, each training pattern a_k is presented once and the following general step is performed.

- First we compute the values of the membership functions $b_i(a_k)$, as defined previously, for all existing regions i. Then we find the regions whose membership values exceed a given threshold value θ ($0 \leq \theta \leq 1$), which is generally taken high (typically, greater than 0.7).

- If all the regions meeting the above criterion belong to the same class as the presented pattern a_k, no further action is taken.

- If one or more of the selected regions belong to classes different than that of pattern a_k, then the latter becomes a new site and its region is constructed by drawing bisecting hyperplanes between this site and its neighboring sites of different classes. No hyperplane is created between the new site and sites belonging to the same class, thus allowing for overlapping. The neighboring regions of the new region are successively determined by applying a simple adaptation of standard techniques used in the creation of Dirichlet tesselations by incremental insertion of sites [3, 4].

- The new site acquires its region by winning territory from the regions of its neighbors (belonging to different classes). As some of the already presented (non-site) patterns may be contained in the affected regions, it should be checked whether such patterns are now included in the newly created region. Thus, these patterns are successively examined and if they are contained in the new region they create their own new regions by winning territory from the latter, following the procedure applied in the previous step for a_k. Obviously, this construction of new regions need not take place for all such points, since several of them may be covered by each newly created region of the correct class.

4 EXPERIMENTAL RESULTS

We have studied the proposed fuzzy neural network classifier on a variety of difficult classification problems. We have tried to select databases whose instances are defined on a high-dimensional space so that the applicability of the Dirichlet tesselation approach on such problems could

be evaluated. In addition, some of the data sets were noisy containing hard examples so as to illustrate the operation and performance of the fuzzy neural network classifier. To evaluate the effectiveness of our model we have mainly compared it with the fuzzy min-max classifier [9].

The first data set is the Johns Hopkins University ionosphere database which is a collection of radar data. The ionosphere data set consisted of 351 feature vectors described by 34 continuous valued attributes with two decision classes (either show evidence of some type of structure in the ionosphere or not). The data set was divided into a training set of 200 examples that were used to adjust the network hypeplanes and convex polytopes, while the remaining 151 examples were applied to the constructed network structure to estimate the performance of the proposed fuzzy neural classifier. In all of our experiments we trained the network for certain θ values and then computed the percentage of correct classification over the test set. Best results were found for $\theta = 0.75$. For this parameter value the network consisted of 127 cells and the success rate was 97%. On the other hand, using the same data set to train a fuzzy min-max neural network classifier several experiments were conducted for different values of θ. The best classification rate obtained was 95.5%

The second data set we used to train and test our fuzzy neural classifier was the Fisher's Iris data. Iris data is a collection of 150 four-dimensional featute vectors in three separate classes, 50 for each class. We considered a training set and a test set of size 75, each of them containing 25 examples of each of the three iris classes. After a series of experiments using different values of the parameter θ we found the best classification rate 97.3% for $\theta = 0.75$ in which we obtained 22 polygon cells. For the fuzzy min-max classifier the best classification rate for the same data set was exactly the same [9].

We have also used the James Cook University Thyroid gland database in our model. Thyroid database is a collection of 215 feature vectors consisting of 5 continuous attributes, such that the vectors belong to three classes. Any of these three decision classes defines a prediction of a patient's thyroid to the class of euthyroidism, hypothyroidism or hyperthyroidism. The database is divided into 150 instances of first class, 35 instances of second and 30 of last class. We used a training set of size 100 while the remaining data set (size 115) was used as testing set. Best performance was obtained for $\theta = 0.8$ (34 polygon cells) with classification rate 94%. Training and testing the fuzzy min-max classifier network with the same data sets we were able to achieve a success rate of 90.5% using parameter value $\theta = 0.082$ (60 cells).

It must be noted that in all the experiments the choice of the value of the parameter θ was not very critical with respect to the success rate as was the case with the fuzzy min-max neural network. There were intervals of θ values where the rate remained the same and only the number of the hyperplanes and the convex polygons being created

were different. Besides, while the value of θ was increasing the network structure (hypeplanes and cells) was reduced, and so we were choosing the maximum θ value of such intervals so as to achieve the least network architecture with the best overall success rate.

5 TREATING DISCRETE ATTRIBUTES

The model of fuzzy neural network based of Dirichlet Tesselations considers as basic assumption that all attributes take continuous values. Thus, we are able to map the pattern space corresponding to each class to a number of regions (convex polygons) by creating perpedincular bisectors (hyperplanes) between sites of different classes. Nevertheless, when the data set consists of both continuous and discrete attributes we cannot treat the discrete features in the same way, and so it is necessary to find another mode of operation.

Suppose that $\mathcal{D}$, $n_D = |\mathcal{D}|$ and $\mathcal{C}$, $n_C = |\mathcal{C}|$ denote the set and the number of the discrete and the continuous attributes respectively. Let also D^j be the domain of each discrete attribute $j \in \mathcal{D}$. A n-dimensional pattern $a = (a_1, a_2, \ldots, a_n)$ having both types of attributes, consists of continuous features a_j for $j \in \mathcal{C}$ and discrete $a_j \in D^j$ for $j \in \mathcal{D}$. Each polygon i is described by providing the proper hyperplanes with respect to the continuous attributes and moreover a set of attribute values $D_{ij} \subseteq D^j$ for discrete attributes $j \in \mathcal{D}$. It is obvious that the sets D_{ij} must be crisp, i.e., an element either belongs to a set (membership value is 1) or not (membership value is 0). Including the above analysis to the computation of the membership function of a pattern a to a polygon i, equation (1) takes the following form:

$$b_i(a) = \frac{1}{2|H_i|} \sum_{h \in H_i} v_{ih} m_h(a^*) + \frac{1}{2} + \frac{1}{n_D} \sum_{j \in \mathcal{D}} m_{D_{ij}}(a_j) \qquad (4)$$

where a^* denotes the subvector of a containing only continuous attributes and $m_S(x)$ is the membership function corresponding to the crisp set S. It must be noted that if a new input pattern a_k is contained in a cell i of the same class, i.e., no creation of new cell takes place, the crisp sets D_{ij} are adjusted as follows: $D_{ij}^{new} = D_{ij}^{old} \cup a_{kj}$.

6 CONCLUSIONS

We have introduced a new model of fuzzy neural network classifier by representing fuzzy sets through a suitable partition of the solution space into a number of convex regions following the principle of Dirichlet tesselations. This type of network has the advantage of fast one-shot training and is very efficient for hard pattern classification problems as indicated by the experiments. Further research is focused on the introduction of a learning component for adaptively determining good parameter values.

References

[1] F. Aurenhammer, "Voronoi Diagrams — A Survey of a Fundamental Geometric Data Structure," ACM Computing Surveys, Vol. 23, No. 3, pp. 345–405, Sept. 1991.

[2] N.K. Bose and A.K. Garga, "Neural Network Design Using Voronoi Diagrams," IEEE Trans. on Neural Networks, Vol. 4, No. 5, pp. 778–787, Sept. 1993.

[3] A. Bowyer, "Computing Dirichlet Tesselations," The Computer Journal, Vol. 24, No. 2, pp. 168–173, 1978.

[4] P.J. Green and R. Sibson, "Computing Dirichlet Tesselations in the Plane," The Computer Journal, Vol. 21, No. 2, pp. 168–173, 1978.

[5] J. Keller and D. Hunt, "Incorporating Fuzzy Membership Functions into the Perceptron Algorithm," IEEE Trans. on Patt. Anal. and Mach. Intell., Vol. 7, pp. 693–699, 1985.

[6] K. Koutroumbas and N. Kalouptsidis, "Nearest Neighbor Pattern Classification Neural Networks," Proc. World Congress on Computational Intelligence, pp. 2911–2915, Orlando, Florida, July 1994.

[7] A. Likas, K. Blekas and A. Stafylopatis, "Application of the Fuzzy Min-max Neural Network Classifier to Problems with Continuous and Discrete Attributes," Proc. NNSP 94, pp. 163–170, Ermioni, Greece, Sept. 1994.

[8] O.J. Murphy, "Nearest Neighbor Pattern Classification Perceptrons," Proc. IEEE, Vol. 78, No. 10, pp. 1595–1598, Oct. 1990.

[9] P. K. Simpson, "Fuzzy Min-Max Neural Networks-Part 1: Classification," IEEE Trans. on Neural Networks, Vol. 3, No. 5, pp. 776–786, Sept. 1992.

Dynamics of Associative Memory with a Self-consistent Noise

Ioan Opriş

Department of Physics, University of Bucharest,
Bucharest-Măgurele, Romania

Abstract

The Glauber dynamics of magnetic systems has been extended to the case of neural networks with a general odd response function. We have derived a set of recursion relations for the overlap parameter, noise average and noise variance taken as macrovariables of the process describing the dynamics of associative memory. The retrieval process has been studied then for a hyperbolic tangent transfer function by the self-consistent signal to noise ratio method. It has been taken into account the fatigue effect of the real neuron. The phase diagrams of the retrieval process reveals an enhanced storage capacity for a certain set of parameter values.

1 Introduction

The neural network models of associative memory are dynamical systems with associated attractors to the cognitive events. A very well known example is the Hopfield model [1,2] successfully carried out by Amit et al. [3] with the equilibrium statistical mechanics tools. The dynamics of neural network with general response function is much more difficult to treat than equilibrium properties because there is no general framework corresponding to the Boltzmann-Gibbs equilibrium theory. Even the stochastic master equation of Glauber dynamics have been considered only for monotonic transfer function of hyperbolic tangent type [7,8].

0-7803-2739-X/95 $4.00 © 1995 IEEE

Despite these difficulties the approximate treating of retrieval process performed by Amari and Maginu [4] gave satisfaction for various transfer functions [13-17], the only macrovariables used being the overlap of the current state onto an embedded pattern and the variance.

The aim of this paper is to develope a scheme to treat the dynamics of associative memories with a general odd transfer function including non-monotonic cases. In order to carry out this task we have employed the method of Horn and Usher [10] by using a discrete time master equation describing the time evolution of the network state. The macrovariables of the process are: the overlap, the variance given through the signal to noise ratio analysis [4] and the noise average. We here deal with a self-consistent extraction process of signal from noise which finally yields to an enhanced storage capacity. This mechanism of enhancing the storage capacity is different from those involving the pseudo-inverse method [5] or the partial reversed method [12].

The paper is organized as follows. In Sec.2 we develop the general framework of the associative memory with a general transfer function [16] and successively we derive the generalized macrovariables recursion relations together with the time dependent probability and the discrete master equation. In Sec.3 we study the retrieval process of an associative memory having a simple hyperbolic tangent output function by following a self-consistent signal to noise ratio method. The conclusions are discussed in Sec. 4.

2 Associative Memory Dynamics

The dynamics of the neural network describes the change of variables in time. Let us consider a neural network of N two-state neurons $S_i = \pm 1; (i = 1, ..., N)$ which interact through the couplings J_{ij} given by the Hebb rule $J_{ij} = \frac{1}{N}\sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu$. The input-output function f sets the relationship between the neuron's new state $S_i(t+1)$ and the previous

network state $\{S(t)\}$

$$S_i(t+1) = f\left(\sum_{j=1}^{N} J_{ij} S_j(t)\right), \qquad (J_{ii}=0). \tag{1}$$

Morita et al. [11,12] have introduced the odd nonmonotonic function $f(h)$ which can be written as a product between the ordinary sigmoid output function and a function $g_{inh}(h,a)$ as follows

$$f(h) = \tanh(\beta h) \cdot g_{inh}(h,a), \tag{2}$$

the output inverting function being

$$g_{inh}(h,a) = \tanh[-\gamma(\mid h \mid -a)/2] \tag{3}$$

with γ a positive constant and a the threshold which makes a transfer function nonmonotone. In the limit $\beta \to \infty$, $\tanh \beta h \to$ sign(h) and for $\gamma \to \infty$, $g_{inh}(h,a)$ resembles a truncated "mexican hat" shape playing the role of inhibition for an input value greater than a.

In order to implement any general input-output relation into the present discrete-time discrete-state network including the nonmonotonic ones, it is thus natural to introduce the following process

$$\mathrm{Prob}(S_i(t+1)) = \frac{1}{2}[1 + S_i(t+1) f(h_i(t))], \tag{4}$$

with a general function f. The absolute value of f should not exceed 1 because otherwise probabilistic interpretation (4) does not make sense.

The postsynaptic potential (PSP) or the local field $h_i(t)$ of the $i-$th neuron can be expressed as a weighted sum of neuron's states $h_i(t) = \sum_j J_{ij} S_i(t) = \sum_\mu \xi_i^\mu m^\mu(t) = \xi_i^1 m^1(t) + N_i(t)$, where the overlap $m^\mu(t)$ with the $\mu-$th pattern is defined by $m^\mu(t) = \frac{1}{N}\sum_i \xi_i^\mu S_i(t)$. Applying the signal to noise ratio method we can split the PSP into two terms, the pure signal and the noise: $h_i(t) = \xi_i^1 m^1(t) + N_i(t)$ with the noise term defined by $N_i(t) = \frac{1}{N}\sum_j \sum_{\mu \neq 1} \xi_i^\mu \xi_j^\mu S_j(t)$.

We further self-consistently [9] decompose the noise part into a pure noise component $N_i^0(t)$ and an output proportional term $\alpha S_i(t)$, the decomposition relation being $N_i(t) =$

$N_i^0(t) + \alpha S_i(t)$, with α the memory loading rate. In order to include the fatigue effect given by the threshold contribution [14] one takes the following recursion relation for the noise term $N_i(t+1) = \lambda N_i(t) + \alpha S_i(t+1)$, with $\lambda < 1$. Using the notation $m = m^1$, the overlap parameter becomes $m(t+1) = \frac{1}{N}\sum_i \xi_i^1 f(\xi_i^1 m(t) + N_i(t))$, through multiplication by ξ_i^1 and statistical averaging.

Because the function f is an odd function in input, the factor $\xi_i^1 = \pm 1$ can be moved into the argument for a class of input-output functions. (For example in the case of $f(x) = \tanh(x)g(x)$ we get $\xi f(x) = \tanh(\xi x)g(x)$, $g(x)$ being an even function.) The new expression of $m(t+1)$ reads

$$m(t+1) = \frac{1}{N}\sum_i f(m(t) + \xi_i^1 N_i(t)). \tag{5}$$

Let us denote the value $\xi_i^1 N_i(t)$ at location i by z and assume this value distributed with the probability $P(z,t)$ given by [10]

$$P(z,t) = \frac{1}{N}\sum_i \delta(\xi_i^1 N_i(t) - z); \tag{6}$$

this probability allow us to write $m(t+1)$ as an integral equation

$$m(t+1) = \int dz P(z,t) f(m(t) + z). \tag{7}$$

The value of z at location i changes in one iteration to $\lambda z \pm 1$ with the probabilities

$$\begin{aligned} \pi^+(m(t), z) &= \frac{1}{2}[1 + f(m(t) + z)], \\ \pi^-(m(t), z) &= \frac{1}{2}[1 - f(m(t) + z)], \end{aligned} \tag{8}$$

extended to the general odd input-output function. These relations give us the discrete master equation as a recursion relation

$$\begin{aligned} P(z, t+1) &= \frac{1}{\lambda}[\pi^+(m(t), \frac{z-\alpha}{\lambda})P(\frac{z-\alpha}{\lambda}, t) \\ &+ \pi^-(m(t), \frac{z+\alpha}{\lambda})P(\frac{z+\alpha}{\lambda}, t)]. \end{aligned} \tag{9}$$

one can observe the convergence of solutions to fixed-point attractor for $m > m_c$, m_c being the critical initial overlap which gives the boundary of the basin of attraction. For $m < m_c$ the retrieval process fails. The other initial conditions are the same as in Fig.1 ($\zeta = \sigma = 0.1$).

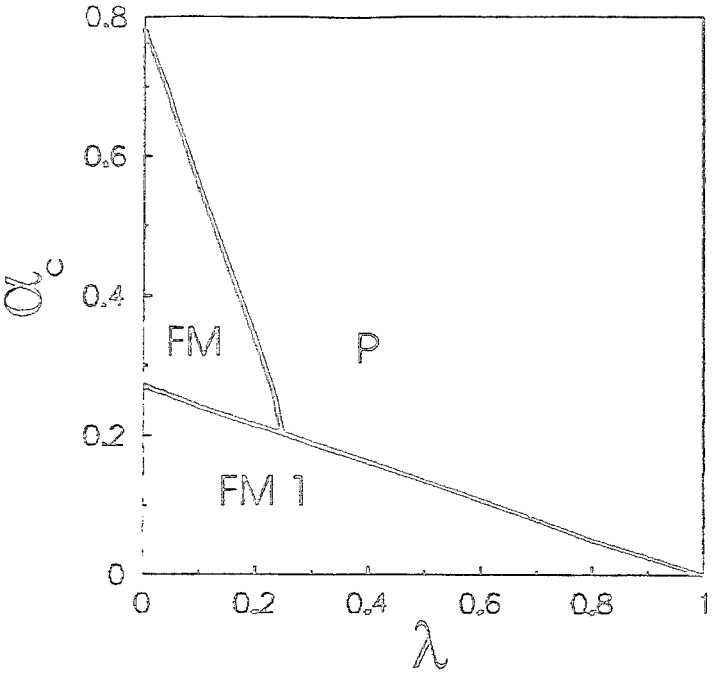


Figure 1. Retrieval process phase boundary showing the critical storage capacity versus fatigue parameter λ. The parameters are $T = 0.05$ with the same initial conditions for the $m-\zeta-\sigma$ set. At $\lambda = 0$ the storage capacity is $\alpha_c = 0.793$. The pure FM solutions located under the lower curve, denoted by FM1 have the final overlap $m = 1$ and a first order phase transition to the others FM solutions having final overlap $m < 1$. The upper line gives a second order phase transition between FM solutions and Paramagnetic ($m = 0$) P phase.

Taking the solutions of the system (11) for $T = 0.05$ with the same initial conditions for the $m-\zeta-\sigma$ set, we have plotted in figure 1 the retrieval process phase boundary showing

Since the most important features of the probability distribution are the average ζ and standard deviation, let us replace $P(z,t)$ by an expression which contains these values and can be easily manipulated,

$$P(z,t) = \frac{1}{2}\delta(z - \zeta(t) - \sigma(t)) + \frac{1}{2}\delta(z - \zeta(t) + \sigma(t)). \quad (10)$$

Replacing the master equation by its first two moments, i.e., expectation values of z and z^2 we are led to the following set of recursion relations

$$\begin{aligned} m(t+1) &= \frac{1}{2}f(m(t) - \zeta(t) - \sigma(t)) + \frac{1}{2}f(m(t) - \zeta(t) + \sigma(t)), \\ \zeta(t+1) &= \lambda\zeta(t) + \alpha m(t+1), \\ \sigma^2(t+1) &= \lambda^2\sigma(t)^2 + \lambda\sigma(t)\left[f(m(t) - \zeta(t) - \sigma(t))\right] + \\ &\quad \lambda\sigma(t)\left[f(m(t) - \zeta(t) + \sigma(t))\right] + 1 - m^2(t+1). \quad (11) \end{aligned}$$

These expressions are different from those obtained in the Amari-Maginu framework [4].

3 Analysis of Retrieval Process

The analysis of the retrieval process is carried out for a hyperbolic tangent transfer function because the mechanism responsible for the enhancement of storage capacity is not caused by the nonmonotonic function as it was expected. Here we deal with a self-consistent extraction of signal from noise in a recurrent maner. The retrieval process exhibits for small values of ζ more or less the same behavior as was obtained by Amari and Maginu [4]. The difference consists in the fact that our model is one biologically motivated by the fatigue effect and by the dynamical threshold incorporated in the noise recursion relation.

A more convincing argument that our model works as an associative memory would be the attraction basin of a memory state [16]. By plotting the time development of the overlap parameter for $T = 0.015$, $\lambda = 0.1$, $\alpha = 0.5$ and initial overlap values between $m = 0$ and $m = 1$, with increasing ratio 0.05

the critical storage capacity α_c versus fatigue parameter λ. The diagram gives a clear cut separation between the paramagnetic phase in which $m \to 0$ and the ferromagnetic one bellow the separation curve corresponding to successful retrieval. The ability of the network to recall an enhanced number of patterns is obtained when the fatigue vanishes $\lambda = 0$, the storage capacity being $\alpha_c = 0.793$. Increasing λ the storage capacity α_c decreases to zero in the $\lambda \to 1$ limit. The pure FM solutions are located under the lower curve and have the final overlap $m = 1$. The pure FM phase denoted by FM1 is separated through a first order phase transition line by the others FM solutions having the final overlap $m < 1$. The upper boundary line corresponds to a second order phase transition between FM solutions and paramagnetic P phase.

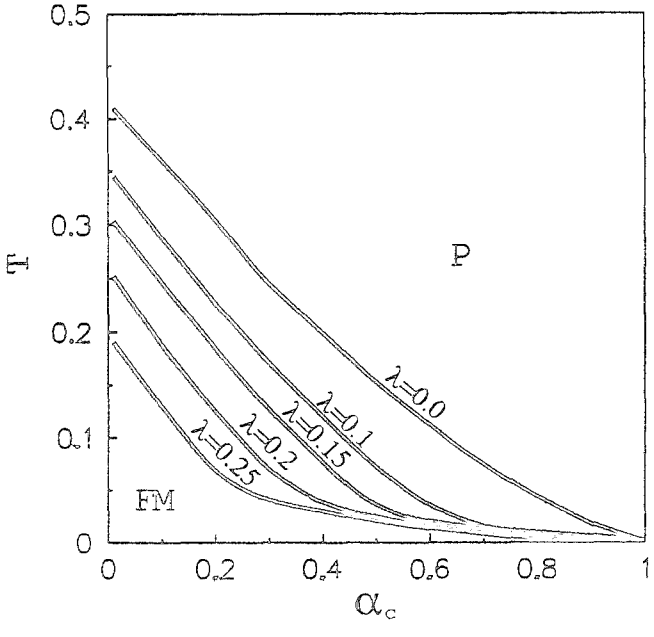


Figure 2 The phase diagram $T = f(\alpha_c)$ of the retrieval process showing a family of curves for $\lambda = 0, 0.1, 0.15, 0.2$ and 0.25.

In figure 2 was plotted the phase diagram $T = f(\alpha_c)$ of the retrieval process showing a family of curves for $\lambda = 0, 0.1, 0.15, 0.2$ and 0.25. Each curve gives the separation boundary between FM phase and the paramagnetic one. Increasing the λ parameter to 1 the boundary approaches to 0, the area of FM phase in $\alpha_c - T$ coordinates is practically vanishing. Thus, the fatigue causes the reducing of associative memory performances.

4 Conclusion

In this paper we have extended the Glauber dynamics from magnetic systems to the case of neural networks [6] with general odd response functions [12-14]. The set of recursion relations of Horn and Usher [10] was extended to the macroscopic variables describing the dynamics of associative memory retrieval process in the self-consistent signal to noise ratio framework. We have solved the equations (11) for a hyperbolic tangent transfer function and one have plotted the phase diagrams showing the boundary between the FM and paramagnetic phases. Our phase diagrams of the retrieval process reveals an enhanced storage capacity of $\alpha \to 1$ when temperature $T \to 0$ and the fatigue vanishes. Finally, a continuous time evolution set of overlap equations for nonmonotone neurons were analytically derived. The biological relevance of nonmonotonic firing rate was pointed out by Horikawa [18].

References

[1] Hopfield J.J. 1982 Proc. Natl. Acad. Sci. U.S.A. 79, 2554

[2] Hopfield J.J. 1982 Proc. Natl. Acad. Sci. U.S.A. 81, 3088

[3] Amit D.J. Gutfreund H.and Sompolinsky H 1985 Phys. Rev.A. 32, 1007

[4] Amari S. and Maginu K. 1988 Neural Networks 1, 63

[5] Kohonen T. 1974 IEEE Transactions on Computers C-23, 59

[6] van Hemmen J. L. and Kühn R. 1986 Phys. Rev. Lett. 57, 913

[7] Shiino M. 1990 J. Stat. Phys. Vol.59, Nos.3/4, 1051

[8] Shiino M., Nishimori H. and Ono M. 1989 J. Phys. Soc. Japan 58, 3, 763

[9] Shiino M. and Fukai T. 1993 private communication

[10] Horn D. and Usher M. 1989 Phys. Rev. A, 40, 2, 1036

[11] Yoshizawa S. Morita M. and Amari S. 1993 Neural Networks 6, 167

[12] Morita M. 1993 Neural Networks 6, 115

[13] Nishimori H. and Ozeki T. 1993 J. Phys. A 22, 859

[14] Nishimori H. and Opriş I. 1993 Neural Networks 6, 1061

[15] Nishimori H. and Opriş I. 1993 Proc. IEEE ICNN, San Francisco, 353

[16] Opriş I. 1995 Phys. Rev. E 51, 3, in press

[17] Nishimori H. Ozeki T. and Opriş I. 1993 Computer Aided Inovation of New Materials II, North-Holland, 383-388

[18] Horikawa Y. 1993 Proc. IEEE ICNN, San Francisco, 473

RECURSIVE NONLINEAR IDENTIFICATION USING MULTIPLE MODEL ALGORITHM

V.Kadirkamanathan
Department of Automatic Control & Systems Engineering
University of Sheffield
Mappin Street, Sheffield S1 4DU, UK
visakan@acse.sheffield.ac.uk

Abstract. In this paper, the *multiple model algorithm* is used in deriving recursive algorithms for the identification of nonlinear systems. The radial basis function (RBF) networks with only linear weights requiring estimation combined with the Kalman filter algorithm forms the essence of the identification algorithm. Multiple networks are used to identify the multi-modes of the system under a Markovian assumption, the model estimation and selection being carried out on-line. Both, 'hard' and 'soft' competition based estimation schemes are developed where in the former, the most probable network is adapted by the Kalman filter and in the latter all networks are adapted by appropriate weighting of the observation.

1 INTRODUCTION

The problem of learning multiple modes in a complex nonlinear system is increasingly being studied by various researchers [4, 10, 5, 2]. The use of a mixture of local experts to model various modes of a system has been developed and applied to learning control by Jacobs and Jordan [4, 5], and a conditional mixture density approach is adopted by Bishop [2]. The development has centred around the problem of model identification from a given set of block data, the model likelihood dependent on the input to the networks. A recursive algorithm for this static case would mimic the iterative procedure required in the block estimation schemes, the recursion being an approximation [5].

In this paper, we consider dynamic systems – developing a recursive algorithm is difficult for the reason that the mode transitions have to be detected on-line whereas in the block estimation scheme, search procedures allow detection of the optimal transition point. However, unlike in the other modular network schemes, the algorithm developed here does not use the mixing coefficients or data conditioned prior model probability. The modelling of the multiple modes in a nonlinear system is carried out by radial basis function (RBF)

0-7803-2739-X/95 $4.00 © 1995 IEEE

networks. The RBF networks were chosen for their property that by carefully selecting the centres and widths, the network estimation problem can be reduced to estimating only the linear coefficients. The recursive algorithm is based on the Kalman filter algorithm, used with RBF networks previously [6, 7, 9]. A growing network as in [7] can alternatively be considered in place of the fixed size RBF network scheme developed here.

The identification algorithm is based on the *multiple model algorithm* [1] as an on-line model selection scheme in addition to the on-line model estimation being carried out by the Kalman filter algorithm [3]. The multiple model algorithm has been commonly used in estimating the states in target tracking where the underlying model is non-stationary and hence the underlying process parameters are unknown. In unimodal recursive nonlinear identification with neural networks, the uncertainties lie in selecting the optimal size of the network, set of basis functions and the Kalman filter parameters *a priori*. The application of the multiple model algorithm in this case results in multiple RBF networks adapting to all of the observations with predictions being made only by the most probable network. For multi-modal systems, each of the multiple RBF networks have to identify the individual modes.

2 RADIAL BASIS FUNCTIONS

Radial basis functions (RBF) were introduced as a technique for multivariable interpolation [12], which can be cast into a neural network architecture [12]. Their functional form is given by,

$$f(\mathrm{x};\mathrm{p}) = \sum_{k=1}^{K} w_k g_k(\mathrm{x}) = \mathrm{w}^T \mathrm{g} \tag{1}$$

where $\mathrm{w} = [\ldots, w_k, \ldots]^T \in \Re^K$ is the linear weight vector and $\mathrm{g} = [\ldots, g_k(\mathrm{x}), \ldots]^T \in \Re_+^K$ are the radial basis functions, where,

$$g_k(\mathrm{x}) = \psi\left(\|\mathrm{x} - \mathrm{m}_k\|\right) \tag{2}$$

and $\|\cdot\|$ is the L_2-norm or the Euclidean distance, $\mathrm{m}_k \in \Re^M$ are the RBF centres or means, $\psi(\cdot) : \Re_+ \mapsto \Re$ is a scalar function. The L_2-norm is often weighted as,

$$\|\mathrm{x} - \mathrm{m}_k\|^2 = (\mathrm{x} - \mathrm{m}_k)^T \mathrm{C}_k^{-1} (\mathrm{x} - \mathrm{m}_k) \tag{3}$$

where C_k^{-1} is a positive definite weighting matrix of the kth basis function which can transform the equidistant lines from being hyperspherical to the hyperellipsoidal. $\mathrm{C}_k = r_k^2 \mathrm{I} \in \Re^{M \times M}$ in this paper, where $r_k \in \Re_+$ is referred to as the width parameter. One of the commonly used RBF networks is the *Gaussian radial basis function (GRBF) network*, also called the localised receptive field network [12]. The basis functions for this network are given by,

$$g_k(\mathrm{x}) = \exp\left\{-\frac{\|\mathrm{x} - \mathrm{m}_k)\|^2}{2r_k^2}\right\} \tag{4}$$

Estimating the parameters w_k, m_k, r_k, for $k = 1, \ldots, K$, based on the observations is a nonlinear optimisation problem. Often, some of the parameters

can be chosen from *a priori* knowledge and estimation is restricted to a subset of those outlined above. In functional interpolation, the number of basis functions K is taken to be equal to the number of observations N and $\mathbf{m}_k = \mathbf{x}_n$, for $n, k = 1, \ldots, N$. The width parameter r_k is also selected *a priori* leaving only the estimation of linear coefficients. Also, the number of basis functions can be chosen to be independent of the number of observations, such as in on-line estimation, and their parameters pre-selected. Typically, the centre parameters are chosen randomly to lie within some bounds on the input space and the width parameter r_k based on the centre nearest neighbour distance [12]. Alternatively, for on-line estimation, the RBF centre parameters can be assigned on-line to be a subset of the input observations with growing network schemes [6, 7, 9]. The RBF networks are used for their property that having chosen appropriate RBF centre and width parameters $\mathbf{m}_k$, r_k, only the linear weights $\mathbf{w}$ need to be estimated for which fast, efficient and optimal algorithms exist.

3 RECURSIVE IDENTIFICATION

Let the set of input – output observations from which the identification is to be made, be denoted as,

$$\mathcal{Z}_N = \{z_n \mid n = 1, \ldots, N\} \tag{5}$$

where, $\mathcal{Z}_N$ includes all observations upto the Nth sample and z_n describes only the nth input – output observation,

$$z_n = \{(\mathbf{x}_n, y_n) \mid \mathbf{x}_n \in \Re^M, y_n \in \Re\} \tag{6}$$

Let the underlying process generating the input – output observations in a nonlinear system be given by,

$$y = f^*(\mathbf{x}) + \eta \tag{7}$$

where η is the noise with unknown distribution and $f^*(\cdot) : \Re^M \mapsto \Re$ is the unknown underlying nonlinear function that needs to be learned or estimated. For the system described by (7), under the assumption that the noise η is zero mean Gaussian and under the assumption that the chosen model can approximate the underlying function arbitrarily closely, the probability distribution $p(z_n|\mathbf{p}, \mathcal{M})$ is Gaussian as well, *ie.*,

$$p(z_n|\mathbf{p}, \mathcal{M}) = \frac{\exp\left\{-\frac{1}{2}R_0^{-1}\,|y_n - f(\mathbf{x}_n; \mathbf{p})|^2\right\}}{(2\pi)^{\frac{1}{2}} R_0^{\frac{1}{2}}} \tag{8}$$

This is the *likelihood* of the observation z_n for the chosen model $\mathcal{M}$, which in our case is the GRBF network, and model parameters $\mathbf{p}$. The vector $\mathbf{g}_n = [g_1(\mathbf{x}_n), \ldots, g_K(\mathbf{x}_n)]^T$ and R_0 is the variance of the noise η. A further assumption made is that the observations are independent and identically distributed so that the likelihood of the observation set is,

$$p(\mathcal{Z}_N|\mathbf{p}, \mathcal{M}) = \prod_{n=1}^{N} p(z_n|\mathbf{p}, \mathcal{M}) \tag{9}$$

Off-line identification schemes estimate the optimal model parameters by maximising this likelihood, or equivalently, minimising the log-likelihood which in turn becomes equivalent to *least squares* identification.

Considering the RBF network with pre-selected RBF parameters m_k, r_k, for $k = 1, \ldots, K$, the parameters that need to be estimated are the linear weights w. If we assign the *prior* probability distribution for the model parameters $p(\mathrm{w}|\mathcal{M})$ to be Gaussian with mean w_0 and covariance matrix (positive definite) $\mathbb{P}_0 \in \Re^{K \times K}$, Bayes law,

$$p(\mathrm{w}|\mathcal{Z}_N, \mathcal{M}) = \frac{p(\mathcal{Z}_N|\mathrm{w}, \mathcal{M})p(\mathrm{w}|\mathcal{M})}{p(\mathcal{Z}_N|\mathcal{M})} \tag{10}$$

combines the likelihood and the prior to give the *posterior* probability distribution $p(\mathrm{w}|\mathcal{Z}_N, \mathcal{M})$ for the parameters which is also Gaussian and given by,

$$p(\mathrm{w}|\mathcal{Z}_N, \mathcal{M}) = \frac{\exp\left\{-\frac{1}{2}(\mathrm{w} - \mathrm{w}_N)^T \mathbb{P}_N^{-1}(\mathrm{w} - \mathrm{w}_N)\right\}}{(2\pi)^{\frac{K}{2}} |\mathbb{P}_N|^{\frac{1}{2}}} \tag{11}$$

The term $p(\mathcal{Z}_N|\mathcal{M})$ is known as the *evidence* for the model $\mathcal{M}$ in the observations $\mathcal{Z}_N$. The final estimate for the weights are w_N.

For on-line or recursive estimation, the Bayes law relation becomes,

$$p(\mathrm{w}|\mathcal{Z}_n, \mathcal{M}) = \frac{p(z_n|\mathrm{w}, \mathcal{M}, \mathcal{Z}_{n-1})p(\mathrm{w}|\mathcal{Z}_{n-1}, \mathcal{M})}{p(z_n|\mathcal{Z}_{n-1}, \mathcal{M})} \tag{12}$$

and the above equation is applied recursively for $n = 1, \ldots, N$. Under Gaussian assumptions as outlined above, the Bayesian approach leads to the Kalman filter algorithm as the on-line optimal estimator [3] for the model parameters w. The Kalman filter equations are the following:

$$e_n = y_n - \mathrm{w}_{n-1}^T \mathrm{g}_n \tag{13}$$

$$R_n = R_0 + \mathrm{g}_n^T \mathbb{P}_{n-1} \mathrm{g}_n \tag{14}$$

$$\mathrm{k}_n = R_n^{-1} \mathbb{P}_{n-1} \mathrm{g}_n \tag{15}$$

$$\mathrm{w}_n = \mathrm{w}_{n-1} + e_n \mathrm{k}_n \tag{16}$$

$$\mathbb{P}_n = \mathbb{P}_{n-1} - \mathbb{P}_{n-1} \mathrm{g}_n R_n^{-1} \mathrm{g}_n^T \mathbb{P}_{n-1} \tag{17}$$

e_n is the on-line *prediction error* based on which the correction to w is made. k_n is known as the *Kalman gain*. R_n is known as the *innovation variance* (see (18)). In this formulation, the Kalman filter estimation of the model parameters is equivalent to the recursive least squares form. In order to ensure continued adaptation a *random walk model* is introduced whereby the term $Q_0 \mathbb{I}$ is added to (17), as in [6], [7].

Using the Bayesian derivation the evidence term, is given by,

$$p(z_n|\mathcal{M}, \mathcal{Z}_{n-1}) = \frac{\exp\left\{-\frac{1}{2} R_n^{-1} |e_n|^2\right\}}{(2\pi)^{-\frac{M}{2}} |R_n|^{-\frac{1}{2}}} \tag{18}$$

The above equation shows that the evidence term used in Bayesian model selection [11] is computed recursively, but for the different priors R_0, $\mathbb{P}_0$ sampled by the multiple models. This is also the likelihood of the n^{th} observation given the model $\mathcal{M}$ and the past observations $\mathcal{Z}_{n-1}$.

4 RECURSIVE MODEL SELECTION

The use of a single neural network for identification is based on the assumption that the chosen network is appropriate to approximate the underlying function and that the noise assumptions are valid with the relevant parameters, such as noise variance being known. In cases where the above fail to hold true, the use of multiple models provides an improvement at the expense of increased computations. Multiple Kalman filters with the *multiple model algorithm* [1] being used for estimating the states in target tracking where the underlying model for the target is linear but nonstationary, is such an example.

In the case of neural networks, the use of multiple networks can overcome the limitation of having to pre-select the number of basis functions and their parameters, by using several networks with varying sizes and parameters. It allows for the selection of appropriate basis functions on-line to a limited extent, similar in spirit to the off-line approach based on the Bayesian criteria adopted in [8]. Furthermore, different choices for the unknown noise variance R_0, the prior weight parameter covariance $\mathbf{P}_0$ and the random walk model parameter Q_0 can be made allowing a wider search for the underlying model. The multiple model algorithm can be viewed as an on-line model selection scheme using Bayesian statistics.

Let the total number of neural networks or models used be H. Applying Bayes law gives the following relation:

$$p(\mathcal{M}_h|\mathcal{Z}_n) = \frac{p(z_n|\mathcal{M}_h, \mathcal{Z}_{n-1})p(\mathcal{M}_h|\mathcal{Z}_{n-1})}{p(z_n|\mathcal{Z}_{n-1})} \tag{19}$$

which can be computed recursively for $n = 1, \ldots, N$. $p(z_n|\mathcal{M}_h, \mathcal{Z}_{n-1})$ is the likelihood given in (18) and $p(\mathcal{M}_h|\mathcal{Z}_n)$ is the posterior probability of model $\mathcal{M}_h$ being the true underlying model amongst the chosen H models, given the observations $\mathcal{Z}_n$. The term $p(z_n|\mathcal{Z}_{n-1})$ is the normalising term given by,

$$p(z_n|\mathcal{Z}_{n-1}) = \sum_{h=1}^{H} p(z_n|\mathcal{M}_h, \mathcal{Z}_{n-1})p(\mathcal{M}_h|\mathcal{Z}_{n-1}) \tag{20}$$

Since the quantities e_n and R_n necessary for the computation of the term $p(z_n|\mathcal{M}_h, \mathcal{Z}_{n-1})$ are obtained from the Kalman filter estimator, all the terms on the left side of (19) are known once initial prior probabilities for models are assigned, for example as,

$$p(\mathcal{M}_h|\mathcal{Z}_0) = p(\mathcal{M}_h) = \frac{1}{H} \tag{21}$$

The above algorithm (18), (19) combined with the Kalman filter estimation equations is known as the multiple model algorithm [1]. Amongst all the networks that are attempting to identify the underlying system, the identified model is the one with the highest posterior probability $p(\mathcal{M}_h|\mathcal{Z}_n)$ at each time n, and hence can vary from time to time, and predictions are based on this most probable model.

5 HARD AND SOFT COMPETITION

One form of nonstationarity commonly found in systems is multi-modality. For example, the speech signal can be viewed as a concatenation of signals from quasi-stationary modes or states such as phonemes undergoing transitions at various time instances. The multi-modal system is represented as,

$$y_n = \begin{cases} f_1^*(\bar{x}_n) + \eta & \text{if } n \in \{[n_{1i}^1, n_{1f}^1], [n_{1i}^2, n_{1f}^2], ..\} \\ f_2^*(\bar{x}_n) + \eta & \text{if } n \in \{[n_{2i}^1, n_{2f}^1], [n_{2i}^2, n_{2f}^2], ..\} \\ \cdots & \\ f_l^*(\bar{x}_n) + \eta & \text{if } n \in \{[n_{li}^1, n_{lf}^1], [n_{li}^2, n_{lf}^2], ..\} \end{cases} \tag{22}$$

where $n \in [n_{li}^1, n_{lf}^1]$ is the first time-interval during which the lth mode is determining the behaviour of the system (i stands for initial f for final). To be general, the same mode can become active again in the future during a second time interval $[n_{li}^2, n_{lf}^2]$ and none of the time intervals overlap. Under this description of the multi-modal system, the task of identification is to estimate or approximate the underlying functions $f_1^*(\cdot)$, $f_2^*(\cdot)$, ..., $f_l^*\cdot)$. This is made difficult by the need for each model to represent each mode or state individually and hence the model parameters have to be estimated on observations pertaining to that mode. This can easily be done if the mode transitions are known *a priori*, which in general is not the case. Hence, detection of mode transitions must also be made along with the model estimation. In block estimation or off-line learning, Levin [10] demonstrated how an additional switching input is used to model a bi-modal system where the switching input signal is jointly estimated with model parameters iteratively. The modular network scheme developed by Jacobs and Jordan [4, 5] makes use of a gating network that chooses an expert network to model individual modes. Both, block and on-line estimation schemes are used to determine the relevant parameters in the expert network scheme, but it is derived for static systems.

To facilitate the on-line identification of multi-modal dynamic systems, a first order Markov assumption is made for the mode transitions. Given that at the time instant $n-1$ the given mode is j, it is predicted under the above assumption that the probability of the mode at time instant n being h is the transition probability P_{hj}. With H modes, $\sum_{h=1}^{H} P_{hj} = 1$.

The predicted probability of the mode being j at time n therefore is given by,

$$p_{n|n-1}(\mathcal{M}_h|\mathcal{Z}_{n-1}) = \sum_{j=1}^{H} P_{hj} p(\mathcal{M}_j|\mathcal{Z}_{n-1}) \tag{23}$$

This can be viewed as the prediction stage of the model selection algorithm. Given the observation z_n, the correction is achieved through the multiple model algorithm of (19) with the following modification:

$$p(\mathcal{M}_h|\mathcal{Z}_n) = \frac{p(z_n|\mathcal{M}_h, \mathcal{Z}_{n-1}) p_{n|n-1}(\mathcal{M}_h|\mathcal{Z}_{n-1})}{p_{n|n-1}(z_n|\mathcal{Z}_{n-1})} \tag{24}$$

where modification to the prior and evidence has been made. Since the posterior probabilities of each mode effectively indicate which mode is dominant at each time n, changes can then be used as means of detecting mode transitions.

The detection mechanism allows two possible methodologies to be used in model parameter estimation for each modes. In the first method, only the model with the highest posterior probability undergoes adaptation using the Kalman filter algorithm while all other models are prevented from adapting. This is the *'hard' competition*. In the second method, all models are allowed to undergo adaptation with appropriate weighting to reflect the modelling performances as measured by the posterior probability. The weighting scheme adopted is to modify the noise variance parameter R_0 at time n for the hth model to become,

$$R_0^{(h)}(n) = \frac{R_0}{\{p(\mathcal{M}_h|Z_{n-1})\}} \tag{25}$$

This value is used in the Kalman filter equation (14) at time n for each model h. It increases the apparent uncertainty in the measurement output according to how unlikely the model is to be the true underlying mode, by increasing the noise variance term of the Kalman filter algorithm. In fact, this is the same weighting that will be achieved in the maximum likelihood iterative estimation procedure, the weight being the posterior model probability conditioned on the data. This is a *'soft' competition*. While hard competition assumes mode transitions to be instantaneous, soft competition allows for transition to take place over a time interval.

6 EXPERIMENTAL RESULTS

The experiments used a number of Gaussian radial basis function (GRBF) networks with basis function parameters chosen randomly. The nonlinear system chosen for the investigation is the quadratic map chaotic time-series, where, the observations are generated by,

$$y_n = 4y_{n-1}(1 - y_{n-1}) \tag{26}$$

such that with $x_n = y_{n-1}$, the underlying function $f^*(\cdot)$ is quadratic.

Figure 1 shows the model posterior probabilities with increasing time and the approximation error for each network over the input range $x \in [0, 1]$ for four GRBF networks (Nets 1,2,3,4) with number of basis functions $K = 3, 5, 10, 20$. The results show that the smaller network predictions are preferred in the initial stages since their parameter estimates are less uncertain than for the larger networks. The results also show that under similar identification performance, smaller complexity model is preferred demonstrating the embodiment of *Occam's razor* [11].

The multi-modal nonlinear system chosen for the experimental demonstration is also based on the quadratic map and was used in [10]. The two modes are given by the equations,

$$\begin{aligned} y_n &= 4y_{n-1}(1 - y_{n-1}) \\ y_n &= 1 - 4y_{n-1}(1 - y_{n-1}) \end{aligned} \tag{27}$$

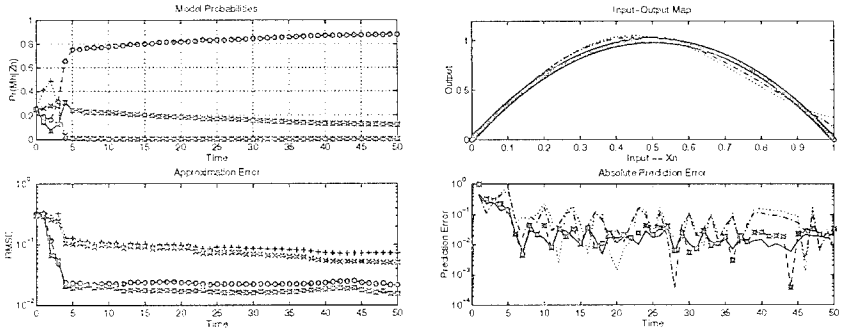


Figure 1: Uni-modal identification: (a) Model probabilities and (b) Approximation accuracy (c) Actual and approximated functions, and (d) Absolute prediction error (Net1 $+\cdots+$, Net2 $\times$–.–$\times$, Net3 $\circ$– –$\circ$, Net4 $*$—-$*$, Underlying function $\circ$, multiple network scheme predictions $*$).

and the system undergoes mode transition after every 50 samples. Two GRBF networks of size $K = 10$ were used for these experiments.

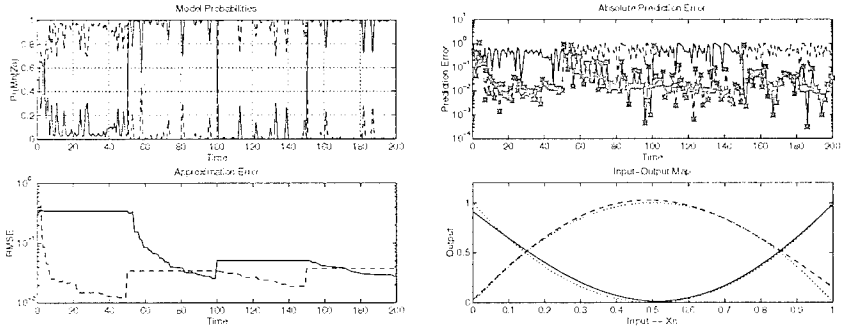


Figure 2: Multi-modal identification ('hard' competition): (a) Model probabilities and (b) Approximation accuracy. (c) Absolute prediction error and (d) Actual and approximated functions (Net1 – –, Net2 —, underlying function $\cdots$, multiple network prediction $*$).

Figure 2 shows that the mode transition is detected quickly and the appropriate hard switching takes place. The networks retain their approximation to a certain degree at the end of mode transitions and the jump is due to the first few observations of the next mode making the parameter estimates drift a little before the switching takes place. It also shows that good predictions are made from the hard competition multiple network scheme and that the two modes are identified by the RBF networks.

Figure 3 shows that the results for the soft competition case is similar to the hard competition case for the example chosen. Since the bi-modal system here undergoes instantaneous transitions, the hard competition is more appropriate. However, the soft competition allows for transition over inter-

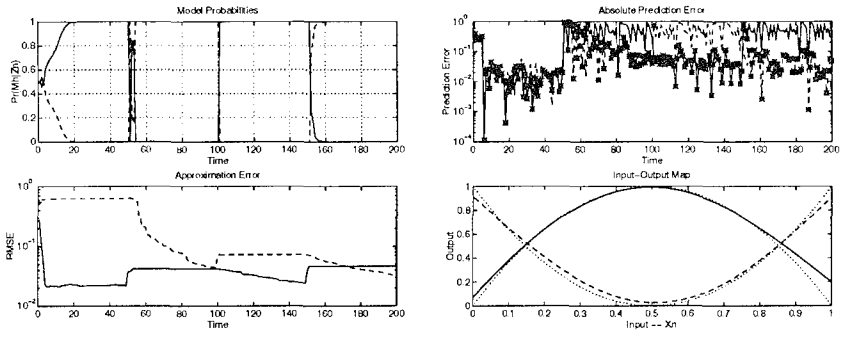


Figure 3: Multi-modal identification ('soft' competition): (a) Model probabilities and (b) Approximation accuracy (c) Absolute prediction error and (d) Actual and approximated functions (Net1 – –, Net2 —, underlying function $\cdots$, multiple network prediction $*$).

vals, and it provides good results on instantaneous mode transitions as well. The results demonstrate the successful operation of the algorithms on the bi-modal nonlinear system used in these experiments.

7 CONCLUSIONS

Recursive identification schemes for nonlinear systems based on the multiple model algorithm are developed in this paper. The neural network used is the radial basis function (RBF) network in which the parameters that need estimation are the linear weights. For uni-modal systems, the multiple model algorithm is directly applied to the multiple networks that allow different network configurations such as number of basis functions and RBF parameters. This is similar to the problem of selecting a subset of appropriate basis functions to approximate the underlying function. For multi-modal systems, a first order Markov assumption for mode transitions is made to facilitate the development of the algorithms. The transitions are detected using the posterior probability of each model representing the observations. Two methods of identification algorithms are developed, where once detection is made, estimation of model parameters is based on either 'hard' or 'soft' competition. In the former, only the mode with the highest posterior probability undergoes adaptation by the Kalman filter and in the latter all modes are adapted by appropriate weighting of the observation.

It should be noted that the mode transitions considered here cannot be predicted beforehand and is a random event. The model probabilities computed based on performance over time. This is in contrast to the modular network schemes of [4, 5, 2], where, modes or experts are identified based on the state-space by the gating network and hence the transitions could be predicted. At present, investigation into combining both approaches is being carried out.

Acknowledgements

The support of EPSRC (UK) under the grant GR/J46661 is acknowledged.

References

[1] Bar-Shalom, Y. and Fortmann, T. E. *Tracking and data association*, Academic Press, New York, 1988.

[2] Bishop, C. M. "Mixture density networks", *Report NCRG/4288*, Computer Science Dept., Aston University, UK, 1994.

[3] Candy, J. V. *Signal processing: The model based approach*, McGraw-Hill, New York, 1986.

[4] Jacobs, R. A., Jordan, M. I., Nowlan, S. J. and Hinton, G. E. "Adaptive mixtures of local experts", *Neural Computation, 3*: 79-87, 1991.

[5] Jordan, M. I. and Jacobs, R. A. "Hierarchical mixtures of experts and the EM algorithm", *Neural Computation, 6*: 181-214, 1994.

[6] Kadirkamanathan, V. *Sequential learning in artificial neural networks*, PhD Thesis, University of Cambridge, UK.

[7] Kadirkamanathan, V. "A statistical inference based growth criterion for the RBF network", In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, 12-21, 1994.

[8] Kadirkamanathan, V. "Bayesian inference for basis function selection in nonlinear system identification using genetic algorithms", In Skilling, J. and Sibisi, S., (eds.) *Maximum entropy and Bayesian methods*, Kluwer, 1995.

[9] Kadirkamanathan, V. and Niranjan, M. "A function estimation approach to sequential learning with neural networks", *Neural Computation, 5*: 954-975, 1993.

[10] Levin, E. "Hidden control neural architecture modeling of nonlinear time varying systems and its applications", *IEEE Transactions Neural Networks, 4*: 109-116, 1993.

[11] MacKay, D. J. C. "Bayesian interpolation", *Neural Computation, 4*: 415-447, 1992.

[12] Moody, J. E. and Darken, C. "Fast learning in networks of locally-tuned processing units", *Neural Computation, 1*: 281-294, 1989.

[13] Powell, M. J. D., 1987, "Radial basis functions for multivariable interpolation: A review", In Mason, J. C. and Cox, M. G., (eds.), *Algorithms for Approximation*, Oxford University Press, Oxford, 143-167, 1987.

MUTUAL INFORMATION IN A LINEAR NOISY NETWORK

Alessandro Campa, Paolo Del Giudice
Physics Laboratory, Istituto Superiore di Sanità
and INFN Sezione Sanità
Viale Regina Elena 299, 00161 Roma, Italy

Nestor Parga
Departamento de Física Teórica, Universidad Autónoma de Madrid
Ciudad Universitaria de Cantoblanco, 28049 Madrid, Spain

Jean-Pierre Nadal
Laboratoire de Physique Statistique, Ecole Normale Supérieure
24, rue Lhomond, 75231 Paris Cedex 05, France

We consider a linear, one-layer feedforward neural network performing a coding task under noisy conditions. We determine the family of synaptic couplings that maximizes the mutual information between input and output distribution. Optimization is performed under different constraints on the synaptic efficacies. We analyze the dependence of the solutions on input and output noises.

INTRODUCTION

A feedforward neural network of a given architecture provides a coding of its input data. In this work we consider a one-layer linear network, and we are interested in the network configurations (i.e., the structure of the synaptic couplings) which are able to resolve as many features as possible of the input data distribution, under noisy conditions. Finding such "optimal" codings can be useful for both the statistical applications of neural networks and the neural modeling of early sensory processing. Works concerned with several aspects of this problem can be found in [1, 2, 3].

The data, representing the environment, are generated according to some probability distribution and sent to the network as its input. The network updates its synaptic weights in an unsupervised way, according to a given rule, possibly inspired by an optimization principle. Several alternatives have been suggested. Oja [4, 5] proposed a Hebbian updat-

0-7803-2739-X/95 $4.00 © 1995 IEEE

ing modified in such a way that the couplings can not grow indefinitely. This rule produces synaptic couplings, between an input layer with N neurons and an output layer with p neurons ($p < N$), that converge to values that span the same subspace as the p principal components of the input data distribution [6]. However, the effect of noise in the network is not considered. Sanger [7] has given a different rule that converges to a solution with a similar behaviour.

An alternative method is to use optimization criteria based on information theory. For instance it has been argued [1, 8] that the network builds an efficient coding by minimizing the redundancy in the data, a criterion that tends to decorrelate the output activities. A related procedure, the infomax principle, maximizes the information that the output has about the input [2]. Several authors [9, 10, 11, 12] have considered the maximization of the mutual information in a linear channel with noise and, under some hypothesis, they exhibited a solution for the optimal couplings. These works, however, leave several points to be clarified, such as the details of the solutions and their stability, and the role played by the different possible constraints imposed on the synaptic configurations,

In this work, using notions derived from information theory, we characterize the optimal solutions for the synaptic configuration. In particular, we determine the family of synaptic couplings that maximizes the mutual information between input and output distribution. This optimization is performed under different assumptions on the allowed synaptic configurations. We study analytically in detail the dependence of the solutions on input and output noises in the case in which the input distribution is gaussian. For this case we perform a rigorous stability analysis of the solutions. A brief account of preliminary results in this direction has been given in [13], while a full account of the calculation is given in [14].

THE MODEL

On general grounds, an information channel, transforming an input (source) set of units $\vec{\xi} \equiv \{\xi_1, \ldots, \xi_N\}$ into an output set $\vec{V} \equiv \{V_1, \ldots, V_p\}$, can be characterized by the mutual information $\mathcal{I}$ given by:

$$\mathcal{I}(\vec{V}, \vec{\xi}) = \int P(\vec{V}, \vec{\xi}) \log \frac{P(\vec{V}, \vec{\xi})}{P(\vec{V})P(\vec{\xi})} \, d\vec{\xi} d\vec{V}, \tag{1}$$

where we use the same symbol P to denote the different probability distributions. For details about information theory see, e.g., [15].

We consider a situation in which the actual realization of the information channel is a neural module, as Figure 1 illustrates. The element

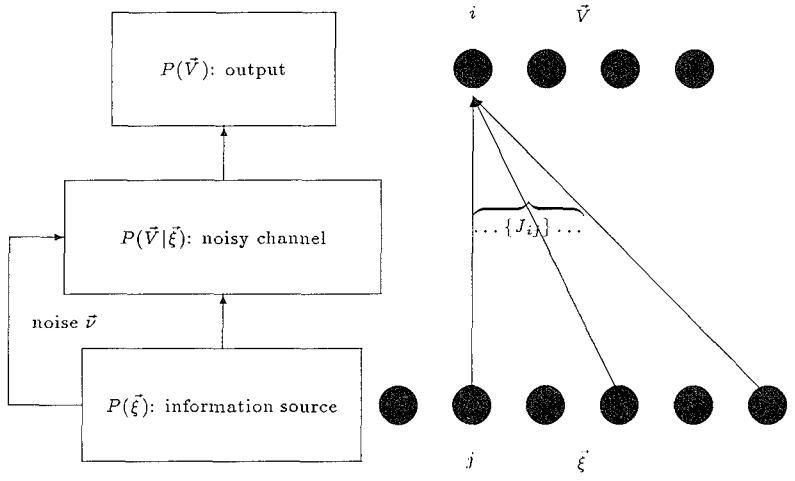


Figure 1: The neural network as information processor.

J_{ij} of the pxN matrix J connects the input unit ξ_j to the output unit V_i; for later convenience we define the N-component vectors $\vec{J}_i$, $i = 1, \ldots, p$: the elements of $\vec{J}_i$ are the connections J_{ij}, $j = 1, \ldots, N$, from all the input units to the i-th output. We consider only the case $p \leq N$.

The input and output variables, $\vec{\xi}$ and $\vec{V}$, take on continuous values, and we assume a linear transfer function for the neurons in the limit of noiseless channel. In the presence of channel noise, characterized by a parameter b, we assume that the conditional probability distribution $P(\vec{V}|\vec{\xi})$ is the gaussian given by:

$$P(\vec{V}|\vec{\xi}) = \frac{1}{(\pi b)^{p/2}} \exp\left\{ -\frac{1}{b} \sum_{i=1}^{p} \left(V_i - \sum_{j=1}^{N} J_{ij}\xi_j \right)^2 \right\}, \qquad (2)$$

that gives a linear deterministic channel for $b \to 0$. This expression has to be modified if there is also an input noise. We assume that there is an additive gaussian noise $\vec{\nu}$ in input, such that the input to the j-th input unit is $\xi_j + \nu_j$, with $\vec{\nu}$ uncorrelated with $\vec{\xi}$: $< \nu_i \xi_j >= 0$, $< \nu_i >= 0$, $< \nu_i \nu_j >= (b_0/2)\delta_{ij}$. In this case (2) is replaced by:

$$P(\vec{V}|\vec{\xi}) = \frac{1}{\sqrt{\pi^p \det[b\mathbf{1}_p + b_0 J J^T]}} \cdot \cdot \exp\left\{ -\left(\vec{V} - J\vec{\xi}\right) \cdot \left[b\mathbf{1}_p + b_0 J J^T\right]^{-1} \left(\vec{V} - J\vec{\xi}\right) \right\}, \qquad (3)$$

where we have adopted matrix notation; $\mathbf{1}_p$ is the unit matrix of dimension p, and J^T is the Nxp transpose matrix of J.

We must make assumptions about the environment; we assume that the input distribution is a gaussian, characterized by the correlation matrix $\mathcal{C}$ defined by $< \xi_i \xi_k >= (1/2)\mathcal{C}_{ik}$. Since $\mathcal{I}$ will not depend on $< \xi_i >$, we also assume for simplicity $< \xi_i >= 0$. Therefore we have:

$$P(\vec{\xi}) = \frac{1}{\sqrt{\pi^N \det \mathcal{C}}} \exp\left(-\vec{\xi} \cdot \mathcal{C}^{-1} \vec{\xi}\right) \tag{4}$$

Now the output probability distribution $P(\vec{V})$, needed for the computation of $\mathcal{I}$, can be easily computed. Finally we obtain the result for $\mathcal{I}$, which is:

$$\mathcal{I} = \frac{1}{2} \log \frac{\det[b\mathbb{1}_p + J(b_0 \mathbb{1}_N + \mathcal{C})J^T}{\det[b\mathbb{1}_p + b_0 J J^T]}. \tag{5}$$

The base of the logarithm simply determines the scale of $\mathcal{I}$; we can therefore take the natural logarithm.

We limit ourselves to a discussion of the properties of the J configurations maximizing $\mathcal{I}$, focusing in particular on the effects of both input and channel noise. We do not consider here any particular dynamics leading the Js to the maxima. Several authors (see, e.g., [3] and references therein, and [2]) have discussed a possible biological relevance of maximizing the mutual information in early sensory processing pathways.

It can be easily seen that, if $b \neq 0$, $\mathcal{I}$ grows asymptotically (to a finite value if $b_0 \neq 0$ or to infinite if $b_0 = 0$), provided the Js are allowed to grow without limit. To cope with the general case, in order to maximize $\mathcal{I}$, we need therefore to limit the growth of the Js; a possibility is to redefine the cost function of our optimization problem adding a "penalty" damping term: $\mathcal{I} \to \tilde{\mathcal{I}} = \mathcal{I} - (\rho/2)\mathtt{Tr}(JJ^T)$, where ρ is a positive parameter; this added term can be generically interpreted as a tendency of the connections J_{ij} to "forget". Another possibility is to impose a constraint on the Js that prevents their unlimited growth; we analyze the case in which a real constraint is imposed on the Js, namely a global constraint of the form $\sum_{ij} J_{ij}^2 = \sigma$, where σ is a constant. We can then have an indication on how the features of the optimal solutions that we find, depend on the particular strategy that we choose to limit the growth of the Js.

RESULTS

We will show few details about the calculations for the damped case, while, for the case of the global constraint, we will only show the differences from the first case.

For the damped case the function to be maximized is now:

$$\tilde{\mathcal{I}} = \frac{1}{2}\log\frac{\det[b\mathbf{1}_p + J(b_0\mathbf{1}_N + \mathcal{C})J^T]}{\det[b\mathbf{1}_p + b_0 JJ^T]} - \frac{1}{2}\rho\mathtt{Tr}(JJ^T). \qquad (6)$$

We note the important property that both $\mathcal{I}$ and $\tilde{\mathcal{I}}$ are invariant under orthogonal transformations $J \rightarrow \mathtt{A}J$, where $\mathtt{A}$ is any orthogonal pxp matrix. This means that the points corresponding to a given value of $\tilde{\mathcal{I}}$ cover an hypersurface in the Nxp-dimensional space of the Js, and that they are connected by orthogonal transformations. We remark that the transformations $\mathtt{A}$ are not rotations in the space of the N-dimensional vectors $\vec{J}_i$, but act on the p-dimensional space of the columns of the matrix J. This invariance property is used throughout all the derivation of the results. To find the maxima of $\tilde{\mathcal{I}}$ we first look for its fixed points, and then, by a stability analysis, we determine which of these fixed points are maxima. Each fixed point is actually an hypersurface, due to the invariance property.

Fixed Points

The fixed points are given by the following matrix equation:

$$\frac{\partial\tilde{\mathcal{I}}}{\partial J} = \frac{\partial\mathcal{I}}{\partial J} - \rho J = 0. \qquad (7)$$

Computing the derivative of $\mathcal{I}$ we find, after some rearrangements:

$$J\mathcal{C} = (b\mathbf{1}_p + b_0 JJ^T)\rho J + J\mathcal{C}J^T(b\mathbf{1}_p + b_0 JJ^T)^{-1}Jb_0 + J\mathcal{C}J^T\rho J. \qquad (8)$$

Now define Γ as the subspace of $\mathbf{R}^{\mathbf{N}}$ spanned by the vectors $\vec{J}_i$, $i = 1,\ldots,p$ at a fixed point (the dimension of Γ is so far unspecified); then consider an N-component vector $\vec{X} \in \Gamma^{\perp}$ and right multiply (8) by $\vec{X}$; from the fact that $J\vec{X} = 0$ by definition, we obtain:

$$J\mathcal{C}\vec{X} = 0 \Longrightarrow \mathcal{C}\vec{X} \in \Gamma^{\perp}. \qquad (9)$$

This means that $\Gamma^{\perp}$ is an invariant subspace of $\mathcal{C}$; since $\mathcal{C} = \mathcal{C}^T$ this also means that Γ is an invariant subspace of $\mathcal{C}$. So our first result is that at the fixed points the vectors $\vec{J}_i$ lie in a subspace spanned by (a so far unknown number of) eigenvectors of $\mathcal{C}$.

It can be proved that, at the fixed points, the same orthogonal transformation *simultaneously* diagonalizes the symmetrical pxp matrices JJ^T and $J\mathcal{C}J^T$. Therefore, in any hypersurface in J space where $\tilde{\mathcal{I}}$ is an extremum, there is a point (apart from permutations of the vectors $\vec{J}_i$), where the matrices JJ^T and $J\mathcal{C}J^T$ are both diagonal; we can loosely say, for short, that when we are at this point we are in the diagonal base. We

continue the study of the properties of the extrema of $\tilde{\mathcal{I}}$ in the diagonal base. In this base $JJ^T \longrightarrow \mathcal{D}$ and $JCJ^T \longrightarrow \mathcal{D}^1$, where $\mathcal{D}$ and $\mathcal{D}^1$ are diagonal pxp matrices; we denote their elements by: $\mathcal{D}_{ij} = \delta_{ij}\, f_i$, and $\mathcal{D}^1_{ij} = \delta_{ij}\, \alpha_i$. Notice that $f_i = ||\vec{J}_i||^2$ in the diagonal base. We right multiply (8) by J^T, and write the resulting equation in the diagonal base, to obtain:

$$\mathcal{D}^1 = (b\mathbf{1}_p + b_0\mathcal{D})\rho\mathcal{D} + \mathcal{D}^1(b\mathbf{1}_p + b_0\mathcal{D})^{-1}b_0\mathcal{D} + \rho\mathcal{D}^1\mathcal{D}. \qquad (10)$$

It can be proved that in the diagonal base the vectors $\vec{J}_i$ are eigenvectors of $\mathcal{C}$ corresponding to eigenvalues $\lambda_{k(i)}$, and that $\alpha_i = \lambda_{k(i)} f_i$. The value $k(i)$ is so far arbitrary, the only condition being that different i are associated to different k, since JJ^T is diagonal. The eigenvalues of $\mathcal{C}$, all positive, are numbered such that $\lambda_1 > \lambda_2 > \ldots > \lambda_N > 0$. Now (10) gives an equation for f_i. For each i, this equation always admits three real solutions; one is always zero, one is always negative, and the third is positive if:

$$\rho b < \lambda_{k(i)}; \qquad (11)$$

if this expression is not satisfied also the third solution is negative. Since negative solutions for f_i are not acceptable, we are left, for each i, with a choice between the solution $f_i = 0$ and the positive solution, provided (11) is satisfied. The appropriate choice to be made is determined by the stability analysis.

Stability Analysis

We give in the following an outline of the procedure, omitting the details of the heavy algebra involved.

To determine, among the fixed points, the maxima of $\tilde{\mathcal{I}}$, we perform a stability analysis. More precisely, we write the matrix expression

$$\Delta J = \frac{\partial \tilde{\mathcal{I}}}{\partial J} = \frac{\partial \mathcal{I}}{\partial J} - \rho J, \qquad (12)$$

where ΔJ is a finite variation of J in which each element J_{ij} changes by a quantity equal to the component of the gradient of $\tilde{\mathcal{I}}$ on the axis labeled by (i, j) of the Nxp-dimensional space of the Js. In (12) we substitute for J the generic fixed point plus a small perturbation, i.e., denoting by J_0 the generic fixed point solution, and by ε the perturbation, we put $J \longrightarrow J_0 + \varepsilon$. We linearize the resulting equation keeping only the terms of the first order in the perturbation; we then project the variation of J onto the possible directions in J space and verify in this way if that fixed point is stable. As before, we work in the diagonal base.

We multiply (12) by a complete base of the N-dimensional space, thus exhausting all the possible directions in the J, Nxp-dimensional

space. For convenience we divide the process in two steps: first we project onto a complete base of $\Gamma^{\perp}$ and then onto one of Γ. At the end of this analysis we can determine which of the fixed points are stable. In the next subsection we show the characteristics of these stable fixed points.

The Stable Fixed Points

We define the number m, determined by the number q of eigenvalues of $\mathcal{C}$ which are greater than ρb: if $q \leq p$, then $m = q$, otherwise $m = p$. Above, studying the generic fixed point, we have seen that, in the diagonal base, each f_i is associated with an eigenvalue $\lambda_{k(i)}$ of $\mathcal{C}$; besides, if $\rho b < \lambda_{k(i)}$ we have the freedom to choose $f_i = 0$ or $f_i > 0$, otherwise only the solution $f_i = 0$ exists. The stability analysis show that the stable fixed points are those for which:

- In the diagonal base, m vectors $\vec{J}_i$ are associated with $\lambda_1, \ldots, \lambda_m$, and the corresponding f_i are positive; if $m < p$, the remaining $(p - m)$ $\vec{J}_i$ are zero. All the other J configurations where $\tilde{\mathcal{I}}$ is maximum can be reached performing an orthogonal transformation $J \rightarrow \mathtt{A}J$. As a consequence, in a generic base, $p - m$ vectors $\vec{J}_i$ are linearly dependent on the other m. The conclusion is that the vectors $\vec{J}_i$, $i = 1, \ldots, p$ lie in a subspace Γ spanned by the first m eigenvectors of $\mathcal{C}$.

It has to be noted that when the channel noise b increases, higher and higher principal components are destabilized: in the diagonal base more and more vectors $\vec{J}_i$ go to zero, while in a generic base the decrease of $\dim \Gamma$ shows up by the decrease of the number of linearly independent vectors. In particular, when $\rho b > \lambda_1$, all the vectors $\vec{J}_i$ are zero. The input noise b_0 is not relevant in the determination of the noise thresholds, but only in fixing the value of $\tilde{\mathcal{I}}$, in particular at the maximum. Another point to be noted is that in the diagonal base the output distribution $p(\vec{V})$ is factorized, and the non-zero $\vec{J}_i$ produce at the output the projection onto the principal components of the input distribution.

In Fig. 2 we show, for the optimal network, in the diagonal base, the output distribution $p(\vec{V})$ and the conditional distribution $p(\vec{V}|\vec{\xi})$.

The Global Constraint

Now the function to be maximized is $\mathcal{I}$ itself, but under the constraint $\sum_{ij} J_{ij}^2 = \sigma$, that means that the sum of the square moduli of the vectors $\vec{J}_1, \ldots, \vec{J}_p$ is constant. We notice that the expression which is to be kept constant can also be written as $\mathtt{Tr}JJ^T$; from here we see that, like $\mathcal{I}$, this quantity is invariant under any orthogonal transformations $\mathtt{A}$. This

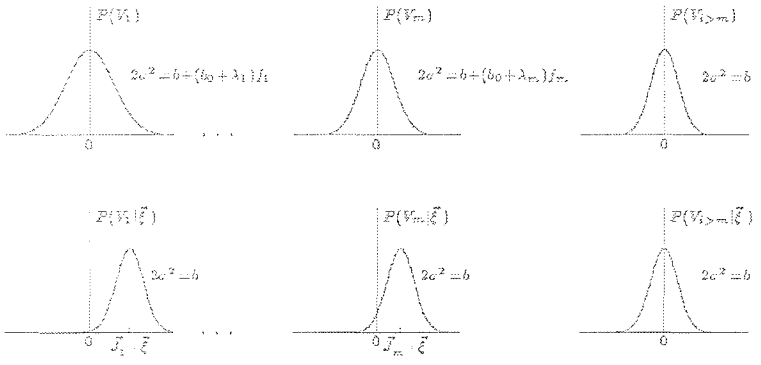


Figure 2: Case $m < p$. First row: the output activity distribution; second row: the conditional distribution of the output for a generic $\vec{\xi}$.

gives the possibility to study the fixed points in the diagonal base, as in the damped case.

To find the fixed point we have to solve the equation:

$$\frac{\partial \mathcal{I}}{\partial J} - \rho J = 0, \tag{13}$$

where now ρ is a Lagrange multiplier, needed to satisfy the constraint. The analysis proceeds as before. The conclusion for the stable fixed points is the same as that emphasized with the black dot in the previous subsection. The difference is in the dependence of the value of m on the noises b and (now also) b_0. Without showing the cumbersome expression that gives this dependence, we point out the most relevant feature:

- At fixed b_0, increasing b starting from $b = 0$ (or from an arbitrarily small positive value if $b_0 = 0$, to avoid $\mathcal{I} \to \infty$), one crosses successively $p - 1$ thresholds, in each one of which the dimension of the space spanned by the vectors $\vec{J}_i$ decreases by one, starting from p; at the end the dimension of the space is one (as expected, at least f_1 must remain positive to satisfy the constraint). At fixed b, and increasing b_0 starting from $b_0 = 0$, the situation is the following. For $b_0 = 0$ the dimension of the space spanned by the vectors J_i depends on the value of b; it can be computed that the dimension is p if $b < (\sigma\lambda_p)/(p - \lambda_p \sum_{i=1}^{p} \frac{1}{\lambda_i})$. Increasing b_0 one crosses successively the thresholds at which the dimension of the space increases by one up to the value p.

To summarize, the maximization of $\mathcal{I}$ under the global constraint leads to J configurations that have the same general properties as in the damped case. The main difference is in the determination of the noise thresholds, where the dimension of Γ changes. Now both the channel and the input noise, b and b_0, are relevant.

REFERENCES

[1] H. B. Barlow, "Unsupervised learning," *Neur. Comp.*, **1**, 295-311 (1989).

[2] R. Linsker, "Self-organization in a perceptual network," *Computer*, **21**, 105-117 (1988).

[3] J. J. Atick, "Could information theory provide an ecological theory of sensory processing?" *Network* **3** 213-251 (1992).

[4] E. Oja, "A simplified neuron model as a principal component analyzer," *J. Math. Biol.*, **15**, 267-273 (1982).

[5] E. Oja, "Neural networks, principal components, and subspaces," *Int. J. Neur. Syst.*, **1**, 61-68 (1989).

[6] A. S. Krogh and J. A. Hertz, "Hebbian learning of principal components," in *Parallel Processing in Neural Systems and Computers*, R. Eckmiller, G. Hartmann and G. Hauske (eds.) (Elsevier, 1990), 183-186.

[7] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neur. Networks*, **2**, 459-473 (1989).

[8] H. B. Barlow, "The coding of sensory messages" in *Current Problems in Animal Behaviour* ed W H Thorpe and O L Zangwill (Cambridge University Press 1990) pp 331-360.

[9] R. Linsker, "An application of the principle of maximum information preservation to linear systems," in *Advances in Neural Information Processing Systems I*, D. S. Touretzky (ed.) (Morgan Kaufmann, 1989), 186-194.

[10] R. Linsker, "Deriving receptive fields using an optimal encoding criterion" in *Advances in Neural Information Processing Systems 5* ed S J Hanson, J Cowan, and C L Giles (Morgan Kaufmann: San Mateo 1993) pp 953-960

[11] J. J. Atick and A. N. Redlich, "Quantitative tests of a theory of retinal processing: Contrast sensitivity curves" *IASSNS-HEP-90/51* (1990)

[12] J. H. van Hateren, "Theoretical predictions of spatiotemporal receptive fields of fly LMCs, and experimental validation" *J. Comp. Physiology* **A 171** 157-170 (1992)

[13] A. Campa, P. Del Giudice, J.-P. Nadal and N. Parga, "Neural networks as optimal information processors," *Int. J. Mod. Phys.*, **C5**, 855-862 (1994).

[14] A. Campa, P. Del Giudice, N. Parga and J-P. Nadal, "Maximization of mutual information in a linear noisy network: a detailed study", *Preprint INFN-ISS 95/3*, submitted to *Network* (1995)
[15] T. M. Cover and J. A. Thomas, *Elements of Information Theory* (Wiley 1991)

CONSTRAINED POLE-ZERO FILTERS AS DISCRETE-TIME OPERATORS FOR SYSTEM APPROXIMATION

ANDREW D. BACK AND AH CHUNG TSOI
Department of Electrical and Computer Engineering
University of Queensland
Brisbane, Queensland 4072. Australia
Email: {back,act}@elec.uq.oz.au

Abstract–

Discrete-time models whether linear or nonlinear, often implicitly use the shift operator to obtain input regression vectors. It has been shown recently that the significantly better performance can be obtained in terms of coefficient sensitivity and output error by using alternative operators to the usual shift operator. These include the delta and gamma operators. In this paper we introduce second order pole-zero operators which have more general modelling properties than those previously considered. We provide some observations on the behaviour of the operators, considering representational issues and convergence chacteristics in particular.

1 INTRODUCTION

In neural networks applied to signal processing applications, various approaches have been proposed to combine the usual linear filtering methods with the nonlinear function approximation capabilities of neural networks. A common method is to simply introduce regression vectors [1] defined as $\mathbf{u(t)} = [u(t), ..., u(t-M)]^T$ for some input signal $u(t)$, to a network architecture such as a multilayer perceptron. This is equivalent to prefiltering the input data by linear filters. An extension of this approach is to allow linear filters to be used in each of the synaptic connections **in all layers** rather than just the input layer. [3, 4, 33, 34].

Agarwal and Burrus [1] proposed the use of an alternative discrete-time operator to replace the shift operator. Their idea was to introduce the delta operator defined as $\delta = \frac{z-1}{\Delta}$ where Δ is the discrete-time sampling interval. Since then, this operator has been considered in linear filtering, estimation and control [13, 21, 22, 32].

Recently, de Vries and Principe [10, 11], have proposed an approach to modelling time series data, where instead of a simple time-delay input window constructed by means of the usual backward shift operator (defined as $z^{-1}x(t) \triangleq x(t-1)$), first order filters, called gamma filters, have been proposed. The gamma operator is defined as $\gamma = (z-(1-c))/c$.

An extension to the basic gamma operator by introducing complex poles, was given by [29]. In this case, the second order operator is derived by considering the usual gamma operator and replacing a shift operator function and feedforward gain within the gamma operator by the initial gamma operator function. This results in an operator having a zero at $z = 1 - c$, and a possible pair of complex conjugate poles. This operator is defined as

$$\gamma_2 = \frac{c_1\left[z-(1-c_1)\right]}{\left[z-(1-c_1)\right]^2 + c_2 c_1^2} \tag{1}$$

[1]The term regression vector is used in system identification literature to denote the input vectors.

0-7803-2739-X/95 $4.00 © 1995 IEEE

where c_i, $i = 1, 2$ are parameters of the γ_2 operator. The results in [29] indicate some success with this method, however it was observed that a multimodal mean square error surface may occur in some modeling situations. In order to overcome this, a subdivision search strategy was proposed, allowing the operator coefficients to be trained first, followed by the neural network, and then finally training all weights simultaneously.

In the γ_2 operator proposed in [29], while there may be a complex pole pair within the unit circle (if $(1 - c_1)^2 + c_1^2 c_2 < 1$), there is only one zero which lies on the real axis. This means that the operator is only capable of producing either low pass (for $0 < c_1 < 1$), or high pass filtering characteristics ($1 < c_1 < 2$).

In this paper we introduce first more general second order operators. Some pertinent observations are made on the behaviour of some of operators which are capable of providing a more general transfer function characteristic (viz, bandpass, bandstop, notch).

It is useful to define some criteria for selecting one operator over another. A first-choice for this has been the improvement in mean-square output error (MSOE) obtained by using the operator in a learning algorithm [6, 13, 26, 27]. The precise reasons for the improvement in MSOE improvement still require some elucidation. It is known that a large eigenvalue spread will lead to poor convergence of parameters in on-line estimation, while significant off-diagonal elements can cause a convergence to a non-optimal point [15].

In the development in [15] it was assumed that the operators contained no adaptive parameters. However, in this paper, we propose to consider the implications of using (i) a range of possible operators, specifically, we are interested in the use of second order pole-zero operators, and (ii) allowing the operators to have parameters which we may require to be adjusted on-line. Further, we are concerned with issues in identification of nonlinear dynamic models, which consists of some linear dynamic input (preprocessing) stage, i.e., this is usually a time-delay input window, but in our case we centre the discussion on using alternatives to the shift operator; followed by a neural network structure and possibly additional dynamic structures.

In this paper we consider representational issues of the operators and some initial considerations of the convergence chacteristics of the operator models. The issue of persistence of excitation of the input signals is raised, and how it relates to the covergence of nonlinear neural network models. We indicate conditions under which persistence of excitation applies to linear dynamic parts of neural network models. We stress at this point that the analysis presented here is not generally applicable to static neural network structures, but rather a subclass of models which includes linear dynamic (in particular, input preprocessing) sections. The general problem of persistence of excitation for neural networks has not been discussed widely, and there are very little known results.

One of the major aims of this paper is to show that there is a need to consider carefully, the implications operators have in the overall convergence of the linear dynamics in a model. These results are strictly true for a linear system, however our conjecture is that the same principles will apply to nonlinear systems which contain linear "sub-models" also requiring parameterization to adequately model some unknown system.

2 OPERATOR MODEL ARCHITECTURES

2.1 Single layer Models

In this section, we describe the neural network architecture and associated operators being considered. A model which generalizes the usual discrete-time linear moving average model, ie, a single layer network is given by

$$\hat{y}(t) = G(\nu, \theta)x(t) \tag{2}$$

$$G(\nu,\theta) = \sum_{i=0}^{M} b_i \nu^{-i} \tag{3}$$

where ν^{-1} is a generic operator. This operator may be defined as $\nu^{-i} = \{z^{-i}, \delta^{-i}, \gamma^{-i}, \gamma_2^{-i}\}$, which are the shift, delta, gamma, and gamma(2) operators respectively. The forward equations for the model are

$$\hat{y}(t) = \sum_{i=0}^{M} b_i u_i(t) \tag{4}$$

where we define

$$u_0(t) \triangleq x(t) \tag{5}$$

$$u_i(t) \triangleq \nu^{-1} u_{i-1}(t) \tag{6}$$

Thus we can define the regression vector $\mathbf{u(t)} = [u(t), \nu^{-1}u(t), ..., \nu^{-M}u(t)]^T$.

2.2 Nonlinear Multilayer Models

A nonlinear model may be defined using a multilayer perceptron (MLP) with the ν-operator elements at the input stage. This model is termed the ν-operator multilayer perceptron or MLP(ν) model. An MLP(ν) is defined in the same manner as a usual MLP with input vector $\mathbf{u(t)}$.

2.3 Operator Preprocessing Stage

In this paper we focus on the effects alternative discrete-time operators have on the covergence of model parameter estimates during learning. The model parameter estimates we are primarily concerned with in the first instance are those of the linear dynamic sections in the model. Our conjecture is that in terms of obtaining a good nonlinear dynamic model, it is necessary to properly model these sections in order to obtain a good nonlinear model, and that the issue of persistence of excitation normally considered for linear model estimation also requires consideration in this framework.

It is known, (and described in more detail later), that model convergence is strongly dependent on the conditioning of the covariance matrix R_{uu} of the regression vector $\mathbf{u(t)}$. The regression variables we consider are obtained from the output of the operators[2]. In particular it is shown that the spectral characteristics of the operator $\nu(z,\theta)$ influences the conditioning of R_{uu}. For that reason we propose a general operator capable of providing arbitrary location of poles and zeros. A general operator model to do this is defined by

$$H_o(z) = \prod_{i=1}^{N} \frac{(z-\beta_i)(z-\beta_i^*)}{(z-\alpha_i)(z-\alpha_i^*)} \tag{7}$$

where α_i, β_i are the complex poles and zeros. Interestingly, a version of this operator has been proposed in the context of frequency transformations for digital filter design [9], where the transformation is all-pass and the unit circle is mapped onto itself. In our development, we will primarily consider the special case where $i = 1$.

[2]As a means of simplifying our approach to convergence in both the operator and other possible linear dynamic section in the neural network, we assume white gaussian noise input to the system, and consider a nominal regression vector consisting of only the operator outputs.

The operator structure we consider therefore is defined as

$$\mu^{-1} = \frac{z^2 + r_z c_z z + r_z^2}{z^2 + r_p c_p z + r_p^2} \tag{8}$$

where r_p, r_z are pole and zero radii respectively, while the pole frequency and zero frequencies are given by $\omega_i = \arccos(c_i/-2)\mathrm{i}$, $i = z, p$. In contrast to allowing only low pass or high pass spectral characteristics as is the case for the gamma(2) operator, this pole-zero operator allows arbitrary positioning of complex conjugate poles and zeros and can therefore implement bandpass, bandstop, notch filters etc.

3 OBSERVATIONS ON THE BEHAVIOUR OF THE OPERATORS

3.1 Algebraic Operator Constraints

The following observations are made concerning the pole-zero characteristics of the operators under consideration. These results are relevant in understanding possible convergence problems discussed in the following section.

O.1 A single γ_2-operator is not capable of producing complex zeros. However, complex zeros may be obtained in the overall model through the interaction of the operator elements.

By inspection, in the γ_2-operator transfer function in (1), it can be seen that since $c_1 \in \Re$, the zero must be on the real axis.

Is there a relationship between the real axis zeros, and the poles ? This question is answered in the following observations.

O.2 The γ_2-operator has a single zero given by $\hat{z} = (1 - c_1)$, and poles given by $\hat{p_{1,2}} = (1 - c_1) \pm j c_1 \sqrt{c_2}$. Thus, the poles and zeros are constrained to move identically in terms of the real axis coordinate.

O.3 The poles and zero of the γ_2-operator are constrained such that increasing c_1 causes a decrease in the position along the real axis of the poles and zero, but simultaneously causes an increase in the imaginary axis coordinate according to the square root of the coefficient. Adjusting c_2 affects only the imaginary component of the poles, however this results in a simultaneous shift in the radius and angular position of the poles, where $r = \sqrt{(1 - c_2)^2 + c_1^2 c_2}$, $\theta = \arctan(\frac{c_1\sqrt{c_2}}{1 - c_1})$. Coefficient adjustments therefore affect the poles and zeros in a nonlinear and nonsymmetric manner.

O.4 The poles of the γ_2-operator will occur in a complex conjugate pair for $c_2 \neq 0$.

O.5 The angular frequency and radius of the μ-operator poles and zeros are independently adjustable. Thus, learning algorithms are able to perform pole-zero updating directly if required, rather than coefficient updating.

O.6 Second order μ-operators can be structured in a coupled-form [17] to allow low sensitivity between coefficients and pole/zero positions.

O.7 The μ-operator may be constrained in numerous ways in order to obtain specific frequency domain characteristics, e.g. a notch filter is obtained as

$$\mu_{notch}^{-1} = \frac{z^2 + c_p z + 1}{z^2 + r_p c_p z + r_p^2} \tag{9}$$

It is possible to make the following observations about the notch operator in Observation O.7:

O.8 Complex conjugate pole pairs are defined for the μ_{notch}-operator, when $-2.0 \leq c \leq 2.0$. A pole on the real axis occurs when $-2.0 \geq c \geq 2.0$.

O.9 A μ_{notch}-operator which models poles on the real axis will be nonminimum phase. Thus the μ_{notch} operator model is likely to have difficulty in learning inverse models of systems which have real axis poles.

This is shown as follows. Consider the system polynomial $B(z) = z^2 + cz + 1$. We wish to find conditions of nonminimum phase for $c \neq 2.0$. Let the zeros of $B(z)$ be given by $\hat{z}_B = 0.5(-c \pm \sqrt{c^2 - 4.0})$. Hence let the zero radius in the complex plane be r, where $r^2 = 0.25(c^2 + \sqrt{c^2 - 4.0})$. Thus, for $c > 2.0$, $r^2 > 1.0$ and the property is evident.

Viewing the root locus plots of the operators as a function of the operator variables would indicate these properties more clearly. There are fundamental differences between how the γ_2-operator and the μ-operator model zeros on the real axis. It would be of interest to understand how this difference relates to modeling systems which require a pole on the real axis.

3.2 Convergence and Persistent Excitation Conditions

We now move our attention to considering the convergence aspects of the operator models. For a multilayer network with an input time delay window, it can be easily shown that the time-delays and weights going to each node in the first layer form separate FIR (finite impulse response) filters $G_j(z)$ $j = 0, ..., N_1$ from the input signal $x(t)$ ($u_0(t) = x(t)$, to the outputs $\hat{y}_j(t)$, the activations of the N_1 first layer units.

As in the usual system identification approach [19], we make the assumption that the dynamics of the model must approximate that of the system sufficiently well in order to obtain a good approximation. This may not be strictly true due to the action of the nonlinearities in the network, and the magnitude of the incoming signals. However for the purposes of our discussion, we restrict the analysis to this "tighter" assumption for the following reasons.

In the course of this analysis, we assume that there may be dynamic filter structures either within the network structure itself (e.g. such as in locally recurrent networks), or occuring after the output of the network. Further, it is likely that there will be some neurons in network structures which do operate in the linear regions for some periods of time whether due to the input signal amplitude, or the input weights.

Thus, we stress that for instances where the neurons are driven well into the nonlinear range, the considerations we present in this paper are not necessarily directly applicable. We conjecture however, that in order to model nonlinear dynamic systems appropriately, it is necessary to consider the implications of persistence of excitation associated with the linear portions of the overall model. We consider that the assumption that the nonlinear components of, for example, an MLP model with dynamic input structure, and possibly some dynamic output structure, will cause there to be no need to admit persistence of excitation conditions to the overall model convergence results, to great to make at this stage. In fact, it is has been observed in the course of simulations of dynamic locally recurrent networks, that in many cases there are units which perform only linear processing [8]. Thus there is evidence to suggest that in modelling "block-oriented" nonlinear systems (i.e. constructed of discrete linear and nonlinear functional blocks) [7] using dynamic neural networks, it will be necessary to consider convergence aspects in this framework. The principles of analysis presented here can be regarded

as a first step in assisting our understanding of neural networks in dynamic system approximation.

Thus, in order to obtain convergence of the network to adequately approximate the system or data in question, it is assumed that the covariance matrix R_{uu} of the regressor variables, u(t) is nonnegative definite. This implies that the incoming signal $u(t)$ (assumed scalar), is persistently exciting (pe) of order M [28][3]. The definition of persistent excitation that we use is given by Anderson and Johnson [2], as

$$k_1 I \quad > \quad \sum_{l=k}^{k+p} \mathrm{u(l)u(l)}^T \quad > \quad k_2 I \tag{10}$$

where $k_1, k_2 > 0$. This condition implies that the input signal must be sufficiently rich in frequencies so that the smallest eigenvalue of $\sum_{l=k}^{k+p} \mathrm{u(l)u(l)}^T$ is bounded above zero.

In introducing operators to a network, an initial question we may ask concerns their effect on the persistent excitation of the incoming signals. In this context, we are interested to consider some basic issues concerning filtering being performed by any given operator. This question is considered in the following principles. For clarity of explanation, we adopt the notation of $H_i(z)$ representing a particular operator i.

Consider a linear operator defined in terms of an ARMA (autoregressive moving average) process, where

$$A_i(z^{-1})u_{i+1}(t) \quad = \quad B_i(z^{-1})u_i(t) \tag{11}$$

represents the operator process from the ith regressor variable $u_i(t)$ to the $i+1$th variable, and $u_0(t) = x(t)$. We have the following properties [28]:

Property P.1 [19]

An ARMA process is persistently exciting of any finite order. This follows due to the fact that the spectral density matrix of an ARMA process due to $A_N(z^{-1})/B_M(z^{-1})$, is positive definite for almost all frequencies in $(-\pi, \pi)$.

Property P.2 [19]

$H(z^{-1}) = A(z^{-1})/B(z^{-1})$ is an asymptotically stable filter with p zeros on the unit circle. If $u_{i-1}(t)$ is a pe(r) (i.e. persistently exciting of order r) scalar signal, then $u_i(t)$ is pe(m), where $r - p \leq m \leq r$.

If $H(z^{-1})$ has no zeros on the unit circle, then $u_{i+1}(t)$ is also pe(r) [28]. Hence for the regressor variables $\{u(t)\}$ processed by operators $\{H(z^{-1}) = A_N(z^{-1})/B_M(z^{-1})\}$, the covariance matrix R_{uu} is positive definite if $u(t)$ is pe($M+N$) and $A_N(z^{-1})$, $B_M(z^{-1})$ are coprime [28].

Now, although it is clear that for a persistently exciting input signal $x(t)$, the output $u_1(t)$ will also be persistently exciting, we are more concerned with the convergence characteristics governed by the the eigenvalue spread of the covariance matrix R_{uu}. According to the definition that the smallest eigenvalue of $\sum_{l=k}^{k+p} \mathrm{u(l)u(l)}^T$ is bounded above zero, we would like to know the effect of operator filtering on the condition number of the covariance matrix R_{uu}.

We would like to ascertain an understanding of the eigenvalue spread $V(R_{uu})$, or since R_{uu} is Hermitian, the condition number $\chi(R_{uu})$. In particular we are interested in an upper bound on $V(R_{uu})$ defined by

$$V(R_{uu}) \quad = \quad \frac{\lambda_{max}(R_{uu})}{\lambda_{min}(R_{uu})} \quad = \quad \chi(R_{uu}) \tag{12}$$

[3]In fact, for nonlinear systems, we also need to consider the amplitude of the signal, and hence can describe it in terms of persistently ranging characteristics [5].

A relationship can be found between the power spectral density $S(\omega)$ of the operator transfer function $H(z)$. It can be shown [18] that

$$\chi(R_{uu}) = \frac{\lambda_{max}(R_{uu})}{\lambda_{min}(R_{uu})} \leq \frac{S_{max}(H(\omega))}{S_{min}(H(\omega))} \quad (13)$$

Thus, the covariance matrix R_{uu} can become ill-conditioned from certain types of operators are used. For example, if $H(z)$ has zeros on or close to the unit circle, or $S_{min} \rightarrow 0$, then $\lambda_{min} \rightarrow 0$, hence $\chi(R_{uu}) \rightarrow \infty$.

The following property has been derived by Stoica [28] which is particularly relevant here.

Property P.3 [31]

For $y(t) = H(z)x(t)$, if $H(z)$ has zeros on or close to the unit circle, then the $M \times M$ covariance matrix R_{yy} will be ill-conditioned for large values of M [28].

Although the above conditions apply to all linear operators, there may be some advantage in using second order pole-zero operators, in the sense that complex conjugate poles and zeros can be directly and easily modelled. This direct control over the frequency response characteristics means that we can better manipulate the eigenvalue spread of the covariance matrix, given some particular input signal. This would allow better convergence properties to be obtained.

4 IMPLICATIONS OF THE OBSERVATIONS

In this section we present examples of the implications of the convergence analysis properties described in the previous section. It is possible to foresee problems that could arise in either of the following circumstances:

1. The incoming signal is not persistently exciting.
2. The operators have zeros at locations on the unit circle which cause the operator output signal $u_i(t)$ to lose persistence of excitation.
3. Either or both of the above conditions cause the covariance matrix to become ill-conditioned.

Example 1.

Suppose we have an operator $H_i(z^{-1})$ with small S_{min}, or the incoming signal resulted in a small S_{min}, then this will cause poor convergence. From this we conclude that some caution may need to be exercised in using filters for operator structures resulting in this condition.

Example 2.

Suppose we have a number of cascaded operators $H_i(z^{-1})$ (i=0,1,...M) each of possibly differing filtering characteristics associated S_{min}. If one operator k has a small S_{min}, then this will result in a "flow-on" to all succeeding operators, resulting in poor conditioning of all $R_{u_iu_i}$ for $i > k$.

Example 3.

Suppose a criterion other than persistence of excitation is used to select $H_i(z)$, e.g. mean square output error of the model. We raise the question of whether it is possible to lose persistence of excitation while adapting the operator parameters. This issue does not appear to have been considered directly in the literature.

Our interest is in introducing a broad set of principles which can govern the use of alternative discrete-time operators. The preceeding discussion indicates broad conditions which need to be placed on the design of operators.

It has been observed that poor learning may occur when using an on-line estimation algorithm to model ARMAX (autoregressive moving average exogenous input) systems [14] when the zeros of the input transfer function approach the unit circle. Friedlander proposed an inmprovement to the recursive maximum likelihood learning algorithm used, where a prefilter was used to pull the roots towards the center of the unit circle using an approach which replaces the usual polynomial $A(z) = \sum_i^N a_i z^{-i}$ by $\tilde{A} = \sum_i^N a_i (cz^{-i})$, where c is a pulling factor. As c decreases, the roots of $\tilde{A}(z)$ move radially inwards towards the origin[4] [14]. We propose that Friedlander's method could also be applied to the operator models by introducing either
(i) a regularization term into the update equations for the operator parameters, or
(ii) bounds on the positions that roots.
In this case, we propose an alternative pole-zero operator,

$$\tilde{\mu}^{-1} = \frac{z^2 + \tilde{r}_z c_z z + \tilde{r}_z^2}{z^2 + \tilde{r}_p c_p z + \tilde{r}_p^2} \tag{14}$$

where $\tilde{r}_z = p_z r_z$, $\tilde{r}_p = p_p r_p$ and p_z, p_p are pulling factors which we seek to minimize. Thus, a cost criterion may be defined as

$$J(t) = \eta_e \left[\frac{1}{2} \sum_{i=0}^{N_o} e_i(t)^2 \right] + \eta_z p_z^2 + \eta_p p_p^2 \tag{15}$$

Estimation of p_z, p_p will also be required during operator parameterization. Performance of this approach is currently being explored.

In the next section, we present numerical examples indicating the difficulties encountered in terms of the ill-conditioned nature of the covariance matrices R_{uu}.

5 NUMERICAL EXAMPLES

As an indication of the variation in conditioning that can occur in the covariance matrices, we show the variation of $\chi(R_{u_0 u_0})$ against variations in the operator parameters (Figure 1). For the purposes of this experiment, we assumed an FIR input stage, with $M = 3$, and use results obtained over 1000 sample points.

The examples presented indicate the large variations in conditioning possible within one operator structure. Further, it is evident that sometimes quite small changes in the operator parameters result in large changes in the condition number of the covariance matrix. It is of interest to note that the pole-zero model allows much smaller eigenvalue spread for some parameter regions than any of the other operators. By comparison, a shift operator obtained an eigenvalue spread of 1136 for the same experiment.

Open questions which we have identified are:
How to select operators which result in the smallest possible eigenvalue spread ?
Can we choose $H(z)$ such that any learning algorithm will always seek to obtain a frequency transformation which seeks to minimize the covariance matrix eigenvalue spread ? Further, is it possible to always choose an operator structure such that the eigenvalue spread is small ? In other words, is it possible to guarantee that such models will always exist ? Further work is required in this area, to understanding these issues in the context of neural networks.

[4]Note that this is in itself, a special case of an alternative operator.

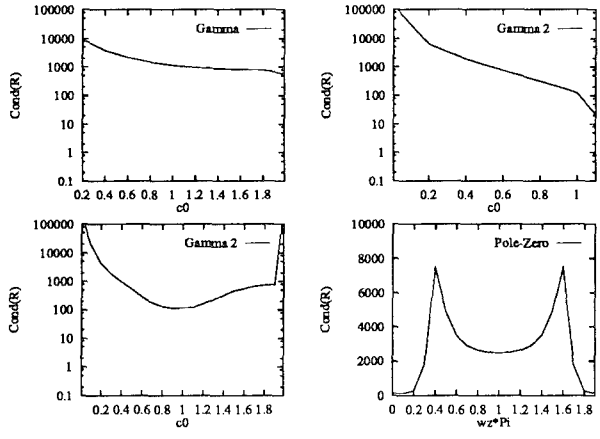


Figure 1: Condition number for data covariance matrix R_{u0u0} versus varying operator pole parameter: (a) Gamma Operator (vary c_0), (b) Gamma(2) Operator ($c_1 = 0.8$,vary c_0), (c) Gamma(2) Operator (vary c_0 linearly, vary c_1 to maintain constant pole radius $r = 0.9$), (d) Pole-Zero Operator ($r_p = 0.5, \omega_p = 0.3\pi, r_z = 1.0$, vary ω_z).

6 CONCLUSIONS

Recently, novel input structures have been proposed to replace the usual time delay (shift) operator commonly used to map time-varying signals to neural network architectures. Various advantages have been demonstrated for operators such as the Gamma operator [12] and the delta operator [21].

We have proposed a general second order pole-zero operator structure allowing complex poles and zeros to be independently adjusted. Various aspects of the operators have been considered, including an initial analysis of persistence of excitation conditions applicable to the (linear) input preprocessing stage to a neural network model. Results have been presented which indicate that the choice of the operator structure is vital if proper convergence is to be obtained in the linear dynamic sections of the model. Some advantages of the pole-zero have been indicated.

In future work we propose that it would be of some interest to closely examine how the operator parameters can be adjusted while maintaining well conditioned covariance matrices. This would enable us to exploit the variable parameters in the operators, but with some possible trade-off. It would also be particularly interesting to consider this development in light of the type of error surface analysis performed by Principe et. al. [26].

Acknowledgements

The first author acknowledges financial support from the Australian Research Council. The second author acknowledges partial support from the Australian Research Council.

References

[1] Agarwal, R.C., and Burrus, C.S., "New recursive digital filter structures having very low sensitivity and roundoff noise", IEEE Trans. Circuits and Systems, vol. cas-22, pp. 921-927, Dec. 1975.

[2] Anderson, B.D.O., and Johnson, C.R. Jr, "Exponential Convergence of Adaptive Identification and Control Algorithms", Automatica, Vol. 18, No. 1, pp. 1-13, 1982.
[3] Back, A.D. and Tsoi, A.C., "A Time Series Modelling Methodology Using FIR and IIR Synapses", Proc. Workshop on Neural Networks for Statistical and Economic Data, Dublin, DOSES, Statistical Office of European Communities, F. Murtagh (Ed.), pp. 187-194, 1990.
[4] Back, A.D. and Tsoi,A.C., "FIR and IIR Synapses, a New Neural Network Architecture for Time Series Modelling". Neural Computation, vol 3. no. 3, pp. 375-385, 1991.
[5] Back, A.D. "New Techniques for Nonlinear System Identification: A Rapprochement Between Neural Networks and Linear Systems". PhD Thesis, University of Queensland, 1992.
[6] Back, A.D., and Tsoi, A.C., "A comparison of discrete-time operator models for nonlinear system identification", Advances in Neural Information Processing Systems 7, G. Tesauro, D. S. Touretzky and T. K. Leen, eds., Cambridge MA: MIT Press, 1995.
[7] Billings, S.A., "Identification of nonlinear systems - a survey", Proc. IEE, Pt. D, Vol 127, No. 6, pp. 272-285, 1980.
[8] Burrows, T.L., and Niranjan, M., "The use of feed-forward and recurrent neural networks for system identification", Tech. Rep. CUED/F-INFENG/TR158., Cambridge University, England. 1993
[9] Constantinedes, A.G., "Design of bandpass digital filters", Proc. IEEE, Vol. 57, pp. 1229-1231, 1969.
[10] de Vries, B. Principe, J.C. "A theory for neural networks with time delays", Advances in Neural Information Processing Systems, 3, R.P. Lippmann (Ed.), pp 162 - 168, 1991.
[11] de Vries, B., Principe, J. and P.G. de Oliveira "Adaline with adaptive recursive memory", Neural Networks for Signal Processing I. Juang, B.H., Kung, S.Y., Kamm, C.A. (Eds) IEEE Press, pp. 101-110, 1991.
[12] de Vries, B. Principe, J. "The Gamma Model – a new neural model for temporal processing". Neural Networks. Vol 5, No 4, pp 565 - 576, 1992.
[13] Fan, H., and Li, Q., "A δ operator recursive gradient algorithm for adaptive signal processing", Proc. IEEE Int. Conf. Acoust. Speech and Signal Proc., vol. III, pp. 492-495, 1993.
[14] Friedlander, B., "A modified prefilter for some recursive parameter estimation algorithms", IEEE Trans. Automat. Control, 27 No. 1, pp. 232-235, 1982.
[15] Gevers, M., and Li, G. Parameterizations in Control, Estimation and Filtering Problems: Accuracy Aspects. London:Springer-Verlag, 1993.
[16] Goodwin, G.C., Middleton, R.H., and Poor, H.V., "High-speed digital signal processing and control", Proc. IEEE, vol. 80, no. 2, pp. 240-259, 1992.
[17] Gold, B., and Rader, C.M., Digital Processing of Signals. NY: McGraw-Hill, 1969.
[18] Haykin, S., Adaptive Filter Theory. Prentice Hall: Englewood Cliffs, NJ, 1986.
[19] Ljung, L., and Söderström, T., Theory and Practice of Recursive Identification, Cambridge, Massachusetts: The MIT Press, 1983.
[20] Middleton, R.H., and Goodwin, G.C., "Improved finite wordlength characteristics in digital control using delta functions", IEEE Trans. Automat. Contr., vol. AC-31, pp. 1015-1021, 1986.
[21] Middleton, R.H., and Goodwin, G.C., Digital Control and Estimation, Englewood Cliffs: Prentice Hall, 1990.
[22] Peterka, V., "Control of Uncertain Processes: Applied Theory and Algorithms", Kybernetika, vol. 22, pp. 1-102, 1986.
[23] Palaniswami, M., "A new discrete-time operator for digital estimation and control". The University of Melbourne, Department of Electrical Engineering, Technical Report No.1, 1989.
[24] Palaniswami, M., "Digital Estimation and Control with a New Discrete Time Operator", Proc. 30th IEEE Conf. Decision and Control, pp. 1631-1632, 1991.
[25] Principe, J.C., de Vries, B., Kuo J-M., and Guedes de Oliveira, P., "Modeling Applications with the Focused Gamma Net", Advances in Neural Information Processing Systems, vol. 4, pp. 143-150, 1991.
[26] Principe, J.C., de Vries, B., and Guedes de Oliveira, P., "Generalized Feedforward Structures: A New Class of Adaptive Filters", ICASSP-92, Vol. IV, pp. 245-248, 1992.
[27] Principe, J.C., de Vries, B., and Guedes de Oliveira, P., "The Gamma Filter - a new class of adaptive IIR filters with restricted feedback", IEEE Trans. Signal Processing, vol. 41, pp. 649-656, 1993.
[28] Söderström, T., and Stoica, P. System Identification. London: Prentice Hall, 1989.
[29] Oliveira e Silva, T., de Oliveira, P.G., Principe, J.C., de Vries, B., "Generalized feedforward filters with complex poles", Neural Networks for Signal Processing II, S.Y. Kung et. al. (Eds) Piscataway,NJ: IEEE Press, pp. 503-510, 1992.
[30] Stoica, P., "On multivariate persistently exciting signals", Bul. Inst. Politehnic Bucuresti, Vol. 43, pp. 59-64. 1981.
[31] Stoica, P., Friedlander, B., and Söderström, T., "An Approximate Maximum Likelihood Approach to ARMA Spectral Estimation", Proc. 24th IEEE Conference on Decision and Control, Fort Lauderdale, 1985.
[32] R. Vijayan, H.V. Poor, J.B. Moore and G.C. Goodwin, "A Levinson-type algorithm for modelling fast-sampled data", IEEE Trans. Automat. Control, vol. 36, pp. 314-321, 1991.
[33] Wan, E.A., "Temporal backpropagation for FIR neural networks", Proc. Int. Joint Conf. Neural Networks, San Diego, , vol I, pp 575-580, 1990
[34] Wan, E.A., "Time Series Prediction by Using a Connectionist Network with Internal Delay Lines'", in A. Weigend and N. Gershenfeld, eds., Time Series Prediction: Forecasting the Future and Understanding the Past. Addison-Wesley, pages 195-218, 1994.

PRIOR KNOWLEDGE AND THE CREATION OF "VIRTUAL" EXAMPLES FOR RBF NETWORKS

Federico Girosi and Nicholas Tung Chan

Center for Biological and Computational Learning
and
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge MA 02139 USA

Abstract

We consider the problem of how to incorporate prior knowledge in supervised learning techniques. We set the problem in the framework of regularization theory, and consider the case in which we know that the approximated function has radial symmetry. The problem can be solved in two alternative ways: 1) use the invariance as a constraint in the regularization theory framework to derive a rotation invariant version of Radial Basis Functions; 2) use the radial symmetry to create new, "virtual" examples from a given data set. We show that these two apparently different methods of learning from "hints" (Abu-Mostafa, 1993) lead to exactly the same analytical solution.

1 Introduction

Lack of examples is very often responsible for poor performances of learning algorithms. In many cases it is difficult, if not impossible, to collect additional data, leaving us with unsatisfactory solutions. However, it is often the case that not only the examples but also some prior knowledge on the learning target are available. Examples of prior knowledge are smoothness, invariance with respect to transformation groups (such as rotations or reflections) or information about time and/or space scale. Most of the existing learning schemes do not make use of prior knowledge, and therefore provide suboptimal solutions that do not fully exploit the amount of information available.

One major contribution to this topic has been given by Abu-Mostafa (1993) who developed a methodology for integrating different kinds of "hints" (prior knowledge) into the usual learning-from-example procedure, and related them to the well-known concept of VC-dimension (Vapnik, 1982). Abu-Mostafa considers, among other things, the case in which the function that has to be learned is invariant with respect to certain transformations. In this case he argues that the "hints" can be represented by new examples, generated from the existing data set by applying transformations that are known

0-7803-2739-X/95 $4.00 © 1995 IEEE

to leave the function to be learned invariant. This approach can be applied independently of the learning technique that is used: the learning technique remains the same, and the data set is augmented with new, "virtual" examples.

An alternative approach consists in leaving the data set unaltered, but to use the prior knowledge to modify an existing technique to ensure that the approximated function has the desired invariance properties. This approach is clearly related to the creation of virtual examples, but it is not obvious that provides the same result.

In the following we apply this alternative technique to the case in which the prior knowledge consists in knowing that a function is radially symmetric. The approximation technique we consider is Radial Basis Functions, because it can be derived as the solution of a functional minimization problem, in which prior knowledge about the smoothness of the function is already used. The presence of a functional to be minimized makes easy to introduce the additional prior knowledge as a constraint over the domain of the functional, and will let us derive an analytical solution, that is as simple as in the Radial Basis Functions case.

Interestingly enough, the solution derived in this way is exactly the same that is obtained if the prior knowledge is used to create virtual examples, showing that the creation of virtual examples is the "right" thing to do, and providing another step in a rigorous mathematical analysis of the technique of virtual examples. Before describing these results we first briefly review the regularization theory approach to function approximation.

2 Regularization Theory and RBF

Suppose that the set $D = \{(\mathbf{x}_i, y_i) \in R^d \times R\}_{i=1}^N$ is a random, noisy sample of some multivariate function h. The problem of recovering the function h from the data D is ill-posed, and can be formulated in the framework of regularization theory (Tikhonov, 1963; Wahba, 1990; Poggio and Girosi, 1990). In this framework the solution is found by minimizing a functional of the form:

$$H[f] = \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2 + \lambda \phi[f] \ . \tag{1}$$

where λ is a positive number that is usually called the *regularization parameter* and $\phi[f]$ is a cost functional that constrains the space of possible solutions according to some form of prior knowledge. The most common form of prior knowledge is *smoothness*, that, in words, ensures that if two inputs are close the two corresponding output are also close. We consider here a very general class of rotation invariant smoothness functionals (Girosi, Jones and Poggio, 1995), defined as

$$\phi[f] = \int_{R^d} ds \, \frac{|\tilde{f}(\mathbf{s})|^2}{\tilde{G}(\mathbf{s})}$$

where $\tilde{}$ indicates the Fourier transform, $\tilde{G}$ is some positive radial function that tends to zero as $||\mathbf{s}|| \rightarrow \infty$ (so that $\frac{1}{\tilde{G}}$ is an high-pass filter). We consider here for simplicity of subsequent notations the case in which G (the Fourier transform of $\tilde{G}$) is positive definite, rather than conditionally positive definite (Micchelli, 1986), and therefore is a bell-shaped function. It is possible to show (see the paper by Girosi, Jones and Poggio, 1995, for a for a sketch of the proof) that the function that minimizes the functional (1) is a classical Radial Basis Functions approximation scheme (Micchelli, 1986; Moody and Darken, 1989):

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i G(\mathbf{x} - \mathbf{x}_i) \tag{2}$$

where the vector of coefficients $(\mathbf{c})_i = c_i$ satisfies the following linear system:

$$(G + \lambda I)\mathbf{c} = \mathbf{y} \tag{3}$$

where I is the identity matrix, and we have defined the vector of output values $(\mathbf{y})_i = y_i$ and the matrix $(G)_{ij} = G(\mathbf{x}_i - \mathbf{x}_j)$. Classical examples of basis functions G include the Gaussian $(G(\mathbf{x}) = \exp(-||\mathbf{x}||^2))$ and the inverse multiquadric $(G(\mathbf{x}) = (1 + ||\mathbf{x}||^2)^{-\frac{1}{2}})$. In the next section we will show how to embed the prior knowledge about radial symmetry in this framework and we will derive the corresponding solution.

3 Regularization Theory in Presence of Radial Symmetry

In the standard regularization theory approach, the minimization of the functional $H[f]$ is usually done on the space of functions Φ for which $\phi[f]$ is finite. If additional knowledge on the solution is known, that can be used to furtherly constrain the space of solutions. If we known that the solution is a function with radial symmetry, then we can restrict ourselves to minimize $H[f]$ over $\Phi \bigcap \mathcal{R}$, where $\mathcal{R}$ is the set of radial functions. The problem we have to solve now is therefore the following:

$$\min_{f \in \Phi \bigcap \mathcal{R}} H[f] = \min_{f \in \Phi \bigcap \mathcal{R}} \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2 + \lambda \phi[f] \; . \tag{4}$$

We now notice that any functions in $\mathcal{R}$ uniquely defines a one dimensional function f^* as follows

$$f(\mathbf{x}) \equiv f^*(\|\mathbf{x}\|) \ . \tag{5}$$

Using this notation and standard results from Fourier theory, we can represent elements of $\mathcal{R}$ by their Hankel transform (Dautray and Lions, 1988)

$$f(\mathbf{x}) = C\|\mathbf{x}\|^{-\frac{d}{2}+1} \int ds \ \tilde{f}^*(s) s^{\frac{d}{2}} J_{\frac{d}{2}-1}(s\|\mathbf{x}\|) \tag{6}$$

where C is a known number, $J_{\frac{d}{2}-1}$ is a Bessel function of the first kind (Gradshtein and Ryzhik, 1981) and $\tilde{f}^*(s)$ is defined by $\tilde{f}(\mathbf{s}) \equiv \tilde{f}^*(\|\mathbf{s}\|)$. The functional of eq. (4) can now be thought as a functional of $\tilde{f}^*(s)$, and the solution of the minimization problem can be found by imposing the stationarity condition $\frac{\delta H[f]}{\delta \tilde{f}^*(s)} = 0$. After some lengthy calculations it is found that the solution of the approximation problem can be written in the following form:

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i H(\|\mathbf{x}\|, \|\mathbf{x}_i\|) \tag{7}$$

where we have defined

$$H(\|\mathbf{x}\|, \|\mathbf{x}_i\|) = (\|\mathbf{x}\|\|\mathbf{x}_i\|)^{-\frac{d}{2}+1} \int ds \tilde{G}^*(s) s J_{\frac{d}{2}-1}(s\|\mathbf{x}\|) J_{\frac{d}{2}-1}(s\|\mathbf{x}_i\|) \tag{8}$$

Although the basis function H does not have a friendly look, notice the similarity of the solution (7) with the standard solution (2). In both cases the final approximating function is a linear superposition of basis functions, and there is one basis function for each data point. From the computational point of view in both cases the coefficients c_i are found by solving a linear system, with the only difference that in the case (7) the matrix $(G)_{ij}$ of eq. (3) is replaced by the matrix $(H)_{ij} = H(\|\mathbf{x}_i\|, \|\mathbf{x}_j\|)$. However, while it is clear that the standard solution is obtained by placing a "bump" function at each data point, this interpretation is not evident from the solution (7). As the following example shows, a very similar thing happens indeed, and this will become clearer in the next section, when we will discuss the creation of "virtual" examples.

Example Let us consider the very common case in which the basis function $G(\mathbf{x})$ is Gaussian. In this case its Fourier transform is also a Gaussian, and therefore $\tilde{G}^*(s) = \exp(-s^2)$. The integral of eq. (8) can be performed (Gradshtein and Ryzhik, 1981), to obtain the following form for H:

$$H(\|\mathbf{x}\|, \|\mathbf{x}_i\|) = e^{-(\|\mathbf{x}\|^2 + \|\mathbf{x}_i\|^2)} I_{\frac{d}{2}-1}(2\|\mathbf{x}\|\|\mathbf{x}_i\|) \tag{9}$$

where $I_{\frac{d}{2}-1}$ is the Bessel function of first kind of imaginary argument (Gradshtein and Ryzhik, 1981, par. 8.406). A plot of this function in 2 dimensions

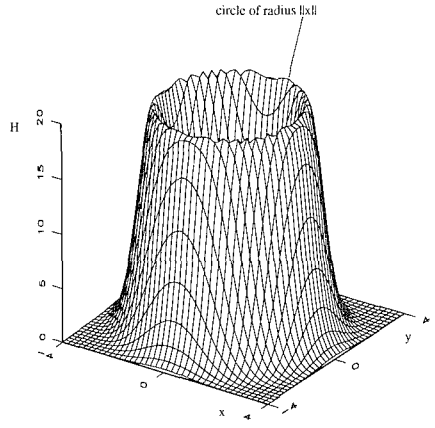


Figure 1: The basis function $H(\|\mathbf{x}\|, \|\mathbf{x}_i\|)$, for $\mathbf{x}_i = (2, 0)$.

is presented in figure (1), where we have set $\|\mathbf{x}_i\| = 2$. It is clear that this function is a radial "bump" function, whose bump is concentrated on a circle of radius $\|\mathbf{x}_i\|$. Any radial section of this function looks like a Gaussian function centered at $\|\mathbf{x}_i\|$, providing a local, radially symmetric, form of approximation.

4 Radial Symmetry and "Virtual" Examples

In this section we follow more closely the approach originally suggested by Abu-Mostafa, and use the prior knowledge to generate new, "virtual" examples, from the existing data set.

Let $D = \{(\mathbf{x}_i, y_i) \in R^d \times R\}_{i=1}^N$ be our data set, and let us assume that we know that the function h underlying the data has radial symmetry. This means that $f(\mathbf{x}) = f(R_\theta \mathbf{x})$ for all the possible rotation matrices R_θ in d dimensions. Here θ is a $d-1$ dimensional vector of parameters that represents a point of Σ_{d-1}, the surface of the d-dimensional unit sphere. This property implies that if $(\mathbf{x}_i, y_i)$ is an example of h, the points $(R_\theta \mathbf{x}_i, y_i)$, for all $\theta \in \Sigma_{d-1}$, are also examples of h, and we call these additional points the "virtual" examples.

Let us now consider a standard Radial Basis Functions approximation technique, of the form (2). Suppose for the moment that the function is invariant with respect to a *finite* number of rotations $R_{\theta_1}, \ldots, R_{\theta_n}$. Each example $\mathbf{x}_i$ will therefore generate n virtual examples $R_{\theta_1}\mathbf{x}_i, \ldots R_{\theta_n}\mathbf{x}_i$, that can now be included in the expansion (2) together with the regular examples.

It is trivial to see that, because of the invariance property of h, the coefficients of the basis functions corresponding to the virtual examples will be equal to the coefficients of the corresponding, original example. As a result we have that eq. (2) has to be replaced by

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i \sum_{\alpha=0}^{n} G(\mathbf{x} - R_{\theta_\alpha}\mathbf{x}_i)$$

where we have defined $\theta_0 = 0$, so that $R_{\theta_0}\mathbf{x}_i = \mathbf{x}_i$. We now relax the assumption that the function is invariant with respect to only a finite number of rotations, and allow θ to span the entire surface Σ_{d-1}. The equation above suggests to replace eq. (2) with the following:

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i \int_{\Sigma_{d-1}} d\Omega_{d-1}(\theta)\ G(\mathbf{x} - R_\theta \mathbf{x}_i) \tag{10}$$

where $d\Omega_{d-1}(\theta)$ is the uniform measure over Σ_{d-1}. Using the Hankel representation (6) for the radial function G in eq. (10), the integral over Σ_{d-1} can be performed, and provides the result:

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i H(\|\mathbf{x}\|, \|\mathbf{x}_i\|)$$

where $H(\|\mathbf{x}\|, \|\mathbf{x}_i\|)$ is given precisely be expression (8)! From this derivation it is clear that the basis function $H(\|\mathbf{x}\|, \|\mathbf{x}_i\|)$ is an infinite superposition of Gaussian functions, whose centers uniformly cover the surface of the sphere of radius $\|\mathbf{x}_i\|$.

Therefore creating virtual examples seems to be, in a sense, the "right thing" to do, leading to the same result that one gets from the more "principled" and sophisticated approach of regularization theory. The appealing feature of the virtual examples technique is the fact that it can be applied in very general cases, in which it might be impossible to derive analytical results as the one derived in section 3.

5 A Simple Experiment

A natural question to ask is how much improvement can be expected by augmenting the data set with "virtual" examples. A similar question would be to ask, as considered by Abu-Mostafa (1993), what is the VC-dimension of the approximation technique (7) once the prior knowledge is included. The case we consider seems to be already complicated enough, since an infinite number of virtual examples can be generated by each example, but this is clearly not equivalent to have an infinite number of examples, because the virtual examples lie on a sphere and are not randomly distributed. This is clearly a difficult problem, and should be studied theoretically and experimentally.

We present here just one very simple example of how the technique works and what kind of results we are looking for.

We consider a 2-dimensional case, in which the function to be approximated is $h(\mathbf{x}) = \cos(||\mathbf{x}||^2)$. We used the standard Gaussian RBF technique, and the corresponding technique (see eq. 7 and 9) that makes use of the prior knowledge. An increasing number of examples, from 16 to 225, has been generated by sampling the function h on a grid in the square $[-1, 1]^2$. The generalization error of the two techniques, computed on a test set of 400 data points also on a regular grid, has been plotted, in a logarithmic scale, as a function of the number of examples (figure 2a). Notice how, in particular for small number of data points, the two techniques differ in generalization error of orders of magnitude. Notice also that when the number of data points becomes large the lower curve, the one computed using prior knowledge, becomes extremely flat: we attribute this to errors encountered in the numerical evaluation of the Bessel function I_0 involved in the computation, that reaches values of the order of 10^{150}, and would need a more careful analysis. Although we have not addressed this problem yet, we do not foresee major complications in doing it, and it should not become a limitation of this technique.

In order to get a feeling of how much is gained by the use of virtual examples we computed how many examples are needed for the two techniques to achieve the same generalization errors. Let $E_2(N)$ and $E_1(N)$ be the generalization errors of the standard Radial Basis Functions technique and the one with prior knowledge respectively. The equation

$$E_2(N^*) = E_1(N) \tag{11}$$

implicitly define N^* as a function of N: N^* is the number of data points that are needed, with no prior knowledge, to achieve the same accuracy achieved by N data points *with* the prior knowledge. Using linear interpolation to approximate $E_1(N)$ we evaluated N^* as a function of N at a number of points and reported the result in figure 1b. For example, it takes approximately 45 data point with the prior knowledge to obtain the generalization error achieved by 225 data points without the prior knowledge. Since 225/45 = 5, we could interpret this result saying that each example generates an "effective" number of virtual examples equal to 4 (5-1).

6 Remarks

We conclude the paper with few remarks:

• One could wonder how often in practice one has to approximate functions with radial symmetry. Probably not often, but one of the goals of this paper was to give an analytical treatment of the problem of creating virtual examples, and set the basis for further results, of more immediate application. Although it has been already proved that creating virtual examples

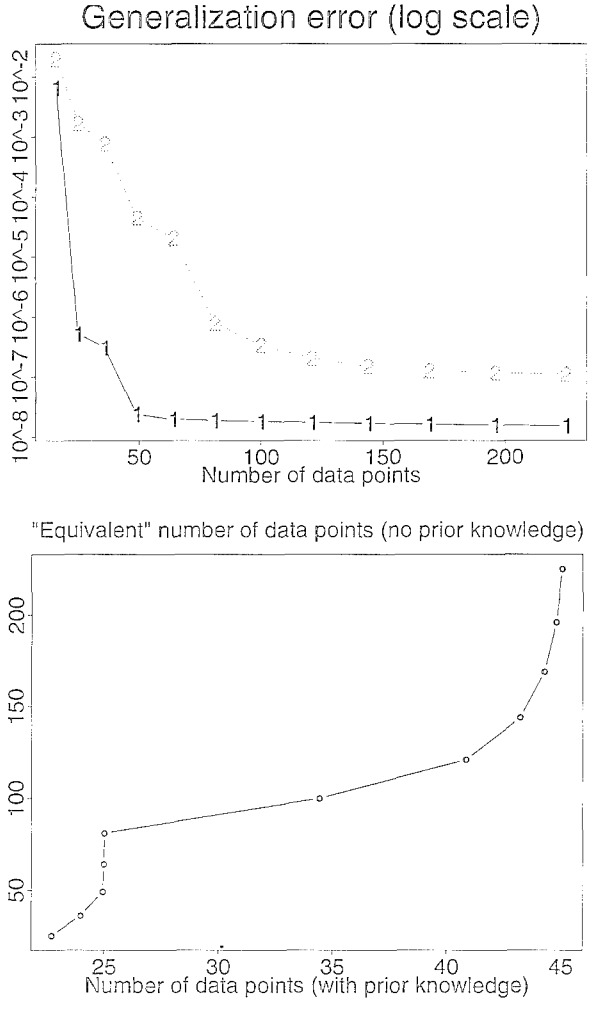


Figure 2: (a) The generalization error of the standard RBF technique (the 2s) and the modified RBF technique (the 1s), that assumes radial symmetry from the beginning(eq. 7); (b) On the vertical axis we plot the number of data points N^* necessary to achieve, with the standard Radial Basis Functions technique, the same generalization error achieved by the modified RBF technique with the number of examples N shown on the horizontal axis (see eq. 11).

reduces the VC-dimension of an approximation technique, it is not obvious that this is strictly equivalent to including the prior knowledge as a constraint on the class of "admissible" functions, that also reduces the VC-dimension of the learning scheme. We plan to consider more complicated cases, of more immediate application. Many real world problems are characterized by invariance properties (Poggio and Vetter, 1992): faces are (approximately) mirror symmetric objects, handwritten characters maintain their identities if their images are rotated, scaled or translated in the image plane. Although these transformations are more complex they all have the property of being a group, and we plan to exploit this property in more details.

• We should notice that in the case considered here we used the prior knowledge as a "hard constraint", and force the solution to have a specific symmetry property, but we could also consider the case in which we only favor solutions with that property. This can be done in regularization theory by considering, instead of the minimization problem (4) the following:

$$\min_{f \in \Phi} \sum_{i=1}^{N} (f(\mathbf{x}_i) - y_i)^2 + \lambda \phi[f] + \alpha \psi[f] \tag{12}$$

where $\psi[f]$ is a functional that penalizes functions that are not radially symmetric. This case has been considered by N. T. Chan (1995) in the one-dimensional case, that is in the case in which the function is known to be even. He considered the choices:

$$\begin{aligned} \psi[f] &= \int_{-\infty}^{+\infty} dx \; (f(x) - f(-x))^2 \\ \psi[f] &= \sum_{i=1}^{N} (f(x_i) - f(-x_i))^2 \end{aligned}$$

The last case is a weaker form of prior knowledge, because the constraint is enforced only at the data points, but in both cases the solution is of the form:

$$f(x) = \sum_{i=1}^{N} c_i G(x - x_i) + \sum_{i=1}^{N} b_i G(x + x_i)$$

and therefore consistent with the creation, for each example (x_i, y_i), of a virtual example $(-x_i, y_i)$.

• Since radial functions are, after all, one-dimensional functions, one could ask why not to use eq. 5 to state the problem in one dimension and use standard regularization theory in one dimension. The reason is that we are ultimately interested in computing a d-dimensional functional, whose smoothness properties are very different from the smoothness properties of the corresponding one-dimensional function defined by eq. 5. Moreover, formulating the problem in one-dimension would be useless in the case in which we want to enforce only a "soft" constraint as in the case of the functional of eq. 12.

◦ It should also be noticed that it has been possible to derive analytical results because in the Radial Basis Functions technique it is clear the role played by the data points, since they explicitly appear in the approximating function. The same kind of analysis is not possible if the approximation technique is substituted with Multilayer Perceptrons, or some other nonlinear approximation technique in which the dependence of the solution on the locations of the data points is much more involved and not known analytically.

References

[1] Y.S. Abu-Mostafa. Hints and the VC-dimension. *Neural Computation*, 5:278–288, 1993.

[2] N.T. Chan. A priori knowledge and the complexity of learning from examples. Master's thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA, May 1995.

[3] R. Dautray and J.L. Lions. *Mathematical Analysis and Numerical Methods for Science and Technology*, volume 2. Springer-Verlag, Berlin, 1988.

[4] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.

[5] I.S. Gradshteyn and I.M. Ryzhik. *Table of integrals, series, and products.* Academic Press, New York, 1981.

[6] C. A. Micchelli. Interpolation of scattered data: distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986.

[7] J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[8] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), September 1990.

[9] T. Poggio and T. Vetter. Recognition and structure from one 2D model view: observations on prototypes, object classes and symmetries. A.I. Memo No. 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[10] A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Math. Dokl.*, 4:1035–1038, 1963.

[11] V. N. Vapnik. *Estimation of Dependences Based on Empirical Data.* Springer-Verlag, Berlin, 1982.

[12] G. Wahba. *Splines Models for Observational Data.* Series in Applied Mathematics, Vol. 59, SIAM, Philadelphia, 1990.

Speech Processing

SPEAKER VERIFICATION USING PHONEME-BASED NEURAL TREE NETWORKS AND PHONETIC WEIGHTING SCORING METHOD

*Han-Sheng Liou**

Kurzweil Applied
Intelligence Inc.
Waltham, MA 02154

Richard J. Mammone

CAIP Center,
Rutgers University,
Piscataway, NJ 08855

ABSTRACT

A text-dependent speaker verification system based on Neural Tree Network (NTN) phoneme model and phonetic weighting scoring method is presented. The system uses a set of concatenated NTNs trained on phonemes to model a password. In contrast to the conventional stochastic approaches which model the phonemes by Hidden Markov Models (HMMs), the new approach utilizes the discriminative training scheme to train a NTN for each phoneme. The phoneme-based NTN is trained to discriminate the phoneme spoken by the speaker with respect to those spoken by other speakers. A weighted scoring method depending on the phoneme's ability for speaker verification is used to improve the performance. The proposed system is evaluated by experiments on the YOHO database. Performance improvements are obtained over conventional techniques.

1. INTRODUCTION

Recently, the speaker verification systems based on characterizing a speaker's password as a sequence of concatenating subword units represented by Hidden Markov Models (HMMs) has been investigated [11, 9]. The subword based model was shown effective in speaker verification tasks for password of connected digits or a randomly prompted sentence. In the previous study [7], we have presented a neural-network-based algorithm for text-dependent speaker verification. In contrast to using HMMs, the algorithm uses a set of concatenated Neural Tree Networks (NTNs) trained on subword units to model a password. The discriminative training is performed in each phoneme model. Thus each phoneme is modeled for the speaker with respect to other speakers.

* This work was done when the author was with CAIP center Rutgers University.

0-7803-2739-X/95 $4.00 © 1995 IEEE

The advantage of using subword models are three fold. First, using subword models can broaden the application of a speaker verification system by allowing unrestricted passwords. Second, in a subword based speaker verification system, the discriminative training can be performed on each subword unit. The differences between two speaker can be discriminated only when their utterances are time aligned. When the speech waveforms corresponding to the same context are aligned, the different way of pronouncing phonemes by two speakers can be differentiated. Third, a scoring method using phonetic weighting can be applied to the subword models. The phonemes which are reliable for speaker discrimination should be emphasized, and those which are more confusing should be suppressed.

In this paper, the phonetic weighting scoring method that combines the confidence measures come out of the phoneme-based NTN is described. The phonetic weights are chosen to refelect the the phoneme's effectiveness in speaker discrimination. To evaluate the performance of this algorithm, we conducted text-dependent speaker verification experiments using the YOHO database. Experimental results show that the proposed hybrid method can achieve better performance than that obtained by HMM classifier.

2. PHONEME-BASED NEURAL TREE NETWORK

2.1. Neural Tree Network

The neural tree network (NTN) [12] is a tree-structured classifier that combines the properties of the feed-forward neural networks [8] and decision trees [1]. The structure of the NTN classifier is similar to a decision tree. Decision tree uses a threshold based on one feature dimension in each node to discriminate feature vectors. In the NTN, the discrimination at each node is implemented by a neuron that can be trained to have the minimum classification error. It has been applied to speaker verification [3], and shown to achieve better performance over conventional methods, such as Vector Quantization (VQ), and Multilayer Perceptron (MLP). In this system, each speaker is modeled by a binary NTN which is trained by the feature vectors of that speaker and the all the other speakers. During training, the feature vectors of the speaker are labeled '1', and those of the other speakers are labeled '0'. The NTN is recursively trained in the following way. Given a set of training data at a particular node, the neuron is trained to split the feature vectors into two subsets that minimizes the classification error. These subsets are subsequently passed to children of the node. This algorithm recurrently proceeds until the subset contains the feature vectors of the same class, or the growth to the prespecified level is reached. The leave at the terminal nodes are labeled by the majority class, and the confidence measure of each leaf is also computed.

2.2. Scoring Method of NTN Model

The NTN is a non-parametric model of the probability distribution of feature vectors. It uses a number of tree structured hyperplanes to partition the feature space into non-overlapping subspaces. In each leaf, a discrete posterior probability, referred to as *confidence*, is derived as follows [6]. The probability of class C_i in a leaf l_j can be approximated by the Parzen density estimate formula [10]:

$$p(\mathbf{x}_j|C_i) = \frac{k_{ij}}{N_i}(\frac{1}{V_j}), \tag{1}$$

where k_{ij} is the number of samples of class C_i in the leaf l_j, V_j is the volume of the region enclosed by leaf l_j, which can be canceled out later, and N_i is the total number of samples of class C_i. The prior probability of a class C_i is defined as:

$$p(C_i) = \frac{N_i}{\sum_{l=1}^{M} N_l}, \tag{2}$$

where M is the total number of class. Given a vector $\mathbf{x}_j$ in a leaf l_j, the posterior probability that the vector belongs to class C_i is defined as

$$p(C_i|\mathbf{x}_j) = \frac{p(C_i)p(\mathbf{x}_j|C_i)}{\sum_{l=1}^{M} p(C_i)p(\mathbf{x}_j|C_l)}. \tag{3}$$

Canceling the common terms, the posterior probability can be simplified to

$$p(C_i|\mathbf{x}_j) = \frac{k_{ij}}{\sum_{l=1}^{M} k_{ij}}. \tag{4}$$

The NTN score of an utterance is defined as the average confidence over the whole utterance.

2.3. Training and Testing of Phoneme-based NTN

The phoneme-based NTN differs from the multiple-word NTN in the sense that they use a different training data set. Instead of using all the words of training data to train a large NTN, the new training algorithm only takes vectors assigned to particular subword units in the training speech to train a phoneme-based NTN. In training of a multi-word NTN (as that trained for text-independent task), the clusters of feature vectors corresponding to the phonemes overlap with each other severely. In NTN training, the classification errors in higher layer may interfere the training result of lower layer. However, the overlaps of the phoneme clusters which are difficult to be separated in feature domain can be separated more easily in time domain. In our previous study, it is shown that training the NTN with homogeneous speech data instead of using all the data improves the verification performance [7].

To train the phoneme-based NTNs for a speaker, a speaker-independent phoneme-based HMMs are first trained to perform speech segmentation. The

parameters of a set of HMMs are initially given by the bootstrap models, then re-estimated by the training utterances in the YOHO database using the Baum-Welch algorithm. For each training utterance, a composite model is synthesized by concatenating phoneme models given by the transcription. After the re-estimation, all the utterances are segmented into subwords and labeled by a Viterbi decoding technique based on the composite models. A speaker-specific phoneme-based NTN is trained for each phoneme using the subword tokens labeled as this phoneme. The NTN trained for this phoneme can provide the ability to discriminate between the speaker and impostors.

During testing, the utterances are first segmented by the concatenation of the subword models given by the prompted password. The subword units are then applied to the corresponding phoneme-based NTNs. The scores are calculated by equation (4), which is described on above. The speaker verification systems using phoneme model have the advantage that the testing passwords are not restricted to fixed passwords. Hence, the security of the system is enhanced.

3. PHONETIC WEIGHTING SCORING METHOD

In conventional scoring methods of speaker verification, the final score is computed by averaging the score of feature frames over the whole utterance. It is assumed that the feature vectors of all the phonemes contain equal abilities of discriminating speakers, so a score averaged over the whole utterance is used to represent the speaker's characteristics. However, the effectiveness of discriminating against the other speakers by each phoneme might be different. Since the vocal tract adopts widely different articulatory configurations during the production of vowels, fricatives, plosives and nasals, the average feature does not represent speaker's characteristics accurately [13]. In Savic's study, a text-independent speaker verification system based on adaptive vocal tract model was proposed. The speech was separately classified into speech segments representing each broad phonetic category as belonging to the impostors or as belonging to the true speaker. A conclusion was drawn that better performance can be achieved by representing each phonetic category by a different model, and by making the final verification decision based on a weighted combination of of scores for individual categories.

In another study [14], a text-independent speaker verification system using two-stage classifiers is described. The first stage consists of a speaker-independent phoneme detector trained to recognize a phoneme that is most effective for speaker verification. The classifier in the second stage is trained to recognize the frames of speech from the target speaker that are admitted by the phoneme detector. The vowel /i/ is found to be the most effective phoneme in characterizing speakers. Improvement was reported in a speaker verification experiment that only considers the phoneme /i/ in the utterances.

A new approach that applies phonetic weighting in calculating the subword NTN score is proposed. As illustrated in Figure 1, the score come out of

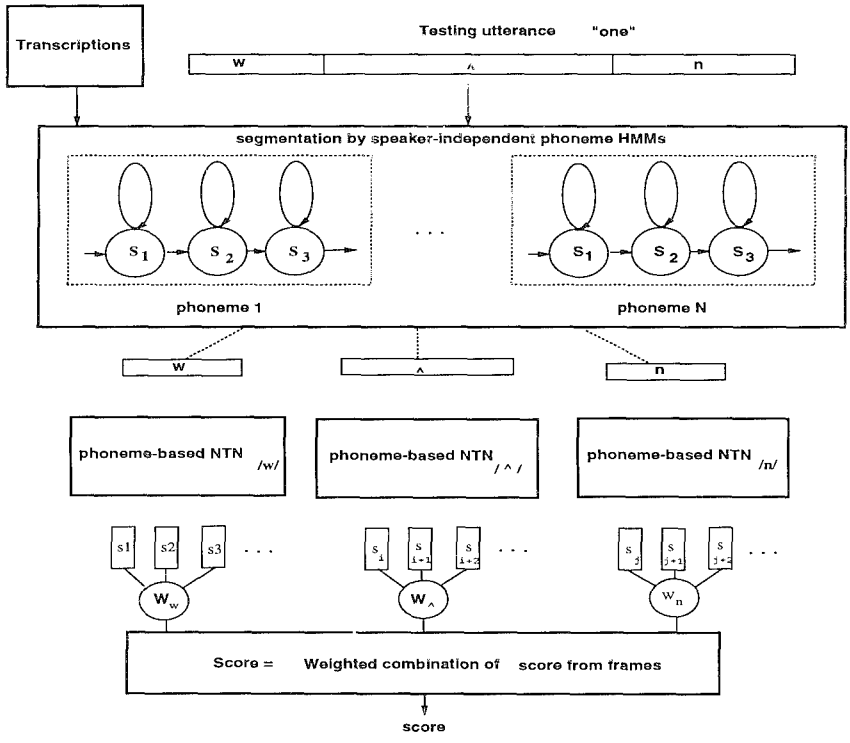


Figure 1: Combining the frame scores with a set of phonetic weighting parameters.

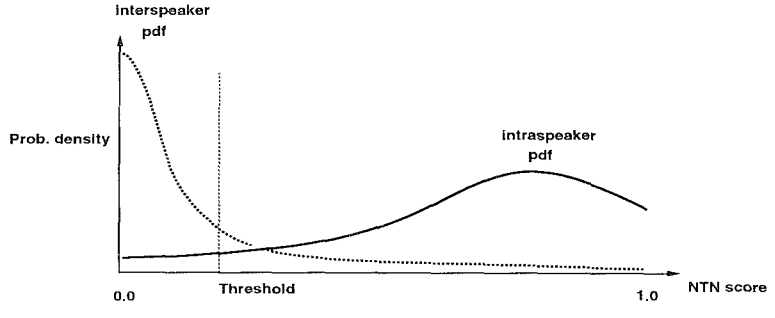


Figure 2: Probability density functions of intra- and inter-speaker NTN score

Rank	Phoneme	Example Word	F-ratio
1	eh	seven	7.65
2	er	thirty	7.02
3	ay	five	6.98
4	ao	four	5.83
5	uw	two	5.25
6	ah	cup	4.96
7	ih	fifty	4.53
8	ax	seven	4.12
9	ey	eight	3.72
10	n	one	3.42
11	r	four	3.27
12	iy	three	3.15
13	w	one	2.84
14	v	five	2.02
15	dx	forty	1.86
16	s	seven	1.78
17	k	key	1.36
18	f	five	0.66
19	th	three	0.65
20	t	two	0.35

Table 1: Average F-ratios of 20 phonemes over 30 male speakers

each subword NTN is multiplied by a phonetic weight which the speech frame corresponds to. The phonetic weights are chosen to reflect the phoneme's effectiveness in discriminating speakers, which depends on the its ability to make correct decision in speaker verification. The classification performances are directly related to the distribution of intra- and inter-speaker scores measured by the phoneme-based NTNs. The more likely a phoneme-based NTN makes classification errors the less discriminative it is. In other words, a phoneme is more effective in discriminating speakers if the phonemes spoken by other speakers are distributed at widely different locations from the speaker's phonemes in the feature space. For example, in two Gaussian distributed pdfs, as shown in Figure 2, the error classification probability is determined by the amount of overlap in the pdf of interspeaker and intraspeaker distance.

The results of the previously stated studies show that the vowels are generally more useful for speaker verification than plosives, because the classification error rate caused by plosives is larger than that caused by vowels. One of the more effective measurement is the ratio of interspeaker to intraspeaker variances, often referred to as F-ratio [15]. More specifically, the F-ratio of

the NTN score for speaker i is defined as [4]

$$F_i = \frac{(\mu_{ii} - \mu_{ij,i\neq j})^2}{\sigma_{ii}^2 + \sigma_{ij,i\neq j}^2} \tag{5}$$

where μ_{ii} and σ_{ii} are the mean and standard deviation of the intraspeaker scores, and the $\mu_{ij,i\neq j}$ and $\sigma_{ij,i\neq j}$ are those of interspeaker scores, respectively. The F-ratio is used to measure the difference of two means normalized by the averaged variance, which also reflects the ability to discriminate speakers of NTN scores. In order to measure the discriminative ability of a phoneme, the frame scores corresponding to this phoneme are used to compute the F-ratio. Table 1 shows the average F-ratio of 20 phonemes over 30 speakers. In general, the vowels are considered having more discriminative ability than the plosives.

It is a common experience that some speakers are easier to be recognized because their pronunciation of certain phonemes are particularly different from others. The weighting vectors are determined so as to emphasize the phonemes which are reliable for speaker verification. As we have discussed, the vowels are demonstrated to be the most speaker discriminative phonemes, and the plosives are the least. Therefore, the feature frames segmented into vowels are weighted with higher values, and the plosives are weighted with lower values.

In the research of speaker recognition, using a weighted cepstral distance measure has been proven effective in improving recognition performance. Most speaker and speech recognition systems use weights in the feature domain [5]. This new approach applies the weighting in phoneme level, which emphasizes theose phones that are effective in speaker discrimination.

Generally, two weighting schemes can be used in speaker recognition systems, which are speaker-dependent and speaker-independent weighting schemes. The speaker-independent weighting scheme is used to determine a general set of weights that applies to all of the speakers. The speaker-dependent weighting scheme is to adapt the weights from speaker to speaker. In speaker identification, the speaker of the testing utterance is unknown to the system, so it usually uses speaker-independent phonetic weighting. However, in speaker verification, a claim to be a particular speaker was given in advance, so a speaker-dependent weighting can be used.

4. EXPERIMENTAL RESULTS

The phoneme-based NTN approach and phonetic weighting scoring method were evaluated by using the YOHO database [2]. The YOHO voice verification corpus was designed particularly for testing text-dependent speaker verification or identification systems. It consists of 138 speakers enrolled (106 males, and 32 females); for each speaker, there are 4 enrollment sessions of 24 utterances each, and 10 verification sessions of 4 utterances each. The testing

subjects spanned a wide range of ages, job descriptions, and educational backgrounds. The speech is acquired by a high quality telephone handset, but not through a telephone channel in a real-world office environment. The syntax used in the YOHO database incorporates "combination lock" phrases, and the phrases used for enrollment and verification are different. The utterances in YOHO are sampled at a rate of 8 kHz, and limited to a 3.8 kHz bandwidth.

4.1. Testing Using Phoneme-based NTN Models

The speech signal is pre-emphasized using a first order digital filter with pre-emphasis factor 0.97. The feature vectors are $12th$ order MFCCs extracted from speech signal over a 25-ms window every 10 ms throughout the utterance. In training the HMM and speech segmentation, the $\triangle$ MFCC, $\triangle^2$ MFCC , energy, $\triangle$ energy, and $\triangle^2$ energy are argumented to each feature vector to form a 39 dimensional vector. In training the phoneme-based NTN, and testing, only the $12th$ order MFCCs are used. The PLU is modeled by a 3-state left-to-right HMM with no skip between states. A total of 20 phonemes, as shown in Table 1, were found enough to transcribe the spoken numbers in the YOHO database.

To evaluate the performance of the system working in the real world, the open-set speaker verification experiments should be conducted. In the experiments, only 80 speakers are enrolled in the system and phoneme-based NTNs are trained for these speaker. The other 58 speakers are considered as impostors, which were never seen in the training. Forty-eight utterances (total 120 sec) of each speaker and those of other speakers are used to train speaker-specific phoneme NTNs.

In the testing, each speaker is treated in turn as a claimant. For each claimant, utterances spoken by speakers of the same gender other than the claimant and the claimant's cohorts are selected as impostors. In other words, those who are in the cohort set of each speaker are excluded as impostors, and there is no testing between speakers of different gender. This arrangement is to avoid an optimistic bias, and make the error estimation closer to what may be found in the real world. The models are trained to inhibit the cohort speakers. If the cohort set speakers are not excluded as impostors, they are likely to be rejected.

classifier	Equal Error Rate	
	4 uttrs.(10s)	1 uttr.(2.5s)
phoneme NTN	0.62%	1.21%
HMM	1.75%	2.91%

Table 2: Speaker verification performance on YOHO database (80 speakers, 58 impostors)

The utterances in the same session are used individually or connected together for testing. The average length of each utterance is about 2.5 sec, and the total length of all utterances in a session is about 10 sec. The scores are cohort normalized with a set of the 5 closest cohort speakers, then compared with a global threshold. A decision of acceptance or rejection is made after comparing the cohort normalized score with a global threshold. The results in Table 2 show that the phoneme-based NTN performs better than the HMM both in single and concatenated utterances.

4.2. Experiments of Phonetic Weighting Scoring Method

Several experiments are performed to investigate the significance of speaker discrimination of the subword NTN trained for each phoneme. With enough data for training the phonetic models, the YOHO database can provide statistically significant estimation of the phonetic discrimination ability. The subword NTN are trained as in the last experiments.

Two experiments were conducted to evaluate the effectiveness of using phonetic weighting. Thirty male speakers in YOHO database were enrolled in the system. The subword NTN were trained by the same procedure as above. In testing against a speaker, all the other enrolled speakers are considered as impostors. The average length of each testing utterance is about 10 second. The NTN scores corresponding to each phoneme are multiplied by the phonetic weights. The confidence measure of a utterance is given by the weighted combination of these NTN scores. The results in Table 3 shows that the phonetic weighting method improve the speaker verification performance.

classifier	Equal Error Rate
without weighting	0.18%
Spkr-Dept. weighting	0.15%
Spkr-Indept. weighting	0.13%

Table 3: Speaker verification Performance on YOHO Database (30 speakers)

5. SUMMARY

We have proposed a phonetic weighting scoring method used to compute the scores for the phoneme-based NTN classifiers. The new approach has been evaluated by the tex ıependent speaker verification experiments. Since the phoneme-based NTN classifier is trained with the discriminant error measure for each speaker, it is shown to provide better discriminant ability than the HMM classifier. The phonetic weighting scoring method is also shown effective for combineing the scors come out of phoneme-based NTN classifiers. Performance improvements are obtained over conventional scoring methods.

6. REFERENCES

[1] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees.* Wadsworth international group, Belmont, CA, 1984.

[2] J.P. Campbell. Testing with the yoho cd-rom voice recognition corpus. In *Proceedings ICASSP*, 1995.

[3] K.R. Farrell, R.J. Mammone, and K.T. Assaleh. Speaker recognition using neural networks and conventional classifiers. *IEEE Tran. on Speech and Audio Processing*, 2(1 part II):194 – 205, January 1994.

[4] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Computer Science and Scientific Computing, ed W. Rheinboldt and D. Siewiorek. San Diego:Academic Press, second edition, 1990.

[5] S. Katagiri and B.-H. Juang. Discriminative feature extraction. *Artificial Neural Networks for Speech and Vision, Edited by Richard J. Mammone*, pages 421–430, 1993.

[6] S. Kosonocky. *A continous density neural tree network word spotting system.* Ph.D. thesis, Rutgers University, October 1994.

[7] H.-S. Liou and R.J. Mammone. A sub-word neural tree network approach to text-dependent speaker verification. In *Proceedings of ICASSP, Detroit*, May 1995.

[8] R. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22, April 1987.

[9] T. Matsui and S. Furui. Concatenated phoneme models for text-variable speaker recognition. In *Proceedings ICASSP*, pages II 391 – II 394, 1993.

[10] E. Parzen. On the estimation of a probability density function and the mode. *Ann. Math. Stat.*, 33:1065–1076, 1962.

[11] A.E. Rosenberg, C.-H. Lee, and F.K. Soong. Sub-word unit talker verification using hidden markov models. In *Proceedings ICASSP*, pages 269–272, 1990.

[12] A. Sankar and R. J. Mammone. Growing and pruning nural tree networks. *IEEE Transactions on Computers*, 42(3):221–229, 1993.

[13] M. Savic and S. K. Gupta. Variable paramter speaker verification system based on hidden markov modeling. In *Proceedings ICASSP*, pages 281 – 284, 1990.

[14] M. Savic and J. Sorensen. Phoneme based speaker verification. In *Proceedings ICASSP*, pages II 165 – II 168, 1992.

[15] J. J. Wolf. Effevient acoustic paramters for speaker recognition. *Journal of Acoustic Society of America*, 51(2):2044 – 2055, June 1972.

Scaling Down: Applying Large Vocabulary Hybrid HMM-MLP Methods to Telephone Recognition of Digits and Natural Numbers

Kristine Ma and Nelson Morgan
International Computer Science Institute, Berkeley, California
1947 Center St, Suite 600
Berkeley, CA. 94704
Tel: (510) 642-4274 Fax: (510) 643-7684
{lingling, morgan}@icsi.berkeley.edu

Abstract. The hybrid Hidden Markov Model (HMM) / Neural Network (NN) speech recognition system at the International Computer Science Institute (ICSI) uses a single hidden layer Multi Layer Perceptron (MLP) to compute emission probabilities of HMM states. This phoneme-based recognition approach was developed for large vocabulary size continuous speech recognition. In this paper, however, such a recognition scheme is applied directly to much smaller vocabulary size corpora, such as the Spoken Language Understanding Numbers'93 database and the TI connected digits. We report here on the development of small baseline systems to facilitate all future research experiments, and also on the use of these systems for experiments in context-dependent hybrid HMM-MLP systems.

1 Introduction

In earlier work from a number of speech laboratories (including our own)[1, 4, 8, 9], the Multi Layer Perceptron (MLP) has been used to estimate emission probabilities for Hidden Markov Models (HMMs). The HMMs are used as the underlying statistical model of speech for large vocabulary speech recognition systems. Over the years, these networks have scaled up to 4000 hidden units and over a million free parameters, trained on millions of feature vectors calculated from speech roughly 100 times per second. As we and others have expanded these systems, we have been pleasantly surprised to learn that the rich and redundant training sets have permitted us to train the nets with very few training passes; in fact, when our largest network was initialized with weights to do phonetic classification from an earlier, smaller data set, we sometimes only needed a single pass through the new data. Even without this initialization, we only required a few passes through the data for these larger problems, using a simple online form of back-propagation with a relative entropy error criterion.

0-7803-2739-X/95 $4.00 © 1995 IEEE

However, despite this result, we have found that training for large corpora is still too time-consuming to promote extensive experimentation with feature extraction and other issues in acoustic modeling. Even using fast specialized hardware for this purpose, we often find that the data handling issues alone lead to unavoidable delays in training that frequently add up to a week or more. Therefore, we decided that we needed to work with a task that was much smaller, but was difficult and representative enough of larger tasks that we could generalize lessons learned from the smaller problems. This approach also would require discipline, in that we would want to eschew particularly specialized solutions to the smaller problem that would not scale.

After some pilot studies with a database of isolated digits spoken over the telephone (and originally collected by Bellcore), we decided to use the "Numbers" task that was being developed at the Oregon Graduate Institute by researchers at the Center for Spoken Language Understanding (CSLU). It is a database of natural numbers spoken continuously and naturally over the telephone, and is reasonably difficult. Due to the moderate size of the Numbers'93 task, we were able to perform a wide range of experiments with context-dependent networks, resulting in good improvements for this task over our standard context-independent system. Some experiments of this type were performed previously at ICSI, but few variants were explored due to computational costs for the larger tasks. In addition, in order to compare our small vocabulary size recognition system performance with that of other speech research sites [2, 8], a recognizer was developed on the TI connected digits using the context-independent form of the hybrid HMM-MLP approach.

Since we wished to use the techniques that we developed for large vocabulary recognition for these tasks, interesting questions of downward scaling arose. Perhaps the huge single-hidden-layer MLP structures were only the province of extremely large data sets, and smaller problems might not work well without specialized structures designed with knowledge of the problem. Some of the design choices made over the years for large vocabulary recognition tasks (such as Resource Management and Wall Street Journal) include:

- Cross-validation test with 0.5% improvement threshold
- Lowering of learning rate (initially set to .008) by a factor of 2 when this threshold was not exceeded
- Stopping of training when this threshold was not exceeded for a pass follwing a reduction in learning rate
- Weights initialized from a network trained on TIMIT or NTIMIT
- Use of a single large sigmoidal hidden layer
- Use of an input window of (typically) 9 frames

- Softmax (exponential of the unit's weighted sum normalized by the sum of exponentials for the entire layer) used as the output nonlinearity
- Stochastic gradient descent using a relative entropy error criterion

In this paper we discuss results and methods for connected digits and numbers, keeping the basic design decisions unchanged. We will show that reasonable systems could be developed without any major design choices, largely by scaling down the hidden layer size in the obvious manner.

2 Baseline System

The hybrid HMM-MLP speech recognition system we are using has the same basic structure as described in [1]. A single hidden layer MLP is employed to estimate the posterior phonetic class probabilities, which are then converted, using Bayes' rule, to likelihoods for Viterbi alignment in the HMM framework.

The MLP and its training procedure were as described above. The 9-frame input to the network yields an input layer of 153 units, where each frame consists of 8 RASTA-PLP cepstra, 8 delta cepstra, and 1 delta energy features. All input features for the network are normalized to have zero mean and unit variance. These choices of inputs, network, and training regimen were unchanged from our larger tasks (though we do use more cepstral parameters for very large vocabulary read speech with wider bandwidth). For all tasks, the MLP was scaled down from our usual range of 500-4000 hidden units to 200 hidden units. For context-independent recognition, the output layer has 61 units, corresponding to one unit per phonetic class. This is also what we use for the larger problems, although in the case of digits and numbers many of the phonetic classes will never be found in the training or test sets.

We are currently using a decoder called Y0 [9] that applies the standard synchronous Viterbi algorithm. In our lexicon, single pronunciation word models with repeated states enforcing minimum phonetic durations are used, and they are based on the most likely TIMIT pronunciations. A null grammar is used for the digit recognizers. For the Numbers'93 task, our language model for the pilot experiments is a class bigram derived from the statistics of the training set.

3 Training Procedure

In our experiments, we use Log-RASTA-PLP [5] as our acoustic pre-processor. Each frame of the feature vector represents 25 msec of speech, with 12.5 msec

overlap of consecutive frames. Log-RASTA-PLP was chosen for its robustness to linear spectral distortions in speech signals that are often introduced by communication channels. This is important in particular for the Numbers'93 database because it is a very realistic set of data that was collected over the public-switched telephone network. However, we use some form of RASTA for all of our tasks currently, large or small (except for some cases with artificial read speech recorded over a standard microphone).

The training of the recognition system is bootstrapped from NTIMIT (Network TIMIT), a database collected by transmitting the TIMIT database speech signals over the telephone network. Even though the Numbers'93 corpus is phonetically hand transcribed, from past experiences we found that it is useful to pre-train a neural network from a much larger data set and use it to initiate our task dependent training. The Numbers'93 corpus has about 100,000 frames of training data, while ten times that number are available from NTIMIT.

The first step in the training procedure is to perform a feedforward pass of the data through a pre-trained NTIMIT net, followed by a phonetic time-alignment of the new corpus using the Y0 Viterbi decoder. This process estimates a set of preliminary target label for the training data. A new MLP is then trained on this set of preliminary alignments. Using this MLP, we reestimate a new set of target labels and so forth. For the training to converge, three or four iterations of forced Viterbi alignment are found to be sufficient. Within each iteration, an independent cross-validation set is used to control the learning rate and to decide when to stop the training. As noted previously, the details of the training heuristics are essentially unchanged from the parameter settings that we used for training up systems with vocabularies of 1000 to 20,000 words, with networks that had over a million parameters.

4 The Numbers'93 Context-dependent Experiment

4.1 Database

As noted earlier, the Numbers'93 corpus is a continuous-speech database collected by the CSLU at Oregon Graduate Institute. It consists of numbers spoken spontaneously over telephone lines on the public-switched network. These numbers are extracted from the addresses spoken by the callers of CSLU's Spelled and Spoken Names Corpus [3]. The Numbers'93 database consists of 2167 speech files of spoken numbers produced by 1132 callers. We used 1534 of these utterances for training and development, saving the remaining utterances for final testing purposes. There are 36 words in the

vocabulary: *zero, oh, 1, 2, 3,...,20, 30, 40, 50,...,100, 1000, a, and, dash, hyphen,* and *double.* CSLU has announced that this task, including a much larger collection of spontaneously spoken numbers, will be made publically available.

4.2 Context-Dependent Experiments

For the Numbers'93 database, other than developing a baseline context-independent system, we also experimented with three context-dependent approaches: single state generalized triphone models, single-state triphone models, and multiple state phonetic models with generalized biphones. All three approaches are bootstrapped from a similar baseline system as described in Section 2, except that the generalized triphone and the triphone methods require an MLP with a larger output layer of 90 and 111 units respectively, while the multiple state phonetic model approach utilizes a different connectionist architecture similar to the one described in [4]. The multiple state phonetic model approach expands the single phonetic state model into a three state model—a context-independent middle state, a generalized left-biphone dependent first state, and a generalized right-biphone dependent last state. To support this formulation a connectionist probability estimator consisting of 17 MLPs is used, with 8 nets corresponding to each of the 8 generalized left-biphones, 8 nets for the generalized right-biphones, and 1 net for the context-independent states.

The context-independent MLP is trained as described in Section 3. The major problem encountered with training context-dependent systems is the lack of data for training highly specific phonetic context classes. One solution, adapted from [4], is to initialize the context specific MLP training with weights from a more general context net. Thus, our generalized triphone context-dependent network is trained by bootstrapping from the previously trained context-independent MLP. The left and right broad categories of our generalized triphones are clustered according to the place of articulation in the vocal tract. As noted in [4], since the training is initialized from a context-dependent net, it is important to smooth the context-dependent priors when converting the context-dependent posterior probabilities to likelihoods. Similarly, our context-dependent triphone net and the set of multiple state model MLPs are trained by bootstrapping from the generalized triphone context-dependent MLP. To limit the number of free parameters and training time for the context-dependent nets, only the hidden-to-output weights were trained. No degradation in recognition performance was found in comparison with training all the weights, as determined by a pilot experiment that was done on the generalized triphone context-dependent training.

4.3 Results from Pilot Experiments

To develop the Numbers'93 recognizers, we used a recognition set of 657 utterances for a total of 2426 words. Pilot experimental results on this development set, for unknown and variable length number strings, are summarized in Table 1.

Table 1: Word level recognition error in % on the Numbers'93 development set.

	Sub	Del	Ins	W.Er	S.Er	# Params
Context-Independent (200HU)	5.3	1.8	1.1	8.2	22.1	38.8K
Context-Independent (400HU)	4.4	1.5	1.2	7.2	19.0	77.6K
Generalized-triphone Context	3.4	1.8	0.9	6.1	17.0	48.6K
Triphone Context	3.4	1.8	0.9	6.1	17.4	52.8K
Multi-phonetic-state Context	3.8	1.5	1.3	6.7	17.4	58.2K
Gen-triphone w/ Xword-Context	3.0	1.4	1.8	6.2	17.2	72.0K

These results were achieved with roughly 5 or 6 backpropagation passes through the Numbers'93 database. This is similar to what we had seen with tasks with 1000 word vocabularies and somewhat larger networks. Typically there were 3 iterations of aligning the training set with the Viterbi algorithm, which was a bit higher than was required for the larger tasks.

From this set of preliminary results we learned that:

- Incorporating context dependence did improve performance but the recognition performance on the Numbers'93 development set did not continue to improve when finer contextual units were incorporated. This is probably due to the lack of training data for those context specific examples (even though we smoothed the training with the context-dependent net). One solution is simply to use a more general context classification, or methods such as dynamic multi-level context clustering as proposed in [7].

- The improvement from 8.2% (context-independent) to the 6.1% (generalized triphone context-dependent) word error rate was partly due to the increase in the number of parameters and partly due to incorporating context-dependence.

- Similar to other authors' observation in the TI Digits corpus, the most confusable pair of digits in the Numbers'93 database is *four* and *oh*. Moreover, a significant percentage of the errors in the Numbers'93 database is due to the confusability between utterance strings such as *four ten* vs. *fourteen*, *six eighty* vs. *sixty*, *eleven* vs. *seven*, *fifteen* vs. *fifty* vs. *sixty* vs. *sixteen*, etc. Also, the word *oh* accounts for more than

half of the deletion error rate. We chose not to repair this with a word-specific phone model for *oh* since this solution would not generalize well to the large vocabulary case.

- A class bigram might be over-smoothing the word pair transition probabilities because the corpus was obtained from street addresses and zip codes. To test this, a new language model was obtained by merging the class bigram with a bigram by averaging their corresponding word pair transition probabilities. This improves the context-independent (200HU) word level recognition error rate from 8.2% to 7.5%, and the generalized triphone with cross-word context-dependence model from 6.2% to 5.8%. All subsequent experiments on the Numbers'93 task are performed using this merged language model.

- Our hybrid recognition method seems to scale easily over a wide range of task sizes, and still achieves good recognition performances.

However, all of these conclusions needed to be confirmed on a final test set for which no design decisions or recognition parameter settings (e.g., language scaling) would be altered.

4.4 Results on Final Test Set

Table 2: Word recognition error in % on the Numbers'93 final test set for the context-dependent experiments. This set of results were obtained using the same parameter settings as the development set.

	Sub	Del	Ins	W.Er	S.Er	# Params
Context-Independent (200HU)	7.2	2.3	2.0	11.5	28.6	38.8K
Context-Independent (400HU)	7.6	2.6	1.4	11.6	27.9	77.6K
Generalized-triphone Context	5.7	2.3	1.3	9.2	23.4	48.6K
Gen-triphone w/ Xword-Context	4.6	1.9	1.7	8.1	22.4	72.0K

For the Numbers'93 test set, increasing the size of the context-independent MLP estimator to include context-dependent units yields an error rate reduction of 30%. This is at the cost of an 86% increase on parameter size. On this test set, the improvement appears to be associated with incorporating contextual information since increasing the number of parameters by merely using a larger hidden layer did not improve performance. The final test set provided by CSLU contains 633 sentences, but 249 of these were eliminated from the final testing because they contain ordinal numbers, which did not appear in the training nor the development sets. We feel that a much larger test set may be required in order to have a more precise evaluation of the various systems. Nevertheless, the results from both the development and the test sets indicates that incorporating context-dependence improves

recognition performance significantly. It is likely that these approaches will generalize well to our larger tasks.

5 Evaluation on the TI Connected Digits

Both the Bellcore Isolated Digits and the Numbers'93 database have been used by other speech research laboratories, but the results have not been widely published. In the meantime, we tested our recognizer on the standard TI/NIST Connected-Digits Recognition Task ("TI-Digits") [6]. While this corpus is inherently less realistic since it was recorded in an artificial studio situation with wide bandwidth, it has been used by many well developed systems(e.g., [2, 8]), so that it will be a calibration point for our methods on small vocabulary size tasks. Ever since this corpus was made available in 1984, it has became a quasi standard for benchmarking small vocabulary speaker-independent recognition systems. The error rate has reduced by more than a factor of 5 since the first published results.

Most of the current state of the art TI-Digits recognizer systems use whole word modeling, cross-word context-dependence, gender dependent models, and more elaborate training procedures. Currently, the best reported TI-Digits recognizer uses whole word model with inter-word triphone context dependence, continuous density HMM for acoustic modeling, a training procedure that performs string error rate minimization, and N-best for decoding [2]. This system yields an error rate of 0.24% on word level and 0.72% on string level. A relatively simple hybrid HMM-MLP system that does not model gender dependence nor context dependence (but still use whole-word models) yields an error rate of 0.89% on the word level and 2.51% on the string level [8]. However, when the likelihood estimations from this MLP system are combined with mixture-Gaussian likelihoods, the resulting error rate is 0.59% on the word level and 1.7% on the string level.

For our experiment, a total of 8623 sentences (1,202,889 frames) are used for training and development, while 8700 sentences are used for testing. Similar to other laboratories, the speech data is digitally filtered to telephone bandwidth (300Hz - 3.2kHz) and downsampled to 8kHz. Since we wished to use the techniques that were developed for large vocabulary recognition on the small tasks, no specialized structures designed with knowledge of the problem were used in developing this system. However, to fully take advantage of the tremendous number of training patterns, we used a larger MLP than we had used on Numbers'93, with 1000 hidden units. The recognition result after four iterations of training on our phonemic based context-independent system was 0.9% word error rate and 2.7% string error rate.

This recognition result is comparable to that of the whole-word model based

hybrid system reported in [8] at the expense of using more parameters. Since whole-word modeling is not the objective of ICSI's research on large vocabulary size continuous speech recognition, we are unlikely to develop such an approach for the TI-Digits database. However, incorporating gender dependence and cross-word context dependence should improve the performance of this particular system significantly.

6 Conclusion

In this paper, we reported experimental results on extending ICSI's speech recognition method, developed for large vocabulary size continuous speech recognition, to digits and numbers. The Bellcore isolated digits task was used to calibrate our methods, and then the approach was ported to a task with continuously and spontaneously spoken numbers; both were recorded over the public-switched telephone network. From working with these databases, we have found that training and recognition could be performed in virtually the same way that we have done for our large vocabulary size tasks, and that further specialized knowledge was not required. Due to the moderate size of the Numbers'93 corpus, a series of experiments with context-dependent networks were made possible, resulting in good improvements for this task over our standard context-independent system. Despite its small vocabulary size, the Numbers'93 corpus is difficult due to its inherently high confusibility factor in the vocabularies, spontaneity of the utterances, and the acoustic channel effects introduced by the telephone network. CSLU has announced plans to distribute the Numbers corpus, and so we hope to see how other sites compare on this task in the coming year. In the meantime, we tested our recognizer on the standard TI Connected Digits corpus. The error rate from our baseline system is higher than a number of other dedicated systems reported over the last few years, but we believe that a more comparable result can be obtained if we were to incorporate cross-word context-dependence and gender dependence. Both of these improvements would generalize to larger tasks as well, although for cases with many context-dependent categories we would not tend to simply associate network outputs with categories; however, as we and others have noted in the past, approaches with multiple networks or multiple output layers will work for this purpose.

Nonetheless, the results presented here appear to show quite convincingly that methods that were developed for much larger tasks scaled fairly easily (and without fundamental changes) to completely different and smaller tasks.

References

[1] H. Bourlard and N. Morgan, *Connectionist Speech Recognition. A Hybrid Approach.* The Kluwer International Series in Engineering and Computer Science. VLSI, Computer Architecture, and Digital Signal Processing, Boston, Massachusetts: Kluwer Academic Publishers, 1994.

[2] W. Chou, C.-H. Lee, B.-H. Juang, "Minimum Error Rate Training of Inter-word Context Dependent Acoustic Model Units in Speech Recognition", Proceedings of the International Conference on Spoken Language Processing, Yokohama, Japan, September 1994.

[3] R.A. Cole, M. Fanty, and T. Lander, "Telephone Speech Corpus Development at CSLU," Proceedings of the International Conference on Spoken Language Processing, Yokohama, Japan, September 1994.

[4] Horacio Franco, Michael Cohen, Nelson Morgan, David Rumelhart, and Victor Abrash, "Hybrid Neural Network/Hidden Markov Model Continuous-Speech Recognition," Processing of the International Conference on Spoken Language Processing, pp. 915-918, 1992.

[5] H. Hermansky, N. Morgan, A. Bayya, and P. Kohn, "RASTA-PLP Speech Analysis Technique," Proceedings of the International Conference on Acoustics Speech and Signal Processing, I.121-124, San Francisco, CA, 1992.

[6] R. Leonard, "A Database for Speaker-Independent Digit Recognition," Proceedings of the International Conference on Acoustics Speech and Signal Processing, pp.42.11.1-42.11.4, San Diego, CA, 1984.

[7] D. Lubensky, "Generalized Context-Dependent Phone Modeling Using Artificial Neural Networks," Proceedings of *EUROSPEECH'93*, September 21-23, Berlin, Germany.

[8] D. Lubensky, A.O. Asadi, and J.M. Naik, "Connected Digit Recognition Using Connectionist Probability Estimators and Mixture-Gaussian Densities," Proceedings of the International Conference on Spoken Language Processing, Yokohama, Japan, September 1994.

[9] T. Robinson, L. Almeida, J.M. Boite, H. Bourlard, F. Fallside, M. Hochberg, D. Kershaw, P. Kohn, Y. Konig, N. Morgan, J.P. Neto, S. Renals, M. Saerens, & C. Wooters. (1993). "A Neural Network Based, Speaker Independent, Large Vocabulary, Continuous Speech Recognition System: The WERNICKE Project", Proceedings of *EUROSPEECH'93*, September 21-23, Berlin, Germany.

COMBINING LOCAL PCA AND RADIAL BASIS FUNCTION NETWORKS FOR SPEAKER NORMALIZATION

C. Furlanello, D. Giuliani
Istituto per la Ricerca Scientifica e Tecnologica
Povo (Trento), Italy
phone: +39 461 314 444; fax: +39 461 314 591
email: {furlan,giuliani}@irst.it

ABSTRACT

Complex multidimensional data may naturally require the decomposition of a regression/classification problem over local regions. Moreover, both global and local anisotropy can be present. We propose to address both problems with a flexible neural network structure embedding data quantization and coordinate transformations. The solution is applied in this paper to speaker normalization. The spectral mapping is realized as a weighted superposition of local neural mappings, estimated between subregions of a new speaker acoustic space and that of a reference speaker, combined with global and local space transformations. The local mappings are realized using the *Generalized Resource Allocating Network (GRAN)* model, a general RBF scheme that allows recursive allocation of kernels. The space transformations are based upon projections over the principal components, separately estimated for the global space and for the local subregions of the input and output acoustic spaces.

1 INTRODUCTION

This paper deals with a major problem arising when speech recognition technology is moved from laboratory to "real world". Changed acoustic conditions in the environment and inter-speaker differences introduce a mismatch between training and testing acoustic data, often significantly affecting speech recognition performance. For this reason, relevant research efforts recently have been devoted to both speaker and channel normalization.

A novel solution, based on an extension of the Radial Basis Function (RBF) model, is applied in this paper to speaker normalization, considered as a regression problem between multidimensional acoustic spaces. The aim of the neural network module for speaker normalization is to minimize differences between transformed acoustic data and the data from which a speech recognition system has been modeled. However,

0-7803-2739-X/95 $4.00 © 1995 IEEE

the relationships between acoustic feature spaces of two speakers are complex, non linear, and hardly can be exploited using a single global regression model. The mapping function between the acoustic parameter space of the *new* speaker and that of the *reference* speaker (for whom the speech recognition system is trained) is also called *spectral mapping* [1, 2]. A similar mapping has been synthesized by Multi-Layer Perceptron networks by other researchers [3, 4].

A previous paper introduced *GRAN* networks for speaker normalization [5]. The neural network was applied as a front-end preprocessor for a continuous speech recognition system (speaker-dependent, HMM-based) to normalize the input acoustic data from a new speaker. The experiments were performed on the speaker-dependent portion of the acoustic-phonetic continuous speech corpus APASCI [6]. The mapping was realized with a single, global *GRAN* network trained on 40 sentences: phone recognition error was reduced with an average adaptability ratio of 25% (see later for a definition of this index).

In this paper, the realization of the global neural mapping is proposed as a weighted superposition of local mappings, estimated between pairs of subregions of the input (that of the new speaker) space and the output (that of reference speaker) space, combined with global and local space transformations. These transformations are based upon projection over the Principal Components, separately estimated for the global space and for the local subregions of the input and output acoustic spaces. The resulting architecture is introduced in Section 2, and the network model presented in 3. Task, speech recognizer and set-up of the normalization experiments are described in Section 4, while experimental results and conclusions are given in Sections 5 and 6, respectively.

2 THE ARCHITECTURE

The rationale for the proposed architecture is that complex multidimensional data can naturally require the decomposition of the regression/classification problem over local regions. Moreover, both global and local anisotropy can be present: it is often useful to deal with "standardized" data, thus "sphering the data" is a common procedure in speech processing applications. Here both the decomposition and the anisotropy problem are addressed at the same time with a flexible neural network structure embedding data quantization and coordinate transformations. The overall structure of the architecture is given in the commutative diagram in Figure 1. The idea is that of combining:

1. a factorization of the regression problem with coordinate transformations designed to reduce speaker-dependent data anisotropy;
2. the estimation of local regression models (locally the "best" models) where subregions are defined by a clustering procedure;
3. the estimation of the global mapping as a weighted superposition of the local mappings.

The coordinate transformations are realized as a linear transformation applied to the data after subtracting their mean. This affine linear mapping may not be invertible, especially if we are interested in considering

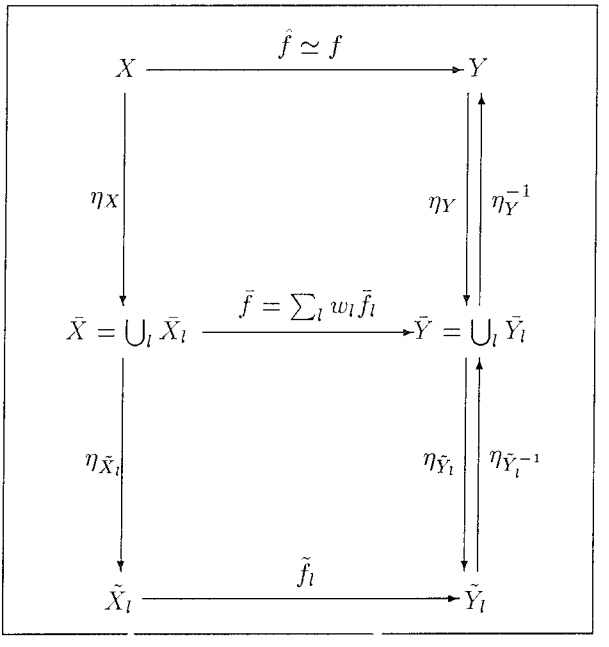


Figure 1

transformation of data into their Principal Components space, and possibly in performing dimensionality reduction. In this case, the global set of data or its subregions can be treated as varieties of lower dimension in the regression problem, but care must be taken in re-embedding the data in their original space, if necessary. This paper considers only invertible coordinate transformations and no dimensionality reduction criterium is applied. Global PCA mappings are first applied to standardize data with respect to *global* anisotropies (see Fig. 1 upper part) of the two speakers. Then, the standardized spaces are subdivided into regions using a clustering algorithm (e.g k-means): for each region, a couple of specific PCA transformations is estimated to reduce *local* anisotropies (Figure 1, lower part). Following the steps detailed in Figure 2, the problem reduces to the estimation of a multivariate regression for each locally standardized subregion.

Any admissible regression scheme can be considered in the realization of the local mappings, and they may actually based on *different* regression schemata. For these mappings, a nonparametric regression smoothing method was applied, the *Generalized Resource Allocating Network (GRAN)* model, a general RBF scheme which allows recursive allocation of kernels. As commented later in this section, the recombination of the local mappings again can be obtained as a realization of an RBF network.

As expected, the selectivity of local basis networks can be exploited specializing learning over L subregions of the acoustic space. Therefore, a superposition of *GRAN* networks, where contribution to the result-

1. *Compute the PCA transformations (mean subtraction included)* $\eta_X : X \to \bar{X}$ *and* $\eta_Y : Y \to \bar{Y}$ *and standardize w.r.t.* global *data anisotropy:*
2. *Partition the standardized spaces* $\bar{X}$ *and* $\bar{Y}$ *into* L *subregions.*
3. *For every pair of local regions* $(\bar{X}_l, \bar{Y}_l)$*, with* $l = 1 \ldots L$*:*
   - 3.1 *Compute the PCA transformations (mean subtraction included)* $\eta_{\bar{X}_l} : \bar{X}_l \to \tilde{X}_l$ *and* $\eta_{\bar{Y}_l} : \bar{Y}_l \to \tilde{Y}_l$ *to standardize w.r.t.* local *data anisotropy;*
   - 3.2 *Estimate the local neural GRAN mapping* $\tilde{f}_l$ *between local standardized spaces* $\tilde{X}_l$ *and* $\tilde{Y}_l$*.*
4. *Estimate the* L *weight functions* $(w_l)_{l=1\ldots L}$*:*
5. *Estimate the global mapping between standardized spaces* $\bar{f} : \bar{X} \to \bar{Y}$ *as a weighted superposition* $\bar{f} = \sum_l w_l \bar{f}_l$ *of the local mappings* $\bar{f}_l = \eta_{\bar{Y}_l}^{-1} \cdot \tilde{f} \cdot \eta_{\bar{X}_l}$*.*
6. *Estimate the global mapping* f *between the original spaces* X *and* Y *as* $f = \eta_Y^{-1} \cdot \bar{f} \cdot \eta_X$*.*

Figure 2

ing model is weighted according to a membership function of the input acoustic data to that subspace, is introduced:

$$\ddot{f}(x) = \sum_{l=0}^{L} w_l(x) \bar{f}_l(x) \qquad \text{where} \qquad w_l(\mathbf{x}) = \frac{k_l(\mathbf{x})}{\sum_r k_r(\mathbf{x})}$$

where $k_l(\mathbf{x})$ measures membership of vector $\mathbf{x}$ to the l-th region in a probabilistic or fuzzy framework. It is a typical choice to model data distribution as a mixture of L Gaussian distributions, each Gaussian modeling a subregion and having distribution parameters initialized from the clustering procedure.

The idea underlying the model is strongly related to the concept of switching regression [7, 8] and to the "mixture of expert" approach [9, 10]. The *Expectation* step in the EM algorithm [7, 9] is an example of optimization of the weights $(w_l)_{l=1\ldots L}$. A similar "softmax" combination of local linear function was realized in [11], with weights estimated by an RBF network, while reference [12] gives indications for a probabilistic treatment of the general case.

Here the joint use of this "architectural" approach is exploited combining local PCA transformations and RBF networks. Although the issue of weight estimation is not the central idea of this paper, we have actually embedded the superposition of the local mappings within a static RBF network: for each local region, a kernel w_l is defined, whose location and bandwidths are initialized by the clustering procedure. Current research regards the problem of different functional forms for the kernel, the use

of statistical information about the characteristics of data distribution within the local regions and that of optimization of weight parameters.

3 THE GENERALIZED RESOURCE ALLOCATING NETWORK MODEL

The RBF network model applied in this paper introduces recursive allocation of elliptical kernels within the Generalized Regularization Network approximation scheme in [13] according to a heuristic procedure of [14]. The *Generalized Resource Allocating Network (GRAN)* model is thus a nonparametric kernel regression method that allows optimization of kernel coefficients and locations, of different local bandwidths for each coordinate. The model includes an affine linear term and a weighted metrics is considered in the input space in the general case. The optimization of the weighted metrics is implemented, as in the generalized versions of RBF networks of [15]. Efficient on-line optimization of parameters is achieved by an automatic differentiation technique, which allows reuse of network evaluation for on-line optimization of parameters with respect to different error functions (L_2, L_1, Entropy fit) with different learning rates for each type of parameter. Implemented kernels are Gauss, Hardy multiquadrics, inverse of Hardy multiquadrics and Epanenchnikov kernels, also in the Nadaraya-Watson ("softmax") normalized form [16].

Classical RBF learning, i.e. kernel placement via clustering and computation of kernel coefficients via SVD estimation is also available. Specific architectural, parameter and learning choices for the speech normalization experiments are discussed in Section 5.

4 THE EXPERIMENTAL SET-UP

4.1 The APASCI Corpus and the Speech Recognizer

Continuous phone recognition experiments were performed on APASCI, an Italian acoustic-phonetic continuous speech corpus [6] containing speech material for speaker independent (3900 utterances from 176 speakers) and speaker dependent (520 utterances for each of 6 speakers) speech recognition. The speaker dependent portion of APASCI consists of speech signals collected from the 6 speakers (3 females and 3 males). For each of them a training set consisting of 400 utterances and a test set of 120 utterances are available. Training and test set were acquired on different days and in multiple acquisition sessions. For each speaker an adaptation set of 60 utterances is also defined as a subset of the training set. On average, the duration of the signals is 5 seconds.

Each signal of the corpus was analyzed and, for each frame, 8 Mel Scaled Cepstral Coefficients (MSCC), the log-energy, their first and second order derivatives were computed obtaining a feature vector of 27 components.

The experiments with the APASCI corpus were carried out with Hidden Markov Model (HMM) recognition systems based on 38 context independent acoustic-phonetic units. Each speech unit was modeled with a left-to-right HMM, with the exception of the HMM modeling the silence for which a single state ergodic topology was adopted. The output prob-

ability distributions were modeled with mixtures of 16 Gaussian probability densities having diagonal covariance matrices. Transitions leaving the same state shared the same output distribution probabilities.

For each of the 6 speakers in the database, an HMM speaker dependent (SD) recognition system was trained with 400 utterances after bootstrapping with sex dependent models using the sex dependent speech material (1000 utterances from 50 male speakers and 1140 utterances from 50 female speakers) available in the speech corpus. For each speaker, the test material consisted of 90 utterances from the speaker dependent test set, corresponding, on average, to 4770 phone units.

4.2 Data Preparation

During experiments a SD recognition system trained for the reference speaker is tested with the test material of new speaker. In order to estimate a feature vector transformation from the acoustic space of the new speaker to that of the reference speaker a set of phrases are uttered by the two speakers. For each phrase, the parametric representations of the signals uttered by the reference speaker and the new speaker are time aligned by means of the Dynamic Time Warping (DTW) algorithm and a set of feature vector pairs are obtained according to the DTW optimal path. In the reported experiments, the parametric representation used for utterance time alignment and spectral mapping consisted of 8 MSCC. Each feature vector pair represents simultaneous observations from the acoustic spaces of the two speakers. The set of feature vector pairs so obtained forms the training set for the speaker normalization module. The utterances used for normalization purpose belongs to the adaptation sets of the two speakers. For details about the time alignment procedure summarized above see [5].

4.3 Performance Measures

In this work, continuous phone recognition experiments were performed without any lexical and phonetical constraint (no phone statistic was used). Recognition results are expressed in terms of the number of insertions (Ins), deletions (Del) and substitutions (Sub) of phone units made by the recognizer. As usual, Unit Accuracy UA and Percent Correct PC performance indicators are defined as $UA = 100\ (1 - (Ins + Del + Sub)/n_{units})$ and $PC = 100\ (1 - (Del + Sub)/n_{units})$, where n_{units} is the total number of units in the test set.

The *adaptability ratios* were also considered as canonical indices of adaptation/normalization performance; let $a = UA$ and $p = PC$, we define the corresponding adaptability ratios as

$$\rho_a = \frac{a^n_{RT} - a_{RT}}{a_{RR} - a_{RT}} \qquad \rho_p = \frac{p^n_{RT} - p_{RT}}{p_{RR} - p_{RT}}$$

where a_{RT} indicate recognition accuracy for reference speaker R and target T without normalization, a_{RR} is the speaker dependent baseline recognition accuracy and superscript n indicates that normalization is performed. The same notation applies for the percent correct adaptability ratio ρ_p.

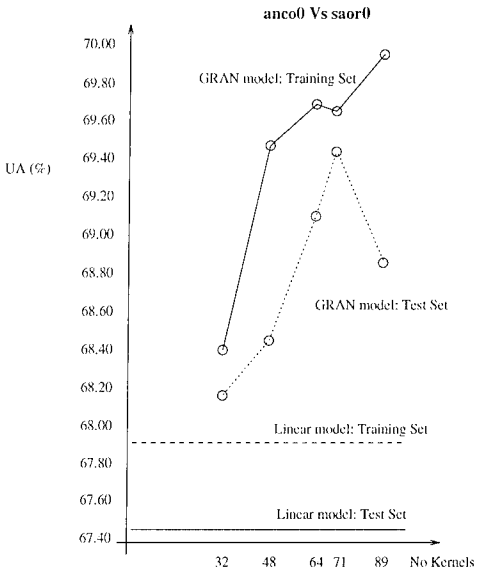


Figure 3

5 EXPERIMENTAL RESULTS

As discussed in the previous Section, the normalization task regarded a multivariate regression problem with 8 inputs and 8 outputs that is difficult for kernel-based methods. The training set for each pair of speakers in the database consisted of 7,000 input-output pairs of vectors on average, but the classical RBF bootstrap from estimation of the kernel locations with a clustering procedure and SVD computation of kernel coefficients did not give encouraging results. It is possible that in this task the relevant variations of the spectral mapping take place in regions of the (input) acoustic space which are not correlated with higher density of data. The recursive allocation of kernels allows local fits to be realized corresponding to data where a significant approximation error is present due to the mapping non-linearity. For this reason, the typical learning procedure was first to fit the linear term to the data, and then to start allocating elliptical Gaussian kernels by fitting the residuals of the linear regression up to maximal number of kernels. The Nadaraya-Watson ("softmax") normalization was considered for each kernel. On-line gradient descent was applied when not allocating a kernel: metrics and offset term were kept fixed, and optimization of kernels regarded center positions, local bandwidths (different for each direction), and coefficients. Moreover, the linear term was re-estimated in this phase and different learning rates were adopted for each parameter classes. The following experiments were performed to test the validity of the model and of the architecture in this application.

The first experiment involved a female reference speaker (*anco0*) and a male new speaker (*saor0*) with a training set of 15 sentences and a test set of 90 sentences (see Subsection 4.1). A single global network was trained for 10 epochs allowing different maximum numbers of ker-

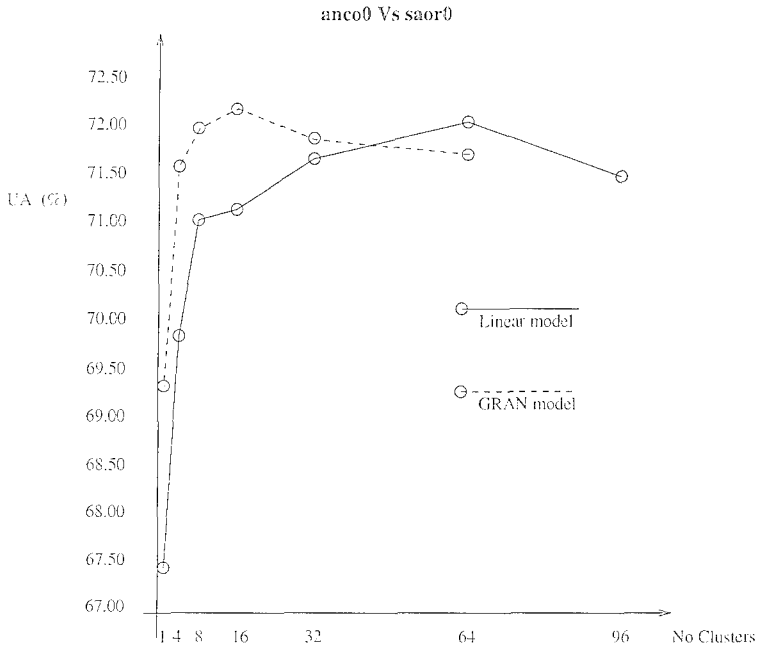


Figure 4

nels and results compared with those of a linear transformation. The recognition results in terms of Unit Accuracy are plotted in Figure 3 with respect to network size. The *GRAN* network outperforms the linear mapping with an increasing recognition rate until an overtraining effect is induced. It is worth noting that, although a maximum upper bound of 96 kernels was allowed, only 89 kernels were allocated. As a comparison, the baseline recognition score for the (*anco0, saor0*) pair of speakers without normalization is $UA = 59.89\%$.

The second experiment was conducted using the same condition as the previous one. Different global models were trained by combination an increasing number of local models. The subregions were obtained by *k-means* clustering. Each local network was trained on data belonging to the corresponding standardized subregions according to the procedure presented in Section 2. Given the results of the previous experiment, the Vapnik-Chervonenkis (VC)'dimension of the problem was controlled by allowing a number of allocable kernels directly proportional to the number of patterns in the local regions in which the regression problem was eventually considered. Again the relative performance of the *GRAN* and linear mappings is compared. Best results were obtained with 16 *GRAN* local mappings (8 kernels each) with a recognition score of $UA = 72.22\%$. With 64 regions the network performance is lower than for the combination of linear mappings. Note that maximum 145 patterns and minimum 10 patterns were found in the 96 cluster case by the clustering algorithm (*k-means* in all these experiments): very small eigenvalues were computed by the local PCA procedure, thus suggesting that lower dimensional local transformations are possible in such situations.

	UA	PC	ρ_a	ρ_p
SD	64.56	70.87	–	–
LIN	69.48	75.33	0.237	0.262
NN	69.78	75.67	0.251	0.282

Table 1

The last experiment was performed as an average performance evaluation over 30 cross-experiments for all the different pairs of $(new, reference)$ speakers. Performance is analyzed in terms of Unit Accuracy, Percent Correct and their adaptability ratios. The average phone recognition rates obtained with normalization modules based on the GRAN model (NN row) are reported in Table 1 and compared with the baseline performance (SD: no adaptation) and those of linear transformations (LIN) in the 4 cluster case. The use of local models allowed an improvement of the UA and PC performance, evidenced by the ρ_a index, comparable with that obtained with a single global $GRAN$ model and 40 sentences, as reported in [5].

6 CONCLUSIONS

The experimental results evidence interesting performance of RBF networks *(GRAN model)* in speaker normalization. The improvement of recognition performance results with the proposed architecture allowed a significative reduction of the adaptation material from 40 to 15 phrases. Further improvements are expected considering different weight functions and their optimization. In particular, it is important to increase the performance with respect to the local linear method. Moreover, the local feature transformation mappings may be exploited to realize local mappings between dimensionality reduction. Finally, the experiments demonstrated once more the importance of controlling the size of the models according to estimates of the VC dimension of the task.

Acknowledgments

This work was developed under a grant of the "Programma Nazionale di Ricerca per la Bioelettronica" assigned by the Italian Ministry of University and Scientific Research to Elsag Bailey. The contributions of G. Lazzari and S. Struthers are gratefully acknowledged.

References

[1] F. Class, A. Kaltenmeier, P. Regel, and K. Troller, "Fast speaker adaptation for speech recognition system," in *Proc. of ICASSP 90*, pp. I–133–136, April 1990.

[2] J. R. Bellegarda, P. V. de Souza, A. J. Nadas, D. Nahamoo, M. A. Picheny, and L. R. Bahl, "Robust speaker adaptation using a piecewise linear acoustic mapping," in *Proc. of ICASSP 92*, pp. 445–448, IEEE, March 1992.

[3] X. D. Huang, "Speaker normalization for speech recognition," in *Proc. of ICASSP 92*, pp. I–465–468, March 1992.

[4] R. Watrous, "Speaker normalization and adaptation using second-order connectionist networks," *IEEE Trans. on Neural Networks*, vol. 4, pp. 21–30, January 1994.

[5] C. Furlanello, D. Giuliani, and E. Trentin, "Connectionist speaker normalization with Generalized Resource Allocating Networks," in *Advances in Neural Information Processing Systems 7* (D. S. T. G. Tesauro and T. K. Leen, eds.), MIT Press, 1995. in press.

[6] B. Angelini, F. Brugnara, D. Falavigna, D. Giuliani, R. Gretter, and M. Omologo, "Speaker Independent Continuous Speech Recognition Using an Acoustic-Phonetic Italian Corpus," in *Proc. of ICSLP*, (Yokohama), September 1994.

[7] N. M. Kiefer, "Discrete parameter variation: Efficient estimation of a switching regression model," *Econometrica*, vol. 46, pp. 427–434, March 1978.

[8] R. J. Hathaway and J. C. Bezdek, "Switching regression models and fuzzy clustering," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 3, pp. 195–204, 1993.

[9] M. Aitkin and G. T. Wilson, "Mixture models, outliers, and the EM algorithm," *Technometrics*, vol. 22, pp. 325–331, August 1980.

[10] R. A. Jacobs and M. I. Jordan, "Learning piecewise control strategies in a modular neural network architecture," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 23, pp. 337–345, March 1993.

[11] Y. S. N. Iwahashi, "Voice adaptation using multi-functional transformation with weighting by radial basis function networks," in *Proc. of ICSLP 94*, pp. III–1599–1602, September 1994.

[12] V. Treps and M. Taniguchi, "Combining estimators using non-constant weighting functions," in *Advances in Neural Information Processing Systems 7* (D. S. T. G. Tesauro and T. K. Leen, eds.), MIT Press, 1995. in press.

[13] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural network architectures," *Neural Computation*, vol. 7, Jan. 1994.

[14] J. Platt, "A resource-allocating network for function interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, 1991.

[15] T. Poggio and F. Girosi, "A theory of networks for approximation and learning," A.I. Memo No. 1140, MIT, 1989.

[16] W. Härdle, *Applied nonparametric regression*, vol. 19 of *Econom. Soc. Monographs*. New York: Cambridge Un. Press, 1990.

DISCRIMINATORY MEASURES FOR SPEAKER RECOGNITION

Kevin R. Farrell
Dictaphone Corporation
3191 Broadbridge Avenue
Stratford, Connecticut, USA 06497
Phone: (203) 381-7210 FAX: (203) 381-4515
email: farrekr@pb.com

Abstract

This paper investigates methods for incorporating discriminatory information into speaker recognition systems. In particular, this information is used to supplement non-discriminative modeling approaches, such as dynamic time warping (DTW) and hidden Markov modeling. The discriminative information is obtained from the neural tree network (NTN) and is integrated with the non-discriminative models via data fusion. Here, the outputs of each model are combined with two data fusion methods know as the linear opinion pool and log opinion pool. These methods are evaluated for text-dependent speaker verification for two databases. For both experiments, the consensus driven system outperformed the systems based on individual models.

INTRODUCTION

The conventional approaches to speech and speaker recognition are generally based on modeling the data for a given class. For example, numerous utterances of a specific word or phoneme can be used to train a hidden Markov model (HMM) for speech recognition. Likewise, numerous utterances from a specific person can be used to train a HMM or vector quantization (VQ) codebook for speaker recognition. Here, the model will capture the acoustic characteristics of that class, such that future utterances of the same word, or from the same speaker, will tend to score well upon that model. However, the information that is typically ignored by these training methods is that of how to discriminate from other classes.

0-7803-2739-X/95 $4.00 © 1995 IEEE

Several methods have been proposed for incorporating discriminatory information into the model design. One method is to use discriminative training [1]. Here, a loss function is used that minimizes the classification error and, hence, incorporates information from other classes. Discriminative training has been used for speech [2] and speaker [3] recognition. Many other approaches for obtaining discriminatory measures are based on hybrid models, i.e., those that integrate a discriminative classifier with a non-discriminative classifier. For example, multilayer perceptrons (MLPs) can be used to estimate the observation probabilities [4] in HMM-based recognition systems. Here, the MLP can provide discriminative information, which is not available from the alternative Gaussian mixture model approach.

This summary explores an alternate means of hybrid model design for incorporating discriminant information into speaker recognition classifiers. The technique used here to combine the models is known as data fusion. Here, the models are trained separately and their respective outputs are combined during testing. The following section provides a brief description of speaker verification along with the modeling approaches that will be considered here. Several methods for combining the scores of these models are then described. The composite systems are evaluated for a text-dependent speaker verification task and the conclusion of this paper is then provided.

1 TEXT-DEPENDENT SPEAKER VERIFICATION

The objective of speaker verification is to verify a person's claimed identity based on an utterance from that person. This is in contrast to speaker identification where a person is to be identified within a population. A distinguishing feature of speaker verification systems regards the form of spoken input, which can be text dependent or text independent. Text-dependent speaker recognition systems require that the speaker utter a specific phrase or a given password. Text-independent speaker identification systems identify the speaker regardless of the utterance. This summary focuses on text-dependent speaker verification.

Several speaker modeling methods are considered in this paper. These are based on the neural tree network (NTN), dynamic time warping (DTW), and hidden Markov models (HMMs). The NTN bases its decision upon discriminant information whereas the DTW and HMM methods utilize a distortion or likelihood measure. Hence, these two classes of methods use criteria that are somewhat complementary. The NTN, DTW, and HMM methods of speaker verification are briefly described as follows.

1.1 NEURAL TREE NETWORK

The NTN [5] is a hierarchical classifier that combines the properties of decision trees and feed-forward neural networks. For speaker recognition, the training data for the NTN consists of data for the target speaker labeled as "one" and data from other speakers labeled as "zero". The NTN partitions feature space into regions that are assigned probabilities which reflect how likely a speaker is to have generated a feature vector that falls within that region. The NTN has been evaluated for text-independent speaker recognition [6] where it was found to perform favorably for open-set problems, such as speaker verification.

There are several approaches for applying the NTN to *text-dependent* speaker recognition. One method is to train the NTN to discriminate between a password spoken by the target speaker and the same password spoken by other speakers. This can be interpreted as a "whole-word" NTN. Another method is to build "sub-word" NTN models [7]. In this case; the password of the target speaker will be segmented into sub-words, i.e., phonemes. A NTN will then be trained for each sub-word unit, where the anti-class data will consist of the data from other speakers' utterances of that sub-word unit. These models can then be concatenated to form passwords. The "whole-word" NTN will be considered here. Note that this method does not utilize the temporal information of the password. This information can be obtained, however, by combining the results of the NTN with a temporal-based model, such as that obtained from the DTW or HMM method.

1.2 DYNAMIC TIME WARPING

The DTW algorithm is a distortion-based approach for time aligning the dynamics of two waveforms. For speaker verification, a reference template can be generated from several utterances of the password [8]. These utterances are combined to form a single reference template using a modified k-means algorithm [9]. Then during testing, a decision can be made to accept or reject the claimed identity based on whether or not the distortion falls below a predetermined threshold. To allow for subsequent fusion with other speaker models, such as NTNs, the DTW distortions must be converted to a compatible scale, i.e., a probability. We accomplish this by simply raising the negative distortion to an exponential.

1.3 HIDDEN MARKOV MODEL

A hidden Markov model (HMM) is a stochastic finite state machine that can be used to model sequences. HMMs are the predominant technology used in speech recognition and have also gained much popularity in text-dependent speaker verification. Here, numerous utterances of a password from the target speaker can be used to train a sub-word [10] or whole-word [11] HMM. During testing, a decision can be made to either accept or reject the claimed identity

based on whether or not the likelihood score exceeds a threshold. HMMs have been shown to outperform DTW [12] for applications where a sufficient amount of training data is available.

2 MODEL COMBINATION METHODS

There are numerous ways to combine the opinions of multiple experts. For example, if the outputs of all experts are probabilities then a simple combination method would be to take a weighted sum of the probabilities or of the logs of the probabilities. These methods are known as the linear opinion pool and log opinion pools [13], respectively. If the outputs of the experts are class labels, then methods such as voting [14] or ranking [15] can be used. For fuzzy decisions, Dempster-Shafer theory can also be used for the combination of experts [14]. This paper evaluates the linear and log opinion pool methods for text-dependent speaker verification. These methods are described in more detail as follows.

2.1 LINEAR OPINION POOL

The linear opinion pool is a commonly used data fusion technique that is convenient due to its simplicity. The linear opinion pool is evaluated as a weighted sum of the outputs for each expert:

$$P_{linear}(x) = \sum_{i=1}^{n} \alpha_i p_i(x), \tag{1}$$

where $P_{linear}(x)$ is the probability of the combined system, α_i are weights, $p_i(x)$ is the probability output by the i^{th} expert, and n is the number of experts. For all experiments in this paper, α is between zero and one and the sum of the α's is equal to one.

The linear opinion pool is appealing in that the output is a probability distribution and the weights α_i provide a rough measure of the expertise of the i^{th} expert. However, it is noted that the probability distribution of the combiner output may be multimodal, which may impose a more complicated decision strategy.

2.2 LOG OPINION POOL

An alternative to the linear opinion pool is the log opinion pool. If the α weights are constrained to lie between zero and one and sum up to one, then the log opinion pool also outputs a probability distribution. However, as opposed to the linear opinion pool, the output distribution of the log opinion pool is unimodal [13].

The log opinion pool consists of a weighted product of the expert outputs:

$$P_{log}(x) = \prod_{i=1}^{n} p_i^{\alpha_i}(x). \quad (2)$$

Note that with this formulation, if any expert assigns a probability of zero, then the combined probability will also be zero. Hence, an individual expert has the capability of a "veto", whereas in the linear opinion pool the zero probability would be averaged in with the other probabilities.

One problem that both the linear and log opinion pools are subject to is the selection of the weights α_i. Several heuristic solutions [13] to this are to 1) use equal weights, i.e., $\alpha_i = 1/n$, 2) use weights proportional to a ranking, i.e., $\alpha_i = r_i / \sum_{r=1}^{n} r$, or 3) evaluate the weights over the range of zero to one for cross-validation data and select the best α_i. For the experiments in this chapter, plots for the third method are provided.

The linear and log opinion pools have been applied to various applications in speech and speaker recognition. The linear opinion pool has been considered in speaker recognition for the combination of features [16], namely cepstrum and delta cepstrum features, and also for the combination of classifiers [17]. Both the linear and log opinion pools were evaluated for speech recognition for the combination of several features [18].

3 EXPERIMENTAL RESULTS

The first database used to evaluate the system was collected over a telephone channel. All of the handsets utilized electret microphones. The speech acquired through the telephone channel is sampled at 8 kHz, and μ-law coded at 8 bits/sample. The speech signal is pre-emphasized with a pre-emphasis factor of 0.95. Features are extracted within 20 millisecond analysis windows having 5 millisecond shifts between consecutive analysis windows. The features extracted from the analysis windows consist of linear prediction (LP)-derived cepstral coefficients.

The system is evaluated with data from 20 male speakers. Fourteen utterances of the state, "New Jersey" are collected from ten of these 20 speakers. Four utterances are used to train each model and the remaining ten are used for testing. For the remaining ten speakers, ten utterances of the state, "New Jersey" are collected and used as imposter utterances for each of the first ten speakers. This database allows for 100 true speaker trials and 1000 imposter trials. Note that a pre-existing database is used to provide the antispeaker data for the NTN training. This antispeaker database contains four utterances of "New Jersey" from 20 male speakers. The 10 imposter speakers used to evaluate the false accept rate are not included in the antispeaker database.

To illustrate the merit in combining a discriminative model with a non-discriminative model, a scatter plot for one speaker is provided in Figure 1. Here, the "x" and "o" labels correspond to the speaker and imposter scores

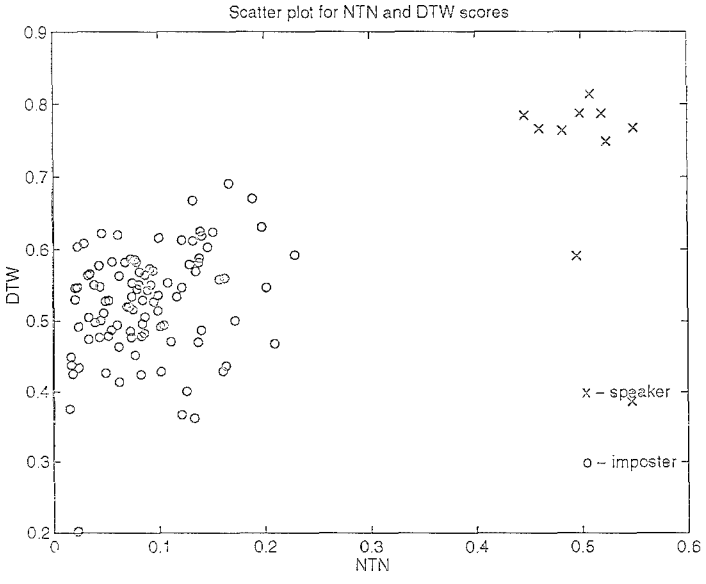


Figure 1: Scatter plot example

from the NTN and DTW models. If only the DTW classifier were used, then all points would be projected onto the y axis. In this case, the data would have some confusion. On the other hand, if the NTN and DTW classifiers are given equal weight, then the data will be projected onto the diagonal coming from the origin (with slope equal to one). Here, the data would not be confused.

The operating curves for the NTN and DTW were evaluated and are shown in Figure 2. It is noted that these curves are based on a posterior positioning of the threshold. Here it is seen that the NTN performs significantly better than DTW, i.e., an equal error rate of 5.2% as compared to 9.8%. However, the equal error rate of the combined system using the linear opinion pool with $\alpha = 0.5$ is 2%, which is better than that of either method used individually. The error rates of the linear and log opinion pools are evaluated as a function of α. The performance of both methods is illustrated in Figure 3.

The next experiment utilized isolated digits from the TI 46 word database. The portion of the database used consisted of 25 utterances of the digits zero through nine for eight male speakers. Of the 25 utterances, the first 10 utterances were used for training, and the remaining 15 utterances were used for testing. In order to keep the experiments unbiased, two tests were evaluated. The first test consisted of training the first four speakers, using eachothers training utterances for "anticlass" data, and then using the last four speakers as imposters. The second test consisted of training the last four speakers

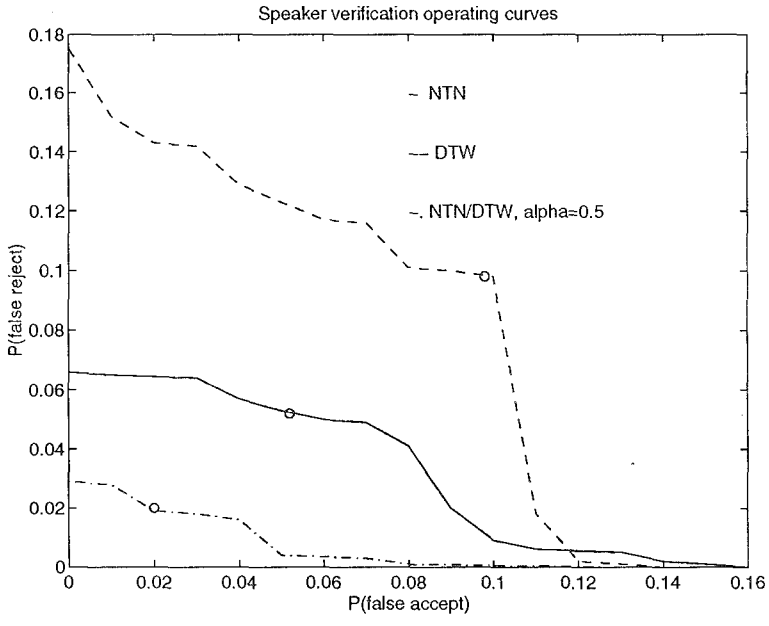


Figure 2: Speaker verification operating curves

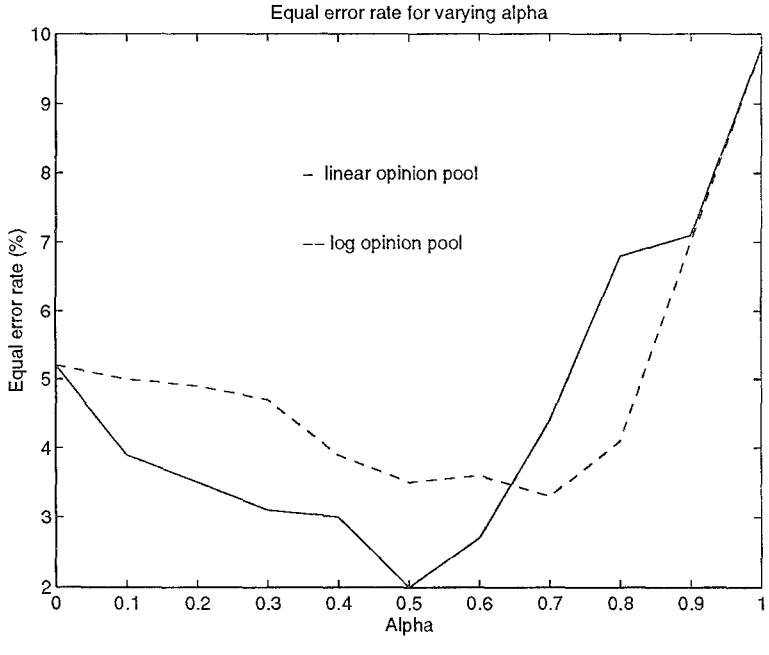


Figure 3: Equal error rate versus alpha

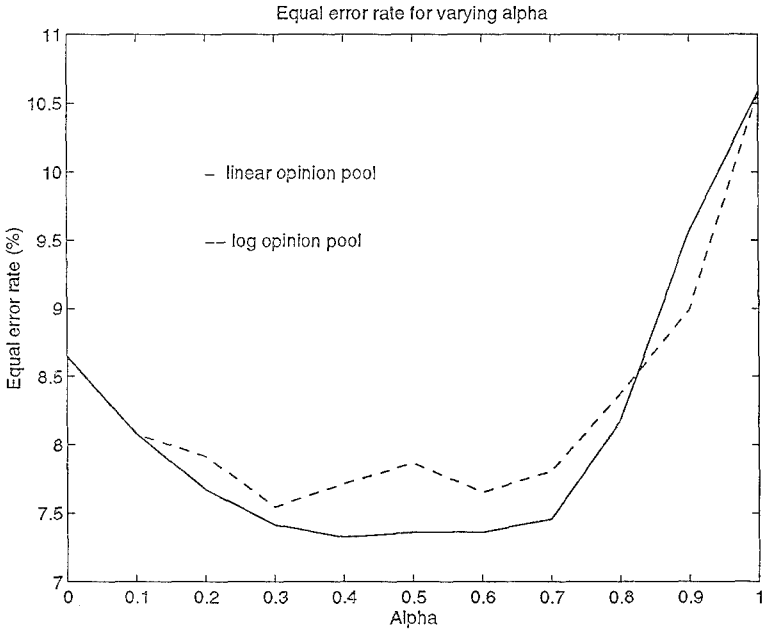


Figure 4: Equal error rate versus alpha

with the first four speakers as imposters. Each target speaker is modeled with both a NTN and HMM. The HMMs each have eight states and are trained with the hidden Markov model toolkit (HTK) developed by Cambridge University. The verification utterances are based on the utterance of one digit, which yields a total of 1200 target speaker trials and 8000 imposter attempts. The equal error rate plots as a function of alpha are shown in Figure 4. The equal error rates of the NTN and HMM classifiers when used individually are 10.6% and 8.7%, respectively. The best performance for the combined system is an equal error rate of roughly 7.3%. Unfortunately, this database does not contain a large population of speakers to be used as imposters. However, the experiments still show the relative improvement that can be obtained when using a consensus-driven system.

4 CONCLUSION

Several consensus-driven systems have been evaluated for incorporating discriminative information into text-dependent speaker verification tasks. The first system combines information provided by NTN and DTW models. The second system combines the NTN and HMM. The linear and log opinion pools are evaluated for combining the multiple sources of information for two text-dependent speaker verification tasks. In both experiments, the combined

classifier system provides a lower equal error rate than either classifier used in isolation.

References

[1] B.H. Juang and S. Katagiri. Discriminative learning for minimum error classification. *IEEE Trans. SP*, 40, December 1992.

[2] E. McDermott and S. Katagiri. Minimum error training for speech recognition. In *Proceedings of the 1994 IEEE Workshop on Neural Networks for Signal Processing*, pages 259–268, 1994.

[3] C.S. Liu, C.H. Lee, B.H. Juang, and A.E. Rosenberg. Speaker recognition based on minimum error discriminative training. In *Proceedings ICASSP*, 1994.

[4] N. Morgan and H. Bourlard. Continuous speech recognition using multilayer perceptrons with hidden markov models. In *Proceedings ICASSP*, 1990.

[5] A. Sankar and R.J. Mammone. Growing and pruning neural tree networks. *IEEE Transactions on Computers*, C-42:221–229, March 1993.

[6] K.R. Farrell, R.J. Mammone, and K.T. Assaleh. Speaker recognition using neural networks and conventional classifiers. *IEEE Transactions Speech and Audio Processing*, 2(1), part 2, 1994.

[7] H. Liou and R.J. Mammone. Text-dependent speaker verification using sub-word neural tree networks. In *SPIE Proceedings, Conference on the Automatic Inspection and Identification of Humans*, July 1994.

[8] S. Furui. Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-29:254–272, April 1981.

[9] J.G. Wilpon and L.R. Rabiner. A modified k-means clustering algorithm for use in isolated word recognition. *IEEE Trans. ASSP*, 33:587–594, June 1985.

[10] A.E. Rosenberg, C.H. Lee, and F.K. Soong. Sub-word unit talker verification using hidden markov models. In *Proceedings ICASSP*, pages 269–272, 1990.

[11] A.E. Rosenberg, C.H. Lee, and S. Gokeen. Connected word talker recognition using whole word hidden markov models. In *Proceedings ICASSP*, pages 381–384, 1991.

[12] J.M. Naik, L.P. Netsch, and G.R. Doddington. Speaker verification over long distance telephone lines. In *Proceedings ICASSP*, 1989.

[13] J.A. Benediktsson and P.H. Swain. Consensus theoretic classification methods. *IEEE Trans. on Systems, Man and Cybernetics*, 22(4):688–704, 1992.

[14] L. Xu, A. Krzyzak, and C.Y. Suen. Methods of combining multiple classifiers and their applications to hand-written character recognition. *IEEE Trans. on Systems, Man and Cybernetics*, 23(3):418–435, 1992.

[15] T.K. Ho, J.J. Hull, and S.N. Srihari. Decision combination in multiple classifier systems. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(1):66–75, 1994.

[16] F. K. Soong and A. E. Rosenberg. On the use of instantaneous and transitional spectral information in speaker recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-36:871–879, June 1988.

[17] K.R. Farrell and R.J. Mammone. Neural tree network/vector quantization probability estimators for speaker recognition. In *Proceedings of the 1994 IEEE Workshop on Neural Networks for Signal Processing*, pages 279–288, 1994.

[18] M.M. Hochberg, G.D. Cook, S.J. Renals, and A.J. Robinson. Connectionist model combination for large vocabulary speech recogition. In *Proceedings of the 1994 IEEE Workshop on Neural Networks for Signal Processing*, pages 269–278, 1994.

FROM ARTIFICIAL NEURAL NETWORK INVERSION TO HIDDEN MARKOV MODEL INVERSION: APPLICATION TO ROBUST SPEECH RECOGNITION

Seokyong Moon, Jenq-Neng Hwang

Information Processing Laboratory
Department of Electrical Engineering, FT-10
University of Washington
Seattle, WA 98195
moon@pierce.ee.washington.edu, hwang@ee.washington.edu

Abstract

The *gradient* based hidden Markov model (HMM) inversion algorithm is studied and applied to robust speech recognition tasks under general types of mismatched conditions. It stems from the gradient-based inversion algorithm of an artificial neural network (ANN) by viewing an HMM as a special type of ANNs. The HMM *inversion* has a conceptual duality to the HMM *training* just as ANN *inversion* does to ANN *training*. The *forward training* of an HMM, based on either the Baum-Welch reestimation or gradient method, finds the model parameters λ to optimize some criteria, e.g., maximum likelihood (ML), maximum mutual information (MMI) and mean squared error (MSE), with given speech inputs s. On the other hand, the *inversion* of an HMM finds speech inputs s, that optimize some criterion with given model parameters λ. The performance of the proposed *gradient* based HMM inversion for noisy speech recognition under additive noise corruption and microphone mismatch conditions is compared with the *robust* Baum-Welch HMM inversion technique along with other noisy speech recognition technique, i.e., the *robust* MINIMAX classification technique.

0-7803-2739-X/95 $4.00 © 1995 IEEE

1 Introduction

The idea of "non-parametric *inversion*" [1] originates from the study of artificial neural networks (ANNs), which are commonly used as non-parametric universal *approximators* [2] to model the functional relationship between inputs $\mathbf{x}$ and outputs $\mathbf{y}$ with interconnection weights $\mathbf{w}$, i.e., $\mathbf{y} = \Phi(\mathbf{w}, \mathbf{x})$. *Training* of an ANN finds the ANN's *weights* $\mathbf{w}$ that optimize some criterion, usually minimum mean-squared error (MMSE) [3]. *Inversion* of an ANN finds the *inputs* $\mathbf{x}$ that optimize some criterion while keeping the weights of an ANN intact [1]. Both the training and the inversion of ANNs commonly adopt the gradient descent based iterative search in the optimization procedure.

The close structural and algorithmic relationship between a hidden Markov model (HMM) and an ANN has been well established [4]. For a continuous density multi-mixture (CDMM)-HMM, the model parameters consist of five major components $\lambda = \{\pi, A, \mu, \Sigma, C\}$, where $\pi = \{\pi_i\}$ denotes the set of initial state probabilities, $A = \{a_{ij}\}$ is the set of state transition probabilities, $\mu = \{\mu_{ik}\}$ is the set of mean vectors of Gaussian mixtures, $\Sigma = \{\Sigma_{ik}\}$ is the set of covariance matrices of Gaussian mixtures, and $C = \{c_{ik}\}$ is the set of intensity weighting of Gaussian mixtures. The HMM output (likelihood) probability $P(\mathbf{s}|\lambda)$ can be defined as a function of model parameters λ and speech inputs $\mathbf{s}$, i.e., $P(\mathbf{s}|\lambda) = \Psi(\mathbf{s}, \lambda)$. Based on the functional dependency of the HMM's output likelihood to the model parameters λ and inputs $\mathbf{s}$, it is easy to generalize the inversion of an ANN to the inversion of an HMM. More specifically, the *inversion* of an HMM finds speech inputs $\mathbf{s}$ that optimize some criterion with given model parameters λ. Unlike an ANN which mainly utilizes the gradient based optimization technique in the training and the inversion process, an HMM has a very efficient Baum-Welch type of training [5] and inversion [6] procedure. The Baum-Welch type of HMM inversion has been successfully applied to robust speech recognition tasks and produced very encouraging results [6]. The gradient-based HMM inversion can similarly be applied to robust speech recognition tasks by moving the input speech features toward the means of Gaussian mixtures with appropriate constraints, e.g., *robustness bound constraint* [7].

2 Inversion of Hidden Markov Models

2.1 Gradient-Based HMM Inversion

Given an N-state and K-mixture CDMM-HMM, the observation probability of the t-th input speech frame $\mathbf{s}_t$ (normally in terms of pre-processed features, e.g., order P cepstral coefficients) of a T-frame speech $\mathbf{s} = \{\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_T\}$ at state i is defined as [8], where $i = 1, \cdots, N$.:

$$b_i(\mathbf{s}_t) = \sum_{k=1}^{K} \frac{c_{ik}}{((2\pi)^P |\Sigma_{ik}|)^{1/2}} \cdot \exp(-\frac{1}{2}(\mathbf{s}_t - \mu_{ik})^T \cdot \Sigma_{ik}^{-1} \cdot (\mathbf{s}_t - \mu_{ik})). \quad (1)$$

Note that the nonlinearity is introduced by the mixture Gaussian density functions in each state. Let θ_t denotes the state where $\mathbf{s}_t$ resides in, we can recursively calculate the forward probability $\alpha_i(t) = P(\mathbf{s}_1, \cdots, \mathbf{s}_t \text{ and } \theta_t = i|\lambda)$ and the backward probability $\beta_i(t) = P(\mathbf{s}_{t+1}, \cdots, \mathbf{s}_T|\theta_t = i \text{ and } \lambda)$ as follows [8]:

$$\alpha_i(t) = b_i(\mathbf{s}_t) \cdot \sum_{j=1}^{N} a_{ji} \cdot \alpha_j(t-1), \qquad i = 1, \cdots, N, \tag{2}$$

$$\beta_i(t) = \sum_{j=1}^{N} a_{ij} \cdot b_j(\mathbf{s}_{t+1}) \cdot \beta_j(t+1), \qquad i = 1, \cdots, N. \tag{3}$$

By treating an HMM as a special type of ANN's, the gradient-based inversion of an HMM for the τ-th coefficient in $\mathbf{s}_t$ can be derived with the metric function $C_{OPT}(\lambda, \{\mathbf{s}_t\})$ to be maximized:

$$s_{t,\tau} \Longleftarrow s_{t,\tau} + \eta \cdot \frac{\partial C_{OPT}}{\partial s_{t,\tau}}, \tag{4}$$

where $C_{OPT} = \log P(\mathbf{s}|\lambda) = \log \alpha_N(T)$ under the maximum likelihood (ML) criterion, $P(\mathbf{s}|\lambda)$ is model probability (likelihood) and is equal to the forward probability from a single final state $\alpha_N(T)$. The gradients $\frac{\partial C_{OPT}}{\partial s_{t,\tau}}$ can be further expanded using the chain rule [9]:

$$\frac{\partial C_{OPT}}{\partial s_{t,\tau}} = \sum_{i=1}^{N} \frac{\partial C_{OPT}}{\partial b_i(\mathbf{s}_t)} \cdot \frac{\partial b_i(\mathbf{s}_t)}{\partial s_{t,\tau}}, \tag{5}$$

and the first derivative in Eq.(5) with respect to $b_i(\mathbf{s}_t)$ becomes:

$$\frac{\partial C_{OPT}}{\partial b_i(\mathbf{s}_t)} = \frac{1}{\alpha_N(T)} \cdot \frac{\partial \alpha_N(T)}{\partial b_i(\mathbf{s}_t)}. \tag{6}$$

The $\frac{\partial \alpha_N(T)}{\partial b_i(\mathbf{s}_t)}$ can be recursively expanded using a chain rule:

$$\begin{aligned} \frac{\partial \alpha_N(T)}{\partial b_i(\mathbf{s}_t)} &= \frac{\partial \alpha_N(T)}{\partial \alpha_i(t)} \cdot \frac{\partial \alpha_i(t)}{\partial b_i(\mathbf{s}_t)} \\ &= \left(\sum_{j=1}^{N} \frac{\partial \alpha_N(T)}{\partial \alpha_j(t+1)} \cdot \frac{\partial \alpha_j(t+1)}{\partial \alpha_i(t)}\right) \cdot \left(\sum_{j=1}^{N} a_{ji} \cdot \alpha_j(t-1)\right) \\ &= \left(\sum_{j=1}^{N} \frac{\partial \alpha_N(T)}{\partial \alpha_j(t+1)} \cdot b_j(\mathbf{s}_{t+1}) \cdot a_{ij}\right) \cdot \left(\sum_{j=1}^{N} a_{ji} \cdot \alpha_j(t-1)\right). \end{aligned} \tag{7}$$

Note that the derivative $\frac{\partial \alpha_N(T)}{\partial \alpha_i(t)}$ is equivalent to backward probability $\beta_i(t)$ defined in the standard HMM (as shown in Eq. (3) and Eq. (7)) [4, 9, 10].

$$\frac{\partial \alpha_N(T)}{\partial \alpha_i(t)} = \sum_{j=1}^{N} b_j(\mathbf{s}_{t+1}) \cdot a_{ij} \cdot \frac{\partial \alpha_N(T)}{\partial \alpha_j(t+1)}, \tag{8}$$

with $\frac{\partial \alpha_N(T)}{\partial \alpha_N(T)} = \beta_N(T) = 1$.

The second derivative $\frac{\partial b_i(\mathbf{s}_t)}{\partial s_{t,\tau}}$ in Eq. (5) can be calculated by differentiating Eq. (1) [9]:

$$\sum_{k=1}^{K} \frac{c_{ik}}{((2\pi)^P |\Sigma_{ik}|)^{1/2}} \cdot (\sum_{l=1}^{P} q_{k,l,\tau}(\mu_{ik,l} - s_{t,l})) \cdot \exp(-\frac{1}{2}(\mathbf{s}_t - \mu_{ik})^T \cdot \Sigma_{ik}^{-1} \cdot (\mathbf{s}_t - \mu_{ik})), \tag{9}$$

where $q_{k,l,\tau}$ denotes the $(l,\tau)-$th element of the i-th state and the k-th Gaussian mixture's inverse covariance matrix Σ_{ik}^{-1} and $\mu_{ik,\tau}$ is the τ-th coefficients of the i-th state and the k-th Gaussian mixture mean.

The gradient-based inversion of an HMM with maximum likelihood optimization criterion C_{OPT} is derived using Eq. (5), (6), (7) and (9).

$$\begin{aligned} s_{t,\tau} \quad \Longleftarrow \quad & s_{t,\tau} + \eta \cdot \sum_{i=1}^{N} \frac{1}{\alpha_N(T)} \cdot \beta_i(t) \cdot \frac{\alpha_i(t)}{b_i(\mathbf{s}_t)} \cdot \sum_{k=1}^{K} \frac{c_{ik}}{((2\pi)^P |\Sigma_{ik}|)^{1/2}} \\ & (\sum_{l=1}^{P} q_{k,l,\tau}(\mu_{ik,l} - s_{t,l})) \cdot \exp(-\frac{1}{2}(\mathbf{s}_t - \mu_{ik})^T \cdot \Sigma_{ik}^{-1} \cdot (\mathbf{s}_t - \mu_{ik})). \end{aligned} \tag{10}$$

2.2 Baum-Welch HMM Inversion

The Baum-Welch HMM inversion maximizes the auxiliary function $Q(\lambda, \lambda; \mathbf{s}, \mathbf{s}')$ defined as:

$$Q(\lambda, \lambda; \mathbf{s}, \mathbf{s}') = \sum_{\theta} \sum_{\mathcal{K}} P(\mathbf{s}, \theta, \mathcal{K}|\lambda) \cdot \log P(\mathbf{s}', \theta, \mathcal{K}|\lambda), \tag{11}$$

where θ and $\mathcal{K}$ denote the possible state transition sequence and the Guassian mixture segmentation sequence, respectively, for a T-frame speech utterance $\mathbf{s} = \{\mathbf{s}_t, 1 \leq t \leq T\}$.

The problem of the inversion of an N-state and K-mixture HMM is to find $\hat{\mathbf{s}}$ that maximizes $Q(\lambda, \lambda; \mathbf{s}, \mathbf{s}')$. Note that, the auxiliary function with observation probability, $b_{ik}(\cdot)$, can be expanded as [5]:

$$\sum_{\theta} \sum_{\mathcal{K}} P(\mathbf{s}, \theta, \mathcal{K}|\lambda) \cdot \{\log \pi_{\theta_0} + \sum_{t=1}^{T} \log a_{\theta_{t-1}\theta t} + \sum_{t=1}^{T} \log b_{\theta_t k_t}(\mathbf{s}'_t) + \sum_{t=1}^{T} \log c_{\theta_t k_t}\}. \tag{12}$$

By equating the derivative of $Q(\cdot)$ with respect to $\mathbf{s}'_t$ to be zero, i.e., $\frac{\partial Q(\lambda, \lambda; \mathbf{s}, \mathbf{s}')}{\partial \mathbf{s}'_t} = 0$ (the steepest ascent method),

$$\begin{aligned} \frac{\partial Q(\lambda, \lambda; \mathbf{s}, \mathbf{s}')}{\partial \mathbf{s}'_t} &= \frac{\partial}{\partial \mathbf{s}'_t} [\sum_{\theta} \sum_{\mathcal{K}} P(\mathbf{s}, \theta, \mathcal{K}|\lambda) \cdot \sum_{t=1}^{T} \log b_{\theta_t k_t}(\mathbf{s}'_t)] \\ &= \sum_{i=1}^{N} \sum_{k=1}^{K} P(\mathbf{s}, i, k|\lambda) \cdot (\mathbf{s}'_t - \mu_{ik}) = 0, \end{aligned} \tag{13}$$

we can find the reestimated inputs $\bar{\mathbf{s}}_t$,

$$\bar{\mathbf{s}}_t = \frac{\sum_{i=1}^{N}\sum_{k=1}^{K} P(\mathbf{s}, i, k|\lambda)\cdot\mu_{ik}}{\sum_{i=1}^{N}\sum_{k=1}^{K} P(\mathbf{s}, i, k|\lambda)}. \tag{14}$$

By substituting $P(\mathbf{s}, i, k|\lambda)$ with $\sum_{j=1}^{N}\alpha_j(t-1)a_{ji}c_{ik}b_{ik}(\mathbf{s}_t)\beta_i(t)$, the Baum-Welch inversion algorithm is derived.

$$\bar{\mathbf{s}}_t = \frac{\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{j=1}^{N}\alpha_j(t-1)a_{ji}c_{ik}b_{ik}(\mathbf{s}_t)\beta_i(t)\mu_{ik}}{\sum_{i=1}^{N}\sum_{k=1}^{K}\sum_{j=1}^{N}\alpha_j(t-1)a_{ji}c_{ik}b_{ik}(\mathbf{s}_t)\beta_i(t)} \tag{15}$$

2.3 Duality Between HMM Reestimation and Inversion

There is a *duality*, in the sense of maximizing paradigms, between HMM reestimation and HMM inversion. As discussed previously, both HMM reestimation and HMM inversion algorithms can adopt either the Baum-Welch or the gradient-based optimization techniques. HMM inversion moves the input speech $\{\mathbf{s}\}$ closer to the mean $\{\mu_{ik}\}$ of a Gaussian mixture by fixing the mean location of each mixture. On the other hand, the HMM reestimation moves the mean $\{\mu_{ik}\}$ location of each Gaussian mixture closer to the input speech $\{\mathbf{s}\}$ by fixing the input speech location. Figure 1 shows the conceptual difference between HMM reestimation and inversion, where mean $\{\mu_{ik}\}$ of each Gaussian mixture is marked as 'o', the noise-free input speech $\{\mathbf{s}\}$ is marked as 'x' (without specifying the time indices), and the noisy input speech $\{\mathbf{y}\}$ is marked as '*'. The HMM inversion algorithm moves noisy speech (*) toward mean location (o) of a model. On the other hand, the HMM reestimation algorithm moves mean location (o) toward noisy speech location (*).

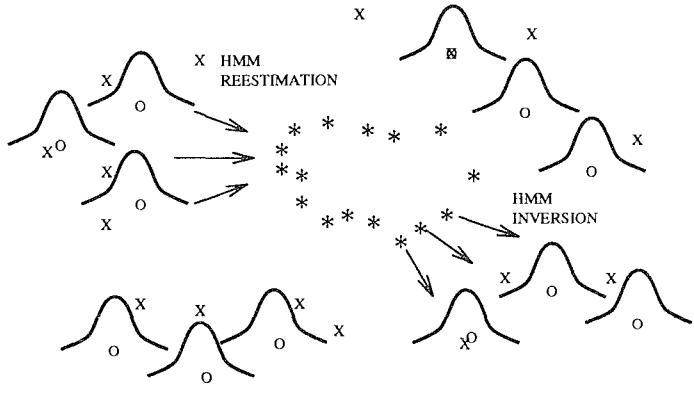


Figure 1: Conceptual difference between HMM reestimation and HMM inversion.

3 Robust HMM Inversion

The proposed HMM inversion, either by gradient-based or Baum-Welch methods, can be applied to recognition of noisy speech by adopting the framework of robust hypothesis testing, namely the *robust* HMM inversion [6]. In the testing phase of the robust HMM inversion for classifying M isolated words, the HMM inversion algorithm is used to obtain $\hat{\mathbf{s}}_m = arg\max_{\mathbf{s}\in\mathcal{S}_m} P(\mathbf{s}|\lambda)$ which maximizes the likelihood $P(\mathbf{s}|\lambda_m)$, for $1 \leq m \leq M$, within the mismatch neighborhood $\mathcal{S}_m$. The mismatch neighborhood $\mathcal{S}_m$ is defined as a non-overlapping subset of m-th nominal speech space around the assumed model λ_m. It starts with the noisy test speech data $\mathbf{s}$ by assuming $\mathbf{s}$ is generated from the model λ_m. Some *constraints* are imposed on the movement of testing speech data $\mathbf{s}$ at each iteration of HMM inversion process. Such constraint as *robust bound* which was utilized by MINIMAX technique [7] is found to be very useful. More specifically, after every iteration of the HMM inversion on the noisy testing speech data $\mathbf{s} = [s^{(1)}, s^{(2)}, \cdots, s^{(P)}]$, the newly inverted speech $\hat{\mathbf{s}}_m$ is checked against the interval $I = [s^{(\tau)} - R\tau^{-1}\rho^{\tau}, s^{(\tau)} + R\tau^{-1}\rho^{\tau}]$ with some predefined constants, $R > 0$, $0 \leq \rho < 1$, of the τ-th (cepstral) coefficient $\hat{s}_m^{(\tau)}$. If $\hat{s}_m^{(\tau)}$ is within the interval, it is used as a newly estimated coefficient for next iteration. Otherwise, the end point of I which is closest to the reestimated coefficient is used. After all $\{\hat{\mathbf{s}}_m,\ m = 1, \cdots, M\}$ are converged, the one resulting in the maximum $P(\hat{\mathbf{s}}_m|\lambda_m)$ will be classified as the winner.

According to [8], the norm of cepstral coefficients shrinks in presence of additive white Gaussian noise (AWGN) and shift due to convolutive mismatch (e.g., microphone mismatch). Therefore, *scaling* (by a factor of γ) and *shifting* of the speech are incorporated "before" the robust HMM inversion so that a *minimal* use of inversion is assured to maintain the feasible structure of original speech [6].

Due to the use of HMM reestimation, which involves temporal averaging of speech frames grouped to the same state in an HMM, the MINIMAX technique cannot efficiently compensate the temporal structure deviation caused by noise corruption. On the other hand, the HMM inversion moves the noisy speech (frame-by-frame) to the Gaussian mixture centers of a model at each time step. Normally, one dominating Gaussian mixture guides the movement of each frame of noisy speech. This fact enables HMM inversion to move the noisy speech to the fine details of an HMM which has been trained with noise-free speech. This fact also enables the HMM inversion to deviate from the original temporal correlation of speech if too much maximization is undergone. To reach a better compromise, the MINIMAX maximization can be combined with the HMM inversion maximization in a batch fashion. More specifically, after completion of MINIMAX maximization which more or less reshapes the original HMM model to be close to the noisy speech, the robust HMM inversion is then performed based on the newly reestimated model $\widetilde{\lambda}$ on the testing speech to fine-tune the speech. Instead of combining the HMM

inversion and MINIMAX in a batch fashion, they can be combined in a sequential manner. In this sequential maximization, one iteration consists of a single-step of MINIMAX maximization and another single-step of HMM inversion.

4 Simulation Results

Robust HMM inversion is applied to noisy speech recognition to deal with various type of mismatch conditions. The speech database used in this experiment is TI isolated digit database ($M = 10$) which consists of 16 speakers' digit utterances. Ten continuous density HMMs (CDHMMs) with $N = 7$ states and $K = 4$ Gaussian mixtures are used to model 10 digits. In the training phase, 256 training tokens (16 speakers, 16 repetitions) are used for training each digit HMM. In testing phase, 300 tokens (6 speakers, 5 repetitions, 10 digits) are used for one experiment and it is repeated 10 times, with different random noise seeds, to get sufficient statistics. Various type of mismatch conditions are simulated, including AWGN, jittering white noise, microphone mismatch. Jittering white noise is generated by randomly multiplying the noise standard deviation for each frame with one of five constants [11], i.e., constant= {3, 2, 1, 1/2, 1/3}. To simulate the microphone mismatch effect, noisy speech data are convolved with a 2nd-order FIR filter, $a_1 = -0.45, a_2 = 0.55$ [11].

Table 1 shows the recognition performance of HMMs in noisy environments when the mismatch is incurred by AWGN at various SNR. The performance of an HMM which can achieve 95.47% accuracy in noise-free (SNR=∞) environments degrades abruptly to accuracy of 25.73% at SNR of 5 dB without any compensation (see Standard). The scaled robust HMM reestimation (see Minimax) greatly improves the HMM performance [6, 7]. For example, 36.73% (see Standard) at SNR of 10 dB is increased to 66.37% (see Minimax). It showed consistent performance improvement over the entire SNR. The robust gradient-based HMM inversion with pre-scaling (see Inversion(G)) slightly improves the performance of HMM when SNR $\geq$ 10 dB. The performance of robust gradient-based HMM inversion (see Inversion (G)) is apparently inferior to robust Baum-Welch HMM inversion (see Inversion (B)). The batch combination of the gradient-based HMM inversion and the MINIMAX technique (see Batch(G)) achieved much better performance. The batch combination of the Baum-Welch HMM inversion with the MINIMAX technique (see Batch(B)) achieved the highest recognition performance. Scaling followed by sequential combination (see Sequential(G)) also achieved similar performance to batch combination procedure. The robustness bound constants (R and ρ), the pre-scaling constant (γ), and the inversion rate (η) used for these experiments are also shown in the table.

Table 2 shows the recognition performance of HMMs when testing speech is contaminated by jitter white noise at various level of SNR. Similar behaviors

SNR(dB)	5	10	15	20	25	30	35	∞
Standard	25.73	36.73	58.90	76.63	87.20	91.97	94.70	95.47
Minimax	37.73	66.37	81.73	86.80	93.27	96.43	96.63	96.63
Inversion(G)	20.93	41.97	69.67	81.60	90.23	94.43	95.47	94.87
Batch(G)	53.90	75.87	86.57	89.33	93.97	96.20	96.03	96.40
Sequential(G)	54.90	75.97	86.30	89.27	93.97	96.53	96.23	96.07
Inversion(B)	35.97	65.40	82.87	88.10	94.13	96.33	96.90	96.70
Batch(B)	58.23	76.23	84.57	89.20	94.70	96.00	95.67	96.30
Sequential(B)	57.73	74.93	84.17	89.40	94.50	96.30	95.67	96.07
R	4.0	4.0	4.0	2.0	2.0	2.0	2.0	2.0
ρ	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
γ	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6
η	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

Table 1: HMM performance for AWGN (%=correct/3000).

SNR(dB)	5	10	15	20	25	30	35	∞
Standard	27.43	45.27	62.63	77.63	87.37	91.10	93.90	95.63
Minimax	43.47	64.60	80.33	85.63	92.10	95.73	96.23	96.30
Inversion(G)	25.50	45.10	68.70	79.37	88.77	92.73	94.67	94.90
Batch(G)	58.20	74.90	85.37	88.80	93.50	95.37	96.00	95.57
Sequential(G)	57.67	75.73	85.43	88.63	93.70	95.57	96.27	95.90
Inversion(B)	45.00	65.97	82.27	87.27	93.23	95.93	96.43	96.83
Batch(B)	60.77	74.90	83.67	89.77	94.07	96.23	96.13	96.40
Sequential(B)	60.33	75.87	84.87	89.00	93.80	95.93	96.07	96.33
R	4.0	4.0	4.0	2.0	2.0	2.0	2.0	2.0
ρ	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
γ	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6
η	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

Table 2: HMM performance for jittering noise (%=correct/3000).

SNR(dB)	5	10	15	20	25	30	35	∞
Standard	19.43	33.63	52.10	71.07	83.57	88.90	93.00	94.73
Minimax	28.83	55.53	73.27	81.87	90.67	93.07	94.63	95.33
Inversion(G)	21.33	42.80	67.13	79.20	88.47	91.67	92.87	93.63
Batch(G)	42.77	65.43	79.80	85.47	90.03	93.17	94.13	95.07
Sequential(G)	42.83	66.77	79.43	86.00	90.03	92.87	93.90	95.03
Inversion(B)	26.97	52.60	72.60	83.47	90.83	93.07	94.43	95.30
Batch(B)	43.33	63.67	77.13	85.17	90.50	92.90	94.10	95.20
Sequential(B)	41.80	63.87	77.10	85.67	90.67	92.97	94.33	95.27
R	4.0	4.0	4.0	2.0	2.0	2.0	2.0	2.0
ρ	0.3	0.3	0.3	0.3	0.3	0.3	0.3	0.3
γ	1.6	1.6	1.6	1.6	1.6	1.6	1.6	1.6
η	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01

Table 3: HMM performance for microphone mismatch (%=correct/3000).

for various noisy speech compensation techniques were observed as AWGN contamination.

Table 3 shows the recognition performance of HMMs when a different microphone is used for capturing the noisy testing speech at various SNR levels. This is a quite difficult task for speech classifiers since the testing speech is corrupted by AWGN (i.e., cepstral coefficients experiences the norm shrinkage and possibly angular distortion) and microphone mismatch (i.e., cepstral coefficients experiences shifting due to convolution with different microphone). The cepstral shifting compensation was incorporated before the proposed compensation techniques. For the cepstral shifting compensation, the signal-to-noise ratio dependent cepstrum normalization (SDCN) technique [12] was used. After incorporating SDCN technique, the behavior of various compensation technique is similar to the one for AWGN (see Table 1).

5 Conclusion

The Baum-Welch HMM inversion proves to be a more reliable compensation technique than the gradient-based HMM inversion, despite the fact that the latter one can sometimes outperforms the former one when best inversion rate η is chosen. This deficiency in performance can be overcome when using other more discriminative criterion, e.g., MMI, where only gradient based robust HMM inversion is applicable. Combination of robust MINIMAX and robust HMM inversion substantially increased the recognition performance of HMMs.

References

[1] J. N. Hwang, C. H. Chan. Iterative constrained inversion of neural networks and its applications. In Proc. 24-th Conf. on Information Systems and Sciences, pp. 754-759, Princeton, March 1990.

[2] Halbert White. Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings. Neural Networks, Vol. 3, pp. 535-549, 1990.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. Parallel Distributed Processing: Exploration in the Microstructure of Cognition, Vol. 1, Chapter 8, MIT Press, Cambridge, Massachusetts, 1986.

[4] J. N. Hwang, J. A. Vlontzos, S. Y. Kung. A systolic neural network architecture for hidden Markov models. IEEE Trans. on ASSP, 37(12):1967-1979, December 1989.

[5] B. H. Juang. Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chains. AT&T Technical Journal, 64(6):1235-1249, July 1985.

[6] S.Y. Moon, J.N. Hwang. Noisy speech recognition using robust inversion of hidden Markov models. IEEE Int'l Conference on ASSP, pp. 145-148, Detroit Michigan, May 1995.

[7] N. Merhav, C. H. Lee. A minimax classification approach with application to robust speech recognition. IEEE Trans. on SAP, 1(1):90-100, January 1993.

[8] B. H. Juang, K. K. Paliwal. Hidden Markov models with first-order equalization for noisy speech recognition. IEEE Trans. on SP, 40(9):2136-2143, September 1992.

[9] Yoshua Bengio, R. D. Mori, G. Flammia, R. Kompe. Global optimization of a neural network-hidden Markov model hybrid. IEEE Trans. on NN, Vol. 3, No. 2, pp. 252-259, March 1992.

[10] Leslie Thomas Niles. Modeling and learning in speech recognition: The relationship between stochastic pattern classifiers and neural networks. Ph.D. Thesis, Brown University, May 1991.

[11] B. A. Carlson, M. A. Clements. A projection-based likelihood measure for speech recognition in noise. IEEE Trans. on SAP, 2(1):97-102, January 1994.

[12] A. Acero, R. M. Stern. Environmental robustness in automatic speech recognition. IEEE Int'l Conference on ASSP, pp. 849-852, April 1990.

Hierarchical Mixtures of Experts Methodology Applied to Continuous Speech Recognition

Ying Zhao *Richard Schwartz* *Jason Sroka*† *John Makhoul*

BBN Systems and Technologies, Cambridge, MA 02138
†MIT, Cambridge, MA 02139
yzhao@bbn.com

Abstract

In this paper, we incorporate the Hierarchical Mixtures of Experts (HME) method of probability estimation, developed by Jordan [1], into an HMM-based continuous speech recognition system. The resulting system can be thought of as a continuous-density HMM system, but instead of using gaussian mixtures, the HME system employs a large set of hierarchically organized but relatively small neural networks to perform the probability density estimation. The hierarchical structure is reminiscent of a decision tree except for two important differences: each "expert" or neural net performs a "soft" decision rather than a hard decision, and, unlike ordinary decision trees, the parameters of all the neural nets in the HME are automatically trainable using the EM algorithm. We report results on the ARPA 5,000-word and 40,000-word Wall Street Journal corpus using HME models.

1 Introduction

Recent research has shown that a continuous-density HMM (CD-HMM) system can outperform a more constrained tied-mixture HMM system for large-vocabulary continuous speech recognition (CSR) when a large amount of training data is available [2]. In other work, the utility of decision trees has been demonstrated in classification problems by using the "divide and conquer" paradigm effectively, where a problem is divided into a hierarchical set of simpler problems. We present here a new CD-HMM system which has similar properties and possesses the same advantages as decision trees, but has the additional important advantage of having automatically trainable "soft" decision boundaries.

2 Hierarchical Mixtures of Experts

The method of Hierarchical Mixtures of Experts (HME) developed recently by Jordan [1] breaks a large scale task into many small ones by partition-

0-7803-2739-X/95 $4.00 © 1995 IEEE

ing the input space into a nested set of regions, then building a simple but specific model (local expert) in each region. The idea behind this method follows the principle of divide-and-conquer which has been utilized in certain approaches to classification problems, such as decision trees. In the decision tree approach, at each level of the tree, the data are divided explicitly into regions. In contrast, the HME model makes use of "soft" splits of the data, i.e., instead of the data being explicitly divided into regions, the data may lie simultaneously in multiple regions with certain probabilities. Therefore, the variance-increasing effect of lopping off distant data in the decision tree can be ameliorated. Furthermore, the "hard" boundaries in the decision tree are fixed once a decision is made, while the "soft" boundaries in the HME are parameterized with generalized sigmoidal functions, which can be adjusted automatically using the Expectation-Maximization (EM) algorithm during the splitting.

Now we describe how to apply the HME methodology to the CSR problem. For each state of a phonetic HMM, a separate HME is used to estimate the likelihood. The actual HME first computes a posterior probability $P(l|x,s)$, the probability of phoneme class l, given the input feature vector x and state s. That probability is then divided by the *a priori* probability of the phone class l at state s. A one-level HME performs the following computation:

$$P(l|x,s) = \sum_{i=1}^{C} P(l|c_i,x,s)P(c_i|x,s) \tag{1}$$

where $l = 1,...,L$ indicates phoneme class, c_i represents a local region in the input space, and C is the number of regions. $P(c_i|x,s)$ can be viewed as a gating network, while $P(l|c_i,x,s)$ can be viewed as a local expert classifier (expert network) in the region c_i [1]. In a two-level HME, each region c_i is divided in turn into C subregions. The term $P(l|c_i,x,s)$ is then computed in a similar manner to equation (1), and so on. If in some of these subregions there are no data available, we back off to the parent network.

3 TECHNICAL DETAILS

As in Jordan's paper, we use a generalized sigmoidal function to parameterize $P(c_i|x)$ as follows:

$$P(c_i|x) = \frac{e^{v_i^T x}}{\sum_j e^{v_j^T x}} \tag{2}$$

where x can be the direct input (in a one-layer neural net) or the hidden layer vector (in a two-layer neural net), and $v_i, i = 1, ..., C$ are weights which need to train. Similarly, the local phoneme classifier in region c_i, $P(l|c_i, x)$, can be parameterized with a generalized sigmoidal function also:

$$P(l|c_i, x) = \frac{e^{\theta_{li}^T x}}{\sum_j e^{\theta_{ji}^T x}} \tag{3}$$

where $\theta_{ji}, j = 1, ..., L$ are weights. The whole system consists of two set of parameters: $v_i, i = 1, ..., C$ and $\theta_{ji}, j = 1, ..., L$, $\Theta = \{\theta_{ji}, v_i\}$. All parameters are estimated by using the EM algorithm.

The EM is an iterative approach to maximum likelihood estimation. Each iteration of an EM algorithm is composed of two steps: an Expectation (E) step and a Maximization (M) step. The M step involves the maximization of a likelihood function that is redefined in each iteration by the E step. Using the parameterizations in (2) and (3), we obtain the following iterative procedure for computing parameters $\Theta = \{v_i, \theta_{ji}\}$:

1. initialize $v_i^{(0)}$ and $\theta_{ji}^{(0)}$ for $i = 1, ..., C, j = 1, ..., L$.
2. E-step: In each iteration n, for each data pair $(x(t), l(t)), t = 1, ..., N$, compute

$$\begin{aligned} z_i(t)^{(n)} &= P(c_i|x(t), l(t), \Theta^{(n)}) \\ &= \frac{P(c_i|x(t), v_i^{(n)}) P(l(t)|c_i, x(t), \theta_{l(t),i}^{(n)})}{\sum_k P(c_k|x(t), v_k^{(n)}) P(l(t)|c_k, x(t), \theta_{l(t),k}^{(n)})} \end{aligned} \tag{4}$$

where $i = 1, ..., C$. $z_i(t)^{(n)}$ represents the probability of the data t lying in the region i, given the current parameter estimation $\Theta^{(n)}$. It will be used as a weight for this data in the region i in the M-step. The idea of "soft" splitting reflects that these weights are probabilities between 0 and 1, instead of a "hard"decision 0 or 1.

3. M-step:

$$\theta_i^{(n+1)} = \max_{\theta_i} \sum_t z_i(t)^{(n)} [\log \frac{e^{\theta_{l(t)i}^T x(t)}}{\sum_j e^{\theta_{ji}^T x(t)}}] \tag{5}$$

$$v_{i=1,...,C}^{(n+1)} = \max_{v_1,...,v_C} \sum_t \sum_k z_k(t)^{(n)} \log \frac{e^{v_i^T x(t)}}{\sum_j e^{v_j^T x(t)}} \tag{6}$$

4. Iterate until θ_{ji}, v_i converge.

The first maximization means fitting a generalized sigmoidal model (3) using the labeled data $(x(t), l(t))$ and weighting $z_i(t)^{(n)}$. The second one means fitting a generalized sigmoidal model (2) using inputs $x(t)$ and outputs $z_i(t)^{(n)}$. The criterion for fitting is the cross-entropy. Typically, the fitting can be solved by the Newton-Raphson method. However, it is quite expensive. Viewing this type of fitting as a multi-class classification task, we developed a technique to invert a generalized sigmoidal function more efficiently, which will be described in the following.

A common method in a multi-class classification is to divide the problem into many 2-class classifications. However, this method results in a positive and negative training unbalance usually. To avoid the positive and negative training unbalance, the following technique can be used to solve multi-class posterior probabilities simultaneously.

Suppose we have a labeled data set, $(x(t), l(t)), t = 1, ..., N$, where $l(t) \in \{1, ..., L\}$ is the label for t-th data. We use a generalized sigmoidal function to model the posterior probability $P(l|x)$, where $l = 1, ..., L$ as follows:

$$P_l(x) = P(l|x) = \frac{e^{\theta_l^T x}}{\sum_k e^{\theta_k^T x}} \tag{7}$$

Obviously, since these probabilities sum up to one, we have

$$P_L(x) = 1 - \sum_{l=1}^{L-1} P_l(x). \tag{8}$$

Now, a training sample $x(t)$ with a class label $l(t)$ can be interpreted as:

$$P_l(x(t)) = \begin{cases} 0.9 & l = l(t) \\ \frac{0.1}{L-1} & l \neq l(t) \end{cases} \tag{9}$$

If we define

$$\theta_l^T x = \log \frac{P_l(x)}{P_L(x)} \tag{10}$$

equation (10) implies that

$$P_l(x) = \frac{e^{\theta_l^T x}}{\sum_l^L e^{\theta_l^T x}} \tag{11}$$

for $l = 1, ..., L$ with $\theta_L^T x = 0$. This expression is the generalized sigmoidal function in (7). This means, we can train parameters in (7) to satisfy

Equation (10) from the data. Using a least squares criterion, the objective is

$$\min \sum_t \left[\theta_l^T x(t) - \log \frac{P_l(x(t))}{P_L(x(t))} \right]^2 \tag{12}$$

for $l = 1, ..., L-1$. Denote a data matrix as

$$X = \begin{bmatrix} x(1) \\ x(2) \\ \cdot \\ \cdot \\ \cdot \\ x(N) \end{bmatrix}$$

A least squares solution to (12) is

$$\theta_l = (\log a)(X^T X)^{-1} \left[\sum_{l(t)=l} x(t) - \sum_{l(t)=L} x(t) \right] \tag{13}$$

for $l = 1, ..., L$, where $a = 9(L-1)$. Substituting (13) into (11), we get

$$P_l(x) = \frac{a^{x^T (X^T X)^{-1} \sum_{l(t)=l} x(t)}}{\sum_k a^{x^T (X^T X)^{-1} \sum_{l(t)=k} x(t)}} \tag{14}$$

Equation (13) and (14) are very easy to compute. Basically, we only have to accumulate the matrix $X^T X$ and sum $x(t)$ into different classes $l = 1, ..., L$. We can obtain probabilities $P_l(x)$ by a single inversion of matrix $X^T X$ after a pass through the training data.

4 Relation to Other Work

The work reported here is very different from our previous work utilizing neural nets for CSR. There, a single segmental neural network (SNN) is used to model a complete phonetic segment [3]. Here, each HME estimates the probability density for each state of a phonetic HMM. The work here is more similar to that by Cohen *et al.* [4], the major difference being that in [4], a single very large neural net is used to perform the probability density modeling. The training of such a large network requires the use of a specialized parallel processing machine, so that the training can be done in a reasonable amount of time. By using the HME method and dividing the problem into many smaller problems, we are able to perform the needed training computation on regular workstations.

Most of the previous work on CD-HMM work has utilized mixtures of gaussians to estimate the probability densities of an HMM. Since a multilayer feedforward neural network is a universal continuous function approximator, we decided to explore the use of neural nets as an alternative approach for continuous density estimation.

5 Experimental Results

In our initial application of the HME method to large-vocabulary CSR, we used phonetic context-independent HMEs to estimate the likelihoods at each state of 5-state HMMs. We implemented a two-level HME, with the input space divided into 46 regions, and each of those regions is further divided into 46 subregions. The initial divisions were accomplished by supervised training, with each division trained to one of the 46 phonemes in the system. All gating and local expert networks in the HME had identical structures — a two-layer generalized sigmoidal network. The whole HME system was implemented within an N-best paradigm [3], where the recognized sequence was obtained as a result of a rescoring of an N-best list obtained from our baseline BYBLOS system (tied-mixture HMM) with a statistical trigram grammar.

We then built a context-dependent HME system based on the structure of the context-independent HME models described above. For each state, the whole training data was divided into 46 parts according to its left or right context. Then for each context, a separate HME model was built for that context. To be computationally feasible, we used only one-level HMEs here. We first experimented using a left-context and right-context model.

We tested the HME implementation on the ARPA 5,000-word Wall Street Journal corpus (WSJ1, H2 dev set). We report the word error rates on the same test set for a number of different systems. Table 1 shows the word error rates for i) the baseline HMM system; ii) the segment-based neural net system (SNN) iii) the hybrid SNN/HMM system iv) a HME system alone, v) a HME system combined with HMM; vi) a HME +HMM system with modified priors.

From Table 1, The performance of the baseline tied-mixture HMM is 7.8%. The performance of the SNN system (8.5%) is comparable to the HMM alone. We see that the performance of a HME (7.6%) is as good as the HMM system, which is better than the SNN system. When combined with the baseline HMM system, the HME and SNN both improve performance over the HMM alone about 10% from 7.8% to 6.8% and from 7.8%

to 7.1% respectively. We found out that the improvement could be made larger for a hybrid HME/HMM by adjusting the context-dependent priors with the context-independent priors, and then smooth context-dependent models with a context-independent model.

More specifically, in a context-dependent HME model, we usually estimate the posterior probability phoneme l, $P(l|c, x, s)$, given left or right context c and the acoustic input x in a particular state s. Because the samples may be sparse for many of context models, it is necessary to regularize (smooth) context-dependent models with a context-independent model, where there is much more data available. However, since the two models have different priors: $P(l|c, s)$ in a context-dependent model and $P(l|s)$ in a context-independent model, a simple interpolation between the two models which is $P(l|c, x, s) = \frac{P(x|l,c,s)P(l|c,s)}{P(x|c,s)}$ in a context-dependent model and $P(l|x, s) = \frac{P(x|l,s)P(l|s)}{P(x|s)}$ in a context-independent model is inconsistent. To scale the context-dependent priors $P(l|c, s)$ with a context-independent prior $P(l|s)$, we weighted each input data point x with the weight $\frac{P(l|s)}{P(l|c,s)}$ for a prior adjusting. After this modification, a context-dependent HME actually estimates $\frac{P(x|l,c,x)P(l|s)}{P(x|s)}$. It combines better with a context-independent model. For the same experiment we showed in Table 1, the word error for the HME (with HMM) droped from 6.8% to 6.2% when priors were modified. For this 5,000-word development set, we got a total of about 20% word error reduction over the tied-mixture HMM system using a HME-based neural network system.

We then switched our experiment domain from a 5,000-word to 40,000-word the test set. During this year, the BYBLOS system has been improved from a tied-mixture system to a continuous density system. We also switched to using this new continuous density BYBLOS in our hybrid HME/HMM system. The language model used here was a 40,000-word trigram grammar. The result is shown in Table 2.

From Table 2, we see that there is about a 10% word error rate reduction over the continuous density HMM system by combining a context-dependent HME system. Compared with the 20% improvement over the tied-mixture system we made for the 5,000-word development set, the improvement over the continuous density system in this 40,000-word development is less. This may be due to the big improvement of the HMM system itself.

6 CONCLUSIONS

The method of hierarchical mixtures of experts can be used as a continous density estimator to speech recognition. Experimental results showed that estimations from this approach are consistent with the estimations from the HMM system. The frame-based neural net system using hierarchical mixtures of experts improves the performance of both the state-of-the-art tied mixture HMM system and the continuous density HMM system. The HME system itself has the same performance as the state-of-the-art tied mixture HME system.

7 Acknowledgments

This work was funded by the Advanced Research Projects Agency of the Department of Defense.

References

[1] Michael Jordan, "Hierarchical Mixtures of Experts and the EM Algorithm," *Neural Computation*, 1994, in press.

[2] D. Pallett, J. Fiscus, W. Fisher, J. Garofolo, B. Lund, and M. Pryzbocki, "1993 Benchmark Tests for the ARPA Spoken Language Program," *Proc. ARPA Human Language Technology Workshop*, Plainsboro, NJ, Morgan Kaufman Publishers, 1994.

[3] G. Zavaliagkos, Y. Zhao, R. Schwartz and J. Makhoul, "A Hybrid Neural Net System for State-of-the-Art Continuous Speech Recognition," in *Advances in Neural Information Processing Systems 5*, S. J. Hanson, J. D. Cowan and C. L. Giles, eds., Morgan Kaufmann Publishers, 1993.

[4] M. Cohen, H. Franco, N. Morgan, D. Rumelhart and V. Abrash, "Context-Dependent Multiple Distribution Phonetic Modeling with MLPS," in *Advances in Neural Information Processing Systems 5*, S. J. Hanson, J. D. Cowan and C. L. Giles, eds., Morgan Kaufmann Publishers, 1993.

	Word Error Rate
HMM	7.8
SNN	8.5
HMM+SNN	7.1
HME	7.6
HME + HMM	6.8
Prior-modified HME + HMM	6.2

Table 1: Error Rates for the ARPA WSJ 5K Development Test, Trigram Grammar

	Word Error Rate
HMM	9.5
HME + HMM	8.7

Table 2: Error Rates for the ARPA WSJ 40K Test Set, Trigram Grammar

A SPEECH RECOGNIZER WITH LOW COMPLEXITY BASED ON RNN

Klaus Kasper, Herbert Reininger, Dietrich Wolf and Harald Wüst
Institut für Angewandte Physik, J.W.Goethe-Universität Frankfurt
Robert-Mayer-Straße 2-4, D-60054 Frankfurt a.M., FRG
Tel.: +49 69 798 23490, Fax: +49 69 798 22386
e-mail:kasper@apx00.physik.uni-frankfurt.de

Abstract. Speech recognition systems (SRS) designed for applications in low cost products like telephones or in systems with energetic constraints like autonomous vehicles are faced with the demand for solutions with low complexity. A small vocabulary consisting of a few command words and the digits is sufficient for most of the applications but has to be recognized robustly. Here we report about investigations concerning the application of Recurrent Neural Networks (RNN) for speaker independent recognition of speech signals with telephone bandwidth. A RNN-SRS with low complexity is developed which recognizes isolated words as well as connected digits in adverse conditions. To enable an efficient hardware implementation of the SRS we introduce Locally Recurrent Neural Networks (LRNN). LRNN are layered networks which have recurrent connections only between the neurons of a hidden layer and their n-nearest neighbours. The neurons of the input and the output layer have unidirectional and sparse connections to the hidden layer. In comparison to RNN the density of the connections is drastically reduced and long distance wiring could be avoided in a VLSI realization.

1. INTRODUCTION

In former work it has been shown that RNN are a promising approach to meet this demand [1, 2]. It has been shown that two basic problems of speech recognition namely exploiting of contextual information between feature vectors during the feature scoring and compensation of variations in time durations of speech segments, could be tackled by incorporating both tasks in a single RNN.

Here we report about investigations concerned with the capacity of the approach and with realization aspects. Firstly, in order to investigate the capacity of the approach the complexity of the recognition task was increased by extending the vocabulary size to 50 words or by recognizing connected digits. Secondly, investigations are presented concerned with three realization aspects, the process of feature extraction from speech signals, the necessary precision of the network parameters and the activity values, and the connectivity structure of the network.

0-7803-2739-X/95 $4.00 © 1995 IEEE

2. SPEECH DATA AND NETWORK TRAINING

The system vocabulary of the SRS consists of the 10 German digits, the word *zwo*, and 12 telephone command words. The speech signals were limited to telephone bandwidth and sampled with 8kHz. From these signals feature vectors were extracted every 12ms, each consisting of 12 cepstral coefficients derived from LPC parameters. Additionally, a parameter characterizing the short-time energy of the signal was used.

In the case of isolated words for the computing of the SRS parameters, feature vectors from 100 utterances of each word spoken from different speakers, were used. Speaker independent recognition rates were measured on a set containing 100 utterances of each word from speakers not included in the training set. For the case of connected digits, utterances of 60 different speakers each containing 3 digits were used for training and the utterances of 60 other speakers for testing.

In simulation experiments the network size and the parameters of the truncated back-propagation through time [3] were optimized for telephone bandlimited speech recognition. The feature vectors were processed in blocks of F vectors of dimension 12 in order to reduce the processing time. For each block the activation of all neurons were updatet and thus a vector of word scores was produced. After processing all blocks of feature vectors belonging to an utterance the word scores were accumulated in order to generate a word hypothesis.

For RNN operating in this manner it was found that a fast and reliable weight training can be done by linearly decreasing the learning parameter during 300 training epochs. Error back-propagation initiated every 30 time steps and considering the 40 preceding time steps turned out to be sufficient for learning of the relevant time dependences. More details are given in [1].

3. EXPLORING THE RECOGNITION CAPACITY OF RNN-SRS

3.1 Recognition of Isolated Words

In the case of isolated word recognition RNN were trained to estimate word likelihoods for groups of $F = 4$ consecutive feature vectors. Word hypotheses were generated by accumulating the likelihood values during the duration of an utterance. The RNN-SRS achieved a recognition rate of R_{23}=97.1% for 23 words and R_{11}=98.2% for the 11 digits. The RNN-SRS outperforms SRS based on Discrete HMM (DHMM) or Continuous HMM (CHMM) using the same kind of features. SRS based on DHMM which uses in addition to cepstral features Delta-Cep (DCep), representing dynamical information of the feature vectors explicitly, achieves with R_{23}=97.3% about the same recognition rate as the RNN-SRS. This indicates that RNN-SRS are able to exploit automatically the informa-

Table 1: Recognition rates for isolated words (R_{11}), for strings with known number of digits R_{CD^*} and for strings with unknown number of digits R_{CD}

Recognizer	R_{11}[%]	R_{CD^*}[%]	R_{CD}[%]
RNN	97.1	95.9	92.1
DHMM	95.8	92.5	84.0
CHMM	96.3	93.8	84.5

tion about the dynamic of the feature vectors.

3.2 Recognition of Connected Digits

For the task of recognizing connected digits with an unknown number of connected digits in an utterance the RNN-SRS was also trained to estimate word likelihoods. A digit i in a digit string is recognized if the condition

$$O_i(t) > \Theta_b \wedge O_i(t + \Delta t) < \Theta_e \quad \Delta t \geq 2$$

is true for the activity of the corresponding output neuron O_i. The threshold Θ_b indicates the begin of a digit while Θ_e indicates the end. Via Δt a minimum word duration constraint is introduced.

First recognition experiments were done with a RNN-SRS trained for recognizing isolated words. Only 19% of the digit strings were recognized correctly. The correctness of the single digits was 55%. The low recognition rate, compared to 98% obtained with digits spoken in isolation, is due to the drastically increased dynamics of continuously spoken digits. In further experiments the network was trained with feature vectors extracted from the digit strings, in order to adapt the network parameters to these dynamics. Furthermore, an additional ouput neuron for detecting the boundary between two digits was added to the RNN-SRS. If the derivation of the activity of this neuron is multiplied with the activities of the neurons representing the digits the segmentation of the digit string is more reliable. On the same data basis a SRS based on Discrete HMM (DHMM) with 5 states and 5 substates and a SRS based on Continuous HMM (CHMM) with 8 states and five mixtures per state were optimized in order to evaluate the results obtained with a RNN-SRS. Both HMM based SRS are using in addition to cepstral features DCeps to parametrize the speech signals. Recognition rates were measured for three recognition tasks: recognition of the 11 digits (R_{11}) which are utteranced as isolated words, recognition of strings containing a known number of connected digits (R_{CD^*}), and recognition of strings contai-

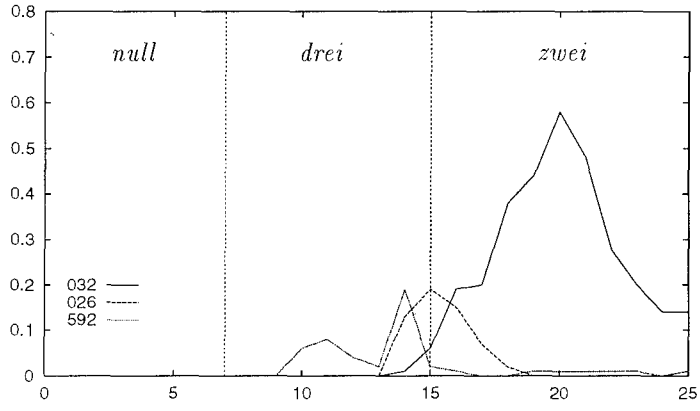


Figure 1: Activities of selected output neurons

ning an unknown number of digits (R_{CD}). As can be seen from Table 1 the RNN-SRS outperforms significantly the HMM based SRS for all three recognition tasks. Especially, the rate for R_{CD} is about 8% higher than those of the HMM-SRS. This result is due to the high discriminative power which is in contrast to HMM inherent to RNN. Interestingly, all investigated SRS, although trained with digit strings, recognized the isolated digit data base with an high accuracy of about 96%. This indicates that isolated digit recognition is indeed a subtask of connected digit recognition.

3.3 Extended Vocabulary Size

In order to evaluate the recognition capacity of the RNN-SRS the vocabulary size was extended to 53 by taking the 53 different digit strings as word categories. A RNN was trained to classify these 53 different digit strings. For the independent test data a recognition rate of about 92% results. This is quite a high rate if it is considered that only 60 utterances of each digit string were used for training. Important to note is that for recognizing 53 words only 20 additional neurons are required compared to the recognition of 23 words.

The experiment of recognizing the digit strings as words was also used to analyze how the decision for a word category is done by the RNN. One could suppose that the RNN is unable to exploit the information included in the feature vectors of the whole utterance so the word discrimination is done on the basis of a small number of feature vectors in the beginning or at the end of an utterance which are typical for a word category. In Figure 1 the activities of three selected output neurons are shown during the recognition of the string $0-3-2$. As can be seen the decision for the correct string occured during the utterance of the last digit of the string. However, information occuring at different time

instances in the feature sequence must have been exploited by the RNN because otherwise the string, $5-9-2$, ending with the same digit, as $0-3-2$, could not have been rejected.

4. REDUCING THE COMPLEXITY OF THE RNN-SRS

4.1 Preprocessing

Using a neural network with universal approximation capabilities like RNN the complexity of feature extraction can be drastically reduced. For calculating P LPC-oriented cepstral features three computational steps are performed. First, $P+1$ values of the autocorrelation function (ACF) are calculated, second, P predictor coefficients (LPC) are derived by solving a system of linear equations defined by the ACF, and third, the predictor coefficients are transformed into P cepstral coefficients (CEP) by means of a recursion. Regarding the information content, the three parameter sets, ACF, LPC, and CEP are equivalent. Unique transformations for converting one parameter set into another exists. Due to the approximation capability of RNN it could be expected that the recognition performance of the RNN-SRS obtained with CEP could also be achieved using ACF as features. Corresponding simulation experiments confirmed this statement. Without increasing the network size, a RNN-SRS with ACF as features reached a recognition rate of 96% on the test data. However, the direct use of ACF reduces the computational load of the feature extraction only about 15%. Most of the computations arise from the calculation of the $P+1$ ACF-values. At a frame size of N samples per feature vector about $(P+1)*N$ multiply-adds for a speech segment have to be performed which is 50% of the total computational complexity of the RNN-SRS. Nevertheless, the elimination of the recursive algorithm is crucial for a monolithic hardware realization of RNN-SRS.

The approach we investigated further for reducing the number of computations is based on the assumption that bandlimited speech signals can be modelled by spherically invariant random processes [4]. In this case, the normalized ACF $\psi(k)$ of a speech signal $s(n)$ can be calculated from the polarity-correlation function (PCF) $\rho(k) = E\{sgn(s(n))sgn(n+k)\}$ according to $\psi(k) = \sin(\frac{\pi}{2}\rho(k))$ [5]. The PCF can be calculated without any multiplications or additions, simply by counting. Two factors have to be considered in practical applications of the PCF in a block oriented feature extraction. If finite segments are used PCF can only be estimated and then also the stationarity assumption is problematic. In first simulation experiments we trained a RNN-SRS on the basis of PCF-values. On the training data we achieved with PCF as features the same recognition rate, near 100%, as with ACF. On the test data the recognition rate is with 94.1% about 2% below that obtained with ACF. Further improvement can be expected by optimizing the va-

lues of P and N. This indicates that in neural SRS it is indeed possible to reduce the complexity of feature extraction significantly by using PCF as features. Therefore, in SRS based on neural networks with universal approximation capabilities features consisting of PCF-values allow to realize a feature extraction with very low computational complexity. In particular with RNN, which exploit the feature dynamics automatically, a recognition performance results which can be achieved by HMM only with a highly complex feature extraction.

4.2 Quantization of Network Parameters and Activity Values

In further simulation experiments we investigated whether the parameters and activities could be coded with a precision lower than the 32 bit usually used. The feature vectors and the output activities were quantized linearly using 8 bit whereas the input activities of the neurons were quantized using 9 bit. It is investigated how an additional quantization of the network weights effects the recognition performance. In Figure 2 the distribution of the network weights together with a gaussian distribution fitted by the mean and the variance of the weights are shown. It can be seen that the distribution is gaussian like which indicates that a nonlinear quantization procedure could be favourable. In order to evaluate the results achieved with an optimum nonlinear quantizer a linear quantizer was also investigated. As can be seen from Figure 3 with a linear quantizer using 7 bit per weight value achieves the same performance as a SRS using full precision of 32 bit for each weight. Using the optimum quantizer a precision of 5 bit is sufficient to achieve the same recognition results. Even with a precision of 4 bit only a slight decrease in recognition performance occurs. Figure 4 shows the K codebook entries c_i of an optimum quantizer in comparison to those of a linear quantizer for values of $K = 16$ and $K = 32$. As can be seen from Figure 4 using an optimum quantizer small weight values are quantized more accurately than with a linear quantizer. Especially, for small K the quantization accuracy of the optimum quantizer is significantly higher than that of the linear quantizer.

4.3 Hardware-Oriented Network Topology

The RNN in the reported experiments consist of about 250 neurons which results in about 40000 connections. Actually, this amount of connections prevent a VLSI realization realization of a RNN-SRS, in particular an analog implementation. In order to take into account this hardware constraint we introduce a new type of recurrent network, the so-called Locally Recurrent Neural Network (LRNN). This is inspired by biological neural networks in a sense that LRNN possess a layered structure with sparse interconnection between the layers, and the neurons in the hidden layer are only locally connected.

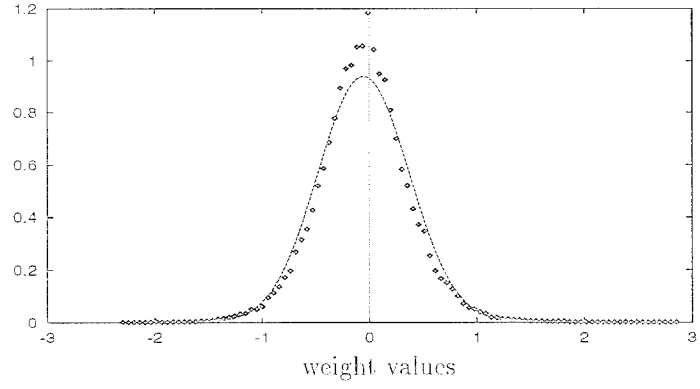


Figure 2: Distribution of weight values together with a fitted gaussian distribution

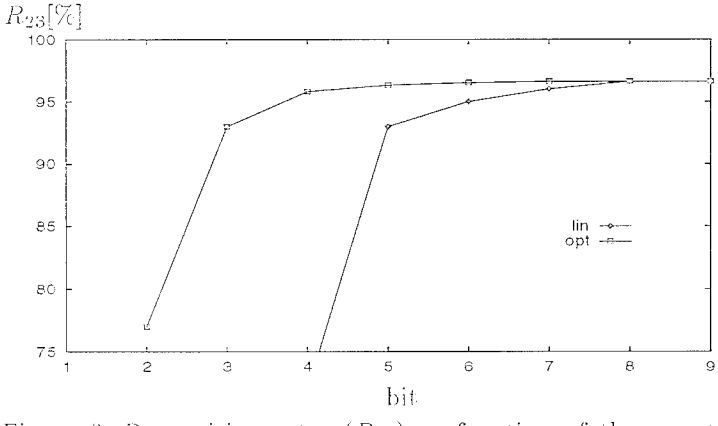


Figure 3: Recognition rates (R_{23}) as function of the quantization precision of the weight values

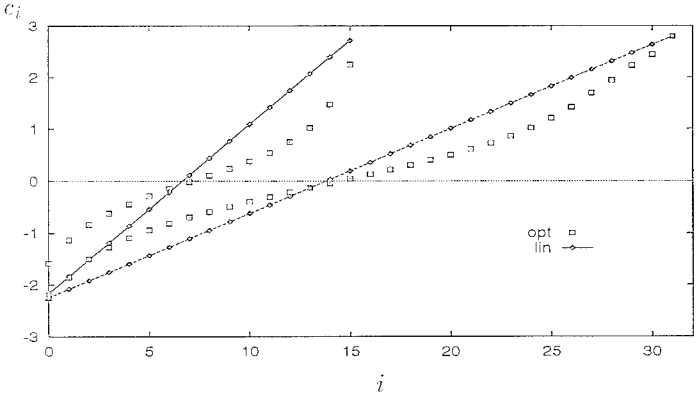


Figure 4: Codebook entries (c_i) for optimum quantizers in comparison to linear quantizers

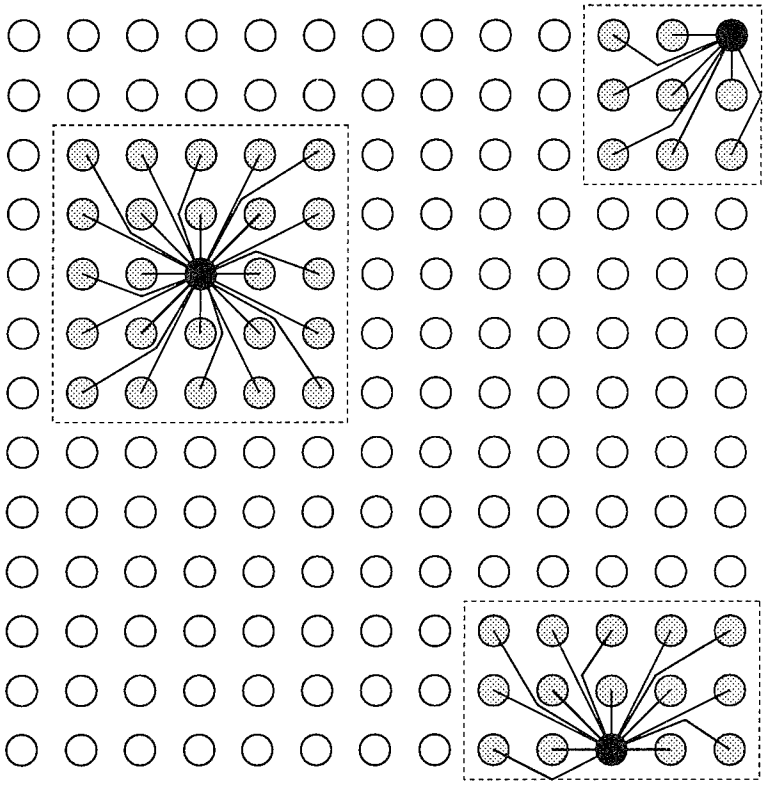


Figure 5: Connections of three exemplary neurons to their 2-nearest neighbours

A LRNN consists of an input layer, a hidden layer, and an output layer. The interactions between the input and the hidden layer as well as between the hidden and the output layer are unidirectional and sparse. The hidden layer consists of recurrent neurons without selfloops arranged in a 2 dimensional topology. The application of LRNN allows the definition of a regular structure which is suitable for hardware realizations and could not be achieved by unsupervised pruning methods. Figure 5 shows the hidden layer of a LRNN with 169 neurons placed on a 13×13 grid. A neuron at position (i, j) is connected to its $n = 2$ nearest neighbours, i.e. to all neurons with indices $([i-2, i+2]; [j-2, j+2])$. Neighbourhoods are ending at the edges of the grid.

In simulation experiments a LRNN with 60 input neurons, 169 hidden neurons, and 23 output neurons was used for realizing noise immunized SRS for the 23 word vocabulary. The neurons of the input and the output layer are connected to only one fourth of the hidden neurons. As can be seen from Table 2, a LRNN with connections to the $n = 5$ nearest

Table 2: Word recognition rates (R_{23}) for different network architectures

Architecture	n	$R_{23}(\%)$
MLP	none	78.0
LRNN	1	95.0
LRNN	2	95.5
LRNN	3	95.8
LRNN	4	96.4
LRNN	5	96.6
RNN	all	96.7

neighbours achieves with 96.6% the same recognition performance as a Fully Recurrent RNN. Even a LRNN with connections only to the $n = 1$ nearest neighbours in the hidden layer achieves with 95.0% a quite good recognition performance. It is remarkable, that a dramatic loss in recognition performance occures if all recurrencies in the hidden layer were removed and so a MLP was simulated. LRNN with connections to the $n = 5$ nearest neighbours have about 20000 connections, so the amount of connections is reduced by one half without loss of recognition performance. Even more important, the density of the connections is significantly smaller than in the case of RNN which is an important issue for a VLSI implementation.

Furthermore, it was investigated if the amount of neurons could be reduced, especially in the case of RNN. It was revealed that a significant reduction of the amount of neurons results in a dramatic decrease in recognition performance even if the amount of connections is significantly higher than 20000. This result indicates that a number of neurons are required for storing input information and internal representations of the speech. Obviously, this is essential for extracting contextual information and for compensating the variation of time duration of utterances.

CONCLUSIONS AND PERSPECTIVES

The results of the simulation experiments show that the capacity of the RNN-SRS is not limited to the recognition of a small number of isolated words. A RNN-SRS trained for recognition of telephone bandlimited digit strings with unknown length outperforms with a recognition rate of about 91% HMM based SRS. Furthermore, it was revealed that the size of the system vocabulary could be increased from 23 to 53 words with only a slightly increased network size.

The experiments concerned with aspects relevant for hardware realizations have shown that the complexity of the RNN-SRS could be significantly reduced. Considering the quantization of the weight values, 7 bit proved to be sufficient if a linear quantizer is used. If an optimum quantizer is used the quantization accuracy of the weight values could be further lowered to 5 bit. By introducing LRNN the amount of connections could be reduced by one half in comparison to RNN without loss in recognition performance. More important, LRNN avoid long distance wires in a VLSI implementation. The reported experiments show that LRNN with recurrent connections to the 5-nearest neighbours in the hidden layer are able to extract information relevant for speech recognition from noise contaminated speech and thus achieve a robust recognition performance.

Currently, a cooperation with hardware experts started in order to realize the described LRNN-SRS on a single chip. In a first step, a LRNN will be implemented. In a second step, strategies will be explored to allow a system on chip design. In this context reducing the computational complexity of the preprocessing is a crucial point. Simulation experiments have shown that using PCF as feature vectors is a promising approach for low complex preprocessing.

This work is supported by the Deutsche Forschungsgemeinschaft in the research program "System- und Schaltungstechnik für hochgradige Parallelverarbeitung".

REFERENCES

[1] Kasper,K., Reininger,H., Wolf,D., Wüst, H., "A Monolithic Speech Recognizer Based on Fully Recurrent Neural Networks", in Neural Networks for Signal Processing IV, Ermioni 1994, pp. 335-344.

[2] Kasper,K., Reininger,H., Wolf,D., Wüst, H., "A Fully Recurrent Neural Network for Recognition of Noisy Telephone Speech", in IEEE Trans. on Acoust., Speech, and Signal Processing, Detroit 1995, vol.5, pp. 3331-3334.

[3] Williams,R.J., and Zipser, D., "An Learning Algorithm for Continually Running Fully Recurrent Neural Networks", in Neural Computation 1, pp. 271-280, 1989.

[4] Brehm,H., Stammler,W., "Description and generation of spherically invariant speech model signals" in Signal Processing 12, 1987, pp.119-141.

[5] Stammler,W., "Correlation Estimation for Spherically Invariant Speech Model Signals" in International Journal of Electronics and Communications, Vol.45, 1991, No.4, pp.210-220.

AUTOMATIC SPEECH SEGMENTATION USING NEURAL TREE NETWORKS

Manish Sharma and Richard Mammone
CAIP Center, Rutgers University,
P O Box 1390, Piscataway, NJ 08855-1390.
msharma@caip.rutgers.edu

Abstract

Segmentation of speech into sub-word acoustic units using Neural Tree Networks (NTNs) is presented. NTN is a hierarchical classifier that combines the properties of both decision trees and feed-forward neural networks. The number of sub-word acoustic units in a given speech segment may or may not be known to the segmentation algorithm. Both these varieties of speech segmentation problems are addressed. The performance of the speech segmentation algorithm using NTN is compared to that obtained using Hidden Markov models (HMMs) and dynamic programming-based approach proposed elsewhere.

1 INTRODUCTION

There is a growing interest to design speech processing systems based on sub-word acoustic units. In speech recognition tasks involving large vocabularies, the phonetic content of different vocabulary words may overlap substantially. Therefore, most of the currently available speech recognition systems build models for phonetic sub-word units, and thereafter build-up word models using this inventory of sub-word models [1]. Sub-word modeling has found success in speaker recognition systems too [2, 3]. Language identification algorithms may also rely on phonetic modeling of a given language [4]. Hence, there is a growing demand to design automatic speech segmentation algorithms to phonetically (or acoustically) segment the available speech corpora.

In the present study, Neural Tree Network (NTN)-based speech segmentation algorithms will be presented. The performance will be compared against Hidden Markov model (HMM)-based ([1], Chap. 6) and dynamic programming-based [5] approaches. Although the NTNs are used in the current experiments, the proposed algorithms are general enough to accommodate any other neural network architecture.

0-7803-2739-X/95 $4.00 © 1995 IEEE

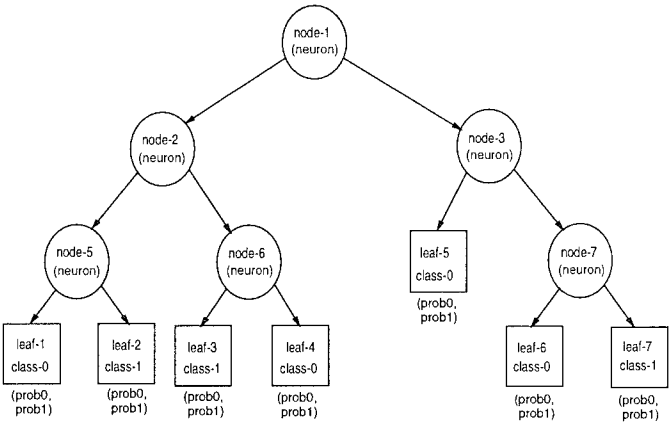


Figure 1: A (binary) neural tree network (NTN)

1.1 Neural Tree Network (NTN)

The neural tree network (NTN) [6] is a hierarchical classifier that combines the properties of feed-forward neural networks and a decision tree type structure. The NTN uses a tree-structure of discriminating elements, with the discriminants being implemented using artificial neurons. The neuron uses all feature elements for the decision, hence the NTN is not constrained to perpendicular discriminant boundaries as are the standard decision trees. The architecture of the NTN is determined during training, and hence the NTN is self-organizing. This is in contrast to multi-layer perceptron (MLP) neural networks, where the architecture must be specified prior to training. Figure 1 shows a binary NTN. A binary NTN is used for a two class classification problem.

Recently, some modifications to the original NTN have been proposed to facilitate the estimation of the posterior class probability given a data vector. A Modified Neural Tree Network (MNTN) was introduced in [7]. Forward pruning technique was used while growing the tree. Since, the leaves of the pruned tree are likely to have data from several classes, a *"confidence"* measure is associated with each class. For illustration, if the leaf of a binary classification NTN has N_0 and N_1 vectors corresponding to *class-0* and *class-1* respectively, then the *"confidence"* of each class will be given as

$$c_0 = \frac{N_0}{N_0 + N_1} \qquad c_1 = \frac{N_1}{N_0 + N_1} \tag{1}$$

The *"confidence"* measure in MNTN is equivalent to estimating the posterior probability using Parzens-window method in the region defined by a leaf. More recently, Continuous Density NTN (CDNTN) has been proposed [8], wherein local parametric models (generally, mixture of gaussian) are created for each class at every leaf of a NTN.

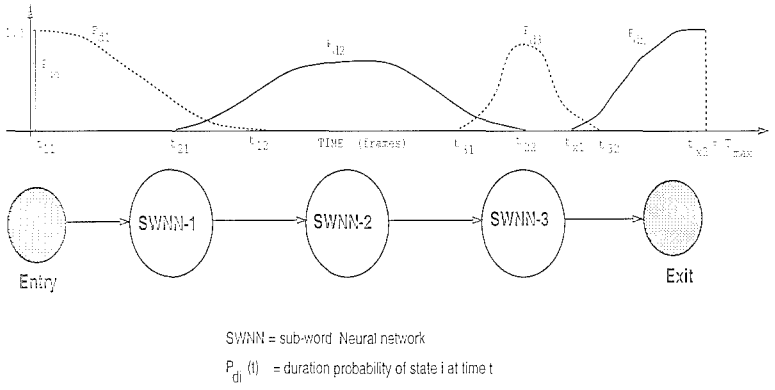


Figure 2: Typical word model with NNs modeling the sub-word segments and their associated durational probability functions

2 AUTOMATIC SPEECH SEGMENTATION

The problem of automatic speech segmentation can be posed under two different scenarios. In the first kind, the phonetic transcription, or alternately the number of phonemes present in a given speech data, is known. The segmentation algorithm is just required to optimally locate the sub-word segment boundaries. The sub-word units thus segmented are generally called the phone-like units (PLUs). In the other kind, however, there is no linguistic knowledge about the given speech data. The segmentation algorithm, therefore, is required to determine the optimal number of sub-word units present, as well as their boundary locations, based on the acoustic cues only. The sub-word units thus obtained may be termed as acoustic segment units (ASUs), since these units are segmented on the basis of *acoustic homogeneity* criteria. In the present study, both the types of segmentation problems have been dealt with. Without any loss in generality, the term "word" will be used in subsequent discussions to mean a given speech data for segmentation.

2.1 Segmentation with known number of sub-word units

2.1.1 Proposed algorithm using NTN

Word modeling using NTN

In the proposed algorithm, a word is modeled as a sequence of sub-word neural tree networks (SWNTNs). A typical word model is shown in figure 2. A duration probability function is associated with each SWNTN. The number of SWNTNs in a word model is set to be equal to the number of phonemes present in the *phonetic spelling* of the word. Each SWNTN effectively models a phoneme in the given word. The proposed word model is similar to a whole-

word Hidden Markov model, with the Markovian states being replaced by the NTNs [9]. For a tutorial on hybrid Neural network & Hidden Markov model systems, refer to [10].

Word model scoring

Consider the task of scoring the model λ, given a sequence of observation vectors $\mathbf{O} = \{\mathbf{x}_1, \cdots, \mathbf{x}_t, \cdots, \mathbf{x}_T\}$. Assume that the output of the SWNTNs are normalized so as to yield probabilistic values [8]. The probabilistic output b_i of the SWNTN q_i can then be interpreted as the *a posterior* probability $P(\mathbf{x}_t/q_i, \lambda)$ of the observation vector $\mathbf{x}_t$, at time t, given the model λ and SWNTN q_i. The durational probability d_i can be argued to represent the probability $P(q_i/\lambda)$ of being in state q_i, at time t. The observation probability $P(\mathbf{x}_t/\lambda)$ of vector $\mathbf{x}_t$ can then be written as:

$$P(\mathbf{x}_t/\lambda) = \sum_i P(\mathbf{x}_t/q_i, \lambda) P(q_i/\lambda) = \sum_i (b_i d_i) \tag{2}$$

The above equation can be approximated by considering only the maximum value within the summation,

$$P(\mathbf{x}_t/\lambda) = \max_i (b_i d_i) \tag{3}$$

If it can be assumed that the successive observations are independent, then the joint probability of the sequence of events can be written as the product of the probability of the individual events.

$$P(\mathbf{O}/\lambda) = P(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T/\lambda_v) = \prod_{j=1}^{T} P(\mathbf{x}_j/\lambda) \tag{4}$$

Equation 4 gives the probability score of an observation sequence $\mathbf{O}$, given the word model λ. Equations 3 and 4 can be easily implemented using dynamic programming techniques (Viterbi algorithm).

Iterative segmentation algorithm

The segmentation of each word is accomplished by using an iterative algorithm. Given an initial word model λ, the algorithm consists of two fundamental steps: (1) segmentation, and (2) re-estimation. The segmentation is carried out by forced alignment using the standard Viterbi algorithm. The optimal sequence of SWNTNs is decoded by backtracking the Viterbi path. The new SWNTNs are now trained based on the new segmentation. Replacing the old word model λ by the new model $\overline{\lambda}$, the two steps are executed iteratively until convergence is reached. The pseudo-code of the algorithm is outlined in figure 3. The basic concepts underlying this iterative algorithm is similar to the Segmental K-means algorithm used in Hidden Markov modeling [11].

initialization:
set max-iteration, min-label-change
get initial segmentation
while *iteration* < *max-iteration*
Train sub-word neural networks
Compute sub-word durational probabilities
Resegment sub-word boundaries (forced alignment using Viterbi)
if *label-change* < *min-label-change*
break
endif
end-while

Figure 3: NTN-based speech segmentation algorithm

2.1.2 Hidden Markov model-based segmentation

Hidden Markov models are stochastic, signal generative models and have been found to be very successful for temporal pattern recognition problems, including, speech recognition ([1], Chap. 6). If a HMM word model is trained with its number of states equal to the number of phonemes in that word, then the alignment of the observation vectors at the state-level can be interpreted as the phonetic segmentation of the word.

2.1.3 Dynamic programming-based segmentation

In [5], the sub-word segment boundaries within a word were found by the use of level-building dynamic programming procedure. The sub-word segment boundaries are found by minimizing the distortion for all possible segmentations of the observation sequence.

2.2 Experiments

The male speakers from the north midland dialect region of the TIMIT speech corpus were selected for the experiments. With the help of the given time-aligned word transcriptions, five words, namely, *dark*, *suit*, *greasy*, *water* and *year*, were extracted from the "SA1"-type sentence of these speakers. Forty utterances of each word were thus obtained. Since the time-aligned phonetic transcriptions are also supplied along with the database, the automatic segmentation obtained by the segmentation procedures can be compared against it. The measure of performance is defined in terms of the *coincidence rate*. The coincidence rate can be found as the percentage of phonetic boundaries, automatically obtained by the segmentation algorithm, that are within x milliseconds of the true boundary locations.

- To obtain segmentation using sub-word NTN-based modeling, the iterative algorithm outlined above in figure 3 was used. The initial segmentation for the iterative algorithm was obtained by equally segmenting the word into required sub-word units.

	NTN	HMM	Dynamic prog.
boundary diff = 25msec	66.6 %	65.7 %	70.9 %
boundary diff = 20msec	56.4 %	54.6 %	61.2 %

Table 1: Speech segmentation coincidence rates

- A continuous density HMM tool-kit (Entropic Lab's HTK software) was used to get the HMM-based segmentation. A whole-word model was trained for each of the above five words, setting the number of states equal to the number of phonemes present in them. Each state in the HMMs was modeled by a single mixture, continuous density gaussian distribution.
- The dynamic programming-based segmentation was obtained by delivering each of the forty utterances of a word, one at a time, to the algorithm mentioned in section 2.1.3.

The speech was analyzed in 32 millisecond frames, with 6.25 millisecond shift between consecutive frames. The spectral features used for the experiments were 12^{th} order linear prediction-derived cepstral coefficients. The experimental results obtained are shown in table 1.

It is conjectured that since the dynamic programming-based segmentation algorithm works on one utterance at a time (as opposed to NTN or HMM -based approaches which use all the utterances of a word at one time), it does not suffer from the deficiency of generalization to different pronunciations of a word by different speakers. This appears to be the possible reason behind its success over HMM and NTN techniques. A small experiment was conducted to verify this conjecture. If the dynamic programming-based segmentation is performed by delivering to the algorithm all the utterances of a word at one time, the coincidence rate (for boundary difference = 20msec) drops from 61.2% to 52.3 %.

2.3 Segmentation with unknown number of sub-word units

Problem formulation : It is desired to segment given sequences of acoustic observation vectors of a word into its constituent sub-word units. The segmented sub-word regions have to be *acoustically homogeneous.* No linguistic knowledge is assumed to be available, and hence the number of sub-word units that would occur in the given word is also unknown.

2.3.1 Proposed algorithm using NTN

The algorithm was first introduced by the authors in [12]. The proposed method can be thought of as being *"self-evolving"*, in the sense that, it progressively segments the given word frames into constituent sub-word units, one unit at a time, until all the frames are exhausted.

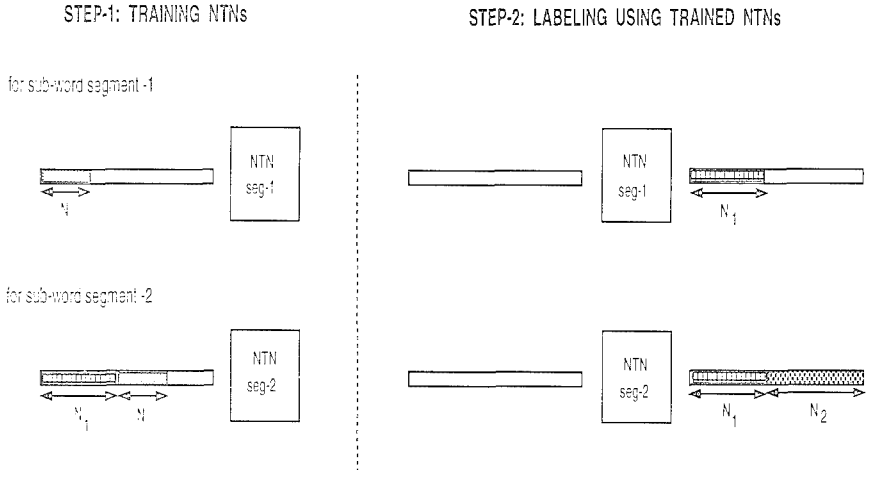


Figure 4: Segmentation with unknown number of sub-word units using NTN

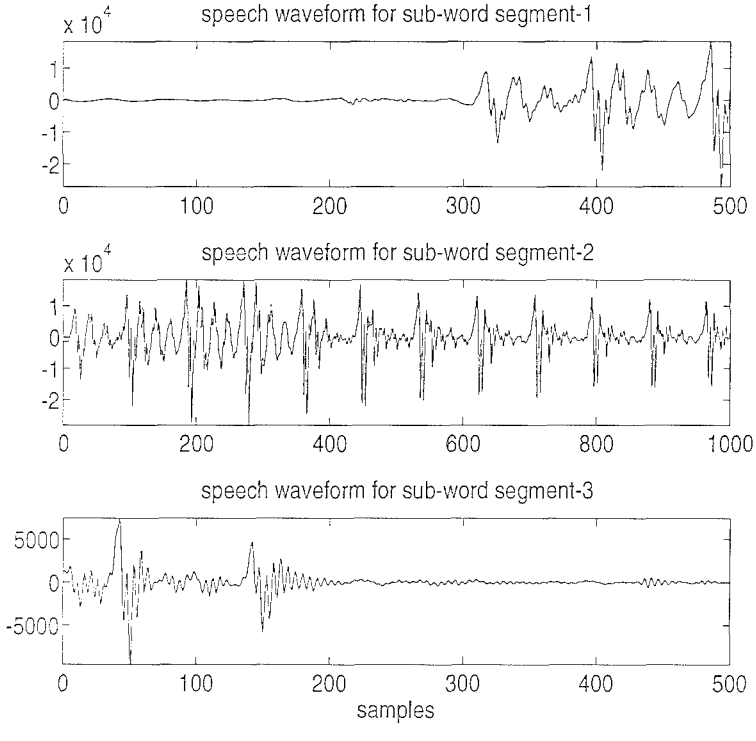


Figure 5: Segmented waveforms of the first three sub-word units of the word "*back-track*"

To initiate the algorithm, the first N frames from each observation sequence are marked as the *"seed"* frames for sub-word segment S_1. A NTN for sub-word segment S_1 is trained using these *"seed"* frames as class exemplars. The observation sequences are then classified (labeled) using the trained NTNs to find the actual N_1 frames that get classified as "in-class". These frames are presumed to be truly belonging to the sub-word segment S_1. It is conjectured that the NTN shall generalize from its training on the *"seed"* frames to classify all other similar vectors in the observation sequence as "in-class". The training for the next sub-word segment S_2 is similar to that outline above for sub-word segment S_1, except that its *"seed"* frames would now be the next N frames beyond the previously estimated sub-word segment S_1 boundary. The procedure is repeated until all the frames of the given observation sequence are exhausted. The training procedure for a word consisting of two sub-word segments is illustrated in figure 4.

The segmented waveforms obtained by running the above algorithm on an instance of word *"backtrack"* is shown in figure 5. Only the first three sub-word segments are shown in the figure, and they depict reasonable *acoustic homogeneity.*

3 CONCLUSIONS

Automatic speech segmentation algorithms using Neural Tree network (NTN) were presented. Two different types of segmentation problems were posed depending on whether the number of sub-word units present in the speech segment is known apriori or not. The proposed NTN-based algorithm performed comparably to the HMM-based segmentation technique. Preliminary findings are encouraging and the proposed approaches will be continued to be further investigated.

Acknowledgment

This research was partly supported through a grant (F30602-91-C-0120) from the U.S. Air Force at Rome Laboratories.

References

[1] B-H Juang and L.R. Rabiner. *Fundamentals of Speech Recognition.* Prentice-Hall, NJ, 1992.

[2] A.E. Rosenberg, C-H Lee, and F.K. Soong. Sub-word unit Talker Verification using Hidden Markov models. In *Proceedings of ICASSP*, 1990.

[3] T. Matsui and S. Furui. Speaker adaptation of tied-mixture-based phoneme models for text-prompted Speaker Recognition. In *Proceedings of ICASSP*, 1994.

[4] Y.K. Muthusamy, E. Barnard, and R.A. Cole. Reviewing Automatic Language Identification. *IEEE Signal Processing magazine*, 11(4):33–41, October 1994.

[5] T. Svendsen and F. Soong. On the automatic segmentation of speech signals. In *Proceedings of ICASSP*, 1987.

[6] A. Sankar and R.J. Mammone. Growing and Pruning Neural Tree Networks. *IEEE Transactions on Computers*, (C-42):221–229, March 1993.

[7] K.R. Farrell, R.J. Mammone, and K.T. Assaleh. Speaker Recognition using Neural Networks and conventional Classifiers. *IEEE Trans. on Speech and Audio Proc.*, 2(1):194–205, January 1994.

[8] S. Kosonocky and R. Mammone. A Continuous Density Neural Tree Network Word spotting system. In *Proceedings of ICASSP*, 1995.

[9] M. Sharma and R. Mammone. Speech recognition using sub-word Neural Tree network models and multiple classifier fusion. In *Proceedings of ICASSP*, 1995.

[10] N. Morgan and H. Bourlard. Neural Networks for Statistical Recognition of Continuous Speech. *Proceedings of the IEEE*, 83(5):742–770, May 1995.

[11] B-H Juang and L.R. Rabiner. The Segmental K-means algorithm for estimating parameters of Hidden Markov models. *IEEE Trans. on Acoustics, Speech and Signal proc.*, 38(9):1639–1641, September 1990.

[12] M. Sharma and R. Mammone. Neural Tree Network for speech segmentation into sub-word acoustic units. *Conference on Neural networks and Artificial Intelligence, SPIE Proceedings*, 2093:2–7, 1993.

Image Processing and Computer Vision

Motion Estimation and Segmentation Using a Recurrent Mixture of Experts Architecture

Yair Weiss and Edward H. Adelson
Dept. of Brain and Cognitive Sciences and the Media Lab
Massachusetts Institute of Technology
E15-384, Cambridge, MA 02139, USA
{yweiss,adelson}@media.mit.edu

Abstract

Estimating motion in scenes containing multiple motions remains a difficult problem for computer vision. Here we describe a novel recurrent network architecture which solves this problem by simultaneously estimating motion and segmenting the scene. The network is comprised of locally connected units which carry out simple calculations in parallel. We present simulation results illustrating the successful motion estimation and rapid convergence of the network on real image sequences.

1 Introduction

Motion estimation is an ill-posed problem. In other words, local motion measurements are inherently ambiguous. When the scene contains only one smoothly varying motion the ill posedness can be overcome by imposing a smoothness constraint on the solution (e.g. (Poggio et al., 1985)). The smoothness assumption, however, is not valid when the scene contains multiple motions, and imposing it leads to erroneous motion estimates especially at occlusion boundaries (e.g. (Horn, 1986)). One way to modify the smoothness assumption is to estimate motion discontinuities via line processes and disable motion smoothing across the line processes (Terzopoulos, 1986; Hutchinson et al., 1988). These algorithms are notoriously slow to converge, and more importantly they produce a representation which is ill suited for dealing with scenes containing occlusion, such as a scene showing a cat walking behind a fence. Motion discontinuities can capture the fact

0-7803-2739-X/95 $4.00 © 1995 IEEE

that the cat fragments and the fence posts are not moving together, but they can not capture the fact that the cat fragments move together. In contrast, the representation we are interested in computing explicitly groups the fragments together (Wang and Adelson, 1994; Darrell and Pentland, 1991; Black and Anandan, 1993).

2 Architecture

Our architecture is based on the "divide and conquer" modularity principle (Jordan and Jacobs, 1994). Rather than have one network estimate motion everywhere, we have multiple *motion expert* subnetworks competing to explain the data by minimizing *motion error*. The error signal to these expert subnetworks is controlled by a *gating* subnetwork which assigns different regions of space to different experts. The advantage of this approach is that it restores the validity of the smoothness assumption: regions undergoing drastically different motions are assigned to different experts, and the motion of regions assigned to a specific expert is indeed smoothly varying. The network simultaneously estimates the motions and the assignments. The assignment is based on two factors: (1) which expert is currently doing a better job of explaining the motion data, and (2) the current assignment of nearby regions having similar intensities. As shown below this simultaneous estimation and segmentation is accomplished using simple parallel updates.

The architecture and flow of information are depicted schematically in figure 1. The motion expert subnetwork is comprised of K sheets of retinotopically organized units (K represents the maximum number of motions in the scene). Each sheet contains units tuned for a specific velocity at a particular retinal location (cf. (Bulthoff et al., 1989)). The distribution of responses of all velocity tuned units at a given location represents the velocity estimate of the motion expert. The input to the motion experts comes from the motion error subnetwork, which also contains units tuned for a specific velocity at a particular retinal location. The exact form of these motion selective units is irrelevant, as long as they represent the local deviation from coherent motion in a given velocity. The calculation can be based on correlation as in (Bulthoff et al., 1989) or motion energy as in (Simoncelli et al.,

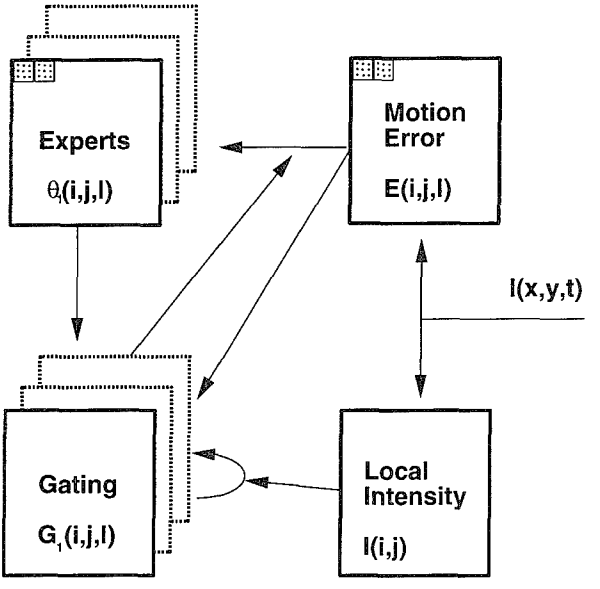


Figure 1: A schematic depiction of the information flow in the network. In accordance with the "divide and conquer" modularity principle, we have multiple motion expert subnetworks competing to explain the data by minimizing motion error. The error signal to the expert networks is modulated by a gating subnetwork which assigns different regions of space to different experts. The assignment is based on the motion error of each expert's estimate as well as the current assignment of neighboring regions with similar intensities. The simultaneous estimation of motion and assignments is accomplished by retinotopically organized units which carry out simple operations in parallel

1991). The input from the motion error to a sheet in the motion experts subnetwork is modulated by a corresponding sheet in the gating subnetwork. The gating subnetwork, in turn, receives input from the experts and motion error subnetworks as well as a *local intensity* subnetwork which modulates the cooperation of nearby gating network units.

3 Dynamics

We denote by $\theta_k(i,j,l)$ the activity of a unit in the kth sheet of the motion experts network at grid location i,j tuned to velocity l, and by $E(i,j,l)$ the activity of a unit in the motion error network. Similarly, we denote by $G_k(i,j)$ the activity of unit i,j in the kth sheet of the gating network and by $I(i,j)$ the activity of a unit in the local intensity network. To emphasize the connection to the EM algorithm we call the dynamics of the gating and expert networks the E and M dynamics respectively.

E dynamics The gating units are updated by a weighted summation of inputs followed by a normalizing nonlinearity:

$$G_k(i,j) := \frac{exp(\tilde{G}_k(i,j))}{\sum_o exp(\tilde{G}_o(i,j))} \tag{1}$$

With:

$$\tilde{G}_k(i,j) = 1/\beta_2 \sum_l E(i,j,l)\theta_k(i,j,l) - \eta \sum_{m,n} \alpha_{ij}^{mn} G_k(m,n) \tag{2}$$

M dynamics Similarly the motion expert units are updated by a weighted summation of inputs followed by a normalizing nonlinearity:

$$\theta_k(i,j,l) := \frac{exp(\tilde{\theta}_k(i,j,l))}{\sum_o exp(\tilde{\theta}_k(i,j,o))} \tag{3}$$

With:

$$\tilde{\theta}_k(i,j,l) = 1/\beta_1 \sum_{m,n} w_{ij}^{mn} G_k(m,n) E(i,j,l) \tag{4}$$

Where w_{ij}^{mn} is a Gaussian window and α_{ij}^{mn} is a Gaussian window modulated by the local intensity network.

4 Energy Function

The dynamics can be derived from the following energy function:

$$\begin{aligned} J(\theta, G; E) &= \sum_{kijlmn} w_{ij}^{mn} G_k(m,n)\theta_k(i,j,l)E(m,n,l) \\ &\quad -\eta \sum_{ijmnk} \alpha_{ij}^{mn} G_k(i,j)G_k(m,n) \\ &\quad +\beta_1 \sum_{ijk} G_k(i,j)\log G_k(i,j) \\ &\quad +\beta_2 \sum_{ijlk} \theta_k(i,j,l)\log \theta_k(i,j,l) \end{aligned} \tag{5}$$

To understand the justification for this energy function, consider the expression:

$$J_{ij}(\theta_k) = \sum_l \theta_k(i,j,l)E(i,j,l) \tag{6}$$

Recall that $E(i,j,l)$ measures the motion error at location i,j and hence the higher the motion error for a velocity the higher the penalty for a unit with that preferred velocity to be active. Due to the ill-posedness of the motion estimation problem, $J_{ij}(\theta_k)$ will have multiple minima. Therefore a smoothness constraint may be imposed via a larger integration window (as in (Lucas and Kanade, 1981)) :

$$J_{ij}(\theta_k) = \sum_l \sum_{m,n} w_{ij}^{mn} \theta_k(i,j,l)E(m,n,l) \tag{7}$$

But a large integration window is likely to contain multiple motions. Hence we gate the errors to the kth expert by G_k:

$$J_{ij}(\theta_k) = \sum_l \sum_{m,n} w_{ij}^{mn} G_k(m,n)\theta_k(i,j,l)E(m,n,l) \tag{8}$$

This gives the first term of the energy function. The second term reflects the fact that nearby points having similar intensities should be assigned to the same expert. Finally the last two terms (the entropies) penalize for distributions where only one unit is active: omitting these terms causes the softmin function in equations 3 and 1 to be replaced by a "hard" winner take all function.

It is easy to show that constrained minimization of the energy function with respect to $\theta_k(i,j,l)$ gives the M dynamics. Minimizing the function with respect to $G_k(ij)$ gives a slightly different

Figure 2: A frame from a sequence presented to the network. The sequence shows a person moving behind a plant.

version of the E dynamics with the term $E(i,j,l)\theta_k(i,j,l)$ replaced by the weighted average $E(i,j,l)\sum w_{ij}^{mn}\theta_k(m,n,l)$. Note, however that the velocity fields of each expert are by construction smooth. Hence, this sum is well approximated by the term $E(i,j,l)\theta_k(i,j,l)$ and the E dynamics can be viewed as an approximate minimization. In practice we have found that the approximate solution works as well as the exact one.

5 Simulation Results

The performance of the network on a real image pair is illustrated in figures 2 and 3. An important parameter in our network is the number of velocity tuned units assumed to exist at every location, i.e. the sampling used in discretizing velocity space. In the simulations reported here, we assumed that the sampling is sufficiently dense such that the distribution of unit activity approximates a continuous function. As a measure of motion error we used the gradient constraint (cf. (Horn, 1986)):

$$E(i,j,l) = (dx^t V_l + dt)^2 \tag{9}$$

Where dx, dt denote the temporal and spatial derivatives at location i,j respectively. Note that this expression is quadratic in V_l. Thus the term $\tilde{\theta}_k(i,j,l)$ in equation 4 is also quadratic in V_l and equation 3 can be evaluated analytically.

Figure 2 shows one frame from a sequence showing a person moving behind a plant. Figure 3 shows the activity in the network

as a function of time. On the left is shown the activity in a sheet of the gating subnetwork. The grey level represents the probability that a pixel be assigned to one of the experts: white regions are confidently assigned, black regions are confidently rejected and grey regions can be equally assigned to both experts (these are regions where there is no motion information). As can be seen, the network converges rapidly to a correct motion estimate and segmentation.

6 Discussion

The energy function in equation 5 is, of course, not the only possible one to use as a cost function for motion estimation and integration. In related work (Weiss and Adelson, 1994) we have experimented with other cost functions. The common feature of the various functions we have explored is that they contain the following three terms:

- a term measuring the local prediction error, i.e. how well does the expert to whom this pixel is assigned predict the local motion measurements.
- a term rewarding coherence of the motion fields of each expert.
- a term rewarding coherence of the assignments. i.e, rewarding assignments in which neighboring pixels of similar intensities are assigned to the same expert.

It is the third term that differentiates our work from many computer vision algorithms for motion segmentation. We believe that the integration of form and motion cues for segmentation is crucial. In our current work we are studying ways to improve this integration by having perceptual organization cues, rather than simple local intensity modulate the local interconnections in the gating network.

A neural net model which also includes gating of motion energy units has been recently suggested by (Nowlan and Sejnowski, 1993). However, their model, unlike the one presented here, does not compute segmentation or grouping. In their algorithm, the gating units are trained off-line and essentially learn to suppress measurements centered on motion boundaries.

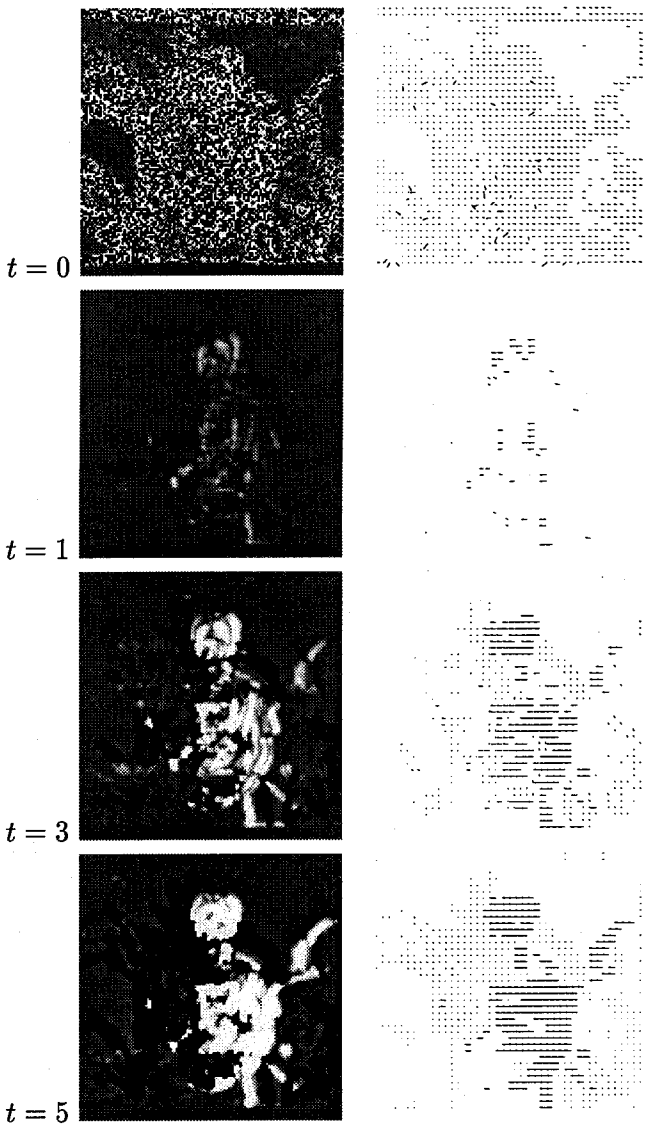


Figure 3: The activity of sheet one of the gating network $G_1(i, j)$ (left) and the estimated flow (right) as a function of time, starting with random initial conditions

The network we have been using is closely related to the EM algorithm for mixture estimation studied by (Jordan and Jacobs, 1994). The main difference is that in their mixture of experts network the experts and the gating networks are assumed to be generalized linear models. This serves to keep the number of parameters estimated significantly smaller than the number of measurements. Here we keep the large number of parameters to estimate (which enables us to segment arbitrarily shaped regions) and add additional smoothness constraints on both the gating parameters and the motion parameters. A second difference between our work and that of Jordan and Jacobs is our emphasis on parallel implementation. Unlike the general mixture estimation problem, motion segmentation has the feature that all measurements are typically acquired simultaneously. One is tempted therefore to look for algorithms that can be implemented in hardware by retinotopic units performing simple operations in parallel. As our simulation results show, units of this type can collectively produce rapid and accurate motion estimation and segmentation.

References

Black, M. J. and Anandan, P. (1993). The robust estimation of multiple motions: affine and piecewise smooth fields. Technical Report spl-93-092, Xerox PARC.

Bulthoff, H., Little, J., and Poggio, T. (1989). A parallel algorithm for real-time computation of optical flow. *Nature*, 337(6207):549–553.

Darrell, T. and Pentland, A. (1991). Robust estimation of a multi-layered motion representation. In *Proc. IEEE Workshop on Visual Motion*, pages 173–178, Princeton, New Jersey.

Horn, B. K. P. (1986). *Robot Vision.* The MIT Press, Cambridge, MA.

Hutchinson, J., Koch, C., Luo, J., and Mead, C. (1988). Computing motion using analog and binary resistive networks. *IEEE Computer magazine*, 21:52–64.

Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6:181–214.

Lucas, B. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Image Understanding Workshop*, pages 121–130.

Nowlan, S. and Sejnowski, T. (1993). Filter selection model for generating visual motion signals. In Hanson, S., Cowan, J., and Giles,

C., editors, *Advances in Neural Information Processing Systems 5*, pages 369–376.

Poggio, T., Torre, V., and Koch, C. (1985). Computational vision and regularization theory. *Nature*, 317:314–319.

Simoncelli, E., Adelson, E., and Heeger, D. (1991). Probability distributions of optical flow. In *Proc. IEEE Conf. Comput. Vision Pattern Recog.*, pages 310–315.

Terzopoulos, D. (1986). Regularization of inverse visual problems involving discontinuities. *IEEE Trans. PAMI*, 8:413–424.

Wang, J. Y. A. and Adelson, E. H. (1994). Representing moving images with layers. *IEEE Transactions on Image Processing Special Issue: Image Sequence Compression*, 3(5):625–638.

Weiss, Y. and Adelson, E. H. (1994). Perceptually organized EM: a framework for motion segmentation that combines information about form and motion. Technical Report 315, MIT Media Lab.

USING PERCEPTRON-LIKE ALGORITHMS FOR THE ANALYSIS AND THE PARAMETERIZATION OF OBJECT MOTION

Marco Mattavelli§ and Edoardo Amaldi†[1]
§Signal Processing Laboratory, Swiss Federal Institute of Technology,
CH-1015 Lausanne, Switzerland
marco@lts.de.epfl.ch
†School of Operations Research and Center for Applied Mathematics,
Cornell University, Ithaca, NY 14853, USA
amaldi@cs.cornell.edu

ABSTRACT

A new approach based on the extraction of maximum consistent subsystems of linear systems is proposed for addressing the general problem of determining the linear motion parameters of unknown moving objects from a sequence of images. This type of task can be tackled using simple but effective variants of the well-known perceptron algorithm that aim at maximizing the number of patterns that are correctly classified. Unlike in the usual perceptron applications, the weight vectors determined during the training phase are not used to classify new patterns but to extract the structure and to provide the parameters of the considered piecewise linear model. The potentialities of the new approach are demonstrated for the segmentation of the optical flow. Experimental results obtained for fields from synthetic and natural images indicate various advantages of our approach with respect to some classical alternatives.

1. INTRODUCTION

The extraction and interpretation of the motion of the sensor and of independently moving objects from visual information is a fundamental problem in image processing and computer vision. Two classes of approaches have been proposed in the literature: one is based on the computation and interpretation of the optical flow [1, 2, 3, 4] while the other extracts motion parameters directly from the image sequences and leads to a layered representation of object motions [5, 6]. The major problem of the first class of approaches is how to combine local features in order to extract global object motions. Since flow fields are usually noisy and partially incorrect, especially near occlusions and motion boundaries, global approaches are considered necessary in order

[1]E. A. is partially supported by a Postdoctoral fellowship of the Swiss National Science Foundation.

0-7803-2739-X/95 $4.00 © 1995 IEEE

to yield robust parameter estimates. But the delicate choice of the objective function and the presence of several moving objects make these techniques complex and unsatisfactory. For the second class of approaches, the difficulty lies in partitioning the data into a consistent layered representation. Unless the segmentation is already correct, a set of pixels being fitted may contain multiple motions instead of a single dominant coherent component.
Methods that combine and iterate the two types of techniques have also been proposed in order to achieve more accurate representations and interpretations of the optical flow [7]. A common difficulty of both classes of methods is the *chicken and egg problem* of the parameter and structure estimation. If the structure is known, it is relatively easy to correctly estimate the parameters. But if the parameters are not known it is difficult to correctly extract the structure. This kind of dilemma, which pertains to several *ill-posed* problems, assumes a fundamental importance in the analysis and interpretation of motion data.
In this paper we address the problem of determining an unknown linear motion structure (several independent moving objects) with unknown parameters from a sequence of images. A new approach based on the extraction of maximum size consistent subsystems of linear systems is proposed for the segmentation of the optical flow. The reported results indicate that a simple perceptron-like algorithm can efficiently extract the motion parameters and the motion structure without requiring a dominant motion or any *a priori* hypothesis on the number and on the spatial location of the moving objects.

The paper is organized as follows. Next section describes how the optical flow segmentation problem can be formulated in terms of finding maximum consistent subsystems of linear systems. In section 3 we briefly recall the thermal variant of the perceptron algorithm [8, 9] and show how it can be used to determine maximum consistent subsystems of linear equations. Section 4 points out the major shortcomings of alternative approaches such as robust estimation [10, 11] and the Hough transform [12, 13]. A few typical experimental results on synthetic optical flow fields are reported in section 5. Finally, some concluding remarks are mentioned in section 6.

2. PROBLEM FORMULATION

Consider an image sequence that evolves in time with the apparent motion given by equation:

$$I(x, y, t) = I(x - d_x(x, y), y - d_y(x, y), t - 1), \tag{1}$$

where x and y are the image coordinates, $I(\cdot)$ is the image intensity and $d_x(x, y)$ and $d_y(x, y)$ denote the image displacements at each point. When the distances between the object surfaces and the camera are large, it is usually possible to approximate the motion as an affine transformation:

$$d_x(x, y) = w_1 + w_2 x + w_3 y$$

$$d_y(x,y) \quad = \quad w_4 + w_5 x + w_6 y \tag{2}$$

for every image point (x, y).
Let us assume that the optical flow has been estimated using a standard technique [1], i.e. we have a couple of components $(d_x(x_k, y_k), d_y(x_k, y_k))$ for n image points (x_k, y_k) with $k \in \{1, 2, ..., n\}$. The problem of the interpretation of the optical flow arising from object motion is to segment the field into an unknown number r of regions that correspond to the same affine motion and to estimate the corresponding set of parameters $w_1^l ....w_6^l$, where $l \in \{1, 2, ..., r\}$. Consider the linear system consisting of the n pairs of linear equations (2) in the six variables $w_1 ....w_6$. For multiple motions and reliable optical flow estimates, the overall system is inconsistent and the regions with the same affine motion correspond to consistent subsystems. Formally, the optical flow segmentation problem can be formulated as that of partitioning the system of $2n$ equations (2) into r disjoint consistent subsystems and of finding a set of corresponding solutions $w_1^l, ...w_6^l$ with $l \in 1, 2, ..., r$. Since all pairs of points with the same affine motion should be associated to the same region, one looks for maximum consistent subsystems, i.e. subsystems containing the largest number of pairs of equations (2). In order to partition the image into a small number of regions, a simple greedy strategy is adopted in which as large as possible consistent subsystems are removed iteratively.

3. PERCEPTRON-LIKE ALGORITHMS AND OPTICAL FLOW SEGMENTATION

A *perceptron* is a simple linear threshold unit whose output

$$y = \begin{cases} +1 & \text{if } \sum_{j=1}^{n} w_j x_j \geq 0 \\ -1 & \text{otherwise,} \end{cases}$$

where $x_j \in \Re$ denotes the jth input and $w_j \in \Re$ the jth weight. Given a task, i.e. a set of p input vectors $\mathbf{a}^k$, $1 \leq k \leq p$, and the corresponding desired output $b^k \in \{-1, 1\}$, the objective of *training* is to find a weight vector $\mathbf{w}$ that classifies all $\mathbf{a}^k$ as well as possible. This amounts to finding a $\mathbf{w}$ that satisfies as well as possible the following linear system:

$$\begin{array}{l} \mathbf{a}^k \, \mathbf{w} \geq 0 \;\; \forall k \text{ such that } \; b^k = 1 \\ \mathbf{a}^k \, \mathbf{w} < 0 \;\; \forall k \text{ such that } \; b^k = -1. \end{array} \tag{3}$$

If the task is linearly separable, the well-known perceptron algorithm [14] is guaranteed to yield in finite time a weight vector that correctly classifies all $\mathbf{a}^k$. The procedure is very simple: start with an arbitrary initial vector $\mathbf{w}_0$, randomly select a sequence of input vectors $\mathbf{a}^k$ and, at each iteration, update the current vector $\mathbf{w}_i$ as follows:

$$\mathbf{w}_{i+1} := \begin{cases} \mathbf{w}_i + \eta_i \; b^k \mathbf{a}^k & \text{if } \mathbf{a}^k \text{ is misclassified by } \mathbf{w}_i \\ \mathbf{w}_i & \text{otherwise,} \end{cases} \tag{4}$$

where η_i is the gain parameter. In fact, when (3) admits a solution, it can be find in polynomial time using an appropriate method for linear programming [15]. In the frequent case where the task is not linearly separable, the best we can ask for is an *optimal* weight vector which correctly classifies as many input vectors as possible. As we have shown in [16, 17], finding an optimal solution turns out to be harder than finding a solution of (3) when it exists. More precisely, the problem is $\mathcal{NP}$-hard and cannot be approximated in polynomial time within every constant factor.
Since the perceptron algorithm never converges for nonlinearly separable tasks, several variants have been proposed and studied for finding optimal weight vectors [8, 9]. In terms of linear systems, these methods aim at finding maximum consistent subsystems of the associated system (3).
The basic idea of the thermal perceptron algorithm [8] is to favor weight updates that aim at correcting errors whose total input $v^k = b^k \mathbf{a}^k \mathbf{w}$ is relatively close to zero. Indeed, since v^k is proportional to the distance from the input to the hyperplane $H = \{\mathbf{x} \in \Re^n | \mathbf{w}\mathbf{x} = 0\}$ defined by the current $\mathbf{w}$, the large weight changes that would be required to correct errors with a large $|v^k|$ are likely to corrupt other well classified $\mathbf{a}^k$. Therefore, the magnitude of the weight modifications decreases exponentially with $|v^k|$. Specifically, the gain parameter in (4) is taken as

$$\eta_i = \frac{t}{t_0} \exp\left(-|v^k|/t\right) \tag{5}$$

where the *temperature* t linearly decreases from an initial value t_0 to 0 in a given number of cycles $C_{\max}$ through the task. This is achieved by setting at the beginning of the cth cycle $t := \gamma t_0$ with $\gamma = 1 - c/C_{\max}$. The annealing process allows to stabilize the weight vector over any given training period.
As shown in [8, 9], the choice of an initial temperature t_0 and of the specific annealing schedule has a strong impact on the quality of the solutions. In particular, faster decreasing schemes provide comparable results more rapidly.

Any variant of the perceptron algorithm can be easily adapted to the optical flow segmentation problem. To cope with noise in the optical flow estimate, the original system (2) of $2n$ equations is replaced by the following system of $4n$ inequalities:

$$\begin{aligned} w_1 + w_2 x_k + w_3 y_k &\geq d_x(x_k, y_k) - \epsilon \\ -w_1 - w_2 x_k - w_3 y_k &\geq -d_x(x_k, y_k) - \epsilon \\ w_4 + w_5 x_k + w_6 y_k &\geq d_y(x_k, y_k) - \epsilon \\ -w_4 - w_5 x_k - w_6 y_k &\geq -d_y(x_k, y_k) - \epsilon \end{aligned} \tag{6}$$

where $k \in \{1, 2, ..., n\}$ and $\epsilon > 0$ corresponds to the maximum acceptable error.
As mentioned in section 2, the idea of the approach is to extract from (6) close-to-maximum consistent subsystems iteratively. Therefore, starting with

an arbitrary initial solution $\mathbf{w}_0 \in \Re^6$, an image point (x_k, y_k) is chosen randomly, the corresponding set of inequalities (6) is considered and the error of the current solution $\mathbf{w}_i$ is evaluated with respect to each one of the four inequalities. In this case, the error is defined as the difference between the right and the left hand side terms. For each violated inequality, the current solution $\mathbf{w}_i$ is then updated as in the thermal perceptron:

$$\mathbf{w}_{i+1} := \mathbf{w}_i + \frac{t}{t_0} \exp\left(-|E_i^k|/t\right) \mathbf{a}^k \tag{7}$$

where E_i^k denotes the error and $\mathbf{a}^k$ is the coefficient vector of the inequality at hand. The next cycle is started when all image points have been considered in random order.

Eliminating a close-to-maximum consistent subsystem and iterating the procedure until the remaining subsystem is consistent yields a partition of the image into a set of regions corresponding to different motions. Although this simple greedy strategy turns out very effective experimentally, it is not guaranteed to lead in polynomial time to partitions containing a minimum number of consistent subsystems. Indeed, we proved that this minimum cardinality partition problem [18] and the maximum consistent subsystem one [16, 17] are $\mathcal{NP}$-hard as well as hard to approximate. Therefore the best we can do is to devise fast heuristic methods with good average-case behavior. In fact, it is easily verified that even if maximum consistent subsystems were available at each step, our greedy strategy would not be guaranteed to yield minimum partitions [18]. Nevertheless, the idea of breaking down the overall partition problem into that of determining a sequence of maximum consistent subsystems is particularly attractive in a setting, like image processing, where the ultimate goal is to achieve real-time computation. This is especially true since, in spite of the inherent worst-case complexity of finding maximum consistent subsystems, the thermal variants of the perceptron algorithm exhibit a good experimental behavior [8, 9, 19].

Finally, it is worth noting that, unlike in the usual perceptron applications, the weight vectors determined during the training phase are not used to classify new patterns. Instead, they allow to extract the structure and they provide the parameters of the piecewise linear model of object motion.

4. COMPARISONS WITH OTHER APPROACHES

In order to evaluate the potentialities, properties and limitations of the proposed approach it is worth comparing it with some other ones that have been used till now to tackle the same problem. Only some key points are recalled and a complete analysis has to be omitted for lack of space.

Robust regression analysis is an important statistical tool frequently employed in computer vision for motion analysis [11, 10]. These least square based methods achieve optimum results when the data error distribution is Gaussian. However, they become unreliable if the noise has non-zero mean components and if outliers are present in the data. Outliers are usually considered

as large measurements errors or impulse noise corrupting the data. In reality, if we consider the object motion estimation problem the definition of *outliers* is too restrictive since all motion data that do not belong to the *dominant* motion should be considered as outliers. The *breakdown point* of a regression method is the smallest amount of the outlier contamination that may force the value of the estimate outside an arbitrary range. Many robust regression methods have been proposed in the literature [11, 10]. The highest breakdown point of 0.5 is reached by some methods such as the repeated-median and the least median of squares. The complexity of those methods is very high, respectively of the order of $O(n^p \log_p n)$ and $O(n^{(p+1)} \log n)$, where p is the number of parameters of the regression and n is the number of data elements.
In the optical flow segmentation problem, it cannot be guaranteed *a priori* that more that 50% of the data belongs to the same coherent parameter model. Therefore, the data needs to be partitioned before applying robust regression techniques for parameter estimation. Many measures have been proposed for the space partitioning of image features or optical flow data. Statistical confidence measures on the optical flow discontinuities, on the gray level discontinuities, on the position of the foci of expansion [4, 3] have been used to generate hypothesis on object motion structure. The clustering of the data into disjoint or fuzzy sets is then generally based on minimizing *distance* measures among the data and the cluster centroids. However, in most of these methods the number of regions in the partition has to be guessed in advance and all of them have a high degree of complexity. For instance, the k-medoids-like algorithms are characterized by an $O(n^k)$ complexity, where n is the number of data elements and k is predetermined the number of clusters [20].
Among the above-mentioned methods, only the Hough Transform [13] (HT) can, in principle, solve the multiple object motion problem without limitations or heuristic hypothesis on the data structure [12]. But in most applications the HT requires a prohibitive amount of time and space in order to reach a good accuracy.

5. EXPERIMENTAL RESULTS

A number of experiments have been carried out with optical flows generated synthetically and extracted from natural images. Figure 1 reports a synthetic affine optical flow that contains five consistent subsystems of the same dimension. The correct segmentation is shown in figure 2. The parameter values have been generated randomly with some constraints in order to yield realistic values of the flow vectors. Subsystems of the same size have been chosen to test the algorithm in the most critical conditions. Note that no robust regression algorithm can cope with this data unless it is preceeded by a preliminary and, in most cases, problematic clustering stage. Results for subsystems with different dimensions show similar behaviors but faster convergence. Figure 3 shows the same flow structure with the addition of pseudo-random gaussian noise with a variance of about 20% of the average

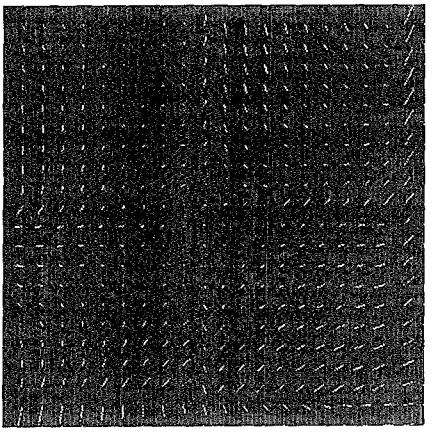


Figure 1: Flow field composed by five synthetically generated affine transformations.

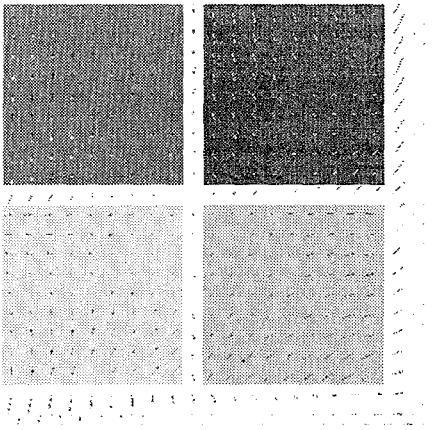


Figure 2: Segmentation of the field into the consistent subsystems. Each gray level corresponds to a consistent subsystem.

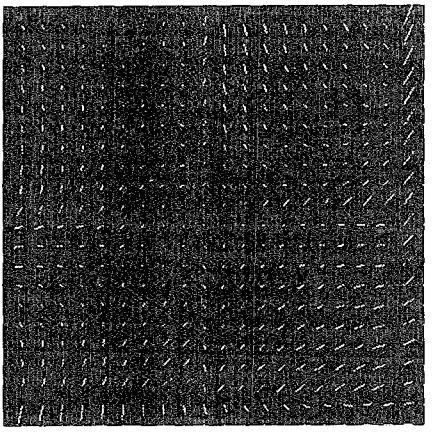


Figure 3: Flow field composed by five synthetically generated affine transformations and superimposed additive gaussian noise.

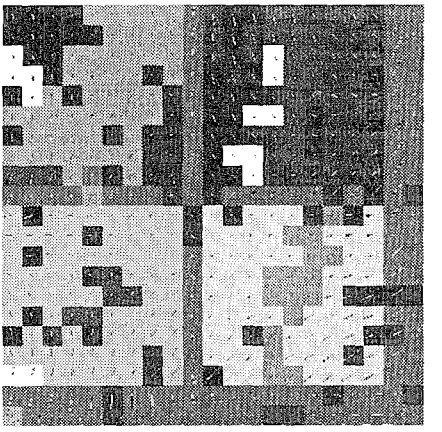


Figure 4: Segmentation of the noisy field into consistent subsystems. Each gray level corresponds to a consistent subsystem.

	w_1	w_2	w_3	w_4	w_5	w_6
Par. subs.1	0.020	0.209	-0.394	-1.828	-0.1255	-0.1165
Estimate	0.0203	0.2094	-0.3938	-1.8277	-0.1254	-0.1166
Noisy est.	0.2543	0.3096	-0.2717	-1.6239	-0.1122	0.0974
Par. subs.2	0.734	-0.473	0.0155	0.29	0.441	0.368
Estimate	0.7342	-0.4729	0.0154	0.2897	0.4409	0.3681
Noisy est.	0.7491	-0.4888	0.0145	0.4718	0.4129	0.3857
Par. subs.3	0.2740	0.283	-0.017	-0.0440	-0.2525	-0.3445
Estimate	0.2739	0.2832	-0.0169	-0.0437	-0.2525	-0.3447
Noisy est.	0.2784	0.2786	-0.0441	0.0015	-0.2576	-0.3363

Table 1: True and estimated parameters for three affine transformations of Figures 2 and 4.

data dynamic. The segmentation result is reported in figure 4. Table 1 provides an example of true and estimated parameters in both noisy and non noisy cases. For optical flows without noise, the thermal perceptron algorithm determines the exact segmentation as well as the true motion parameters in less than 300 cycles through all image points. In the case of noisy flows, the relevant structure of the segmentation is detected but some small consistent subsystems appear due to noisy vectors. However, the parameter estimation of the dominant systems remains quite accurate and the spurious subsystems can be easily identified based on their (small) size or poor spatial coherence. In fact, the corresponding noisy vectors could even be corrected by performing an additional optical flow estimation according to the spatial consistency of the current partition.

The results obtained for natural images and presented elsewhere [18], are similar to those for noisy synthetic flows. Indeed, the algorithm can estimate in a very small number of operations (compared to classical approaches), the number of moving objects, their motion parameters and their spatial structure.

Given its computational requirements and performance, our approach fits particularly well into the layered motion representation scheme described in [7] and allows to circumvent the "a priori" clustering problem without facing the structural limitation of classical estimation techniques.

6. CONCLUSION

A new approach based on the extraction of maximum consistent subsystems of linear systems has been proposed for the analysis and parameterization of motion. Simple variants of the perceptron algorithm can be used to tackle the general problem of determining the linear motion parameters of moving objects from image sequences. Unlike in the usual perceptron applications,

the goal of the training process is not to generalize to new patterns, but to determine as large as possible consistent subsets of known data (image points) and then to interpret the resulting weight vector components as the motion parameters of the corresponding regions. The experimental results obtained for motion fields generated synthetically and extracted from natural images indicate that our approach allows to overcome some intrinsic limitations of other classical alternatives and compares very favorably in terms of complexity and performance.

References

[1] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performances of optical flow techniques. Technical Report Report no. 299, Dept. of Computer Science University of Western Ontario, London, Ontario, Canada, July 1992.

[2] G. Adiv. Determining three-dimensional motion and structure from optical flow generated by multiple moving objects. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-7, No. 4, pp. 384–401, July 1985.

[3] A. Rognone, M. Campani, and A. Verri. Identifying multiple motions from optical flow. In G. Sandini editor, editor, *Proceeding of the second European conference on computer vision*, pages 258–266, S. Margherita Ligure, Italy, May 1992.

[4] M. Pertrou, M. Bober, and J. Kittler. Multiresolution motion segmentation. In IEEE Computer Society Press, editor, *Proceedings of the 12th international conference on pattern recognition*, volume 1, pages 743–746, Jerusalem, Israel, October 9-13 1994.

[5] J. Berger, P. Anadan, K. Hanna, and R. Hingorani. Hierarchical model based motion estimation. In G. Sandini editor, editor, *Proceeding of the second European conference on computer vision*, pages 237–252, S. Margherita Ligure, Italy, May 1992.

[6] J. Wang and E. Adelson. Layered representation for motion analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 361–366, New York, USA, June 1993.

[7] S. Hsu, P. Anadan, and S. Peleg. Accurate computation of optical flow by using layered motion representation. In IEEE Computer Society Press, editor, *Proceedings of the 12th international conference on pattern recognition*, volume 1, pages 743–746, Jerusalem, Israel, October 9-13 1994.

[8] M. Frean. A "thermal" perceptron learning rule. *Neural Computation*, Vol. 4, No. 6, pp. 946–957, 1992.

[9] E. Amaldi and C. G. Diderich. On the probabilistic and thermal perceptron training algorithms. Technical Report ORWP-8-94, Department of Mathematics, Swiss Federal Institute of Technology, Lausanne, 1994.

[10] A. M. Leroy P.J. Rousseeuw. *Robust regression and outlier detection.* John Wiley & Sons, Inc, New York, 1987.

[11] P. Meer, D. Mintz, A. Rosenfeld, and D. Kim. Robust regression methods for computer vision: a review. *International journal of computer vision*, Vol. 6, pp. 59–70, 1991.

[12] R.A. Samy and C.A. Bozzo. Moving object recognition using motion enhanced Hough transform. In V. Cappellini and A.G. Costantinides, editors, *Digital Signal Processing-84*, pages 770–775, Amsterdam, The netherlands, 1984.

[13] J. Illingworth and J. Kittler. The adaptive hough transform. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. PAMI-9, No. 5, pp. 690–698, September 1987.

[14] M. L. Minsky and S. Papert. *Perceptrons: An introduction to computational Geometry.* MIT Press, Cambridge, MA, 1988. Expanded edition.

[15] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, Vol. 4, pp. 373–395, 1984.

[16] E. Amaldi. On the complexity of training perceptrons. In T. Kohonen, K. Mäkisara, O. Simula, and J. Kangas, editors, *Artificial Neural Networks*, pages 55–60, Amsterdam, 1991. Elsevier science publishers B.V.

[17] E. Amaldi and V. Kann. The complexity and approximability of finding maximum feasible subsystems of linear relations. Technical Report ORWP-11-93, Department of Mathematics, Swiss Federal Institute of Technology, 1993. To appear in Theoretical Computer Science.

[18] M. Mattavelli and E. Amaldi. Manuscript in preparation, 1995.

[19] E. Amaldi. *From finding maximum feasible subsystems of linear systems to feedforward neural network design.* PhD thesis, Department of Mathematics, Swiss Federal Institute of Technology, Lausanne, 1994.

[20] P.J. Rousseeuw L. Kaufman. *Finding groups in data.* John Wiley & Sons, Inc, New York, 1990.

A MULTIPLE SCALE NEURAL SYSTEM FOR BOUNDARY AND SURFACE REPRESENTATION OF SAR DATA

Stephen Grossberg[1], Ennio Mingolla[2], and James Williamson[3]

Department of Cognitive and Neural Systems and Center for Adaptive Systems, Boston University, 111 Cummington Street, Boston, MA 02215[4]

1 INTRODUCTION

A neural network model of boundary segmentation and surface representation is developed to process images containing range data gathered by a synthetic aperture radar (SAR) sensor. Synthetic aperture radar sensors can produce range imagery of high spatial resolution under difficult weather conditions (Munsen, O'Brien, and Jenkins, 1983; Munsen and Visentin, 1989) but the image data presents some difficulties for interpretation by human observers or automatic recognition systems. Among these difficulties is the large dynamic range (five orders of magnitude) of the sensor signal, which requires some type of nonlinear compression merely for an image to be represented and viewed on a typical computer monitor. Another problem is image speckle, which is generated by coherent processing of radar signals, and has characteristics of random multiplicative noise.

To date, many approaches for speckle suppression have relied on simple statistical models for the signal and the noise which are insufficient for accurately representing natural scenes. Processing based on these models has thus tended to suppress the signal as well as the speckles (Lee, 1983). Other approaches have used the iterative application, within a small window, of nonlinear filtering techniques which aim to preserve the signal while smoothing speckle noise. Our approach capitalizes instead on the form-sensitive operations of a neural network model in order to detect and enhance structure

[1]Supported in part by the Air Force Office of Scientific Research (AFOSR F49620-92-J-0499), the Advanced Research Projects Agency (ONR N00014-92-J-4015), and the Office of Naval Research (ONR N00014-95-1-0409 and ONR N00014-95-1-0657).

[2]Supported in part by the Air Force Office of Scientific Research (AFOSR F49620-92-J-0334) and the Office of Naval Research (ONR N00014-94-1-0597).

[3]Supported in part by the Air Force Office of Scientific Research (AFOSR F49620-92-J-0225), the Advanced Research Projects Agency (ONR N00014-92-J-4015), the National Science Foundation (NSF IRI-90-24877), and the Office of Naval Research (ONR N00014-91-J-4100).

[4]Acknowledgments: The authors wish to thank Diana Meyers for her valuable assistance in the preparation of the manuscript.

0-7803-2739-X/95 $4.00 © 1995 IEEE

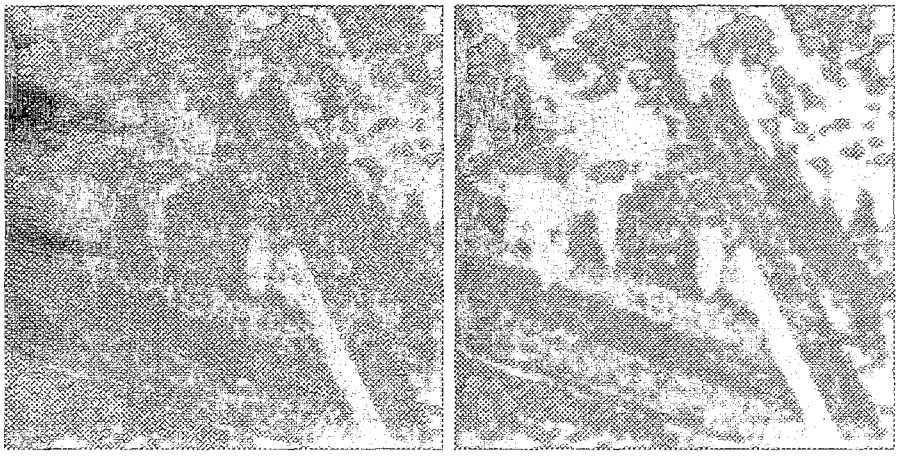


Figure 1: Left: Nonlinearly compressed SAR image of New York State Thruway. Right: Multiple scale BCS/FCS output.

based on information over large, variably sized and variably shaped regions of the image. In particular, the multi-scale implementation of the neural model reported here is capable of exploiting and combining information from several nested neighborhoods of a given image location to determine the final intensity value to be displayed for that pixel. By "neighborhood" is here meant a region whose form varies as a function of nearby image data, not some fixed (weighted) radial function for all pixel locations.

Our approach is illustrated in Figure 1. Figure 1 (left) shows a 400x400 pixel SAR image. The image was obtained using a 35-GHz synthetic aperture radar with 1 ft by 1 ft resolution and a slant range of 7 km (Novak, Burl, Chaney, and Owirka, 1990). The image is from upstate New York, of a highway with bridge overpass. The original 512x512 pixel image was reduced via gray-level consolidation to 400x400 pixels before processing, and was nonlinearly compressed via the function $f(I) = I/(2000 + I)$ for displaying. Figure 1 (right) shows the multiple-scale output of the BCS/FCS system. The BCS/FCS automatically compresses the image intensities and smooths over image speckle while preserving informative structures.

Boundary and surface processing are herein accomplished by an improved Boundary Contour System (BCS) and Feature Contour System (FCS), respectively, that have been derived from analyses of perceptual and neurobiological data. BCS/FCS processing makes structures such as motor vehicles, roads, and buildings more salient and interpretable to human observers than they are in the original imagery. The neural network model used here is summarized in Figure 2. It is a refinement of the Boundary Contour System (BCS) for boundary segmentation that was developed by Grossberg and Mingolla (1985a, 1985b, 1987) and the Feature Contour System (FCS) for surface representation that was developed by Cohen and Grossberg (1984) and Grossberg and Todorović (1988) through an analysis of biological vision. Several of these improvements were introduced in Cruthirds *et al.* (1992). Taken to-

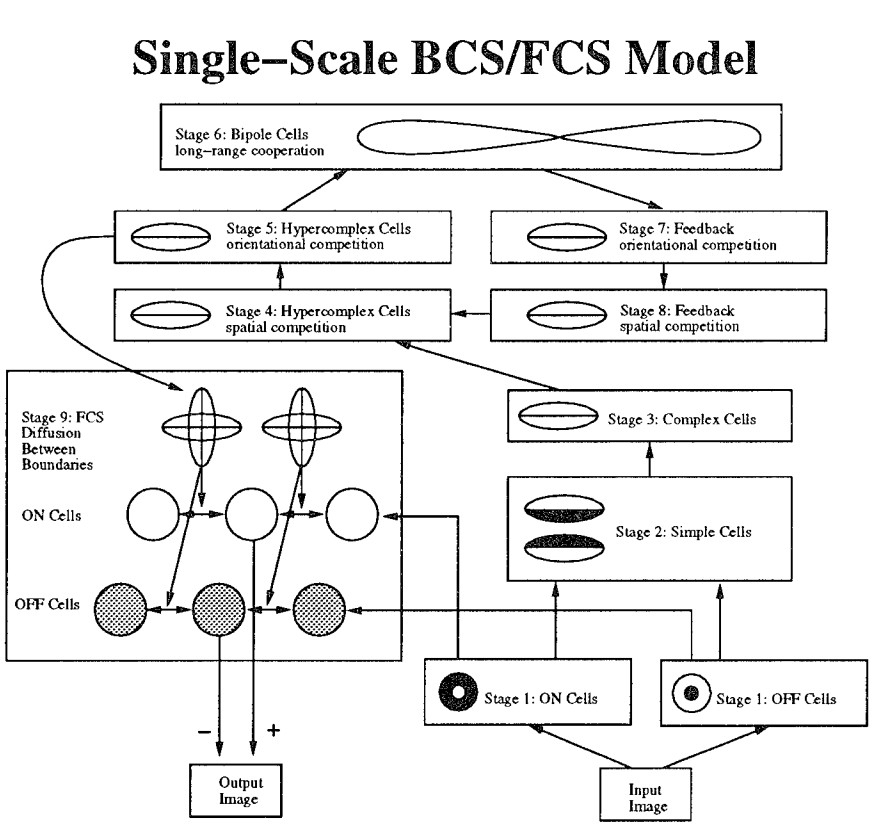


Figure 2: A single scale of processing in the BCS/FCS model.

gether, the BCS and FCS form part of the FACADE theory of biological and machine vision (Grossberg, 1994), so called because the acronym FACADE stands for the representations of Form-And-Color-And-DEpth that are suggested to occur at the final FCS surface representations of the full binocular theory. In the present work, only monocular, or single detector, processing is described, so the model is considerably simpler than its binocular version.

Early processing by ON cells and OFF cells embedded in shunting center-surround network models preprocessing by lateral geniculate nucleus (LGN). Such preprocessing compensates for illumination gradients, normalizes input dynamic range, and extracts local ratio contrasts. ON cell and OFF cell outputs are combined in the BCS to detect, regularize, and complete coherent boundary representations, while suppressing image noise, using multiple-scale filtering and cooperative-competitive feedback interactions. Multiple copies of the BCS are defined, each corresponding to a different receptive field size. Each BCS copy inputs to a corresponding copy of the FCS at which filling-in of a surface representation occurs.

Boundary segmentation is performed by three copies of the BCS at small, medium, and large filter scales, whose subsequent interaction distances covary with the size of the filter. Filling-in of multiple surface representations

occurs within the FCS at each scale via a boundary-gated diffusion process. Diffusion is activated by the normalized LGN ON and OFF outputs within ON and OFF filling-in domains. Diffusion is restricted to the regions defined by gating signals from the corresponding BCS boundary segmentation. The filled-in opponent ON and OFF signals are subtracted to form double opponent surface representations.

These surface representations are sensitive to both image ratio contrasts and background luminance. The three scales of surface representation are then added to yield a final multiple-scale output. The BCS and FCS are shown to perform favorably in comparison to several other techniques for speckle removal.

2 DESCRIPTION OF BCS AND FCS

The first model stage, schematized as unoriented annuli in Figure 2 represents the shunting on-center off-surround, and off-center on-surround, interactions at the retinal and LGN levels. These ON and OFF cells compensate for variable illumination and compute the ratio contrasts in the image. The model LGN cells generate half-wave rectified outputs. The difference of the ON and OFF outputs are shown in Figure 3 (row 1) at three spatial scales, for a section of the Figure 1 image.

The ON and OFF cell outputs are received by pairs of like-oriented simple cells (Stage 2 in Figure 3) that are sensitive to opposite contrast polarity, or direction-of-contrast. The simple cell pairs, in turn, pool their rectified and oppositely polarized output signals at like-oriented complex cells (Stage 3). Complex cells are hereby rendered insensitive to direction-of-contrast, as are all subsequent cell types in the model. Complex cell activations, summed across orientation, are shown in Figure 3 (row 2) at three spatial scales.

Complex cells activate hypercomplex cells through an on-center off-surround network (Stage 4) whose off-surround carries out an endstopping operation. In this way, complex cells excite hypercomplex cells of the same orientation and position, while inhibiting hypercomplex cells of the same orientation at nearby positions. One role of this spatial competition is to spatially sharpen the neural responses to oriented luminance edges. Another role is to initiate the process at Stage 5 called *end cutting*, whereby boundaries are formed that abut a line end at orientations perpendicular or oblique to the orientation of the line itself (Grossberg, 1987; Grossberg and Mingolla, 1985b).

Output from the higher-order hypercomplex cells feed into cooperative bipole cells at Stage 6. The bipole cells initiate long-range boundary grouping and completion. Bipole cells realize a type of statistical AND gate, since they fire if both of their receptive fields are sufficiently activated by appropriately oriented hypercomplex cell inputs. Bipole cells hereby realize a type of long-range cooperation among the outputs of active hypercomplex cells. For example, a horizontal bipole cell, as in Figure 2, is excited by activation of horizontal hypercomplex cells that input to its horizontally oriented recep-

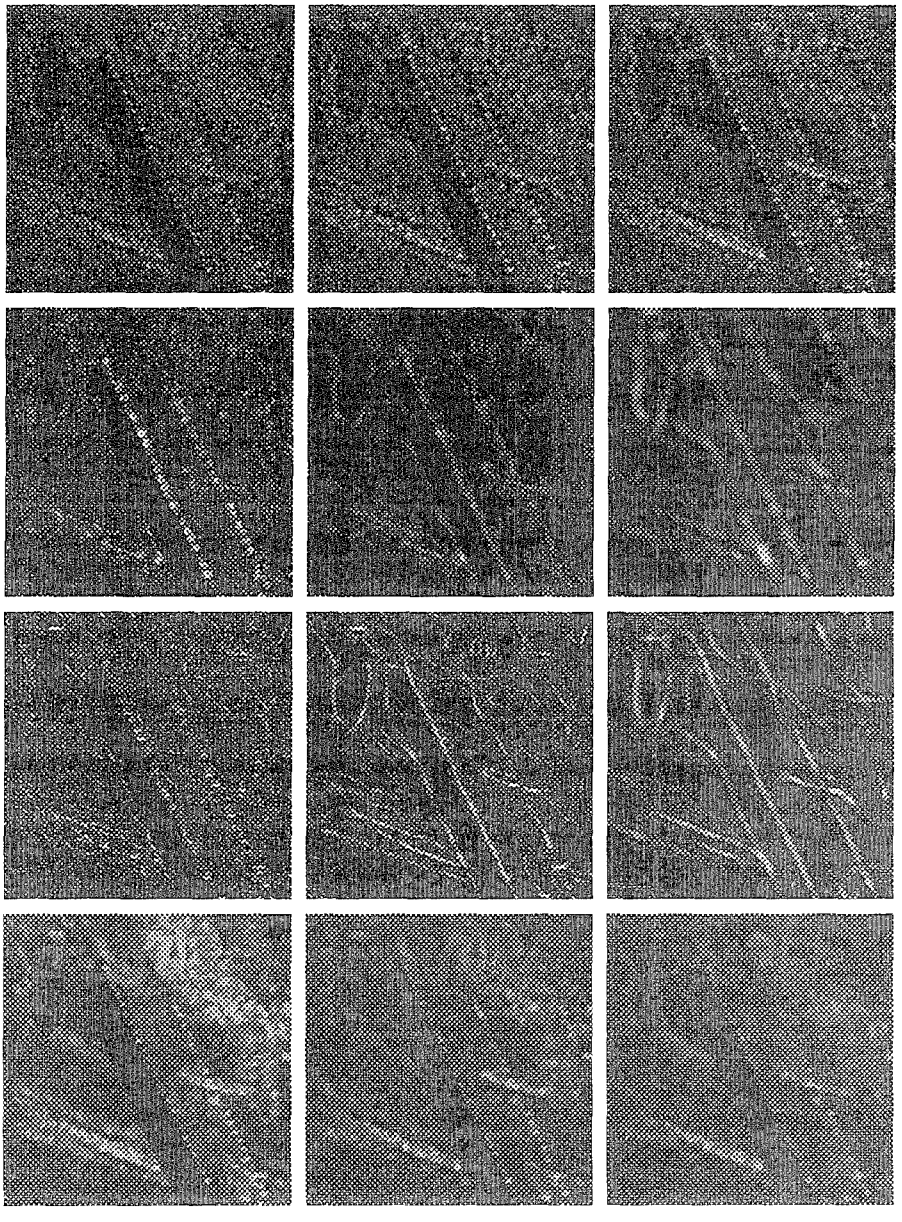


Figure 3: Row 1: Stage 1 difference of ON and OFF cell processing at three scales. Row 2: Stage 3 complex cell processing at three scales. Intensity of each pixel depicts the total activity of the oriented complex cells at that position. Row 3: Stage 5 hypercomplex cell processing at three spatial scales. Intensity of each pixel depicts the total activity of the cells at that position. Row 4: Stage 9 processing result at three different scales on example image. A linear combination of these images is used to obtain the final multiple-scale output in Figure 1 (right).

tive fields. A horizontal bipole cell is also inhibited by activation of vertical hypercomplex cells. This inhibition prevents boundaries from being colinearly completed across regions that contain sufficiently many perpendicular or oblique contrasts, a property called spatial impenetrability (Grossberg, 1987; Grossberg and Mingolla, 1985b).

Bipole cells were predicted to exist in Cohen and Grossberg (1984) and Grossberg (1984) shortly before cortical cells in area V2 with similar properties were reported by von der Heydt, Peterhans, and Baumgartner (1984). At around the time of the von der Heydt *et al.* report, Grossberg and Mingolla (1985a, 1985b) used bipole cell properties to simulate and explain data about illusory contour formation, neon color spreading, and texture segregation, among other topics. These same properties play a role in our simulations of SAR data.

Bipole cells generate feedback signals to like-oriented hypercomplex cells within Stages 7 and 8 of Figure 2. These feedback signals help to create and enhance spatially and orientationally consistent boundary groupings, while inhibiting inconsistent ones. In particular, bipole cell outputs compete across orientation at each position within Stage 7 to select the cooperatively most favored orientation, or orientations. These outputs then undergo spatial competition that excites cells at the same orientation and position while inhibiting cells at nearby positions. Cells which derive the most cooperative support from bipole grouping after these competitive selection processes thereupon further excite the corresponding bipole cells. This cycle of bottom-up and top-down interaction between hypercomplex cells and bipole cells rapidly converges to a final boundary segmentation. Feedback among bipole cells and hypercomplex cells hereby drives a resonant cooperative-competitive decision process that completes the statistically most favored boundaries, suppresses less favored boundaries, and coherently binds together appropriate feature combinations in the image. Stage 5 hypercomplex cell activations, following several iterations of cooperative feedback, are shown at three scales in Figure 3 (row 3). The hypercomplex cell boundaries in row 3 are sharper and more complete than the corresponding complex cell boundaries in row 2.

Cohen and Grossberg (1984) and Grossberg and Todorović (1988) developed the FCS model to simulate many data about brightness perception. Arrington (1994) has shown that the Grossberg-Todorović (1988) model also accurately simulates the dynamics of brightness filling-in as reported in the psychophysical experiments of Paradiso and Nakayama (1991). The BCS produces boundary signals that act as barriers to diffusion within the FCS in response to ON and OFF inputs from which the illuminant has been discounted. As diagrammed in Figure 2, these boundary signals act to gate diffusion of signals from the ON and OFF cells at Stage 9. That is, for image pixels through which no boundary signals pass, resulting intensity values become more homogeneous as the diffusion evolves. Where boundary signals intervene, however, they block the diffusion, leaving a resulting difference of intensity level on either side of the boundary signal. The result of

such boundary-gated diffusion is a form-sensitive computation that adapts to each unique combination of image inputs, rather than a correlation derived through a fixed kernel.

The ON and OFF signals may or may not be combined to generate the final FCS surface representation. There are several related ways to do this that all lead to essentially equivalent results. See Grossberg, Mingolla, and Williamson (1995) for a more complete description. The basic idea in all cases is to combine FCS surface measures that depend upon both the ratio contrasts and the averaged background luminances of the image. To achieve this, the OFF responses are subtracted from the ON responses, either before or after the filling-in stage. This strategy was introduced in Grossberg (1987b) and has been applied in several studies since; e.g., Grossberg (1994), Grossberg and Wyse (1991), Neumann (1993), and Pessoa, Mingolla, and Neumann (1995). Such a subtraction of OFF (off-center on-surround) signals from ON (on-center off-surround) signal cells is said to create *double* opponent cells, since it combines two successive competitive (or opponent) interactions.

In perhaps the simplest double opponent computation, that of Grossberg and Wyse (1991), the OFF cells have a higher tonic, or baseline, activity than do the ON cells. When the OFF cell responses are subtracted from the ON cell responses, there are two terms: one is sensitive to ratio contrast and the other, which arises from the asymmetric baseline activities, increases as a function of a low-pass nonlinearly-compressed luminance estimate. The net double opponent signal diffuses across an FCS filling-in domain, or FIDO. Alternatively, the ON and OFF inputs could first diffuse within their own ON and OFF FIDOs, each gated by the same boundary segmentation, before the net ON-minus-OFF double-opponent response is computed. The net filled-in signal is shown at three scales in Figure 3 (row 4).

3 COMPARISON TO PRIOR METHODS

The BCS/FCS results are now compared to previously published methods for speckle noise reduction. The alternative methods considered are: smoothing with a median filter (Scollar, Weidner, and Huang, 1984), adaptive averaging with a sigma filter (Lee, 1983), and smoothing with a geometric filter (Crimmins, 1985). The parameters of these methods are set to obtain a similar net amount of smoothing—as determined by informal observation—as the BCS/FCS, in order to evaluate how well they remove noise while retaining actual image features with respect to the BCS/FCS.

Because it tends to ignore outliers, the median filter is a sensible method for reducing speckle noise (Scollar *et al.*, 1984). A 3x3 median filter was applied for 3 iterations. Alternatively, averaging with a mean filter blurs real edges too much. This problem is addressed with the sigma filter, which only averages those pixels with intensity within two standard deviations of the center pixel. However, this approach leaves many outliers, which are due to speckle noise, untouched. This problem is addressed by locally averaging

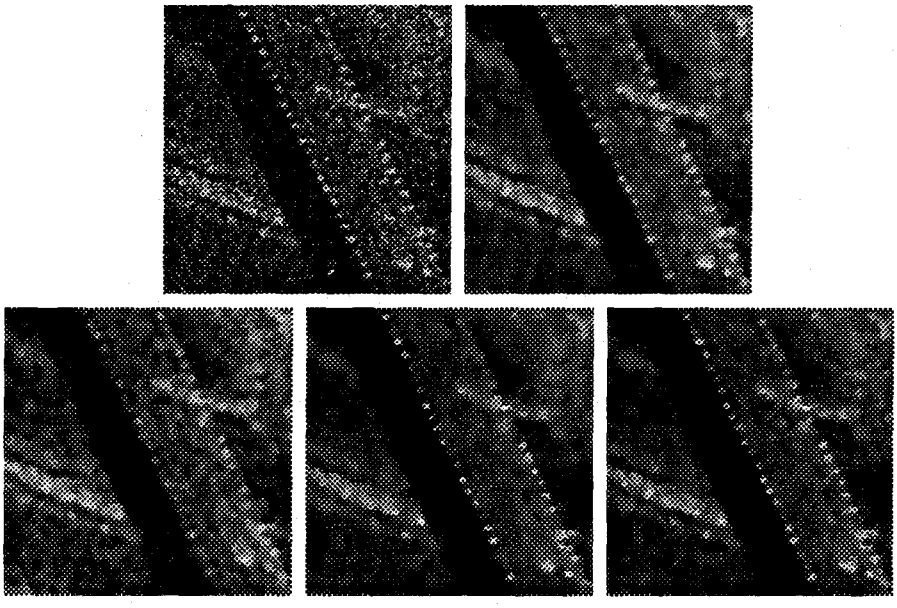


Figure 4: Comparison of speckle noise reduction methods. (a) Top Left: SAR image of scene with overpass over New York State Thruway, taken from image in Figure 1. (b) Top Right: Multi-scale BCS/FCS. (c) Bottom Left: 3 iterations of 3x3 median filter. (d) Bottom Middle: Adaptive averaging using 2 iterations of 5x5 sigma filter. (e) Bottom Right: 3 iterations of geometric filter.

those pixels for which K or fewer other pixels in the averaging window lie within two standard deviations (Lee, 1983). Adaptive averaging was done for 2 iterations, using a 5x5 sigma filter, with the standard deviation estimated at a relatively flat image region, and with a threshold of $K = 3$ for removing spot noise. Another method for speckle noise reduction, the geometric filter, was also used. The geometric filter iteratively enforces a minimum constraint for curvature, in pixel intensity space, between neighboring pixels (Crimmins, 1985). Each iteration of the geometric filter involves two successive applications of four nearest-neighbor intensity curvature rules, in four directions 45 degrees apart, horizontally, diagonally, vertically, and diagonally. The first application reduces the curvature from above, filling in holes or narrow valleys. The second application reduces curvature from below, reducing spikes or narrow ridges.

Figure 4 (top left) shows a section of the image from Figure 1, following nonlinear compression of signal values, used as input to the noise reduction methods. This image contains an overpass of the New York State Thruway. Note that the detail of the overpass guardrails is maintained by the BCS/FCS (top right), while speckled but homogeneous regions are smoothed over. The median filter method (bottom left) and adaptive averaging method (bottom middle) do not do as well at maintaining important detail while smoothing away noise. The geometric filter after 3 iterations (bottom right), also does a

good job at smoothing noise while maintaining detail. An important consideration in evaluating the alternative approaches is that the BCS/FCS reliably produces a result like that shown in Figure 4 as a final, equilibrium solution. The geometric filter, however, iteratively smooths the image. Therefore, with a geometric filter the user must choose how many iterations to use to achieve the desired level of smoothness for a given set of imagery.

4 REFERENCES

Arrington, K.F. (1994). The temporal dynamics of brightness filling-in. *Vision Research*, **34**, 3371–3387.

Cohen, M. and Grossberg, S. (1984). Neural dynamics of brightness perception: Features, boundaries, diffusion, and resonance. *Perception and Psychophysics*, **36**, 428–456.

Crimmins, T. (1985). Geometric filter for speckle reduction. *Applied Optics*, **24**, 1438–1443.

Cruthirds, D., Gove, A., Grossberg, S., Mingolla, E., Nowak, N., and Williamson, J. (1992). Processing of synthetic aperture radar images by the Boundary Contour System and Feature Contour System. **Proceedings of the International Joint Conference on Neural Networks (IJCNN-92)**, IV-414–419.

Ferster, D. (1988). Spatially opponent excitation and inhibition in simple cells of the cat visual cortex. *Journal of Neuroscience*, **8**, 1172–1180.

Grossberg, S. (1984). Outline of a theory of brightness, color, and form perception. In E. Degreef and J. van Buggenhaut (Eds.), **Trends in Mathematical Psychology**. Amsterdam: North Holland.

Grossberg, S. (1987). Cortical dynamics of three-dimensional form, color, and brightness perception, I: Monocular theory. *Perception and Psychophysics*, **41**, 87–116.

Grossberg, S. (1994). 3-D vision and figure-ground separation by visual cortex. *Perception and Psychophysics*, **55**, 48–120.

Grossberg, S. and Mingolla, E. (1985a). Neural dynamics of form perception: Boundary completion, illusory figures, and neon color spreading. *Psychological Review*, **92**, 173–211.

Grossberg, S. and Mingolla, E. (1985b). Neural dynamics of perceptual grouping: Textures, boundaries, and emergent segmentations. *Perception and Psychophysics*, **38**, 141–171.

Grossberg, S. and Mingolla, E. (1987). Neural dynamics of surface perception: Boundary webs, illuminants, and shape-from-shading. *Computer Vision, Graphics, and Image Processing*, **37**, 116–165.

Grossberg, S., Mingolla, E., and Todorović, D. (1989). A neural network architecture for preattentive vision. *IEEE Transactions on Biomedical Engineering*, **36**, 65–84.

Grossberg, S., Mingolla, E., and Williamson, J.R. (1995). Synthetic aperture radar processing by a multiple scale neural system for boundary and surface representation. *Neural Networks*, in press.

Grossberg, S. and Todorović, D. (1988). Neural dynamics of 1-D and 2-D brightness perception: A unified model of classical and recent phenomena. *Perception and Psychophysics*, **43**, 241–277.

Grossberg, S. and Wyse, L. (1991). A neural network architecture for figure-ground separation of connected scenic figures. *Neural Networks*, 4, 723–742.

Lee, J. (1983). A simple speckle smoothing algorithm for synthetic aperture radar images. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC–13, 85–89.

Munsen, D. Jr., O'Brien, J., and Jenkins, W. (1983). A tomagraphic formulation of spotlight-mode synthetic aperture radar. *Proceedings of the IEEE*, 71(8), 917–925.

Munsen, D. Jr., and Visentin, R. L. (1989). A signal processing view of strip-mapping synthetic aperture radar. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12), 2131–2147.

Neumann, H. (1993). Toward a computational architecture for unified visual contrast and brightness perception: I. Theory and model. In **Proceedings of the World Conference on Neural Networks (WCNN-93), I**, 87–91. Hillsdale, NJ: Erlbaum.

Novak, L., Burl, M., Chaney, R., and Owirka, G. (1990). Optimal processing of polarimetric synthetic-aperture radar imagery. *The Lincoln Laboratory Journal*, 3(2), 273–290.

Paradiso, M.A. and Nakayama, K. (1991). Brightness perception and filling-in. *Vision Research*, **31**, 1221–1236.

Pessoa, L., Mingolla, E., and Neumann, H. (1995). A contrast- and luminance-driven multiscale network model of brightness perception. *Vision Research*, in press.

Scollar, L., Weidner, B., and Huang, T. (1984). Image enhancement using the median and the interquartile distance. *Computer Vision, Graphics, and Image Processing*, **25**, 236–251.

von der Heydt, R., Peterhans, E., and Baumgartner, G. (1984). Illusory contours and cortical neuron responses. *Science*, **224**, 1260–1262.

A Neural Network Approach to Face/Palm Recognition*

S. Y. Kung Shang-Hung Lin
Princeton University

Ming Fang
Siemens, SCR

Abstract

This paper proposes a face/palm recognition system based on decision-based neural networks (DBNN)[5]. The face recognition system consists of three modules. First, The face detector finds the location of a human face in an image. The eye localizer determines the positions of both eyes in order to generate meaningful feature vectors. The facial region proposed contains eyebrows, eyes, and nose, but excluding mouth. (Eye-glasses will be permissible.) Lastly, the third module is a face recognizer. The DBNN can be effectively applied to all the three modules. It adopts a hierarchical network structures with nonlinear basis functions and a competitive credit-assignment scheme. The paper demonstrates its successful application to face recognition applications on both the public (FERET) and in-house (SCR) databases. In terms of speed, given the extracted features, the training phase for 100-200 persons would take less than one hour on Sparc10. The whole recognition process (including eye localization, feature extraction, and classification using DBNN) may consume only a fraction of a second on Sparc10. As to be elaborated in Section 4, experiments on three different databases all demonstrated high recognition accuracies. Nevertheless, particularly for improving false acceptance/rejection, a new variant of DBNN is proposed in a accompanied paper [6]. Finally, our preliminary study also confirms that a similar DBNN recognizer can effectively recognize palms, which could potentially offer a much more reliable biometric feature.

1 Introduction

With its emerging applications in security, financial transactions and many others, biometric recognition systems (e.g. face, palm, finger print) have recently taken on a new importance. There are many challenges in order to

*This research was supported in part by the Electronic Eye Program. This work is based upon a closely related works of an earlier SCR face recognition system collaborated with Drs. S.P. Liou and J.S. Taur[13]. The authors also wish to acknowledge the invaluable contributions by Drs. L.J. Lin and I. Chakravarty of SCR.

0-7803-2739-X/95 $4.00 © 1995 IEEE

meet the application requirements. Most features, either artificial or biometric have major flaws. Some (e.g. signatures or retinal patterns) are hard to scan and match/recognize in real time, some (e.g. PIN numbers or passwords) may be stolen or forgot, and almost all artificially ID features can be potentially forged. With technological advance on microelectronic and visualization, high performance automatic techniques on biometric recognition are now becoming economically feasible.

The reader is referred to [11] and references therein for a more complete survey of previous research works on face recognition. Very briefly, in the pioneering work by Kelly[7], various kinds of facial features were examined. They include width of head, distance between eyes, top of head to eyes, between eye and nose and the distance from eye to mouth. Several promising statistical approaches have been proposed: for examples, KL Transform[12], SVD[14], and eigenfaces[1] techniques. The eigenface approach has a large database, with 7562 images for about 3000 persons. It used 12 eigenvectors processed from 128 random selected images. In a test set of 200 images, 95% recognition rate was reached. However, the performance severely degraded when large variations in lighting (96 %), orientation(85 %), and size (64 %) were present.

Neural networks have shown convincing applicability for face recognition and related applications, e.g. gender[3],[10], facial expression classification[2]. How to construct a neural network model is crucial for successful recognition. For examples, the Dynamic Link Architecture[9] uses Gabor wavelets as features. Another is the Cresceptron[4], which is a multi-resolution pyramid structure, similar to the classic Fukushima's Neocognitron. The present paper proposes a face/palm recognition system based on decision-based neural networks (DBNN)(see also [13]). A modified probabilistic DBNN (with improved performance) is reported in another publication [6].

2 DBNN-Based Face Recognition System

System Configuration The main issues include face isolation, eye location, recognition accuracy (including false acceptance/rejection) and finally system expandability. The overall configuration of the proposed face recognition system is depicted in Figure 1. The image acquisition hardware consists of a CCD camera and a digitizer board. Following CCD imaging, there are three digital processing modules: face detector, eye localizer, and face recognizer. (The DBNN are used for each of these three modules.) Given a greyscale or color image, the DBNN-based face detector determines whether there exists a human face in the image. If there is one (i.e., the confidence score, or the discriminant function value of the winner, is high), the DBNN-based eye localizer is activated to locate the exact locations of both eyes. A subimage corresponding to the face region will then be extracted from the image. After proper normalization of the facial size and illumination, and

adjustment of the facial orientation, a vector of facial features can then be obtained. Finally, the feature vector is fed into a face recognizer to either recognize the person or reject him as a stranger.

Face Detection Both static or temporal techniques have been considered for face detection. For still-image systems, static (e.g. template matching) technique appears suitable. On the other hand, temporal based processing would be powerful in video-based face recognition systems. The results reported in the sequel are based on the static approach. For training the *face detector*, the face pattern is normalized (affine transformed) so that the distance between both eyes remains constant and both eyes are on a horizontal line. Each face/non-face pattern is further processed into a sobel edge map of size 16 by 16 pixels, and this sobel edge map is the inputs to the DBNN-based face detector. To detect a face in an input image, each of the possible subimages is processed to see if it represents as a face. The face location basically corresponds to the location of the subimage which is considered a face pattern.

Defining Facial Region One crucial step of facial feature extraction is to isolate a most reliable facial region from a given 256x256 image. Our study shows that eyes, nose, and mouth each individually provided 20-30 % confidence; nose/mouth or eyes/mouth combinations raise confidence to 60+ %, while the eyes/nose/mouth reaches 90+ % confidence. Still a facial region consisting of eyes and nose (excluding mouth) can yield even highest confidence. So our facial region - containing eyes, eye brows, and nose - can provide distinctive facial features and yet offer relative stability against different facial expressions, hair styles, and mouth movement.

Eye Localization After the face detection phase, an eye localizer is applied to a (grey-scale or color) face image to locate the left and right eyes. This is a key feature of our approach, since the eye positions provide a very effective means to normalize the face size and reorient the face image. The pattern resolution used for eyes is much higher than that used for faces. Both conventional and DBNN based techniques have been developed:

- The eye location algorithm described in [8] may be adopted to extract the location of left and right eyes from a grey-scale facial image. By exploiting dense horizontal structure (specially pertaining to eyes) the more exact locations of eyes can be determined. Since it does not incur the usage of eye template, so even when eyes are closed (relative position of two eyes lead to eye-nose region Some special features of our eye localizer are as follows. It can detect eyes even when they are closed. It is insensitive to small change of the head size and face orientation as well as the shapes of the eyes. The resultant facial region (containing eyebrows, eyes, and nose) is depicted in Figure 1.

- For training a DBNN *eye localizer*, we need only to train a left eye detector and then create the mirror image of the input image for right eyes. For more robust training, a warping technique may be used to create virtual training exemplars.

Intensity Normalization and Resolution Reduction Based on the given location of left and right eyes, a facial region can be extracted. Intensity in the facial region are normalized (to a range between 0 and 1) to compensate for changing illumination. Directional edge filters along predetermined directions are applied to the normalized (140 x 100) images and reduce them to 8 by 8 feature vectors. Some advantages for adopting low resolution facial features are:

1. Lower resolution reduces the computational cost for face recognition.
2. Low resolution means also feature vector dimension and thus has small storage space.
3. Low resolution means also high tolerance on the error of eye location.

Face Recognizer The feature vectors for reduced image resolution (8 by 8) are used as input vectors for the DBNN face recognizer. Since a eye localizer cannot be generally imperfect, the face recognizer is trained to tolerate eye location errors. In order to assure sufficient diversity of real facial images in the training set, the algorithm takes the acquired sensor image and transforms it to create additional training exemplars, i.e. *virtual training patterns*. The technique we use in this approach is called image warping, a commonly used technique in the field of computer graphics. In order for warping to work, the algorithm identifies the left and right eyes as well as the mouth and draws a line through the center of the mouth perpendicular to the line connecting both eyes. The warped image will then be normalized (intensity normalization) and reduced to 8x8 (resolution reduction). Due to reduced resolution, the accuracy of the image warping is not critical.

3 DBNN-Based Face Recognizer

The proposed technique uses a hierarchical decision-based neural network (DBNN), using a distributed and localized updating rule based on reinforced and anti-reinforced learning strategy. The gradient of the discriminant function with respect to weight parameters is used as updating direction. The main merit is that it allows the border between any two classes to be settled mutually, with minimum side-effects on other borders. In DBNN, the teacher only tells the correctness of the classification for each training pattern. The teacher is a set of symbols, $\mathcal{T} = \{t_i\}$, labeling the correct class for each input pattern. Unlike the approximation formulation, the exact values of teachers

are not required. *The objective of the training is to find a set of weights which yields a correct classification.*

Decision-based Learning Rules In order to to accommodate the complex decision boundary, a hierarchical nonlinear structure is adopted. Based on reinforced and anti-reinforced learning strategy a distributed and localized updating rule is proposed. The gradient of the discriminant function with respect to weight parameters is used as updating direction. For complex pattern distribution, the discriminant function is usually a priori unknown. This leads to a *credit assignment rule* on when, what, and how to perform network updating. Its main purpose is to alleviate the problem of overtraining the networks.

There are three main aspects of the training rule:

(1) *When to update?* A selective training scheme can be adopted, e.g. weight updating only when misclassification.

(2) *What to update?* The learning rule is distributive and localized. It applies *reinforced learning* to the subnet corresponding to the correct class and *antireinforced learning* to the (unduly) winning subnet.

(3) *How to update?* Adjust the boundary by updating the weight vector $\mathbf{w}$ either in the direction of the gradient of the discriminant function (i.e., reinforced learning) or opposite to that direction (i.e., antireinforced learning).

The following describes Decision-Based Learning Rule: Suppose that $S = \{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(M)}\}$ is a set of given training patterns, each corresponding to one of the L classes $\{\Omega_i, i = 1, \cdots, L\}$. Each class is modeled by a subnet with discriminant functions, say, $\phi(\mathbf{x}, \mathbf{w}_i) \quad i = 1, \ldots, L$. Suppose that the m-th training pattern $\mathbf{x}^{(m)}$ is known to belong to class Ω_i; and

$$\phi(\mathbf{x}^{(m)}, \mathbf{w}_j^{(m)}) > \phi(\mathbf{x}^{(m)}, \mathbf{w}_l^{(m)}), \qquad \forall l \neq j \tag{1}$$

That is, the winning class for the pattern is the j_{th} class (subnet). When and only when $j \neq i$, (i.e., when $\mathbf{x}^{(m)}$ is misclassified), the following update will be performed:

$$\begin{array}{ll} \text{Reinforced Learning:} & \mathbf{w}_i^{(m+1)} = \mathbf{w}_i^{(m)} + \eta \nabla \phi(\mathbf{x}, \mathbf{w}_i) \\ \text{Antireinforced Learning:} & \mathbf{w}_j^{(m+1)} = \mathbf{w}_j^{(m)} - \eta \nabla \phi(\mathbf{x}, \mathbf{w}_j) \end{array} \tag{2}$$

Learning Rules for Elliptic Basis Function (EBF) Neurons The basic RBF version of the DBNN discussed before is based on the assumption that the feature space is uniformly weighted in all the directions. In practice, however, different features may have varying degrees of importance depending on the way they are measured. This leads to the adoption of a (more versatile) elliptic basis function(EBF). The EBF decision-based learning is very effective for many practical applications when features may use different weighting. For examples, if features pertaining to eye-glasses is assigned minimum weighting, the recognition rate would remain robust even if a person wear different eye-glasses in the testing phase.

In practice and for most applications, the EBF discriminant function is confined to the following (upright) version: The discriminant function (for each subnet l) can be generalized to an (upright) elliptic function:

$$\phi(\mathbf{x}, \mathbf{w}_l) = -\frac{1}{2}\sum_{k=1}^{N} \alpha_{lk}(x_k - w_{lk})^2 + \theta_l \tag{3}$$

where N is the dimension of the input patterns, and $\mathbf{w}_l$ is the vector comprising all the weight parameters $\{\alpha_{lk} \geq 0, w_{lk}, \theta_l\}$. The learning formula leads to

$$\begin{array}{ll} \text{Reinforced Learning:} & w_{ik}^{(m+1)} = w_{ik}^{(m)} + \eta\alpha_{ik}(x_k - w_{ik}^{(m)}) \\ & \alpha_{ik}^{(m+1)} = \alpha_{ik}^{(m)} + \eta(x_k - w_{ik}^{(m)})^2 \\ \text{Antireinforced Learning:} & w_{jk}^{(m+1)} = w_{jk}^{(m)} + \eta\alpha_{jk}(x_k - w_{jk}^{(m)}) \\ & \alpha_{jk}^{(m+1)} = \alpha_{jk}^{(m)} + \eta(x_k - w_{jk}^{(m)})^2 \end{array} \tag{4}$$

Learning Rule for Hierarchical Networks The network hierarchy can be divided into several levels:

- **Person-level: supervised mutual training**

  Decision-based learning rule is used.

- **Angle-level: hybrid supervised and unsupervised training**

  If the angle information is available in the feature extraction phase supervised training may be used. Otherwise, unsupervised training is an optional schema.

- **Instantiation level: initialization by unsupervised clustering**

  The purpose is to enhance the robustness of the network. Several approaches can be used to estimate the number of subclusters. When too many subclusters are adopted, the improved approximation is often a result of over-training (i.e. overfitting). This in turn will hamper the model's generalization capability. The number of subclusters can be either predetermined based on some prior knowledge on the face feature distribution. In general, it is determined based on the unsupervised clustering technique subnet growing strategy adopted.

Hybrid Locally Unsupervised and Globally Supervised Learning The training scheme of DBNN is based on the so-called **LUGS** (Locally Unsupervised Globally Supervised) learning[5]. There are two phases in this scheme: during the locally-unsupervised (LU) phase, each subnet is trained individually, and no mutual information across the classes may be utilized. After the LU phase is completed, the training enters the Globally-Supervised

(GS) phase. In GS phase teacher information is introduced to reinforce or anti-reinforce the decision boundaries obtained during LU phase. The discriminant functions in all clusters will be trained by the two-phase learning.

4 Performances on Face Databases

Experiments have been conducted both on public (FERET) and in-house (SCR) databases. The performances are summarized below.

4.1 Frontal View Experiment - FERET Database

A front-view experiment was conducted on the ARPA/ARL FERET database. There are 304 persons, and each of them has two frontal view images. The variation between the two images is much larger than SCR 80x20 and 40x150 database, in terms of illumination, size, and facial expression. We selected the the images which passed both the face detector and eye localizer. Based on this criterion, 200 persons have passed and are used for our face recognizer experiment. One image per person is used for training and the other for testing. The current DBNN reaches 100% in training accuracy and 96% in testing accuracy. An improved probabilistic variant of DBNN can achieve an even higher recognition rate (99%), cf. [6].

4.2 Frontal View Experiment - SCR 80x20 Database

The SCR 80x20 database consists of 80 people of different races, ages, and genders. The database contains 20 images for each person. (If a person wears glasses, 10 of the image are with glasses and 10 without.) All of the images were taken under natural indoor illumination condition. All of the images has clean background. The facial orientations in our image database are roughly between -15 and 15 degrees. In many of those images, the person's head is tilted up to 15 degrees. The face recognizer was trained by half of the images in the database.

We have created a training data set by using 4 images per person (2 with eye glasses and 2 without, if the person wears eye glasses). The testing image set includes 16 images per person, 1280 images in total. For all of the images, the face detector always correctly detected the center of the face (i.e., 100% success rate). Eye localization is a more difficult task than face detection, in particular when eye glasses are present. The eye is typically 20 pixels wide and 10 pixels high. Among the 1280 images, eye localizer mis-detected the eyes in 5 images by errors of more than 5 pixels. For the remaining 1275 images, the DBNN face recognizer can achieve 100% recognition rate.

4.3 Experiment on Large Orientation Variations - SCR 40x150 Database

In this experiment, we have used another image database of 40 people. The database contains 150 images for each person, which were taken continuously while the person was slowly moving/rotating his head. Heads rotate not only in wide angle (up to 45 degree) but also along various axes (i.e., left-right, up-down, and tilted rotations). The face detector and eye localizer worked correctly for 75% of the 6000 images in this database. (They are considered a *valid data set.*) Note that The current face detector and eye localizer were trained only on frontal view faces. They nevertheless can handle faces within 30 degree reasonably reliably. Indeed, most of the failures occurred for faces with large rotation/tilt angle (45 degree or more).

Of the valid data set, 20% were used as training set for the face recognizer. In the test phase, we have used 60 images per person for testing. That yields 2400 images for 40 persons. To ensure fairness, the 2400 images were randomly picked from the entire database (excluding those used in the training set). As a result, only 2176 images of the chosen were from the valid data set. Nevertheless, the recognition rate remains very high (98%).

4.4 Performance in Training Time and Recognition Speed

The processing speeds of DBNN face recognizer are fast for both training and recognition. The training times for all the three databases are less than one hour on SUN Sparc10 workstation. As to the recognition speed, for the whole recognition process including eye localization, feature extraction, and classification using DBNN, less than 0.2 second will be needed, again based on the Sparc10 processing speed.

5 Application to Palm Recognition

Facial biometric feature has difficulty incurred by often uncontrollable variations on lighting, orientation, size, facial expression, and aging. Recently, a system that makes use of infrared images of faces is being developed. It matches heat patterns around the forehead and eyes to file images. The advantage as claimed lies in the observation that the patterns that do not change with age. However, in order to circumvent all the variations just mentioned, palm patterns offers a relatively more reliable biometric feature.

We are developing a DBNN-based palm recognition system by a similar DBNN-based technique. The preliminary study verifies that the DBNN recognizer is effective for palm recognition. Figure 2 illustrates the system diagram. When a palm is placed in front of the camera, the system will extract the edge feature from the image, down-sample it to a 18 by 16 dimensional feature vector, which will be processed by a DBNN-based palm recognizer. The system can also provide indication on whether this palm belongs to the

permissible database or a possible intruder. For our preliminary study, a small (32x3) palm database was created. Each of the 32 persons has 3 images of his/her right palm taken, one will be included in the training set, and the other two in the testing set. Only one out of 64 test images is mis-recognized, i.e. nearly 99% recognition rate.

References

[1] A.Pentland, B.Moghaddam, T.Starner, and M.Turk. View-based and modular eigenspaces for face recognition. In *Proc. IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition*, pages 84–91, 1994.

[2] A.Rahardja, A.Sowmya, and W.Wilson. A neural network approach to component versus holistic recognition of facial expressions in images. *SPIE Proc.: Intell. Robots and Computer Vision X: Algorithms and Techn.*, 1607:62–70, 1991.

[3] B.A.Golomb and T.J.Sejnowski. SEXNET: A neural network identifies sex from human faces. In D.S.Touretzky and R.Lipmann, editors, *Advances in Neural Information Proceedings Systems 3*, San Mateo, CA, 1991. Morgan Kaufmann.

[4] J.Weng, T.S.Huang, and N.Ahuja. Learning recognition and segmentation of 3D objects from 2D images. In *Proc. IEEE Int. Conf. on Computer Vision*, pages 121–128, 1993.

[5] S.Y. Kung and J.S. Taur. Decision-Based Neural Networks with Signal/Image Classification Applicationsf. *IEEE Trans. on Neural Networks*, 6(1):170–181, 1995.

[6] Shang-Hung Lin, S.Y. Kung, and Long-Ji Lin. A probabilistic DBNN with applications to sensor fusion and object recognition. In *NNSP'95*.

[7] M.D.Kelly. Visual identification of people by computer. Technical Report AI-130, Stanford AI Proj., 1970.

[8] M.Feng, A.Singh, and M.-Y.Chiu. A Fast Method for Eye Localization. Technical Report SCR-94-TR-488, Siemens Technical Report, 1994.

[9] M.Lades, J.Vorbruggen, J.Buhmann, J.Lange, and C.v.d.Malsburg. Distortion invariant object recognition in dynamic link architecture. *IEEE Trans. Computers*, 42:300–311, 1993.

[10] R.Brunelli and T.Poggio. Face recognition: Faatures versus templates. *IEEE Trans. Patt. Anal. and Mach. Intell.*, 15:1042–1052, 1993.

[11] R.Chellappa, C.L.Wilson, and S.Sirohey. Human and Machine Recognition of Faces: A Survey. *Proc. of the IEEE*, 83(5), May 1995.

[12] S.Akamatsu, T.Sasaki, H.Fukamachi, and Y.Suenaga. A robust face identification scheme-KL expansion of an invariant feature space. *SPIE Proc.:Intell. Robots and Computer Vision X: Algorithms and Techn.*, 1607:71–84, 1991.

[13] S.Y.Kung, M.Fang, S.P.Liou, and J.S.Taur. Decision-based neural network for face recognition system. In *ICIP'95*.

[14] Y.Cheng, K.Liu, J.Yang, and H.Wang. A robust algebraic method for face recognition. In *Proc. 11th Int. Conf. on Patt. Recog.*, pages 221–224, 1992.

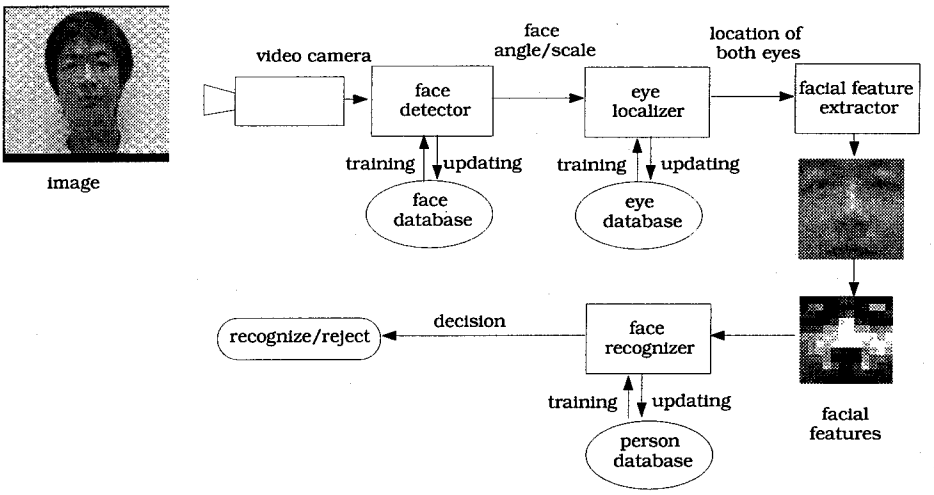


Figure 1: System flow chart of face recognition system. Face recognition system acquires images from video camera. Face detector determines if there are faces inside images. Eye localizer indicates the exact positions of both eyes, and pass their coordinates to facial feature extractor to crop images and take out facial features.

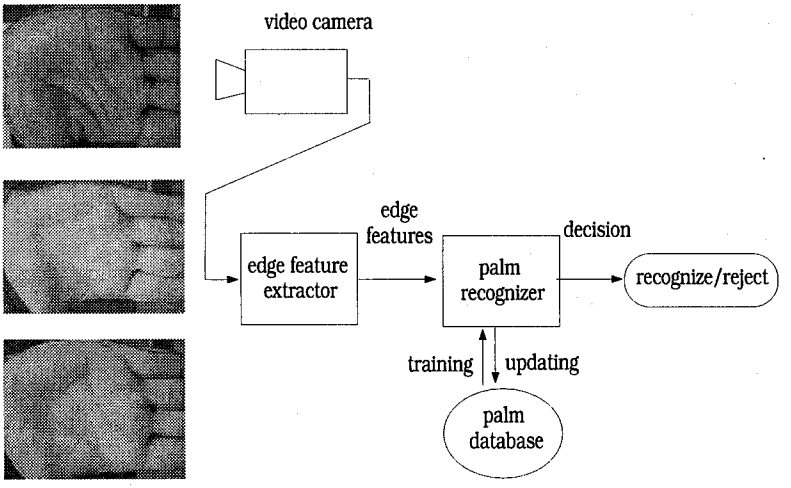


Figure 2: DBNN-based palm recognition system.

A Probabilistic DBNN with Applications to Sensor Fusion and Object Recognition*

Shang-Hung Lin S. Y. Kung Long-Ji Lin
Princeton University Siemens SCR

Abstract

Given an input vector say x, a classifier is supposed to tell which class is most likely to have produced it. Thus most data classifiers are designed to have K output nodes, each corresponding to to one of K classes, $\{\omega_i : i = 1, \cdots, K\}$. When pattern classes are clearly separated, this kind of data classifier usually performs very well. A specific model is the Decision-Based Neural Network (DBNN)[2], which has been shown to be an effective classifier in many signal/image classification applications. This is particularly the case when pattern classes are clearly separable. However, for those recognition applications which has complex pattern distribution with two or more classes overlapping in pattern space, the traditional DBNN may not be effective or appropriate. For such applications, it is preferable to adopt a probabilistic classifier. In this paper, we develop a new probabilistic variant of the DBNN, which is meant for better estimate probability density functions corresponding to different pattern classes. For this purpose, new learning rules for probabilistic DBNN are derived. In our experiments on public (FERET) and in-house (SCR) face databases, we have observed noticeable improvement in various performance measures such as recognition accuracies and, in particular, false acceptance/rejection rates. Taking advantage of probabilistic output values of the DBNN, we construct a multiple sensor fusion system for object classification. In a sense, it represents an extension of the traditional hierarchical structure of DBNN. This will be discussed in section 3, where experimental results on car recognition will also be reported.

1 Introduction

Basic Gaussian Formulation Suppose the likelihood density of input x given class ω_i is a D-dimensional Gaussian distribution, then posterior prob-

*This research was supported in part by the Electronic Eye Program. The authors also wish to acknowledge the invaluable contributions by Drs. M. Fang and I. Chakravarty of SCR.

0-7803-2739-X/95 $4.00 © 1995 IEEE

ability $P(\omega_i|x)$ by Bayes rule:

$$P(\omega_i|x) = \frac{P(\omega_i)p(x|\omega_i)}{p(x)}, \quad p(x|\omega_i) = N(\mu_i, \Sigma_i)$$

where $P(\omega_i)$ is the prior probability of class ω_i ($\sum_{i=1}^{K} P(\omega_i) = 1$), and $p(x) = \sum_{k=1}^{K} P(\omega_k)p(x|\omega_k)$.

The class likelihood function $p(x|\omega_i)$ can be extended to mixture of Gaussian distributions. Define $p(x|\omega_i, \Theta_r)$ to be one of the Gaussian distributions which consist of $p(x|\omega_i)$, where $\Theta_r = \{\mu_r, \Sigma_r\}$ is the parameter set for this distribution.

$$p(x|\omega_i) = \sum_{r=1}^{R} P(\Theta_r|\omega_i)p(x|\omega_i, \Theta_r)$$

where $P(\Theta_r|\omega_i)$ is the prior probability of cluster r when input patterns are from class ω_i. By definition $\sum_{r=1}^{R} P(\Theta_r|\omega_i) = 1$.

In this paper, we propose modifying each subnet in a tradition DBNN (cf. Figure 1(a)) into a probabilistic subnet, shown in Figure 1(b). The purpose of this modification is to make the output of the subnet i emulate $P(\omega_i|x)$. If so, then the new DBNN becomes essentially a Bayes classifier. Notice that the weighted sum of each subnet in multi-cluster DBNN is performed *after* the exponential normalization. In the special case, when $P(\omega_i) = P(\omega_j)$, the subnet output becomes $y_i(x) = \frac{e^{\phi(x,\omega_i)}}{\sum_{k=1}^{K} e^{\phi(x,\omega_k)}}$, a *softmax* function[1].

Elliptic Basis Function(EBF) In most general formulation, the basis function of a cluster should be able to approximate the Gaussian distribution with full-rank covariance matrix, i.e., $\phi(x, \omega_i) = -\frac{1}{2}\mathrm{x}^T\Sigma_i\mathrm{x}$, where Σ_i is the covariance matrix. A hyper-basis function (HyperBF) is meant for this. However, for those applications which deal with high dimensional data but finite number of training patterns, the training performance and storage space discourage such matrix modelling. A natural simplifying assumption is to assume uncorrelated features of unequal importance. That is, suppose that $p(x|\omega_i, \Theta_r)$ is a D-dimensional Gaussian distribution with uncorrelated features, that is,

$$p(x|\omega_i, \Theta_r) = \frac{1}{(2\pi)^{D/2}\prod_d^D \sigma_{id}} exp(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - w_{id})^2}{\sigma_{id}^2}) \sim N(\mu_i, \Sigma_i) \quad (1)$$

where $\mu_i = [w_{i1}, w_{i2}, \cdots, w_{iD}]^T$ is the mean vector, and diagonal matrix $\Sigma = diag[\sigma_{i1}^2, \sigma_{i2}^2, \cdots, \sigma_{iD}^2]$ is the covariance matrix.

As described in [2] and [6], DBNN has K subnets for K-class classification problems. Inside each class subnet the elliptic basis functions (EBF) is used to serve as the discriminant function for each cluster:

$$\phi(x, \omega_i, \Theta_r) = -\frac{1}{2}\sum_{d=1}^{D}\alpha_{id}(x_d - w_{id})^2 + \theta_i \quad (2)$$

If θ_i is set to $\theta_i = -\frac{D}{2}\ln 2\pi + \frac{1}{2}\sum_{d=1}^{D}\ln\alpha_{id}$ then $exp\{\phi(x,\omega_i,\Theta_r)\}$ can be viewed the same Gaussian distribution as described in equation 1, except a minor notational change: $\frac{1}{\alpha_{id}} = \sigma_{id}^2$.

2 Probabilistic DBNN

Let us now look into a log-likelihood formulation for the DBNN. To model the log-likelihood functions, it is desirable to have the output of a neural subnet approximates

$$y_i(x) = \log p(x|\omega_i) = \log[\sum_r P(\Theta_r|\omega_i)p(x|\Theta_r,\omega_i)] \tag{3}$$

Referring to Figure 1(b), the final node will be a nonlinear log operator. Note that an explicit teacher value would not be required, although it is a supervised training because teacher's knowledge on correct classification is crucial in the training.

Learning Rules for Probabilistic DBNN The training scheme for probabilistic DBNN has two phases. The Locally Unsupervised (LU) phase is the same as the original version[2, 6]. Several unsupervised learning schemes (e.g., LVQ, k-mean, EM...) can be applied to decide the initial values of parameters. The second phase is Globally Supervised (GS) learning. The goal of the GS learning is to increase $y_i(x)$ if $x \in \omega_i$ (reinforced learning) and decrease $y_i(x)$ if $x \notin \omega_i$ (antireinforced learning). We can use gradient ascent method in the GS phase:

$$\begin{aligned}
\frac{\partial y_i(x)}{\partial \mu_j^{(r)}} &= \pm h_j(x)\cdot\alpha_j^{(r)}(x_j - \mu_j^{(r)}) \\
\frac{\partial y_i(x)}{\partial \alpha_j^{(r)}} &= \pm h_j(x)\cdot\frac{1}{2}(\frac{1}{\alpha_j^{(r)}} - (x_j - \mu_j^{(r)})^2)
\end{aligned} \tag{4}$$

where

$$h_j(x) = \frac{P(\Theta_k|\omega_i)p(x|\omega_i,C_k)}{\sum_r P(\Theta_r|\omega_i)p(x|\omega_i,\Theta_r)}$$

Here $\pm$ are for reinforced and anti-reinforced learning respectively. $P(\Theta_k|\omega_i)$ (and $P(\omega_i)$, if necessary) can be learned by the EM algorithm [3]: At epoch j,

$$\begin{aligned}
h_k^{(j)}(t) &= \frac{P(\Theta_k|\omega_i)^{(j)}p(x^{(t)}|\omega_i,C_k)}{\sum_r P(\Theta_r|\omega_i)^{(j)}p(x^{(t)}|\omega_i,\Theta_r)} \\
P(\omega_k)^{(j+1)} &= (1/N)\sum_{t=1}^{N} P(\omega_k|x^{(t)})^{(j)}
\end{aligned}$$

$$P(\Theta_k|\omega_i)^{(j+1)} = (1/N)\sum_{t=1}^{N} h_k^{(j)}(t) \quad (5)$$

False Rejection/Acceptance False rejection/acceptance is an important issue in signal detection. For data classification problems, in some applications it is possible that the input pattern does not belong to any of the classes in the database. For such kinds of applications, the feature space will not be simply divided into K regions (classes). There will be many "open" spaces left for unknown (and unseen) classes i.e. classes which do not appear in the training set. Therefore the problem of false rejection/acceptance has to be considered in those applications.

The probabilistic DBNN is very suitable for tackling false rejection/acceptance problem. The reason is that probabilistic DBNN focus only on density distributions in individual classes (rather than global partitioning), no additional stranger class subnet is needed. Also, since density distribution usually drops as the distance to the center goes farther, the decision regions tend to be more locally conserved, and thus a lower false acceptance rate will be expected. Numerous experiments indicate this is the case.

Since DBNN provides probabilistic outputs, we can follow the similar procedure to the Neyman-Pearson hypothesis testing by setting a threshold to network outputs and computing the probability of false acceptance and false rejection. In order to find out the most possible regions for patterns from class ω_i, it is recommended to choose a threshold T_i so that an input x is classified to class ω_i if $\log p(x|\omega_i) > T_i$. If we set T too low, then the false acceptance rate will increase. In contrast, if T is set too high, then the false rejection rate will raise. So an adaptive learning rule to train the threshold T is preferred, instead of having the user preassign the threshold values. For an input x, if $x \in \omega_i$ but $\log p(x|\omega_i) < T_i$, then T_i should reduce its value. On the other hand, if $x \notin \omega_i$ but $\log p(x|\omega_i) > T_i$, then T_i should increase. Here we propose a learning scheme for thresholds: Define $d \equiv T_i - \log p(x|\omega)$. Also define a cost function $l(d)$. $l(d)$ can be either a step function, a linear function, or a fuzzy-decision sigmoidal function. Once the network finishes the training, the threshold values can be trained as follows: Given positive learning parameter η, at step j,

$$\begin{aligned} T_i^{(j+1)} &= T_i^{(j)} - \eta l'(d) \quad \text{if } x \in \omega_i \quad \text{(reinforced learning)} \\ T_i^{(j+1)} &= T_i^{(j)} + \eta l'(d) \quad \text{if } x \notin \omega_i \quad \text{(antireinforced learning)} \end{aligned} \quad (6)$$

Experimental Results: Face Recognition system At SCR and Princeton University, a DBNN-based face recognition system[6] is being developed, which consists of three core modules: face detector, eye localizer, and face recognizer. In this paper, we will focus on comparing the face recognition performances by the traditional DBNN, probabilistic DBNN, and MLP.

	Training Accuracy	Testing Accuracy
Probabilistic DBNN	97%	99%
Traditional DBNN	100%	96%
Multi-Layer Perceptron	99.5%	87.5%

Table 1: Performance of different face recognizers on FERET database.

	Traditional DBNN	Probabilistic DBNN
Success	97.92%	98.34%
Correct reject	0.96%	0.59%
False reject	0.40%	0.38%
Misclassify	0.72%	0.69%

Table 2: Performance of face recognition system using database with large head orientation.

Experiment 1: Frontal View Faces We have conducted the experiment on two image database: the SCR 80x20 database and the ARPA/ARL FERET database. More detail about the SCR 80x20 database can be found in [6]. For the 1275 images which successfully passed the eye localizer, face recognizer can achieve 100% recognition rate, no matter it was implemented by traditional DBNN or probabilistic DBNN.

As to the experiments on the ARPA/ARL FERET database, 200 persons have qualified for our front-view face recognizer experiment (see [6]). One image per person is used for training and the other for testing. The face recognizer experimental results are shown in table 1. We can see that under reasonably high training accuracy (97%), probabilistic DBNN achieved higher recognition rate (99%) than traditional DBNN (96%). We also tried to use Multi-Layer Perceptron(MLP) to implement face recognizer. The performance is inferior to both types of DBNN.

Experiment 2: Faces with Large Variations The experiment is on the SCR 40x150, database, which was discussed in [6]. Trained by the so-called *valid data set* and tested under a broader dataset, cf. [6], the face recognition results of the two DBNN models are summarized in Table 2. The recognition rate is very high for both types of DBNN. The results also indicate that confidence scores can be used to select "good" images for final recognition when a sequence of images are available and not all of them are in good quality.

	w/o negative examples	w/ negative examples
Probabilistic DBNN	13.75%	8.13%
Traditional DBNN	33.75%	22.5%
Multi-layer Perceptron	33.75%	12.5%

Table 3: False acceptance rates of face patterns on various types of neural network.

Experiment 3: False Acceptance/Rejection Sometimes it is more crucial for a recognition system to have the ability to reject strangers. Thus the false acceptance rate and false rejection rate of the system are important performance metrics. We have conducted an experiment for false acceptance/rejection. Among the 80-person frontal-view database, we have chosen 20 persons as our "known person" database and picked up 40 persons from the rest 60 to be the "stranger" database. The remaining 20 persons serve as "negative examples" for the network training. The training datasets are formed by randomly choosing 2 images per person from database and then generating 50 *virtual training patterns*[6] from each images. Both "known person" and "negative" training sets have $20 \times 2 \times 50$ training patterns. The test sets are formed by selecting 4 images from each person. There are 20×4 test patterns from known person database and 40×4 patterns from stranger database. If a test pattern is from known person database but is rejected by the network, then the pattern is falsely rejected. On the other hand, if the pattern is from stranger database but is misrecognized to one of the person in the known person database, then this pattern is falsely accepted.

We have compared the performance of probabilistic DBNN, traditional DBNN[2] and multi-layer perceptron (MLP). The training algorithm of DBNNs are shown in section 2 and [2, 6]. MLP is trained by back-propagation algorithm. We have chosen the number of clusters (or hidden neurons) which can generate the best performance. There are 2 clusters for each subnet in either probabilistic or traditional DBNN, and there are 30 hidden neurons in MLP. The experiment results are as follows. After training, all three networks reach 0% false rejection rate. The false acceptance rates are shown in Table 3. As mentioned in section 2, likelihood density generates more locally conserved decision regions than posterior probability. Therefore, the probabilistic DBNN can achieve lower false acceptance rate than the traditional DBNN and MLP, whose outputs can be proven to converge to posterior probabilities[4, 5]. We also note that using negative training examples can significantly improve false acceptance rate. This effect can be observed in all of the three networks.

3 Multi-Channel Fusion Networks

The problem of combining the classification power of several classifiers is of great importance to various applications. First, for several recognition problems, numerous types of features could be used to represent and recognize patterns. Also, for those applications which deal with high dimensional feature data, it makes sense to divide feature vector into several lower dimensional vectors. By extending the hierarchical structure of DBNN, we propose several multi-classifier fusion networks. The multi-classifier DBNN consists of several "classifier channels", each of which receives input vectors either from different sensors or from a portion of a higher dimensional feature vector. The outputs of channels are combined by some proper weightings. The weighting factor is assigned based on the *confidence* the corresponding channel has on its recognition result. Since DBNN generates probabilistic outputs, it is natural to design the channel weightings to have probability properties. The overall configuration of multi-channel fusion network is depicted in Figure 2(a), where the score functions from two channels are combined after some proper pre-weightings.

In the so-called **class-dependent channel fusion**, the weighting factors correspond to the confidence $P(C_k|\omega_i)$ for each channel. Here $P(C_k|\omega_i)$ represents the indicator on the confidence on channel k when the test pattern is originated from the ω_i class. (By definition, $\sum_{k=1}^{K} P(C_k|\omega_i) = 1$, so it has the property of a probability function.) Suppose that there are K channels in the subnet ω_i, and within each channel there are R clusters. The probability model of the DBNN-based channel fusion network can be described as follows. (See Figure 2(b).)

$$p(x|\omega_i) = \sum_{k=1}^{K} P(C_k|\omega_i) p(x|\omega_i, C_k)$$

where $p(x|\omega_i, C_k)$ is the output of subnet i in channel k, and $p(x|\omega_i)$ is the combined output for class ω_i. Note that $x = [x_1^T, \cdots, x_K^T]^T$, and since $p(x|\omega_i, C_k)$ is conditional on C_k, only x_k is involved in the above formula. After all the parameters within channels complete their training, channel confidence $P(C_k|\omega_i)$ can be learned by the following: Define $\alpha_k = P(C_k|\omega_i)$. At beginning, assign $\alpha_k = 1/K$, $\forall k = 1, \ldots, K$. At step j,

$$h_k^{(j)}(t) = \frac{\alpha_k^{(j)} p(x^{(t)}|\omega_i, C_k)}{\sum_r \alpha_r^{(j)} p(x^{(t)}|\omega_i, C_r)} \qquad \alpha_k^{(j+1)} = (1/N) \sum_{t=1}^{N} h_k^{(j)}(t) \qquad (7)$$

Once the NN is trained, then the fusion weights will remain constant during the retrieving phase.

A more general version of multi-channel fusion is called **data-dependent channel fusion**. Instead of using the likelihood of observing x given a class ($p(x|\omega_i, C_k)$) to model the discriminant function of each cluster, we shall use

the posterior probabilities of electing a class given x ($p(\omega_i|x, C_k)$). For this version of multi-channel network, a new confidence $P(C_k|x)$ is assigned, which stands for the confidence we have on channel k when the input pattern is x. Accordingly, the probability model is also modified to become

$$P(\omega_i|x) = \sum_{k=1}^{K} P(C_k|x)P(\omega_i|x, C_k)$$

$P(\omega_i|x, C_k)$ can be obtained by $P(\omega_i|x, C_k) = P(\omega_i|C_k)p(x|\omega_i, C_k)/p(x|C_k)$, and the confidence $P(C_k|x)$ can be obtained by the following equations:

$$P(C_k|x) = \frac{P(C_k)p(x|C_k)}{\sum_l P(C_l)p(x|C_l)}$$

where $p(x|C_k)$ can be computed straightforwardly by equation $p(x|C_k) = \sum_i P(\omega_i|C_k)p(x|\omega_i, C_k)$ and $P(C_k)$ can be learned by 7 (but replace $p(x|\omega_i, C_k)$ with $p(x|C_k)$). The term $P(C_k)$ can be interpreted as "the general confidence" we have on channel k. Unlike the class-dependent approach, the fusion weights need to be computed for each testing pattern during the retrieving phase.

Multi-Channel Data Fusion for Object Recognition We have conducted a preliminary experiment for testing the performance of the multi-channel network shown in Figure 2(a), where the class-dependent fusion is adopted. To facilitate the training phase, we employ (what we consider to be) faster DBNN training, based on EBF (elliptic basis function) modules. In order to obtain a probability function expression, a normalizing scaling factor λ will be introduced for every channel in each class used. In the fusion phase, we made the following simplifying approximation: we use the dominant cluster to express the probability function $p(x|\omega_i, \Theta_r, C_k)$. Such a simplification leads to the following formulation:

$$p(x|\omega_i, C_k) \approx \lambda Exp[-\sum_{k=1}^{N} \beta_{lk}(x_k - w_{lk})^2 + \theta_l]$$

A simple experiment was conducted: Six car models from different view angles were used to create the data base, cf. Figure 3(a). 28 to 29 256 $\times$ 256 pixel sized images were taken for each car model from various viewing directions. Two sensor channels were built from two different feature extraction methods: one uses intensity information and the other edge information, cf. Figure 3(b). After proper size/illumination normalization, images were down-sampled to 12 $\times$ 12 pixel size. The down-sampled pixels were used as input features for each of the sensor channels. Ten different simulations are conducted. Each one randomly choose 90% of the images (within each class) for training DBNN. The remaining 10% were saved as testing patterns. We then took the

average of the error rates of the 10 simulations. According to our simulation, the fusion of two channels (with 94% and 85% recognition rate each), the recognition rate reached 100%. At least in this experiment, the proposed class-dependent channel fusion did improve the recognition performance over individual channel classifiers.

References

[1] J.S. Bridle. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters . *Neural Information Processing Systems*, 2:211–217.

[2] S.Y. Kung and J.S. Taur. Decision-Based Neural Networks with Signal/Image Classification Applications . *IEEE Trans. on Neural Networks*, 6(1):170–181, Jan 1995.

[3] Shang-Hung Lin and S.Y.Kung. Sensor Fusion via Expectation-Maximization (EM) on Hierarchical DBNN. In *ICIP'95*.

[4] M.D. Richard and R.P. Lippmann. Neural Network Classifiers Estimate Bayesian *a posteriori* Probabilities. *Neural Computation*, (3):461–483, 1991.

[5] D.W. Ruck, S.K. Rogers, and et al. The Multilayer Perceptron as an Approximation to a Bayes Optimal Discriminant Function. *IEEE Trans. on Neural Networks*, 1(4):296–298, Dec 1990.

[6] S.Y.Kung, M.Fang, and J.S. Taur. Decision-Based Neural Networks for Face Recognition. In *NNSP'95*.

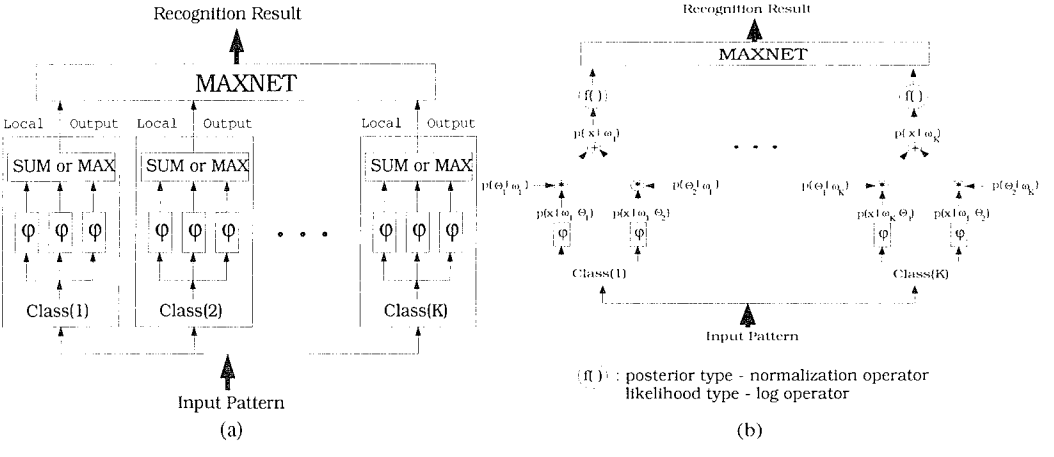


Figure 1: (a) Schematic diagram of DBNN. (b) Structure of Probabilistic DBNN.

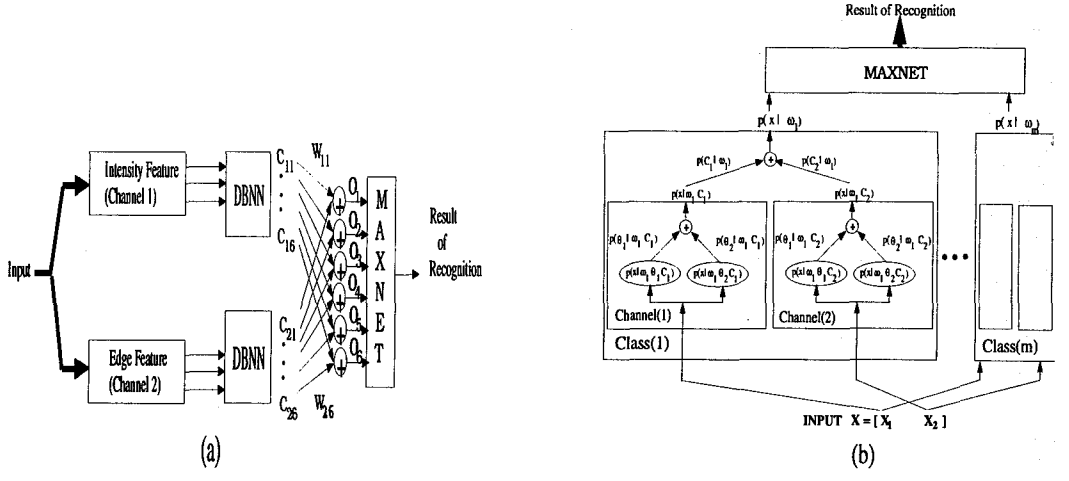


Figure 2: (a) An exemplar DBNN-based multi-channel fusion network. Here C_{ki} denotes the output of the i-th subnet in channel k ($= p(x|\omega_i, C_k)$) and W_{ki} ($= P(C_k|\omega_i)$) denotes the channel confidence. The final decision is based on O_i which is the combination of the weighted outputs for subnet i. Theoretically, O_i can be equated to $p(x|\omega_i)$. (b) The derivation of $p(x|\omega_i)$ may be better explained by this probability model for DBNN, see Section 1. For the discussion on multi-channel DBNN, see Section 3.

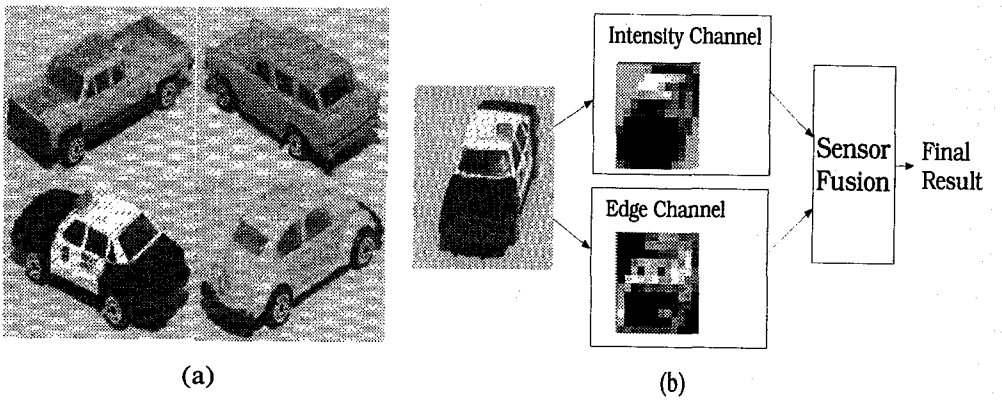


Figure 3: (a) Some examples of car models. (b) Given an original image, two sensor channels are used to extract features. The recognition is then obtained by channel fusion.

SAMPLE WEIGHTING WHEN TRAINING SELF-ORGANIZING MAPS FOR IMAGE COMPRESSION

Jari Kangas
Helsinki University of Technology
Neural Networks Research Centre
Rakentajanaukio 2 C, FIN-02150, ESPOO, FINLAND
tel: +358 0 451 3275, fax: +358 0 451 3277
email: Jari.Kangas@hut.fi

Abstract — Image compression is an essential task for image storage and transmission applications. Vector quantization is often used when high compression rates are needed. Self-Organizing Map (SOM) algorithm can be used to generate codebooks for vector quantization. Previously it has been demonstrated that using the special property of the SOM algorithm that the codebook entries are ordered one can use prediction coding of codewords to make the compression more effective. In this paper it is shown that training the SOM algorithm by using different weighting for sample blocks having different statistical characteristics one can further increase the compression efficiency.

IMAGE COMPRESSION BY SELF-ORGANIZING MAPS

Image compression is an essential task for image storage and transmission [1]. Lately the image compression using Vector Quantization (VQ) techniques has received large interest [2]. VQ methods offer good performance when high compression rates are needed. In VQ approaches adjacent pixels are taken as a single block, which is mapped into a finite set of codewords. In decoding stage the codewords are replaced by corresponding model vectors. The set of codewords and the associated model vectors together is called a codebook. In VQ the correlation which exists between adjacent pixels in a block is taken into account, and with a comparatively small codebook one achieves a small quantization error in reconstructed image.

The main problem in vector quantization is to find a codebook which minimizes the quantization mean error. Many design algorithms have been proposed. One of the best known is the Linde-Buzo-Gray (LBG) algorithm [3], which iteratively searches clusters in the training data. The cluster centers are used as the codebook model vectors, while the

0-7803-2739-X/95 $4.00 © 1995 IEEE

codewords are selected arbitrarily. An unsupervised neural network algorithm, the Self-Organized Map (SOM) algorithm [4][5] has also been used. In SOM the unit model vectors form the codebook, and the unit coordinates are used as codewords. Comparisons of the distortion errors in using the codebooks produced by either the SOM algorithm or the LBG algorithm indicate that these two methods give rather similar performance [6][7]. In [7], the SOM algorithm was found to be more robust than the LBG algorithm to initialization.

The main difference between LBG and SOM algorithms is in the order of the codebook model vectors. The LBG algorithm does not define any order in the codebook; the codewords for model vectors can be selected arbitrarily. On the other hand, the codebook trained using the SOM algorithm has an internal order; adjacent codebook entries which have similar codewords, have similar codebook model vectors.

Lately, it has been recognized, that the ordered codebooks can be used to improve the image compression efficiency [8][9][10]. Because the neighboring pixels in images correlate to each other, one can use the knowledge of the previous pixel value to predict the next pixel value, and code only the difference between the prediction and the real value. Therefore one can reduce the transmission rate considerably. Because the similarities between adjacent blocks in the original image are converted to similarities between the corresponding codewords, the prediction coding can now be used together with the vector quantization algorithm by predict coding the codewords (instead of pixel values). One can then improve the compression rates attainable by vector quantization by 15 to 20 % [10].

IMPROVING THE EDGE PRESERVATION IN IMAGE VECTOR QUANTIZATION BY SAMPLE OCCURRANCE WEIGHTING

Edge preservation is a serious problem in vector quantization of image data. The design algorithms tend to find good model vectors for samples which occur frequently but rare samples are not well represented. The edges in images form such a rare group of samples, where there are only few instances of separate edge orientations and amplitudes. However, the edges in images are important for the human acceptance of the decoding results. Ragged and uneven edges in images are very easily noticed and decrease the subjective quality of compression result, altough the signal-to-noise rations may show rather good decoding quality.

In [11] it was proposed that by modifying the training law of the SOM algorithm used in training the codebooks, one can improve the edge preservation capabilities. The proposal was based on the heuristic rule that one should pay more notice to image blocks where there are edges

or other 'features'. The 'edgeness' of the block was computed using a variance measure of the gray level distribution of pixels x_k inside the block x:

$$\sigma^2 = \frac{1}{N}\sum_{k=1}^{N} x_k^2 - \bar{x}^2, \tag{1}$$

where $\bar{x}$ is the mean value of gray levels in block and N is the number of pixels. In [11] the variance was then used to compute a weighting factor $w(\sigma)$, which was considered as multiplying the number of occurrences of the sample x. In the following experiments we adopted a similar approach using (2) to compute the multiplier for sample count:

$$w(\sigma) = (\text{max_weight} - 1.0)\frac{\sigma^2}{\sigma^2_{\max}} + 1.0, \tag{2}$$

where $\sigma^2_{\max}$ is the maximum variance value possible. Thus using the 'max_weight' value of 1.0, the algorithm behaves as the original.

Results with edge preservation

We used codebooks with 512 entries and coded 4 by 4 pixel subimages, which gave a 0.5625 bpp (bits per pixel) raw bit rate on the image compression. The codebooks were trained using 28 images of faces (both males and females) and tested using another face image, which was not contained in the training set. All the images were 512 by 680 pixel 256 gray level images. The topology of the Self-Organizing Map was three dimensional having eight units on each dimension.

The image degradation was measured using a peak signal-to-noise ration (PSNR) defined as:

$$\text{PSNR} = 10\log\frac{255^2}{\text{MSE}}\ \text{dB}, \tag{3}$$

where MSE was the mean square error. The resulting PSNR values and standard deviations of the results on ten training runs for each different weightings are collected to Table 1.

From the results one can easily see that the sample occurrence weighting improves image quality. The PSNR values grow up to weighting 20 and even after that up to weighting 200 the PSNR values are larger than using the standard version of the SOM algorithm.

In Figure 1 some details of decoded images are shown. The edge between ear and background has been decoded using codebooks which have been trained using weighting factors 1.0, 10.0, and 100.0, respectively. The continuity of the edge is improved when the weighting factor increases.

Max weight in training	PSNR value	Std. dev.
1	34.56	0.03
2	34.56	0.05
5	34.66	0.04
10	34.74	0.06
20	34.80	0.08
50	34.72	0.10
100	34.69	0.07
200	34.65	0.04
500	34.17	0.04
1000	33.82	0.00

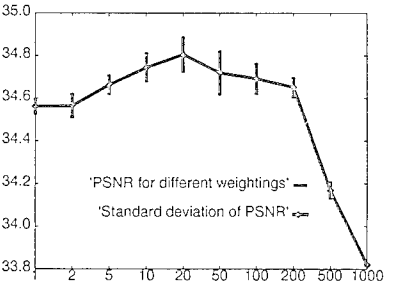


Table 1: *PSNR values in image coding and decoding experiments using different values for the maximum weighting. Ten independent training runs have been tried for each weighting.*

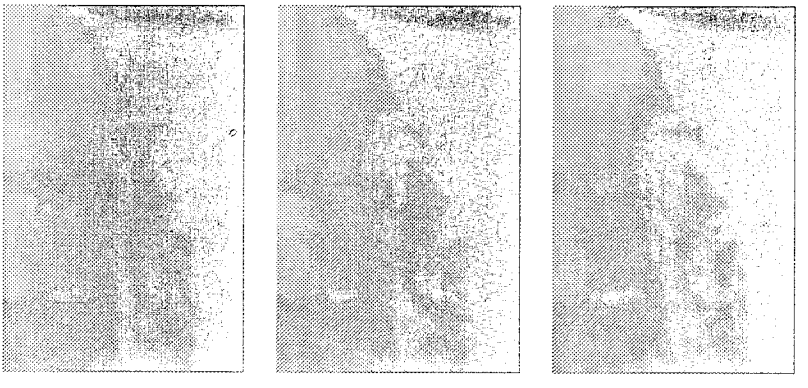


Figure 1: *The three images are details of decoded images where the compression codebooks have been trained using weighting factors 1.0, 10.0 and 100.0, respectively. Notice the increasing continuity of the edges in images from left to right.*

INCREASING THE COMPRESSION EFFICIENCY BY SAMPLE OCCURRANCE WEIGHTING

In modifying the sample count for SOM training one affects the distribution of the model vectors so that the flat blocks (where pixel gray

values are more or less the same) get fewer representatives in codebook compared to the original training algorithm. Therefore the number of different codewords used in coding flat areas in images is less than if the original codebook were used. If prediction coding of codewords is used, decreasing the number of possible codes leads to more compact coding of the flat areas in images.

In the following some exeriments are described where the previous idea has been tried. Test image has been compressed using different weightings in codebook training and prediction coding the resulting codeword sequences. The following results are means over ten training experiments.

Results with original training algorithm and prediction coding

The test image was first vector quantized using the original SOM algorithm. The raw bpp rate for the coding is 0.5625, but some improvement can already be gained if the frequency of occurrence of each codeword is taken into account (using for example Huffman coding). Because the test image was not used in training the codebook not all codewords are used equally probably. After Huffman coding the bpp rate for the test image was 0.544, which is 3 % less than the raw value.

The image is vector quantized sequentially, one row at a time. Because there naturally exists some correlation between adjacent blocks in the image, it is advantageous to make a prediction of the codebook vector which is going to be used based on codewords used for the nearby blocks and then code the prediction error only. In our experiments we computed the prediction $\hat{a}$ of the codeword $a(i,j)$ to be used in location (i,j) as in [9]:

$$\hat{a}(i,j) = \begin{cases} a(i,j-1), & \text{if } \begin{array}{r} d(a(i-1,j),a(i-1,j-1)) < \\ d(a(i,j-1),a(i-1,j-1)) \end{array} \\ a(i-1,j), & \text{otherwise,} \end{cases} \tag{4}$$

where the distance $d(a,b)$ between codes is defined on SOM as $d(a,b) = ||a-b||$. In image borders the prediction was based on available codewords.

After prediction and Huffman coding the codewords the bpp rate for the test image was 0.443. That means 19 % reduction in bpp from the Huffman coded raw codeword sequence, and 21 % reduction in bpp from the raw codewords. These figures are well in correspondence with the figures given in [10].

Results using the sample occurrence weighting

The bits per pixel rates and PSNR values of the experiments have been

collected to Table 2. Different weighting factors have been used in the SOM codebook training and prediction coding of the codewords has been utilized before computing the bpp rates. The results show that when the weighting increases, the compression factor will also increase. The bpp rates start to decrease rapidly when the maximum weighting is increased beyond value 20.

Max weight in training	Huffman coding	Prediction and Huffman coding	Decrease from raw code	PSNR value
1	0.544	0.443	21 %	34.56
2	0.539	0.437	22 %	34.56
5	0.544	0.439	22 %	34.66
10	0.543	0.430	24 %	34.74
20	0.545	0.425	24 %	34.80
50	0.543	0.406	28 %	34.72
100	0.540	0.382	32 %	34.69
200	0.540	0.359	36 %	34.65
500	0.527	0.317	44 %	34.17
1000	0.525	0.296	47 %	33.82

Table 2: *Bits per pixel rates and PSNR values using different values for the maximum weighting. The relative decrease in bpp rate is computed against the raw bpp rate 0.5625.*

The quantization error is in its minimum when the weighting is between 10 and 50. After that it starts to increase again. When the weighting is more than 200 the quantization error has significantly increased from that of smaller weightings. To get a similar quality of images as in original quantization algorithm, one can use weightings up to 200. At that level the relative decrease in bpp rate is about 36 %.

In Figure 2 bits per pixel rates and PSNR values for different weightings are shown in another format. The bpp rates are on the vertical axis and the lower points on the Figure mean better compression ration. PSNR values are on the horizontal axis and the best quality of images is on the right side.

DISCUSSION

The Self-Organizing Map can be used to train vector quantization codebooks for image compression. Because the SOM algorithm orders the codebook entries prediction coding can be used to further improve the compression efficiency. In this paper it has been shown that training the SOM algorithm by using different weightings for sample blocks having different statistical characteristics one can further increase the compres-

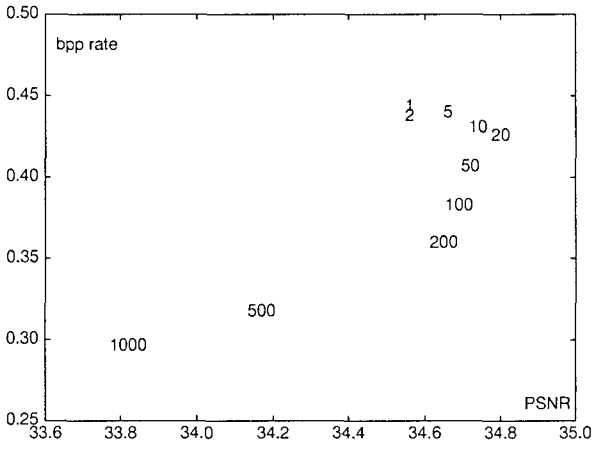


Figure 2: *Bits per pixel rates (after codeword prediction and Huffman coding) and PSNR values as a function of weighting factor in a two dimensional diagram.*

sion efficiency. Without affecting the image quality compression rates can be improved by 36 %. It is also possible to improve the image quality if somewhat smaller improvement in compression rates are allowed.

In these experiments only a variance of gray levels was computed of each image sample. Some other statistical measures might produce better results. For example, preserving the true lines going through the image sample is more important for subjective acceptance of the decoded image, than preserving random samples even if the variance of samples were the same. In future studies some other statistical measures are to be tested.

REFERENCES

[1] Anil K. Jain. Image data compression: A review. *Proc. IEEE*, 69(3):349–389, 1981.

[2] Nasser M. Nasrabadi and Robert A. King. Image coding using vector quantization: A review. *IEEE Trans. on Comm.*, 36(8):957–971, 1988.

[3] Robert M. Gray. Vector quantization. *IEEE Acoustics, Speech and Signal Processing Magazine*, 1:4–29, April 1984.

[4] Teuvo Kohonen. Self-organizing formation of topologically correct feature maps. *Biol. Cyb.*, 43(1):59–69, 1982.

[5] Teuvo Kohonen. The self-organizing map. *Proc. IEEE*, 78:1464–1480, 1990.

[6] Nasser M. Nasrabadi and Yushu Feng. Vector quantization of images based upon the Kohonen self-organization feature maps. *Neu-*

ral Networks, 1(1 SUPPL):518, 1988.

[7] J. D. McAuliffe, L. E. Atlas, and C. Rivera. A comparison of the LBG algorithm and Kohonen neural network paradigm for image vector quantization. In *Proc. ICASSP-90, Int. Conf. on Acoustics, Speech and Signal Processing*, volume IV, pages 2293–2296, IEEE Service Center, Piscataway, NJ, 1990.

[8] E. A. Riskin, L. E. Atlas, and S.-R. Lay. Ordered neural maps and their applications to data compression. In B. H. Juang, S. Y. Kung, and C. A. Kamm, editors, *Proc. Workshop on Neural Networks for Signal Processing*, pages 543–551, IEEE Service Center, Piscataway, NJ, 1991.

[9] Sergio Carrato, Giovanni L. Siguranza, and Luigi Manzo. Application of ordered codebooks to image coding. In *Neural Networks for Signal Processing. Proc. 1993 IEEE Workshop*, pages 291–300, IEEE Service Center, Piscataway, NJ, 1993.

[10] Gilles Burel and Jean-Yves Catros. Image compression using topological maps and MLP. In *Proc. ICNN'93, Int. Conf. on Neural Networks*, volume II, pages 727–731, IEEE Service Center, Piscataway, NJ, 1993.

[11] K. Y. Kim and J. B. Ra. Edge preserving vector quantization using self-organizing map based on adaptive learning. In *Proc. IJCNN-93-Nagoya, Int. Joint Conf. on Neural Networks*, volume II, pages 1219–1222, IEEE Service Center, Piscataway, NJ, 1993.

ESTIMATING IMAGE VELOCITY WITH *CONVECTED ACTIVATION PROFILES:* ANALYSIS AND IMPROVEMENTS FOR SPECIAL CASES

Robert K. Cunningham and Allen M. Waxman[1]

Machine Intelligence Group
MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02173

Dept. of Cognitive and Neural Systems
Boston University
111 Cummington Street
Boston, MA 02215

The Method of *Convected Activation Profiles* was developed to measure short-range visual motion of edge and point features in time-varying imagery. Each feature is assumed to generate a spatiotemporal Gaussian activation profile that results in a shape-preserved activity wave that is convected along with that feature, and the phase velocity of the wave provides a velocity estimate of the feature. By this method, both explicit feature tracking (a complex and computationally expensive operation) and the assumption that intensity is convected (which is rarely justified) are avoided. The method is suitable for real-time implementations [7,16] and can be described in terms of shunting dynamics of neural systems [7]. Spatiotemporal filters that measure the velocity of lines and points were described and demonstrated in the earlier work: this paper presents a detailed analysis of the accuracy of the method in scenes consisting of highly textured objects with fixed projections onto the image plane. We also describe how to accurately measure the velocity of short lines and line ends; in the past the velocity of short lines was severely underestimated, and the velocity of line ends could only be measured by recognizing line end features and evaluating the speed of these "point" features in isolation. This new method simplifies velocity extraction yet requires no additional computation. Finally, we clarify our earlier suggestion for selecting a velocity estimate from among several filters of different scales.

I. INTRODUCTION

Image flow is the projection on the image plane of object motion in the 3D world [15]. Any motion system which examines motion in large neighborhoods must

1. This work sponsored by the Air Force Office of Naval Research, Department of the Navy, and the Advanced Research Projects Agency under Air Force Contract F19628-95-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the author and are not necessarily endorsed by the United States Air Force.

0-7803-2739-X/95 $4.00 © 1995 IEEE

begin with a mechanism to obtain local estimates of motion. In this paper we concentrate on what has been traditionally classified as the short-range motion process [4], although we have also modeled the long-range motion process [6]. In addition, the technique used here to establish correspondence between proximal features across short times is closely related to another model of the long-range process [9]. There are two classes of short-range motion models, luminance-based models, which compare local filtered luminance at one position to local filtered luminance at a later time, and feature-based models, where geometrical features invariant to illumination are identified and tracked. Luminance-based models fall into three classes: correlation models [2,3,12], local phase velocity models [8], and gradient models [1,10,11]. Feature based models track features via explicit correlation [14] or by the growth and decay of a Gaussian activity wave as in the method described here. This method can provide direct velocity estimates by measuring the phase velocities of waves generated by and convected along with the features. The advantage of feature-tracking models is that the image flow features (e.g. edges and corners) usually represent projections of physical events such as object or texture boundaries that are stable and identifiable across short expanses of space and time. Luminance-based methods often assume that image intensity is convected along with the object itself. This is only true when an object with a lambertian surface is stable with respect to light sources, thus allowing only motion of the viewer when in man-made light conditions.

II. CONVECTED ACTIVATION PROFILES IN TWO DIMENSIONS

As formulated in the original model of Convected Activation profiles, each feature generates a wave of activity, which is convected along with the feature. The phase velocity of the wave is measured to obtain an estimate of the feature velocity itself. We model the activation function as a Gaussian spatiotemporal kernel, with a symmetric influence of space and time:

$$G^{\sigma,\tau} = Ce^{-\left[\frac{(x^2+y^2)}{2\sigma^2}+\frac{t^2}{2\tau^2}\right]}. \qquad (1)$$

(In later work we have modeled the activation kernel as a spatiotemporal Gaussian with a temporal component that acts as an exponential fading memory [7].) The activation distribution is modeled as the convolution of the activation function with the feature map:

$$A(x,y,t) = G^{\sigma,\tau}\{x,y,t\} * F(x,y,t) \ . \qquad (2)$$

To obtain the velocity estimate for an isolated point (with respect to the scale of the filter), compute (subscripts of A designate partial derivatives)

$$v_x(x, y, t) = -\left[\frac{A_{xt}A_{yy} - A_{yt}A_{xy}}{A_{xx}A_{yy} - (A_{xy})^2}\right]_{point} \quad \text{and}$$

$$v_y(x, y, t) = -\left[\frac{A_{yt}A_{xx} - A_{xt}A_{xy}}{A_{xx}A_{yy} - (A_{xy})^2}\right]_{point} . \qquad (3)$$

To obtain the normal velocity of an edge, compute, in local edge line coordinates (ξ, t), with ξ normal to an edge and η parallel to the edge:

$$\vec{v}_n(\xi, \eta, t) = -\left[\frac{A_{\xi, t}}{A_{\xi, \xi}}\right]_{line} . \qquad (4a)$$

If the line is long and in isolation (with respect to the scale of the filter), then we can convert from local edge coordinates to image coordinates:

$$\vec{v}_n(x, y, t) = -\left[\frac{(\nabla A)_t}{\nabla^2 A}\right]_{line} . \qquad (4b)$$

A more extensive derivation of these results can be found in [16], but some of the underlying assumptions are clear from even this abbreviated explanation.

First, there is the assumption of isolation, which says that only one line or point is present in a neighborhood on the scale of each spatiotemporal filter. Incorrect over- and under-estimates of velocity can occur when multiple features excite a single spatiotemporal filter. When a highly textured (with respect to the filter size) surface moves together as a whole, a complex activation profile is formed and convected. Equation *(3)* can be used to recover the underlying velocity, because these equations hold for the case of arbitrary motion of an arbitrary convected profile, *provided the features move together as a whole.* To obtain local velocity estimates, the components measured with *(3)* need to be interpreted with respect to the local texture features; in the case of a line, only the normal component (n_x, n_y, t) of the velocity of the line can be measured, but the wave may provide velocity measures for the entire neighborhood texture. To measure just the normal component of the line, compute:

$$v_n(x, y, t) = (n_x, n_y, t) \cdot (v_x, v_y, t)_{feature} . \qquad (5)$$

Interactions can be detected by examining the Gaussian curvature of the profile (i.e., the denominator of *(3)*). When the Gaussian curvature is positive (indicative of a convex or concave surface), equations *(3)* apply. When the Gaussian curvature is zero, the activation surface is locally either a cylinder or a plane. To differentiate the two, consider the mean curvature:

$$H = (1/2)(A_{xx} + A_{yy}) \qquad (6)$$

When the mean curvature is also zero, the surface is a plane and no velocity information is available. When H is non-zero, the activation surface is a cylinder, and *(4a)* and *(4b)* apply. Finally, consider the case where the Gaussian curvature is negative: this indicates that the activation profile is locally a saddle, which implies that severe interactions must be occurring on the scale of the filter. For the simulations presented in this paper, these velocity estimates will be ignored. Note that since we need to detect when H is zero, or the sign of H, we can omit the multiplication by 1/2, and we are left with the Laplacian (center-surround) operator that must be computed in any case for *(4a)* and *(4b)*. If we assume that *(3)* and *(4b)* are computed in parallel and only one (or none) is used at each location and scale, then the Laplacian and the Gaussian curvature can be used to select the appropriate local velocity estimate (i.e., to act as a gate). If no velocity estimate is available for a given feature at a given resolution, another scale must be examined.

The remaining assumptions are not obvious from the preceeding explanation, but were noted in [16]. In order to derive *(4a)* and *(4b)*, the line was assumed to be locally straight, to move along its normal, and to extend through the spatiotemporal filter. When the line does not extend all the way through the filter, the curvature ($A_{\eta\eta}$) along the line in the denominator of *(4b)* will be overestimated, resulting in further underestimation of the velocity. An accurate estimate is again provided by *(3)*. Application of the Gaussian/mean curvature test will indicate when the line is long enough and straight enough to use *(4b)*. In the next section we will show that this technique works well for lines that both translate and rotate in the image plane.

In an earlier paper, we suggested that if several different filters were all measuring the same underlying motion, a local competition between differing spatial scales could be based on the "characteristic speed normalization" of each filter, so that the choice would be based on:

$$Max_{\sigma,\tau}\{\frac{\tau}{\sigma}\left|v^{\sigma,\tau}\right|\}\ . \qquad (7)$$

This metric penalizes filters with large spatial scale in favor of filters with small spatial scale, thus reducing the opportunity for spatial interactions within a single measurement. Unfortunately, it also increases the opportunity for temporal interactions, by rewarding filters with large temporal scale. We intend to have this metric apply for filters with a variety of different σ at a single value of τ. In the human visual system the temporal scale τ appears to be set by the available light [5], and there are known to be cells tuned to a range of speeds. Below we show that when τ is allowed to vary, *(7)* does not always select the best filter.

III. SIMULATIONS AND RESULTS

In all simulations (except the first) the standard deviation of the spatial Gaussian is set to 4.0 pixels, and the standard deviation of the temporal Gaussian is set to 1.0

frame time. This implies that the filter will respond well to velocities less than or equal to about 4.0 pixels/frame. There is a trade-off between feature interactions and accurate velocity estimates. To minimize interactions, small filters should be used so only isolated lines or points activate a single filter. To capture the true velocity, the spatial scale of the filter must be large enough to capture feature displacement. Both of these constraints will be exceeded in the following simulations, and the resulting velocity estimates will be discussed.

We can describe the number of elements (E) in the activation mask (i.e., convolution) in terms of the standard deviation S and the number of standard deviations N present in the mask as:

$$E=SxNx2+1 \qquad (8)$$

The choice of N (and thus E) involves a trade-off between accuracy in representing the convected profile and computational demands: larger masks more accurately represent that profile, while smaller masks require less computation. To accurately measure velocity, the derivative of the activation function must be accurately modeled to the limit of its linear regime. For a moving feature that generates a Gaussian activity wave, the linear regime in the vicinity of the feature extends out to

$$O\left(\left(\sigma^2+\upsilon^2\tau^2\right)^{1/2}\right), \qquad (9)$$

where υ is the true velocity of the point. The extension over σ is due to the superposition of Gaussian activity from times other than the present and beyond N=1 standard deviations away. By choosing N=2, these interactions can occur (which enables the filter to represent speeds beyond its characteristic speed), and 95% of the total energy in an ideal Gaussian is present in the mask. This is adequate for all but those measurements at the upper limit of the filter's response, and is used here in all simulations except those that explicitly examine mask size.

Mask size affects the size of the linear regime of activation about a feature point, which in turn affects the maximum velocities that can be measured by a filter. When the mask size is limited, the linear regime can only be extended to $Nx\sigma$, because this is the limit of the spatial interactions at one moment in time. This implies that no filter should ever indicate a velocity greater than $(Nx\sigma)/\tau$; such a response would be due to quantization effects and can be safely ignored.

The simulations presented here were performed on a Sparc 1+ computer using floating-point calculations[1] and explicitly computing the derivatives indicated. This has the unfortunate effect of amplifying noise in the original image, but enables the simulations to be run rapidly. To obtain more accurate results and to avoid noise

1. The method does not rely on floating-point accuracy: in an earlier paper we demonstrated that useful results could be obtained even with an 8 bit integer real-time machine [7,16].

amplification, filters that respond directly to Gaussian profile derivatives should be placed at each spatial location. We suggested [16] that such an inherently parallel implementation may be used by the visual system, in analogy with the velocity-tuned cells of MT [14].

There are other consequences of approximating the derivative as a difference equation. We estimate the velocity by the first derivative of the activation function, but the derivative of the Gaussian is a curve (not a line) that reaches its maxima at one standard deviation (σ, τ) from the center. The Gaussian Derivative is almost linear until near the maxima, at which point it becomes slower than linear as it begins to bend over. Differences taken when the velocity is small and the samples fall in the linear regime approximate the derivative well. When one approximates the derivative by sampling the curve near the maxima, the velocity is slightly underestimated. When the phase shift is greater then the scale of the filter (i.e., when the motion extends beyond the maximum of the Gaussian Derivative), the velocity can be significantly underestimated.

For each of the following plots, individual data points are marked with either an x, an o, an *, or a +, and the data points that were constructed with the same parameters are tied together with lines. A line with no data point on it indicates the ideal response. Missing data points occur when no velocity estimate could be made (because the point was on a local plane of activity or on a saddle of activity). When Gaussian curvature κ equals zero and mean curvature H equals zero, velocities on the plane will be "filled-in" from velocities on the boundary of the plane.

A. Points in Motion

In this section we show that the point speed estimates are extremely accurate for small constant velocities, and slightly underestimate large constant velocities. We also provide an example which indicates when the characteristic filter criteria will not select the correct estimate across different values of τ, and show how the choice of mask size affects velocity measurements. The response of three different filters is depicted in Figure 1, where the left graph provides velocity estimates obtained with a smaller mask size than the right graph. The two graphs differ only where the filter is being used to measure speeds that exceed its characteristic speed (σ/τ). While the larger mask sizes provide better speed estimates, the truncated masks also provide good results. Finally, the characteristic filter criteria suggested above would incorrectly select the speed measured by the filter with $\sigma = 2, \tau = 2$ for the modeled speed of 4 pixels per frame, while the best estimate is provided by the filter with $\sigma = 4, \tau = 1$.

Interactions between a point and a nearby feature can cause inaccurate velocity estimates, and is related to confusion on the scale of the filter between the correspondence of the two features. When such confusion occurs, competition between the

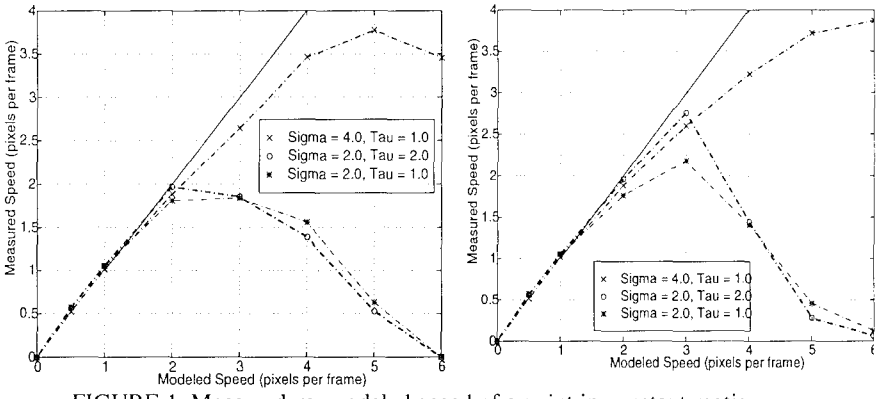


FIGURE 1. Measured vs. modeled speed of a point in constant motion. Left is with masks that extend to N=2; Right masks extend to N=3.

scales as described in *(7)* will guide the selection of the true velocity. Low speed features can be correctly measured by small scale filters which effectively separate the features from their neighbors. High speed features moving together can be correctly measured by large scale filters which effectively group multiple features together as a texture. These filters then measure the phase shift of the convected complex profile as described above. The velocity estimation results for two points moving as a unit with each separated from the other by a small amount is depicted in Figure 2. The left side indicates the measured speed of the pair of points; these should be compared with the $\sigma = 4, \tau = 1$ graph on the left of Figure 1. For all separations, the measurable error is small out to the speed of 3 pixels per frame. At this point, the activity waves generated by the points begin to interact, causing confusion between the two waves. The speed reported here is for the leading of a pair of points, and the underestimate is due to phase confusion of the leading point with its trailing neighbor. A different scale filter will correct this problem; a filter with $\sigma = 5$ accurately (less than .06 pixels/frame perturbation) measures speed out to 4 pixels per frame at all the indicated separations.

B. Lines in Motion

Isolated line velocity is very accurately estimated by the method of *Convected Activation Profiles;* the graphs are nearly identical to those depicted for point velocity. As explained above, the velocity of short lines (where the line is small when compared to the filter size) is not accurately estimated by Equations *(4a)* and *(4b)*, but is well estimated by Equations *(3)*. By using the Gaussian/mean curvature of the activation profile to decide between the estimation techniques, the velocity estimate is significantly improved. Figure 3 demonstrates this fact for a line of varying sizes moving through a filter with $\sigma = 4, \tau = 1$ along its normal at the rate of two pixels per frame. The original decision mechanism based on spatial proximity is quite

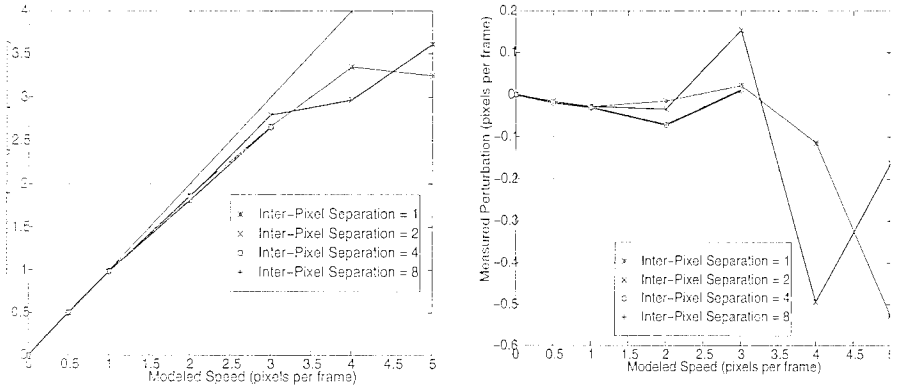


FIGURE 2. Left: Measured vs. modeled speed of a pair of points in constant motion. Right: Difference between measured speed of a pair of points, and measured speed of a single point.

good for the isolated point and extended line, but severely underestimates velocity for intermediate length lines. The new technique is significantly better.

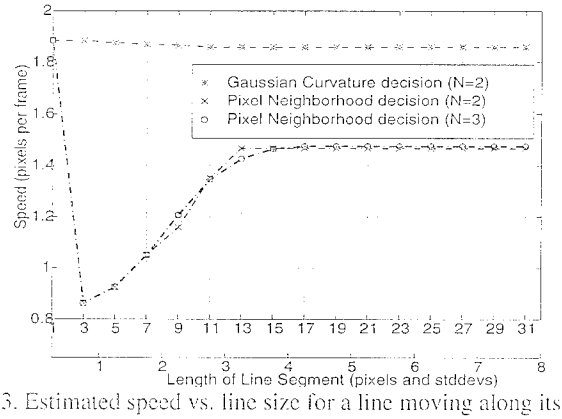


FIGURE 3. Estimated speed vs. line size for a line moving along its normal at 2 pixels/frame. The different graphs show the rate as described by the original pixel neighborhood decision, and the new Gaussian curvature decision. Two mask sizes are presented to demonstrate the effects caused by mask truncation.

Although originally formulated for translating lines, the method also works well for rotating lines. Figure 4 shows the results for half of a 100 pixel line rotating in the image plane. Notice that the speed estimates degrade smoothly as the normal velocity increases along the line. The final bowing of the estimates is due to the line end sweeping through only a portion of the filter.

C. Curves in Motion

The velocity of moving curves are also accurately measured by Convected Activation Profiles. In Figure 5 , the velocity estimates are given for half circles of different radii moving at a rate of 0 to 6 pixels per frame. The estimates are nearly

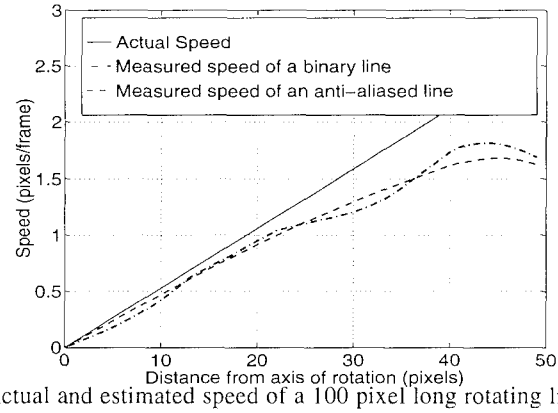


FIGURE 4. Actual and estimated speed of a 100 pixel long rotating line. Some small errors due to quantization of the line are removed when tracking an anti-aliased line. The estimates degrade smoothly as the normal velocity increases along the line. The final bowing of the curve is due to the line end.

perfect, only deviating significantly from estimates for a translating straight line (Figure 1 , $\sigma = 4, \tau = 1$) when the half circles move at a rate beyond the characteristic speed of the filter. In this case, a larger scale filter gives results comparable to those of a translating line.

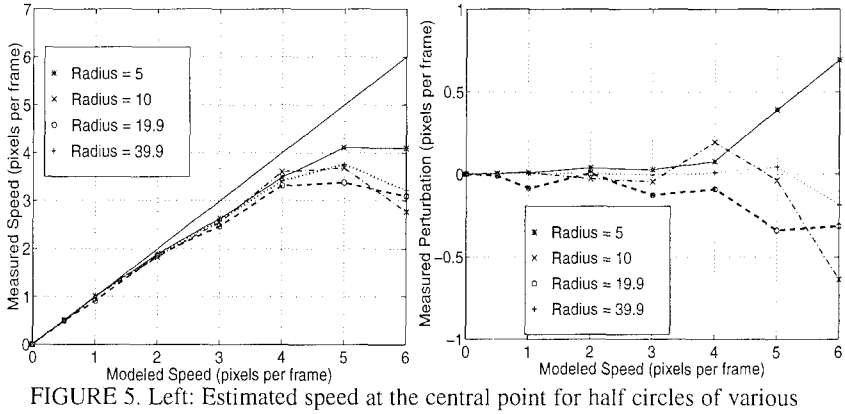


FIGURE 5. Left: Estimated speed at the central point for half circles of various radii moving towards their open end. Right: Difference between these results and those obtained for a translating line.

IV. CONCLUSIONS

In this paper we have presented additional analysis and simulations of image feature velocity extraction by the method of Convected Activation Profiles, and implemented an improvement to the algorithm for the specific case of objects moving parallel to the image plane where the features are short lines or where the features are close enough to interact. The central concept of the improvements is that inter-

actions between a line and neighboring features produce a complex activity profile that is convected and has a phase velocity that can be measured, just as in the simple cases of isolated points and lines. The characteristic speed normalization [16] used to select between different scale filters is shown to apply only when the spatial scale parameter σ is varied but not when the temporal scale parameter τ is varied. Finally, Convected Activation Profiles is demonstrated to be robust even when multiple features interact and for the condition of a line in rotation.

V. REFERENCES

[1] Adelson, E. H. and Bergen, J. R., "Spatiotemporal energy models for the perception of motion." *Journal of the Optical Society of America A*, 2, pp. 284-299, 1985.

[2] Anandan, P., "A computational framework and an algorithm for the measurement of visual motion," *International Journal of Computer Vision 2*, pp. 283-310, 1989.

[3] Barlow, H. B. and Levick, W. R., "The mechanism of directionally selective units in the rabbit retina," *Journal of Physiology*, pp. 477-504, 1965.

[4] Braddick, "A short-range process in apparent motion," *Vis. Res. 14*, pp. 519-527, 1974.

[5] Breightmeyer, B. G., "A relationship between the detection of size, rate, orientation and direction in the human visual system," *Vision Research 13*, pp. 41-58, 1973.

[6] Cunningham, R. K. and Waxman, A. M.,. "Diffusion-Enhancement Bilayer: Realizing long-range apparent motion and spatiotemporal grouping in a neural architecture," *Neural Networks*, pp. 895-924, 1994.

[7] Fay, D. A. and Waxman, A. M., "Neurodynamics of Real-time Image Velocity Extraction," **Neural Networks for Vision and Image Processing,** Eds.: G. Carpenter and S. Grossberg, 1992.

[8] Fleet, D. J. and Jepson, A. D., "Computation of Component Image Velocity from Local Phase Information," *International Journal of Computer Vision 5:1*, pp. 77-104, 1990.

[9] Grossberg, S. and Rudd, M. E., "A neural architecture for visual motion perception: Group and element apparent motion," *Neural Networks 2, pp. 421-450,* 1989.

[10] Horn, B. K. P. and Schunck, B. G., "Determining optical flow," *Artificial Intelligence,* pp185-203, 1981.

[11] Marr, D. and Ullman, S. "Directional selectivity and its use in early visual processing," *Proceedings of the Royal Society of London B 211*, pp. 151-180, 1981.

[12] Reichart, W., "Autocorrelation, a principle for the evaluation of sensory information by the central nervous system," **Sensory Communication**, Wiley Press, New York, 1961.

[13] Maunsell J. H. R. and Van Essen, D. C., "Functional Properties of Neurons in Middle Temporal Visual Area of the Macaque Monkey. I. Selectivity for Stimulus Direction, Speed, and Orientation," *Journal of Neurophysiology 49,* 1127-1147, 1983.

[14] Ullman, S., **The Interpretation of Visual Motion**, The MIT Press, Cambridge, MA, 1979.

[15] Waxman, A. M. and Wohn, K., Image Flow Theory: A framework for 3D inference from time-varying imagery," **Advances in computer vision**, Earlbaum, Hillsdale, NJ, 1988.

[16] Waxman, A. M. and Wu, J. and Bergholm, F., "Convected activation profiles and the measurement of visual motion," Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 717-723, 1988.

Pruning Projection Pursuit Models for Improved Cloud Detection in AVIRIS Imagery

Charles M. Bachmann *, Eugene E. Clothiaux †, John W. Moore ‡, and Dong Q. Luong ‡

* Airborne Radar Branch
Radar Division,Code 5365
Naval Research Laboratory,
Washington, D. C. 20375
e-mail:
bachmann@radar.nrl.navy.mil

† Department of Meteorology
Pennsylvania State University
University Park,
Pennsylvania 16802
e-mail: cloth@essc.psu.edu

‡ Lockheed Martin
2711 Jefferson Davis Hwy.
Arlington, VA 22202.

Abstract - A Projection Pursuit (PP) method is used to find structure and reduce the complexity of high-dimensional remote sensing data. Individual Projection Pursuit networks extract features from Gray-Level Difference Vector distributions, Sum and Difference Histograms, or simple normalizations of raw pixel intensity from one of four spectral bands used in the study. A PP pruning technique, based on an online perturbation analysis similar to that of (LeCun, Denker, and Solla, 1990), is used to remove parameters of low significance. The four AVIRIS spectral channels studied here were chosen because of their similarity to those which will be available from the Multi-Angle Imaging Spectro-Radiometer, an instrument which will be on EOS satellites. Ensemble models, which combine features extracted from AVIRIS imagery by multiple Projection Pursuit networks, use backward error propagation with a cross-entropy objective function to obtain pixel classifications. Predicted cloud masks are compared against human interpretation masks.

1 Introduction and Approach

In this paper, we investigate an approach to cloud detection in remote sensing imagery using Projection Pursuit techniques. Our goal is to develop textural feature extraction and cloud detection methods, which would be applicable to the imagery that will be produced by the Multi-Angle Imaging Spectro-Radiometer (MISR). MISR will be one of the instruments on the Earth Observing System (EOS) satellites and will obtain data in four spectral channels in the visible and near infra-red spectrum. The first phase of our investigation, described here, examines pixel-level detection of clouds in Airborne Visible/Infra-Red Imaging Spectrometer (AVIRIS)[19] imagery. While AVIRIS has extensive spectral coverage (0.4-2.4μm) of the visible and part of the infra-red spectrum in 224 channels, our purpose in choosing just 4 channels from AVIRIS was motivated by our desire to obtain channels similar to those of MISR. The second stage of our investigation, to be described in a future paper, will be an analysis of a large corpus of 1km Advanced Very High-Resolution Radiometer (AVHRR) data. Though the AVHRR visible channels are spectrally broad compared to those planned for MISR, we have opted for this data set because of its global coverage.

Textural feature extraction has been considered for GOES (Geostationary Observing Earth Satellite) and AVHRR (Advanced Very High-Resolution Radiometer) [9], [13] [5], [6], [20], [17], [8]. Typical examples of these features have been moments of gray-level difference vector (GLDV) statistics

0-7803-2739-X/95 $4.00 © 1995 IEEE

[21], [4], sum and difference histograms (SADH) [18], [4] and gray-level run length (GLRL) [21]. Neural networks have been used to find relationships among these features that are useful for automatic classification [20], [17]: classification results compare favorably to traditional statistical analysis [20]. What distinguishes our approach to textural feature extraction, in particular for the GLDV and SADH representation, is that we use Projection Pursuit to extract features from the histograms, treating them as input vectors in a high-dimensional space, rather than calculating pre-specified moments of the histograms.

In this paper we extend the results of our previous research on this data set [2], in which only simple normalizations of the raw data were used for pre-processing. We use unsupervised BCM [3], [12], [11], [1] [2] Projection Pursuit networks to extract features from the raw intensity data in four spectral bands after applying simple preprocessing schemes: bands 5 ($\sim 0.440\mu m$), 17 ($\sim 0.558\mu m$), 28 ($\sim 0.666\mu m$), 52 ($\sim 0.863\mu m$). [1] Preprocessing was either GLDV, SADH [2], or simple normalization of raw pixel intensity by the scene dynamic range. Multiple views of data structure from each of the low-level BCM networks improves classification. Inputs to the individual BCM networks may be chosen from a variety of patch sizes: in this paper most experiments used 12x12 and 6x6 input patches.

At higher levels of our model architecture, we obtain pixel-level classifications by pooling and adaptively weighting projections from ensembles of BCM Projection Pursuit networks. To accomplish this, we use the supervised learning procedure backward propagation (BP) [16] with a cross-entropy (BPCE) Cost Function [15]. The identity of individual pixels, e.g. cloud vs. no-cloud, within a particular sub-block of the input patch is the desired output of the system: network hierarchies may be trained to predict the identities of individual pixels in any sub-block within the input patch. A schematic diagram of the approach appears in Fig. 1. Error signals for adaptation are generated by comparing the output of the network hierarchy with that of a human interpretation mask. Smoothing of multiple estimates of pixel identity can be accomplished by using BPCE to adaptively weight the output of several BCM-BPCE ensemble estimators (ensemble of ensembles) [2], although this is not done here.

In [2] and in this paper, the ground truth was taken to be human interpretation masks: these masks required detailed, local analysis and thresholding over numerous patches within the image frame. AVIRIS image size is 614x512 pixels per band with a resolution of 20m per pixel. Typical ground truth masks took anywhere from 10 minutes to an hour to generate for each scene, depending on the complexity. In the present study, there are 17 scenes: in this study 8 scenes were used for training, 2 for cross-validation during training

[1] Calibration of particular scenes in the set varies, and in some cases a neighboring spectral band was substituted.

[2] Several different directions and separations were used to compute GLDV and SADH histograms.

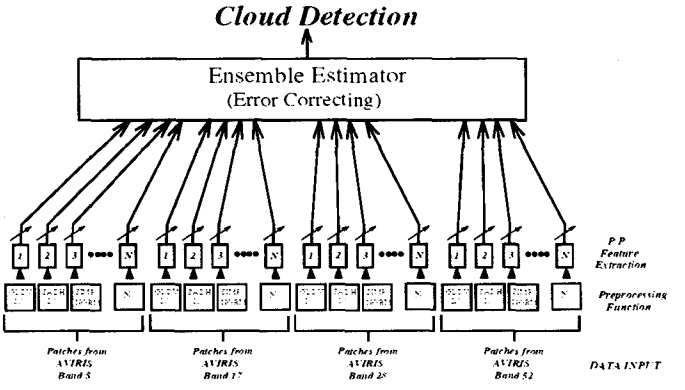


Figure 1: Schematic diagram of the cloud detection model. At the lowest level patches are input from one of four spectral bands used in the study. Several different preprocessing methods may be employed in parallel (GLDV and/or SADH in a variety of directions, as well as simple normalization of pixel intensities); these become the input to separate Projection Pursuit networks that extract features; these features are adaptively weighted by a 3-layer error correcting network (BPCE) that outputs pixel predictions (no cloud/cloud); error corrections are achieved by comparison with human interpretation masks.

and 7 scenes for assessing generalization to novel data.

2 Feature Extraction by Projection Pursuit

In general, Projection Pursuit methods seek to find structure and reduce complexity in high-dimensional data spaces by discovering a low-dimensional set of statistically interesting projections [7], [10]. Our particular approach discovers structure in the data by adaptively minimizing a Projection Index that favors statistically skew, bi-modal, or multi-modal projections of the data distribution. Projections are jointly optimized. Our approach is based on the BCM model [3], [12],[11], [1] [2] that has recently been linked [12] to Projection Pursuit. In the BCM Projection Pursuit model, the ith projection in layer n of a multi-layer BCM network is: [3]

$$\tilde{c}_i^{(n)} = \sigma(\sum_j L_{ij}^{(n)} c_j^{(n)}) \quad with: \quad \sigma(x) = a\,\tanh(a\lambda x), \tag{1}$$

$$c_j^{(n)} = \vec{w}_j^{(n-1)} \cdot \vec{\tilde{c}}^{(n-1)} + b_j^{(n)}, \quad and \quad L_{ij}^{(n)} = \begin{pmatrix} 1-\mu, for\ i=j \\ -\mu, for\ i \neq j \end{pmatrix}$$

where $L_{ij}^{(n)}$ is a fixed constraint matrix, $\vec{w}_j^{(n-1)}$ is the jth modifiable projection vector, which weights inputs from layer $(n-1)$, and $b_j^{(n)}$ is the bias.

[3] In the present study, BCM networks had only a single layer of projections.

Projection vectors and biases are modified by minimizing the cost of the projected data distribution over the cost function, $\xi^{(n)}$ (Eq. 2) : this leads to a semi-local Projection Index, $E[\xi^{(n)}]$ (Eq. 3) , with no reference to training labels (unsupervised learning): [4]

$$Cost\ Function: \ \xi_i^{(n)} = -\left(\frac{(\tilde{c}_i^{(n)})^3}{3} - \gamma\tilde{\theta}_i^{(n)}\ \frac{(\tilde{c}_i^{(n)})^2}{4}\right) : \ \theta_i^{(n)} = E[(c_i^{(n)})^2] \tag{2}$$

$$Minimize: \ E[\xi^{(n)}] = -\left(\sum_i \frac{E[(\tilde{c}_i^{(n)})^3]}{3} - \gamma\frac{E^2[(\tilde{c}_i^{(n)})^2]}{4}\right) \tag{3}$$

$$\begin{aligned} Using: \ \Delta\vec{w}_k^{(n-1)} &= -\eta\frac{\partial E(\xi^{(n)})}{\partial\vec{w}_k^{(n-1)}} \\ &= \eta E\left[\sum_i \phi(\tilde{c}_i^{(n)}, \tilde{\theta}_i^{(n)})\ (\tilde{c}_i^{(n)})'\ L_{ik}^{(n)}\ \vec{\tilde{c}}^{(n-1)}\right] \end{aligned} \tag{4}$$

$$\begin{aligned} with: \ \phi(\tilde{c}_i^{(n)}, \tilde{\theta}_i^{(n)}) &= \tilde{c}_i^{(n)}(\tilde{c}_i^{(n)} - \gamma\tilde{\theta}_i^{(n)}), \\ (\tilde{c}_i^{(n)})' &= \lambda(a - \tilde{c}_i^{(n)})(a + \tilde{c}_i^{(n)}) \end{aligned}$$

$\gamma\theta_i^{(n)}$ is the dynamic modification threshold above which the response to a

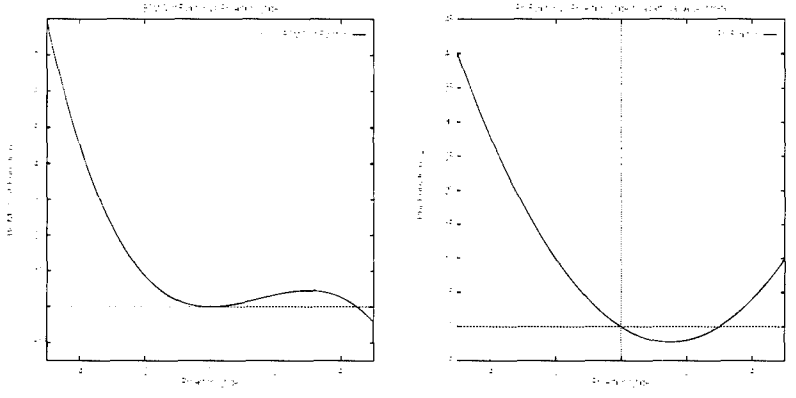


Figure 2: (Left) The BCM Cost Function vs. $\tilde{c}_i^{(n)}$, the Cost Function favors at least bi-modal projections of the data; the two wells move during training, and the time scale is set by τ in Equation 6 ; the Projection Index is obtained by minimizing the cost of the data projections over this function, and then averaging over all projections. (Right) the ϕ-function vs. $\tilde{c}_i^{(n)}$ in a single projection model. ϕ determines whether the projection response to a pattern is reinforced or weakened based on whether the pattern is above or below the dynamical threshold (the rightmost crossing point), $\gamma\theta_i^{(n)} = \gamma E[(c_i^{(n)})^2]$.

[4] $E_{..}^{--}$ represents the expectation value.

particular input pattern is reinforced and below which it is weakened in a single projection model; this can be seen by writing:

$$\frac{d\tilde{c}^{(n)}}{dt} = \nabla_{\vec{w}} \tilde{c}^{(n)} \cdot \frac{d\vec{w}}{dt} \approx \eta \left((\tilde{c}^{(n)})' \, \vec{\tilde{c}}^{(n-1)} \right)^2 \phi \tag{5}$$

Thus, the sign of ϕ, depicted graphically in Fig. 2, determines the sign of the change in the response to a particular pattern. Typically the expectation value in Eq. 2 is only approximated as a leaky integrator:

$$\theta_i^{(n)} = \frac{1}{\tau} \int_{t_0}^{t} e^{-\frac{t-s}{\tau}} (\tilde{c}_i^{(n)})^2(s) \, ds \tag{6}$$

This minimization procedure biases projection vectors toward directions where the input pattern distribution is statistically skew or multi-modal when projected onto them. For a small and decreasing step-size, Equation 4 is well approximated by stochastic gradient descent [12].

If we examine pairs of these projections in a Projection Pursuit network, we find that the *projected distribution* of high-dimensional inputs contains a rich underlying structure that is useful for cloud detection; two-dimensional histograms of projections of AVIRIS band 5 data are shown in Fig. 3.

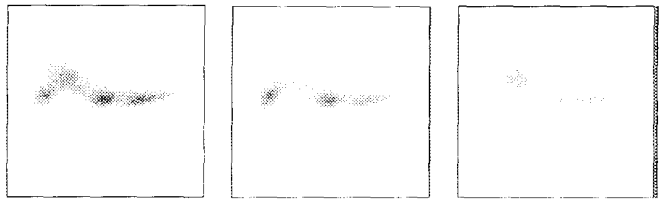


Figure 3: Two-dimensional histograms of projected AVIRIS data from Band 5 ($0.420\mu m$) input patches (25x25pixels); 60000 samples drawn from eight training scenes. Inputs were vectors whose elements were the magnitudes of GLDV histogram elements in the (2,0) direction. Each axis (not shown) is the response of a particular projection from a PP network. Each row of three histograms corresponds to a particular pair of projections: (Left column) projected distribution of all 60000 samples; (middle column) projected distribution of samples that are completely cloud free; (right column) projected distribution of samples containing 50% or more cloud cover.

3 Pruning BCM Projection Models

A number of researchers have developed methods for adapting the size of the model structure in non-parametric estimators. In [14] a method is described for pruning vector elements in supervised adaptive classifiers. Their approach

is a second order perturbation analysis about a local minimum of the least-mean-squares optimization procedure in a BP network. We apply a similar analysis to unsupervised BCM Projection Pursuit networks. The incentive for pruning is to determine an overall detection model of sufficient complexity to describe the available data but without overfitting. Models that have too many parameters will tend to overfit the training data and will generalize poorly to novel examples drawn from the same distribution.

For unsupervised BCM Projection Pursuit, the removal of a parameter from the model changes the Projection Index according to:

$$\Delta E(\xi) = E(\xi_{pruned}) - E(\xi) = E(\xi_{w_{ij}=0}) - E(\xi) \tag{7}$$

If the weights are small, we assume that the impact of removing the weight can be calculated by a second order Taylor Expansion:

$$\Delta E(\xi) = -w_{ij} E(\frac{\partial \xi}{\partial w_{ij}}) - \frac{w_{ij}^2}{2} E(\frac{\partial^2 \xi}{\partial w_{ij}^2}) \tag{8}$$

We also have made the approximation that only the diagonal terms in the Hessian are important. For the BCM model, the second order gradient is: [5]

$$\frac{\partial^2 E(\xi)}{\partial w_{ij}^2} = -2E(\sum_k L_{ki}^2 f_j^2 c_k' [c_k' (c_k - \theta(c_K E(c_k) + \frac{E(c_k^2)}{2})) - \lambda \alpha_k c_k]) \tag{9}$$

We observe that the gradient descent relationship is:

$$\frac{dE(\xi)}{dt} = \nabla_{\vec{w}_i} \xi \cdot \frac{d\vec{w}_i}{dt} \approx -\eta(t) \mid \nabla_{\vec{w}_i} E(\xi) \mid^2 \tag{10}$$

and, therefore, set the following criterion for pruning:

$$\mid \nabla_{\vec{w}_i} E(\xi) \mid^2 < \alpha_{initiate\ pruning\ cutoff} \tag{11}$$

Choosing a larger cutoff $\alpha_{initiate\ pruning\ cutoff}$ will lead to pruning earlier; this is desirable if we wish to narrow the search to a subspace earlier in training before encountering the local minimum, thereby lowering the computational burden.

4 Results and Conclusions

AVIRIS scenes used in the experiments are summarized in Table 1.

Table 1
AVIRIS Scenes. Inputs were from Bands 5, 17, 28, and 32

Training Scene	Cross Validation Scene	Novel Test Scene
050285A.R6.82	910628R.R5.84	920703R.R2.85
910920A.R2.82	960815R.R12.82	920925C.R5.82
900810A.R6.84		920624C.R3.82
900805A.R3.85		920602R.R17.85
900722A.R9.84		910720R.R6.86
960815A.R9.83		910709R.R7.82
970812A.R6.83		970509R.R7.81
900813A.R5.82		

[5] Here we assume a single search layer of projections, dropping the layer superscript and denoting projection outputs by c_j and projection inputs by f_i.

We examined a variety of ensemble configurations. The size of the input patch was varied: 12x12 or 6x6. The number of intermediate processing nodes in the BPCE component of the ensemble model was also varied. Ensembles were created in which Projection Pursuit networks were either pruned or unpruned. We were quite aggressive in the experiments with pruning and found that we could obtain performance from pruned networks comparable to that of their unpruned counterparts with significantly fewer parameters for ensembles with the same initial size. Generalization results on the novel test data are summarized in Fig. 4 for a set of experiments in which the preprocessing was GLDV or simple normalization of the intensity data: ensembles with and without pruning are shown on the same curve. Fig. 4 shows the expected

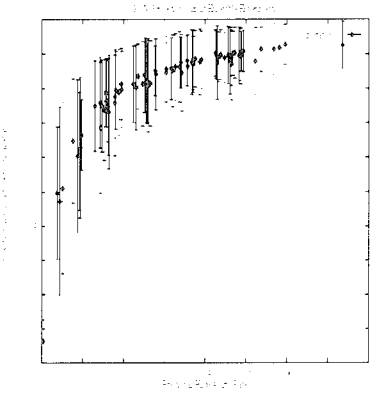


Figure 4: Generalization Results: fractional rate of cloud detection vs. fractional false alarm rate for the AVIRIS scenes in the novel test set for a variety of Projection Pursuit experiments. Error bars for rate of detection are artificially high because the number of scenes in the test set was small. Ensembles containing pruned BCM networks appear on the same curve with those in which no pruning occurred. The curve contains generalization test results evaluated at several points during the training cycle for each ensemble (starting points are omitted). One of the best results was obtained by an ensemble in which lower level networks received inputs from two different input patch sizes surrounding each 4x4 output prediction box, a cloud detection rate of $90.3 \pm 6.7\%$ with a false alarm rate of $4.3 \pm 4.6\%$.

tradeoff between rate of cloud detection and false alarm rate. One of the better results was a cloud detection rate of $90.3 \pm 6.7\%$ with a false alarm rate of $4.3 \pm 4.6\%$. Fig. 5 shows the results of one of the experiments, comparing the PP ensemble prediction masks with those of a human interpreter, and the original AVIRIS scene (only one of the four spectral bands used as input is shown): difference masks depict false negatives and positives. The results on the novel test data are generally good, particularly given the very limited size of the training set.

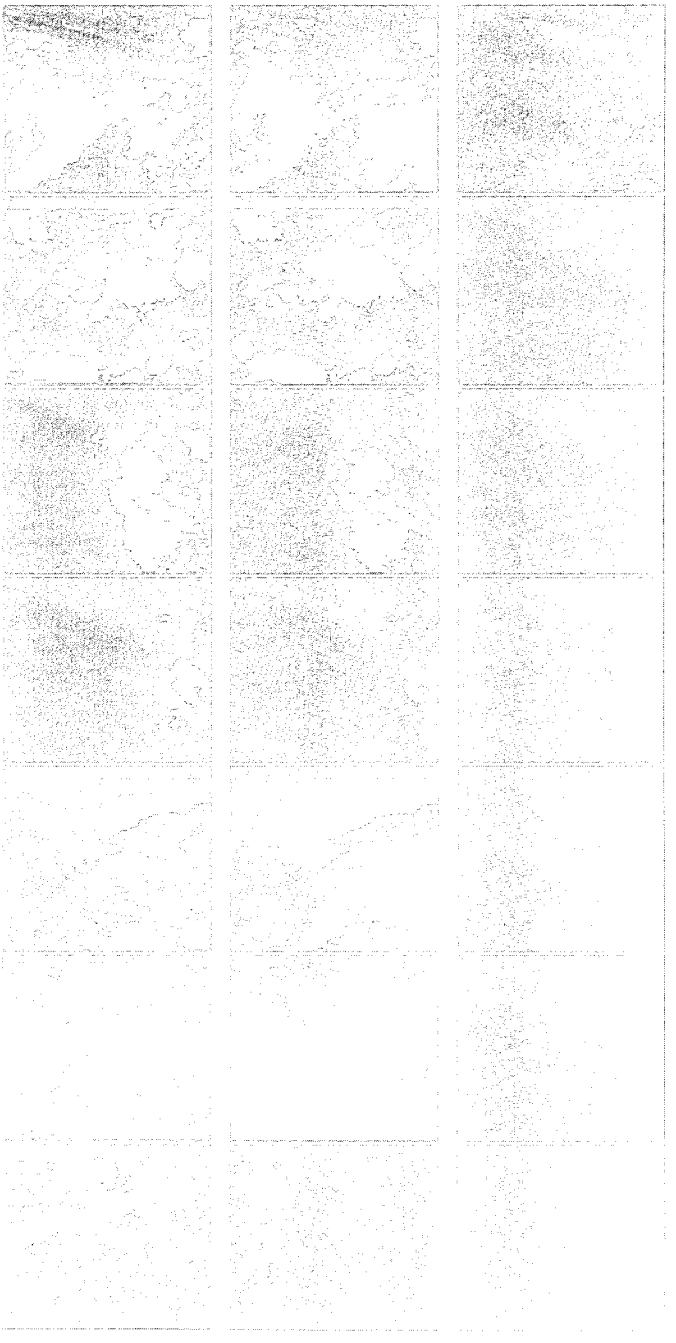


Figure 5: An example of cloud detection for novel test data. (Column 1) AVIRIS images (Band 32) from the novel test set, colormap is white (low) to black (high); (Column 2) detection of clouds by a Projection Pursuit Ensemble, white = cloud detection;(Column 3) human interpretation of location of clouds, white = cloud location ; (Column 4) Difference Masks: black = no error, white = false negative, grey = false alarm. Most errors occur on the boundaries of clouds. Inputs were from 4 spectral bands: 5, 17, 28, and 32. Prediction and difference masks have a small 4 pixel border in which no predictions were made due to the particular architecture of the experiment; this is not a permanent limitation of the approach.

Acknowledgment- This work was supported in part by a grant of HPC time from the DOD HPC Center, Maui High Performance Computing Center. C. Bachmann is supported by grants from ARPA (Dr. Barbara Yoon) through Program Element 62712E, ONR (Dr. Thomas McKenna) through grant # N0001495WX30304, and additional ONR support through grant # N0001495WX40002. J. Moore and D. Luong contributed to this work under contract to NRL.

References

[1] C. M. Bachmann, S. Musman, D. Luong, and A. Schultz, "Unsupervised BCM Projection Pursuit Algorithms for Classification of Simulated Radar Presentations," *Neural Networks*, Vol. 7, No. 4, pp. 709-728, 1994.

[2] C. M. Bachmann, E. E. Clothiaux, J. W. Moore, K. J. Andreano, "An Ensemble Approach to Automatic Cloud Identification in AVIRIS Imagery Using BCM Projection Pursuit," in Neural Networks for Signal Processing IV: Proceedings of the 1994 IEEE Workshop, Sept 6-8, 1994, Ermioni, Greece, pp. 394-403.

[3] E. L. Bienenstock, L. N. Cooper, P. W. Munro, "Theory for the Development of Neuron Selectivity: Orientation Specificity and Binocular Interaction in Visual Cortex," *J. Neuroscience*, Vol. 2, No. 1, pp. 32-48, 1982.

[4] D. W. Chen, S.K. Sengupta, and R.M. Welch, "Cloud Field Classification Based upon High Spatial Resolution Textural Features: 2. Simplified Vector Approaches," *J. Geophys. Res.*, 94, No. D12, 14749-14765, 1989.

[5] E. Ebert, "A Pattern Recognition Technique for Distinguishing Surface and Cloud Types in the Polar Regions," *J. Climate Appl. Meteor.*, 26, 1412-1427, 1987.

[6] E. Ebert, "Analysis of Polar Clouds from Satellite Imagery Using Pattern Recognition and a Statistical Cloud Analysis Scheme," *J. Appl. Meteorology*, Vol. 28, pp. 382 - 399, 1989.

[7] J. H. Friedman, J. W. Tukey, "A Projection Pursuit Algorithm for Exploratory Data Analysis," *IEEE Trans. Computers*, Vol. c-23, No. 9, pp. 881 - 890, 1974.

[8] S. C. Gallegos, J. D. Hawkins, "A New Automated Method of Cloud Masking for Advanced Very High Resolution Radiometer Full-Resolution Data Over the Ocean," *Journal of Geophysical Research*, Vol. 98, No. C5, 8505-8516, May 15, 1993, in AVHRR imagery)

[9] L. Garand, "Automated Recognition of Oceanic Cloud Patterns, Part I: Methodology and Application to Cloud Climatology," J. Climate, 1, 20-39, 1988.

[10] P. J. Huber, "Projection Pursuit," *Annals of Statistics*, Vol. 13, No.2, 435-475, 1985.

[11] N. Intrator, "Feature Extraction Using an Unsupervised Neural Network," in Proceedings of the 1990 Connectionist Models Summer School, (eds.) D. S. Touretzky, G. L. Ellman, T. J. Sejnowski, Morgan Kaufmann, San Mateo, CA.

[12] N. Intrator, L. N. Cooper, "Objective Function Formulation of the BCM Theory of Visual Cortical Plasticity: Statistical Connections, Stability Conditions," *Neural Networks*, Vol. 5, pp. 3-17, 1992.

[13] J. Key, "Cloud Cover Analysis with Arctic Advanced Very High Resolution Radiometer Data, 2: Classification with Spectral and Textural Measures," *J. Geophys. Res.*, 95, No. D6, 7661-7675, 1990.

[14] Y. LeCun, J. Denker, S. Solla, "Optimal Brain Damage," in Advances in Neural Information Processing Systems, ed. D. Touretzky, Vol. 2, pp. 598-605, Morgan Kaufmann, San Mateo, CA, 1990.

[15] M. D. Richard, R. P. Lippman, "Neural Network Classifiers Estimate Bayesian a posteriori Probabilities," *Neural Computation*, 3, 461-483, 1991.

[16] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning Internal Representations by Error Propagation," in *Parallel Distributed Processing. Explorations in the Microstructure of Cognition*, Vol. 1, Rumelhart, D. E., McClelland, J. L. (eds.), MIT Press, Cambridge, MA, pp. 318-362, 1986.

[17] V. R. Tovinkere, M. Penaloza, A. Logar, "An Intercomparison of Artificial Intelligence Approaches for Polar Scene Identification," *J. Geophys. Res.*, Vol. 98, No. D3, pp. 5001-5016, March, 20, 1993.

[18] M. Unser, "Sum and Difference Histograms for Texture classification," *IEEE Trans. PAMI-8*, No.1, 118-125, 1986.

[19] G. Vane, R. O. Green, T. G. Chrien, H. T. Enmark, E. G. Hansen, W. M. Porter, *The Airborne Visible/Infrared Imaging Spectrometer (AVIRIS)*, Remote Sens. Environ., 44: 127-143 (1993).

[20] R. M. Welch, S. K. Sengupta, A. K. Goroch, P. Rabindra, N. Rangaraj, M. S. Navar, "Polar Cloud and Surface Classification Using AVHRR Imagery: An Intercomparison of Methods," *J. Appl. Meteorology*, Vol. 31, pp. 405-420, May, 1992.

[21] J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A comparative Study of Texture Measures for Terrain Classification," *IEEE Trans. Syst., Man, Cybern.*, SMC-6, 269-285, 1976.

A NEW LEARNING SCHEME FOR THE RECOGNITION OF DYNAMICAL HANDWRITTEN CHARACTERS

Fidimahery ANDRIANASY and Maurice MILGRAM

Université Pierre et Marie Curie, Laboratoire PARC,
4 Place Jussieu , T66, 2ème étage, 75 005 Paris, France.
email: milgram@robo.jussieu.fr

Abstract. Vector comparison is essential in pattern recognition. Numerous methods based on distance computation are available to carry out such comparison. Unfortunately, most of them are applicable only if the vectors are of the same length or do not take into account components misalignment. This paper presents a new distance between two representations called the Elastic Distance and based on the dynamic programming technique. Properties are studied. We show that it leads to a variant of the Léast Vector Quatisation technique that learns the best representants of a group of prototypes. A new centroïd computation algorithm is proposed. Finally, the learning scheme algorithm has been successfully applied on an on-line numerical handwritten character recognition problem using a previously computed centroïd of a set of prototypes.

Keywords – Handwritten character recognition, clustering, elastic matching.

I INTRODUCTION

Vector comparison is essential in pattern recognition. Classical distances deriving from quadratic forms (such as the euclidean distance) are suitable tools to perform comparisons. Unfortunately they have a major drawbacks since not only vectors must belong to the same vector space, (i.e they must have the same length) but corresponding components (same rank) must have the same meaning too. These distances can not cope with a misalignements. Let be given two vectors $\mathbf{x} = (x_1, x_2, ..., x_n)$ and $\mathbf{y} = (y_1, y_2, ..., y_m)$ that are members of vector spaces with m not always equal to n. To simplify, we will consider the i-th component of x as the sample at time i and vectors will be treated like signals. If such a signal is shifted by just one unit of time and compared to the original non-shifted one, it will appear to be very different with the euclidean distance. We face here the well known Temporal Alignment Problem (TAP) that occurs as soon as two signals have to be compared.

This problem has been encountered for a while in the area of speech recognition and handwritten drawing recognition dealing with data provided by a tablet digitizer (scripts, sketches and musical score). Three main classes of solutions are known at the present time to tackle the temporal alignment problem. These are the Hidden Markovian Models,

0-7803-2739-X/95 $4.00 © 1995 IEEE

the adaptive approaches stemed from Adaptive Filtering methods and the techniques using Dynamic Programming.

All of the techniques mentionned above have been proven to be very efficient in speech and locutor recognition. Occasionally we could find them combined with the Hidden Markov Models or the Dynamic Programming technique [2, 3]. The Hidden Markov Models require a very time-consuming learning phase and large memory space for the training. Moreover, data must completly fill in the space of possibles. Numerous inspired works exhibit the great efficiency of these models [4, 5, 6]. Time Delay Neural Networks (TDNN) technique is another very interesting approach that circumvent the temporal alignement problem without solving it straightfully. Adaptive approaches [1] such as Predictive Networks or the Vector Prediction technique are also valid methods that realize a kind of automatic locking to deal with the TAP. Among the various methods based on Dynamic Progamming we can quote the Wagner & Fisher algorithm [3] for computing the editing distance between chains and the Dynamic Time Warping (DTW) algorithm wich is applied extensively in speech recognition. The Elastic Distance (ED) wich can be viewed as a DTW variant will be introduced in the next section.

II The Elastic Distance

Let $\mathbf{x} = (x_1, ..., x_i, ..., x_r)$ and $\mathbf{y} = (y_1, ...y_j, ..., y_s)$ be two signals where the lengths r and s are not always equal. Assume the components x_i and y_j to be vectors, with $x_i \in \mathbb{R}^p$ and $y_j \in \mathbb{R}^p$. Here p corresponds to the number of parameters required by a shape representation. We need two functions $i = U(n)$ and $j = V(n)$ in order to control the temporal flow of components of $\mathbf{x}$ and $\mathbf{y}$ during the matching process. One can note that i (resp. j) is the index of the component of $\mathbf{x}$, (x_i), to be taken into account a time n. The underlaying idea of the elastic distance calculation is to find a pair of functions (U, V) such that the distance between $x_i = x_{U(n)}$ and $y_j = y_{V(n)}$ is minimal for all values of index n. The sequence of pairs $(i_n, j_n) = (U(n), V(n))$ is called the *matching path*.

First of all, time must be allowed to flows differently for $\mathbf{x}$ and $\mathbf{y}$, while taking into account the whole components of both of the signals. In addition, it seems to be reasonnable to prohibit the path from stepping back or staying on the same point during the matching. These lead to the following conditions:

$$\forall\, n \in \mathbb{N}, \quad \begin{cases} U(n+1) - U(n) = 0 \text{ or } 1, & \text{and} \\ V(n+1) - V(n) = 0 \text{ or } 1, & \text{and} \\ U(n+1) - U(n) + V(n+1) - V(n) > 0 \ . \end{cases} \tag{1}$$

One could note that, it would be perfectly valid to allow skiping (as in DTW algorithm), i.e. $U(n+1) - U(n) > 1$. But so, we would have to penalize this type of path and this would give rise to the critical problem

of penalities tuning. We must also take into account the fact that U and V have different boundaries because $\mathbf{x}$ and $\mathbf{y}$ exhibit different lengths as signals (i.e. they do not have the same dimension as vectors). Let l_x (resp. l_y) be the length of $\mathbf{x}$ (resp. $\mathbf{y}$). We impose a second condition :

$$\exists\, n \;\setminus\; U(n) = l_x \text{ and } V(n) = l_y \;. \tag{2}$$

The integer n is the final time of the matching process (n is obviously unique). To enforce the fact that n depends on U and V, it will be noted n_{UV}. Among all the possible matchings that verify conditions (1) and (2), we have to find the optimal one according to a criterium based on the distances between components matched by the pair (U, V). This includes all the distances between $x_{U(n)}$ and $y_{V(n)}$ for n varying from 0 to n_{UV}. The criterium could be chosen as :

$$\mathcal{J}(U, V) = \sum_{n=0}^{n_{UV}} d(x_{U(n)},\, y_{V(n)}) \;. \tag{3}$$

for a fixed (U, V) and where $d(.)$ is a distance over the components. This *instantaneous* distance could simply chosen to be the euclidean distance.

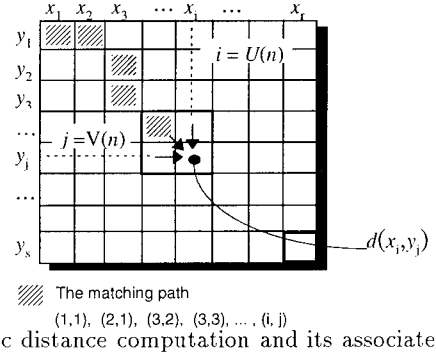


Fig. 1. The elastic distance computation and its associated matching path.

III Centroids Computation

III.1 Motivations

When a large number of prototypes (reference vectors of a class S_i) is at hand, classifying is always hard (time-space consumming) even if it is more accurate to use all prototypes. That is the case with the K-Nearest Neighbors algorithm and especially with the Parzen kernel approach wich perform a non-parametric estimation of the density of probability of the vectors for each class, in order to perform a Bayesian classification. Given N the number of reference vectors, the computation cost is of an order of $\mathcal{O}(N)$. Such calculation can be monumental for a large number of prototypes. Fortunately, we can take benefit form the intrinsic redundancy of the data. Neural Network techniques afford a wide range of interesting tools. The Learning Vector Quantification algorithm (LVQ)

proposed by T. Kohonen [5] substitutes a whole set of prototypes for a few numbers ones wich coincid with the centers of the balls that cover optimally all the classes. Three conditions must be satisfied in order to find these centroïds :

1. Balls corresponding to two distingable classes must be disjoint. We want to avoid ambiguity.
2. Each class must be covered as much as possible by the balls. We do not want unclassifiable areas.
3. A minimum number of balls must be used to get a maximum speed up factor. Otherwise one could have taken each of the prototypes as centers of the balls (i.e. we would have the same number of balls and prototypes.)

These conditions are obviously hard to get simultaneously.

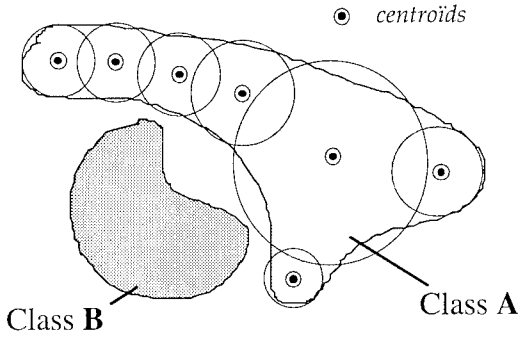


Fig. 2. Some centroïds.

III.2 Centroïds Definition

Let $\mathbf{x} = \{x^1, x^2, ..., x^N\}$ be a set of N points of $\mathbb{R}^n$. Each point x^i is a n dimensionnal real valued vector. We want to represent the entire set with a vector g called the *centroïd* of $\mathbf{x}$.

Definition 1. The vector g is a *p-centroïd* of $\mathbf{x}$ if g minimize the criterium

$$\mathcal{C}_p(g) = \sum_{i=1}^{N} d(g, x^i)^p \tag{4}$$

where $d(.)$ refers to any valid distance defined on $\mathbb{R}^n$.

If $d(.)$ is chosen to be the euclidean distance and $p = 2$, that is $d(x, y) = ||x - y||$ then g is the *2-centroïd* of $\mathbf{x}$, is unique and coincid with the center of gravity of the set $\mathbf{x}$. In the following, we will assume that $p = 2$. Parenthetically it should be pointed out that if the points x^i belong to some space $\mathbb{S}$ that does not have a vector-space structure (i.e. dot product

is not possible and euclidean distance is not authorized) then one could nevertheless define and find the 2-centroïd without guarantee of unicity.

The $\mathcal{C}_2(g)$ function is continuous and bounded. The lower bound is equal to zero. Then it admits local minimas. Let g_0 be a regular point and $a_0 = \mathcal{C}_2(g_0)$. The centroïd g must check in the relation $0 < \mathcal{C}_2(g) < a_0$ which means $g \in \mathcal{C}_2^{-1}([0, a_0])$. The set $g \in \mathcal{C}_2^{-1}([0, a_0])$ is generally a compact subset of $\mathbb{S}$ (a closed bounded on $\mathbb{R}^p$). Therefore, we are guaranteed to find at least one minima of $\mathcal{C}_2(g)$ on $\mathbb{S}$.

III.3 Centroïd Computation Algorithm

The centroïd g will be practically computed by the mean of a gradient descent based iterative algorithm. In the particular case of the elastic distance, we set on two hypothesis:

Hypothesis 1. *The shapes to be classified are represented by an identical number of components.*

Hypothesis 2. *The first derivative of the distance in respect of each of its components exists.*

Now, let g be the vector obtained at time t while minimizing the criterium $\mathcal{C}_2(g) = \sum_{i=1}^{N} d(g, x^i)^2$ where $g = (g_1, ..., g_j, ..., g_n)$ and $x^i = (x_1^i, x_2^i, ..., x_n^i)$ a prototype. Taking the partial derivative of $\mathcal{C}_2(g)$ with respect to g_j,

$$\frac{\partial \mathcal{C}}{\partial g_j} = 2 \sum_{i=1}^{N} d(g, x^i) \frac{\partial d(g, x^i)}{\partial g_j} \quad . \tag{5}$$

Let $K(j)$ be the set of indexes k met by the components g_j along the matching path during the computation of the elastic distance between g and the prototype x^i. Then

$$d(g, x^i) = \sum_{k=0}^{n} d(g_{U(k)}, x_{V(k)}), \tag{6}$$

and as seen before, $K(j) = \{V(k) \setminus U(k) = j\} = V \circ U^{-1}\{j\}$. Finally,

$$\frac{\partial d(g, x^i)}{\partial g_j} = \sum_{k \in K(j)} \frac{\partial d(g, x_k^i)}{\partial g_j} \quad . \tag{7}$$

Now with $(p = 2)$, it can be shown that

$$\frac{\partial \mathcal{C}}{\partial g_j} = 2 \sum_{i=1}^{N} d(g, x^i) \sum_{k \in K(j)} (g_j - x_k^i) \tag{8}$$

This last expression exhibits many similarities with the LVQ algorithm [5].

IV Centroïds Computation Algorithm

This section outlines the new centroïd computation algorithm using the elastic distance. The following notations are used within the pseudo-code: `NbProto` is the number of prototypes, `NbCentro` is the desired number of centroïds, `a` is the learning rate, where $0 < a < 1$, `e` is the stopping threshold, `ItMax` is the maximum iterations allowed. `Centroid[k][i]` is the i-th component of the centroïd number k, `DISTANCE(.)` is the Elastic Distance between two vectors and `Nuv` is the final matching time.

```
[0] Initalization
      Choose NbCentro
      Choose Centroid[k] for k = 1,...,NbCentro          /* among prototypes */
      Choose a, e, ItMax
      NumIt  0

[1] Centroids Learning
      Repeat
          [1-a]                              /* Distances computation */
          [1-b]                          /* Adaptation of the centroids  */
          Evaluate the Criterium;
      Until (Criterium < e) or (NumIt > ItMax).

[1-a] Distances computation
      NumIt := NumIt + 1; DMax := 0;
      For i = 1,...,NbProto,
          For k = 1,...,NbCentro,
              d[k] := DISTANCE(Prototype[i], Centroid[k]);
          Find k(i) such that d[k(i)] = MinK d[k];
          Set DMax  Max(DMax, d[k(i)]);

[1-b] Adaptation of the centroids
      For i = 1, ... ,NbProto,

          k := k(i);                    /* the closest Centroid for Proto[i] */
          d := DISTANCE(Prototype[i], Centroid[k]);

          For r = 1, ... ,Nuv
              (U(r), V(r)) := MATCHING PATH for Centro[k] & Proto[i]

          For r = 1, ... ,Nuv
              Centroid[k][U(r)] := (1 - a.(d/DMax)).Centroid[k][U(r)]
                                 + a.(d/DMax).Prototype[i][V(r)].
```

Fig. 3. Centroids Computation Algorithm based on the Elastic Distance.

A detailed implementation of the centroïd computation algorithm is given on Figure 3. This is actually an *on-line* variant of the true gradient descent algorithm (*batch method*) stated in the equation (8). The adaptation process could be seen as a gradual tweeking of the centroïd's components toward the components of the closest (in the sense of the elastic distance) prototype. This scheme is carried out at each iteration for the corresponding components provided by the matching path.

V Tests and Results

We propose to classify new references vectors by computing the elastic distance between those new references and the centroïd(s) of a given set of references.

V.1 The testbed

The patterns displayed on Figure 4 are chosen as testbed for the experiments. Numerical handwritten characters have been drawn on a

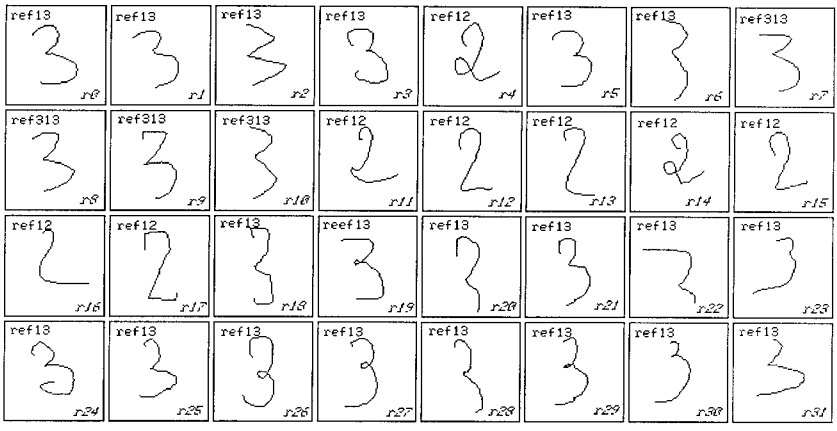


Fig. 4. Sample of numeral handwritten characters used to test the Centroïds Computation Algorithm.

tablet digitizer, then normalized, centered and finally re-sampled to end in a fixed spaced sequences of points. In order to minimize the effect of translations, the actual vectors references are coded as sequences of directions of the elementary segments defined by two consecutive points. The experimental testbed consists of three groups of prototypes characters : Ref13 (digit "3"), Ref313 (another samples of digits "3") and Ref12 (digits "2"). Ref13 is used for training. Both Ref12 and Ref313 are used for testing. The centroid computation algorithm was run with various learning rate ($a = 0.05, a = 0.1, a = 0.2$) and initializations.

V.2 Results

Table 1 summarizes the results for the computations of the centroïd of Ref13 class. The standard deviation of the distances between the centroïd and the references in Ref13 decreases dramatically (60%) within few iterations. The centroïd is obviously moving toward the *middle* of the class – in the elastic distance sense – as the adaptation takes place. Learning seems to be faster with greater learning rate.

V.3 Discussions

As shown in Table 1 the maximum of the distances between the computed centroïd and the references of class Ref313 stay far below the minimum distance between the same centroïd and the class Ref12 test pattern. It can be point out that even with this simple decision process, the centroïd computation and the elastic distance prove to be valuable tools for classification purpose and prototype compilation.
Unfortunately the elastic distance alone does not provide enough information on the exact location of new patterns within the multidimensionnal space of shapes. It would be desirable to construct a vector composed of the distances between the new pattern and all the centroïds found. Many challenging questions remains: how to choose the optimal number of centroïds needed to cover an entire set ? One can for example stop finding new centroïd when two centroïds coincid. Actually the choice of

Table 1. Experimental results for the computation of one centroïd of the ref13 prototypes with (r0) as initial reference.

TRAINING: computation of the centroid of ref13						
Learning rate	a = 0.05		a = 0.10		a = 0.20	
@ Iteration	1	23	1	12	1	7
Criterium	1628	1341	1628	1398	1628	1477
max d(centro, ref13)	3970	2743	3970	2877	3970	3344
mean d(centro, ref13)	1394	1221	1394	1267	1394	1334
std d(centro, ref13)	842	555	842	591	842	635

TESTING (1): classification of ref313 using the centro of ref13 (max d(ref13, ref313) = 705) (min d(ref13, ref313) = 524).						
	pat#	d(.)	pat#	d(.)	pat#	d(.)
max d(c13, ref313)	(r34)	1329	(r34)	1343	(r34)	1344
min d(c13, ref313)	(r8)	770	(r8)	792	(r8)	825

TESTING (2): classification of ref12 using the centro of ref13 (max d(ref13, ref12) = 2630) (min d(ref13, ref313) = 1764).						
	pat#	d(.)	pat#	d(.)	pat#	d(.)
max d(c13, ref12)	(r11)	4674	(r11)	4680	(r11)	4675
min d(c13, ref12)	(r13)	3837	(r13)	3878	(r13)	3986

Note1: **(r#i)** refers to the prototype ri according to the labels in **Fig. 4.**
Note2: **c13** is the centroïd of ref13.

initial references have an effect on the final centroïd location. How to choose the best initial references ?

VI Conclusions

A new learning scheme that computes from several prototypes a reduced representation have been presented. Within the context of handwritten on-line character recognition using the Dynamic Programming approach, the proposed new learning procedure, wich is a variant of a clustering algorithm, reduces a set of prototypes into few centroïds. The centroïd computation algorithm makes an extensive use of the Elastic Distance that does take into account the matching path between two patterns. It has been applied successfully on a reduced scale classification problem. As expected, the adaptive algorithm behaved very well and did not lead to misclassifications even if distances were modified. We plan to run more extensive tests in order to exhibit the effects and benefits of the new scheme.

The proposed learning scheme could be interestingly included in a monoscriptor character recognition system where the problem is to adapt each class model of existing *multiscriptor* characters to the new samples feed in by the user. A more complete version devoted to math formulas recognition and edition is under study and implementation.

REFERENCES

[1] S. Amari,
"A theory of Adaptive Pattern Classifiers,"
IEEE Trans. on Elec. Com. , EC16, pp. 279 - 307, 1967.

[2] H. Sakoe and S. Chiba,
"Dynamic Programming Optimization for Spoken Word Recognition,"
IEEE Trans. ASSP, vol. ASSP - 26, p. 43, Feb. 1978.

[3] R. A. Wagner and M. J. Fischer,
"The String-to-String Correction Problem,"
J. A. C. m. , vol 21, no 1, pp. 168 - 173, 1974.

[4] B. Widrow and M. E Hoff,
"Adaptive Switching Circuits,"
IRE WESCON Conv. Record, part 4, pp. 96 - 104, Aug. 1960.

[5] T. Kohonen,
Self - Organization and Associative Memory,
Springer Verlag, 3rd Edition, 1989.

[6] Y. Le Cun,
Modèles Connexionnistes de l'Apprentissage,
PhD Thesis, Université de Paris VI, 1987.

Velocity Measurement of Granular Flow with a Hopfield Network

Jingeol Lee, Jose C. Principe and Daniel M. Hanes*

Department of Electrical Engineering
University of Florida, Gainesville, FL32611
principe@synapse.ee.ufle.du

* Department of Coastal and Oceanographic Engineering
University of Florida, Gainesville, FL32611

1 Abstract

The transport of granular flow is common to many industrial processes. This paper discusses a methodology to measure the velocity of dry granular solids down an inclined chute using high speed digital images. Acrylic particles have been used as granular solids in our experiment. First, particles are located using normalized correlation. A technique for measuring the velocities of individual acrylic particles is developed based on a Hopfield network to solve the particle correspondence problem between successive images. A new, rigidity constraint is applied to the Hopfield energy function, and the results show better performance than the conventional cost function.

2 Introduction

The objective of this paper is to suggest algorithmic methods to measure the velocity of dry granular solids flowing down an inclined chute under action of gravity using high speed digital images. Measurements of individual particle velocities would represent a major advance in understanding the dynamics of flowing granular materials. One of the challenges in experimental studies is that there are no standard means of measuring the local velocity of granualr materials [1], [2], [3].

Laboratory tests are conducted in a flow simulation system which consists of a $144'' \times 6'' \times 18''$ inclined chute with transparent side walls, a conveyer for particle recirculation, an upper hopper for granular storage and a lower hopper for guiding the discharge to the conveyer. We have used acrylic particles of 3 mm in diameter as granular solids for our experiment. Images have been obtained on a system which takes digital video images up to 1000 frames/ sec. with 239 by 192 resolution. The images have been taken from the side wall.

0-7803-2739-X/95 $4.00 © 1995 IEEE

We propose a method for velocity measurements of individual acrylic particles. This method, first, locates the positions of particles with variance normalized correlation as a pattern recognition technique, followed by a clustering technique, which produces point patterns. Then, correspondence between successive point patterns is solved by a Hopfield network for the velocity measurements of individual particles.

3 Velocity Measurement of Acrylic Particles using Point Correspondence

The variance normalized correlation is proposed here to detect acrylic particles in images. This process also identifies particle locations. Some modifications are made to the variance normalized correlation [4] in order to extract the image data of a particle centered at the mask and to attenuate the image data between the boundary of the particle and that of the mask.

$$r'(m,n) = \frac{\sum_x \sum_y [f'(x,y) - \bar{f}'(x,y)]\,[w'(x-m, y-n) - \bar{w}']}{\left[\sum_x \sum_y [f'(x,y) - \bar{f}'(x,y)]^2 \sum_x \sum_y [w'(x-m, y-n) - \bar{w}']^2\right]^{1/2}} \quad ((Eq.1)$$

In Eq. 1, $f'(x,y) = g(x,y)f(x,y)$ and $w'(x,y) = g(x,y)\,w(x,y)$ where $\bar{w}'$ is the average intensity of the mask, $\bar{f}'(x,y)$ is the average value of $f'(x,y)$ in the region coincident with $w'(x,y)$. The kernel $g(x,y)$ is a 2-D Gaussian function with appropriately chosen standard deviation such that it's distribution has the same size as a particle in our images. The $r'(m,n)$ ranges from -1 and 1, independent of changes in the amplitude of $f(x,y)$ and $w(x,y)$.

This method is applied to two successive images of flowing acrylic particles, as shown in Fig. 1. The camera is tilted such that the chute base is parallel to the horizontal edge of the images. Particles move leftward in this setup. The image A precedes image B by a sampling time of 1 msec. The image size is one fourth of total image (119 by 96). Fig. 2 shows the correlation output from image A with a threshold of 0.985. This result is produced with the template labelled 't' in image A. The standard deviation of the 2-D Gaussian function is 2.5. Each contour corresponds to a correlation output greater than 0.985. Since the variance normalized correlation is independent of amplitude changes in the brightness pattern of both the template and the image area coincident with the template, it produces comparable outputs for particles with differing brightness. Fig. 2 shows that a single template containing the typical brightness pattern of a particle is able to recognizes particles that can be perceived by the human eye.

In order to locate the particle, a search for a point that has a maximum correlation value among the points inside each contour is implemented. The location of the maximum correlation will define the coordinates of the particle center in the image. Since one particle diameter is typically 12 pixels, and particles are densely packed, a clustering technique [5] is applied to group points of high correlation such that the distance between a point and its corresponding clustering center is less than a cluster distance of 5 pixels. Then, the locations that have maximum value of correlation output in every group are searched for. Fig. 3 shows the resultant point pattern for image A (circles) and image B (crosses).

Most particles in one image have correspondence in the other image, however some particles are not detected, or they may have moved away from the field of view. These missed detections are caused by noise in the images and particle occlusions. We will discuss now how to make the correspondence between two point patterns that do not have perfect correspondence.

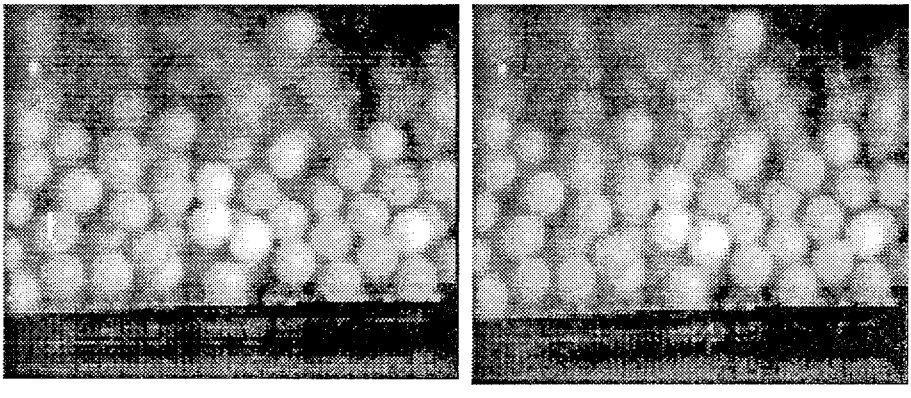


Figure 1 *Two successive images: left - image A, right - image B*

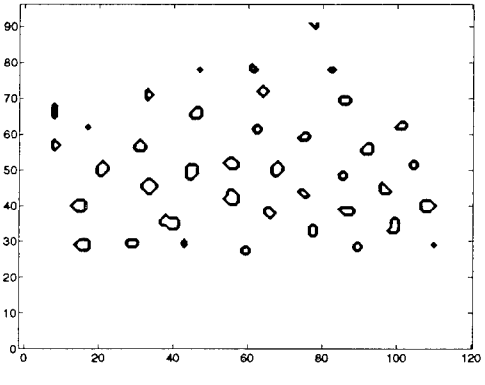


Figure 2 *Correlation output from the image A*

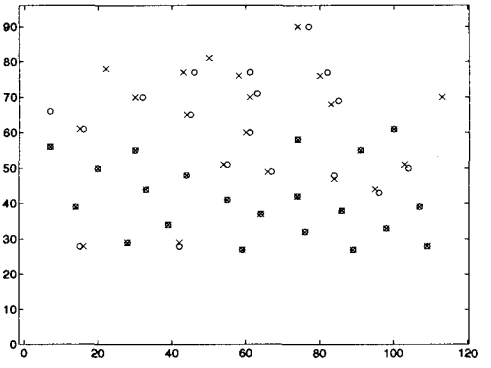


Figure 3 *Point patterns detected in successive images*

4 Point correspondence with Hopfield networks

Many approaches have been attempted for solving the correspondence problem such as graph matching [6], or relaxation techniques [7, 8]. These methods measure the compatibility between prototype features and the image features to obtain the best solution. The problem associated with the conventional graph matching is computationally complexity. The relaxation technique is inherently a local optimization method, and it is sometimes difficult to determine a suitable update rule for the relaxation algorithm. However, a neural network can be employed for the correspondence problem by formulating it as a constrained optimization where all the constraints on the solution can explicitly be included in the cost function. Minimization of the cost function can then be achieved by a recurrent network such as the Hopfield network.

The point correspondence problem has been formulated as a constrained optimization where the cost function (i.e., the energy function in the Hopfield network) representing the constraints on the point correspondence problem, is minimized [9, 10]. To solve the correspondence problem, we construct the energy function in the form of equation (2), whose local minima correspond to solutions for the point correspondence

$$E = -\frac{1}{2}\sum_{x=1}^{M}\sum_{i=1}^{N}\sum_{y=1}^{M}\sum_{j=1}^{N} T_{xi,yj}V_{xi}V_{yj} - \sum_{x=1}^{M}\sum_{i=1}^{N} V_{xi}I_{xi} \quad \text{Eq. 2}$$

where T are the connection weights, V the node activations, I the external input, and M and N are the number of processing elements in a row and column respectively.

It is assumed that the 2-dimensional coordinates of points from two successive images have been obtained and denoted by $A = \{a_1, a_2, \cdots, a_N\}$, $B = \{b_1, b_2, \cdots, b_M\}$. The dynamics of the processing elements arranged in

a matrix converge to the stable state corresponding to the best point matching. The output states of a $M \times N$ permutation matrix V, whose rows refer to point pattern B and columns refer to point pattern A, is interpreted as

$$V_{xi} = \begin{cases} 1 & \text{if there is a match between } A \text{ and } B \\ 0 & \text{otherwise} \end{cases}$$

This means that there can be at most one "1" output in each row and column. If there is no correspondence for a given point in one image to the other image, the corresponding row or column will have only zero entries. Calculating all distances among points between two patterns produces a $M \times N$ distance matrix $dAB = \{dAB(x, i)\}$. Likewise, calculating all distances among points within each pattern produces a $N \times N$ distance matrix $dA = \{dA(i, j)\}$ and $M \times M$ distance matrix $dB = \{dB(x, y)\}$. Since images are taken fast enough to guarantee that the displacements are less than the diameter of a particle, the point correspondence is found under the constraint that the sum of displacements between matched points is minimized.

This method was tested with real images and failed to converge to the global solution many times. A new constraint called the rigidity constraint was incorporated in the cost function. The rigidity constraint requires that the distance $dA(i, j)$ between points i and j of pattern A should equal the distance $dB(x, y)$ between matched points x and y of pattern B, respectively. Based on the above discussion, we construct the following extended energy function,

$$E = \frac{C_1}{2} \sum_{x=1}^{M} \sum_{i=1}^{N} \sum_{j \neq i, j=1}^{N} V_{xi} V_{xj} + \frac{C_2}{2} \sum_{i=1}^{N} \sum_{x=1}^{M} \sum_{y \neq x, y=1}^{M} V_{xi} V_{yj}$$

$$+ \frac{C_3}{2} \left(\sum_{x=1}^{M} \sum_{i=1}^{N} V_{xi} - N_1 \right)^2 + \frac{C_4}{2} \sum_{x=1}^{M} \sum_{i=1}^{N} V_{xi} dAB(x, i)$$

$$+ \frac{C_5}{2} \sum_{x=1}^{M} \sum_{y \neq x, y=1}^{M} \sum_{i=1}^{N} \sum_{j \neq i, j=1}^{N} V_{xi} V_{yj} |dA(i, j) - dB(x, y)| \quad \text{Eq.(3)}$$

the minimum of which corresponds to the best match. The first term equals zero if and only if there is no more than one "1" at each row of V, which means that one point in pattern B is not allowed to be matched to more than one point in pattern A. The second term equals zero if and only if there is no more than one "1" at each column of V. The third term equals zero if and only if there are N_1 entries

of "1" in the matrix V. The fourth term refers to the minimization of the sum of displacements between corresponding points, and the fifth term refers to the rigidity constraint. C_1 through C_5 are constants experimentally determined [11]. We experimentally set N_1=min(M,N)+2. Comparing Eq. (2) with Eq. (3), the connection matrix turns out to be

$$T_{xi,yj} = -C_1\delta_{xy}(1-\delta_{ij}) - C_2\delta_{ij}(1-\delta_{xy}) - C_3 - \qquad \text{(Eq. 4)}$$
$$C_5(1-\delta_{xy})(1-\delta_{ij})|dA(i,j) - dB(x,y)|$$

where $\delta_{ij} = 1$ if $i = j$ and 0 otherwise.The external input is

$$I_{xi} = CN_1 - DdAB(x,i)$$

The outputs of the stable state of the above system, that correspond to the global minimum of the energy function, give the solution for the correspondence problem. Two points must be discussed. First, an energy function with a sum of terms normally produces an energy landscape with saddle points or even local minima. However, here it is more important to constrain the possible solutions such that the network can reach acceptable solutions. The rigidity constraint may seem to conflict with the particle motion, however we expect that the rigidity is maintained between successive frames to some extent due to the high speed images and the particle flow characteristics. The advantage of the rigidity constraint is that it decreases the probability of being caught in local minima during relaxation. A series of experiments that went beyond the scope of this paper monitored the performance of correspondence with and without the rigidity constraint [11]. They showed that the performance with the rigidity constraint was far better, providing almost always the right solution.

4.1 Experiments of point correspondence using the Hopfield network

The point patterns in Fig. 3, obtained with our preprocessing, are used for this experiment. There are four points in the patterns (one marked by circles and three marked by crosses) that have no correspondence due to the image edges, noise in the images and occlusions of particles. The number of circles is 39, and the number of crosses is 41. Since we have lots of points in both patterns, the computation time to obtain a final solution would be large if we compute the correspondence with the total number of points. In order to speed up the computation, we tried a windowing scheme over the patterns. The total pattern area is divided into four rectangular areas such that neighboring areas overlap each other by 10 pixels. The diameter of a particle image is typically 12 pixels. The images are taken with sufficient speed to ensure that particles move less than one diameter between successive images.

This scheme provides correspondences for the missing points caused by the boundary of the rectangular area because the correspondences are updated as the rectangular area is shifted with the overlap. The initial values are selected with a random number generator in the range of - 0.5 to 0.5, which is shown to produce good convergence. The following parameter values were found appropriate for our images.

$$A = B = 500 \qquad E = 73$$

$$C = 800 \qquad D = 160$$

$$N_1 = min\,(M, N) + 2$$

Fig. 4 shows that the correspondence is solved for all points including missing points, as expected, and the velocity field for each particle is derived with one pixel resolution.

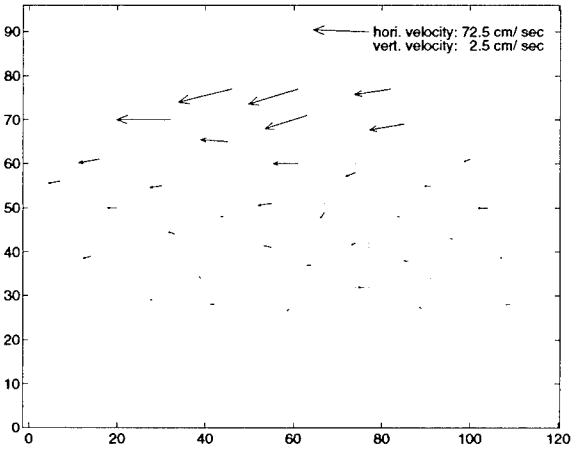


Figure 4 *Velocity field for the particles of Fig 3.*

The results of the correspondence obtained with this method were validated in many artificial images (points in a grid), and the method always found the correct solution. We also collected images from a spinning wheel with glued acrylic particles, and the method always performed perfectly.

5 Conclusion

We propose digital image processing to estimate the velocity of grains flowing down an inclined chute under the action of gravity. A symbolic-token based matching technique is proposed for the measurements of individual acrylic particle velocities, in which the points corresponding to the locations of particles are taken as the symbolic representation. The particles are located with the variance normalized correlation using a 2-D gaussian function to extract the particle from the image data, and the clustering technique to produce a point pattern. The correspondence between point patterns in successive images is formulated as a

constrained optimization problem, and solved by the Hopfield network.

Our experiments show that the added rigidity constraint in the energy function provides a more robust solution to the correspondence problem. The parameters of the energy function must be set experimentally at this point. Further testing of the method is under way, but we have successfully quantified velocity profiles in granular flow with this method.

6 References

[1] Drake T. G., Granular flow: physical experimetns and their implications for microstructural theories", J. Fluid Mech., vol 225, 121-152, 1991.

[2]Savage S.B., Gravity flow of cohesionless granualr materials in chutes and channels, J. Fluid mech. vol 92, part 1, 53-96, 1979.

[3] Xie C.G., Scott A.L., Huang S.M., Plaskowski A., Beck M.S., Mass-flow measurements of solids using electrodynamic and capacitance transducers, J. Physic. Sci. Instru. vol 22, 712-719, 1989.

[4] Peter J. Burt, Chihsung Yen and Xinping Xu, Local Correlation Measures for Motion Analysis, a Comparative Study, IEEE Computer Society Conference on Pattern Recognition and Image Processing (1982), Las Vegas, Nev.

[5] Yoh-Han Pao, Adaptive Pattern Recognition and Neural Networks (1989), Addison-Wesley

[6] W. H. Tsai and K. S. Fu, Error-Correcting Isomorphisms of Attributed Relational Graphs for Pattern Analysis, IEEE Trans. Syst., Man, Cybernet. (1979), Vol. SMC-9, pp 757-768, December

[7] Sanjay Ranade and Azriel Rosenfeld, Point Pattern Matching by Relaxation, Pattern Recognition (1980), Vol. 12, pp 269 - 275

[8] Daryl J. Kahl, Azriel Rosenfeld and Alan Danker, Some Experiments in Point Pattern Matching, IEEE Trans. Syst., Man, Cybernet. (1980), Vol. SMC-10, No. 2, February

[9] Etienne Barnard and David P. Casasent, Multitarget Tracking with Cubic Energy Optical Neural Nets, Applied Optics (1989), Vol. 28, No. 4, February

[10] Nasser M. Nasrabadi and Chang Y. Choo, Hopfield Network for Stereo Vision Correspondence, IEEE Transactions on Neural Networks (1992), Vol. 3, No. 1, January

[11] Lee J., Measurement of granular flow dynamics with high speed digital images, Ph.D. dissertation, University of Florida, 1994.

Neural network-based image segmentation for image interpolation

Stefano Marsi, Sergio Carrato

D.E.E.I., University of Trieste

via A. Valerio, 10, 34127 Trieste, Italy

tel: +39 40 6767147; fax: +39 40 6763460

e-mail: marsi@imagets.univ.trieste.it

ABSTRACT — **A novel image interpolation scheme is presented, in which a neural network is used to segment the image according to the presence of oriented edges; a set of different directional adaptive filters then interpolate the image, the filter outputs being weighted according to the neural network output. The filters are designed in order to accurately reproduce both smooth areas and sharp edges.**

In the paper, the structure is presented and the neural network and the directional filters are described. Simulation results show that both objective and subjective image quality obtained by the proposed method are higher than using linear interpolators.

1 INTRODUCTION

Image interpolation is an important field of research in image processing. There are several applications where an accurate expansion of an image is needed, e.g. in medical imaging (to zoom on details which can correspond to pathological tissues) and in cartography (to obtain accurate maps from satellite data). Image expansion is also one of the key steps in pyramidal coding, which is an efficient method used to code images: its features include low bit rate, multiresolution approach (useful e.g. in progressive transmission or in multimedia applications), and absence of artifacts (which in turn may be present in block-based techniques).

Several linear [6] and nonlinear [4, 8, 2, 3, 10] algorithms have been presented in the literature. However, they generally smooth the edges, so that a blurred image is often obtained. Cubic spline interpolation also tends to add an overshoot in presence of steps [9], with a consequent ringing effect along the border of the objects. It is then interesting to develop algorithms which are able to accurately reconstruct the edges in order to give sharp images with good subjective quality. In [9], an interesting interpolating scheme is proposed, which is based on the minimization of a suitable functional using

0-7803-2739-X/95 $4.00 © 1995 IEEE

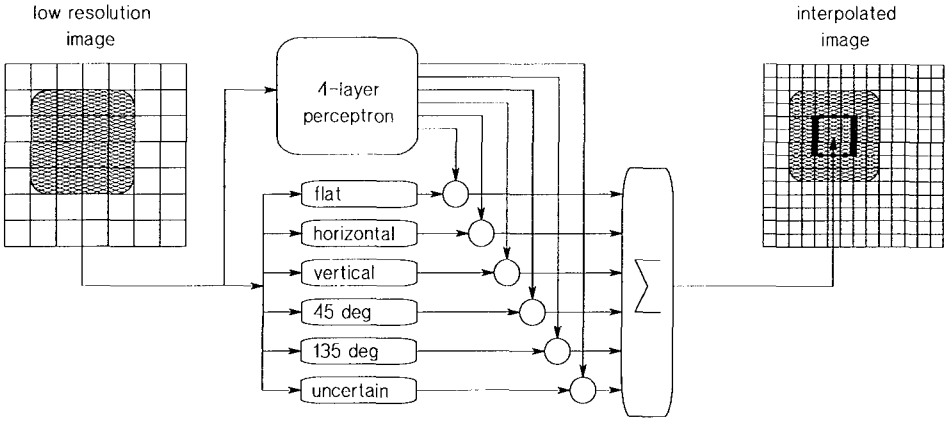


Figure 1: Block diagram of the proposed interpolating scheme.

maximum *a posteriori* estimation according to a bayesian approach.

In this paper, a two-step algorithm is proposed where a neural network (NN) segments the image according to the presence and the orientation of object contours; then, several adaptive filters, controlled by the NN outputs, actually interpolate the image by smoothing the flat zones while preserving the sharp edges.

In the following sections, we first describe the proposed filtering structure, then we give more details on the neural network training and on the adaptive filters used. Some simulation results are then presented, which demonstrate the good objective and subjective quality of the obtained images. We also show the effectiveness of the proposed scheme in a pyramidal-based coding scheme.

2 THE ALGORITHM

The block diagram of the proposed interpolating scheme is presented in Fig. 1.

The low resolution image is scanned using a 4×4 window. For each window position, the gray values of its pixels are fed into a 16-input 4-layer perceptron. The neural network has 6 outputs, and has been trained in order to recognize flat zones (first output), oriented edges (i.e., horizontal, vertical, and according to the two main diagonals) and "uncertain" situations as very small details or edges not centered in the mask (sixth output). The use of a neural-based algorithm for edge recognition is expected to give better performances with respect to classical gradient-based techniques, especially in presence of noise.

If an interpolation factor equal to 2 is used, the data in the 4×4 window have to be expanded into a 7×7 square (see Fig. 2), but only its 9 central

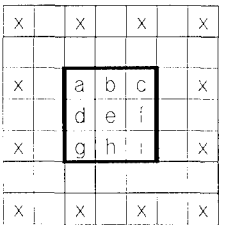


Figure 2: Each 4×4 block of the input image is expanded into a 7×7 block. b, d, e, f, and h are the pixels to be evaluated, the others being already known.

pixels (referred to as a to i in the figure) are to be estimated, the other pixels being considered when the window is moved to the appropriate position.

We copy the four pixel at the corners of the central square (i.e., a, c, g and i) from the input image. To evaluate the other 5 pixels (b, d, e, f, and h) 6 different filters are used, which are specialized for the six different situations mentioned above; their output is weighted according to the outputs of the NN. For example, if a vertical edge is encountered, the NN—which hopefully recognizes it—activates the corresponding output, so that only the output of the filter dedicated to vertical edges is used to reconstruct the output image. In case of intermediate situations, the NN partially enables more than one output, so that a weighted mean of several filters is computed.

It has to be noted that while pixel e is estimated only once, this is not true for the other pixels, which are estimated twice. For example, the pixel below e is considered as h when processing the block considered in the figure, and as b when dealing with the corresponding block of the following line. This is taken into account by assigning to the output image the mean of the two estimates.

In the following, some details are given related to the perceptron training and the adaptive filters.

2.1 Perceptron training

As already mentioned, a multilayer perceptron is used in order to segment the image according to the presence and orientation of edges. More precisely, we use a $16 \times 25 \times 25 \times 6$ perceptron [7] with sigmoidal activation function and bias. We set to 25 the number of nodes in the two hidden layers as a compromise between efficient pattern recognition and reasonable computational complexity.

The NN is trained using the classic backpropagation algorithm with momentum [7]; weights are set to random values during initialization. The training set is composed of 2000 synthetic 4×4 image blocks of the already mentioned 6 types: flat, 4 oriented edges, and uncertain zones. Flat blocks

Figure 3: Example of blocks taken from the training set: flat, horizontal, vertical, diagonal (two edges at 45° and two edges at 135°), and uncertain zone blocks.

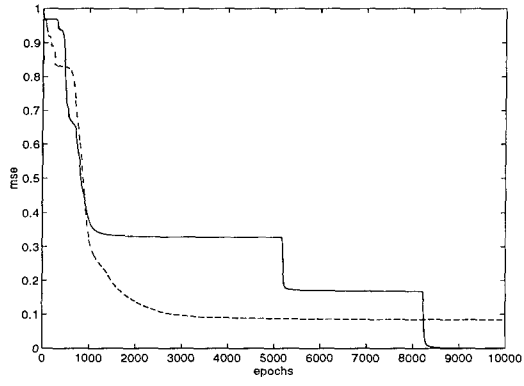


Figure 4: Multi-layer perceptron training: mean output error versus training time for a $16 \times 25 \times 25 \times 6$ NN, with (solid line) and without (dashed line) weight perturbation.

are obtained giving the same random value to all its 16 pixel, and subsequently summing some low amplitude random noise. Horizontal and vertical edges are generated similarly, but two different values are used for the two halves of the block. For the diagonal edges the process is analogous, but two possible edge positions (over or under the diagonal) have to be considered. Uncertain zone blocks are obtained using random values. An example of blocks taken from the training set is shown in Fig. 3.

In order to increase the probability of escaping from local minima, we also add some noise to the learning process by slightly perturbing the weights after each update; similarly to what is done in simulated annealing minimizations [5], the noise is progressively reduced while the training proceeds. The training convergence is shown in Fig. 4, where the mean output error is plotted versus learning time for the two cases (with and without weight perturbation). We experimentally found that, in this specific problem, networks with fewer inner nodes (e.g., $16 \times 15 \times 15 \times 6$) do not converge to a reasonably good local minimum.

2.2 The adaptive filters

As already mentioned, six different filters are used for the image interpolation process.

When a flat zone is recognized, a linear low-pass ($\omega_c = 0.5$) filter with a 5×5 mask is used. A 3×3 linear filter is also used in case of uncertain areas: if no other information is available, it is reasonable to use a very small mask of the type

$$\frac{1}{16}\begin{vmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{vmatrix}$$

in order to take into account only pixels which are very close to those which are being interpolated.

The edge reconstruction is more complex. We assume that the low resolution image has been generated by low-filtering and down-sampling a higher resolution image. This is actually what is explicitly done in a pyramidal coding scheme; however, even if no higher resolution image exist (e.g., in medical imaging or in cartography) a low pass filter is—or should be—used before sampling, in order to reduce aliasing.

When an edge has to be interpolated, there is an uncertainty about its exact position (1 pixel for horizontal and vertical edges, and 1/2 pixel for diagonal ones). We designed some nonlinear filters which estimate the exact position of the edge and reconstruct and possibly enhance it.

Let us consider the one-dimensional case for simplicity, with reference to Fig. 5. The task is to accurately reconstruct the central element of a 7-element window (reduced to 4 elements after decimation) in which a step is present.

In row A, the step is considered in the two possible cases of interest, which will be referred to as "even" (column a) and "odd" (column b) position. The other possible positions, more at the right or more at the left, should be taken into account when the window is shifted by two (by one for the decimated data); indeed, in these cases the NN recognizes them as uncertain situations, and the outputs related to oriented edges are not activated.

After low-pass filtering (row B) and decimation (row C) some information has been missed. However, the position of the original step may be guessed by considering the decimated data $d(i), i = 1 \ldots 4$: if $|d(2) - d(1)| < |d(4) - d(3)|$ then the step position is even, otherwise it is odd. Consequently, the value of the central element may be set closer to $d(2)$ or $d(3)$ in case of even or odd position, respectively. In the particular case of the figure, for example, it is possible to set it *equal* to $d(2)$ or $d(3)$, being $|d(2) - d(1)| = 0$ or $|d(4) - d(3)| = 0$ in the two cases. The result is a reconstructed edge which is much sharper than using linear interpolation.

The extension to the two-dimensional case is rather simple, even if four possible step orientations have to be considered (horizontal, vertical, and

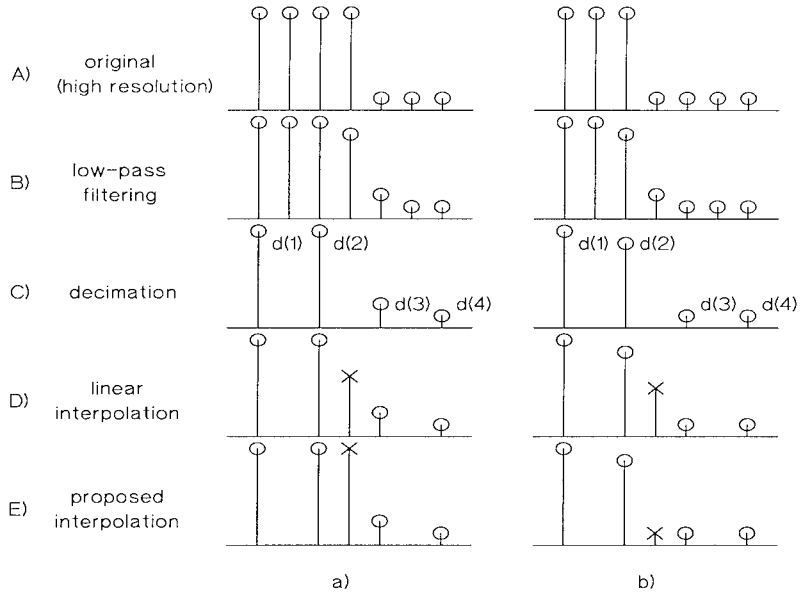


Figure 5: Example of edge reconstruction in the one-dimensional case, for an "even" (a) and "odd" (b) step position (see text);'o': known pixels; '×': pixel to be evaluated.

according to the two main diagonals).

It has to be observed that, being only the decimated data available, there is no way to distinguish between a step which has been smoothed by the low-pass filter and a gentle slope in the original data, which is therefore incorrectly sharpened. However, it is generally acknowledged that a moderate increase in the image contrast (with noise being kept controlled) is beneficial to the subjective quality; this is by the way the rationale of various *unsharp masking* algorithms [6] which may be found in the literature.

3 SIMULATION RESULTS

We tested the proposed algorithm on several 512×512 images ("Lenna", "airport", and "pentagon"), which have been decimated after low-pass filtering with a linear FIR filter with $\omega_c = 0.5$. In Tab. 1, the mean square error (mse) in the reconstructed images are reported in case of linear interpolation and using the proposed method; it may be noticed that a significant reduction in the mse is obtained with our technique.

More results are reported in the following, which are related to the image "Lenna".

In Fig. 6, the activation of the six NN output nodes is reported. It may

Table 1:
COMPARISON OF DIFFERENT INTERPOLATION SCHEMES

Image	mse (linear interp.)	mse (proposed scheme)
Lenna	45.0	44.3
Airport	155.3	146.1
Pentagon	87.7	80.1

be seen that the NN accurately locates the oriented edges. The robustness of the edge extraction with respect to noise is shown in Fig. 7, where a comparison with a gradient-based edge technique is presented; for simplicity only the vertical edges are considered (similar results are obtained for the other orientations). It is apparent that gradient is more sensitive to noise.

As already mentioned, the proposed interpolating scheme should result in a higher visual quality with respect to linear interpolation. In order to allow a better comparison, in Fig. 8 only a detail of the test image is reported, together with the output of the two methods. It may be noticed that the presence of sharper edges improves the subjective quality of the image.

We also present an application of our algorithm in the field of image coding, and in particular within a pyramidal coding scheme [1]. More precisely, we replace the linear interpolation in the lower (higher resolution) layer with the proposed algorithm: thanks to the good subjective image quality that can be obtained, the corresponding error image may not be transmitted, with a consequent substantial bit rate reduction. In Fig. 9, the resulting image is reported: bit rate is 0.13 bpp, and PSNR is 27.41 dB. The PSNR is not quite high, but the visual quality of the image is good due to the presence of sharp edges.

4 CONCLUSIONS

We presented an image interpolation algorithm which uses a neural network-based classifier to recognize perceptually significant parts of the image. The information of the classifier is exploited to control the operation of several filters, so that an edge-preserving (or even edge-enhancing) interpolation is obtained.

References

[1] P. J. Burt and E. H. Adelson, "The laplacian pyramid as a compact image code," *IEEE Trans. Commun.*, vol. COM-31, pp. 532–540, Apr. 1983.

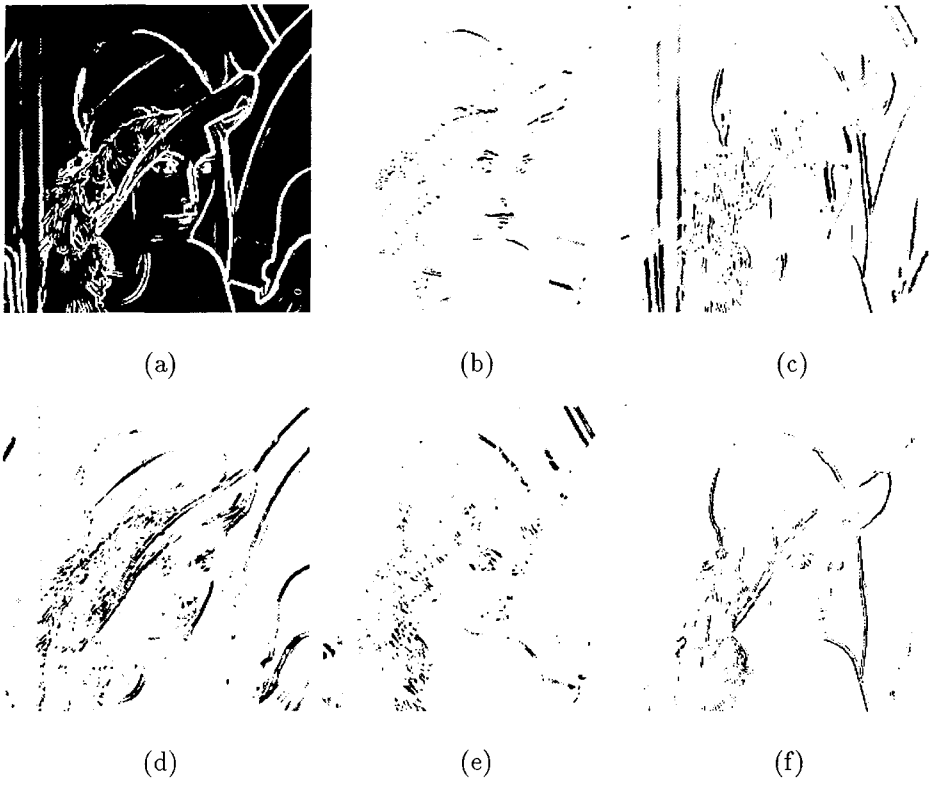


Figure 6: Outputs of the multilayer perceptron for the test image "Lenna". The activation of each output node is shown (white and black mean low and high output, respectively): (a) flat zones; (b) horizontal edges, (c) vertical edges, (d) and (e) diagonal edges, (f) uncertain areas.

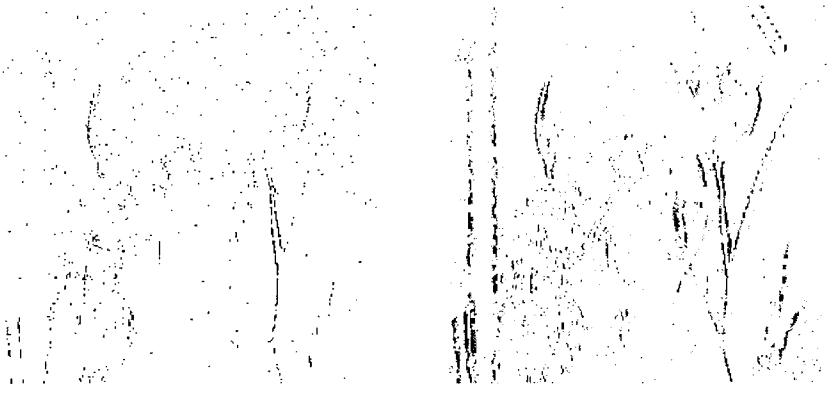
[2] T. C. Chen and R. J. P. Figueiredo, "Two-dimensional interpolation by generalized spline filters based on partial differential equation image models," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, no. 3, pp. 631–642, 1985.

[3] N. B. Karayiannis and A. N. Venetsanopoulos, "Image interpolation based on variational principles," *Signal Processing*, vol. 25, pp. 259–288, 1991.

[4] R. G. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, no. 6, pp. 1153–1160, 1981.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, May 1983.

(a)



(b) (c)

Figure 7: Extraction of the vertical edges in case of noise: (a) noisy image (gaussian, $\sigma = 20$, in the upper part, and impulsive, with probability 5 %, on the lower one); (b) gradient; (c) proposed neural-based approach.

[6] J. S. Lim, *Two-dimensional signal and image processing.* London: Prentice-Hall, Inc., 1990.

[7] R. P. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Magazine*, pp. 4–21, Apr. 1987.

[8] A. J. Parker, R. V. Kenyon, and D. E. Troxel, "Comparison of interpolating methods for image resampling," *IEEE Trans. Med. Imaging*, vol. MI-2, no. 1, pp. 31–39, 1983.

[9] R. R. Schultz and R. L. Stevenson, "A bayesian approach to image expansion for improved definition," *IEEE Trans. on Image Processing*, vol. 3, pp. 233–242, May 1994.

(a) (b) (c)

Figure 8: Detail of the test image: (a) decimated image, (b) linear interpolation, (c) proposed method.

Figure 9: "Lena" compressed at 0.13 bpp using pyramidal coding; PSNR is 27.41 dB.

[10] U. Unser, A. Aldroubi, and M. Eden, "Fast B-spline transforms for continuous image representation and interpolation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 13, no. 3, pp. 277–285, 1991.

LEARNING A DISTRIBUTION-BASED FACE MODEL FOR HUMAN FACE DETECTION

Kah-Kay Sung and Tomaso Poggio

Artificial Intelligence Laboratory
and
Center of Biological and Computational Learning
Massachusetts Institute of Technology
Cambridge, MA 02139, USA

Abstract

We present a distribution-based modeling cum example-based learning approach for detecting human faces in cluttered scenes. The distribution-based model captures complex variations in human face patterns that cannot be adequately described by classical pictorial template-based matching techniques or geometric model-based pattern recognition schemes. We also show how explicitly modeling the distribution of certain "face-like" non-face patterns can help improve classification results.

1 Introduction

Finding human faces automatically in a cluttered image is an important first step to a fully automatic face recognition system. It also has many potential applications ranging from surveillance and census systems to human-computer interfaces. Human face detection is difficult because there can be huge variations in the appearance of face patterns. Because many of these variations are difficult to parameterize, traditional fixed template pattern matching techniques [2] [3] and geometrical model-based object recognition approaches [1] tend to perform inadequately for detecting faces. Some non-parametric approaches [6] [4] have been recently proposed for representing and detecting face patterns, but so far, they have only been successfully demonstrated on images with little background clutter.

This paper describes an example-based learning approach for finding un-occluded vertical frontal views of human faces in cluttered scenes. To capture the full range of permissible variations in face patterns, the approach synthesizes a distribution-based model of frontal face views from an example database of face images. In order to perform pattern matching with the model, it learns a set of classification thresholds and parameters for separating "face" and "non-face" patterns, based on a set of distance measurements between the test pattern and the model.

0-7803-2739-X/95 $4.00 © 1995 IEEE

Figure 1: (a): A "canonical" face pattern. (b): A 19×19 mask for eliminating near-boundary pixels of canonical face patterns. (c): An example of a naturally occuring pattern that resembles a human face when viewed in isolation, but is clearly not a face when viewed in the context of its environment.

2 System Overview and Approach

Our approach finds faces by searching the image for square patches of the human face, whose upper boundary lies just above the eyes and whose lower edge falls just below the mouth (see Figure 1(a)). We shall henceforth refer to square patterns of this general structure as "canonical face patterns". The search for these face-like window patterns is done over multiple scales. At each scale, the image is divided into overlapping square windows of the current size. The system then attempts to classify the enclosed image pattern under each window as being either "a face" or "not a face" of the current scale, by matching the enclosed pattern with our face model.

Clearly, the most critical part of our system is the algorithm for classifying window patterns as "faces" or "non-faces". Section 3 describes the distribution-based model for representing face patterns. Section 4 describes the distance measurements for matching new window patterns to the model, and the decision procedure for classifying window patterns based on their distance measurements. In Section 5, we analyze our face detection technique and evaluate its performance.

3 A Distribution-based Face Model

Our distribution-based modeling scheme tries to represent canonical faces as the set of all masked 19×19 pixel patterns that are canonical face views. Suppose we apply the mask of Figure 1(b) to each 19×19 image and treat each unmasked pixel as a vector dimension, then the class of all masked 19×19 images forms a vector space whose dimensionality equals the number of unmasked image pixels. Each masked 19×19 image pattern maps to a specific vector space location, and the set of all 19×19 pixel canonical face patterns maps to a fixed volume in this multi-dimensional vector space. So, in theory, one can model the class of all canonical face views by identifying the

portion of this multi-dimensional vector space that corresponds to canonical face patterns, and representing the region in some tractable fashion.

Figure 2 explains how we *approximate* the volume of canonical face patterns with limited data. Basically, we use a reasonably large example database of 19×19 canonical face patterns to obtain a coarse but fairly reliable representation of the actual canonical face manifold. We also use a carefully chosen database of non-face patterns to help refine the manifold representation by explicitly carving out regions around the "face" sample distribution that do not correspond to canonical face views. We shall explain how we synthesize our special database of "non-face" patterns in Section 3.2.

3.1 Modeling the Distribution of "Face" Patterns

We use a database of 4150 masked canonical "face" patterns to infer the volume of face views in our multi-dimensional image vector space. The database consists of 1067 real face patterns and 3083 *virtual* [5] face patterns, artificially generated from the real patterns via some simple affine transformations. The virtual patterns help ensure that our final database contains a reasonably dense and representative sample of canonical face patterns.

We approximate the "face" pattern distribution in a piecewise-smooth fashion using 6 multi-dimensional Gaussian clusters. Each Gaussian cluster consists of a centroid location and a full covariance matrix that describes the local data distribution around the centroid. The 6 clusters are obtained by performing *elliptical k-means* clustering on the "face" sample distribution (see [7] for details). The *elliptical* clustering algorithm differs from the traditional *k-means* algorithm in that it fits full covariance Gaussian clusters instead of isotropic Gaussian clusters to the data distribution. It approximates the "face" data distribution more closely with the same number of clusters, because locally, the "face" data distribution can be several orders of magnitude more elongated in certain image vector space directions than others.

The piecewise-smooth modeling scheme serves two important functions. First, it generalizes by applying a prior smoothness assumption to the observed "face" data distribution. Second, it serves as a tractable scheme for representing an arbitrary data distribution by means of a few Gaussian basis functions.

3.2 Refining the "Face" Distribution by Modeling "Non-Face" Patterns

There are many naturally occuring "non-face" patterns in the real world that look like faces when viewed in isolation (see for example Figure 1(c)). Because we are coarsely representing the canonical face manifold with 6 Gaussian clusters, some of these face-like patterns may even be located nearer the "face" cluster centroids than some real "face" patterns. This may give rise to misclassification problems, because in general, we expect the opposite to

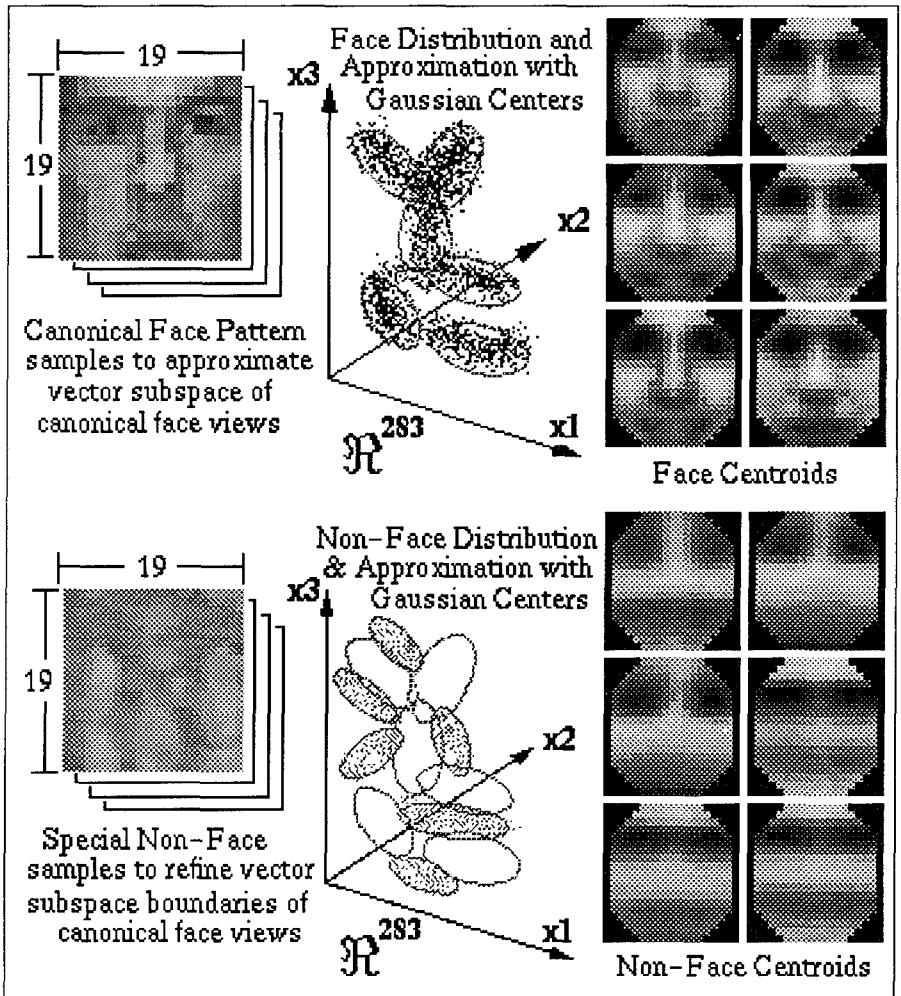


Figure 2: Our distribution-based canonical face model. **Top Row:** We use a representative sample of canonical face patterns to approximate the volume of canonical face views in a *masked* 19×19 pixel image vector space. We model the "face" sample distribution with 6 multi-dimensional Gaussian clusters. **Bottom Row:** We use a selection of non-face patterns to help refine the boundaries of our Gaussian mixture approximation. We model the "non-face" sample distribution with 6 Gaussian clusters. Our final model consists of 6 "face" clusters and 6 "non-face" clusters. Each cluster is defined by a centroid and a covariance matrix. The 12 centroids are shown on the right. **Note:** The two distribution plots are fictitious and are shown only to help with our explanation. The 12 centroid images are real.

be true, i.e. face patterns should lie nearer the "face" cluster centroids than non-face patterns.

We use our *elliptical k-means* algorithm to obtain 6 "non-face" clusters and their covariance matrices from a database of 6189 face-like patterns. The database was incrementally generated in a "boot-strap" fashion by first building a reduced version of our face detection system with only "face" clusters, and collecting all the *false positive* patterns it detects over a large set of natural images without faces.

Our final distribution-based model thus consists of 6 "face" clusters for coarsely approximating the canonical face pattern manifold in the image vector space, and 6 "non-face" clusters for refining the manifold by carving out non-face regions in the image vector space near the "face" clusters.

4 Matching and Classifying Patterns with the Model

To detect faces, our system resizes each candidate window pattern to 19×19 pixels and matches the resized pattern against our distribution-based face model to obtain a set of "difference" measurements. A trained classifier determines, based on the set of "difference" measurements, whether or not the test pattern is a frontal face view.

Each set of "difference" measurements is a vector of 12 distances between the test window pattern and the model's 12 cluster centroids in our multi-dimensional image vector space. One can interpret our vector of distances as the test pattern's displacement from 12 key reference location on the canonical face pattern manifold, and hence as a crude "difference" notion between the test pattern and the entire "canonical face" pattern class.

We use a 2-Value metric to encode the distance between the test pattern and a cluster centroid. The first distance value is a directionally dependent *Mahalanobis* distance between the test pattern and the cluster centroid, in a vector sub-space spanned by the cluster's 75 largest eigenvectors. This component computes a normalized pattern difference along the major directions of the local data distribution. It ignores pattern differences in the smaller eigenvector directions where the eigenvalue estimates are poor.

Let $\vec{x}$ be the column vector test pattern, $\vec{\mu}$ be the cluster centroid, E_{75} be a matrix with 75 columns, where column i is a unit vector in the direction of the cluster's i^{th} largest eigenvector, and W_{75} be a diagonal matrix of the corresponding 75 largest eigenvalues. The covariance matrix for the cluster's data distribution in the 75 dimensional sub-space is $\Sigma_{75} = (E_{75} W_{75} E_{75}^T)$, and the first distance value is:

$$\mathcal{D}_1(\vec{\mathrm{x}}, \vec{\mu}) = \frac{1}{2}(75 \ln 2\pi + \ln |\Sigma_{75}| + (\vec{\mathrm{x}} - \vec{\mu})^{\mathrm{T}} \Sigma_{75}^{-1} (\vec{\mathrm{x}} - \vec{\mu})).$$

The second distance component is a standard Euclidean distance between

the test pattern $\vec{x}$ and its projection $\vec{x_p}$ in the 75 dimensional largest eigenvector sub-space. It is a robust directionally independent measure that accounts for pattern differences not captured by the first component.

$$\mathcal{D}_2(\vec{x}, \vec{\mu}) = ||(\vec{x} - \vec{x_p})|| = ||(I - E_{75}E_{75}^T)(\vec{x} - \vec{\mu})||.$$

We use a *multi-layer perceptron* (MLP) net to classify new window patterns as "faces" or "non-faces", based on their vector of distance measurements to the 12 cluster centroids. The net has 12 *pairs* of input terminals (for the 12 pairs of distance values) and one output node that returns a '1' for "face" patterns and a '0' otherwise. Our experiments show that the number of hidden units and network connectivity do not significantly affect the classifier's performance. We trained the net on a database of 4150 "face" patterns and 43166 "non-face" patterns. The non-face patterns include the 6189 face-like patterns used by our model for synthesizing the "non-face" distribution. The net was trained with a standard backpropagation learning algorithm until the output error stabilized at a very small value.

5 Results and Performance Analysis

Figure 3 shows some sample face detection results by our system. The system operates on window sizes of 19×19 pixels to 100×100 pixels at width increments of 120%. "Face" window patterns are marked with an appropriately sized dotted box in the output image. Many of the faces in Figure 3 are enclosed by multiple dotted boxes because the system has detected those faces either at a few different scales or at a few slightly offset window positions. The results show that the system (1) does not make many false positive errors (none in this case) even for fairly complex scenes, (2) detects faces successfully at different scales, and (3) detects real faces as well as hand-drawn faces. The system has a 96.3% face detection rate on a test database of 301 high quality CCD images of real people, and rarely makes any false detects, even on images with very cluttered background patterns.

We conducted the following experiment to determine the importance of "non-face" clusters in our distribution-based face model. We built two new systems: the first with 12 "face" clusters and no "non-face" clusters, and the second with only 6 "face" clusters. Table 1 compares the performance statistics of the two new systems with that of our original system. As expected, the original system with "non-face" clusters clearly out-performs the two systems without "non-face" clusters in terms of having a higher detection rate and fewer false positives. This suggests that the "non-face" distance measurements are in fact a very discriminative set of additional features for face pattern classification.

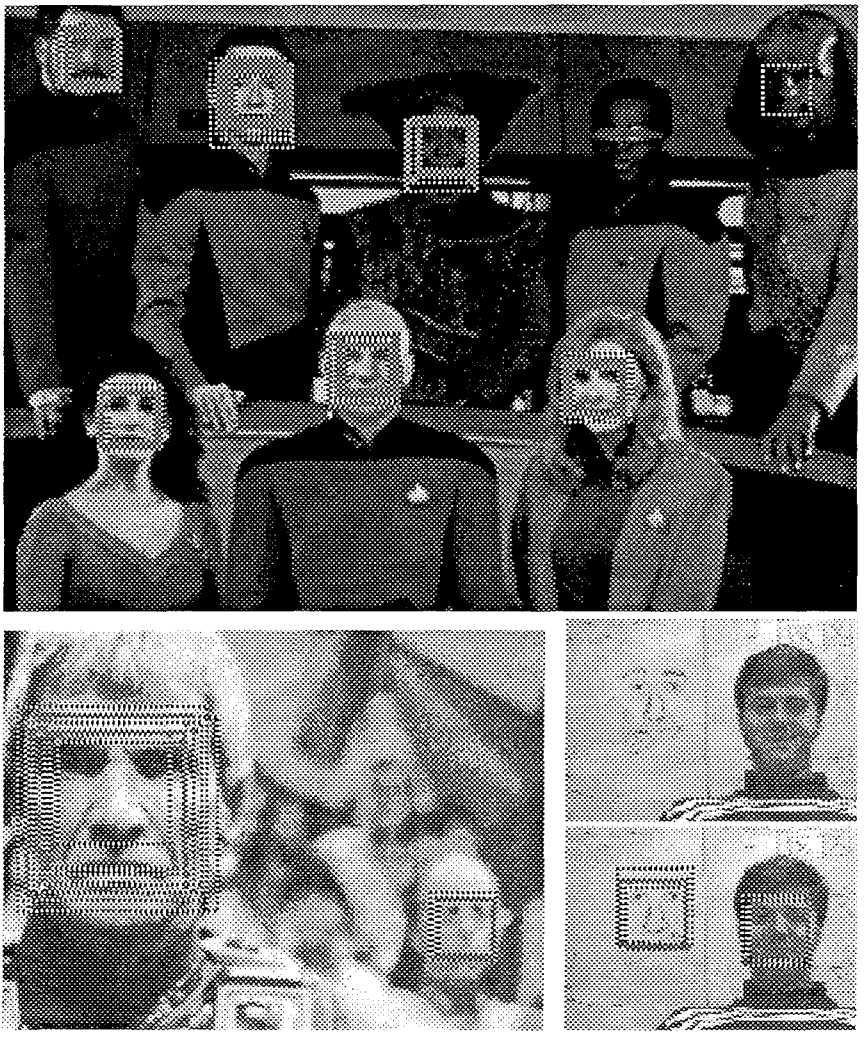


Figure 3: Some face detection results by our system. "Face" window patterns are marked with an appropriately sized dotted box in the output image. Many of the faces are enclosed by multiple dotted boxes because the system has detected those faces either at a few different scales or at a few slightly offset window positions. Notice that the current system does not detect Geordi's face in the top image. This is because Geordi's face differs significantly from the notion of a "typical" face pattern in our training database of face patterns — his eyes are totally occluded by an opaque metallic visor.

Classifier Architecture	Composition of Clusters in Model					
	6 Face & 6 Non-Face		12 Face		6 Face	
Multi-layer Perceptron	96.3%	3	85.3%	21	59.7%	17
Single Perceptron	96.7%	3	52.1%	6	66.6%	25

Table 1: Summary of performance figures for experiment on importance of "Non-Face" clusters. Detection rates versus number of false positives for different classifier architectures and composition of clusters in distribution-based model. The two numbers for each entry are: **Left:** detection rate on the test database of 301 high quality images. **Right:** total number of false positives for the test database.

6 Conclusion

We have successfully developed a distribution-based modeling cum example-based learning technique for representing and detecting frontal views of human faces in images. The distribution-based model captures pattern variations in face images that are difficult to parameterize using traditional pattern matching and object recognition techniques. We believe that the same distribution-based modeling cum example-based learning methodology can be easily extended to take on feature detection and pattern recognition tasks in other problem domains.

References

[1] W. E. L. Grimson and T. Lozano-Perez. Model-Based Recognition and Localization from Sparse Range Data. In A. Rosenfeld, editor, *Techniques for 3-D Machine Perception*. North-Holland, Amsterdam, 1985.

[2] B. Kumar, D. Casasent, and H. Murakami. Principal Component Imagery for Statistical Pattern Recognition Correlators. *Optical Engineering*, 21(1), Jan/Feb 1982.

[3] A. Mahalanobis, A. Forman, N. Day, M. Bower, and R. Cherry. Multi-Class SAR ATR using Shift-Invariant Correlation Filters. *Pattern Recognition*, 27(4):619–626, April 1994.

[4] A. Pentland, B. Moghaddam, and T. Starner. View-based and Modular Eigenspaces for Face Recognition. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 84–91, June 1994.

[5] T. Poggio and T. Vetter. Recognition and Structure from One (2D) Model View: Observations on Prototypes, Object Classes, and Symmetries. A.I. Memo No. 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[6] P. Sinha. Object Recognition via Image Invariants: A Case Study. In *Investigative Ophthalmology and Visual Science*, volume 35, pages 1735–1740, Sarasota, Florida, May 1994.

[7] K. Sung and T. Poggio. Example-based Learning for View-based Human Face Detection. Technical Report AIM–1521, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, December 1994.

ACTION-BASED NEURAL NETWORKS FOR EFFECTIVE RECOGNITION OF IMAGES

Vassilios Alexopoulos and Stefanos Kollias
Computer Science Division
Department of Electrical and Computer Engineering
National Technical University of Athens, Greece
e-mail : valex@theseas.ntua.gr

Abstract - This paper presents a novel approach to the recognition of images or scenes, by associating human perception actions to them and introducing neural network architectures that are able to learn the derived representations. The approach, is related to recent research efforts towards a deeper understanding of human information processing and uses appropriate recurrent neural networks for generating the desired associations in time varying environments. Initial results obtained when applying the proposed approach to the problem of recognition of images of objects, that are deformed and/or corrupted by noise, are very encouraging.

1. INTRODUCTION AND PROBLEM DESCRIPTION

There is presently increasing pressure to develop an effective approach to Machine Intelligence (MI). This arises from numerous areas: the increasing load of complex information traversing the internet and other communication networks, the possibility of achieving more effective human/computer interaction, the need to develop autonomous agents to visit hostile environments, or the creation of robots able to recognise their environment without human intervention and to collaborate effectively with each other.

At the same time, and clearly with a close relationship to the above demands for MI, pressure has developed to obtain a deeper understanding of human information processing (HIP). Since the human brain is the best (and only) known version of an intelligent machine (intelligent, that is, at the highest level) then this avenue of investigation is not one to be neglected in the pursuit of MI. One of the basic barriers in this area is that of understanding meaning. This can be seen to arise in many fields. Thus, one of the main problems of human computer interaction (HCI) is that of giving meaning to messages emitted by humans, as well as interpreting in a human-based linguistic system the states of the machine. In communication networks the contents of the symbols being transmitted need to be given a semantic structure, so that compression and regeneration of noisy signals can be

0-7803-2739-X/95 $4.00 © 1995 IEEE

achieved using higher-order meanings of messages. In order to develop an autonomous robot it appears necessary to be able to allow it to create a higher order representational system in order to be able to manipulate the object representations it builds up in its explorations of its environment.

These pressures indicate the need for the development of a general theory of semantics, which would give meaning to the symbols being used by humans about their experience in the real world and allow a machine to develop a relevant semantics of its own experience. There is presently no accepted or effective theory of this sort, nor one which linguists or those involved with HCI are able to use. In particular there is no way of developing such a semantics, so as to be able to use it in a variable environment [1].

The new approach proposed in this paper includes the development of an 'action'-based structure for input processing in image recognition problems. According to it, visual inputs are to be encoded simultaneously, for example, with eye movements so as to give a conjoined pattern-eye movement sequence record. Other actions, such as movements in relation to visual patterns (as occur in obstacle avoidance), can also be simultaneously encoded. Such techniques are already under analysis [2], [3] and are proving effective in visual target recognition. Action-based imagery (analogous to REM sleep) can be viewed as a method of developing suitable object representations, in which the action-dependence is reduced. The target is to develop a representational system which is rooted in perception on one hand and action on the other.

The proposed approach involves a variety of artificial neural network modules for learning higher order structures, including action-based patterns associated with visual inputs. These networks learn correlations of lower order features, which are extracted from the observed scenes or data over suitable time intervals. Recurrent networks, trained by temporal backpropagation or real-time recurrent algorithms [4], [6], are able to generate time-evolving sequences of decisions, in one dimensional or two dimensional (image) form, at their outputs. This makes them suitable for correlating action-based data, such as eye-movement patterns, with the visual scenes during training, and then generating and using corresponding eye-movement patterns during testing, based on the actual observed scenes and the correlations extracted during training.

Other structures that can be used for the same purpose include networks trained by Hebbian rule with decay, or by reinforcement learning rules where appropriate values are attached to the inputs. Mean firing rate neurons may be considered in this framework, as well as spiking neurons [13] and hardware implementations [14] both for non-temporal and temporal neurons.

The above-mentioned networks will generally constitute modules of higher order architectures that select and combine the responses of the different modules, generating the required semantics. Hierarchical neural network architectures, including pyramidal schemes and constructive algorithms [12], which have shown capable of efficiently implementing higher order networks for invariant image recognition [10], [11], can be considered as a means for designing efficient implementations of this scheme.

Section 2 presents the proposed implementation scheme for capturing and using 'action'-based perceptual information in neural networks classifiers. Section 3 presents an experimental study illustrating the capabilities of the proposed algorithm, while results and suggestions for further research are given in the conclusions of the paper.

2. EFFECTIVE IMAGE RECOGNITION BY ACTION-BASED NEURAL NETWORKS

An environment of capturing and processing of images has been created, where a camera records images of the face reactions of one or more individuals that are observing and trying to recognize visual representations of objects or scenes. For purposes of simplicity, the visual representations to be recognized are stored in a computer system and the observer views them through a projection mechanism attached to the computer system which provides the magnification that is necessary for capturing local eye movements.

A subsystem has been created for processing the captured images of the observer's reactions. The main goal of this subsystem is the extraction of features, particularly, related to eye-movements during the perception procedure. These features are in the form of motion vectors computed from consecutive image frames. To compute such features, various preprocessing tasks are being performed, including image segmentation using morphological operators and transformation of resolution of the recorded images in order to reduce the volume of the information that is used next for recognition purposes [7].

A crucial aspect in the above frame-extraction procedure is the generation of 2-D image representations, of low dimensionality, of both visual input images and associated frames of the eye-movements obtained through a frame or video grabber. Multiresolution analysis is a technique that can efficiently reduce the size of these representations [15], by including as much as possible from the original image content in the approximate representations [8] even in the presence of additional noise [9].

Following the above, visual inputs are recorded together with the associated observer's eye movements over a short time period in which human perception is assumed to occur, and are subsequently processed to extract a sequence of 2-D motion vectors corresponding to eye movements. These images, together with the original images or scenes constitute the input to the neural network that will attempt to correlate them for more effective recognition purposes.

In this paper we propose to use recurrent neural networks as a means for handling the extracted information. At each instant the network receives at its input two images, i.e., the image to be recognized and a frame of the action/perception motion vector sequence. During the training phase, the network sets the structure and the weights of its connections in such a way so as to recognize the displayed object, while at the same time it learns to create and use, for this purpose, the sequence of the observer's eye-movements that have been extracted during the previous stage.

After training, and during the application phase, the network receives at its input the image or images to be recognized and starting from an initial motion vector image creates, based on the already acquired knowledge, the sequence of eye movement representations, which it also uses for recognition of the presented image or scenes.

The ability of recurrent networks to generate meaningful action/perception sequences, which when correlated with the original visual input, give the networks the possibility to converge to the desired stable states, is the crucial aspect examined in this paper.

The training of recurrent neural networks is a quite complex procedure. We have been using a variety of learning algorithms for this purpose, such as a temporal extension of backpropagation algorithm with or without adaptive delays [4], [6], as well as real time supervised learning algorithms [5]. These algorithms can be applied to networks with feedback and hidden neurons that have the capability of following dynamically evolving systems.

3. EXPERIMENTAL STUDY

The experiment that has been used to test the performance of the proposed approach was the T/C recognition problem, including deformed and corrupted by noise versions of the two letters. Characteristic cases that have been used are shown in Figures 1 and 2. These binary images consist of 256x256 pixels. Corruption by noise at 40%, for example, level is equivalent to erroneously changing 40 per cent of the image pixel values from 1 to 0 or 0 to 1. We set up the experiment described in the previous section and captured the corresponding eye movements of various subjects observing the

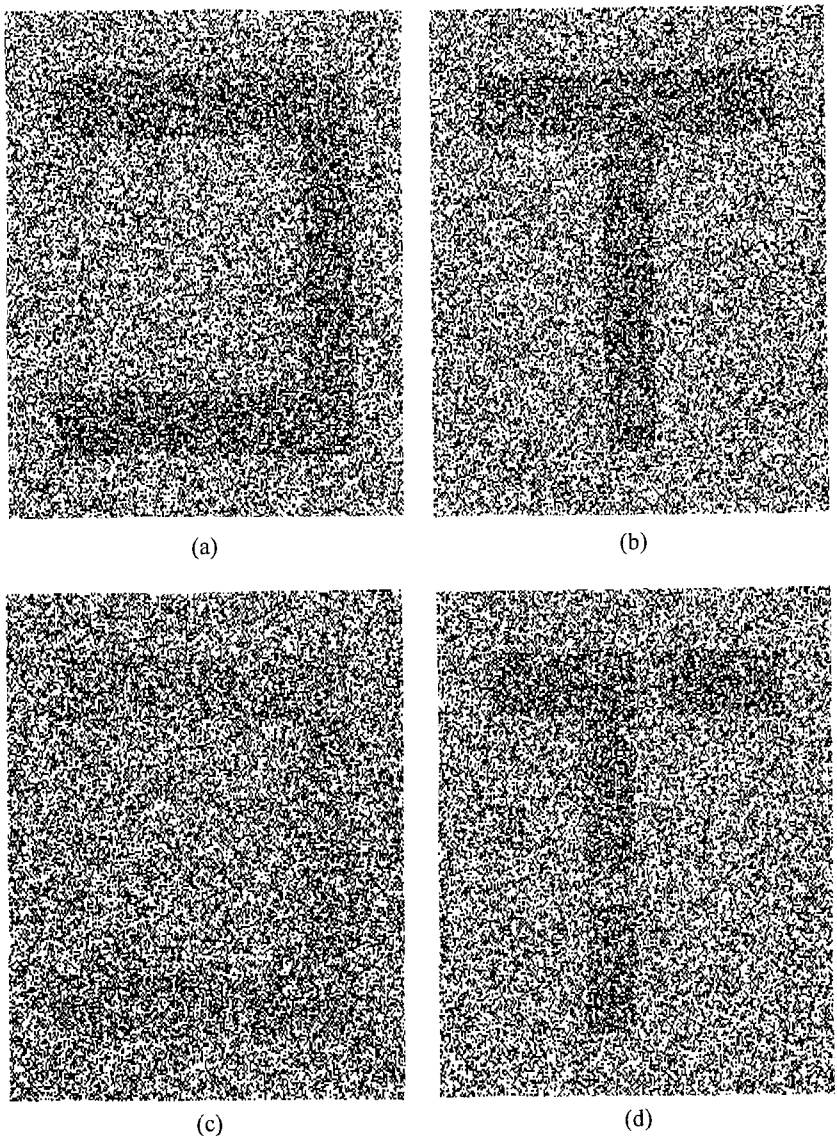
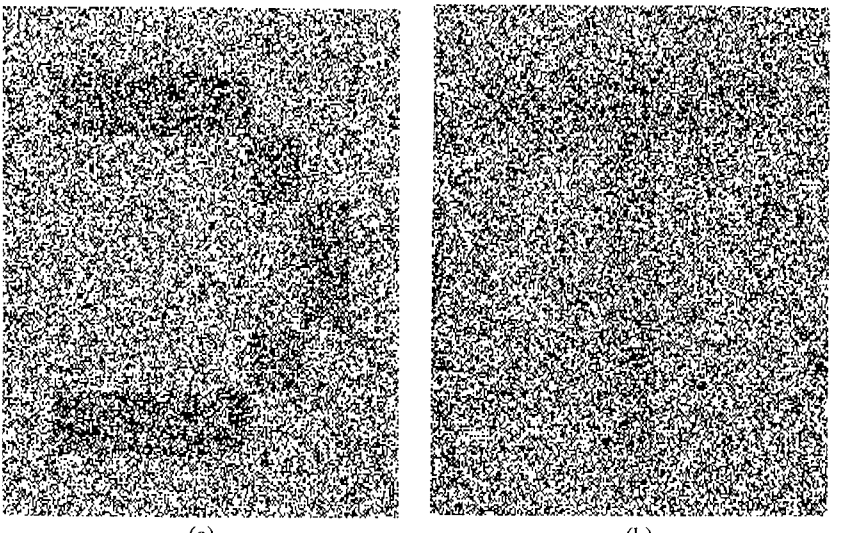


Figure 1: (a) letter C corrupted by 40% noise
(b) letter T corrupted by 40% noise
(c) letter C corrupted by 45% noise
(d) deformed letter T



(a) (b)

Figure 2: (a) deformed letter C
(b) letter T corrupted by 45% noise

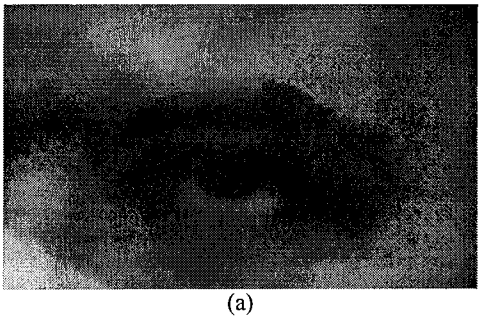
above letter representations. Figure 3 shows four characteristics scenes extracted from the sequence of the subject observing letter C corrupted by 45% noise that was shown in Figure 1. Figure 4 shows the corresponding sequence for deformed letter T shown in Figure 1. Using the above four frames of each image sequence we computed the center of gravity of the iris and then extracted the corresponding motion vectors between consecutive frames of each sequence. Table 1 shows the coordinates of the motion vectors for each case that was examined.

	C 40%	C 45%	C def.	T 40%	T 45%	T def.
fr. 1-2	8,1	11,2	6,1	-14,0	-17,0	-23,3
fr. 2-3	2,6	2,5	2,7	5,2	5,0	10,1
fr. 3-4	-8,1	-11,0	-7,0	1,6	0,7	-2,7

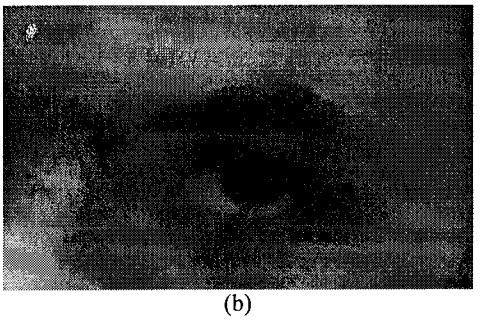
Table 1

Next we examined the ability of recurrent neural networks to perform the following tasks:

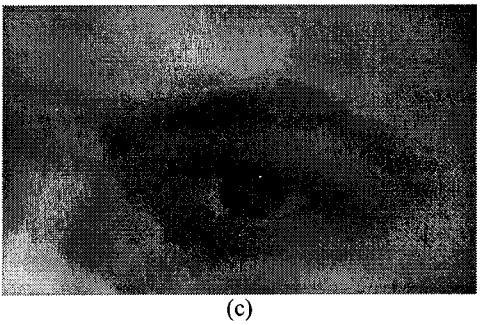
1. Using the letter representations as input images and each motion vector, represented as a 5x9 template of zeros and ones neuron states, as the system output the network learns to create (in the learning phase) and predicts (in the test phase) each motion vector from the temporally preceding one.



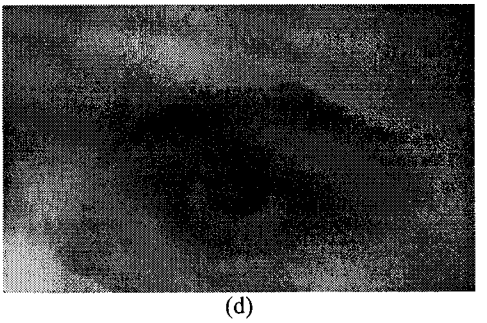
(a)



(b)

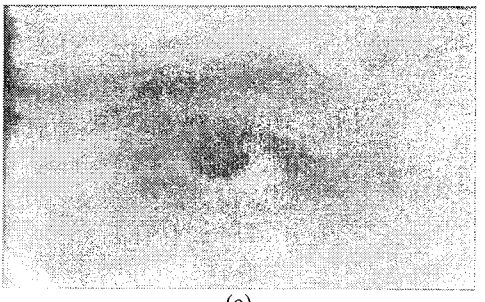


(c)

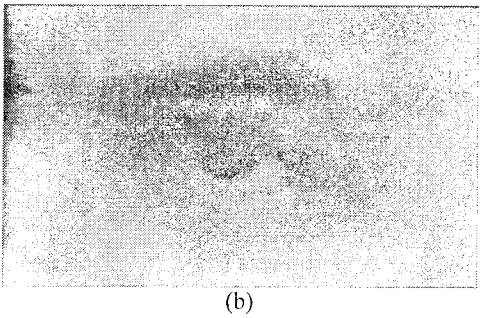


(d)

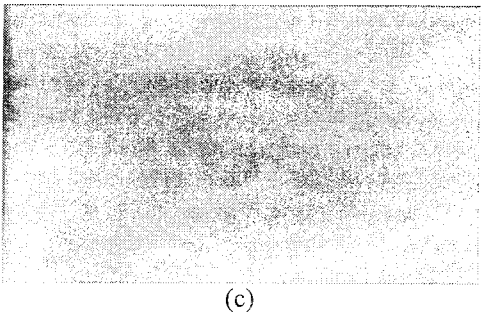
Figure 3: 'C' observing eye movement



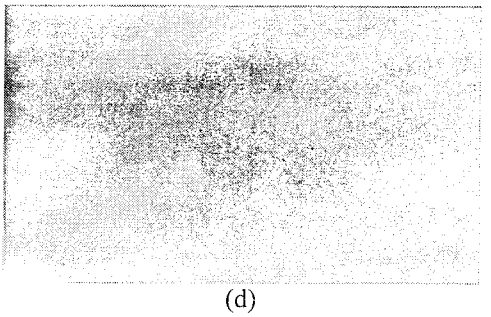
(a)



(b)



(c)



(d)

Figure 4: 'T' observing eye movements

2. Using the actual or predicted motion vectors and the letter representations at their inputs to classify correctly the latter ones.

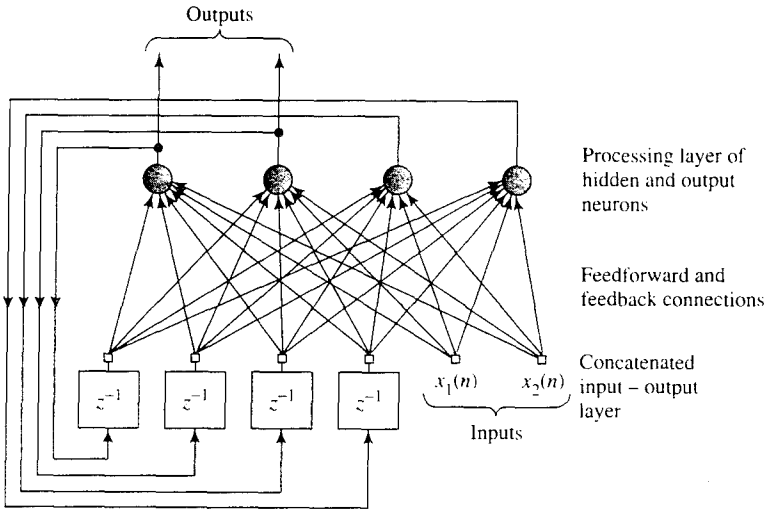


Figure 5: Architectural graph of a real-time recurrent network

We found most appropriate for this purpose the real time recurrent network shown in Figure 5 [6]. We performed three experiments with training and test letter representations (corresponding samples are the ones in Figures 1 and 2), Applying this procedure to motion vectors between frames 1-2, 2-3 and 3-4, we defined the network architectures, comprised of about 20 units in the hidden layer, which were adequate for predicting the motion vector sequence (task 1). We then used a feedforward multilayer network consisting of about 60 hidden units to make the final classification task , using the derived motion vector sequence. The results which we obtained in this experiment were very good, since almost all examined representations were correctly classified. For comparison purposes we applied a non-action-based, conventional similar backpropagation network for the same classification task. The network performance was very good in training, but rather poor in test operation. The letters shown in Figure 2 are one of the examples which the latter network failed to classify correctly, in contrast to the proposed one.

4. CONCLUSIONS

Neural network architectures have been used in this paper as a means to improve recognition of images corrupted by noise or deformed, based on motion vectors estimated from eye movements of subjects observing the scenes. Initial results were very encouraging. However, further work is needed to extend them to more complex real life scenes.

REFERENCES

[1] J.G. Taylor, "Relational Neurocomputing", presented at the IEEE Conference on Symbolic Computing, London, 1994.

[2] I. Rybak et al, "Modeling of a neural network for active visual perception and recognition", 12th Conf. Pattern Recognition, Israel, 1994.

[3] I. Otto et al, "Direct and Indirect Cooperation between Temporal and Parietal Networks for Invariant Visual Recognition", J. Cog. Neuroscience, 4, pp. 35-37, 1994.

[4] E.A. Wan, "Temporal Backpropagation: An efficient algorithm for finite impulse response neural networks", In Proceedings of the 1990 Connectionist Models Summer School, pp. 131-140, San Mateo, CA, 1990.

[5] R.J. Williams & J. Peng, "An efficient gradient-based algorithm for on-line training of recurrent network trajectories", Neural Computation 2, pp. 490-501, 1990.

[6] S. Haykin, "Neural Networks, a comprehensive foundation", Macmillan College Publishing Company, New York, 1994.

[7] W.K. Pratt, "Digital Image Processing", John Willey & Sons, 1991.

[8] A.Tirakis, A. Delopoulos and S. Kollias, "2-D Filter Bank Design for Optimal Reconstruction using Limited Subband Information", IEEE Trans. Image Processing, August 1995.

[9] A. Delopoulos and S. Kollias, "Optimal Filterbanks for Signal Reconstruction from Noisy Subband Components", Asilomar Conference, CA, 1994, also accepted for publication in IEEE Trans. on SP, 1995.

[10]S. Kollias, A. Stafylopatis and A. Tirakis, "Performance of Higher Order Neural Networks in Invariant Recognition", Neural Networks: Advances and Applications, pp. 79-108, North Holland, 1991.

[11]A. Delopoulos, A. Tirakis and S. Kollias, "Invariant Image Classification using Triple-Correlation-Based Neural Networks", IEEE Trans. Neural Networks, vol. 5, pp. 392-408, May 1994.

[12]S. Kollias, "Hierarchical Neural Networks for Invariant Image Recognition", Neurocomputing, 1995.

[13]G. Bugmann and J.G. Taylor, "A Stochastic Short-term Memory Using a pRAM Neuron and its Potential Applications" in Recent Advances in Neural Networks, R. Beale and M. Plumbley, eds, Prentice Hall, 1993.

[14]J.G. Taylor, "The pRAM: A New Learning Chip", in 1994 UK IT Forum, pp. 241-250, JFIT, 1994.

[15]S. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", IEEE Trans. on PAMI, vol 11, pp. 674-693, July 1989.

FEATURE-LOCKED LOOP AND ITS APPLICATION TO IMAGE DATABASES

Alex Sherstinsky and Rosalind W. Picard

Media Laboratory, E15-383
Massachusetts Institute of Technology, Cambridge, MA 02139
shers@media.mit.edu and picard@media.mit.edu

ABSTRACT

In this paper, we present a new dynamical system called the "feature-locked loop". The inputs to this feedback neural network are a set of feature vectors and a one-parameter function that characterizes the data. We show that the feature-locked loop is locally stable for one example of the characteristic function and determines the value of its unknown parameter. We apply this property of the feature-locked loop to the problem of sorting textures by their similarity. One known method for image sorting by similarity is to determine the Euclidean distance between the input feature vector and that of each texture in the database and then sort the resulting distances in the ascending order. While this procedure produces a set of similar images, it is prone to introduce outliers into this set. What this technique lacks is the measure of how similar the input image is to the reported set of images as a whole. We use the feature-locked loop and a priori information to quantify this degree of similarity. The prior knowledge is encoded in the form of the one-parameter function and a general assumption about the number of perceptual outliers in the reported set. The unknown parameter, computed by the feature-locked loop, is then related to the entire set of image features produced by the retrieval.

1. INTRODUCTION

Over the last several years, researchers in the image processing and the pattern recognition communities have been showing an increasing amount of interest in content-addressable image databases. Quantifying content, an inherently subjective entity, has the appeal of equipping large databases with efficient user-friendly search engines [1]. While the problem of searching text databases is well-understood, it is much more difficult to come up with a universal definition of "grep" for information contained within still images and video. Thus, making precise the notion of similarity is paramount in developing query-by-content algorithms.

Several models of similarity and the corresponding search approaches have been proposed by Picard *et al.* [1], [2]. For instance, one model considers two textures similar if the Euclidean distance between their representative features is "small". Defining features is a crucial step. Picard and Kabir report

This research was sponsored by Hewlett-Packard Laboratories, Palo Alto, California.

0-7803-2739-X/95 $4.00 © 1995 IEEE

that projections onto the principal components of the power spectrum make up feature vectors suitable for characterizing the Brodatz texture album [3]. Picard and Minka recognize the difficulty of working with a single model for similarity in natural scenes and develop a method for choosing the appropriate criteria from the "society of models" [2].

As part of a typical interaction with an image database the user chooses an icon and the system comes up with a sorted set of similar images, depending on the similarity measure. As noted above, there has been considerable effort in finding the right models for the given types of images (faces, textures, *etc.*) so that the perceptual similarities are reflected in the model representations. However, the issue of cross-validating the outcome of a search in the absolute sense has not been addressed. In other words, suppose that the query returns a sorted set of images that are the closest to the input image, according to some model. It is still unclear whether or not the closest of the set is close enough, or whether or not the output set as a whole is sufficiently close to the input.

The main contribution of this paper is the introduction of the feature-locked loop, a new iterative non-linear feedback neural network, and the demonstration of how its collective properties can be used to cross-validate the similarity of the ensemble of images to the input image. In other words, the algorithm provides the measure of closeness of the input image to the entire retrieved set of images in some absolute (e.g., perceptual) sense. This eliminates one drawback of the method of finding the Euclidean distance between the input feature vector and those of each member of the retrieved set and then using a threshold as the acceptance/rejection criterion, namely treating the output images individually and not as a set, which may produce in misleading results. For intance, if the query operation results in a large disparity in the Euclidean distance between feature vectors, then those that are accepted as "close" may be perceptually dissimilar to either one another, or the input texture, or both. The classification aspect of the feature-locked loop helps alleviate this difficulty. In addition, since this is a relaxation technique, it can provide a running evaluation of the image database retrieval process.

The rest of this document is organized as follows. Section 2 develops the feature-locked loop and analyzes some of its properties. Section 3 applies the feature-locked loop to the problem of quantifying the output of a content-based query. Section 4 summarizes the report.

2. FEATURE-LOCKED LOOP

Suppose that a model for image representation is given and consider N data points in that model's space, each represented by a normalized feature vector, $\vec{V}_i$, $i = 1, \ldots, N$. Likewise, the input texture is characterized by the normalized feature vector $\vec{W}$. Thus we have the set of the following normalized feature vectors: $\vec{W}, \vec{V}_1, \vec{V}_2, \ldots, \vec{V}_N$. Assume that the likelihood of each feature vector is determined by a PMF $p(\vec{y})$ of a discrete vector-valued random variable $\vec{y}$, which takes on the values $\vec{V}_1, \vec{V}_2, \ldots, \vec{V}_N$. In other words, $\vec{Y}_i$, $i = 1, \ldots, N$ are all the possible realizations of $\vec{y}$. We are interested in the PMF $p(\vec{x})$, where $\vec{x}$ is the new discrete vector-valued random variable, whose realizations, $\vec{X}_i$, include the presence of the feature vector for the input image, $\vec{W}$: $\vec{X}_i$, $i = 0, \ldots, N$, where $\vec{X}_0 = \vec{W}$, $\vec{X}_i = \vec{Y}_i$, $i = 1, \ldots, N$.

The loop operates by multiplying $\vec{X}_i$ for each model, i, by a quantity $0 < r_i < 1$, then summing all the $\vec{X}_i r_i$ contributions, and feeding the result

back. As an example, we will later consider the form of $r_i(\cdot)$ that depends on the distance between $\vec{X}_i$ and the quantity, $\vec{b}$, which is fed back [4].

2.1. LOOP ANALYSIS

We now analyze local stability properties and study the equilibrium state. Let $p(\vec{x})$ be defined as follows:

$$p(\vec{x}_i = \vec{X}_i) = \frac{\Pr(W)}{1 + \Pr(W)}, \; i = 0, \tag{1}$$

$$p(\vec{x}_i = \vec{X}_i) = \frac{p(\vec{y}_i = \vec{Y}_i)}{1 + \Pr(W)}, \; i = 1, \ldots, N, \tag{2}$$

where the event W stands for $\vec{x}_0 = \vec{W}$. Clearly, we have that $\sum_{i=0}^{N} p(\vec{x}_i = \vec{X}_i) = 1$.

The second equation confirms that since the input $\vec{w}$ is constantly injected into the loop, its contribution is always the same. For example, if $\Pr(W) = 1$, then the input image is weighed as much as the sum of the rest of the models' contributions.

From the description above, the operation of the loop is expressed as follows:

$$E(\vec{x}) = \vec{W}\frac{\Pr(W)}{1 + \Pr(W)} + \sum_{i=1}^{N} \vec{X}_i \frac{p(\vec{X}_i)}{1 + \Pr(W)}. \tag{3}$$

2.2. DYNAMICS VIA EXAMPLE

Recall that the features, $\vec{X}_i$, are normalized and, as an illustration, consider the following example, whose architecture is depicted in Figure 8):

$$p(\vec{y}_i = \vec{Y}_i) = A\exp\{-\|\vec{Y}_i - \vec{m}\|^2\}, \tag{4}$$

$$\sum_{i=1}^{N} p(\vec{y}_i = \vec{Y}_i) = 1; \tag{5}$$

$$\Pr(W) = 1, \tag{6}$$

and $\vec{m}$ is some parameter of $p(\vec{Y})$. Then:

$$E(\vec{x}) = \left(\frac{1}{2}\right)\left(\vec{W} + \sum_{i=1}^{N} p(\vec{x}_i = \vec{X}_i)\vec{X}_i\right) \tag{7}$$

$$= \left(\frac{1}{2}\right)\left(\vec{W} + \sum_{i=1}^{N} A\vec{X}_i \exp\{-\|\vec{X}_i - \vec{m}\|^2\}\right). \tag{8}$$

We see that here $r_i = A\exp\{-\|\vec{X}_i - \vec{m}\|^2\}$. Now $\vec{b} \stackrel{\text{def}}{=} \vec{m}$. As we iterate the loop with the iteration index n, we obtain:

$$\vec{b}_n = \left(\frac{1}{2}\right)\left(\vec{W} + \sum_{i=1}^{N} A_{n-1}\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}_{n-1}\|^2\}\right). \tag{9}$$

Note that the dependence of the normalization factor, A, on the iteration index, n, signifies that $p(\vec{Y})$ is re-normalized on every iteration.

2.2.1. LOCAL STABILITY

Assume that (9) has a fixed point. In other words:

$$\vec{b} = \left(\frac{1}{2}\right)\left(\vec{W} + \sum_{i=1}^{N} A\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\}\right) \tag{10}$$

is assumed to have a real solution. In order for this fixed point, $\vec{b}$, to be stable, the linearized version of the map must be stable as well [5]. Taking the derivative with respect to $\vec{b}$ of the right hand side gives:

$$\sum_{i=1}^{N} A\vec{X}_i{}^T(\vec{X}_i - \vec{b})\exp\{-\|\vec{X}_i - \vec{b}\|^2\} = \sum_{i=1}^{N} A\vec{X}_i{}^T\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\} \tag{11}$$

$$-\sum_{i=1}^{N} A\vec{b}^T\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\} = 1 - \sum_{i=1}^{N} A\vec{b}^T\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\}, \tag{12}$$

where in the last line, we made use of the fact that $\vec{X}_i{}^T\vec{X}_i = \|\vec{X}_i\|^2 = 1$.

The necessary (but not sufficient) condition for the convergence of (9) is its local stability, which requires (11) to be less than one in magnitude for any fixed point $\vec{b}$. In other words:

$$-1 < 1 - \textstyle\sum_{i=1}^{N} A\vec{b}^T\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\} < 1, \tag{13}$$

$$0 < \textstyle\sum_{i=1}^{N} A\vec{b}^T\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\} < 2. \tag{14}$$

Substituting (10) into (14) gives:

$$\sum_{i=1}^{N} A\vec{b}^T\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\} \tag{15}$$

$$= \left(\frac{1}{2}\right)\sum_{i=1}^{N} A\left(\vec{W}^T + \sum_{j=1}^{N} A\vec{X}_j{}^T \exp\{-\|\vec{X}_j - \vec{b}\|^2\}\right)\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\}$$

$$= \left(\frac{1}{2}\right)\sum_{i=1}^{N} A\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\}\sum_{j=1}^{N} A\vec{X}_j{}^T \exp\{-\|\vec{X}_j - \vec{b}\|^2\} +$$

$$\left(\frac{1}{2}\right)\sum_{i=1}^{N} A\vec{W}^T\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\}. \tag{16}$$

We now examine the two terms in (16). For the first term, we recall the triangle inequality and obtain:

$$\|E(\vec{x})\| = \|\sum_{i=1}^{N} A\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\}\| \tag{17}$$

$$\leq \sum_{i=1}^{N} \|A\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\}\| = \sum_{i=1}^{N} A\|\vec{X}_i\| \exp\{-\|\vec{X}_i - \vec{b}\|^2\} \tag{18}$$

$$= \sum_{i=1}^{N} A \exp\{-\|\vec{X}_i - \vec{b}\|^2\}\| = 1. \tag{19}$$

For the second term, we use (5) and the fact that $\|\vec{X}_i\|^2 = 1$ and $\|\vec{W}\|^2 = 1$, which assures that:

$$\|\sum_{i=1}^{N} A\vec{W}^T\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\}\| \leq \sum_{i=1}^{N} A\|\vec{W}^T\vec{X}_i\| \exp\{-\|\vec{X}_i - \vec{b}\|^2\} \quad (20)$$

$$\leq \sum_{i=1}^{N} A \exp\{-\|\vec{X}_i - \vec{b}\|^2\} = 1. \quad (21)$$

Therefore, we have:

$$\|\sum_{i=1}^{N} A\vec{b}^T\vec{X}_i \exp\{-\|\vec{X}_i - \vec{b}\|^2\}\| \leq \left(\frac{1}{2}\right) + \left(\frac{1}{2}\right) = 1. \quad (22)$$

Hence, (14) is satisfied and the system is stable. It is interesting to note that the system converged to a steady state in all computer simulations.

2.2.2. INTERPRETATION

The loop equations indicate that it computes the expectation of the feature vector, which describes an ensemble of points in the model space, including that corresponding to the input image. Moreover, in the case of convergence, this expectation becomes the value of the unknown parameter vector. Thus, the equilibrium value of the parameter vector is consistent with the requirements of a valid PMF.

It is important to indicate that the computed value of the feature vector $\vec{b} = \vec{m}$ can have the meaning of the mean of the Gaussian-like PMF, (4), if the sample size is large enough. In such cases, $\vec{b}$ is not "robust" against outliers. The task of remedying this shortcoming is a part of an on-going effort. The goal is to realistically estimate the amount of outliers in retrieved image sets for the particular class of the input image.

3. CROSS-VALIDATING SIMILARITY

The previous section establishes that the steady state of the feature-locked loop is a value (in the feature space) that describes an entire set of model images, subject to an external input. In this section, we use this property in order to attribute a confidence parameter to the result of querying a large image database. The iterative nature of the system allows it to serve as a real-time co-processor, dynamically evaluating the performance of the database.

3.1. PROGRAMMING THE LOOP

The feature-locked loop takes the following four inputs:

- The feature vector describing the input image. Here, we will demonstrate usage with the input image shown in Figure 1.
- Feature vectors for the set of images produced by the query operation. For example, the sixteen images in Figure 9.
- The weight of the input image, relative to the set of images resulting from the query operation. This parameter reflects the prior knowledge about the average number of perceptual outliers produced by the query operation.

- The PMF describing the prior knowledge about the points in the model space. In this study we use (4).

Upon convergence, the loop computes the expected value of the feature vector, which describes all the images in the similar set as well as the driving input. In addition, the loop yields (or "learns") the values of the PMF for each image in the set. These "firing" weights determine the amount each feature contributes to the value of the ensemble feature vector.

3.2. EXPERIMENTS

We use the Brodatz texture album [3] in our computer simulations. Eigenvectors of the covariance matrix of the power spectrum have been computed for the Brodatz
database [1]. Then feature vectors have been produced by retaining the highest 100 projections of the power spectrum of the given image onto these eigenvectors.

Using these feature vectors, we conduct three computer experiments involving the images in Figure 1 and Figure 9. The experiments differ by how much is known about perceptual outliers. In the first run, we assume that the retrieval step produces no outliers and thus set $\Pr(W) = 1$ (as in the example of Section 2. In the second run, we assume that outliers are frequent and set $\Pr(W) = 0$. In the third run, we assume an intermediate chance of outliers and set $\Pr(W) = \frac{1}{2}$. The motivation for assigning $\Pr(W)$ in this fashion is that a high weight on the input lowers the relative contributions of the images in the set, including those of the outliers. In contrast, a low weight on the input invites all images in the set to contribute to the value of the characteristic parameter, $E(\vec{x}) = \vec{b}$. For each experiment, we give two graphs:

- the feature vectors for the input image and $E(\vec{x})$, plotted on the same axes; and
- the relative contributions of all the textures, or $p(\vec{x})$.

The plots show that when the input weighting is high, the equilibrium feature vector is very close to the original. The contributions from the outliers are small. In fact, only the "brick" texture from the set contributes significantly, reflecting a high intolerance to deviations from the input texture. However, when the input is weighted the same as all the members of the set, the loop locks on to a collectively-determined equilibrium feature vector. In this particular case, the biggest contributions came from the high-frequency grass textures, and not from the textures that more resemble the input image. We observe that the difference between the input and the equilibrium feature vectors is large, meaning that the absolute degree of similarity is small. Finally, when the input weighting reflects a non-zero bias, a few outlier textures "fire" non-zero contributions, and the degree of similarity is intermediate.

During the normal operation of the loop, the input weight is held constant. If $\|\vec{W} - \vec{b}\|$ is small for the given input and the retrieved set, then the perceptual match is good. Conversely, if $\|\vec{W} - \vec{b}\|$ is large, then the choice of features for the database might need to be revised.

4. SUMMARY

We have presented the feature-locked loop, a novel non-linear dynamical system. The loop computes the expected value of a vector-valued random variable and thereby determines the underlying probability mass function with an

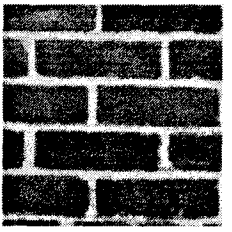


Figure 1: Input image.

unknown parameter, and that parameter itself upon reaching the equilibrium. This property is explored in quantifying the perceptual similarity between an input texture and the set of textures obtained by querying an image database. We show that given information about average performance of the query algorithm and knowledge about the distribution of the feature vectors in the database, the feature-locked loop provides a confidence measure for the result. Hence, the feature-locked loop can be employed in the design of built-in self-test mechanisms that flag inconsistencies in the retrieval algorithm.

5. REFERENCES

[1] R. W. Picard and T. Kabir, "Finding similar patterns in large image databases," in *Proc. ICASSP*, (Minneapolis, MN), pp. V–161–V–164, 1993.

[2] R. W. Picard and T. M. Minka, "Vision texture for annotation," *Journal of Multimedia Systems*, vol. 3, pp. 3–14, 1995.

[3] P. Brodatz, *Textures: A Photographic Album for Artists and Designers.* New York: Dover, 1966.

[4] A. S. Sherstinsky, "*M*-Lattice: A System For Signal Synthesis And Processing Based On Reaction-Diffusion," Tech. Rep. 300, M.I.T. Media Lab Perceptual Computing Group, May 1994. Sc.D. Thesis Report.

[5] S. H. Strogatz, *Nonlinear Dynamics And Chaos: With Applications To Physics, Biology, Chemistry, And Engineering.* Reading, MA: Addison-Wesley, 1994.

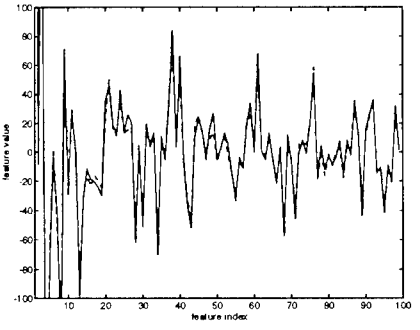


Figure 2: The input (solid line) and the equilibrium (dashed line) un-normalized features for the input weighting of 1.

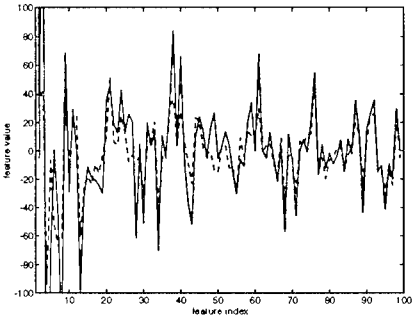


Figure 3: The input (solid line) and the equilibrium (dashed line) un-normalized features for the input weighting of 0.

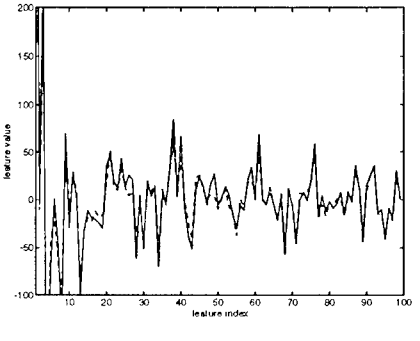


Figure 4: The input (solid line) and the equilibrium (dashed line) un-normalized features for the input weighting of $\frac{1}{2}$.

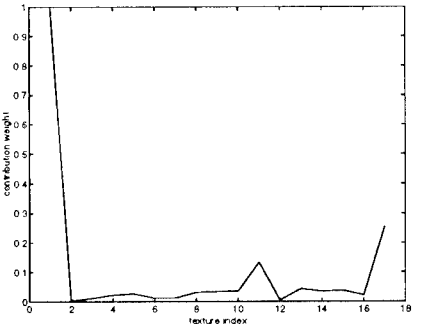


Figure 5: $p(\vec{x})$ for the input weighting of 1.

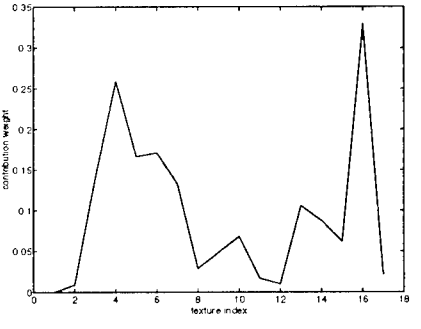


Figure 6: $p(\vec{x})$ for the input weighting of 0.

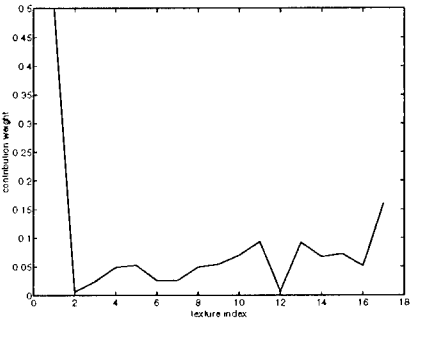


Figure 7: $p(\vec{x})$ for the input weighting of $\frac{1}{2}$.

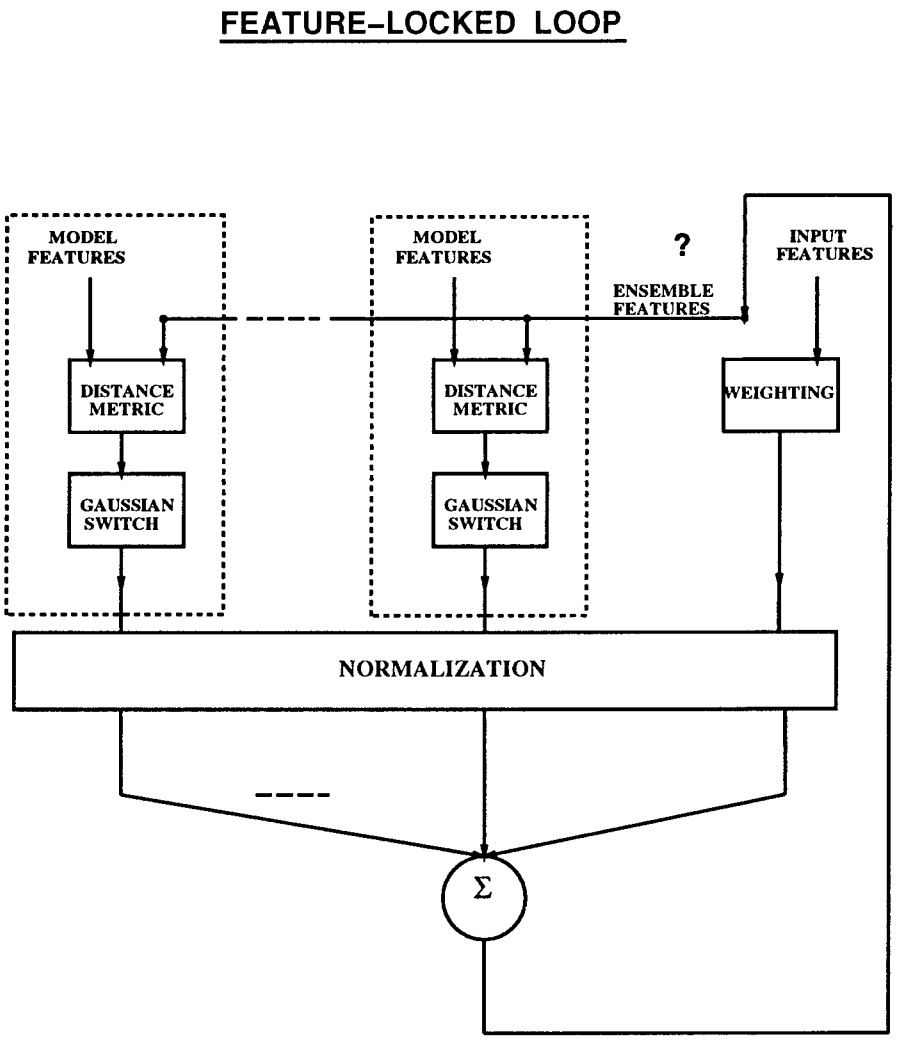


Figure 8: Feature-locked loop diagram for the example of Section 2. Here, the model features are the elements of $\vec{V}_i$, $i = 1, \ldots, N$; the input features are the elements of $\vec{W}$; the computed ensemble features are the elements of $E(\vec{x}) = \vec{b}$; the input weighting is $\Pr(W) = 1$; the distance metric is $d \stackrel{\text{def}}{=} \|\vec{X}_i - \vec{m}\|$; and the "Gaussian switch" is $\propto \exp\{-d^2\}$.

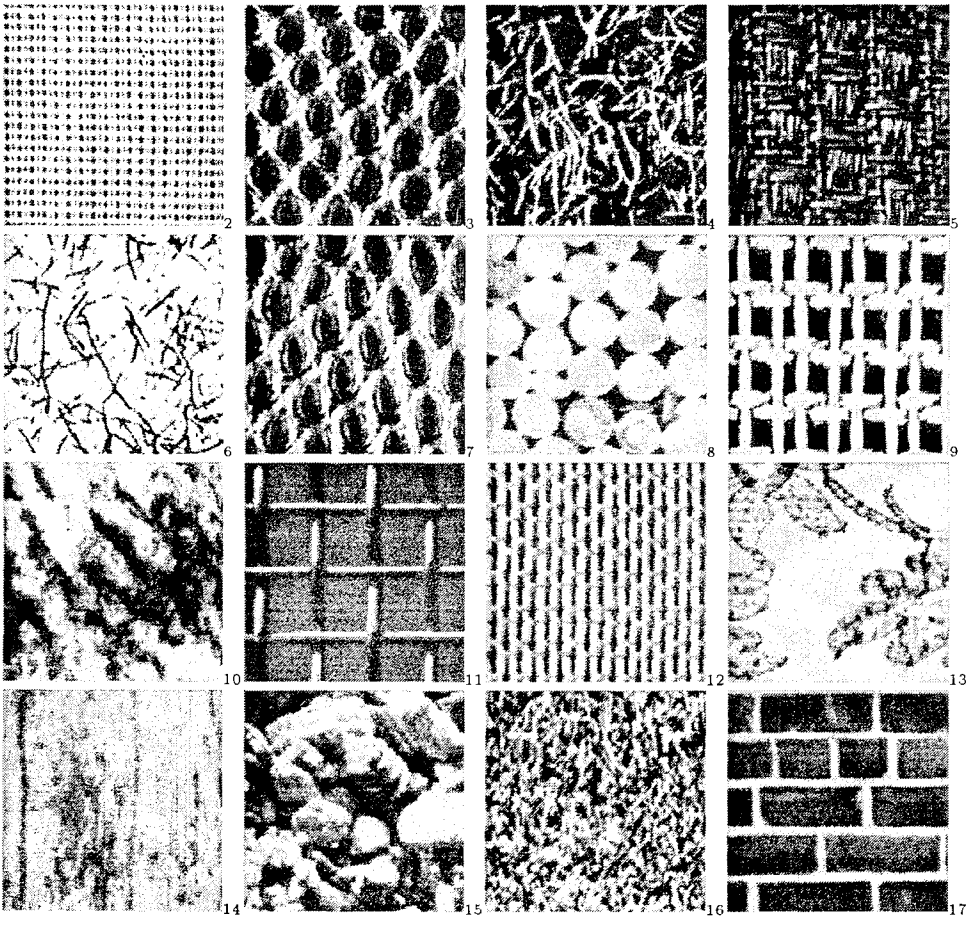


Figure 9: A set of sixteen images from the Brodatz album. The indices increase from left to right, top to bottom, from 2 to 17 (index 1 is reserved for the input image; this reflects the *Matlab* notation, which is offset by +1 from the notation used in the analysis).

AN ERROR DIFFUSION NEURAL NETWORK FOR DIGITAL IMAGE HALFTONING

Barry L. Shoop and Eugene K. Ressler
Department of Electrical Engineering and Computer Science
United States Military Academy
West Point, New York 10996

Abstract

A novel technique for digital image halftoning is proposed based on a symmetric error diffusion algorithm and a new form of artificial neural network. Using an *error diffusion neural network*, all pixel quantization decisions are computed in parallel and therefore visual artifacts resulting from the causality of the diffusion filter in classical error diffusion techniques are reduced and the resulting halftoned image quality is improved.

I. Introduction

Error diffusion is a common method of quantization in which an error associated with a particular point process is diffused within a local region and subsequent filtering methods employed in an effort to improve some performance metric such as the signal-to-noise ratio. In most applications to date, the error is the result of a nonlinear quantization process. In oversampled analog-to-digital (A/D) conversion, temporal error diffusion and oversampling techniques in conjunction with digital low pass filtering are used to achieve resolution in the range of 16- to 20-bits. In digital image halftoning, where a gray scale image is displayed or printed using only binary pixel values, the error is diffused in two spatial dimensions with the low pass filtering operation being performed by the human visual system.

The digital halftoning problem may be formally described as follows: given a continuous tone input image with intensity values in the interval $x_{m,n} \in [0,1]$, find a bilevel output image constrained to $y_{m,n} \in \{0,1\}$ such

0-7803-2739-X/95 $4.00 © 1995 IEEE

that some predefined distortion measure $d_{x,y}$ is minimized. Although the error diffusion technique can be shown to provide an optimum solution to this problem subject to a fidelity criterion, classical approaches suffer from implementation constraints. In conventional *unidirectional* error diffusion, the algorithm raster scans the image, and for each pixel, a binary quantization decision is made based on the intensity of the individual pixel and the weighted error from pixels within a diffusion region of previously processed pixels. As a result of this unidirectional processing, the diffusion filter is necessarily causal resulting in undesirable visual artifacts. In an effort to improve halftone image quality, Anastassiou [1] proposed the use of a frequency weighted mean square error distortion measure based on the frequency response of the human visual system. He showed that minimizing this metric was analogous to minimizing the energy function in a classical Hopfield-type neural network. This neural network approach provided a mechanism for symmetric error diffusion which resulted in improved halftone image quality.

In this paper, we begin by describing the theoretical foundations of error diffusion coding as applied to the digital halftoning problem. Next, we describe the problems associated with conventional unidirectional error diffusion in terms of visual artifacts in the image, memory requirements, and total convergence time of the halftoning process. We then introduce the concept of bidirectional or symmetric error diffusion as an optimum solution to the digital halftoning problem and describe the advantages associated with this approach. A neural network formalism of the halftoning problem is then presented based on a Hopfield-type neural network and a frequency-weighted mean square error distortion measure. We then introduce the fully-symmetric error diffusion neural network which directly implements the bidirectional error diffusion algorithm. Advantages, convergence and stability criteria, implementation issues, and performance metrics are then discussed. Halftoned images are presented throughout to describe the problems associated with conventional halftoning and the improvement achievable with this new neural network approach.

II. Error Diffusion

Figure 1 shows a block diagram of the error diffusion architecture. Here, $H(z_1, z_2)$ represents the two-dimensional z-transform of a causal, unity dc gain filter and $q[u_{m,n}] \in \{0, 1\}$. The unity gain criterion ensures that no amplification or attenuation of the error signal $\varepsilon_{m,n}$ occurs during the diffusion process. In this architecture, the error associated with the quantizer decision at location (m, n) is *diffused* within a local region $R_{m,n}$ to influence adjacent

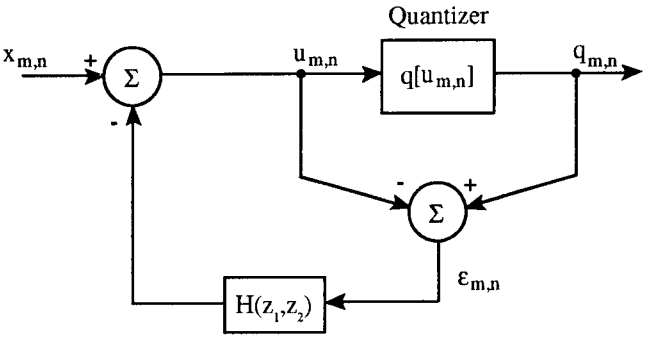


Figure 1: Block diagram of recursive error diffusion modulator.

quantization decisions. In the general case where $H(z_1, z_2)$ is assumed to be a finite impulse response (FIR) filter with coefficients $w_{i,j}$, the quantizer input state $u_{m,n}$ can be written as

$$u_{m,n} = x_{m,n} - \sum_{i,j \in R_{m,n}} w_{i,j} \varepsilon_{m-i,n-j}. \tag{1}$$

Here $w_{i,j} = 0$, for all $i = j$ since we are not considering *temporal* error diffusion in this application.

To mathematically quantify the effect of error diffusion, consider the relationship between the quantizer error $\varepsilon_{m,n} \equiv q[u_{m,n}] - u_{m,n}$ and the overall quantization error $\hat{\varepsilon}_{m,n} \equiv q[u_{m,n}] - x_{m,n}$ in the frequency domain. Assuming that the error is uncorrelated with the input and has statistical properties consistent with a white process, we can use z-transform techniques to show

$$G(z_1, z_2) \equiv \frac{\hat{E}(z_1, z_2)}{E(z_1, z_2)} = 1 - H(z_1, z_2). \tag{2}$$

Appropriate selection of $H(z_1, z_2)$ can spectrally shape the quantizer noise in such a way as to minimize the effect of the low-resolution quantization process on the overall halftoning process.

In unidirectional error diffusion, the image is processed in a raster fashion, proceeding from the upper left of the image to the lower right. As a result, the diffusion kernel $w_{i,j}$ is necessarily non-symmetric. Figure 2 shows two popular error diffusion kernels attributable to Floyd and Steinberg [2] and Jarvis, et al. [3]. The normalization factors ensure that the filter coefficients sum to one and therefore meet the unity gain criterion.

Figure 3 shows a 348×348 image of the Cadet Chapel at West Point which was halftoned using the filter coefficients shown in Figure 2(a). The original image was scanned at 150 dpi and then the halftoned image was printed using a 300 dpi laser printer. The unidirectionality of both the processing and the

$$\begin{array}{ccc} & \bullet & 7 \\ 3 & 5 & 1 \end{array} \times \frac{1}{16}$$

(a)

$$\begin{array}{ccccccc} & & & \bullet & 7 & 5 & \\ 3 & 5 & 7 & 5 & 3 & & \times \frac{1}{48} \\ 1 & 3 & 5 & 3 & 1 & & \end{array}$$

(b)

Figure 2: Error diffusion kernels for (a) Floyd and Steinberg and (b) Jarvis, et al. filters. The "•" represents the origin.

diffusion kernel result in undesirable visual artifacts in the halftoned image. These include directional hysteresis, which is manifested as "snakes" running from northwest-to-southeast, and transient behavior near boundaries, which appears as "shadows" below and to the right of sharp intensity changes. A logical conclusion to draw from this analysis is that if we could symmetrically diffuse the error and simultaneously process the entire image we could reduce some of these visual artifacts and therefore improve the overall halftoned image quality.

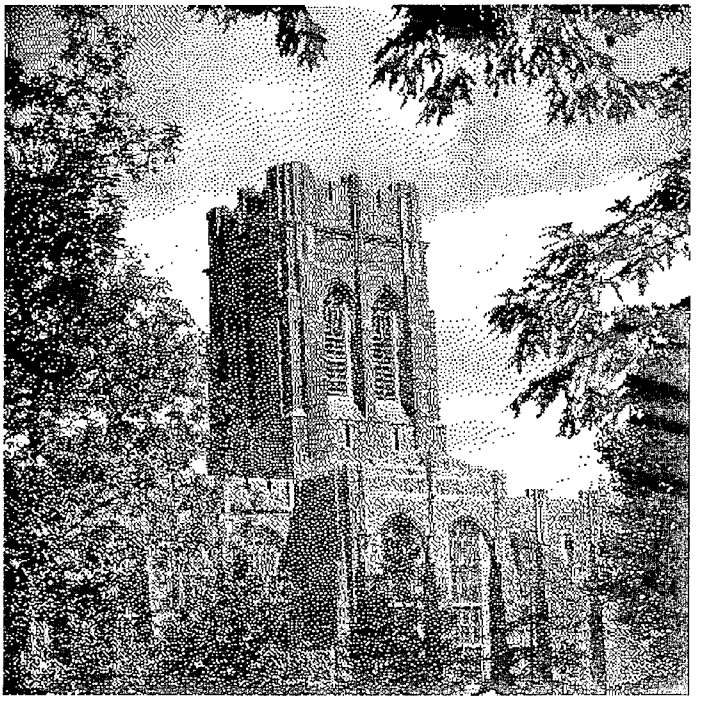


Figure 3: Halftoned image of the Cadet Chapel using Floyd-Steinberg weights.

III. Neural Network Formalism

Artificial neural networks and their application to image halftoning have received considerable attention lately [4][5]. This popularity lies in their ability to minimize a particular metric associated with a highly nonlinear system of equations. In the ideal case, a single neuron in the network is interconnected to every other neuron in the network with a weighting determined *apriori*. The output of each neuron is then a nonlinear transformation of the weighted sum of all of the inputs where the nonlinearity usually takes the form of a monotonic sigmoid function. The dynamic behavior of the network then causes the system to stabilize to one of the minima of a well-defined energy function.

A. The Hopfield-Type Neural Network

The dynamic behavior of an N-neuron Hopfield-type neural network can be described by the following system of N nonlinear differential equations:

$$c\frac{\mathrm{d}u_i(t)}{\mathrm{d}t} = -u_i(t) + \sum_j w_{i,j}\mathcal{F}[u_j(t)] + x_i, \tag{3}$$

where $i = 1, 2, ...N$, $\mathcal{F}[\cdot]$ is a monotonically increasing sigmoid function, x_i is an input N-vector, and c is a scaling factor. In equilibrium, Equation (3) implies

$$u_i = x_i + \sum_j w_{i,j}\mathcal{F}[u_j]. \tag{4}$$

Hopfield showed that when the matrix of interconnection weights $\mathbf{W}$ is symmetric with zero diagonal elements and the high-gain limit of the sigmoid $\mathcal{F}[\cdot]$ is used, the stable states of the N functions $y_i(t) = F[u_j]$ are the local minima of the energy function [6]

$$E = -\frac{1}{2}\mathbf{y}^T\mathbf{W}\mathbf{y} - \mathbf{x}^T\mathbf{y} \tag{5}$$

where $\mathbf{y} \in \{0, 1\}$ is an N-vector of quantized states. Here $\mathbf{W}$ is an $N \times N$ circulant matrix derived from the original error diffusion weights $w_{i,j}$. Figure 4 shows an electronic implementation of a four-neuron Hopfield-type neural network. Here the individual neurons are represented as amplifiers (standard and inverting) and the synapses by the physical connections between the input and output of the amplifiers. Resistors are typically used to make these connections.

Anastassiou [1] argued that Equation (1) corresponded to a unidirectional feedforward neural network if the nonlinearity $\mathcal{F}[u_j]$ is replaced with the non-monotonic quantizer error $\varepsilon_n(u_n)$. He showed that a frequency weighted distortion measure could equivalently be expressed as the energy function of a

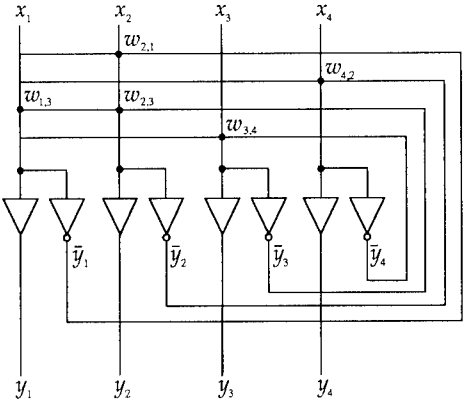


Figure 4: Electronic realization of a four-element Hopfield-type network.

$$\begin{array}{ccccc} 1 & 3 & 5 & 3 & 1 \\ 3 & 5 & 7 & 5 & 3 \\ 5 & 7 & \bullet & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{array} \quad \times \frac{1}{96}$$

Figure 5: Symmetric error diffusion kernel $w_{i,j}$ based on the Jarvis, et al. filter. The "•" represents the origin.

symmetric Hopfield-type neural network where the frequency weighting came from the response of the human visual system. By selecting an appropriate symmetric diffusion kernel and using the coefficients as the interconnection strengths for the neural network, a fast massively parallel analog implementation of the error diffusion algorithm could be realized which minimized the frequency weighted mean squared error. One method of generating a symmetric diffusion kernel is to include the coefficients of a causal kernel in the anticausal portion of the matrix. This is shown in Figure 5 for a 24-neighbor symmetric version of the filter shown in Figure 2(b). The corresponding halftoned image is shown in Figure 6. Comparison of Figure 6 and Figure 3 shows that the directional artifacts which are characteristic of unidirectional error diffusion are significantly reduced. Also, symmetric error diffusion provides increased detail in the halftoned image. This can be seen in the fine detail in the leaves and the edges of the chapel structure.

B. The Error Diffusion Neural Network

The results of the previous section coupled with the architecture of Figure 1 and the spectral noise shaping described by Equation (2) can be formulated

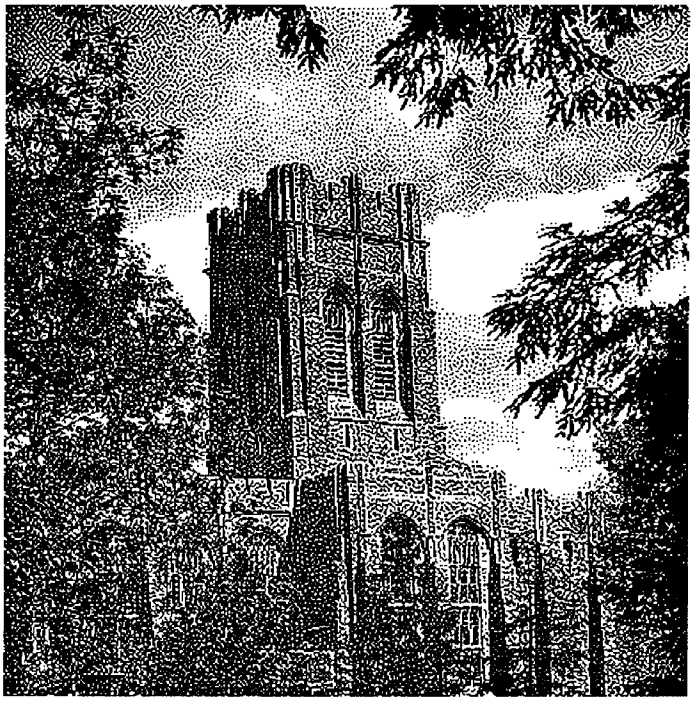


Figure 6: Halftoned image using symmetric Jarvis weights.

in the context of a symmetric *error diffusion neural network*. Here, $H(z_1, z_2)$ is designed to spectrally shape the quantization noise spectrum according to some predefined criteria.

In equilibrium the error diffusion neural network satisfies

$$\mathbf{u} = \mathbf{W}(\mathbf{y} - \mathbf{u}) + \mathbf{x}, \tag{6}$$

which results in an equivalence to the Hopfield network which can be described by

$$\mathbf{u} = (\mathbf{A}\mathbf{x}) + (-\mathbf{A}\mathbf{W}\mathbf{y}), \tag{7}$$

where $\mathbf{A} = (\mathbf{I} - \mathbf{W})^{-1}$. Effectively, the error diffusion network includes a pre-filtering of the input image $\mathbf{x}$ by the matrix $\mathbf{A}$ while still filtering the output $\mathbf{y}$ by a new matrix $\mathbf{W}' = -\mathbf{A}\mathbf{W}$. The energy function of the error diffusion neural network can be written as

$$E = -\frac{1}{2}\mathbf{y}^T\mathbf{W}\mathbf{y} - \mathbf{x}^T\mathbf{y} \tag{8}$$

where $\mathbf{y} \in \{0, 1\}$ is again an N-vector of quantized states. Convergence of the error diffusion network is guaranteed if

$$\forall\ k: \quad \left[-(\mathbf{I} - \mathbf{W})^{-1}\mathbf{W}\right]_{k,k} \geq 0, \tag{9}$$

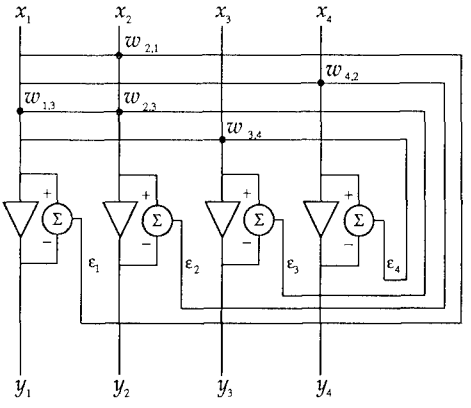


Figure 7: Electronic realization of a four-element error diffusion network.

or equivalently

$$(\mathbf{I} - \mathbf{W})^{-1} \leq 1. \tag{10}$$

Figure 7 shows an electronic implementation of a four-neuron error diffusion-type neural network using similar hardware as in Figure 4. Figure 8 shows the same 348×348 image of the Cadet Chapel halftoned using our new artificial neural network. There is clear improvement in the halftoned image quality over the images shown in Figures 3 and 6. Notice particularly the uniform distribution of pixels in the cloud formation in the upper left of Figure 8 compared to Figure 6. Also noteworthy is the improvement around the fine detail portions of the tree branches and next to the vertical edges of the chapel.

One of the advantages of the error diffusion neural network over the Hopfield-type network for digital halftoning is full connectivity. The matrix $\mathbf{W}$ used by Anastassiou is a sparse circulant matrix and as a result, complete diffusion of the error across the entire image is not achieved. The matrix $\mathbf{AW}$ in the error diffusion network, however, is a full-rank matrix and therefore complete diffusion of the error is achieved. Another advantage of the error diffusion architecture is that it provides a more direct method of calculating the coefficients of the diffusion kernel based on linear filtering and spectral noise shaping techniques.

In this application, the feedback filter for the error diffusion neural network was designed using conventional two-dimensional filter design techniques. Circular symmetry of the frequency response of this filter is critical because of the sensitivity of the human visual system to directional artifacts like those present in Figure 3. A two-dimensional FIR low pass filter with a kernel size of 11×11 was designed using windowing techniques. The impulse response

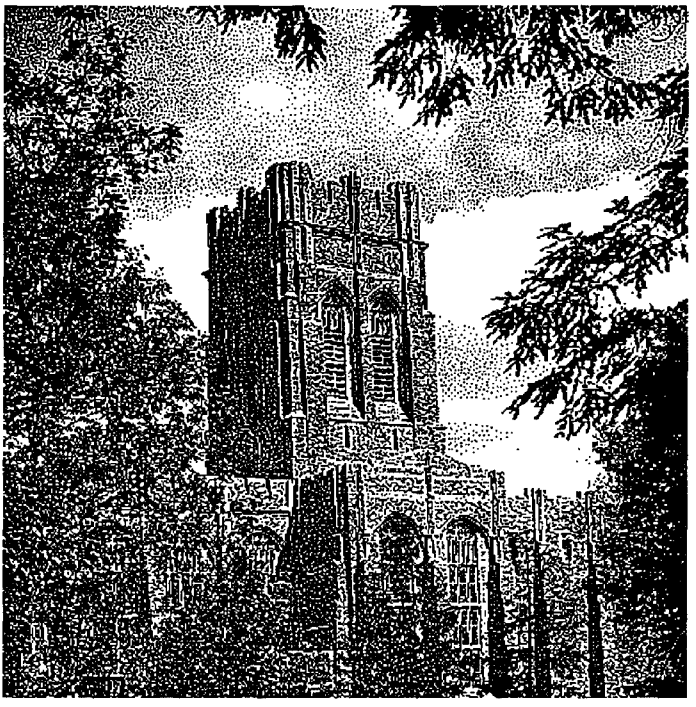


Figure 8: Halftoned image using the error diffusion neural network.

of the ideal low pass circularly symmetric filter is

$$h(n_1, n_2) = \frac{R}{2\pi} \frac{J_1\left(\sqrt{n_1^2 + n_2^2}\right)}{\sqrt{n_1^2 + n_2^2}}, \tag{11}$$

where $0 \leq R = 0.43\pi \leq \pi$ is the cutoff frequency, $J_1(\,)$ is a bessel function of the first-kind, and n_1 and n_2 are the two spatial dimensions. The window used was a Kaiser window with parameter α

$$w_k(n_1, n_2) = \begin{cases} \frac{I_0\left[\alpha\sqrt{1-((n_1^2+n_2^2)/25)}\right]}{I_0^2[\alpha]}; & n_1^2 + n_2^2 \leq 25 \\ 0\ ; & \text{otherwise} \end{cases} \tag{12}$$

where I_0 is a modified bessel function $I_0 = J_0[j\alpha]$. This approach to the error diffusion filter design was motivated by the spectral analysis of the halftoned image analysis in [7].

IV. Summary and Observations

In this paper, we have presented a novel method for digital image halftoning based on a symmetric version of the error diffusion algorithm and an error

diffusion neural network. This symmetric error diffusion approach provides a means of improving halftoned image quality while reducing computational complexity and storage requirements. A significant advantage of this new implementation is that arbitrary size and shape diffusion kernels can be implemented thereby reducing the visual artifacts which have become characteristic of halftoned images. This can be accomplished with no penalty in terms of computation speed. The error diffusion neural network computes the halftoned image asymptotically faster than a conventional Hopfield-type neural network. The conventional network requires $\sim O(N^2B)$ time per iteration to compute an image of $N \times N$ pixels with a diffusion support region of B pixels. The error diffusion neural network implementation can perform the same iteration in $O(1)$ time.

V. Acknowledgements

This research was supported by the Army Research Office.

References

[1] D. Anastassiou, "Error diffusion coding for A/D conversion," *IEEE Trans. Circuits Syst.*, vol. 36, no. 9, pp. 1175–1186, 1989.

[2] R. Floyd and L. Steinberg, "An adaptive algorithm for spatial gray scale," *SID 75 Digest*, vol. 36, pp. 35–37, 1975.

[3] J. F. Jarvis, C. N. Judice, and W. J. Ninke, "A survey of techniques for the display of continuous-tone pictures on bilevel displays," *Computer Graphics and Image Processing*, vol. 5, pp. 13–40, 1976.

[4] R. Geist, R. Reynolds, and D. Suggs, "A markovian framework for digital halftoning," *ACM Trans. Graph.*, vol. 12, no. 2, pp. 136–159, 1993.

[5] K. R. Crounse, T. Roska, and L. O. Chua, "Image halftoning with cellular neural networks," *IEEE Trans. Circuits Syst.*, vol. 40, no. 4, pp. 267–283, 1993.

[6] D. W. Tank and J. J. Hopfield, "Simple "neural" optimization networks: A/D converter, signal decision circuit, and linear programming circuit," *IEEE Trans. Circuits Syst.*, vol. 33, no. 5, pp. 533–541, 1986.

[7] R. A. Ulichney, *Digital Halftoning.* Cambridge, Massachusetts: MIT Press, 1987.

Applications and Implementations

Estimation of the Glucose Metabolism from Dynamic PET-scans using Neural Networks.

Claus Svarer, Ian Law, Søren Holm, Niels Mørch, Olaf Paulson
Dept. of Neurology,
The National University Hospital
DK-2100 Copenhagen Ø, Denmark
email: csvarer@ei.dtu.dk

Lars Kai Hansen, Torben Fog
CONNECT, Electronics Institute, B349
Technical University of Denmark,
DK-2800 Lyngby, Denmark

Abstract **– A method for fast pixel by pixel estimation of the Glucose Metabolism in the brain using the tracer [^{18}F]flourodeoxyglucose in dynamic PET-scan data is described. A neural network is trained to estimate the glucose metabolism on data generated by direct fitting of the rate constants in Sokoloff's model. The generalization ability of the neural network is tested on data from subjects not included in the training set. This method can be used to estimate changes of the metabolism in different brain regions for subjects with serious brain disorders. By using the neural estimation procedure the processing time for a brain scan volume is reduced from 48 hours to 4 minutes!**

INTRODUCTION

The Positron Emission Tomography (PET) technique is an important tool for mapping the brain metabolism and functionality [3]. The primary aim of PET is reconstruction of concentrations of certain radioactive tracers in the brain. Useful tracers emit positrons that are locally annihilated to produce two 511 keV gamma rays propagating in opposite directions. The 3D distribution of the tracer can be reconstructed from the geometric constraints of coincident counts, using standard techniques (filtered backprojection). An important class of tracers are chemically equivalent to substrates of the basic brain metabolism. By reconstructing such tracer distributions important aspects of brain metabolism have been revealed. Furthermore, by investigating the *transient response* to a tracer injection, it is possible to identify fundamental kinetic rate constants on a set of dynamic PET images. The estimation of

0-7803-2739-X/95 $4.00 © 1995 IEEE

the glucose metabolism in the present study has been applied on the Sokoloff model [6]. To gain maximal spatial resolution of the rate constant distribution we will compute the estimate on a pixel by pixel basis as originally described in the work of Kanno et al. [5].

We demonstrate the ability of a neural network to substitute for tedious parameter fitting procedures. The neural network system is trained to produce a smooth map relating a given transient pixel activity to a key parameter of metabolism, viz. the glucose utilization rate R. By identification of the inverse kinetics the estimate of the spatial distribution of R may be obtained three orders of magnitude faster than by direct fitting of the kinetic model.

The work described in this article is a continuation of our work presented in the proceedings of NNSP'94 [1]. In particular we show that the neural net scheme can *generalize*, i.e., that the inverse kinetics map trained on data from one set of subjects may be used for interpretation of data from other test subjects.

SOKOLOFF's KINETIC MODEL

In the present work we are considering the kinetics of the compound [^{18}F]flourodeoxyglucose (FDG). The kinetics of this tracer are similar to glucose in the initial phases of metabolism. It passes through the blood-brain barrier (BBB), and is phosphorylized intracellularly in a process analogous to glucose. The phosphorylized [^{18}F]flourodeoxyglucose compound does not enter into the Krebs cycle of glucose metabolism therefore it is effectively trapped. The kinetics can be modeled by a compartemental model involving one compartment representing the tracer density in the arterial blood outside the BBB, C_{P}^{*}; one compartment representing the so-called *precursor pool*, C_{E}^{*}; and finally a compartment representing the intracellular phosphorylized fraction behind the BBB, C_{M}^{*}; see figure 1. In current experiments the arterial concentrations are measured continuously with the scans.

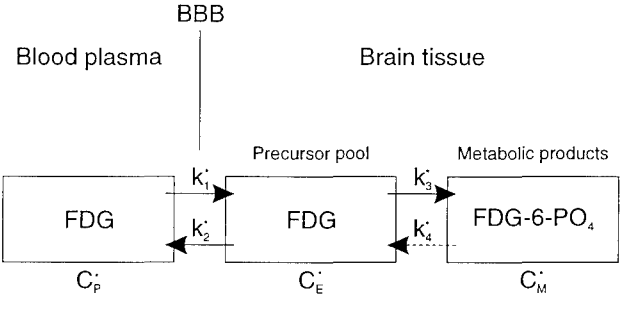


Figure 1: Sokoloff's three compartment model applied to phosphorylization of [^{18}F]flourodeoxyglucose (FDG). The star on the concentrations signifies that we consider tracer amounts and constants.

Following the injection of the tracer, hence, the rise of the arterial blood concentration C_{P}^{*}, the flow through the BBB starts. The measured PET tracer activity is the sum of the activities of the two compartments to the right of the BBB in figure 1 and a certain fraction, V_P^*, of the arterial blood concentration. This can be calculated as

$$C_i^* = C_E^* + C_M^* + V_P^* C_P^*. \tag{1}$$

The dynamics of the three compartment model is given by:

$$\frac{dC_i^*}{dt} = \frac{dC_E^*}{dt} + \frac{dC_M^*}{dt} + V_P^* \frac{dC_P^*}{dt} \tag{2}$$

with

$$\frac{dC_M^*}{dt} = k_3^* C_E^*, \tag{3}$$

and

$$\frac{dC_E^*}{dt} = k_1^* C_P^* - k_2^* C_E^* - k_3^* C_E^*. \tag{4}$$

The reverse reaction rate constant k_4^* corresponding to k_3^* is neglected; it is not identifiable within the measurement time of the present experimental setup.

These linear differential equations describing the response of the activity of the arterial blood ($C_P^*(t')$) are straightforward to integrate yielding the two time dependent concentrations,

$$C_E^*(t) = k_1^* e^{-(k_2^*+k_3^*)t} \int_0^t e^{(k_2^*+k_3^*)t'} C_P^*(t')\ dt', \tag{5}$$

and

$$C_M^*(t) = k_1^* k_3^* \int_0^t \left[e^{-(k_2^*+k_3^*)t'} \int_0^t e^{(k_2^*+k_3^*)t''} C_P^*(t'')\ dt'' \right] dt'. \tag{6}$$

Following injection these solutions describe the transient activity in terms of the measured $C_{\mathrm{P}}^{*}(t)$, the three rate constants (k_1^*, k_2^*, k_3^*) and the plasma volume fraction V_{P}^{*}. Conversely, for a given transient $C_{\mathrm{P}}^{*}(t)$ and for given measured sum of concentrations $C_{\mathrm{i}}^{*}(t)$ we may fit the three rate constants and the plasma volume fraction. We use a simple least squares cost function for the fit, hence implicitly assuming Gaussian residuals. Optimization over the four parameters $(k_1^*, k_2^*, k_3^*$ and $V_{\mathrm{P}}^{*})$ is carried out using a second order Newton scheme[1]

From these parameters we compute the important glucose utilization parameter R

[1]Based on the solution to the kinetic model it is straightforward to compute the second order derivatives.

$$R = \frac{c_{gl}}{LC} \frac{k_1^* k_2^*}{k_2^* + k_3^*} \tag{7}$$

where c_{gl} is the glucose concentration measured in the blood for each subject and LC the so-called *lumped constant* for the model. We use the value ($LC = 0.55$) [4].

Two different approaches have been used when estimating the kinetic constants. In [6, 3] it is assumed that the parameters are homogeneous in regions and therefore can be fitted using the region averaged activity. Alternatively the rate constants can be fitted at a pixel by pixel basis as proposed in [5]. However, this approach has not found widespread use since it is rather tedious to fit the kinetic model in all pixels.

The database used for these experiments are PET data collected at the PET center at Rigshospitalet, Copenhagen, and stems from 10 normal subjects. Data are acquired on a GE4096 plus (General Electric Medical Systems), sampling 15 slices simultaneously. The dynamic scans after injection of 200 MBq F-18 labeled FDG are performed over 60 minutes, providing 34 contiguous time frames of increasing duration in order to provide a reasonable sampling of the C_i^* curve (10@6 sec; 3@20 sec; 8@60 sec; 5@120 sec; 8@300 sec). Two examples of input blood curves are shown in figure 2.

The images were reconstructed in $128x128$ matrices ($2mm^2$ pixels) by standard Filtered backprojection (Ramp filter with Hann window). Correction for attenuation was based on a separate transmission scan with a rotating Germanium pin source. For further introduction to PET scan techniques see e.g., [3]

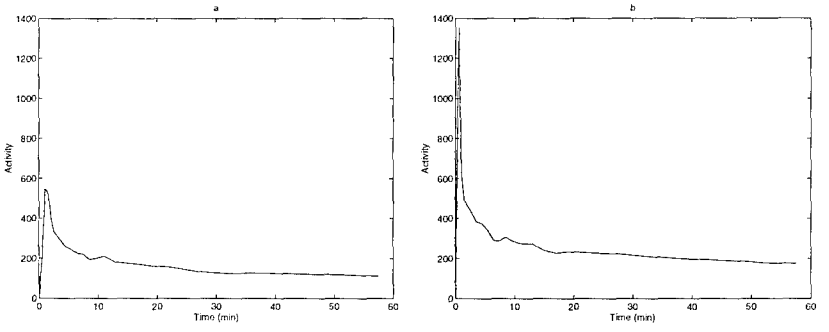


Figure 2: Input blood curve as a function of time for two different subjects. Note the large variation in the shape (value and time for maximum) of the blood curves for different subjects.

In figure 3 a fit of the three rate-constants and the calculated glucose utilization R on a pixel by pixel basis in one slice is shown. We note that these rate constant pictures are rather noisy, reflecting the noise level of the

original pixel activity curves also shown in figure 4. To provide less noisy data for the neural net training procedure we have averaged and subsampled so that a "superpixel" in the new image is the average of 4×4 pixels of the original image. Another reason for subsampling is that it takes approximately 48 hours (on HP9000/735 workstation) to fit the parameters in Sokoloff's model on 1 slice, 128×128, image, using a Newton based method. A full brain volume consist of 15 slices.

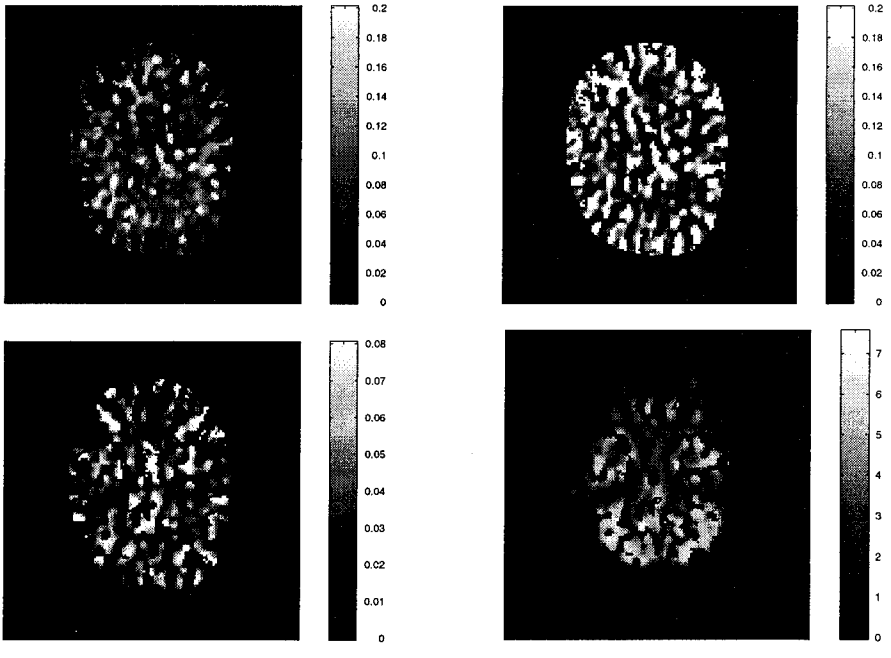


Figure 3: Parameters in Sokoloff's model fitted on a pixel by pixel basis, Upper-left: K_1^*, Upper-right: K_2^*, Lower-left: K_3^*, and Lower-right: R^*. R is calculated based on equation (7). Especially, the rateconstants K_2^* and K_3^* are very noisy.

To avoid the tedious fitting procedure we here investigate the possibility of identifying the *inverse model* of the kinetics: we search for a *map* that provides an estimate of the glucose utilization R in each pixel [1]. Our basic vehicle will be a simple feed-forward network.

The input to the neural network is the measured activity of a given pixel. Futhermore, we saw in figure 2, that the input blood curve (C_p^*) varies considerably between subjects due to variations in the way doses were injected and a individual blood circulation. Hence, we need to provide this information to the neural network model if we want it to be able to generalize from one subject to another. C.f. (7) the total blood glucose concentration is also provided as an input to the neural network. This concentration can be

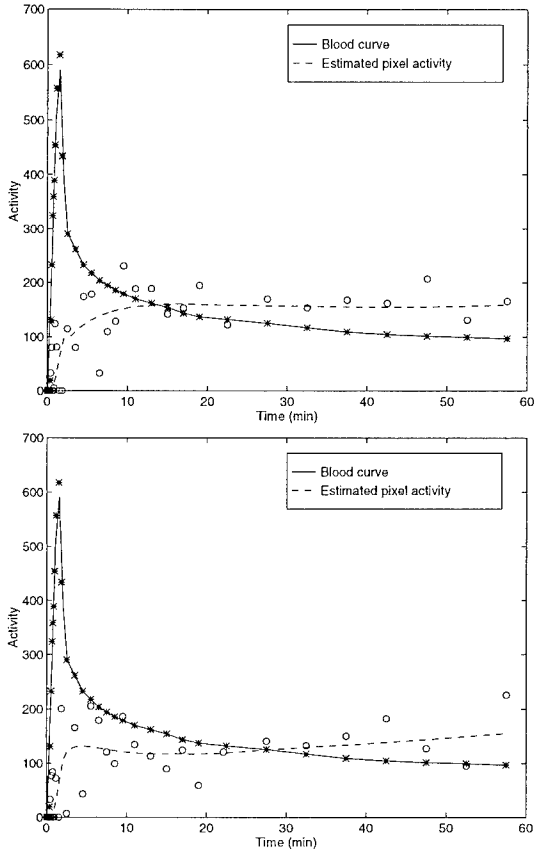


Figure 4: Result of simulating the Sokoloff model in two different pixels within the brain. The measured activity data for a single pixel, shown by 'o's, is very noisy, noise levels are highest for the first data points where the measuring time is only 5-10 seconds. The measured blood curve data is shown by '*'s. The dashed curve shows the result of a simulation using the estimated model.

considered constant during a scan. This gives us a total of 69 inputs to the neural network: 34 inputs describing the transient activity in the given pixel, another 34 measurements of the activity in the blood (the blood curve), and 1 measurement of the glucose concentration in the blood.

The fact that we have subsampled the data for the training set does not prevent us from using the network on the individual pixels in the test set.

NETWORK DESIGN

The feed-forward neural network used comprises 6 hidden units using *tanh* activation functions and a linear output unit. The network is trained using a pseudo-Gauss Newton method (a diagonal approximation to the Hessian matrix is used) as described in [7], and pruned using the Optimal Brain Damage method described in [2] to minimize computation and to improve generalization. The "optimal" network architecture is chosen by using a validation data set from a subject which has not been incorporated in the training or the test sets.

4000 examples from 8 subjects (500 from each) are used for training the neural network, and the generalization ability is then tested using 8000 examples from the same group of subjects (1000 from each). The examples which go into the training and test set are chosen so the estimated glucose utilization (R) is in the interval $[1.0, 12.0]$. Pixels with R-values outside this interval were typically found outside the brain or were discarded as severe noise outliers. Further, the generalization ability is tested on a data set consisting of 1500 examples from a subject not included in the training set. The "optimal" network is chosen using a validation data set consisting of 1500 examples from still another subject which has not been used for training or testing.

From equation (7) it can be seen that when R is calculated the rateconstant factor is multiplied by a global factor c_{gl} for each subject: the glucose concentration in the blood. To assist the training problem, the glucose utilization R is divided by this global factor for each subject (it is difficult for a feed-forward net to learn a multiplication). The neural network is then trained to estimate R/c_{gl}.

The data sets used for training and testing the neural network are normalized to have unit variance. Further, intial experiments have shown that it is very difficult to train the neural network to estimate the global mean glucose utilization level. The problem is that only 8 different subject are represented in the training set. The global mean factor for each subject is therefore subtracted from the output of the network and the learning/test set outputs. The normalized squared errors shown in the following figures are calculated from the mean subtracted data.

In figure 5 the result of a pruning session of weights in the network is shown. We see that the network easily generalizes to the (interpolation) test data derived from the same 8 subjects that were used to collect the training set. The minimum of the validation error is seen to be for a network with 278 weights. This network architecture also generalizes well to the independent test set.

In figure 6 the left panel shows the direct estimate of the glucose utilization for a subject from the training set (4×4 superpixels), while the right panel shows the output from the trained neural network. The neural network generalizes very well from the 500 examples taken from this subject to the

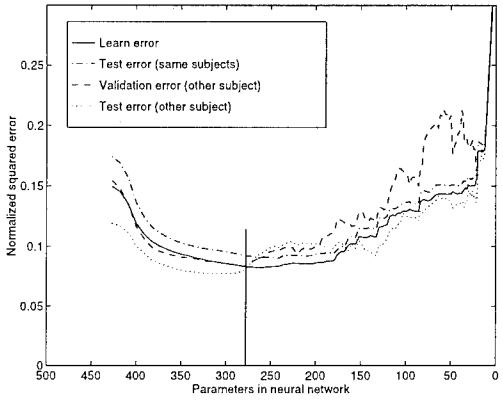


Figure 5: Normalized training and test errors (unit variance, mean subtracted for each subject) during a pruning session of the neural network. The parameters are pruned using the OBD method. For test data from the same subjects as used in the training set the test error closely follows the training error curve. The validation error is used for chosing "optimal" network architecture. It can be seen that this network also generalizes well for the independent test set.

rest of the pixels in the image.

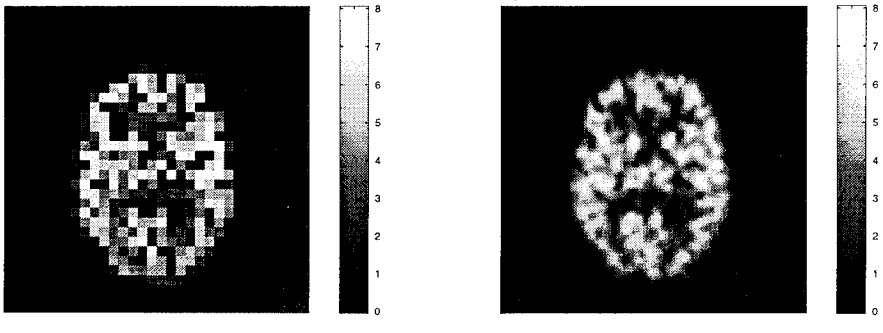


Figure 6: Images showing the glucose utilization as determined by fitting the kinetic model (subsampled 4×4 superpixels) using a second Newton scheme (left panel) and as determined by the neural net operating as inverse model for the kinetics (right panel). A total of 500 examples from the 15 slices for this subject is used as a training set, together with 500 examples from each of 7 other subjects. The right panel shows the result of evaluating the neural network on each pixel in the slice, the network generalizes very well for this subject.

Figure 7 shows the estimate of the glucose utilization for a subject, which was not used in the training set. Qualitatively the similarity is striking.

However, it can be seen that there is a problem in estimating the global level. Especially, the low level (the background) can be seen to have an offset.

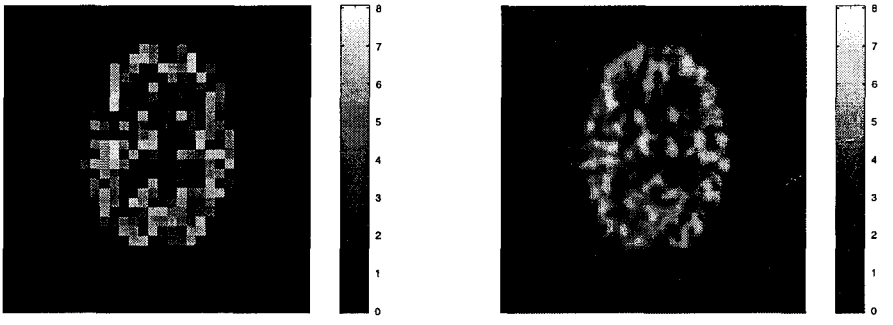


Figure 7: Images showing the glucose utilization as determined by fitting the kinetic model (pixel by pixel) using a second Newton scheme (left panel) and as determined by the neural net operating as inverse model for the kinetics (right panel). This subject has not been incorporated in the training set. A major part of the generalization error is caused by a global offset; the origin of this offset it is not clear at present.

It is important to note the difference in the computational burden of the two methods. The execution time for the neural network is orders of magnitude less than the time needed to fit the rate constants in Sokoloff's model by the Newton method. Using the neural network it is possible to estimate the glucose metabolism in 15 slices of 128×128 pixel images in less that 4 minutes on the HP9000/735 workstation, compared to about 48 hours required by the fitting procedure.

CONCLUSION

In this article it has been shown how a neural network can be trained to learn the inverse model for the three compartment PET tracer kinetic model. The neural network way of estimating the glucose utilization is significantly faster than fitting the kinetic rate constants directly and the generalization ability of the neural network solution seems very good even to subjects which have not been incorporated in the training set. We are currently pursuing robust estimation of the global mean glucose utilization, also we expect that the neural network can be used to estimate the glucose utilization in pixels from a subject with serious diseases, e.g. Alzheimers disease.

REFERENCES

[1] T. Fog, L.H. Nielsen, L.K.Hansen, S. Holm, I. Law, C. Svarer, and O. Paulson: "Neural Estimation of Kinetic Rate Constants from Dynamic PET-Scans" in *Neural Networks for Signal Processing*, Vols. IV, IEEE, Piscataway, NJ (1994).

[2] Y. Le Cun, J.S. Denker, and S.A. Solla: "Optimal Brain Damage" in *Advances in Neural Information Processing Systems*, Vols. II, Morgan Kaufmann, San Mateo (1990).

[3] M.E. Phelps: "Positron Emission Tomography (PET)". In: J.C. Maziotta and S. Gilman (Eds.):*Clinical Brain Imaging, Principles and Applications*, F.A. Davis Company, Philadelphia (1992).

[4] M. Reivich, A. Alavi, A. Wolf, J. Fowler, J. Russell, C. Arnett, R.R. MacGregor, C.Y Shiue, H. Atkins, A. Anand, R. Dann, and J.H Greenberg: "Glucose metabolic rate kinetic model parameter determination in humans: the lumped constants and rate constants for $[^{18}F]$fluorodeoxyglucose and $[^{11}C]$deoxyglucose." in Journal of Cerebral Blood Flow **5** 179-192 (1985).

[5] H. Sasaki, I. Kanno, M. Murakami, F. Shishido, and K. Uemuda: "Tomographic Mapping of Kinetic Rate Constants in the FDG model Using Dynamic PET". Journal of Cerebral Blood Flow **6** 447-454 (1986).

[6] L. Sokoloff, M. Reivich, C. Kennedy, M.H. Des Rosiers, C.S. Patlak, K.D. Pettigrew, O. Sakurada, and M. Shinohara: "The C-14-Deoxyglucose Method for the Measurement of local Cerebral Glucose Utilization: Theory, Procedures and Normal Values in the Conscious and Anesthetized Albino Rat" Journal of Neurochemistry **28** 897-916, (1977).

[7] C. Svarer, L.K.Hansen, and J. Larsen: "On design and evaluation of tapped-delay neural network architectures" in *IEEE International Conference on Neural Networks*, Vols. I, IEEE, Piscataway, NJ (1993).

NONLINEAR ECHO CANCELLATION USING A PARTIAL ADAPTIVE TIME DELAY NEURAL NETWORK

A. N. Birkett, R. A. Goubran
Department of Systems and Computer Engineering
Carleton University, 1125 Colonel By Drive
Ottawa, Canada, K1S 5B6
Tel: (613) 788-2600 ext. 5740, Fax: (613) 788-5727
e-mail: birkett@sce.carleton.ca

Abstract: System identification of a nonlinear loudspeaker/microphone acoustic system is necessary to achieve high acoustic echo cancellation in the handsfree telephony environments where the loudspeaker often operates at high volumes. In this paper, a partial adaptive process consisting of a small order tapped delay line neural network (TDNN) followed by a delayed Normalized Least Mean Squares (NLMS) adaptive filter is used to model a loudspeaker/microphone acoustic system. The TDNN models the first part of the acoustic impulse response (AIR) where most of the energy is contained and the delayed NLMS filter models the remaining echo. Experimental measurements confirm that a short length TDNN is capable of improved identification in an undermodelled system and that by extending this to the partial adaptive TDNN structure, the ERLE performance improves by 5.5 dB at high loudspeaker volumes when compared to a NLMS structure.

INTRODUCTION

In this paper, a partial adaptive process consisting of a tapped delay line feedforward neural network (TDNN) and normalized least mean squares (NLMS) structure are employed in an attempt to model loudspeaker nonlinearities at high volumes. The specific application here is improved steady state performance for acoustic echo cancellers in the handsfree environment using conference type speakerphones. Most of these consumer products employ inexpensive audio components which are susceptible to nonlinear distortion at low frequencies and high volumes. In this paper, the identification of the nonlinear loudspeaker/microphone system is considered. In a real environment however, the AEC structure must be capable of identifying and tracking not only the reflected signals from the room, i.e. its acoustic impulse response (AIR), but also of modelling the plastic enclosure vibrations and nonlinear loudspeaker response, as shown in Figure 1.

0-7803-2739-X/95 $4.00 © 1995 IEEE

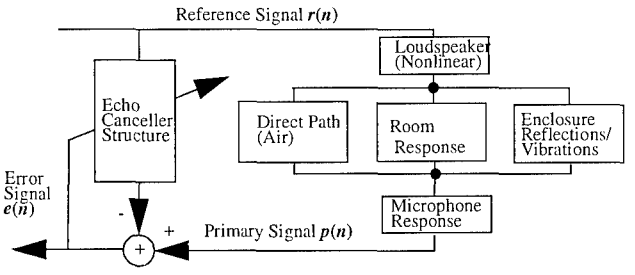


FIGURE 1. Acoustic Echo Canceller Structure. The AEC must identify not only the AIR but nonlinear and vibration effects as well.

Conventional AECs utilize a linear adaptive transversal filter to model the AIR and cancel the echo signal. At low volumes where nonlinearities are absent, this is a classical system identification problem whereby the adaptive filter adjusts its coefficients via the NLMS algorithm [4] to model the echo path, $H(z)$ between the loudspeaker and the microphone so the system output, $e(n)$ is minimized. The NLMS algorithm is the baseline by which performance of alternative models is measured but it is incapable of reducing nonlinear distortion. A measure of the AEC performance is the Echo Return Loss Enhancement (ERLE) which is defined as [5];

$$ERLE\,(dB) \;=\; \lim_{N\to\infty}\left[10\log\frac{E\,[p^2\,(n)\,]}{E\,[e^2\,(n)\,]}\right] \cong 10\log\left[\frac{\sigma^2{}_p}{\sigma^2{}_e}\right] \tag{1}$$

where $\sigma^2{}_p$ and $\sigma^2{}_e$ refer to the variances of the primary and error signals respectively and E is the statistical expectation operator.

Limitations of AEC Performance

1) *TIP/TP Ratio*: One of the limitations of AECs is undermodelling of the AIR. As shown in [5] the achievable ERLE is determined in part by the degree of undermodelling of the unknown system. The results show that the achievable ERLE is determined by the Total Impulse Power to Tail Power (TIP/TP) ratio, defined as;

$$\frac{TIP}{TP} \;=\; 10\log\left[\frac{\sum_{i=0}^{M-1} h^2\,(i)}{\sum_{i=N}^{M-1} h^2\,(i)}\right] \tag{2}$$

where h is an impulse of length M and N is the discrete point at which the "tail" is considered to start. The TIP/TP ratio is invaluable for determining the optimum

number of AEC filter taps to use given a certain loudspeaker, microphone and enclosure. A TDNN can be used in place of a NLMS filter to identify a loudspeaker [7] and to see the effect on the TIP/TP ratio when operating at high loudspeaker volumes. The experimental results shown in Figure 6 indicate that the TDNN is capable of achieving a higher ERLE than the NLMS when in an undermodelled state, i.e. when the number of delays in the delay line is less than the AIR. This improvement in performance can be used in the partial adaptive structure (described below) to obtain improved performance at high volumes.

2) *Nonlinear Distortion:* The nonlinear parameters of a loudspeaker may be described by the force deflection characteristics of the loudspeaker cone suspension system and nonlinear flux density as described in references [1][2]and [3]. Suspension system nonlinearity manifests itself as soft clipping at the loudspeaker output and results in odd-order harmonics under large signal conditions. The nonlinear distortion consists mainly of cubic terms and can easily be 5 to 10 percent of the total output, especially when dealing with small loudspeakers that have low power ratings. Simulations and experimental results indicate that neural network models can identify this nonlinear distortion more effectively than linear adaptive structures.

3) *Enclosure Vibration*: Vibration is a serious problem that occurs under the same conditions as nonlinear distortion, namely at low frequencies and high volumes. It is important that this be addressed in a practice but is considered beyond the scope of this paper and will not be discussed further.

Figure 2 shows the general ERLE performance in a typical echo environment [5]. In a simulated experiment, the ERLE will follow the TIP/TP ratio very closely, however, in actual measurements, limitations such as room noise, vibration and loudspeaker nonlinearities will limit the achievable ERLE as indicated.

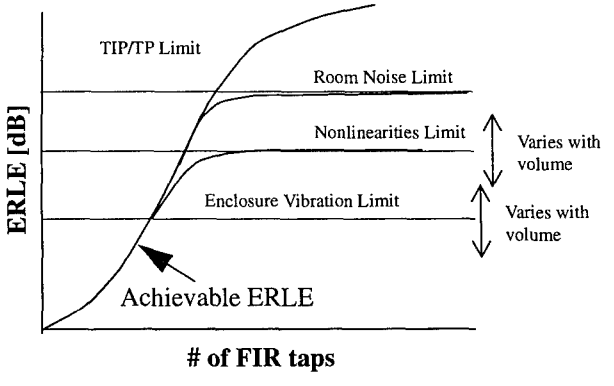


FIGURE 2. Achievable ERLE as a function of Physical Limitations. In the absence of vibration, nonlinear distortion and room noise, the achievable ERLE is determined by the TIP/TP ratio.

NEURAL NETWORK MODELS

Tapped Delay Line Neural Network

A tapped delay line neural network previously presented in [6] is shown below in Figure 3. It can be used to perform a nonlinear system identification of the loudspeaker/microphone system. It consists of a tapped delay line input layer, two hidden layers which have a piecewise linear/sigmoidal activation function and a linear output node. The piecewise linear/sigmoidal activation function is linear below +/- 0.2 and then follows a squashed hyperbolic tangent sigmoid beyond this point such that the output is squashed between +/- 1.0. As shown in [6], this improves system identification at low volumes where the loudspeaker is essentially linear but also allows for the soft clipping effect observed at higher loudspeaker volumes. It should be noted that for activation levels less than the +/- 0.2 linear region, the gradient calculations are trivial and the filter complexity approaches that of the NLMS filter.

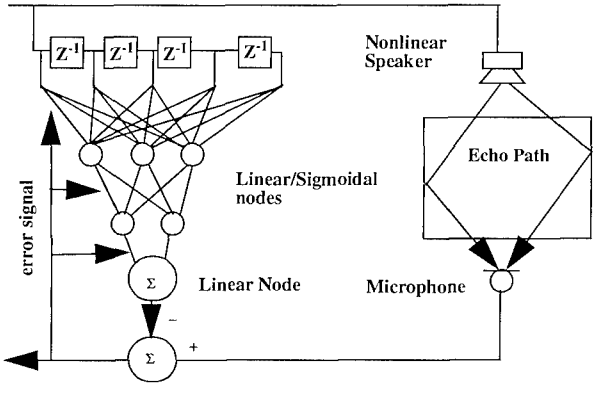


FIGURE 3. Tapped Delay Line Neural Network Adaptive Echo Canceller Structure (TDNN).

Partial Adaptive Model

The partial adaptive process utilizing a neural network preprocessor is shown in Figure 4. It consists of a low order TDNN to model the large part of the AIR and a NLMS filter to model the tail of the echo. A fixed delay line equivalent to the delay line length of the TDNN is inserted before the NLMS filter. A similar algorithm incorporating a multi-microphone linear NLMS structure is presented in [8]. In this paper however, the structure has been modified to incorporate a neural network as a nonlinear preprocessor.

Referring to Figure 4, the output $e_2(n)$ is given as;

$$e_2(n) = e_1(n) - y_2(n) = p(n) - y_1(n) - y_2(n) \tag{3}$$

where $p(n)$ is the microphone (primary) signal, $y_1(n)$ is the output of the TDNN and $y_2(n)$ is the output of the delayed NLMS filter. Expanding, we obtain;

$$e_2(n) = p(n) - y_1(n) - \sum_{i=N_1+1}^{N_2} w(n)\, x(n-i) \tag{4}$$

where $w(n)$ are the NLMS tap weights and $x(n)$ is the information vector, N_1 is the delay length of the TDNN section and N_2 is the total impulse length.

In the proposed structures there is no feedback hence the backpropagation algorithm [9] is employed to train the networks. A normalized step size [4] is employed during the training and tracking phase for both the NLMS and neural network sections. The stepsizes μ_{NLMS} and μ_{TDNN} are individually calculated and updated after each new sample is shifted into the tapped delay line.

The TDNN consists of 150 taps in the delay line, and 2 and 3 nodes respectively in the 1st and 2nd hidden layers. The NLMS section has 450 taps such that the total impulse response is 600 taps.

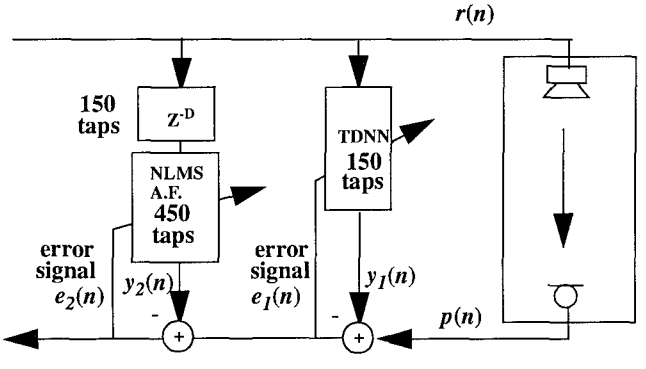


FIGURE 4. Partial adaptive structure utilizing a TDNN to cancel the first part of the AIR and a NLMS to cancel the tail portion. Signal $e_2(n)$ is the residual signal left after the echo has been cancelled.

COMPUTER SIMULATIONS

Simulations were performed using a computer generated white noise source as the reference signal, which was then filtered and convolved with an artificial room impulse function. The reference and primary files were then applied to the corresponding algorithms. For each run, the reference signal is distorted by adding both quadratic and cubic distortion according to the following equation;

$$y = \frac{ax + bx^2 + cx^3}{|a| + |b| + |c|} \tag{5}$$

where a, b, and c refer to the amplitude of the linear, quadratic and cubic factors, x is the input signal and y is the output signal level. The coefficients b and c were increased such that the distortion level increases relative to the primary signal level. The signal to distortion ratio is calculated by dividing the variance of the undistorted signal portion by the variance of the distorted signal portion. For each run, the algorithm was allowed to converge for 80000 samples and then a mean converged ERLE was obtained. The results shown below in Figure 5 indicate that the partial adaptive network outperforms the NLMS in high distortion environments, i.e. at low Primary/Distortion ratios.

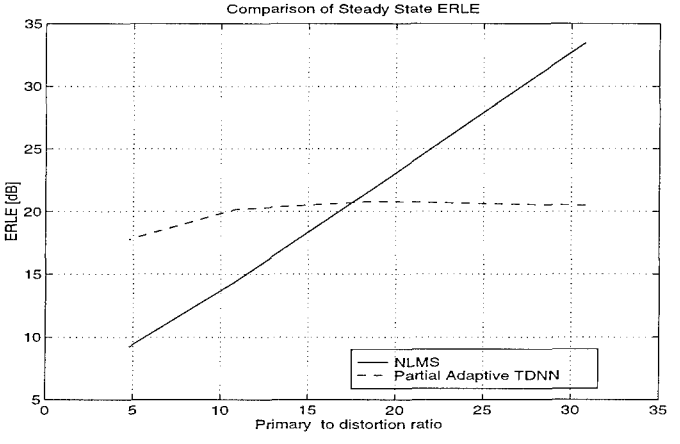


FIGURE 5. Simulation results show that the partial adaptive TDNN outperforms the NLMS in high distortion environments.

EXPERIMENTAL RESULTS

Experimental Setup

In order to remove the effects of vibration and room noise, the loudspeaker and microphone from a commercially available speakerphone were removed and placed

in a standard baffle inside an anechoic chamber. Filtered "reference" signals are applied to the loudspeaker and the microphone picks up the reflected or "primary" signal. Both the reference and primary data signals are recorded on a Digital Audio Tape and later sampled at 16 kHz and stored to disk for off-line processing. The loudspeaker volume is varied from levels of 75 dB Sound Pressure Level (SPL) to 100 dB SPL, measured at a distance of 0.5 meter. Both the partial adaptive TDNN structure and the NLMS algorithm are applied to the measured data and a number of ERLE curves are obtained for various SPL levels.The algorithm is allowed to converge for 32000 samples and then the average ERLE is obtained from the last 8000 output values.

TIP/TP Performance for the TDNN

The recorded data was applied to the TDNN structure to determine the optimum length for the TDNN section for the highest volume (100 dB SPL) case. The results shown in Figure 6 illustrate that for a system with undermodelling of the impulse length, the TDNN has improved ERLE performance compared to the stand alone NLMS. The best performance comes at approximately 150 taps where the difference between the TDNN and NLMS ERLE value is approximately 5.5 dB.

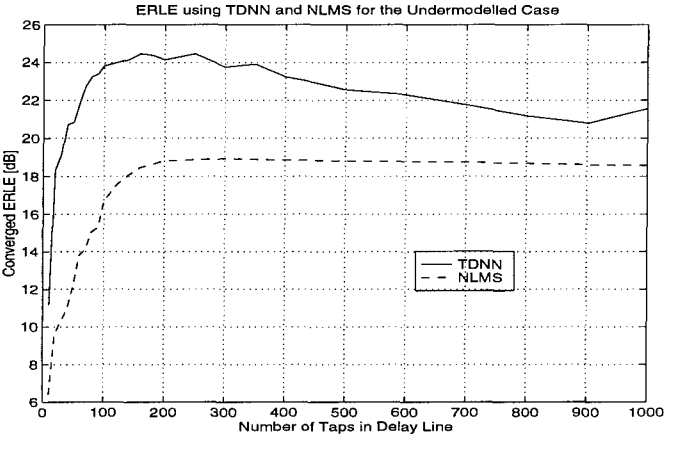


FIGURE 6. Experimental Results. A TDNN is capable of obtaining a better ERLE in an undermodelled state as compared with the NLMS algorithm. Results obtained at a high volume level of 100 dB SPL measured at a distance of 0.5 meter.

Partial Adaptive Structure Performance with Increasing SPL

As shown in Figure 7, the converged ERLE for the partial adaptive structure decreases from a high of 32dB at 80dB SPL to 18.5dB at 100 dB SPL when using the NLMS algorithm. This agrees with results presented in [4] and [8] which show that ERLE is low for low speaker volumes (where acoustic, thermal and DSP related noise are significant) but increases as the reference signal increases, eventually reaching a plateau. Any increase in reference signal level to the loudspeaker after this point results in a *decrease* in the ERLE due to nonlinear distortion. Also shown for comparison is the partial adaptive TDNN algorithm which outperforms the NLMS algorithm at high volume levels. The TDNN section consisted of 150 taps as determined from Figure 6.

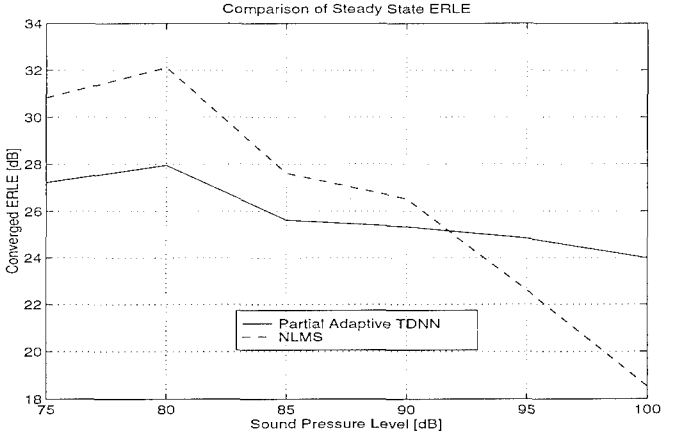


FIGURE 7. Experimental Results. Converged ERLE performance of the partial adaptive TDNN structure compared to the NLMS structure. A 5.5 dB improvement in ERLE can be obtained at high volumes.

The length of the total impulse response is the same for both the partial adaptive TDNN structure and the baseline NLMS structure and is truncated to 600 taps. Note the improvement in ERLE over the NLMS case is significant in the high SPL volume ranges and is greater than 5.5 dB at volume levels in the vicinity of 100 dB SPL.

Convergence

Figure 8 illustrates the ERLE convergence of the partial adaptive TDNN structure compared with the NLMS structure, obtained using data recorded at 100 dB SPL. The convergence rate of the new structure is slightly worse than the NLMS and will affect the tracking performance of the AEC. Methods to reduce this are currently under investigation.

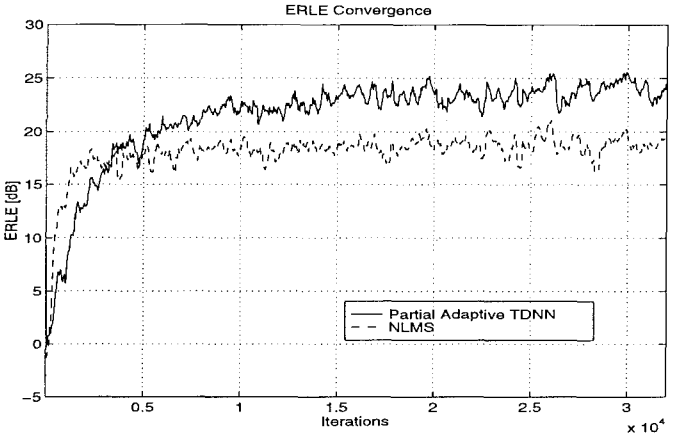


FIGURE 8. Experimental Results. ERLE convergence curves for the partial adaptive TDNN structure and the stand alone NLMS AEC at the highest volume.

CONCLUDING REMARKS

A TDNN structure has been shown to improve the achievable ERLE of a loudspeaker/microphone system at high volumes. This suggests the use of a partial adaptive structure incorporating a short delay TDNN to replace a section of the NLMS filter. The partial adaptive TDNN structure was found to improve the ERLE performance over the NLMS baseline AEC by 5.5dB at high volumes where loudspeaker nonlinearities limit the achievable ERLE. All measurements were performed using real audio components. Although the new structure clearly offers improvements at high volume (i.e. high distortion) levels, it does not quite match the performance of the NLMS structure a low distortion levels. It also has a slightly slower convergence rate, although acceleration techniques were not employed. This is the subject of future research.

REFERENCES

[1] H.F. Olsen, *Acoustical Engineering*, Toronto, D. Van Nostrand Company,Inc., 1964.

[2] X.Y. Gao, W. M. Snelgrove, "Adaptive Nonlinear Recursive State-Space Filters", *I.E.E.E. Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 41, No. 11, Nov. 1994, pp. 760-764.

[3] X. Y. Gao, W. M. Snelgrove, "Adaptive Linearization of a Loudspeaker", *ICASSP* 1991 Vol. 3, pp 3589-3592.

[4] S. Haykin, *Adaptive Filter Theory*, 2nd ed., Prentice-Hall, Toronto, 1991

[5] M.E. Knappe, R.A. Goubran,"Steady State Performance Limitations of Full-Band Acoustic Echo Cancellers", *ICASSP* 1994, Adelaide, South Australia, Vol. 2, pp. 73-76.

[6] A.N. Birkett, R. A. Goubran, "Acoustic Echo Cancellation for Hands-free Telephony Using Neural Networks", *Neural Networks for Signal Processing 1994, IEEE Workshop Proceedings,* Sept. 1994,pp. 249-258.

[7] P. Chang, C. Lin, B. Yeh, "Inverse Filtering of a Loudspeaker and Room Acoustics Using Time-delay Neural Networks", *Journal of the Acoustic Society of America,* Vol 95, No. 6, June 1994, pp. 3400-3408.

[8] S. E. Kuo, J. Chen, "Multiple Microphone Acoustic Echo Cancellation System with the Partial Adaptive Process", *Digital Signal Processing* 3, 1993, pp. 54-63.

[9] Y. Pao, *Adaptive Pattern Recognition and Neural Networks,* Addison-Wesley Publishing Company, Inc, 1989.

ACKNOWLEDGEMENTS

The authors wish to thank NSERC, Carleton University, BNR and the Telecommunications Research Institute of Ontario (TRIO) for their financial support.

Customized ECG Beat Classifier Using Mixture of Experts

Yu Hen Hu, Surekha Palreddy, and Willis J. Tompkins

Department of Electrical and Computer Engineering
University of Wisconsin
1415 Johnson Drive
Madison, WI 53706-1691
hu@engr.wisc.edu
(608) 262-6724 (ph), (608) 262-1267 (fax)

I. ECG Beat Classification Problem

Computer-assisted monitoring and analysis of (surface) ECG signals has received much attention in the past few decades [2, 3, 4, 5, 6, 8, 10, 11, 12, 16, 19]. Over years, the performance of computer based ECG monitoring has gradually improved and been accepted in many hospitals and clinics. However, certain performance obstacles remain to be solved. One major problem faced by today's automatic ECG analysis machine is the wild variations of ECG wave form morphologies of different patients and patient groups. An ECG beat classifier which performs well for a given training database often fails the test when presented with a different patient's ECG wave form. Such an inconsistency in performance is a major hurdle preventing fully automated ECG processing systems to be widely used clinically. A naive, yet impractical solution to this problem is to customize the ECG processing algorithm to each individual patient manually. To do this, physicians must laboriously edit the ECG record for each patient, and engineers must program such changes into the ECG processing algorithm. The cost involved in such a practice quickly defeats the original purpose of using automated ECG machine – to save time and labor. Indeed, today's commercial ECG monitoring machines are designed for the use by general patient populations, and hence potentially suffer from this "patient adaptation" problem.

In this paper, we report an effort to develop a user-adaptation (customization) algorithm which, with minimal human supervision, is capable of adapting the performance of an ECG beat classification algorithm to the special characteristics of individual patients. In our algorithm, we assume a black box ECG beat classifier, and a small set of user-specific training data, edited by human experts, are given. The objective is to devise a method to take advantage of the small user-specific training data set to achieve high classification rate than using the black box ECG beat classifier alone.

The black box ECG beat classifier model is used in our work to model the commercial ECG processing machines whose internal processing algorithm and training database are highly guarded company proprietary information and can not be made available to end users. Under the black box model, we are allowed to observe the output of classification when presenting a feature vector to the ECG beat classifier, but not to temper the internal ECG beat classification algorithms. Moreover, we can not assume the availability of original training data which helped producing the black-box ECG beat classifier. Instead, at our disposal is the much smaller user-specific training data set.

0-7803-2739-X/95 $4.00 © 1995 IEEE

Proposed Approaches

This user adaptation problem bears certain resemblance to the incremental learning problem in that new data is to be incorporated to improve existing classifier's performance. However, the black box model of the existing classifier prevents us to directly modify the classifier structure as incremental learning algorithms do. Instead, we propose two different methods to circumvent this problem:

Mixture of Experts

The basic idea is to leave the existing black box classifier intact. Instead, we use the given small, user-specific training data set to develop a "local expert" classifier. Then we invoke a modified "mixture of experts" approach [18], [13], [17], [7], [9] to combine these two classifiers, hoping to achieve better performance. Briefly, let $y_1(x)$ and $y_2(x)$ be the output of the two respective classifiers, then the combined output (passing through a threshold unit) of that class will be

$$y(x) = f[g_1(x)y_1(x) + g_2(x)y_2(x)] \quad 1 \le c \le C \tag{2}$$

where C is the number of classes, f is a threshold unit, $g_i(x)$s are scalar outputs from a *gating network*, and are defined by [1]

$$g_i(x) = \exp(v_i^t x) / \sum_{j=1}^{2} \exp(v_j^t x). \tag{3}$$

One interesting question is whether the classification rate of y(x) can be made higher than that of $y_1(x)$ or $y_2(x)$ alone.

Theorem 1. Define $R(y_i(x)) = \{x | x \in X, \text{ and } y_i(x) = z(x)\}$, i = 1,2 to be the sub-region in the feature space where the classifier $y_i(x)$ makes correct classification of x and let R(y(x)) be defined the same way. Assume $y_i(x) \in \{0,1\}$ and $z(x) \in \{0,1\}$, then

$$R(y(x)) \supseteq R(y_1(x)) \cup R(y_2(x)). \tag{4}$$

Proof: We need only to prove that if both $y_1(x)$ and $y_2(x)$ mis-classify a given feature vector x, then y(x) can not give correct classification on x. Since the correct classification output z(x), the combined output y(x), and individual classifier output $y_1(x)$ and $y_2(x)$ are all binary vectors of the same dimension, if both classifiers mis-classify a given feature vector x which belongs to class c, we must have, for the c^{th} elements of these binary vectors,

$$z_c(x) \& y_{1c}(x) = z_c(x) \& y_{2c}(x) = 0$$

where "&" is the "exclusive-OR" operator in Boolean algebra. Since from eq. (3), $g_1(x) + g_2(x) = 1$, we conclude $y_c(x) = 0$ if $y_{1c}(x) = y_{2c}(x) = 0$, and $y_c(x) = 1$ if $y_{1c}(x) = y_{2c}(x) = 1$. Hence $z_c(x) \& y_c(x) = 0$. In other words, y(x) must also mis-classify the same feature vector x regardless the choice of $g_{1c}(x)$ and $g_{2c}(x)$. This is to say if $x \notin R(y_1(x))$, and if $x \notin R(y_2(x))$, then $x \notin R(y(x))$. n

The implication of theorem 1 is that maximum performance enhancement of a mixture of experts approach occurs when $R(y_1(x)) \cap R(y_2(x)) = \phi$ (empty set). An example is to designate each classifier to be responsible for classifying a particular class. The assumption that $y_i(x) \in \{0,1\}$ is essential in this theorem. If $y_i(x) \in [0, 1]$ (interval between 0 and 1), it is possible to find a counter example. Let $y_1(x) = [.1\ .5\ .4]$, $y_2(x) = [.5\ .1\ .4]$, and $z(x) = [0\ 0\ 1]$. Then $y(x) = [0.1g_1 + 0.5g_2\ \ 0.5g_1 + 0.1g_2\ \ 0.4]$. If $g_1 = g_2 = 0.5$, then $y(x) = [0.3\ \ 0.3\ \ 0.4]$ which yields correct classification.

On the other hand, regardless whether $y_i(x)$ take binary values, if both classifiers made correct classification, so will the combined classifier.

Theorem 2. With the same definitions as in theorem 1, and $y_i(x) \in [0, 1]$,

$$R(y(x)) \supseteq R(y_1(x)) \cap R(y_2(x)). \tag{5}$$

Proof: Assume $x \in$ class c^*, and $y_{1c^*}(x) = \max_c y_{1c}(x)$, $y_{2c^*}(x) = \max_c y_{2c}(x)$. Then

$$h_{c^*}(x) = g_1(x)y_{1c^*}(x) + g_2(x)y_{2c^*}(x) = \max_c [g_1(x)y_{1c}(x) + g_2(x)y_{2c}(x)] \tag{6}$$

Thus the output $y_{c^*}(x)$ is correctly classified. n

>From theorem 2, it is clear that if both classifiers #1 and #2 correctly classify a pattern x, then the combined classifier will also correctly classify the same pattern. Hence this pattern can be excluded from the user-adaptation training set as it will not affect the result.

Adaptation Algorithm

Based on the result indicated in theorem 1, and theorem 2, the design objective of the mixture of experts network in eq. (3) is to devise a training algorithm to estimate the parameter vectors $\{v_i;\ i = 1, 2\}$. Given that $y_1(x)$ and $y_2(x)$ are fixed classifiers, this problem can be solved by a gradient procedure as follows: Let us assume $\{x_k;\ 1 \le k \le K',\ x_k \in X\}$ be a set of training data used for searching the optimal gating functions $g_1(x)$ and $g_2(x)$ such that the square error at the output $E = (1/2K')\sum_{k=1}^{K'} \|z_k - y(v_1, v_2, x_k)\|^2$ is minimized. A gradient search algorithm can be devised as the following:

$$v_i(t+1) = v_i(t) - \mu\, \nabla_{v_i} E \tag{7}$$

The initial values of v_1 and v_2 are set to be the centroids of the regions $R(y_1(x))$ and $R(y_2(x))$ respectively for x in the user-specific training data set. The gradient of E with respect to v_i can be calculated as:

$$\nabla_{v_i} E = (1/K')\sum_{c=1}^{C}\sum_{k=1}^{K'} [y_c(v_1, v_2, x_k) - z_{kc}] \cdot \nabla_{v_i} y_c(v_1, v_2, x_k) \tag{8}$$

$$= (1/K')\sum_{k=1}^{K'}\sum_{c=1}^{C}\{[y_c(x_k)-z_{kc}]\cdot f'(h_c(x_k))\cdot[\sum_{j=1}^{2} y_{jc}(x_k)\cdot\nabla_{vi}\, g_j(v_1,v_2,x_k)]\} \qquad (9)$$

where $h(x) = g_1(x)\cdot y_1(x) + g_2(x)\cdot y_2(x)$. In (7), we assumed the transfer function f is a differentiable threshold function, and is applied to a vector element by element. Finally, with eq. (3), we have

$$\begin{aligned}\nabla_{vi}\, g_j(v_1,v_2,x_k) &= \nabla_{vi}\, \exp(v_j^t x_k)/\sum_{m=1}^{2}\exp(v_m^t x_k)\\ &= \frac{\nabla_{vi}[\exp(v_j^t x_k)]\cdot\sum_{m=1}^{2}\exp(v_m^t x_k) - \exp(v_j^t x_k)\cdot\nabla_{vi}\exp(v_i^t x_k)}{[\sum_{m=1}^{2}\exp(v_m^t x_k)]^2}\\ &= (-1)^{i-j}\, x_k\cdot g_1(x_k)\cdot g_2(x_k) \qquad (10)\end{aligned}$$

Hence, for i, j [{1, 2}, i ≠ j, we have

$$\nabla_{vi}\, E = (1/K')\sum_{k=1}^{K'}\sum_{c=1}^{C} x_k\{[y_c(x_k)-z_{kc}]\cdot f'(h_c(x_k))\cdot[y_{ic}(x_k)-y_{jc}(x_k)]\cdot g_1(x_k)\cdot g_2(x_k).$$

Note that in above derivation, the error E is accumulated over the entire epoch (K' feature vectors). The summation over k may be removed if we use on-line update of v_i's for each sample. This yields the following expression: for i, j [{1, 2}, i ≠ j,

$$\nabla_{vi}\, E = x_k\cdot[y(x_k)-z_k]^t\cdot\mathrm{diag}\{f'(h(x_k))\}\cdot[y_i(x_k)-y_j(x_k)]\cdot g_1(x_k)\cdot g_2(x_k)\} \qquad (11)$$

Clearly, we have $\nabla_{v1}\, E = -\nabla_{v2}\, E$. This is not surprised because with two parameter vectors facilitate a decision hyperplane $(v_1 - v_2)^t x = 0$.

Until now, we have assumed that the user-specific ECG beat classifier $y_2(x)$ is readily available. In reality, it needs to be trained with the user-specific training data set. On the other hand, the combined classifier y(x) needs also be trained by the same data set in order to determine the gating network parameters. Therefore, if $y_2(x)$ is trained with 100% correctness of the user-specific data set, then the gating network of choice may be that $g_2(x) \neq 1$, and $g_1(x) \neq 0$. In light of the results of Theorems 1 and theorem 2, we devised the following strategy to alleviate this problem: First, we construct the user-specific training data set to contain only those feature vectors which the original classifier misclassified. We further partition this training data set into two subsets. One for the training of the user-specific classifier $y_2(x)$, and the other for estimating $g_1(x)$ and $g_2(x)$.

Experiment Results

ECG Data Sets and Feature Vectors

For this study, ECG data was acquired from a MIT/BIH arrhythmia CDROM database[14, 15]. The database includes 48 fully annotated two-channel ECG records of 30 minutes duration each. In this study, we selected 33 records out

of the 48 records to investigate the problem of PVC (premature ventricular contraction) beat detection. The data was sampled at 360 Hz with 12 bits/sample resolution. The annotation file for each record specifies the type of ECG beats, rhythms, and their corresponding time coordinates. Four classes, N (normal), V (ventricular ectopic beat, ventricular premature beat, R-on-T ventricular premature beat, or ventricular escape beat), F (fusion of ventricular and normal beat) and Q (unknown, others) are used in this work.

Each ECG beat is represented by a 51-point template extracted from the ECG record (first channel only), centered at the peak of the QRS complex, and extends 25 points to each side. This 1 by 51 vector is then normalized to have a dynamic range between 0 and 1. In addition to the morphological features, we also incorporate four temporal features: the instantaneous R-R interval, the average R-R interval, the difference between the instantaneous and average R-R intervals and the width of the QRS complex. Thus the feature space dimension is 55.

Incremental Training of a Trained Modeling SOM/LVQ Network

A classifier was developed using Kohonen's self-organizing map(SOM) and fine-tuned with Learning vector quantization(LVQ). In the testing phase, the classifier's performance was observed to improve with supervised fine tuning of the classifier with the first five minutes of data from each record. For the record#121, the classification rate improved from 96.61% to 99.74%. However, to overcome the need to have an annotation file of the data, an unsupervised clustering algorithm was used to cluster the data and label the clusters with the original classifier. The labeled clusters were then used to fine-tune the original classifier, improving it's performance. The performance results on MIT/BIH record#121 with this technique provided an accurate classification rate of 99.81%. In another example, with record#201, the classification rate improved from 90.87% to 90.98%.

Mixture of Experts Approach

In the mixture of experts approach, the clusters developed with an unsupervised algorithm constitute a second classifier 'local expert'. The output of this classifier is threshold to provide a classification output only when the inputs fall within the threshold region of the clusters, else the classifier declares that, it has not seen that input before. A gating network was then trained to weigh the outputs of the original classifier and the local expert. A classification rate of 96.54% was obtained for record #121 using the gating network. For record#201, a classification rate of 92.33% was obtained which is a significant improvement over the previous method. This result indicates that there is a potential for improvement in accurate classification rates using the method of mixtures. More extensive experiments are being performed to evaluate the complete database using this method.

Acknowledgment:

The authors would like to thank Dr. Shen Luo at Burdick Inc., Milton, WI for many helpful discussions and suggestions.

References

[1] Bridle, J., "Probabilistic interpretation of feedforward classification network outputs, with relationship to statistical pattern classification," in *Neuro-*

computing: algorithms, architectures, and applications, F. Fogelman-Soulie and J. Herault, Ed. New York: Springer-Verlag, 1989,

[2] Cox, J. R. J., F. M. Nolle, and R. M. Arthur, "Digital analysis of the electroencephalogram, the blood pressure wave, and the electrocardiogram," *Proc. IEEE*, vol. 60, pp. 1137, 1972.

[3] Drazen, E. L., and E. F. Garneau, "*Use of computer-assisted ECG interpretation in the United States*," in Proc. *Computers in Cardiology*, New York, IEEE Press, 1979,

[4] Foster, F. K., W. D. Weaver, "Recognition of ventricular fibrillation, other rhythms and noise in patients developing the sudden cardiac death syndrome," *Computers in Cardiology*, pp. 245--249, 1982.

[5] Gerlings, F. D., Bowers, D. L., and Rol, G. A., "Detection of abnormal ventricular activation in a coronary care units," *Comp. Biomed. Res.*, vol. 5, pp. 14, 1972.

[6] Helppi, R. K., Unite, V., and Wolf, H. K., "Suggested minimal performance requirements and methods of performance evaluation for computer ECG analysis programs," *Can. Med. Assoc. J.*, vol. 108, pp. 1251, 1973.

[7] Jacob, R. A., M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mistures of local experts," *Neural Computation*, vol. 3, pp. 79-87, 1991.

[8] Jenkins, J. M., . and Arzbaecher, R., "On-line computer pattern recognition and classification of ventricular and supraventricular arrhythmias," in *Progress in Electrocardiology, Macfarlane*, Ed. Glasgow: Pittman Medical, 1979,

[9] Jordan, M. I., and R. A. Jacob, "Hierarchies of adaptive experts," in *Advances in Neural Information Processing Systems 4*, J. Moody S. Hanson, and R. Lippmann, Ed. San Mateo, CA: Morgan Kaufmann, 1992, pp. 985-993.

[10] Klingeman, J., and Pipberger, H. V., "Computer classification of electrocardiograms," *Comp. Biomed. Res.*, vol. 1, pp. 1, 1967.

[11] Kumar, V. V., "A novel approach to pattern recognition in real-time arrhythmia detection," in Proc. *IEEE 10thInt. Conf. BME*, 1988, pp.7--8.

[12] Macfarlane, P. W., Lorimer, A. R., and Lowrie, T. D. V., "3 and 12 lead electrocardiogram interpretation by computer. A comparison in 1093 patients," *Br. Heart J.*, vol. 33, 226, pp. 1971.

[13] Mandler, E., and J. Schuermann, "Combining the classification results of independent classifiers based on the Dempster Shafer theory of evidence," in *Pattern Recognition and Artificial Intelligence*, G. a. Kanal, Ed. Amsterdam: Elsevier Science, 1988, pp. 381-393.

[14] Mark, R., and G. Moody, "MIT-BIH Arrhythmia Database Directory", MIT, 1988.

[15] Moody, G. B., "ECG Database Programmer's Guide", Harvad-MIT Division of Health Science and Technology, Sept. 9, 1989, 1989.

[16] Murray, A., Campbell, R. W. F., Julian, D. G., "Characteristics of the ventricular fibrillation waveform," pp. 275--278, 1985.

[17] Wolpert, D. H., "Combining generalizers using partitions of the learning set," in *1992 Lectures in Complex Systems*, L. Nadel and D. Stein, Ed. Addison-Wesley, 1993,

[18] Xu, L., A. Krzyzak, and C. Y. Suen, "Methods of combining multiple classifiers and their applications to handwriting recognition," *IEEE Trans. Syst. Man, Cybernetics*, vol. 22, 3, pp. 418-435, 1992.

[19] Zywietz, C., Joseph G., and Grable, W., "A new intelligent electrocardiograph for stand-alone ECG analysis," in Proc. *Computers in Cardiology*, 1980.

SEMIAUTOMATED EXTRACTION OF DECISION RELEVANT FEATURES FROM A RAW DATA BASED ARTIFICIAL NEURAL NETWORK DEMONSTRATED BY THE PROBLEM OF SACCADE DETECTION IN EOG RECORDINGS OF SMOOTH PURSUIT EYE MOVEMENTS

Peter K. Tigges, Norbert Kathmann & Rolf R. Engel
Department of Psychiatry, University of Munich
Nussbaumstr.7, D-80336 Muenchen, Germany
Tel.: +49 89 5160-3403 Fax: +49 89 5160-4736
E-mail: tp@psy.med.uni-muenchen.de

ABSTRACT

Visual identification of saccades in electrooculographic (EOG) recordings of smooth pursuit eye movements (SPEM) is a very time consuming process for the individual experts. Algorithmic approaches to overcome this problem automatically produce high rates of false positive errors. Artificial neural networks (ANN) are excellent tools for pattern recognition problems when signal to noise ratio is low. An automated decision process based on modified raw data inputs showed successful proceeding of a backpropagation ANN with an overall performance of 87% correct classifications with previously unknown data. Investigating the specific influences of prototypical input patterns on a specially designed ANN led to a sparse and efficient data coding, based on a combination of expert knowledge and the internal representation structures of the ANN. Data coding obtained by this semiautomated procedure yielded a list of feature vectors, each representing the relevant information for saccade identification. The feature based ANN produced a reduction of the error rate of nearly 40% and reached an overall correct classification of 92% with unknown data. The proposed method of extracting internal ANN knowledge is not restricted to EOG recordings, and could be used in various fields of signal analysis.

INTRODUCTION

Electrooculographic recordings of smooth pursuit eye movements (SPEM) are signals with the typical problems of most biologically generated signals.

0-7803-2739-X/95 $4.00 © 1995 IEEE

A great amount of noise combined with additional biological artefacts not generated by the eye movements themselves are the prominent disturbing factors. Similar data qualities are EEG and ECG recordings. The identification of saccades in SPEM recordings suffers from similar limitations, and most scientists identify them by visual inspection of the data [4,8]. There are no definite rules which could be used for a knowledge based identification algorithm for an automated analysis.

In recent years artificial neural networks (ANN) demonstrated good classification properties when used with data of low signal to noise ratio [1,5].They are especially promising in situations where the rules for a decision are only given by expert knowledge without a clear algorithmic strategy. It was the aim of the present investigation, to classify saccades in SPEM recordings with an ANN and to show the importance of data coding for the classification quality. In a first step a raw data based decision model was examined. Even with raw data it is necessary and useful to take into account all serious information about the decision problem. Since saccades are errors in a SPEM task, a time point when saccades are expected cannot be specified. This is opposite to event related potentials in EEG recordings, where a known and controlled stimulus to determines the interesting epoch of the recorded data. For an effective presentation of the data to an ANN, a shift invariant process of pattern identification is needed. In this process all possible target patterns have to be detected. Then in a further step the ANN could be used as a classifier, like in situations with a trigger stimulus. The developed raw data coding was shift invariant regarding the time axis, and quality oriented regarding the amplitude values. Other clinical studies with similar data and identification problems have shown the usefulness of feature oriented data coding and presentation [2,6]. A central problem in developing sufficient feature lists is to know a priori, which part of the raw data holds relevant information. Since there was diverging information about this step an objective determination process was followed. It is possible to extract information about the internal decision process of an ANN [9]. In order to produce efficient data coding the influence of individual parts of the raw data vectors on the network decision was thought to give useful information for the development of an effective feature list. An automatically guided, interactive way of knowledge extraction realized the production of an objective and efficient data coding scheme.

METHODS

This study used data obtained in clinical studies of the Psychiatric Hospital of the University of Munich [3]. The subjects were instructed to follow a horizontally moving target with smooth pursuit eye movements (SPEM). Data were recorded using electrooculography (EOG). Data of 20 subjects were selected. After digital low pass filtering at 15 Hz, the resulting data were stored in epochs of 1.28 s (320 data points, 250 Hz sampling rate). This

is the time the target takes to move across 22 degrees of the visual field. Since saccades are fast movements interrupting the slow phase of smooth pursuit movements, it was necessary to examine the velocity of the horizontal eye movements. To this aim a numerical differentiation method was used, calculated by a two point central difference quotient. 1354 possible saccadic events were identified and classified by experts, to have a gold standard for the training and testing of the ANN. They were divided in two groups, 903 training and 451 test vectors.

ANN oriented data preprocessing

A threshold of 50°/s peak velocity was considered to be necessary for the existance of a possible saccadic movement. The velocity channel was scanned and vectors with 69 data points for each possible saccadic event were produced. They all contained the same number of data points prior and after the point of maximum velocity. All data were scaled, namely to have a qualitatively oriented approach versus a quantitative determined information processing. The interval (mean velocity, maximum velocity) was transformed into (0, +1) and a raw data point located outside was scaled into the outside range of the (0, +1) intervall respectively. Before these data were presented to the network, two context variables, the standard deviation and the mean velocity of the eye movement velocity of a whole set of 320 data points containing the possible saccadic event were added to each vector. This was done because reliable context information is said to have significant effect on the learning of the network [10].

Raw data ANN decision

ANN simulation was done using NeuralWorks Professional II Plus. For the presented data, an ANN with 71 units in the input layer, various units in the hidden layer (2 to 15) and a single output unit was constructed, and trained with a backpropagation algorithm [7]. An additional bias unit was integrated into the network, with constant numerical input of 1 and variable connections towards all units in the hidden layer and the output layer. The desired or actual output of 0 and 1 represents the absence respectively the presence of a saccade in the possible saccadic situation.

Raw data network decision strategy analysis

We trained a network with only two hidden units to realize two effects of further data processing. On the one hand the network was forced to build up a high degree of generalization, on the other hand we obtained the possibility to create a visualization of the hidden layer activity in a two-dimensional cartesian coordinates system. The position of individual patterns in the hidden unit space was calculated from the scalarproducts of the input vector $P=(p_1,\ldots, p_n)$ and the weight vectors $W_{hl}=(w_{hl1},\ldots, w_{hln})$ and

$W_{h2} = (w_{h21}, \ldots, w_{h2n})$ attached to the hidden units. This led to the following individual hidden layer pattern of the input vector P

$$H(P) = (\Sigma p_i w_{h1i}, \Sigma p_i w_{h2i})$$

respectively

$$H_{transfer}(P) = (\tanh(\Sigma p_i w_{h1i}), \tanh(\Sigma p_i w_{h2i}))$$

to obtain the pattern, with each component individually transferred into the range of (-1, +1). The network was trained with a modified backpropagation algorithm, combined with a consequent purging of connections. Connections with an individual absolute weight value of less than 10% of the maximum absolute weight value of all connections were disabled from learning, and set to zero. This forced the network to concentrate on relevant information of the input data. The influence of specific qualities of the prototype patterns on the desired hidden unit activation was evaluated, and we obtained a list of feature vectors, that were used as input data for a new ANN.

RESULTS

Raw data ANN performance

The gold standard of the experts visual classification showed about 70% (963/1354) saccades in all possible events, which were equally distributed in the training and test set. The raw data ANN reached an overall classification rate of 98,45% on the training set. 636 of 641 saccades were correctly identified (false negative rate: 0,78%) and 253 of 262 nonsaccadic events were correctly rejected (false positive rate: 3,44%). This shows that information was extracted in order to decide whether an input pattern is a saccade or not. With the untrained data in the test set the network correctly classified 87,36% test patterns, resulting in a false positive rate of 27,48% (36 of 131) and a false negative rate of 6,56% (21 of 320).

Feature list resulting from the raw data ANN decision strategy analysis

In Fig. 1 the hidden layer activation is displayed, in which the correctly classified saccadic patterns form a coherent set in the second quadrant of the coordinates system. Furthermore, there exist two clusters of nonsaccadic patterns, located in the first and third quadrant of the coordinates system. A correctly classified nonsaccadic pattern required at least one hidden unit activation in an opposite direction than the saccadic patterns. Two main groups could be found, nonsaccadic artefacts group A ($h_{no\ transf\ 1} < 0$, $h_{no\ transf\ 2} < 0$) and a group B ($h_{no\ transf\ 1} > 0$, $h_{no\ transf\ 2} > 0$), see [Fig. 1a]. Especially in the transferred version one can observe the linear seperability of the displayed set, the need for a successful decision process in the

remainig part of the ANN, a kind of linear perceptron. The individual location of the pattern relative to the linear function $h_2=1,012*h_1+0,953$ representing the ANN decision, shows the resulting classification [Fig. 1b].

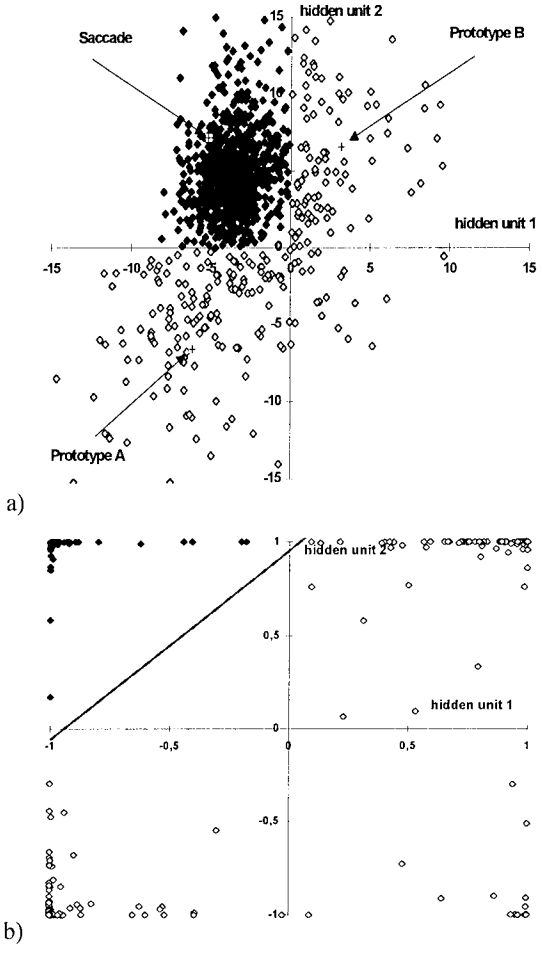


FIGURE 1. HIDDEN LAYER ACTIVITION
a) Cumulated weighted input (Saccades ♦, nonsaccadic patterns ◊, prototype patterns +) b) Transfered hidden unit activation (Saccades ♦, nonsaccadic patterns ◊, ANN decision function —)

The intragroup means of the maxima and minima amplitudes, and the medians of the latencies corresponding to the individual oscillations were calculated. Due to the sinusoidal nature of EOG data the resulting points (medians on the time axis and means on the scaled velocity axis) were interpolated by parts of cosinus waves, to attempt a smooth, i.e. a $\mathcal{C}^1$ function of the prototype patterns. We focused our interest on the relevance of these extrema for the network decision.

Which parts of the prototype patterns, and probable of the individual patterns, are responsable for the fact, that the hidden unit activation located the patterns in the different quadrants of the coordinates system? To get an insight in the process the interaction of the quality of the raw data vector, or in other termini, the activation of the input units and the weight vectors W_{h1} and W_{h2} had to be examined. The central sectors of the typical saccadic input pattern and nonsaccadic patterns e.g. showed the different effects on the activation level of the hidden units [Fig.2].

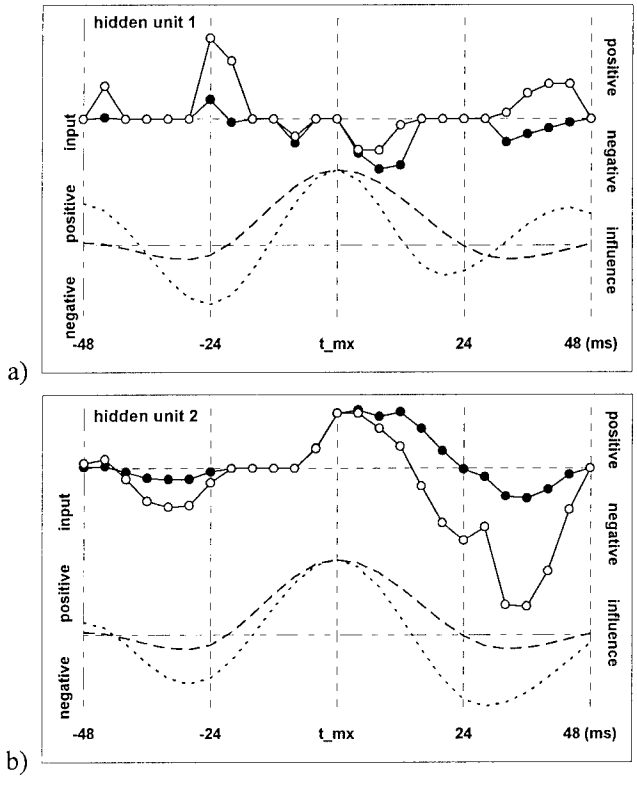


FIGURE 2. INFLUENCE OF PROTOTYPE INPUT VECTORS ON HIDDEN LAYER UNIT ACTIVATIONS

a) Input of Saccade (—— ——), of group B nonsaccadic artefact (- - -), influence on the act. of hidden unit 1 of Saccade (-•-•-), of group B nonsaccadic artefact (-o-o-)
b) Input of Saccade (—— ——), of group A nonsaccadic artefact (- - -), influence on the act. of hidden unit 2 of Saccade (-•-•-), of group A nonsaccadic artefact (-o-o-)

The classification as a saccade required negative activation of hidden unit 1. This negative activation resulted here out of mostly negative weights attached to input units with prototypical positive inputs, i.e. namely the region of the main peak from 20 ms before the point of maximum velocity up to 12 ms behind it [Fig. 2a]. Positive weights are connected with typical

negative inputs, e.g. the intervall of 32 ms til 44 ms behind the main peak. Controversely the nonsaccadic group B artefact required an exhibtion of the first hidden unit. The first minimum before the main peak produced a strong exhibition due to the negative input values and the negative weights attached with the corresponding interconnections. Depending on the shorter latency of the first maxima appearing after the main peak, the positive weights in the region of 32 ms up to 44 ms behind the main peak produced further exhibition of the first hidden unit, whereas the saccadic pattern inhibited the hidden unit on these interconnections. The first maximum before the main peak in the region arround 44 ms in front of the main peak showed positive influence on the first hidden unit activation.

To classifiy an input pattern as a saccade the second hidden unit required a positive activation [Fig. 2b]. The input values starting 4 ms before up to 24 ms behind the point of maximum velocity produced a large part of this desired positivity. The possible negative activation due to the positive weights in the regions of the surrounding minima (-44 ms to -24 ms and +28 ms to +44 ms) was less important respecting the small negative input values. The nonsaccadic artefact of Group A, however, inhibited the second hidden unit namely because of the intensive minima surrounding the main peak (-40 ms to -20 ms and +16 ms to +44 ms). The positive influence of the main peak was less important since there were fewer exhibiting data points compared to the inhibiting ones, as well as the wider activating effect of the saccadic input pattern.

TABLE 1. SCORING OF THE EXTREMA
BASED ON THE RAW DATA ANN DECISION STRATEGY

Extrema	**v_max**	**min_1**	**min_-1**	**max_1**
Influence score	31	31	18	13
Importance ranking	1	1	2	3
Variable types	**v,t,t,c,c**	**v,q,t,c,c**	**v,q,t**	**q,t**
Extrema	**min_-3**	**min_3**	**max_-1**	**min_2**
Influence score	11	10	8	8
Importance ranking	3	3	3	3
Variable types	**q,t**	**q,t**	**q,t**	**q,t**
Extrema	**max_2**	**min_4**	**min_-2**	**max_-2**
Influence score	8	6	5	3
Importance ranking	3	4	4	4
Variable types	**q,t**	**q**	**q**	**q**

v = absolute velocity variable, q = scaled velocity variable, t = time variable, c = combined variable

An importance ranking concerning the individual impact of the single oscillation of the prototype patterns on the hidden unit activations was developed. It depended on the number of raw data points with desired influence on the hidden unit activation. According to the degree of importance, different numbers of variables were added to a feature list, concerning the quality of a possibly saccadic input pattern. Five categories of influence qualities were derived. According to the degree of importance, different numbers of variables were added to a feature list, concerning the quality of a possibly saccadic input pattern. All parts of the entire pattern with an influence score of two or less were eliminated from the variable list. The result was a feature list consisting of 28 descriptive variables, each representing a specific quality of the input pattern. After determining the number of influence variables the experts decided about the individual outlook of these, depending on the way they might influence the decision [Table 1]. Three additional context variables were added, containing information about the noise in the surrounding of the main velocity peak (two intervals of 100 ms before and behind the peak) and one with information about the complete raw data pattern. So each possible saccadic input pattern was represented by a feature vector consisting of 31 components of qualitative and quantitaive description. Due to the smaller number of interconnections and input variables the network produced a higher degree of generalization and less memorizing of qualities of single raw data input pattterns.

Feature data ANN performance

Using this feature vectors of the same 903 events we trained a network with 31 input units and a similar architecture as before. One major effect of the sparse coding of the input data was a nearly independent behaviour concerning the number of processing elements in the hidden layer. The change of the overall classification rates of the test and training set was very small, no matter if the hidden layer consists of 2 or 9 hidden units.

TABLE 2. CLASSIFICATION RATES FOR THE FEATURE BASED ANN

Classification rates	*Training set*		*Test set*	
false positive	4,20%	(11/262)	10,69%	(14/131)
false negative	1,40%	(9/641)	6,88%	(22/320)
overall correct classification	97,79%	(883/903)	92,02%	(415/451)

The overall classification rate for a 7 hidden unit ANN was 97,79% on the training set. This showed the information density of the derived feature list, since the result was nearly the same as in the raw data approach. The main effect of the smaller ANN was a better generalization capability resulting in a

better overall classification rate with the unknown test data (92,02%) [Table 2]. A significant reduction of the false positve errors was observed, namely 10,69% (14 of 131) instead of 27,48% (36 of 131) in the raw data decisions. The false negative errors (22 of 320) were nearly the same as in the raw data ANN (21 of 320).

DISCUSSION

The results obtained with a raw data ANN to identify saccades in SPEM recordings showed encouraging results. The problem of a standard ANN application of this type is, that the system is used as a black box mechanism. Even if a priori knowledge about the decision process is available it is nearly impossible to use it for a better performance of the system.
The way of analyzing the internal structures of the resulting ANN described in this study leads to a solution not restricted to the black box mechanism. The a priori knowledge about the examined situation was very useful to find a new way of preprocessing SPEM data. But not only the a priori knowledge of the experts is used for the development of an optimal representation of the examined patterns. Asking the individual experts which part of the raw data curves should be considered critical, they always preferred the main peak and the nearest neighbourhood behind the main peak. No one respected the possible influence of other parts ahead of the main peak in his introspective description of the decision process. The network showed various parts of influence, located both before and after the main peak. It is unclear if the human experts actually use this information in their decision process. The internal structure of the network objectively shows which data points are important and which are not, and the proposed way shows how to extract this information. The more objective knowledge is available the better and easier the overall representation of the patterns could be managed. If there is an objective and sparse representation used as input data for an ANN, the generalization properties are used in a more effecient way than this could be done if useless data is presented or if relevant data is missing. The improved performance of the feature based ANN showed the advantages of the sparse data coding compared to the raw data situation. A further advantage is less training time and less processing time of smaller networks due to less but better input data. The proposed method is not limited to SPEM data, since there are no specific processing steps that could not be applied to other classification problems. In each step of signal based classification problems, there is a large amount of available expert knowledge. This could be used in different steps of the overall analysis as could be demonstrated in this study. Therefore, the use of ANN in pattern recognition is not only restricted to a black box machine model. Existing knowledge could be integrated in all steps of the automated processing.

REFERENCES

[1] Anderer P. et al., Discrimination between demented patients and normals based on topographic EEG slow wave activity: comparison between *z* statistics, discriminant analysis and artificial neural network classifiers, **EEG and clinical Neurophysiology**, 91, 108-117, 1994

[2] Bankman I.N. et al., Feature-Based Detection of the *K*-Complex Wave in the Human Electroencephalogram Using Neural Networks, **IEEE Transaction on Biomedical Engineering**, Vol. 39, No. 12, 1305-1310, 1992

[3] Kathmann N., Uwer R., Hochrein A., Schöchlin C., Effects of different attentional demands on the accuracy of smooth pursuit eye movements, **Journal of Psychophysiology**, Vol. 30, Suppl.1, 38, 1993

[4] Matsue Y. et al., Smooth pursuit eye movements and express saccades in schizophrenic patients, **Schizophrenia Research**, 12, 121-130, 1994

[5] Pike T. and Mustard R.A., Automated recognition of corrupted arterial waveforms using neural networks, **Computers in Biology and Medicine**, Vol. 22, No. 3, 173-179, 1992

[6] Rose, R.D., Karnavas, W.J., Neural nets identify sensory receptors from somal spikes, **Brain Research**, 630, 345-348, 1993

[7] Rumelhart D.E., McClelland J.L. (Eds.), **Parallel Distributed Processing, Explorations in the Microstructure of Cognition**, Volume 1: Foundations, MIT Press,Cambridge, MA, 1986

[8] Versino, M. et al., A clinically oriented approach to smooth pursuit eye movement quantitative evaluation, **Acta Neurologica Scandinavia**, 88, 273-278, 1993

[9] Wu, F.Y. et al., Neural network approach in multichannel auditory event-related potential analysis, **International Journal of Bio-Medical Computing**, 35, 157-168, 1994

[10]Wythoff B.J. et al., Spectral Peak Verification and Recognition Using a MultilayeredNeural Network, **Analytical Cemistry**, 62, 2702-2709, 1990

EEG Signal Classification with Different Signal Representations

Charles W. Anderson
Dept. of Computer Science
Colorado State University
Fort Collins, CO 80523
anderson@cs.colostate.edu

Saikumar V. Devulapalli
Dept. of Computer Science
Colorado State University
Fort Collins, CO 80523

Erik A. Stolz
Dept. of Electrical Engineering
Colorado State University
Fort Collins, CO 80523

Abstract

If several mental states can be reliably distinguished by recognizing patterns in EEG, then a paralyzed person could communicate to a device like a wheelchair by composing sequences of these mental states. In this article, we report on a study comparing four representations of EEG signals and their classification by a two-layer neural network with sigmoid activation functions. The neural network is implemented on a CNAPS server (128 processor, SIMD architecture) by Adaptive Solutions, Inc., gaining a 100-fold decrease in training time over a Sun Sparc 10 for a large number of hidden units.

1 INTRODUCTION

Computerized analysis of EEG signals has evolved over the past three decades [5] with much of the effort directed towards a better understanding of the functioning of the brain. The work reported here has a different goal—to extract information from EEG signals with which mental states can be discriminated and thereby serve as a mode of communication for a paralyzed person. In the literature, we find two approaches towards this goal.

One approach is based on the discovery that a characteristic signal appears in the EEG approximately 300 ms following the occurrence of a relatively rare, but expected, stimulus. Such signals are referred to as event-related potentials, or ERP's. An example of how ERP's can be used to communicate with a computer is the work of [3], who used ERP's to detect which letter of the alphabet a human subject wished to select. This kind of interaction between a person and

0-7803-2739-X/95 $4.00 © 1995 IEEE

stimulus device is currently too cumbersome and slow to be useful in a real-time control application.

Much more practical would be a system for detecting patterns in normal EEG without the aid of an external stimulus device. This is the second approach, often referred to as spatial analysis, because patterns are sought in EEG signals simultaneously recorded from multiple electrodes. A number of studies have found differences in the power of the alpha band (8-13 Hz) in signals recorded from left and right hemispheres, depending on the tasks [4, 2, 9]. Asymmetries were most reliably found for motor tasks. In a recent study, [11] used a frequency-based representation of EEG from two electrodes to obtain 85% accuracy in predicting movements of the finger of a subject moving a joystick left or right.

The detection of patterns in EEG produced from normal mental states is a very difficult problem. EEG signals are recorded by surface electrodes and can contain noise as a result of electrical interference and movement of the electrodes on the scalp. Another problem is that EEG can be corrupted by eye blinks and other muscular activity that produce signals of greater magnitude than produced by cortical activity. Other problems are more cognitive in nature; the concentration of a person can vary while the person is supposedly performing a single mental task.

The work described in this article is based on previous work [7] using a Bayesian classifier trained and tested on a small subset of data recorded from subjects performing several mental tasks. We extended their study by 1) replacing the Bayesian classifier with neural networks of various sizes trained using error back-propagation, 2) using all of the data from each recording session, and 3) testing four signal representations. The objective of these experiments was to determine which of the four representations results in the best classification accuracy. If the information needed to discriminate mental state can be extracted from the unprocessed EEG signals, or at most preprocessed by projecting to a relatively small number of principal components, then we can dispense with other forms of preprocessing, such as the frequency analysis used in [7].

In one other study [10] neural networks were applied to classify EEG signals collected in a paradigm similar to that of [7]. Kohonen's SOM algorithm [8] was used to train a matrix of units to identify clusters of similar patterns and associate each cluster with a particular mental task. They trained their classifier on data for all tasks performed by one subject in one recording session and tested the resulting classifier on data from other sessions and other subjects. Most tests showed very poor classification accuracy, though the accuracy tended to be higher for some tasks, particularly the mental arithmetic task described in the next section.

In Section 2, we describe the methods used to collect, process, and classify the data. Results of experiments with data from a single subject and a pair of tasks are shown in Section 3 and conclusions are discussed in Section 4.

2 METHOD

2.1 DATA COLLECTION

All data used in this study were recorded during the previous study [7]. Selection of the set of mental tasks was guided by [4] whose results showed detectable hemispheric differences for certain tasks. Here we focus on two tasks:

Baseline—Alpha Wave Production The subject is asked to relax and try to think of nothing in particular. This will be considered to be the baseline session for alpha wave production, and other asymmetries.

Mental Arithmetic The subject is given a non-trivial multiplication problem to solve and, as in all of the tasks, is instructed not to vocalize or make overt movements while solving the problem. An example of such a task would be to multiply the numbers 49 times 78. The problems are non-repeating and designed so that an immediate answer is not apparent.

Subjects were seated in a sound-proof, dimly-lit, room. As shown in Figure 1, electrodes were placed at O_1, O_2, P_3, P_4, C_3, and C_4, standard electrode locations in the 10-20 System [6]. The electrodes were connected to Grass 7P511 amplifiers that bandpass filtered the signals at 0.1-100 Hz. The EEG signals were sampled at 250 samples per second and digitized with 12 bits of accuracy. Data was recorded from each subject for a duration of 10 seconds while the subject was performing a single task with their eyes open. Each session resulted in 250 samples/second x 10 seconds x 6 channels, or 15,000 values. For the experiments reported here, artifacts, such as eye blinks, were not removed.

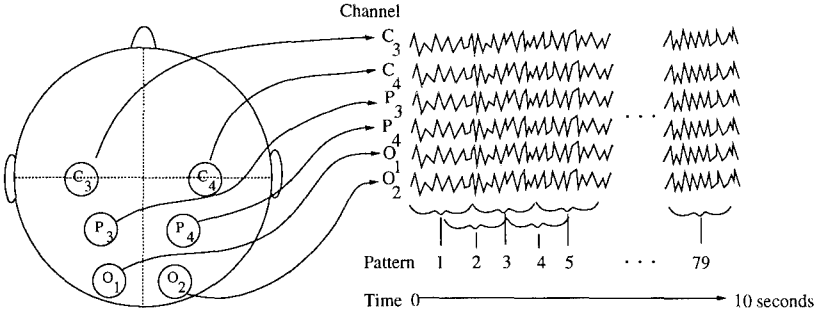


Figure 1: Location of the six surface electrodes and division of the six time series into overlapping quarter-second windows. (Data shown is fictitious.)

2.2 UNPROCESSED REPRESENTATION

In [7] it was found that quarter-second segments of the 10 second data resulted in classification accuracy approximately the same as that obtained from two-

second segments. Therefore, we divided the data into 79 overlapping, quarter-second segments consisting of 62 samples from each of the 6 channels, or 372 values per segment, as shown in Figure 1. We collected 79 such patterns from one subject performing the baseline and mental arithmetic tasks 10 times each, for a total of 1,580 patterns.

It is generally believed that frequencies above 40 Hz convey little information related to mental state. To test this, a second unprocessed representation was created by applying an FIR lowpass filter with a cut-off frequency of 40 Hz. This was followed by down-sampling by a factor of 2 to reduce the number of data samples by half, resulting in an effective sampling rate of 125 Hz. Pre-filtering is required to prevent aliasing.

2.3 K-L REPRESENTATION

When classifying high-dimensional data, equivalent or better generalization accuracy is often achieved by classifying data obtained by projecting the original data onto the first n eigenvectors, where n is much smaller than the dimensionality of the original data. To perform this Karhunen-Loève decomposition (K-L), the covariance matrix of the mean-subtracted set of the 1,580, 372-dimensional, patterns was calculated.

The number of eigenvectors to project to can be chosen empirically or determined by examining the eigenvalues. One estimate of dimensionality is the global Karhunen-Loève estimate, given by the index i for which $\lambda_i/\lambda_{max} \leq 0.01$, where the λ_i are in decreasing order for $i = 1, 2, \ldots$. For the baseline and mental arithmetic data used in our experiments, $i = 50$. Therefore, the K-L representation was formed by projecting each 372-dimensional pattern onto the first 50 eigenvalues.

2.4 FREQUENCY-BAND REPRESENTATION

In [7] features were extracted from spectral density estimates using asymmetry ratios given by $(R-L)/(R+L)$, where R is the area under the spectral density curve of a right hemisphere channel for a specific frequency band and L is defined similarly for the corresponding left hemisphere channel. These asymmetry ratios were calculated for each possible right-to-left combination of channels and for each of four frequency bands: delta (0-3 Hz), theta (4-7 Hz), alpha (8-13 Hz), and beta (14-20 Hz). This results in 36 asymmetry ratios. In addition the 24 power values themselves (R and L) were added for a total of 60 features. In the current study, the spectral density was estimated from autoregressive (AR) parameters calculated using the Burg method [12]. An AR model of order 6 is used here since it yielded good results in [7].

2.5 CLASSIFICATION

Classification was performed by fully-connected, feed-forward neural networks with single hidden layers of varying size. All units use an asymmetric sigmoid

activation function. The work reported here involved a single pair of tasks, the mental arithmetic and baseline tasks, so the networks require only one output unit. An output value greater than 0.5 represents the mental arithmetic task and a value less than 0.5 represents the baseline task. The target value for the output during training is set to 0.9 for mental arithmetic and 0.1 for baseline.

Our data was obtained from ten different 10-second recording sessions for each task, so it naturally divides into ten parts. We trained on eight parts, used a ninth part as a cross-validation, and tested on the tenth part. These three partitions were used as follows. The network is trained on the training set for a given number of epochs. After each epoch, the RMS error for the current network is calculated for the cross-validation set. After training, the weights for the epoch at which the cross-validation error was lowest are obtained and the RMS error of the network using these weights is calculated for the test set. This test error is the estimate of how well the network will classify novel data. Thus, we train, cross-validate and test on data from different recording sessions. With ten parts, we can choose 90 different pairs for cross-validation and testing. The results in the next section are averages over 90 runs constructed in this way.

Training was performed with the `bnlib` error back-propagation library on a 128-node CNAPS Server II from Adaptive Solutions, Inc.. [1] present details of the implementation. The CNAPS implementation ran for 4 minutes to train a 40 hidden unit net for 1,000 epochs on the unprocessed data representation, whereas a Sparc 10 required about 9 hours, over 100 times longer.

Networks were trained on the training data until the squared error for 200 successive epochs did not decrease. The epoch at which the squared error on the cross-validation data was at a minimum was identified, and the network was retrained to this epoch, starting with the same initial weights. At this point the network's weights were written to a file. All of this was performed on the CNAPS hardware. The network was evaluated on a Sun Sparc by calculating the number of test patterns classified correctly. A pattern is said to be correctly classified if the network's output falls on the correct side of 0.5.

3 RESULTS

Table 1 shows the results of all classification experiments as the average percent of test patterns classified correctly, out of 158 patterns for all but the unprocessed, 125 Hz, representation, which consisted of 154 patterns. This table includes 90% confidence intervals, based on 90 repetitions.

Clearly the best classification accuracy is achieved with the frequency-band representation, giving an average accuracy of about 74% for a network with 40 hidden units, though the accuracy varies little for other network sizes, including a network with a single hidden unit. This shows that the dependence between the signals in the various representations and the correct classification is primarily linear on average. The performance of the raw and K-L representations is significantly lower, ranging from 50% to 53%. These results suggest that the energy within standard frequency bands is more useful in discriminating the two

Hidden Units	Unprocessed (250 Hz)	Unprocessed (125 Hz)	K-L	Frequency-Bands
40	53.2 ± 0.6	51.1 ± 0.6	51.7 ± 0.6	73.9 ± 0.7
5	52.8 ± 0.6	52.1 ± 0.8	51.8 ± 0.7	72.9 ± 0.6
1	52.1 ± 0.5	52.1 ± 0.7	50.4 ± 0.5	73.1 ± 0.6

Table 1: Percent of test patterns classified correctly for different sized networks and different signal representations.

mental tasks than is the raw data, or dimensionally-reduced data. This hypothesis must be tested by further experimentation. It appears that the performance of the raw and K-L representations is increasing with network size, but the differences are not statistically significant. We have not yet tried networks with more than 80 hidden units.

The results reported above dealt with each quarter-second segment in isolation. One way to consider more than one interval is to average the output of the network over successive segments. We have investigated this using the frequency-band representation for one combination of training, cross-validation, testing data and obtained the following preliminary results.

Table 2 shows how the number of successive quarter-second segments over which the network output is averaged affects classification accuracy. For the mental arithmetic task, only five segments are needed to get 100% classification accuracy for the single test data set considered. However, 28 successive segments were required to get 100% accuracy for the baseline task. Thus, correct discrimination between baseline and mental arithmetic could be achieved by averaging over 28 successive segments, or about 3.6 seconds (1/4 + 27 x 1/8).

One hypothesis for this difference is that a subject performing the baseline task is less focused on a particular set of mental states than is a subject performing the mental arithmetic task. Plots of the network output, averaged over five segments, is shown for both tasks in Figure 2. Recall that the target for the network's output for the mental arithmetic task is 0.9, while for the baseline task it is 0.1. These plots suggest that while performing the baseline task, the subject's mental state varied considerably. During the middle of the 10 second recording session, the subject's mental state more resembled the mental arithmetic state than the baseline state, using the frequency-bands representation.

When this process of averaging over successive segments is repeated for different groupings of the data into training, validation, and test sets, we occasionally find cases for which the classification accuracy does not converge on 100% as more segments are averaged. We are currently performing experiments to determine the average accuracy over different groupings of the data. Preliminary results indicate that the accuracy on average converges near 90% accuracy as the averaging is extended to cover full 10 second recording period.

Number of Successive Segments	Mental Arithmetic Task	Baseline Task
1	79%	67%
2	88	76
3	92	84
4	93	78
5	100	81
.	.	.
.	.	.
.	.	.
27	100	98
28	100	100
.	.	.
.	.	.
.	.	.

Table 2: Percent correct when network output is averaged over successive segments.

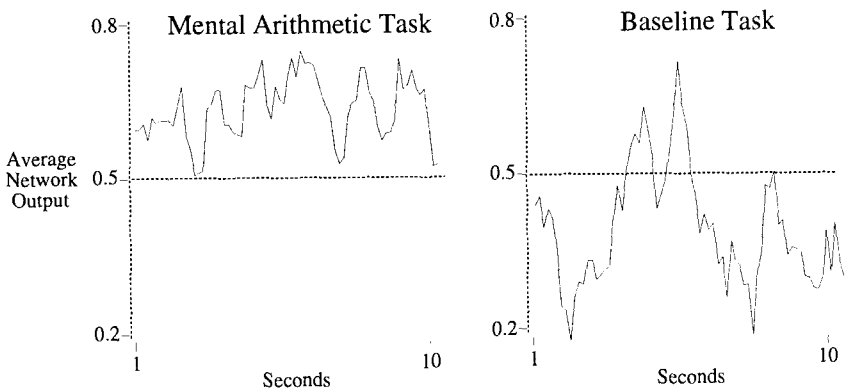


Figure 2: Network output using frequency-band representation averaged over five successive segments.

4 CONCLUSIONS

The frequency-based representation resulted in significantly more accurate classification than the unprocessed or K-L representations. A second conclusion from this study is the utility of the parallel implementation of the error back-propagation algorithm. A much greater number of network sizes and initial weight vectors could be evaluated on the CNAPS server than could be completed in a comparable amount of time on a serial machine.

The practicality of this result is limited by the long period, 3.6 seconds, that was required to get reliable classification. This is too long for use in controlling a wheelchair. However, this estimated time period is conservative, because averages were over quarter-second segments that overlapped only by an eighth of a second. Instead, the network output could be calculated for all quarter-second segments, with successive segments overlapping by one time sample, or $1/250^{th}$ of a second. This might reduce the length of the data sample needed to reliably discriminate tasks.

We are currently studying other representations, including AR model coefficients, Hidden Markov Models, LVQ, and "bottleneck" networks to perform nonlinear principal components analysis. Generalization studies across subjects with additional tasks are being performed.

Acknowledgements

The authors would like to thank Jorge Aunon and Zachary Keirn for providing the EEG data, Ed Orosz for installing and interpreting the data. This research was funded by the National Science Foundation through grant IRI-9202100.

References

[1] Charles W. Anderson, Sai Devulapalli, and Erik Stolz. Determining mental state from EEG signals using neural networks. *Scientific Programming*, to appear.

[2] J. C. Doyle, R. Ornstein, and D. Galin. Lateral specialization of cognitive mode: II EEG frequency analysis. *Psychophysiology*, 11:567–578, 1974.

[3] L. A. Farwell and E. Donchin. Talking off the top of your head: Toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70:510–523, 1988.

[4] D. Galin and R. E. Ornstein. Hemispheric specialization and the duality of consciousness. *Human Behavior and Brain Function*, 1973.

[5] Alan S. Gevins. Overview of computer analysis. In A. S. Gevins and A. Rémond, editors, *Methods of Analysis of Brain Electrical and Magnetic Signals*, volume 1 of *EEG Handbook*, chapter 3, pages 31–83. Elsevier Science Publishers Biomedical Division, 1987.

[6] H. Jasper. The ten twenty electrode system of the international federation. *Electroencephalographic Clinical Neurophysiology*, 10:371–375, 1988.

[7] Zachary A. Keirn and Jorge I. Aunon. A new mode of communication between man and his surroundings. *IEEE Transactions on Biomedical Engineering*, 37(12):1209–1214, December 1990.

[8] T. Kohonen. *Self-Organization and Associative Memory*. Springer-Verlag, Berlin, 1984.

[9] R. H. Kraft, O. R. Mitchell, M. L. Languis, and G. H. Wheatley. Hemispheric asymmetry during six- to eight-year old performance of piagetian conservation and reading tasks. *Neuropsychologia*, 22:637–643, 1984.

[10] Shiao-Lin Lin, Yi-Jean Tsai, and Cheng-Yuan Liou. Conscious mental tasks and their EEG signals. *Medical & Biological Engineering & Computing*, 31:421–425, 1993.

[11] G. Pfurtscheller, D. Flotzinger, and J. Kalcher. Brain-computer interface - a new communication device for handicapped persons. *Journal of Microcomputer Applications*, 16(3):293–299, 1993.

[12] J. G. Proakis and D. G. Manolakis. *Digital Signal Processing*. MacMillan, New York, 1992.

Design and Evaluation of Neural Classifiers Application to Skin Lesion Classification

Mads Hintz-Madsen, Lars Kai Hansen and Jan Larsen
CONNECT, Electronics Institute, build. 349
Technical University of Denmark,
DK-2800 Lyngby, Denmark
emails: hintz, lkhansen, jlarsen@ei.dtu.dk

Eric Olesen and Krzysztof T. Drzewiecki
Dept. of Reconstructive Surgery S,
The National Hospital of Denmark,
Rigshospitalet, Blegdamsvej 9,
DK-2100 Copenhagen, Denmark

Abstract

We address design and evaluation of neural classifiers for the problem of skin lesion classification. By using Gauss Newton optimization for the entropic cost function in conjunction with pruning by Optimal Brain Damage and a new test error estimate, we show that this scheme is capable of optimizing the architecture of neural classifiers. Furthermore, error-reject tradeoff theory indicates, that the resulting neural classifiers for the skin lesion classification problem are near-optimal.

1 INTRODUCTION

Melanoma is the most lethal of skin cancers. However, patients may be saved from this life threatening cancer if their lesion is detected at an early stage. Computer imaging may assist and improve the detection of such early lesions. The "State of the art" in this field was recently reviewed in an editorial in the journal "Computerized Medical Imaging and Graphics" [1]. Although applied to the problem of skin lesion classification, the main objective of this paper is to introduce and apply a new methodology for optimization of neural classifiers. The methods applied may be considered as an extension of the *Designer Net* time series processing tool [8, 9] to the realm of classifiers.

0-7803-2739-X/95 $4.00 © 1995 IEEE

In particular we derive the Hessian for the so-called *entropic* cost function and apply the Hessian for Gauss Newton second order optimization and for estimation of weight saliency for use in Optimal Brain Damage pruning. A key ingredient of the proposed method is a new test error estimate for the entropic cost function [2]. The test error estimate enables us to select the optimal network within a nested family of pruned networks.

2 NEURAL CLASSIFIERS

The aim in classification is to model the probability of classification, $P(y|\mathbf{x})$, of a given input vector $\mathbf{x}$ where y is the class label. In the context of skin lesion classification the input vector for the classifier is formed from n_I feature measurements on a given skin lesion. If provided with a training set, D, consisting of p input-output pairs[1]: $(\mathbf{x}_\alpha, y_\alpha)$, where $\mathbf{x} \in \mathcal{R}^{n_I}$ and $y = \pm 1$, the likelihood of the neural classifier, $F_u(\mathbf{x})$, with parameters (weights) u is given by [7],

$$P(D|u) = \prod_{\alpha=1}^{p} \left(\frac{1+F_u(\mathbf{x}_\alpha)}{2}\right)^{\frac{1+y_\alpha}{2}} \left(\frac{1-F_u(\mathbf{x}_\alpha)}{2}\right)^{\frac{1-y_\alpha}{2}} \tag{1}$$

Hence, for the well trained network, $\frac{1}{2}(1+F_u(\mathbf{x})) \sim P(y|\mathbf{x})$. Training is based on minimization of the negative log-likelihood:

$$E(u) = -\log P(D|u) = \sum_{\alpha=1}^{p} \epsilon\left(\mathbf{x}_\alpha, y_\alpha, u\right) \tag{2}$$

with the error measure given by

$$\epsilon\left(\mathbf{x}_\alpha, y_\alpha, u\right) = -\frac{1+y_\alpha}{2} \log\left(\frac{1+F_u(\mathbf{x}_\alpha)}{2}\right) - \frac{1-y_\alpha}{2} \log\left(\frac{1-F_u(\mathbf{x}_\alpha)}{2}\right) \tag{3}$$

The cost function (2) is in turn recognized as the entropic cost function (see, e.g., [6]). In order to eliminate overfitting, and for numerical convenience, we often augment the cost function by a weight decay term corresponding to minimizing instead the negative log-posterior,

$$C(u) = E(u) - \log P(u). \tag{4}$$

The log-prior, $-\log P(u)$, is conveniently chosen to be a quadratic form in the weight parameters, e.g, representing a simple weight decay.

[1] We discuss binary classification for convenience.

2.1 Generalization

For a given network u the generalization or test error may be defined as,

$$\mathrm{E}_{\mathrm{test}}(u) = \int D\mathbf{x}dyP(\mathbf{x},y)\epsilon\left(\mathbf{x},y,u\right). \tag{5}$$

Here, the "true" underlying distribution of examples, $P(\mathbf{x},y)$, need not be a singular measure, - we do allow noisy classifications, i.e., contradicting labels for the same input, corresponding to $|F_u(\mathbf{x})| < 1$. Since the generalization error involves an average over all possible patterns it is not observable, but may be estimated by invoking additional statistical assumptions. The training set error is given by

$$\mathrm{E}_{\mathrm{train}}(u) = \frac{1}{p}\sum_{\alpha=1}^{p}\epsilon\left(\mathbf{x}_\alpha,y_\alpha,u\right), \tag{6}$$

hence, the average entropic cost on the training set. In the limit, $p \to \infty$, $\mathrm{E}_{\mathrm{train}}(u) \to \mathrm{E}_{\mathrm{test}}(u)$; asymptotic theory quantifies this limiting behavior.

While the above quantifies the generalization of a single classifier we are, in fact, interested in the typical behavior of the test error. Therefore we compute the training set averaged quantities:

$$\langle \mathrm{E}_{\mathrm{test}} \rangle = \int DuP_p(u)\mathrm{E}_{\mathrm{test}}(u) \tag{7}$$

$$\langle \mathrm{E}_{\mathrm{train}} \rangle = \int DuP_p(u)\mathrm{E}_{\mathrm{train}}(u). \tag{8}$$

Here, $P_p(u)$, is the distribution of optimal networks obtained by minimizing the cost function based on randomly selected samples of size p from the example distribution $P(\mathbf{x},y)$. $P_p(u)$ could be thought of as the distribution of network parameters in an ideal cross validation ensemble.

It is possible to show that $P_p(u)$ is asymptotically *normal* with $P_p(u) \sim N(u^*,\mathbf{\Sigma})$, where u^* are the parameters that minimize the regularized test error, $\mathrm{E}_{\mathrm{test}}(u) - \frac{1}{p}\log P(u)$ [2].

If we now choose the particular network architecture:

$$F_u(\mathbf{x}) = \tanh\left(\phi_u(\mathbf{x})\right), \tag{9}$$

$$\phi_u(\mathbf{x}) = \sum_{j=0}^{n_H} W_j h_j(\mathbf{x}), \tag{10}$$

$$h_j(\mathbf{x}) = \tanh\left(\sum_{k=0}^{n_I} w_{j,k}x_k\right), \tag{11}$$

with n_H hidden units, n_I input units, and parameters $u = (w, W)$, the covariance matrix has the particularly simple form,

$$\mathbf{\Sigma} = (\mathbf{H} + \mathbf{R})^{-1} \tag{12}$$

$$\mathbf{H}_{ii'} \approx \int D\mathbf{x}dyP(\mathbf{x}, y)\left(1 - F^2_{u^*}(\mathbf{x})\right)\frac{\partial\phi_{u^*}(\mathbf{x})}{\partial u_i}\frac{\partial\phi_{u^*}(\mathbf{x})}{\partial u_{i'}} \tag{13}$$

where we further simplified the Hessian using the Levenberg-Marquart approximation [10]. $\mathbf{R}$ is the second derivative matrix of the regularization term. The quantity $\mathbf{H}_{ii'}$ may be approximated by the Hessian of the entropic cost function,

$$\mathbf{H}_{ii'} \approx \mathbf{H}^p_{ii'} \equiv \frac{1}{p}\sum_{\alpha=1}^{p}\left(1 - F^2_{u(D)}(\mathbf{x}_\alpha)\right)\frac{\partial\phi_{u(D)}(\mathbf{x}_\alpha)}{\partial u_i}\frac{\partial\phi_{u(D)}(\mathbf{x}_\alpha)}{\partial u_{i'}} \tag{14}$$

where $u(D)$ are the best weights found for the actual training set D.

With the asymptotic form of the cross validation ensemble distribution we are in a position to compute the averaged quantities in (7) and (8), and find,

$$\langle \mathrm{E}_{\mathrm{test}} \rangle = \epsilon_0 + \frac{N_{\mathrm{eff}}}{2p} \tag{15}$$

$$\langle \mathrm{E}_{\mathrm{train}} \rangle = \epsilon_0 - \frac{N_{\mathrm{eff}}}{2p}. \tag{16}$$

where the effective number of parameters is $N_{\mathrm{eff}} = \mathrm{Tr}[\mathbf{H}\mathbf{\Sigma}]$, and the asymptotic test error given by the average entropic cost of the teacher parameters is,

$$\epsilon_0 = \int D\mathbf{x}dyP(\mathbf{x}, y)\epsilon\left(\mathbf{x}, y, u^*\right). \tag{17}$$

One may interpret ϵ_0 as a "noise level" for the classifier; if the classifier is "crisp", i.e., if $|F_{u^*}(\mathbf{x})| \approx 1$ for almost all inputs, $\mathbf{x}$, $\epsilon_0 \approx 0$. On the other hand, if the classifier is "fuzzy", i.e., $|F_{u^*}(\mathbf{x})| \approx 0$ for almost all inputs, $\epsilon_0 \approx \log 2$.

In a practical situation one only has access to a single training set, and the two averages may be combined to provide a test error estimate,

$$\widehat{\langle \mathrm{E}_{\mathrm{test}} \rangle} = \mathrm{E}_{\mathrm{train}}(u(D)) + \frac{N_{\mathrm{eff}}}{p} \tag{18}$$

where the average training error is estimated using the actual training error. The estimator (18) may be used to select the optimal network, e.g., among a family of pruned networks, hence, be used as a *pruning stop criterion* similarly to the criterion previously developed for regression type problems in [8, 9].

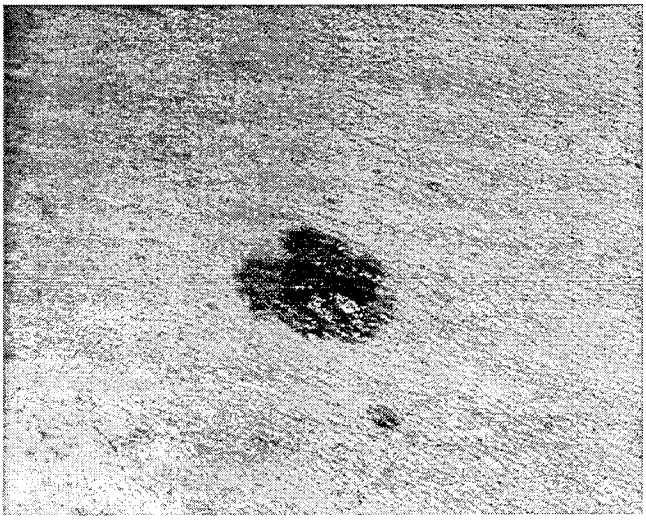


Figure 1: Skin lesion showing the characteristical large variations in texture and coloring associated with melanoma (original is in color).

2.2 Pruning

The architecture (11) may be subjected to pruning by *Optimal Brain Damage* (OBD), developed by Le Cun and coworkers [10]. In OBD the parameters of a network are ranked for pruning according to their importance for the training error. If the importance is estimated using a second order expansion of the training error around its minimum, we find the weight *saliency*:

$$s_i = \frac{1}{2}\mathbf{H}^p_{ii}u_i^2(D), \tag{19}$$

where the Hessian is given as in (14).

3 EXPERIMENTAL

We evaluated the optimizing scheme and the classifier theory on the skin lesion classification problem. This classification involves the classes: the malignant, the premalignant and the benign skin lesions. However we focused on the classification problem of separating malignant lesions from benign. An example of a malignant lesion is shown in figure 1 (original is in color). Generic characteristics of melanoma are large variations in coloring, absence or presence of certain texture features and irregular boundaries. The samples

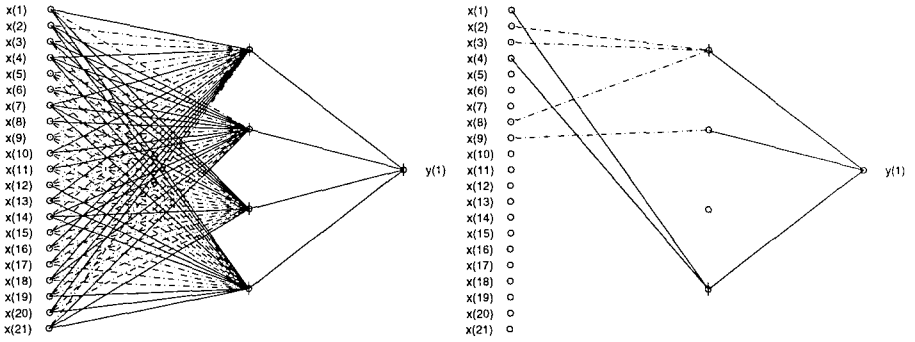


Figure 2: Left: Fully connected network for classification of skin lesions, the 21 inputs represent color and texture features. Right: Architecture of the optimal pruned network. Note that only a few of the available inputs are used by the net. The used features include first and second order texture statistics.

used in the present study were all taken from a photographic library of skin lesions, that were collected at the Dept. of Recontructive and Plastic Surgery at the National Hospital of Denmark, and which all had been considered potentially malignant. Hence, the sampling of the benign group is rather biased. In previous studies [3, 4] it is unclear how the sampling of the classes has been done, making comparisons difficult. The data set consisting of a total of 160 images was split into a training set of 120 images and a test set of the remaining 40 images each containing an even split of the two classes. As input to the network a group of 21 features incorporating color and texture statistics were selected. Boundary features were not incorporated, since they were found not to contribute significantly to the classification of the two classes. We believe that this might be a result of the biased sampling, since most of our benign samples, in fact, have irregular boundaries.

A fully connected network with 4 hidden units was chosen initially, see the left panel of figure 2. In figure 3 we show the development of training and test errors during training of the fully connected network; the weight decay parameter was set to 0.8. Next we pruned the network iteratively according to the OBD saliency ranking, pruning 5% of the remaining weights per iteration. After each pruning session the remaining weights were retrained for 30 epochs. For the resulting nested family of networks training errors, test errors and *estimated* test errors were computed. We also computed the sample standard deviation of the test errors. The development of these error measures during pruning are shown in figure 4. The estimated test error was used to stop the pruning and the architecture of the selected network is shown in the right panel of figure 2. The pruning process is successful in identifying a much smaller network with better test performance than the fully connected network. Furthermore, the variance of the test error of the networks in the

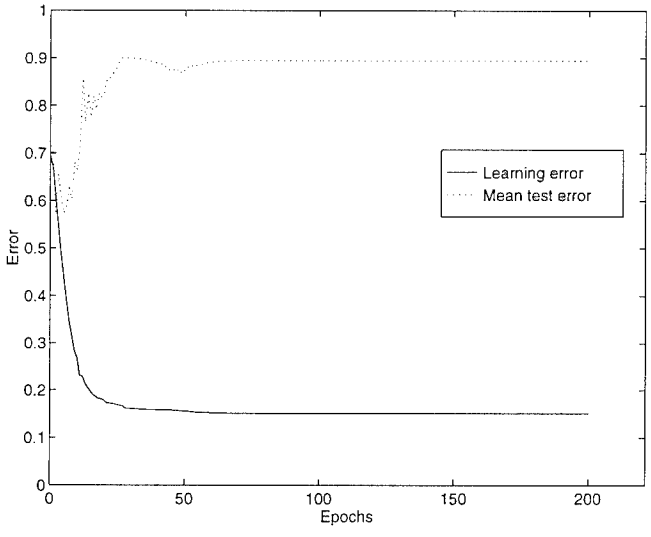


Figure 3: Development of the entropic test and training errors for the fully connected network. Note that the test error shows significant overtraining.

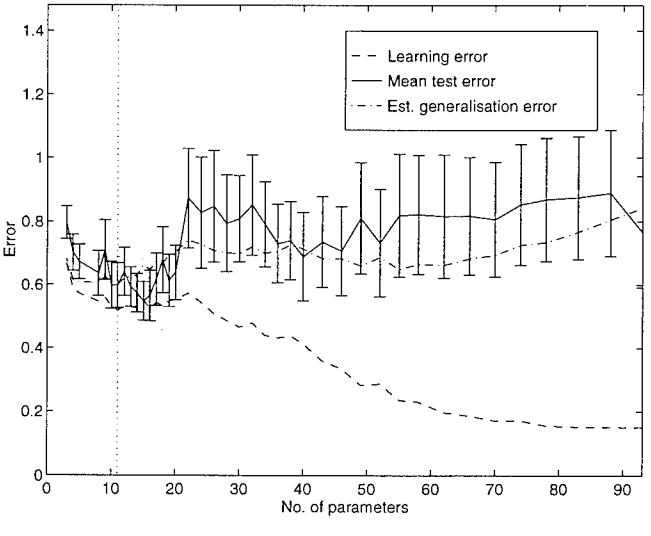


Figure 4: Development of training error, test error, and estimated test error. The vertical dotted line indicates the location of the optimal network according to the estimated test error. The error bars on the test set errors indicate sample standard deviations. Note the close correspondence between the estimated and empirical test errors.

vicinity of the optimal network is significantly lower; hence we can be more confident in the properties of these networks. Although the entropic test error is indeed smaller for the pruned network than for the fully connected network, it is of interest to see what the *classification* error of the two networks are. Following Bayes decision theory we selected the class label according to the sign of the network output when converting probabilities to classifications. In this way we found, that the pruned network classified 74% of the lesions correctly on the training set and 66% on the independent test set. The fully connected network classified 98% correctly on the training set and 66% on the test set, ie. when converted to classifications the performances of the two networks are similar.

For comparison we have performed a *k-Nearest Neighbor* (k-N-N) analysis of the data sets. Within k-N-N a pattern is classified according to a simple majority vote among its k nearest neighbors (using the simple Euclidean metric). The training error may be computed from the training set by including the actual pattern in the vote. A *leave-one-out* "validation" error on the training set may be computed by *excluding* the actual pattern from the neighbor vote. Finally, the test patterns may be classified by voting among the k nearest neighbors found among the training patterns. Using the validation error we found that $k = 3$ was optimal for this data set. The training error for the 3−N-N scheme was found to be 83%, while the test error was 63%, ie., the network classifiers have slightly better performance than the k-N-N standard algorithm.

Since the neural classifier is trained to produce classification probabilities (and not only Bayes classifications), we can inspect the error-reject trade-off induced by a reject threshold on the probability (rejecting decisions for which $|F_u(\mathbf{x})| < \tau$). The error-reject trade-off was recently discussed in [5]. Denoting the classification error rate, at reject rate R, by $E(R)$, it was argued that near-optimal binary classifiers should obey the relation (for low reject rates),

$$E(R) = E(0) - (1/2 - E(0)) \cdot R \tag{20}$$

Since $E(0) \approx 0.35$ for the present system, we expect the slope of the trade-off curve (the efficiency of the reject mechanism) to be $\gamma \equiv 1/2 - E(0) \approx 0.15$. This is indeed confirmed by the actual error-reject trade-off curve presented in figure 5.

4 CONCLUSION

We have developed a methodology for design and evaluation of neural classifiers. The approach was applied to the problem of skin lesion classification. The new test error estimator for classifiers was shown to be able to produce

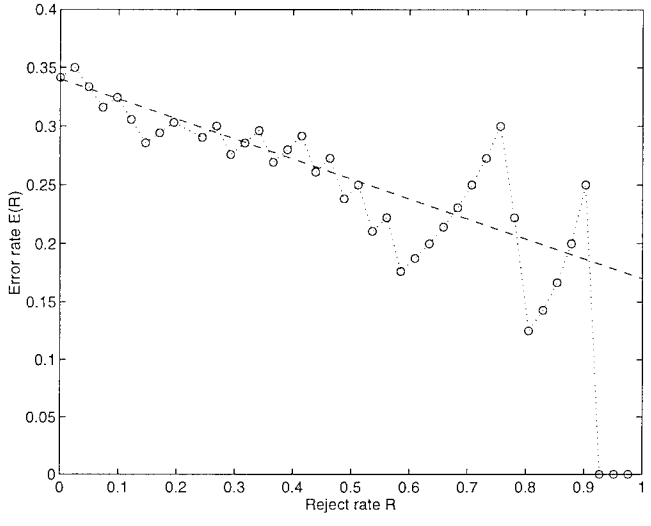


Figure 5: The error rate $E(R)$ versus reject rate R computed on the test set examples. The initial slope of the trade-off curve is approximately given by $1/2 - E(0)$ in line with a recent theoretical analysis of probability driven reject mechanisms in near-optimal binary classifiers.

valid estimates of the empirical test error and could be used to select optimal networks among a family of pruned networks. The optimal network for the skin lesion classification problem based its classification on texture statistics. Currently, the aim is to establish more empirical data for validation of the neural classifier design approach and to compare our classification results with other recent neural network approaches for solving the skin lesion classification problem.

ACKNOWLEDGMENT

This research is supported by the Danish Research Councils for the Natural and Technical Sciences through the Danish Computational Neural Network Center. MHM wishes to acknowledge a generous donation from Novo Nordisk A/S Academic Affairs. JL thanks the Radio-Parts Foundation for financial support.

REFERENCES

[1] W.V. Stoecker, R.H. Moss, F. Ercal and S.E. Umbaugh: "Editorial: Digital Imaging in Dermatology". Computerized Medical Imaging and Graphics **16**, 145-150, (1992).

[2] L.K. Hansen et al.: "Asymptotic generalization in neural classifiers". In preparation (1995).

[3] A. Kjoelen, S.E. Umbaugh, R.H. Moss and W.V. Stoecker: "Artificial Intelligence Applied to Detection of Melanoma". IEEE Engineering in Medicine and Biology, 15th Annual Conference, Vol. 2, 604-605, (1993).

[4] F. Ercal, A. Chawla, W.V. Stoecker, H-C. Lee and R.H. Moss: "Neural Network Diagnosis of Malignant Melanoma from Color Images". IEEE Transactions on Biomedical Engineering, Vol. 41, No. 9, September (1994).

[5] L.K. Hansen, Chr. Liisberg, and P. Salamon: "The Error Reject Trade-off". Submitted for publication. Available by anonymous ftp: eivind.ei.du.dk/dist/hansen.reject.ps.Z.

[6] J. Hertz, A. Krogh and R.G. Palmer: "Introduction to the Theory of Neural Computation", Addison Wesley, New York (1991).

[7] D. MacKay: "The Evidence Framework Applied to Classifier Networks". Neural Computation **4**, 720-736, (1992).

[8] C. Svarer, L.K. Hansen and J. Larsen: "On Design and Evaluation of Tapped-Delay Neural Network Architectures" The 1993 IEEE Int. Conference on Neural Networks, San Francisco. Eds. H.R. Berenji et al., 45-51, (1993).

[9] C. Svarer, L.K. Hansen, J. Larsen and C.E. Rasmussen: "Designer Networks for Time Series Processing". The 1993 IEEE Workshop on Neural Networks for Signal Processing (NNSP'93) Baltimore. Eds. C.A. Kamm et al., 78-87, (1993).

[10] Y.L. Cun, J.S. Denker and S.A. Solla: "Optimal Brain Damage" In Advances in Neural Information Processing Systems II (Denver 1989), ed. D.S. Touretzky, 396-404. San Mateo: Morgan Kaufmann, (1989).

A Study of the Application of the CMAC Artificial Neural Network to the Problem of Gas Sensor Array Calibration[†]

Parag M. Bajaria
Bruce E. Segee, PhD
University of Maine
5708 Barrows Hall,
Orono, ME 04469
Telephone:(207) 581-2212
Fax:(207) 581-2220
E-Mail: segee@eece.maine.edu

ABSTRACT

This paper explores the application of the Cerebeller Model Arithmetic Computer (CMAC) artificial neural network to the analysis of multicomponent gas mixtures using an array of eight nonselective, nonlinear and noisy Taguchi gas sensors. Various network parameters that affect the performance of the CMAC are discussed. Results of the analysis of three gas component mixtures are presented.

1.0 INTRODUCTION

In this paper, it is our interest to evaluate the performance of neural networks for the calibration of chemical gas sensors. The specific problem under investigation is to calibrate eight non-selective, nonlinear and noisy Tauguchi gas sensors to analyze multicomponent gas mixtures using the CMAC (Cerebeller Model Arithmetic Computer) artificial neural network. The goal is to accurately predict the concentrations (in parts per million) of each of the gases in the mixture.

This study has two purposes: (1) to develop a practical approach based on neural network computing for calibrating chemical sensors, and (2) to study the effects of network parameters on this class of multidimensional function approximation problem. The paper is organized as follows: First a statement of the problem is described, following which design of a neural network is described. Next, an analysis of different network configurations is presented. Finally, the paper concludes with a discussion of results and findings.

[†] This work was sponsored by NSF grant number ECS-9309012

0-7803-2739-X/95 $4.00 © 1995 IEEE

2.0 STATEMENT OF THE PROBLEM

The response of a chemical gas sensor varies with changes in gas concentrations. Although these sensors are sensitive, their non-selective and nonlinear response characteristics are important limitations to the analysis of multicomponent gas mixtures. Due to the above limitations, mathematically, it is very difficult to model the responses of a sensor. Also, the use of an array of sensors and multicomponent gas mixtures makes the mathematical modeling so complex that writing meaningful model equations becomes impossible.

Artificial neural networks are among the most promising nonlinear methods for analysis of multidimensional nonlinear data. In the past work has been done on calibrating array of chemical gas sensors using the Multilayer Perceptron artificial neural network [3][5]. However Multilayer Perceptrons have the following disadvantages :

1)They require many iterations to converge;

2)They are not suitable for real time training;

3)They require a large number of computations per iteration;

4)They have an error surface that can have relative minima;

5)They are not amenable to incremental training.

The CMAC has the advantages of the following properties :

1)Local generalization;

2)Rapid algorithmic computation based on LMS training;

3)Incremental training.

3.0 NETWORK ARCHITECTURE

The CMAC network, first proposed by Albus [1] and later expanded by Miller[4] provides a computationally efficient, memory efficient, localized basis function neural network. While it is by no means necessary, it is common for CMAC inputs, outputs, and weights all be integer values.

The diagram shown in Figure 1 from Miller [2] is a model of how a CMAC operates. The training and testing vectors are taken from the input space. Each input can be thought of as a point in the N dimensional input space. The input space is theoretically infinite. In practice, since fixed precision integer values are used, the input space is finite but very large. Fortunately, for virtually any problem

we are concerned with only a small portion of the input space. Whether this small portion is a single region, or multiple regions does not matter.

The CMAC, based on scaling, and quantization maps input values into a conceptual space. In this conceptual space (denoted A in the figure) are the receptive fields of the network. With CMAC, the receptive fields in the conceptual space are of fixed size, and arranged along a lattice structure. In the CMAC proposed by Albus, the receptive fields were organized along the main diagonals of hypercubes in the conceptual space, and the receptive field shape was rectangular. In the CMAC of Miller [4] the lattice structure and the receptive field shape can be chosen.

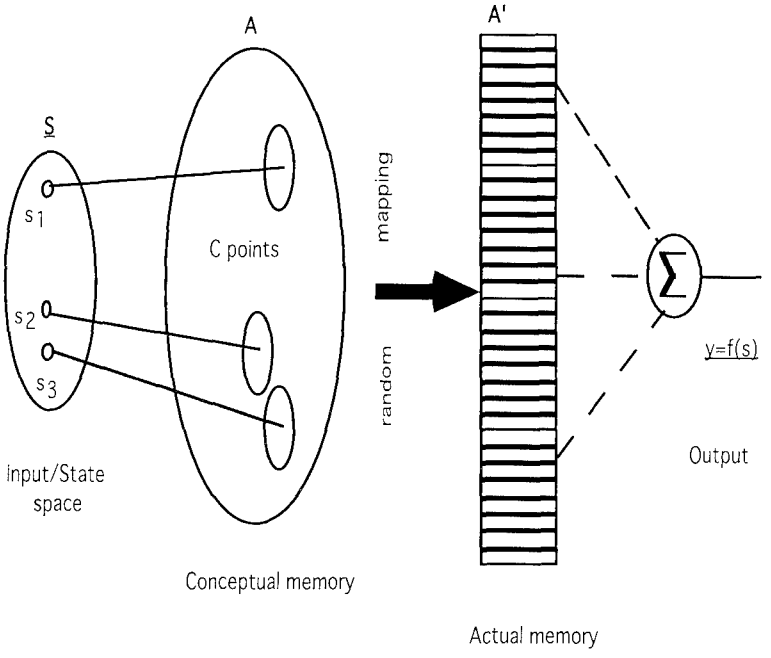


Figure 1: Block diagram of the CMAC

A key point to note with CMAC is that while the entire input space may be very large, in practice only a small subset of the space will actually be visited. Thus, the large conceptual memory can be supported by a relatively small number of randomly-mapped actual weights. It is not uncommon, for instance to have a conceptual space consisting of many billions of points mapped to a physical memory consisting of only a few thousand weights. Obviously, for such a mapping to work it is necessary for at least one of the following to be true. Either a single physical weight maps to many conceptual points, or only a small region of the conceptual memory can be utilized. When physical weights are used in more than one region of the space, this is called a collision. Miller's CMAC code allows collisions to be disabled.

The mapping from the conceptual space to the physical memory maintains key relationships between the actual weights and points in the conceptual memory. Specifically, points in the conceptual space map to a fixed number of weights in the actual memory. This number is the generalization parameter. Points in the conceptual memory that are close together have a number of physical weights in common. Points in the conceptual memory that are further apart have fewer weights in common. Points in the conceptual memory that are far apart have no weights in common (unless a collision has occurred).

4.0 NETWORK PARAMETERS:

All sensor data discussed in this paper was obtained from University of Washington[3]. The data is the response of an array of eight sensors to concentrations of three gases toluene, trichoroethylene, and acetone. There were 100 data points. These were divided into 75 training and 25 testing points. All the data points were multiplied by a suitable factor to bring data into the integer range. The software for CMAC was obtained from University of New Hampshire and was customized for this problem.

Figures 2 to 5 show how the change of one of the network parameters, while keeping the other parameters unchanged, affects the network error over the testing set.

Network parameters that affect the output of network are discussed in the sections that follow.

4.1 Input Scaling

Responses of sensors vary within the range of 0 to 1. Input scaling is necessary in this case in order to bring the values to the integer range. All the inputs were multiplied by a factor of 10,000. Inputs were further multiplied by a small value, as shown in column 2 of Table 1, to fine tune the scaling. A change in scaling factor from 1 to 2 increased the root mean square error (RMSE) on average by about 10 ppm. The lowest RMSE for toluene was obtained with a scaling factor of 0.8. However RMSE of other gases increased. The lowest RMSE for trichoroethylene and acetone were obtained with scaling factors of 1.05 and 1.1 respectively. While changing input scaling factors all the other network parameters were kept constant.

Generalization	Input Scaling factor	Output Scaling factor	RMS error in ppm			Beta	Shape of Receptive field
			Toluene	*TCE*	*Acetone*		
256	1	1	18.96	16.99	14.51	1	linear
256	2	1	33.52	26.25	18.72	1	linear
256	0.8	1	**16.80**	23.85	19.12	1	linear
256	1.05	1	18.08	**15.49**	12.98	1	linear
256	1.1	1	21.25	17.26	**12.98**	1	linear

Table 1: The effect of input scaling with other parameters held constant.

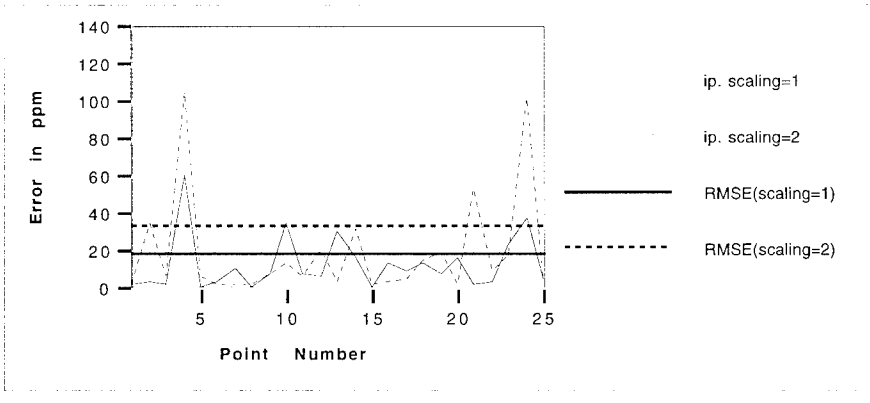


Figure 2: Comparison of error due to different input scaling factor

4.2 Output Scaling Factor

Values of gas concentrations varied from 6 to 400 ppm. These were represented in the data set as values from 0.006 to 0.400. Since the CMAC produces integer values, all the data set values were multiplied by a scaling factor of 10,000 to bring them into the integer range. As expected changes in output scaling factor did not affect RMSE of the testing set.

4.3 Generalization

The generalization parameter indicates the number of receptive fields excited for any input. The generalization parameter in the CMAC software (UNH_CMAC Version 2.1) must be a power of 2. Careful selection of generalization parameter is required. Lowest RMSE was achieved at generalization of 256. Increasing and decreasing generalization affected RMSE of all the gases significantly. Increasing the generalization parameter also increases computational time and memory usage. The effect of the generalization parameter is summarized in Table 2.

Generalization	Input Scaling factor	Output Scaling factor	RMS error in ppm			Beta	Shape of Receptive field
			Toluene	*TCE*	*Acetone*		
256	1	1	**18.96**	**16.99**	**14.51**	1	linear
128	1	1	39.74	28.66	24.48	1	linear
512	1	1	19.50	33.51	31.80	1	linear

Table 2: The effect of generalization with other parameters held constant

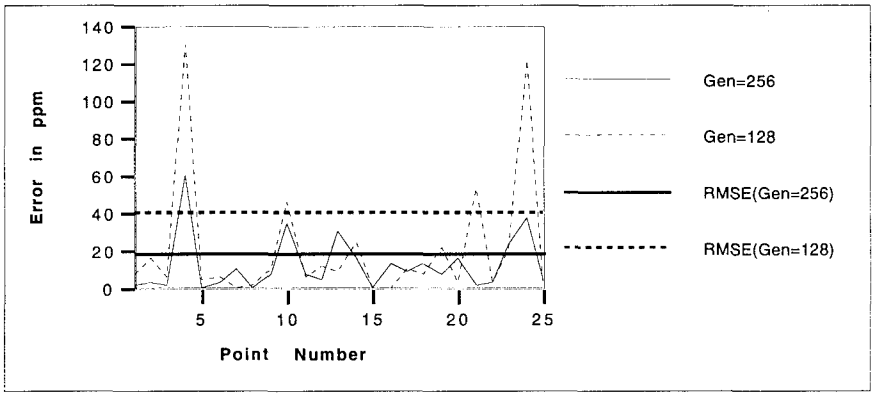


Figure 3: Comparison of error due to different generalization factor

4.4 Shape of Receptive fields:

Four different kinds of arrangements of receptive fields were used [4]. The arrangement denoted ALBUS is the arrangement used in the original CMAC proposed by Albus [1]. The ALBUS option creates a hyperdiagonal arrangement of on-off receptive fields. The RECTANGULAR option creates on-off receptive fields in a more uniform arrangement. The LINEAR option creates linearly-tapered receptive field functions in a uniform arrangement. SPLINE creates cubic-spline approximation tapered receptive field sensitivity functions in a uniform arrangement. LINEAR and SPLINE arrangements outperform RECTANGULAR and ALBUS consistently. On the other hand, the computational time required for RECTANGULAR and ALBUS arrangements was less than for LINEAR and SPLINE. The generalization of 256, for the testing set, gave the lowest RMSE. The effect of receptive field shape results are summarized in Table 3.

Generalization	Input Scaling factor	Output Scaling factor	RMS error in ppm			Beta	Shape of Receptive field
			Toluene	*TCE*	*Acetone*		
256	1	1	**18.96**	**16.99**	**14.51**	1	linear
256	1	1	19.32	17.90	15.13	1	spline
256	1	1	21.70	23.52	18.61	1	rectangular
256	1	1	27.80	35.20	20.10	1	albus

Table 3: The effect of shape of receptive field with other parameters held constant.

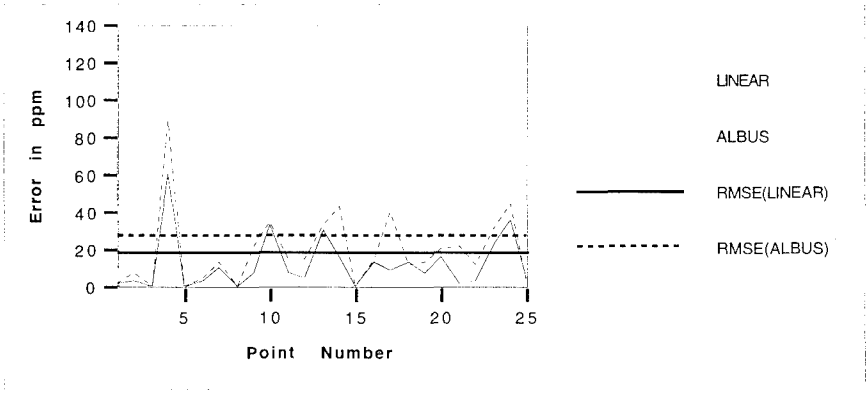


Figure 4: Comparison of error due to different arrangement of receptive fields

4.5 Learning Rate:

The weight adjustment equation for ALBUS CMAC is

$$\Delta W = (\text{desired output} - \text{network output}) \times 2^{-\text{Beta}}$$

Thus, the higher the value of beta the lower is the weight change. Convergence is faster with lower beta, but oscillations may be observed in network outputs because of larger weight adjustments. For the given data set a beta of 1 gave the lowest RMSE. The results of various learning rates are summarized in Table 4.

Generalization	Input Scaling factor	Output Scaling factor	RMS error in ppm			Beta	Shape of Receptive field
			Toluene	*TCE*	*Acetone*		
256	1	1	**18.96**	**16.99**	**14.51**	1	linear
256	1	1	20.25	18.12	15.17	2	linear
256	1	1	22.17	20.14	17.13	4	linear

Table 4: The effect of learning rate with other parameters held constant.

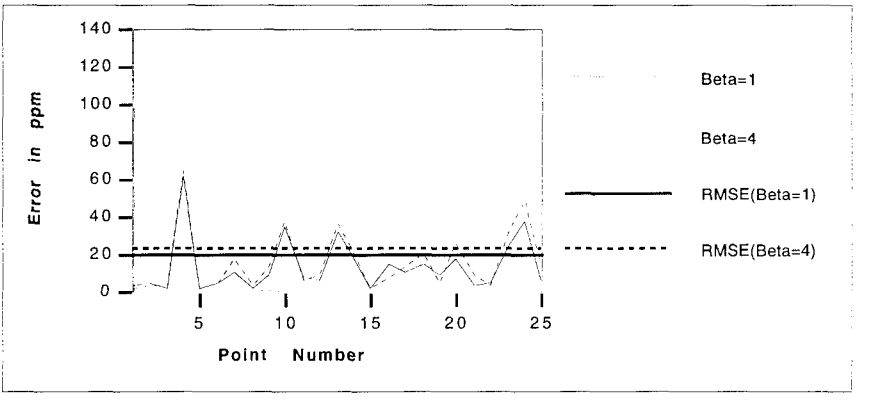


Figure 5: Comparison of error due to different learning rates

Figures 6 to 8 show the plots of desired the output and network output for Toluene, Trichloroethlene and Acetone for each of the 25 testing points for the best network obtained.

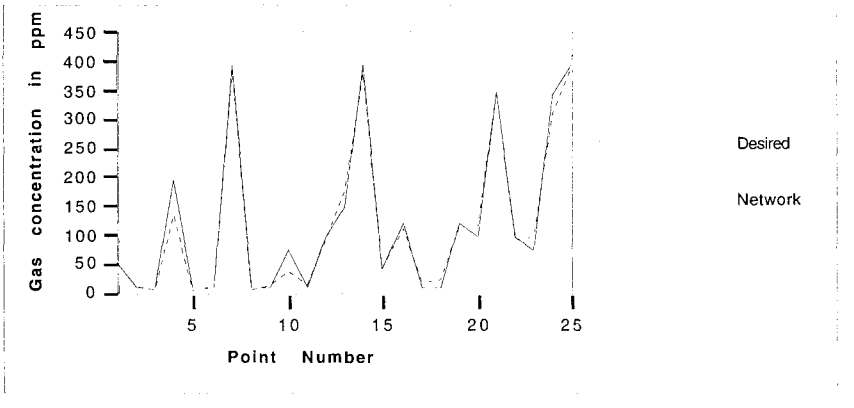


Figure 6: Plot of desired and network output for Toluene

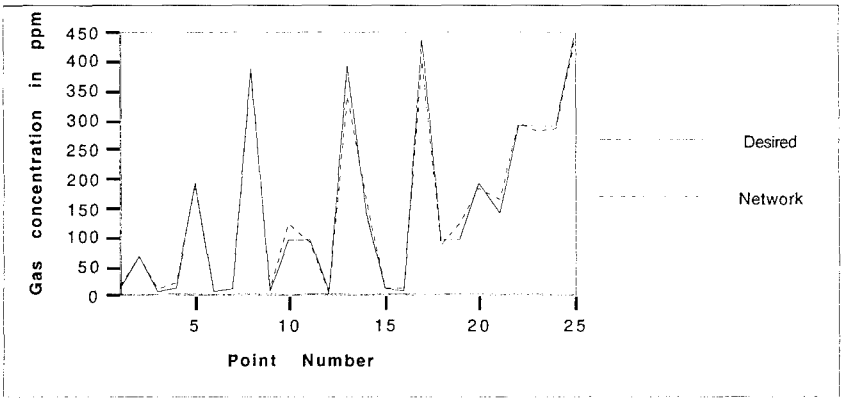


Figure 7: Plot of desired and network output for Trichloroethlene

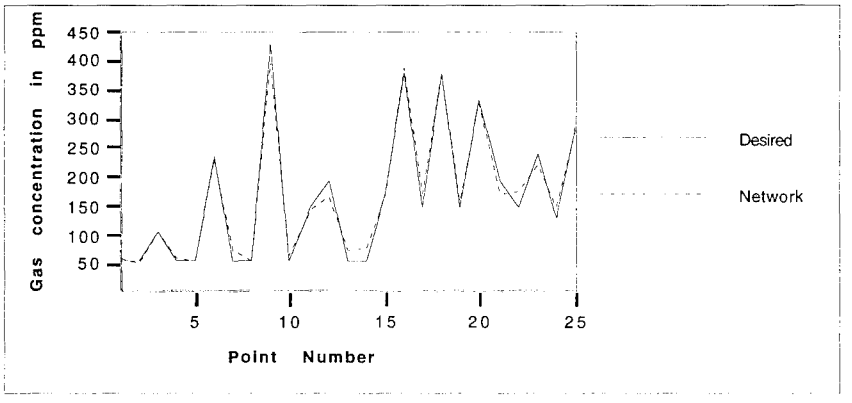


Figure 8: Plot of desired and network output for Acetone

5.0 Conclusions:

The low RMSE errors in predictions of gas concentration proves the suitability of the use of neural networks in gas sensor technology. The above discussion of the network parameters of CMAC leads us to following conclusions.

1) CMAC is very sensitive to network parameters.

2) There is no clear method for choosing network parameters.

3) The shape of CMAC receptive field do have noticeable impact on network performance. Computationally expensive receptive fields such as SPLINE and LINEAR unfortunately improve performance of the network.

4) Due to a low RMSE errors of 18.96 ppm on toluene, 16.99 ppm on trichloroethylene and 14.51 ppm on acetone, as well as the ability to do on-line and incremental learning, we feel that CMAC is a very promising artificial neural network for calibration of chemical gas sensors.

5) These results can be applied to a wide range of multidimensional function approximation problems involving noisy data and/or partially redundant sensors.

6.0 References:

[1] J.S. Albus, "Mechanisms of planning and problem solving in the brain," Mathematical Biosciences, Vol. 45, pp. pp. 247-293, 1979

[2] W.T. Miller, F.H. Glanz, and L.G. Kraft, "CMAC: an associative neural network alternative to backpropogation," IEEE Proceedings, Vol. 78, pp. 140-145, 1990

[3] Xiaodong Wang, Jing Fang, Patrick Carey and Sinclair Yee, "Mixture analysis of organic solvents using nonselective and nonlinear Taguchi gas sensors with artificial neural networks," Sensors and Actuators B, 13-14 (1993) 455-457.

[4] W. Thomas Miller and Filson H. Glanz, "The University of New Hampshire Implementation of the Cerebellar Model Arithmetic Computer -CMAC,".

[5] Srinivasan T, A study of Multilayer Perceptron Neural Network for Gas Sensor Calibration, M.S. Thesis, University of Maine, August, 1994, Orono, ME

CLASSIFICATION OF GAMMA RAY SIGNALS USING NEURAL NETWORKS

N.G.Bourbakis, A.Tascillo and M.Tascillo
Binghamton University, Dept. EE, AAAI Lab, Binghamton, NY 13902

ABSTRACT

One of the difficult problems in the gamma ray signals (GRS or GRB) research area is the extraction, recognition, and classification of the information contained in the signal data. In this paper, a methodology using fuzzy logic and neural networks is presented for the recognition and classification of gamma ray signals. The proposed recognition-classification approach of GRS is divided into two basic steps. In the first step, a global classification scheme is used. The global scheme is based on the processing of the local minima and maxima of GRS, then a global recognition is obtained by using features from the shape of the GRS global envelope. In the second step, a local classification scheme is used. The local scheme is the extraction of the local features of GRS, and on the correlation of those features to achieve a more specific GRS classification. Real GRS signals are used to illustrate the results obtained by this method.

1. INTRODUCTION

Signal recognition is always an important topic for research. Several methodologies (syntactical,stochastic) have been developed for this purpose[1,2]. Gamma ray bursts are signals which may include important information about the scientific history of the universe. The processing of Gamma Rays is an interesting research area with the possibility for many significant contributions. There are, however, some important open problems in the study of Gamma Ray Signals (GRSs). These problems are generic recognition, classification of the GRSs, and creation/extraction of information and rules regarding their structure.

Significant contributions have been made by using classical signal processing methodologies, such Fourier transforms[3,4], for the classification of and information extraction from a GRS. These efforts, however, do not provide a generic solution to the problems mentioned above. These and other information sources can be included in the proposed methodology.

Under these conditions, the use of neural networks as a new approach to GRS problems can provide a significant contribution and a generic approach to the problem[5]. Although classical approaches such as statistical clustering (which measures how well members of one set can be differentiated from all others) and correlation functions (which measures how relatively close one candidate resembles a representative of a given set) can aid in the sorting of

0-7803-2739-X/95 $4.00 © 1995 IEEE

features into categories, a self-organizing neural network can form its own categories based on all features given in a vector representing each GRS. A strictness criterion is adjusted to vary the number of categories created, and as new examples are added, a new candidate can be sorted using the existing network weights, or the network can be allowed to reorganize itself to account for the additional data. The extent of recategorizing can be employed as a relative measure of the network's category robustness. A time relationship of features may also be learned during this categorization.

When the sampling rate is relatively slow, some feature information that is lost may have to be inferred from what remains. Neural networks, via a distributed internal representation of the nonlinearities in a given relationship, can assume missing information and also ignore noise, aliasing effects, and other information that does not lead to a significant relationship. Outputs of a standard or modified feedforward neural network, with a weight training method geared to the nature of the relationship to be learned, can be trained by example to act as diagnostic classifiers[8] and generate the feature vector for the self-organizing network directly. Further insights can be implemented to modify this feature vector with an interpretive fuzzy logic routine[11], supervisory neural network or expert system.

In particular, the methodology presented here includes three main steps. The first step is the global recognition-classification of a GRS, using filtering, detection of minima-maxima, and feature extraction/recognition. The second step is the local recognition and classification of the GRS features. Based on these two steps, it is possible to provide a generic hierarchical classification scheme for the GRS. The third step is the training of neural network models in order to learn the important features and rules that interrelate the GRSs.

2. THE RECOGNITION AND CLASSIFICATION METHODOLOGY (RCM)

The RCM methodology attempts to classify automatically GR Signals by combining neural networks and syntactic pattern recognition approaches. The RCM methodology is divided into three main parts:

1) The Global Recognition-Classification (GRC) of a GRS,
2) The Local Recognition-Classification (LRC) of a GRS, and
3) The Categorization of using fuzzy logic and neural nets

2.1. The GRC Scheme

The GRC scheme consists of four main processing parts:

- The filtering process of the GRS values [v(t)],
- The detection process of the local minima-maxima of the GRS,
- The features extraction process, and
- The recognition and classification of the GRS features.

The Filtering Process

The filtering process is based on a Kalaman filter [7] to remove the noise

from the GRS signal, under the consideration that the noise follows a Gaussian distribution.

The Detection of the Local Minima-Maxima (Envelops)

The detection of the local minima and maxima of the GRS's continuous form is based on the determination of the straight line segments (SLS) of the GRS form, and the selection of those values v(t) that belong to the starting points (SP) and ending points (EP) of each straight line segment. A local minimum value (lmin[v(t)]) is defined as a value v(t) with $v(t)<v(t-1)$ and $v(t)<v(t+1)$, where v(t-1) and v(t+1) represent the previous and the next values of a GRS in time. A local maxima value (lmax[v(t)]) is defined as a value v(t) with $v(t-1)<v(t)$ and $v(t)>v(t+1)$.

Using the definitions above, the local minima and maxima of the GRS signal can be determined.

Feature Extraction Process

A curve fitting process is applied to the local minima and maxima to produce the curves that comprise the global shape of the GRS. In particular, the curve fitting process is applied to all the local minima, and separately to all the local maxima.

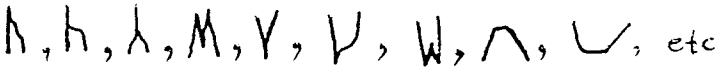
The lines produced by curve fitting will be processed in order to extract global features of the GRS. These global features are curved and straight line segments as well as their specific characteristics (starting point, orientation, curvature, size, etc.).

Recognition and Classification of the GRS global Features

At this stage, the extracted GRS global features are compared and interrelated for the better understanding of the main characteristics of the GRS and its global classification.

2.2. The LRC Scheme

The LRC scheme is based on the extraction of the local features of the GRS, especially useful when conventional function approximators cannot properly capture the sharp transitions of complex time profiles. These local features are several primitive shapes (PSs), such as:



which exist in the original GRS shape.

The extraction of the primitive shapes is based on a matching process applied to the original GRS shape and the existing set of primitive shapes. Since the PSs will be detected and extracted, an interrelation process starts which compares and synthesizes these PSs to form either larger recognizable shapes, or to provide additional information about the specific characteristics of the GRS, to assist a more specific classification of the GRS. Note that each primitive shape extracted from the GRS shape includes its own properties, such as size,

orientation, location, period, and so forth.

During the training of the neural network model with the existing GRS database, the global and local features of each GRS will be interrelated with all the set of GRSs to attempt to generalize "rules", perhaps related to the process that generates GRSs. The neural network model will also learn the set of the PSs existing in the GRS database (unsupervised learning). Later on these PS features will be used for the classification and processing of new GRSs.

3. SIMULATED RESULTS

The results of this simulation show that classification proceeds as expected; in this case the network forms a decision surface that correctly classifies all 500 examples, but also classifies new examples not necessarily in the training set with an error rate less than 1%. More complex problems may require more complex neural network models that exhibit more complex behavior, but they still use the same principles of operation. In particular, multi-layer networks with backpropagation training rules are a solution to arbitrarily difficult classification problems, i.e. more difficult than the above linear classification problem.

An additional simulation, employing the self-organizing network described earlier for some representative GRSs, three categories emerged. Examples of each GRS category are shown in Figures 1-3, with local and global curve fitting shown as dotted lines. Extracted features included gains and constants of exponentials, sinusoids, and parabolae of the associated shape primitives. Table 1 shows the hierarchical classification for the previous three GRSs.

3.1.Categorization via Fuzzy Logic and Neural Networks

A fuzzy spectral filter approach is employed to classify a candidate signal given a representative reference signal for each category. A radial basis function neural network can then be used to interpret the category memberships suggested for each envelope. If no one category stands out, a fuzzy supervisor looking at the network's outputs will identify it as a possible
new category and the signal can be established as its own reference.

For demonstration purposes two unclassified signals were compared to three reference signals (Figures 4-6) via the memberships for the upper outer envelope. The task at hand is to recognize
slight permutations of known signals which are contaminated or altered by ambient noise. Unclassified Signals 1 and 2 represent two such possibilities. Unclassified Signal 1, (Figure 7) is a slight permutation of Signal 2 and Unclassified Signal 2 (Figure 8) is a doubled version of Signal 1. All signals are presented with both inner and outer, upper and lower envelopes already computed.

The following procedure was employed:

1. *Select the signal waveform to be classified.*
2. *Compute inner and outer envelope waveforms.*
3. *Compute power spectral density of envelopes.*
4. *Define relevant spectra (those above background) for each reference*

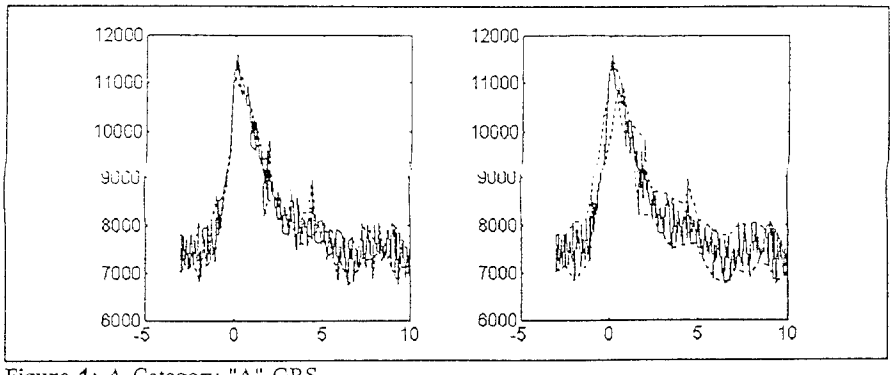


Figure 1: A Category "A" GRS.

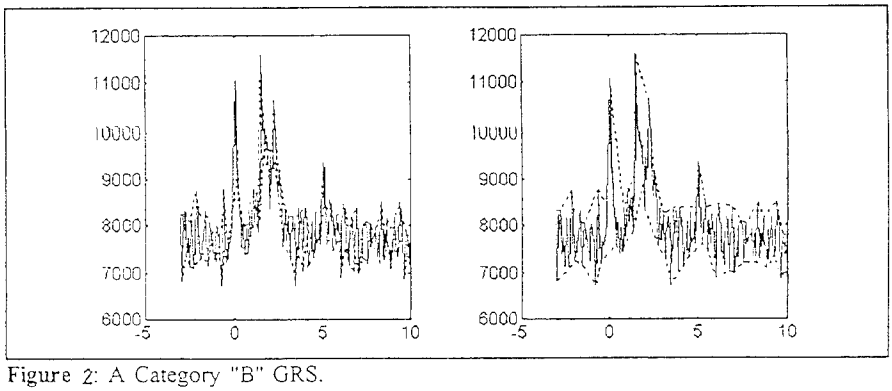


Figure 2: A Category "B" GRS.

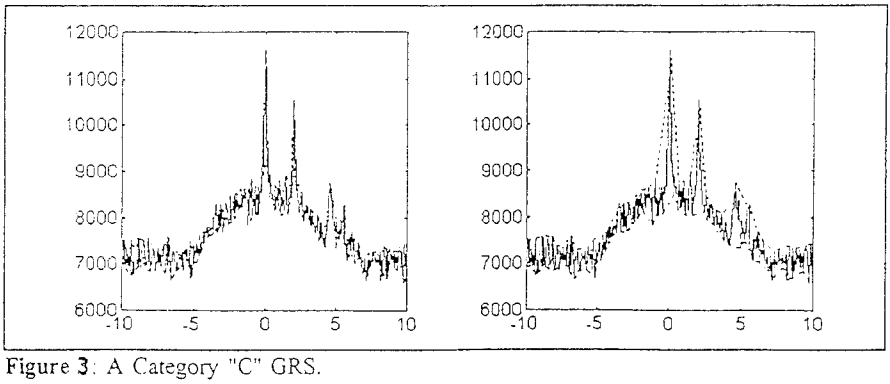


Figure 3: A Category "C" GRS.

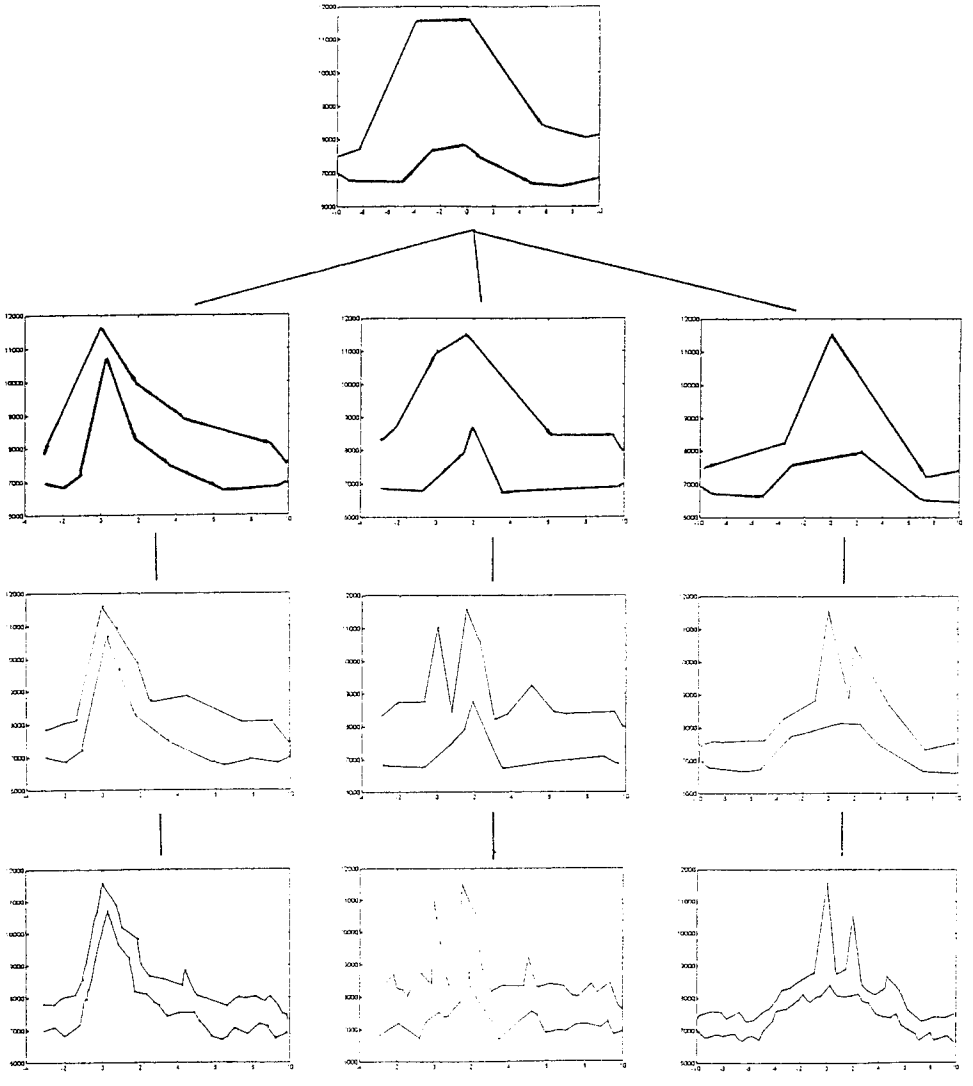


Table-1: A hierarchical classification scheme for three GRBs. The top level of this scheme is a global envelope which covers all these three GRBs. The intermediate levels present envelopes that maintain only the most visible features of each GRB.

signal.

5. *Define Gaussians centered about selected spectra for each reference signal.*
6. *Define relevant spectra for each candidate signal.*
7. *For the location of each candidate spectra on frequency axis, note the total value of reference spectra.*
8. *Find the total value of candidate spectra for each reference and normalize by number of reference spectra.*
9. *Result in values such as those of Table 2.*

Table 2

	Reference 1	Reference 2	Reference 3
Candidate 1	1.39	*1.7*	1.37
Candidate 2	*1.05*	0.04	0.7

The fuzzy spectral filter correctly classified the two signals as subsets of one of the established references. Each of the candidate signals is correctly identified by the high normalized value as belonging to the correct family. For practical application a tolerance can be applied for the definition of a virtual tie between two family comparisons for ambiguous signals. Such signals could be singled out for special treatment and further analysis.

Step 3 of the analysis implies that one can separate relevant spectra from background noise. This is done here by finding the mean of the power spectral density, and adjusting it by a factor of 1.5 to lift it just above the baseline noise (Figure 9). This can be used as a general tuning factor so long as it is applied in a consistent fashion to any one group or family of signals to be compared. A peak finder algorithm is then applied to locate the peaks of the power spectrum. The location of each peak on the frequency axis is cataloged.

The fuzzy logic analysis applies a Gaussian centered about each of the selected peaks on the frequency axis, each by definition having a maximum magnitude of 1.0. The fuzzy logic approach alleviates both spectral location and magnitude uncertainties for small data sets often encountered. It assumes that the relative peak magnitude is of no concern so long as it is above the predefined background noise level. Furthermore, small errors in peak location on the frequency axis are taken into account by the gaussian mapping function.

As seen in Figure 10, the Gaussians for the reference signal (in this case, 2) are situated on the frequency axis with the candidate frequency peaks superimposed. The candidate frequencies are mapped as point locations with their total weight value for any one point being equal to the Gaussian magnitude at that point. An overall goodness of fit is computed as the sum of the various candidate locations, normalized by the number of potential Gaussians into which they could fall. The normalization is achieved by dividing each total weight by the number of reference frequency buckets available. This essentially normalizes each reference signal to a total of one effective bucket. The resulting goodness of fit weights are then tabulated as seen above in Table 2.

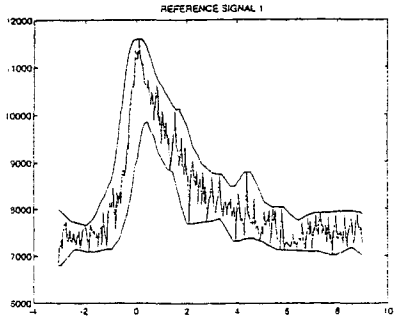


Fig 4

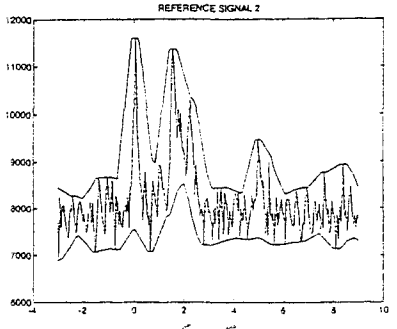


Fig 5

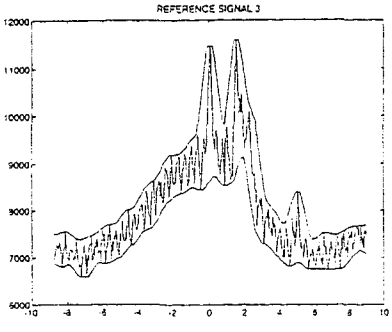


Fig 6

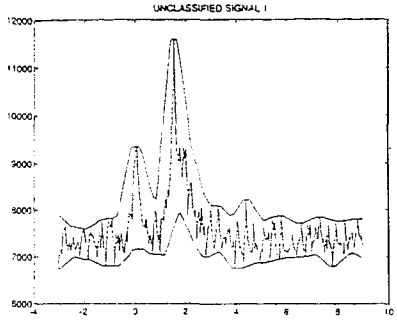


Fig. 7

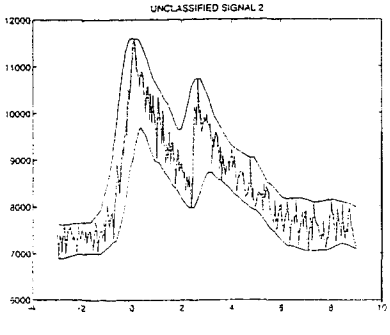


Fig. 8

CONCLUSIONS

This paper a methodology for the automatic classification of GRS using neural networks presented. The method was based on the study of the global and local characteristics of GRS and attempts a hierarchical classification of the GRS signals by interrelating these features. It also offers a frame by using the neural networks, to extract additional knowledge and interrelation rules from existing GRS databases.

References

1. NATO Advanced Study Institute on Pattern Recognition and Signal Processing, Paris 1987.
2. Proc. IEEE Workshop on Signal Processing and Neural Nets, Sept. 1993, Baltimore MD
3. C.Kouveliotou et.al.,"Fast Fourier Transformation results from Gamma Ray Burst profiles", NASA, 1992.
4. C.Kouveliotou et.al.,"Description of a subset of single events from the BATSE Gamma Ray Burst data", NASA, 1992.
5. N.Bourbakis,"A Language Methodology for Speech and Signals Understanding IJEAAI,1996.
6. D.A.Pomerleau, "Input Reconstruction Reliability Estimation," Advances in Neural Information Processing Systems eds C.Giles, S.Hanson & J.Cowan, CA, M.Kaufmann,1993.
7. A.Sage and J.Melsa,"Estimation theory with applications to communication, McGraw Hill 1971
8. A.Tascillo, Diagnostic, adaptive and redundant planning and control of a robotic hand, PhD 1995,SUNY-B.

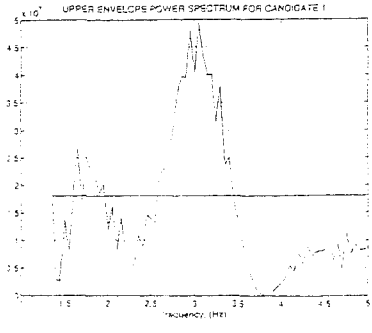


Fig 9

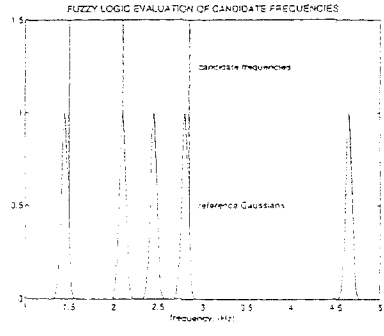


Fig 10

Adaptive Preprocessing for On-Line Learning with Adaptive Resonance Theory (ART) Networks

Harald Ruda & Magnús Snorrason
Charles River Analytics
55 Wheeler St., Cambridge, MA 02138
hxr@cra.com, mss@cra.com

Abstract—Neural networks based on Adaptive Resonance Theory (ART) are capable of on-line learning. However, a limiting factor in on-line processing has been the need to preprocess input patterns so that features fall in the range [0.0, 1.0][1], typically done with scale-factors that depend on the input range of each feature. This paper demonstrates a method by which the scaling of features becomes adaptive, eliminating the need to batch-process patterns before presenting them to the ART network. The resulting network implementation for on-line learning does not call for any knowledge of the feature signals, ranges or otherwise. A variety of implications of this scheme are analyzed.

INTRODUCTION

A classifier is capable of on-line learning if it can learn to classify patterns as they are presented without storing the patterns for reference. It has been suggested that ART networks are capable of such on-line learning [1,2]. Even though this has not been proven theoretically, one finds that learning of the whole set is usually accomplished in only one cycle (especially with the learning rate parameter beta = 1.0). Therefore it is practical to use ART networks for on-line learning tasks.

Traditionally, classification algorithms have been restricted to batch mode learning. In other words, the whole training data set must be stored and available for use by the algorithm. In such situations the scaling of input features is trivial: the training set can be searched for the extreme values of each feature and those values (possibly with added safety margins) can be used as estimates of the true ranges of each feature.

[1]This is strictly true for Fuzzy-ART and Fuzzy-ARTMAP, but ART2 and ART2-A does not technically require preprocessing of features other than ensuring that all features are non-negative. In practice however, scaling of features is often necessary.

0-7803-2739-X/95 $4.00 © 1995 IEEE

There are applications where batch operation is not desirable. For example, if there are real-time constraints acting on the system, one does not want to retrain the classifier from scratch each time a new pattern class is identified. Rather, one wants an on-line classifier that can add knowledge about the new pattern to its previous knowledge. As another example, the classifier may be required to train on such a large number of patterns that presenting the patterns more than once is too time-consuming or otherwise resource intensive.

These are some situations in which on-line learning is *required.* Unless the true range for each feature is already known, these situations present a real problem for feature preprocessing. It is no longer possible to *a priori* determine the extreme values of features in the training set because the whole training set is never available at once. This is the problem that our adaptive preprocessing solves by linking the scaling of features with the scaling of weights. Adaptive preprocessing is also very convenient, even in batch-processing situations, because it eliminates the need for analysis of input variable ranges.

ALGORITHM

There are two key parts to the algorithm needed for implementing adaptive preprocessing.

- The observed range for each feature must be tracked. The maximum ($fmax_i$) and minimum ($fmin_i$) values of each feature i have to be stored and updated every time they change.
- When input pattern p contains a value for feature i (f_i^p) which falls outside of the previously observed range, the range must be updated and the weights relating each cluster (w_{ji}, $\forall j$) must be adjusted such that previously seen patterns would be coded the same way if they were presented again.

The adjustment is performed according to the equations below, which apply specifically to Fuzzy-ARTMAP [3] and Fuzzy-ART [4]. First the new range has to be determined according to:

$$fmin'_i = \min(fmin_i, f_i^p) \text{ and } fmax'_i = \max(fmax_i, f_i^p) \qquad (1)$$

where (') indicates the updated value, such as in $fmin_i$. If this new range is different from the old, then the cluster weights are scaled:

$$w'_{ji} = w_{ji}\left(\frac{fmax_i - fmin_i}{fmax'_i - fmin'_i}\right) + \left(\frac{fmin_i - fmin'_i}{fmax'_i - fmin'_i}\right) \qquad (2)$$

and the complement weights:

$$\bar{w}'_{ji} = \bar{w}_{ji}\left(\frac{fmax_i - fmin_i}{fmax'_i - fmin'_i}\right) + \left(\frac{fmax'_i - fmax_i}{fmax'_i - fmin'_i}\right) \tag{3}$$

where for each feature i, the weights to every cluster j are adjusted.

Whether the range changed or not in (1), each feature in every pattern must be normalized by mapping the feature range to [0.0, 1.0]:

$$fnormalized_i^p = \frac{f_i^p - fmin'_i}{fmax'_i - fmin'_i} \tag{4}$$

Equations (2 – 4) provide a linear mapping of feature and weight values, but the general approach will also work with nonlinear mappings. As shown in Figure 1, clusters that were not selected by the current input pattern merely shrink through (2) and (3) as the observed range is expanded. Patterns which previously selected those clusters will still do so, because the expanded range also shrinks features by (4). Cluster growth occurs in the normal Fuzzy-ART manner of weight update in response to training patterns. If a given input feature value is outside the previously observed range, the cluster weight representation for that feature will expand if it is selected by the current input pattern. In summary, for the selected cluster, the weight representation can shrink for some features and simultaneously expand for others.

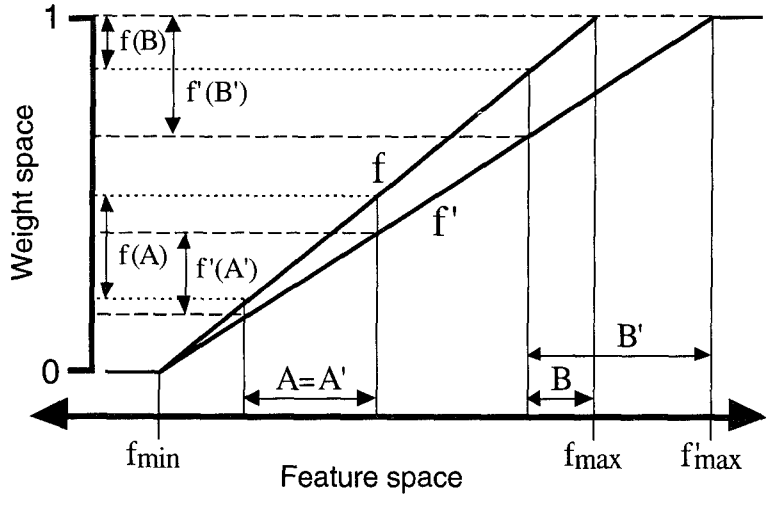


Figure 1. A graphical representation of one feature (f) and the mapping of values from the original feature space to the [0.0, 1.0] weight space. At a given point in the training cycle, values of f from fmin to fmax have been observed and two clusters have been learned, A and B. Later in training, f'max is observed, which is larger than fmax. According to (2), the weight values representing A and B change. f'(A') shrinks because the mapping function now has a shallower slope. This is true for any cluster, other than the one which contains f'max, i.e. f'(B').

RESULTS

Figures 2 and 3 show results of tests to verify that adaptive preprocessing does not interfere with the proper functioning of the network. We used benchmark machine learning databases from the University of California, Irvine repository at ftp://ftp.ics.uci.edu/pub/machine-learning-databases.

The first test is a two-class problem based on categorical data. The *Agaricus-Lepiota mushroom* database contains over 8000 patterns of features, observed from 23 species of mushrooms that belong to the genera Agaricus and Lepiota. A pattern consists of 22 features (such as "stalk-shape" or "odor"), each with 2 - 12 possible categorical values (such as "tapered" or "pungent"). The classification task is to distinguish between edible or poisonous mushrooms, which often look very similar.

The second test is a difficult six-class problem based on quantitative data. The *glass* database contains 214 patterns of 9 different measurements of chemical properties of glass. The classification task is to separate the data into 6 classes based on these 9-dimensional patterns.

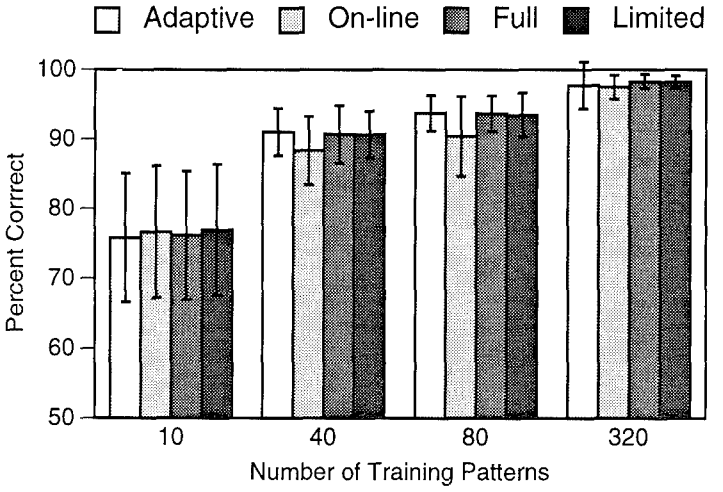


Figure 2. A test to demonstrate that adaptive preprocessing does not cause degradation of classification performance. Fuzzy ARTMAP was used to classify the Agaricus-Lepiota mushroom database. Four processing conditions are shown: (Adaptive) uses the adaptive preprocessing. (On-line) uses the adaptive preprocessing with the on-line constraint of allowing only one cycle of learning for each set of training patterns. (Full) uses batch-scaling of features based on $fmin_i$ and $fmax_i$ over the whole data set. (Limited) uses batch-scaling of features based on $fmin_i$ and $fmax_i$ over the training set only. The mean and standard deviation of each bar were estimated from 50 different random selections of training sets. "Percent Correct" is the average of the evaluation set, which contains 8124 patterns minus the number of training patterns.

Notice the minimal effect of on-line learning on the results in Figure 2, i.e. presenting each pattern only once. The penalty for on-line learning using adaptive preprocessing is never more than 3.5% and the best average performance (for 320 training patterns), it is almost identical.

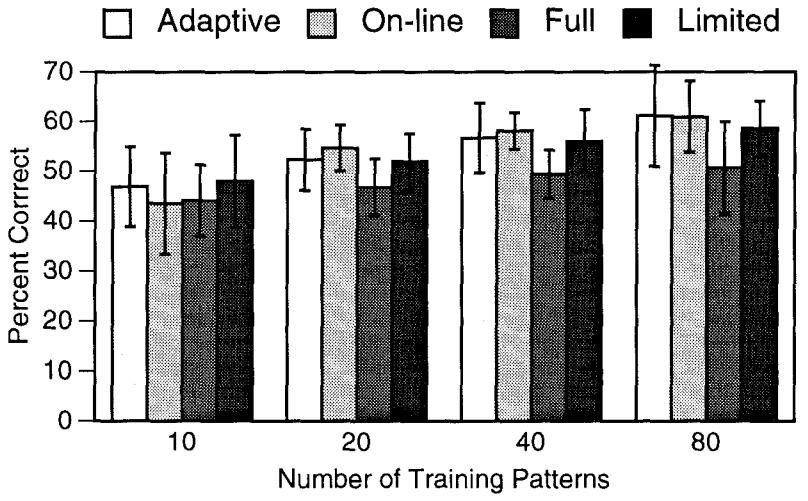


Figure 3. Same information and methods as in Figure 2, but for the glass database. The evaluation set contains 214 patterns minus the number of training patterns.

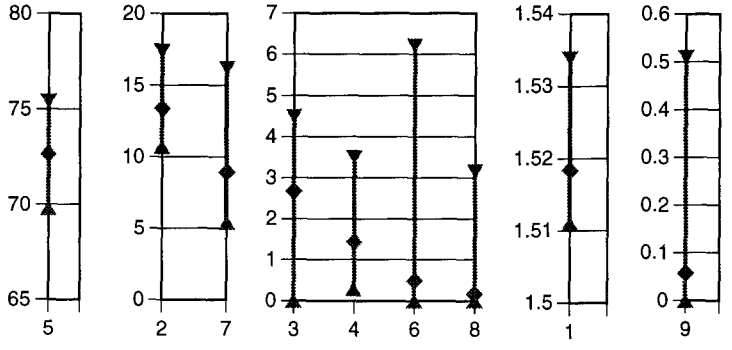


Figure 4. The ranges and means of each feature from the glass database. Feature 1 represents the refractive index of sample; the other features represent different concentrations of oxides.

It is evident from the lower overall scores in Figure 3 that discriminating between the 6 glass categories is much harder than classifying the mushrooms. Still, adaptive preprocessing works just as well as fixed preprocessing based on the training part of the set, and consistently better than fixed preprocessing based on the whole set. Figure 4 shows that the

natural range of the features is large for some features (such as 2, 5, and 7) and very small for others (1 in particular), and the distribution within that range is very skewed for some (such as features 6, 8, and 9).

TYPES OF LEARNING IN FUZZY-ARTMAP

When adding the adaptive preprocessing to Fuzzy-ARTMAP, the result is a network with four distinct types of learning which can be individually controlled (in high to low level order):

- Supervised learning, or learning in the map-field. This type of learning determines the connections between clusters and output classes. It is performed in training mode, i.e. when the current input pattern has an associated output class. Several clusters can map to the same output class, implementing decision surfaces of arbitrary complexity.
- Creation of new cluster nodes. When no previously established clusters meet the "vigilance" criterion new clusters are formed. This can be disallowed, forcing the choice of a previously created cluster. Conversely, cluster creation can be allowed during testing, although the mapping to an output class has to be deferred until relevant training data is presented.
- Modification of cluster weights. This is the type of learning Fuzzy-ARTMAP has in common with most other neural networks. Learning of this type is usually allowed when in training mode. In certain situations, the network can also be allowed to update weights in testing mode.
- Allow updating of input feature ranges and the associated adjustment of cluster weights. Ordinarily, this type of learning applies both in training and testing modes (but see section below on dealing with noisy data).

The choice of learning types allowed during training and testing modes is application dependent and the number of options makes Fuzzy-ARTMAP with adaptive preprocessing applicable in a variety of situations.

ADAPTIVE PREPROCESSING

Nonlinear Mappings

Mapping the range of the input feature to the usable [0, 1] range in order to maximize the effectiveness of the network also includes the possibility of using a nonlinear mapping rather than linear transformations. Indeed, any monotonic (and therefore one-to-one) function can be used for this purpose. Some of the functions which might be used instead of lines are sigmoids (logistic or tanh), logarithmic, exponential, and power functions. At the moment the choice of scaling function has to be determined manually, with some knowledge of the features used. A different function may be used for

each feature.

An intriguing (but computationally expensive) option is to track the mean and variance of each input feature instead of the min and max. Then a sigmoid is the natural choice for a mapping function. This would seem an ideal mapping for features that are normally distributed.

Applicable Network Types

This adaptive preprocessing scheme is designed for analog ART networks that use complement coding, but it will work for any analog ART network, such as ART2, ART2-A, and Fuzzy-ART. The procedure should work equally well for any network where the weights can be considered *templates* of input features. Another consideration is whether a situation calls for feature normalization, which is often used in practice even if not theoretically required for the given paradigm.

A mean-and-variance tracking version of the procedure could be applied to multi-layer perceptrons. The input features would be normalized to zero mean and unit variance. Changes in variance of a given feature affect the importance of weights propagating from that feature. These weights can be adjusted according to the reciprocal of the variance change. Changes in the mean have higher order effects on the required weight adjustments. These effects could possibly be ignored, or accounted for with bias adjustments.

A deeper issue is the amount of training required for multilayer perceptrons. As mentioned in the introduction, on-line learning is a major impetus for the described procedure. It would seem that multilayer perceptrons are particularly ill suited for on-line learning, due to their extensive training requirements. Until on-line learning abilities are developed for multilayer perceptrons, we do not see a need to adapt our procedure to such networks.

Noise Tolerance

The adaptive preprocessing method introduced here can be sensitive to outliers, just as traditional fixed scaling methods. One solution is to detect and discard extreme outliers that would otherwise adversely compress the useful range in weight space. If outliers cannot be detected, there are still several methods available to avoid problems.

The simplest possibility is to predetermine absolute bounds for each feature and clip any features that fall outside these bounds. Another possibility is to keep a running mean and variance, as discussed above, and to clip any values that fall outside a predetermined number of standard deviations.

A more elegant method is the use of a compressive non-linearity, such as a sigmoid function. In that case, no feature value is large enough to adversely compress the useful range. It is also possible to use a learning rate parameter for the feature and weight adjustments, such that the range will not immediately expand to encompass the noisy data. Several such data points would be necessary to establish a new bound. In a real application, the simultaneous use of some of these methods would be prudent.

Note that by (1), the observed range for each feature can only grow. A continuous baseline shift in a given feature will cause the range to expand continuously, mapping the useful range of that feature into an ever decreasing portion of the [0.0, 1.0] range. This can eventually lead to resolution problems, but with the floating precision representation used in most computers, an extensive baseline shift is required. This method has the distinct advantage of breaking down gracefully as the resolution decreases, unlike methods based on clipping.

CONCLUSION

The use of adaptive preprocessing is necessary for true on-line learning with Fuzzy-ARTMAP and it can greatly simplify the network's use in a variety of other applications. The procedure is a computationally simple addition that does not sacrifice performance and yet increases system robustness tremendously. Additionally, if classification rules are to be extracted from the network, then the cluster weights can be given direct meaning in terms of non-normalized feature values. Finally, there is potential for adapting the method to other on-line learning paradigms.

REFERENCES

[1] Carpenter, G. & Grossberg, S. (1987a). A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. *Computer Vision, Graphics, and Image Processing*, 37, 54-115.

[2] Carpenter, G. & Grossberg, S. (1987b). ART2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns. *Applied Optics*, 26, 4919-4930.

[3] Carpenter, G.,Grossberg, S.,Markuzon, N.,Reynolds, J.H., & Rosen, D.B. (1992). Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps. *IEEE Transactions on Neural Networks*, 3(5), 698-713.

[4] Carpenter, G.,Grossberg, S., & Rosen, D.B. (1991). Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System. *Neural Networks*, 4, 759-771.

Intelligent Network Monitoring

Cynthia S. Hood and Chuanyi Ji

Electrical, Computer and Systems Engineering Department
Rensselaer Polytechnic Institute
Troy, NY 12180
hoodci@ecse.rpi.edu, chuanyi@ecse.rpi.edu

Abstract

To improve network management in today's increasingly complex communication networks, we propose an intelligent monitoring hierarchy. The hierarchy is comprised of Hidden Markov Models (HMMs) and neural networks. As demonstrated on real network data, this hierarchy can detect abnormal behavior at high levels using only readily available low-level fault models. This allows the node to provide the network manager a complete picture of the nodes health.

1 Introduction

As communication networks continue to increase in size and complexity to support new applications and the large number of new users, understanding and managing network behavior becomes increasingly difficult. Many of the new applications (such as video) will require communication networks to maintain a higher standard of network availability and reliability, thereby making effective network management crucial. Thus, at a time when network management is critical, the complexities of the network make it more difficult than ever.

Network management is a broad term that has been defined according to the Open Systems Interconnection (OSI) systems management specification developed by the International Organization for Standardization (ISO). It

0-7803-2739-X/95 $4.00 © 1995 IEEE

includes the following five functional areas: fault management, performance management, accounting management, configuration and name management, and security management. These functional areas are covered by a Network Manager (NM) which may be an automated system, a human, or a combination of both. An NM may be responsible for managing an entire network or just part of the network, depending on the size of the network. The NM does it's job by processing information received from the network nodes (servers, routers, bridges, terminals, etc.). How well the NM performs it's job is highly dependent on the quality and completeness of the information received from the nodes.

The collection and processing of data at each node is referred to as network monitoring. There is a tremendous amount of data collected at each node that must be processed to extract pertinent information to be sent to the NM. The definition of pertinent varies, but in general the NM expects each node to notify it of any abnormal or anomalous behavior. To accomplish this, the monitoring function at the node must be able to process the data and detect the anomalous behavior. In this paper, we focus on anomaly detection for fault and performance management although the methods developed can be generalized to the other functional areas as well.

The task of anomaly detection is extremely difficult due to the fact that network behavior is dynamic and it depends on many factors such as network load, traffic characteristics, and network configuration. In addition, even if the network was static, since there are no large-scale system models to model the complex behavior at a node[1], anomaly detection must be based on anomaly or fault models.

The types of existing fault models used to detect anomalies can be categorized as either low-level or high-level models. Low-level models are simplistic one-dimensional data thresholds. These models are complete (i.e. a data point is either above or below a particular threshold). When the threshold is exceeded, the NM is notified with a message from the node called an alarm. The NM must understand the temporal relationship between alarms so the problem can be identified at the highest level and the appropriate action(s) can be taken. If the symptoms are identified as the problems and the higher-level issue is not addressed, the actions taken may

[1]Note that techniques such as queuing models and finite-state machines have been used to model various pieces of a node, but integrating these pieces or designing a large-scale model to accurately and completely model a node's behavior is an open problem.

actually degrade the network further. Currently, due to the way data is collected and reported [7], temporal information is lost, making correlation extremely difficult.

High-level models attempt to address these issues by correlating the low-level information at the node. Instead of notifying the NM every time a threshold is exceeded, these models correlate the information locally and notify the NM only when a known anomaly is detected. These models are comprised of known fault information such as fault signatures. For example, a simple high-level model may detect an anomaly when both variable 1 has exceeded a threshold for at least 5 minutes and variable 2 exceeds a threshold. These models are incomplete since it is not feasible to identify every abnormal situation. Previous research approaches focusing on high-level models include expert systems [1], Finite State Machines [4], and advanced database techniques [6]. These approaches consider temporal relationships, but require fault specifications to set up the required constructs. They do not address the fact that anomaly information is not available in most cases. In addition, since the definition of anomalous behavior changes as the network evolves, the accuracy of high-level models decreases as the network evolves.

The key open problem for anomaly detection is how to detect unknown anomalies or faults. The major contribution of this work is a structure and method for detecting high-level anomalies using only low-level models which are readily available and complete. High-level fault models are not required. To accomplish this, we propose an intelligent monitoring hierarchy at each node. The hierarchy consists of Hidden Markov Models (HMMs) and neural networks. The neural networks are used to learn the normal behavior of the network, and the HMMs provide temporal context.

Our results demonstrate on real network data that this type of hierarchy can detect high-level anomalous situations using only low-level fault models. The state (normal or abnormal) probabilities are determined at each level in the hierarchy, thereby providing a full picture of a node's health to the network manager. The network manager can then more accurately determine if the alarm is a problem itself (no more general problems exist) or a symptom of a bigger problem.

2 Intelligent Monitoring Hierarchy

We have taken advantage of the breakdown of network architectures into a set of hierarchical layers to construct a network monitoring hierarchy. The three-level hierarchy detects anomalies in a Management Information Base (MIB) variable, within a protocol layer, and spanning both protocol layers (defined as network anomalies in this paper) (Fig. 1(a)). This allows both low- and high-level information to be passed on to the network manager.

The detection is accomplished by defining two states (normal and abnormal), corresponding to the health of the element in the hierarchy (i.e. MIB variable, protocol layer, or network). At the lowest level, the state of each MIB variable is estimated from the data. At the higher levels, the state of each element is estimated using the state probabilities from the previous or lower level as input. Elements identified as being in the abnormal state are considered to be anomalies.

At each level, the state estimation involves an HMM. We have assumed that the underlying process is Markov in nature, but we cannot observe the states (i.e. they are hidden), therefore an HMM is chosen. This model provides the ability to consider temporal context which is very important in the network monitoring application.

2.1 Lowest Level of the Hierarchy

At the lowest level (MIB variable), each MIB variable is the input into a hybrid neural network-HMM as proposed by Smyth[5](Fig. 1(b)). Although the structure of the neural network-HMM is the same, Smyth estimated a high-level fault model with it, whereas we are estimating a low-level fault model. A feedforward network is trained using the standard backpropagation algorithm to provide the a posteriori state probabilities $p(s_i|\theta(t))$ [3], where s_i is the state, $\theta(t)$ is the time-series data of the MIB variables at time t, and m is the number of states. Given these probabilities, the HMM then provides temporal context by estimating the state probabilities as

$$p(s_i|\theta(t)) = \frac{\alpha_i(t)}{\sum_{j=1}^{m} \alpha_j(t)},$$

where

$$\alpha_i(t) = p(\theta(t)|s_i) \sum_{j=1}^{m} a_{ij}\alpha_j(t-1).$$

We assume a_{ij}, the state transition probabilities to be known ahead of time. By Bayes' rule

$$p(\theta(t)|s_i) = \frac{p(s_i|\theta(t))p(\theta(t))}{p(s_i)},$$

where $p(s_i)$ are the assumed prior state probabilities. Since the probability of the measured data $p(\theta(t))$ is independent of the state, this can be omitted from the calculation without loss of generality. The outputs of the HMMs (state probabilities) are used to estimate the state of the MIB variable and also passed up to the second level. The state is estimated by selecting the state with the highest posterior state probability. If the state is determined to be abnormal, notification will be sent to the NM (along with the states of the next two levels).

2.2 Higher Levels of the Hierarchy

At each of the next two levels (Fig. 1(b)), the outputs from the previous level are combined linearly to estimate the higher-level state probabilities $p(S_i|\Phi(t))$, where S_i is defined as a higher-level state probability and $\Phi(t)$ is $\theta(t)$ and the outputs from previous level HMM(s).

$$p(S_1|\Phi(t)) = p(s_{11}, s_{12}|\Phi(t)) + 0.5[p(s_{11}, s_{22}|\Phi(t)) + p(s_{21}, s_{12}|\Phi(t))]$$

$$p(S_2|\Phi(t)) = p(s_{21}, s_{22}|\Phi(t)) + 0.5[p(s_{11}, s_{22}|\Phi(t)) + p(s_{21}, s_{12}|\Phi(t))]$$

where s_{ij} represents the i^{th} state of the j^{th} output from the previous level. In the cases where one of the inputs is normal and the other is abnormal, the probability mass is split evenly. The variables we monitored were assumed to be independent (a reasonable assumption based on the data), so the joint probabilities can be easily calculated. These state probabilities are then used as input into the HMM as described in Section 2.1. Once again the state is estimated and the state estimate is included in the information sent to the network manager if notification is warranted (i.e. an abnormal state has been detected within the hierarchy).

3 TCP/IP Network Monitoring

The proposed hierarchy is applied to the TCP and IP layers of a single node on the Internet. Time-series data is collected by polling the MIB of

the node. The MIB was chosen as a data source due to the clear separation of data by protocol layer and the availability. Each Internet node serving as a Simple Network Management Protocol (SNMP) agent collects and maintains a standard set of information in its MIB[2]. The Management Information Tree (MIT) is analogous to the MIB in OSI networks[7], making this approach viable on OSI networks as well.

4 Experimental Results

4.1 Data

The concepts of the intelligent monitoring hierarchy detailed in Section 2 were tested using data collected from a single node on the Internet. The MIB of the node was polled every 15 seconds to collect data. The data was gathered over a period of fourteen days. During this time, we created a network congestion situation by flooding the network with broadcast packets.

A total of four MIB variables were monitored. In the IP layer, the number of input datagrams received from interfaces (ipInReceives) and the number of IP datagrams supplied to IP by local users for transmission (ipOutRequests) were monitored. In the TCP layer, the number of segments received (tcpInSegs) and the number of bytes retransmitted (tcpRetransBytes) were monitored.

4.2 Design of Monitoring Systems

The neural networks used had one input, a hidden layer consisting of three nodes and two outputs. They were trained by backpropagation with 6000 training samples. Artificially generated abnormal data (approx. 10%) was added to the training set to compensate for the lack of abnormal samples in the network data (approx. 0.5%). The abnormal samples were drawn from a uniform distribution.

The HMM transition probabilities were chosen based on discussions with network managers and the collected data. HMMs at the same level in the hierarchy had the same transition matrices. We assumed that the prior probabilities of both states were equally likely.

4.3 Results

Figure 2(a)-(g) illustrates the performance of the intelligent monitoring hierarchy on the network congestion situation we created. Each figure indicates the posterior probability of being in an abnormal state. We use a simple Maximum A Posteriori (MAP) scheme to estimate the state, so any time the p(abnormal) is greater than 0.5, the state is deemed abnormal. As described in Section 2, the posterior state probabilities calculated at level n are used as inputs into the level n+1 system.

We assume that the network manager would be notified (i.e. an alarm would be generated) any time part of the hierarchy reaches an abnormal state. The distinction between this hierarchy and conventional systems is that when an abnormal state is reached, the network manager is notified of the specific anomaly along with the state of the rest of the hierarchy. Thus the network manager has both the low- and high-level information necessary to identify the problem.

We observed two periods of time where the network was so severely congested that traffic was stopped. Both of these times were detected at all layers of the hierarchy, correctly indicating a network problem. Starting from the lowest level, Figs. 2(d)-(g) show the alarms generated after each MIB variable is temporally correlated through the hybrid neural network-HMM. When you go to the next level, the amount of time in the abnormal state decreases as expected. There are only four intervals in which the IP layer is identified as abnormal (Fig. 2(b)) and only one where the TCP layer is identified as abnormal (Fig. 2(c)). At the highest level of the the hierarchy, there are two intervals where the entire network is in an abnormal state (Fig. 2(a)) corresponding to the the situation that we observed.

These results demonstrate that the system is capable of generating temporally correlated alarms at each level, thus providing the network manager with a full picture of the node's health. The power of this system can be illustrated by examining the second abnormal state interval for the network (Fig. 2(a)). During this interval, the IP layer is also in an abnormal state (Fig. 2(b)), but the TCP layer is not (Fig. 2(c)) although the probability of TCP being in an abnormal state is non-zero indicating that things may not be totally normal in that layer. Given the structure of the protocol, we know that when IP is abnormal, problems may "snowball" up to the TCP layer. Therefore, in this case, the monitoring hierarchy has detected the

network problem before the TCP layer has totally become abnormal. We have thus demonstrated the system's ability to notify the network manager of a "snowballing" problem.

5 Conclusion

The intelligent network monitoring hierarchy proposed in this paper has demonstrated the ability to detect unknown anomalies at various levels using only readily available low-level fault models. High-level models which are incomplete and much less accurate are not needed. The information generated at each level can be compiled to create a complete picture of the health of a node. This information can be of great assistance to the network manager by identifying the most general problem so the appropriate actions can be taken.

Future work will address incorporating more of the MIB variables into the monitoring system. As these variables are not all independent, this will involve investigation of other methods of combining the outputs of the HMM.

Acknowledgements

We would like to thank David Durham for his work in generating the network congestion situations and Isabel Nirenberg for her helpful discussions on Internet network management.

The support from National Science Foundation (ECS-9312504) is gratefully acknowledged.

References

[1] G. Jakobson and M.D. Weissman, "Alarm Correlation," *IEEE Network,* November 1993, pp. 52–59.

[2] K. McCloghrie and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II," RFC 1213, March 1991.

[3] M.D. Richard and R.P. Lippmann, "Neural network classifiers estimate Bayesian a posteriori probabilities," *Neural Computation,* 3(4), 1992, pp. 461–483.

[4] I. Rouvellou, "Graph Identification Techniques Applied to Network Management Problems," Ph.D dissertation, Columbia University, 1993.

[5] P. Smyth, "Markov Monitoring with Unknown States," *IEEE Journal on Selected Areas in Communications*, vol. 12, 1994, pp. 1600-1612.

[6] O. Wolfson, S. Sengupta, and Y. Yemini, "Managing Communication Networks by Monitoring Databases," *IEEE Transactions on Software Engineering*, vol. 17, no. 9, 1991.

[7] Y. Yemini, "A Critical Survey of Network Management Protocol Standards," Telecommunications Network management Into the 21st Century, S. Aidarous and T. Plevyak (eds.), New York, NY, IEEE Press, 1994.

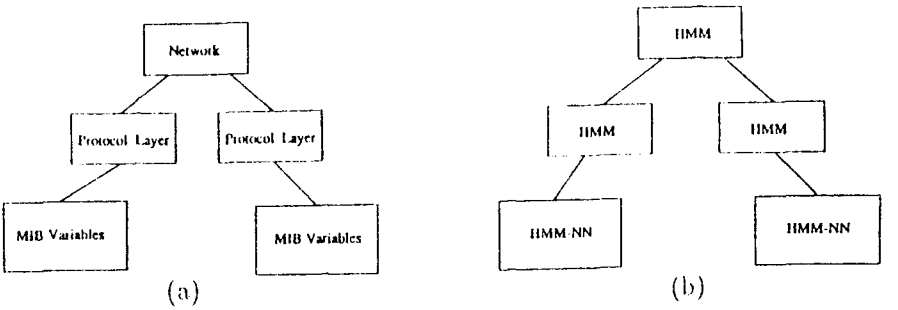


Figure 1: (a) Logical structure of hierarchy and (b) Implementation of hierarchy.

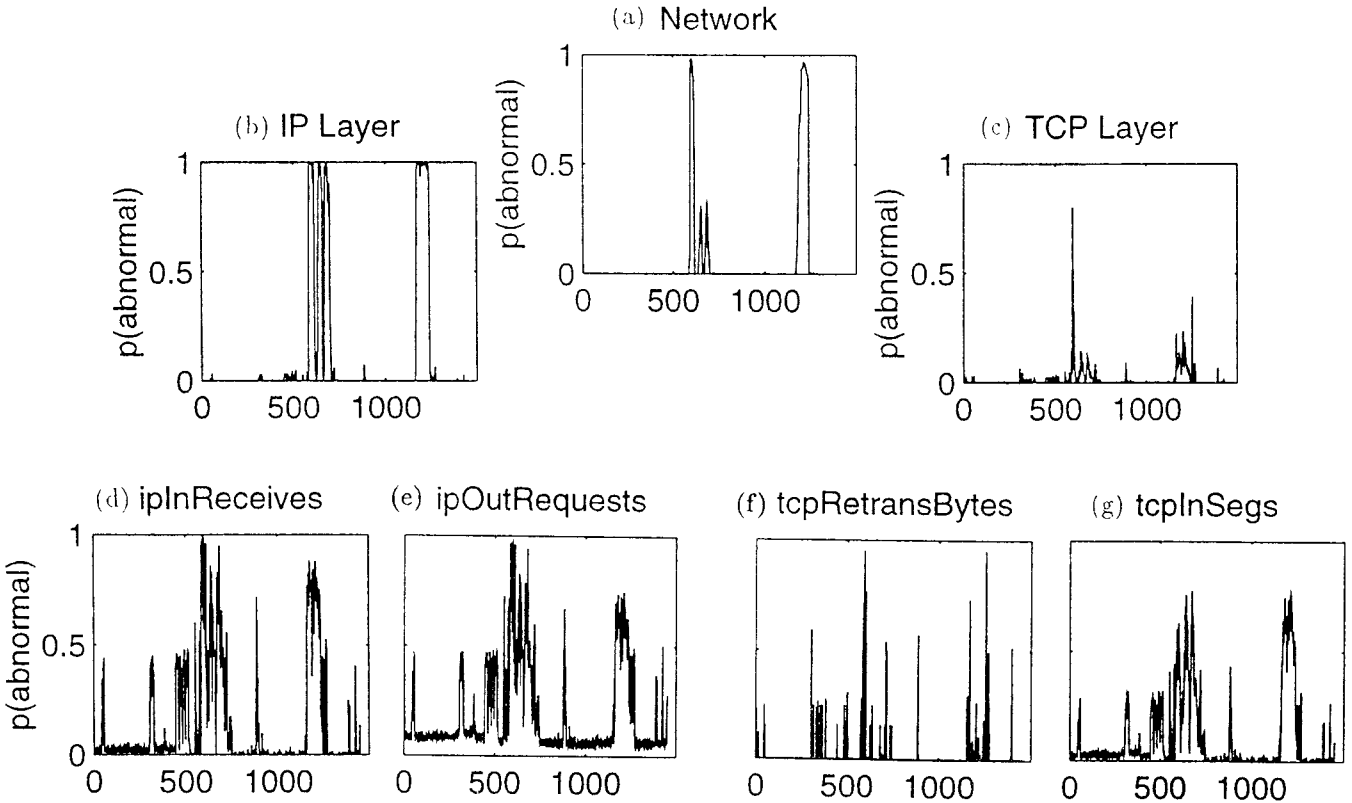


Figure 2: Hierarchy output at each level for a network anomaly

A ROBUST BACKWARD ADAPTIVE QUANTIZER

Dominique MARTINEZ[1,2] & **Woodward YANG**[2]

(1) Laboratoire d'Analyse et d'Architecture des Systèmes
LAAS - CNRS, 7 Av. du Col. Roche, 31077 Toulouse, France.

(2) Harvard University
Division of Applied Sciences
29 Oxford St., Cambridge, MA 02138, USA.

Abstract

This paper describes an adaptive encoder/decoder for efficient quantization of nonstationary signals. The system uses a robust backward adaptive encoding method such that the adaptation of the encoder and decoder is only determined by the transmitted codeword and does not require any additional side information. By incorporating a forgetting parameter, the quantizer is also robust to transmission errors and encoder/decoder mismatches. It is envisioned that practical applications of this algorithm can be used in the design of adaptive codecs (A/D and D/A converters) or as an efficient source coding algorithm for transmission of digitized speech.

1 INTRODUCTION

A typical waveform coder transmits information across a communication channel such that the resulting reconstruction of the signal at the decoder, say u', is as close as possible to the original source u. In this communication context, a quantizer $Q(.)$ can be viewed as an encoder/decoder pair connected to a digital channel. The encoder $\mathcal{E}$ converts a sampled version u_n of the source into a digital form $I_n = \mathcal{E}(u_n)$, sends it over the channel $\mathcal{C}$ to the decoder $\mathcal{D}$ which reconstructs the signal $u'_n = \mathcal{D}(I'_n)$ according to the codeword $I'_n = \mathcal{C}(I_n)$ received. Note that $\mathcal{C}(.)$ corresponds to the deterministic identity mapping for a noiseless channel such that $I'_n = I_n$ and to a probabilistic mapping for a noisy channel. Then, as shown in Fig. 1, $u'_n = Q(u_n) = \mathcal{E} \circ \mathcal{C} \circ \mathcal{D}(u_n)$.

0-7803-2739-X/95 $4.00 © 1995 IEEE

Throughout the paper, signals or parameters of the decoder which are similar to those of the encoder are denoted with primes.

A quantizer is optimal with respect to the probability density function (p.d.f) $p(u)$ of the source to be encoded if it minimizes a r-th power distortion measure $d(u,u') = D_r \equiv \int \mid u - u' \mid^r p(x) \quad dx$ with $0 < r \leq \infty$. In quantization, the most commonly used powers are $r = 1$ (mean absolute error) (see *e.g.* [1]), $r = 2$ (mean squared error) (see *e.g.* [2, 3, 4]) and $r = \infty$ (mean maximum absolute error). In case $p(x)$ is not known, the most widely used design algorithm for scalar quantizers is the (standard) Lloyd I algorithm, which can be extended for minimizing r-th power law distortion (generalized Lloyd I) [5]. However, because they are batch algorithms, the design of the quantizer can only begin after the entire training set is available. By consequence, these algorithms are not able to accommodate on-line changes in the input p.d.f.

In many applications, communication systems may have to carry signals of changing statistics, e.g. speech inputs with different variances. The most efficient way to handle nonstationary inputs is to continuously adapt the encoder/decoder pair in such a way as to match observed local statistics of the input sequence. For performing adaptive quantization, a number of researchers have developed unsupervised competitive learning algorithms (for references, see [6]), of which the Kohonen algorithm is one of the most well-known. Because the weight updates are a function of the input u, this scheme is referred to as forward adaptation and suffers the serious drawback that an excessive amount of side information is required to transmit the updates to the decoder [5].

On the contrary, backward adaptation is primarily of interest for designing adaptive quantizers because it does not require transmitting additional bits. As depicted in Fig. 1, the digital codewords I' and I are now used instead of the input u to adjust the quantizing parameters. Commonly used backward adaptive algorithms are gain or step size adaptive and expand or contract the dynamic range of a time invariant quantizer according to an estimate of the signal variance [7]. These algorithms assume symmetrical zero-mean p.d.f.s of known shape and therefore are not ideal devices for quantizing non stationary inputs.

A new backward adaptive algorithm has been introduced for buiding scalar quantizers "on-line" and without assuming any particular p.d.f's shape [8]. The learning rule, called generalized Boundary Adaptation Rule (BAR_r), minimizes r-th power law distortion D_r in the case of high resolution quantization. Unfortunately, in a communication situation, such a backward adaptive process is sensitive to transmission errors. If $I'_n \neq I_n$ is received at time n, then the decoder will not adapt appropriately and the resulting mismatch between encoder and decoder will remain in the system unless an alignment or recalibration is performed.

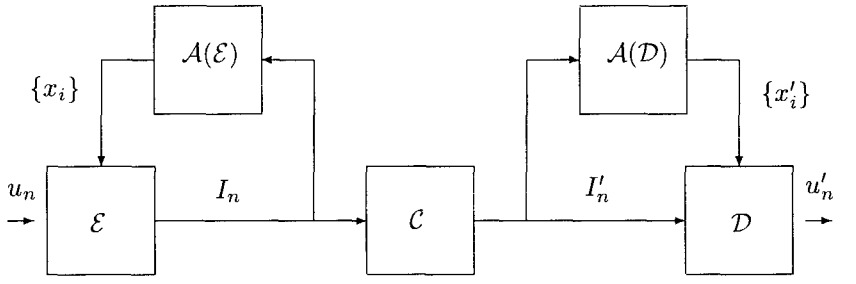


Figure 1: Block diagram of a backward adaptive quantizer. Encoder ($\mathcal{E}$), Channel ($\mathcal{C}$), Decoder ($\mathcal{D}$), Encoder and Decoder Adaptation ($\mathcal{A}(\mathcal{E})$ and $\mathcal{A}(\mathcal{D})$). Prime notations stand for the decoder's parameters. The decoder presents a symmetrical structure in order to track the encoder adaptive parameters.

In this article, we introduce a modified Boundary Adaptation Rule which is robust to channel errors and initial encoder/decoder mismatches.

2 QUANTIZER DESCRIPTION AND ANALYSIS

A regular N-point scalar quantizer $Q(.)$ maps the scalar-valued input u_n into one of N reconstruction levels. The quantizing system of interest is illustrated in Fig. 1. The encoder is specified by an ordered set of boundary points $x_0 < \cdots < x_{i-1} < x_i < \cdots < x_N$ delimiting N disjoint quantization intervals $R_1, \cdots, R_i, \cdots, R_N$, with $R_i = [x_{i-1}, x_i)$. Its output I_n is usually defined by the N-bit binary vector $(\mathbb{1}_{R_1}, \cdots, \mathbb{1}_{R_N})$ in which $\mathbb{1}_{R_i}(u_n) = 1$ if $u_n \in R_i$ and $= 0$ otherwise. In a communication situation however it is usually mapped into a more compact R-bit codeword representation with $R \geq \log_2 N$ for transmission. The decoder has a structure similar to the encoder but with prime notations standing as its parameters. The decoding operation corresponds to $u' = \sum_{i=1}^{N} \mathbb{1}'_{R_i} y'_i$. In high resolution quantizers, the number of quantization intervals N is very large so that the interval lengths are very small and the reconstruction levels y'_i can be approximated by the midpoints of their corresponding quantization intervals: $y'_i = (x'_{i-1} - x'_i)/2$.

The boundary points delimiting the quantization intervals are therefore the only parameters to adapt. At the encoder, the backward adaptation of x_i can be written as:

$$\Delta x_i(n) = x_i(n+1) - x_i(n) = \eta f(I_n) \tag{1}$$

where η is the learning rate, a positive scalar. In its simplest form, the generalized unsupervised learning rule called BAR_r (generalized Boundary Adaptation Rule) reduces to Eq. (1) with:

$$f(I_n) = \delta^r_{i+1} \mathbb{1}_{R_{i+1}} - \delta^r_i \mathbb{1}_{R_i} \tag{2}$$

where $\delta_i = x_{i-1} - x_i$ is the length of the interval R_i. Assuming that for input u_n, $\mathbb{1}_{R_i} = 1$, then only the interval R_i is reduced in size by increasing x_{i-1} and decreasing x_i. A faster rule, called $FBAR_r$, is obtained by updating all boundary points each time an input is presented:

$$f(I_n) = \sum_{k=i+1}^{N} \delta_k^r \frac{\mathbb{1}_{R_k}}{N-i} - \sum_{k=1}^{i} \delta_k^r \frac{\mathbb{1}_{R_k}}{i} \tag{3}$$

$FBAR_r$ and BAR_r minimize r-th power law distortion [8]. At convergence, all the N quantization intervals R_i will have the same distortion $D_r(i) = D_r/N$. This property guarantees an optimal high resolution quantization [9].

To track the encoder parameters, the same learning equations are used at the encoder but, again with prime notations.

3 A ROBUST BOUNDARY ADAPTATION RULE

In the case of a noisy channel, I'_n may be different to I_n resulting in a mistracking of the encoder. Indeed, with Eq. (1), the mismatch from inappropriate adaptations will remain within the system. To help the system readjust, a leakage or "forgetting" factor β needs to be introduced in the learning rule. Eq. (1) therefore becomes:

$$\Delta x_i(n) = \eta f(I_n) - \beta x_i(n). \tag{4}$$

To assess the effect of β on the robustness of the system, lets $d_n(.)$ denote the difference at time n between two similar parameters, one at the encoder and the other at the decoder. $d_n(x_i) = x_i(n) - x'_i(n)$ can be calculated recursively and expressed as:

$$d_n(x_i) = (1-\beta)^n d_0(x_i) + \eta \sum_{k=0}^{n-1} (1-\beta)^{(n-k-1)} d_k(f(I))$$

Clearly, in the absence of transmission errors, $d_n(x_i) = (1-\beta)^n d_0(x_i) \to 0$ when $\beta < 1$ and $n \to \infty$ and the initial encoder/decoder difference $d_0(x_i)$ decays with time. Similarly, the offset $d_k(f(I))$ from each transmission error $I'_k \neq I_k$ decays exponentially with time.

Eq. (4) is equivalent to Eq. (1) at convergence only if $\beta << \eta$. However a high value of β is necessary to improve performance in the presence of transmission errors. Furthermore, larger values of β in Eq. (4) also modify the basic performance criterion which will be no longer an r-th power law distortion D_r.

To maintain the ability of fully recovering from transmission errors while minimizing D_r, we propose the following modified learning rule

$$\Delta x_i(n) = \eta f(I_n) - \beta(x_i(n) - x_i^\star) \tag{5}$$

in which x_i now decays to some (fixed a-priori) nominal value $x_i^\star$. Note that Eq. (5) is identical to (4) when $x_i^\star = 0$ and to (1) when $\beta = 0$. More interesting is the fact that, at convergence, Eq. (5) and (1) are equivalent if $x_i^\star = E[x_i]$ as found by (5) at convergence, no matter the value of β.

The nominal boundary point values should be chosen equivalent for the decoder and for the encoder and be comparable to those given by a quantizer optimized for the long term p.d.f. at hand (e.g. a μ-law quantizer for speech coding or a quantizer optimized for a gaussian p.d.f in case of differential waveform coding assuming gaussian residuals). For high values of β, forgetting will be predominant over adaptation and the boundary points will eventually fluctuate around the nominal values specified by the user.

4 APPLICATION TO WAVEFORM CODING

We have previously shown that, for large N, $FBAR_r$, Eq. (1) and (3), outperforms the generalized Lloyd I algorithm in minimizing r-th power law distortion D_r with $r = 1$ and 2, for a gaussian and an exponential p.d.f. [8]. Here, we show the performance of the modified algorithm (Eq. (5)) in quantizing speech signals with noisy channels. $FBAR_r$ is preferred above BAR_r in waveform coding applications because convergence is $\mathcal{O}(N)$ time faster as noted in [8], and for this reason Eq. (3) will be used in conjunction with Eq. (5) in numerical simulations.

The performance of a quantizer is often assessed in terms of signal-to-noise ratio (SNR) specified in units of decibels (dB):

$$SNR = 10 \ \log_{10} \frac{E[u^2]}{D_2}$$

Another measure of interest reflecting temporal variations of SNR in waveform coding is the segmental signal-to-noise ratio (SEGSNR) which is simply a time-average of the ratio computed for each segment of the input sequence. Typically, appropriate segment length in speech coding is in the order of 16ms [7]. Although the maximization of both SNR and SEGSNR implies the minimization of the mean square error D_2, $FBAR_r$ with $r = 1$ has been employed in our adaptive quantizer for simplicity. To avoid dependency on a particular choice of fixed overload points, x_0 and x_N were taken after each time step as follows: $x_0 = x_1 - \delta_2$ and $x_N = x_{N-1} + \delta_{N-1}$.

Ideally, in presence of a locally stationary input like a speech signal, an adaptive quantizer would adapt quickly to abrupt changes in the input

p.d.f. and stop adapting in the presence of a stationary segment. Fig. 2 shows the effect of the value of the learning rate η on the speed of adaptation of a typical speech signal originated from TIMIT database.

A value of $\eta = 0.49$ causes intantaneous adaptation, i.e. sudden changes following the local waveform peaks. On the contrary, a value of $\eta = 0.049$ causes syllabic adaption, i.e. slower changes that do not follow the local peaks, but only their envelope. The value of η mediates therefore the tradeoff between instantaneous and syllabic adaptation. We have found that a value of $\eta = 0.2$ is acceptable such that $FBAR_1$ tracks the changing statistics of speech with appropriate speed and with high values of SEGSNR. Table 1 compares the performance of $FBAR_1$ ($\eta = 0.2, \beta = 0.0$) with those of fixed quantizers such as uniform and μ-law. The results indicate that $FBAR_1$ gives at least a 4 dB gain in SNR over μ-law. In addition, unlike non-adaptive quantizers, $FBAR_1$ maintains a fairly constant and non-negative SNR-versus-time characteristic for all input speech segments. The resulting SEGSNR value attained is therefore superior to that of a μ-law or uniform quantizer, especially at low rates: 12.71 dB $\pm$0.044 instead of 5.11 and -5.46 dB for 3 bits, respectively.

Mismatches and long term drifts in hardware characteristics are a common drawback in analog-to digital converters. The self-correcting capability of our adaptive quantizer from initial encoder/decoder mismatches is illustrated in Table 2 for an initial mismatch $m_0 = 1.5$ and for different values of the forgetting factor β. The quantizer mismatch is defined at time n by $m_n = \sum_{i=1}^{N}[d_n(x_i)]^2$. In this particular example, the time average mismatch M was approximately $(m_0 10^{-6})/\beta$ for the range $10^{-5} \leq \beta \leq 10^{-1}$. For high values of β, the SNR and SEGSNR values are very closed to those obtained with a fixed quantizer having nominal values as its boundary points. This is easily verified by comparing the SNR values given in Table 2 for $\beta = 0.1$ and μ-law as nominal values, to those given in Table 1 for a fixed 4-bit μ-law quantizer. Similarly for $\beta = 0.1$ and zero as nominal values, the algorithm yields a constant zero output signal $u' = 0$ such that $D_2 = E[(u - u')^2] = E[u^2]$ and therefore SNR=0 dB. Use of a forgetting factor, together with μ-law nominal values, in $FBAR$ increases the coder robustness and performance: SNR=16.79 dB and SEGSNR=14.09 dB for $\beta = 10^{-3}$ over SNR=1.20 dB and SEGSNR = -6.76 dB for $\beta = 0$.

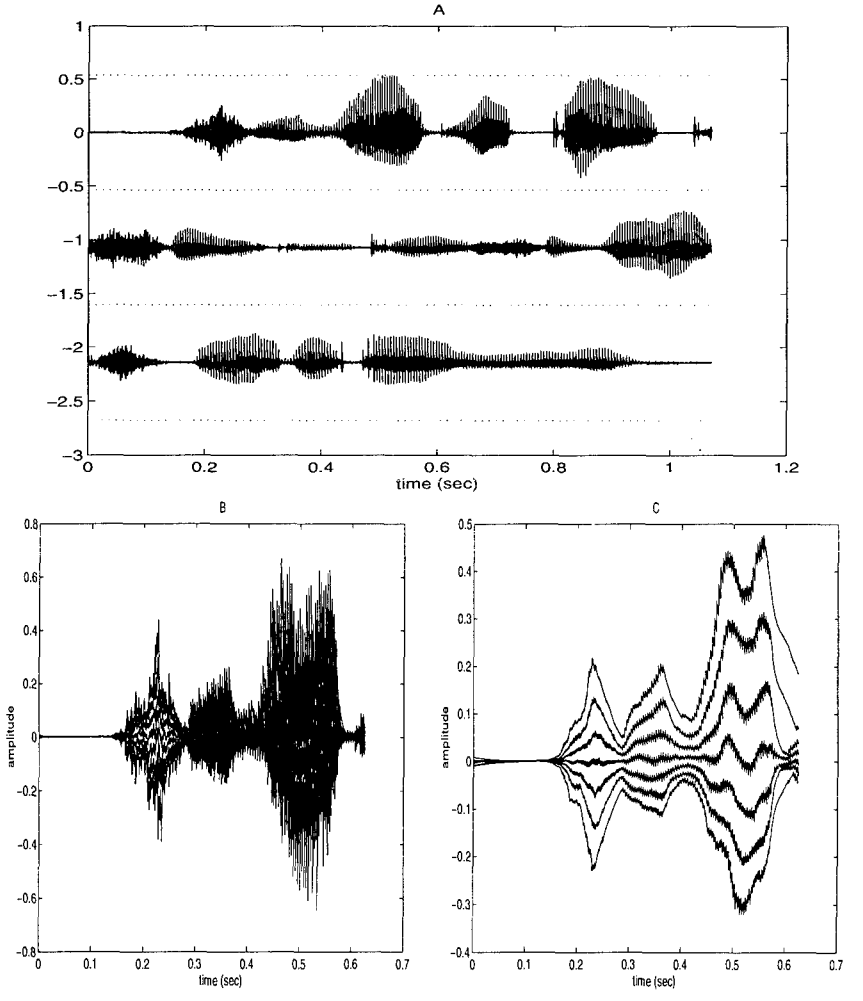


Figure 2: Instantaneous and Syllabic adaptation in quantizing speech signal.

Top: input speech sentence /She had your dark suit in greasy wash water all year/ of 51406 samples used in the simulations. *Bottom left:* Time evolution of the boundary points for a 3-bit instantaneous adaptive quantizer ($\eta = 0.49, \beta = 0.0, r = 1$). The quantizer has been adapted over the entire duration of the signal but only the first 0.6 seconds of the evolution are represented here. *Bottom right:* Time evolution of the boundary points for a 3-bit syllabic adaptive quantizer ($\eta = 0.049, \beta = 0.0, r = 1$).

Quantizer type ↓	$R = \log_2 N$ bits →	3	4	5
Uniform	SNR	1.95	8.64	15.06
	SEGSNR	-5.46(*)	1.21	7.74
μ-law ($\mu = 255$)	SNR (**)	8.44	13.51	19.67
	SEGSNR	5.11	11.52	18.10
$FBAR_1$ (***)	SNR	12.10 (± 0.052)	18.41 (± 0.057)	23.14 (± 0.091)
	SEGSNR	12.71 (± 0.046)	17.53 (± 0.044)	22.06 (± 0.062)

Table 1: SNR and SEGSNR values attained in 3-,4- and 5-bit coding of example speech signal. All entries in dB.
(*) The presence of zero amplitude input segments may yield occasionally extremely large negative SEGSNR values.
(**) Experimental SNR values for μ-law agree with those found by the theoretical formula $SNR_{\mu-\text{law}}(dB) = 6.02R - 10.1$ derived by Smith for $\mu = 255$ [10].
(***) For $FBAR_1$ results, the values given are averages over 20 runs with their standard deviations.

β	Nominal values $\{x_i^\star\}$	Average mismatch M	SNR (dB)	SEGSNR (dB)
0.0	——	0.15	1.20	-6.76
1E-1	0.0	1.2E-5	0.0	-0.23
1E-1	μ-law	1.2E-5	14.36	11.82
1E-2	μ-law	1.4E-4	16.11	13.09
1E-3	μ-law	1.5E-3	16.79	14.09
1E-4	μ-law	1.5E-2	10.66	9.40
1E-5	μ-law	0.94E-1	3.18	-4.61

Table 2: SNR and SEGSNR (dB) values attained in 4-bit coding of example speech signal in presence of an initial encoder/decoder mismatch $m_0 = 0.15$.

The effect of transmission errors is evaluated by simulating a binary symmetric channel which produces errors with a probability $p_e = 10^{-3}$. Because the probability of one or more errors in the received R-bit codeword is approximately equal to Rp_e for small p_e [7], there were approximately 200 transmission errors over the entire duration of the signal. For the channel studied, performance results are given in Table 3 for a 4-bit quantizer with μ-law nominal values and different values of the forgetting factor β. Results clearly indicate that the use of a forgetting factor yields significant improvements in SNR and SEGSNR. It is reasonable to think that large displacements caused by the most significant bit (MSB) in error, most significantly contribute to a decrease in SNR. However, Table 3 reports that when the MSB is protected, improvements in adaptating are not greater than 1.5 dB when $\beta \neq 0$.

β	Average mismatch	Performance (dB)	
		SNR	SEGSNR
0.0	1.52 ± 1.68 [0.75 ± 1.13]	-6.66 ± 5.35 [-1.44 ± 6.67]	-7.25 ± 6.26 [-4.10 ± 6.51]
1E-1	1.30E-5 ± 2.0E-6 [0.80E-5 ± 1.0E-6]	12.63 ± 0.20 [13.28 ± 0.19]	9.13 ± 0.36 [10.38 ± 0.25]
1E-2	1.04E-4 ± 1.9E-5 [6.40E-5 ± 1.0E-5]	13.91 ± 0.32 [14.98 ± 0.15]	10.24 ± 0.30 [11.70 ± 0.18]
1E-3	1.12E-3 ± 4.8E-4 [8.05E-4 ± 3.2E-4]	15.35 ± 0.50 [16.70 ± 0.37]	11.36 ± 0.45 [12.92 ± 0.39]
1E-4	6.54E-2 ± 5.2E-2 [6.91E-2 ± 5.3E-2]	5.88 ± 3.43 [5.62 ± 3.12]	3.29 ± 3.25 [2.67 ± 2.36]
1E-5	5.07E-1 ± 4.4E-1 [6.11E-1 ± 6.9E-1]	-2.51 ± 4.15 [-2.61 ± 4.58]	-4.27 ± 4.69 [-4.58 ± 4.71]

Table 3: SNR and SEGSNR (dB) values attained in a 4-bit coding of example speech signal in presence of a noisy binary symetric channel with $p_e = 10^-3$. The quantizers have μ-law nominal values but different values of forgetting factor β. The values given are averages over 20 runs with their standard deviations. The values in brackets are given when the MSB is protected.

5 CONCLUSION

In this article, a related robust backward adaptive quantizer has been introduced for waveform coding in the presence of noisy channels. It has been shown that the forgetting term used in the learning rule dissipates transmission errors or initial quantizer mismatches while maintaining the ability to adapt itself to changes in the input p.d.f. and minimizing r-th power law distortion measure. Furthermore, if a hardware implementation is envisioned, the unsupervised learning rule will be able to adjust itself to initial mismatches and to long term drifts in component characteristics, hence avoiding the problem of pair-wise tuning parameters between encoders and decoders.

REFERENCES

[1] S.A. Kassam, "Quantization based on the mean-absolute-error criterion," **IEEE Trans. on Communications**, Vol. COM-26, pp. 267-270, 1978.

[2] S.P. Lloyd, "Least squares quantization in PCM's," **Bell Telephone Laboratories Paper**, Murray Hill, NJ, 1957.

[3] S.P. Lloyd, "Least squares quantization in PCM," **IEEE Trans. Inform. Theory**, Vol. IT-28, pp. 127-135, 1982.

[4] J. Max, "Quantizing for minimum distortion," **IRE Trans. on Inform. Theory**, Vol. IT-6, pp. 7-12, 1960.

[5] A. Gersho and R.M. Gray, **Vector quantization and signal compression**, Boston, Dordrecht, London: Kluwer, 1992.

[6] J. Hertz, A. Krogh, and R.G. Palmer, **Introduction to the theory of neural computation**, Reading MA: Addison-Wesley, 1991.

[7] N.S. Jayant and P. Noll, **Digital coding of waveforms**, Englewood Cliffs, New Jersey: Prentice-Hall, 1984.

[8] D. Martinez and M.M. Van Hulle, "Generalized Boundary Adaptation Rule for minimizing r-th power law distortion in the high resolution case," **Neural Networks**, Accepted for publication, 1995.

[9] A. Gersho, "Asymptotically optimal block quantization," **IEEE Trans. Inform. Theory**, Vol. IT-25, no. 4, pp. 373-380, July 1979.

[10] B. Smith, "Instantaneous companding of quantized signals," **Bell System Tech. J.**, Vol. 36, no. 3, pp. 653-709, 1957.

A MAXIMUM PARTIAL LIKELIHOOD FRAMEWORK FOR CHANNEL EQUALIZATION BY DISTRIBUTION LEARNING [1]

Tülay Adalı†, *Xiao Liu*†, *Ning Li*†, *and M. Kemal Sönmez*‡
†Information Technology Laboratory, Dept. of Electrical Engineering
University of Maryland Baltimore County, Baltimore, MD 21228-5398
‡Institute for Systems Research, University of Maryland
College Park, MD 20742
{adali,xliu,nli2}@engr.umbc.edu kemal@isr.umd.edu

Abstract- We present the general formulation for adaptive equalization by distribution learning [1] in which conditional probability mass function (pmf) of the transmitted signal given the received is parametrized by a general neural network structure. The parameters of the pmf are computed by minimization of the accumulated relative entropy (ARE) cost function. The equivalence of ARE minimization to maximum partial log-likelihood (MPLL) estimation is established under certain regularity conditions which enables us to bypass the requirement that the true conditionals be known. The large sample properties of MPLL estimator are obtained under further regularity conditions, and the binary case with sigmoidal perceptron as the conditional pmf model [1, 2] is shown to be a special case of the new framework. Results are presented which show that the multilayer perceptron (MLP) equalizer based on ARE minimization can always recover from convergence at the wrong extreme whereas the mean square error (MSE) based MLP can not.

INTRODUCTION

As more complex channels are required to carry increasing amounts of data in today's demanding communications applications, the need to develop more sophisticated equalization schemes has become more evident. To overcome the inherent limitation of linear equalizers, a number of neural network adaptive equalizers have been introduced (see e.g. [5, 6, 9]), and it is shown that these equalizers can successfully equalize nonlinear channels where linear equalizers might fail. The neural network equalizer also offers the advantage of low-power low-complexity analog hardware implementation which is particularly important in portable applications. These neural network equalizers view channel equalization as a classification problem and are based on the traditional mean square error (MSE) performance criterion. Recently, we have introduced a new approach to channel equalization which is based on probability distribution learning [1], and uses relative entropy (RE) between the true and estimated conditional probability density functions as the performance measure to be minimized. The conditional probability mass function

[1]Research supported in part by Engineering Foundation grant RI-A-94-08.

0-7803-2739-X/95 $4.00 © 1995 IEEE

(pmf) of the transmitted signal given the received signal is parametrized by a general neural network architecture. It is shown that when multilayer perceptrons (MLP) are chosen as the parametrized model, the equalizer can successfully combat multi-path [1] and nonlinear distortions [2], and can always recover from convergence at the wrong extreme as opposed to the MSE based MLP's [1, 2].

In this paper, we extend our distribution learning formulation to finite symbol alphabets by working in the *partial likelihood* framework. Partial likelihood, a relatively new method in estimation theory, allows for inference as the time unfolds. Hence it bypasses the problem with maximum likelihood or quasi-maximum likelihood estimation which require that the auxiliary information be known in full throughout the period of observation when they are extended to dependent observations [4]. The general formulation we present for channel equalization here encompasses both supervised and unsupervised (blind) mode of operation for a general neural network structure. In this framework, adaptive channel equalization can be considered as a conditional pmf estimation problem by accumulated relative entropy (ARE) minimization which we show to be equivalent to *maximum partial log-likelihood* (MPLL) statistical estimation problem. Unlike ARE minimization, MPLL estimation does not require that we know the true conditionals which in general are never available, hence the parameters of the conditional distribution model can be directly learned on the chosen neural network model. We show that the consistency and asymptotic normality of MPLL estimator can be obtained under further regularity conditions. In [2], we consider a simple binary communication channel equalization problem and use the sigmoidal perceptron to parametrize the conditional pmf. We then employ first order stochastic approximation of the true conditionals to write the stochastic variant of the RE cost function, and then note its equivalence to MPLL estimation. Here, we consider the binary case with the sigmoidal model as an example and show that it is a special case of the new framework. Also, for the perceptron model, we present simulation studies which show that the ARE based MLP equalizer can always recover from convergence at the wrong extreme whereas the MSE based MLP can not. This property of the RE based equalizer is discussed in [2] within an extension of the *well-formed* cost functions framework of Wittner and Denker [11].

CHANNEL EQUALIZATION BY DISTRIBUTION LEARNING

We formulate adaptive equalization problem as follows: A sequence of symbols $x(n)$, taking values from a finite alphabet $\mathcal{S} = \{a_0, a_1, ..., a_M\}$, is transmitted through a channel h which acts as a nonlinear operator on the incoming signal. Let $\mathcal{F}_n$ be the σ-field generated by the events of the form $\mathbf{z}(n) = [x(n-1), \cdots, x(1), x(0); y(n), \cdots, y(1), y(0)]$, where, $y(n)$'s are the time dependent covariates of $x(n)$. Typically, $y(n)$ is the noise corrupted channel output. A

common model for the channel output would be $y(n) = h(\mathbf{x}_K(n)) + v(n)$ where $v(n)$ is the additive noise component, $\mathbf{x}_K(n) = [x(n), x(n-1), ..., x(n-K+1)]$, and $h : \mathbf{R}^K \rightarrow \mathbf{R}$. If $\mathcal{F}_n$ does not include the transmitted sequence $x(n)$ but only its covariates, this results in unsupervised (blind) mode of operation for the equalizer. Thus $\mathcal{F}_n = \sigma\{1, \mathbf{z}(n)\}$ represents all that is known to the observer at time n, and $\mathcal{F}_{n-1} \subset \mathcal{F}_n$. Note that since $\mathcal{F}_n$ includes the entire history $p_\theta(x|\mathcal{F}_n)$ can have a recurrent structure as well.

Our aim is to estimate the conditional pmf $p(x|\mathcal{F}_n), \forall x \in \mathcal{S}$. We parametrize the conditional probability by a neural network as follows:

$$p_\theta(x|\mathcal{F}_n) = f(x, c(\boldsymbol{\theta}), g(z_N(n), \boldsymbol{\theta})). \tag{1}$$

Here, $\boldsymbol{\theta}$ is the vector of network weights, $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ where $\boldsymbol{\Theta}$ is a compact parameter set and $\mathbf{z}_N(n)$ is a subset of $\mathbf{z}(n)$ containing the most recent N values of $\mathbf{z}(n)$. The term $g(\mathbf{z}_N(n), \boldsymbol{\theta})$ is the output of the neural network, $f(\cdot)$ and $g(\cdot)$ are continuous differentiable functions, and $c(\boldsymbol{\theta})$ and $f(\cdot)$ are chosen such that $\sum_{x \in \mathcal{S}} p_\theta(x|\mathcal{F}_n) = 1$.

The relative entropy (RE), or the Kullback-Leibler distance, [8] a fundamental information theoretic measure of how accurate the estimated conditional pmf is an approximation to the true conditional pmf,

$$D_n(p_{\theta_0} \| p_\theta) = \sum_{x \in \mathcal{S}} p_{\theta_0}(x|\mathcal{F}_n) \ln \frac{p_{\theta_0}(x|\mathcal{F}_n)}{p_\theta(x|\mathcal{F}_n)} \tag{2}$$

arises as the natural cost function for this formulation. Note that it is non-negative, and is equal to zero only when $p_{\theta_0} = p_\theta$. In (2) we assumed that $\boldsymbol{\theta}_0$ is the weight vector for which $f(\cdot)$ achieves the true conditional pmf. The goal is then to learn $\boldsymbol{\theta}$ which minimizes the *accumulated* relative entropy (ARE) given by

$$\mathcal{I}_n = \sum_{i=1}^{n} D_i(p_{\theta_0} \| p_\theta) \tag{3}$$

in the sequence of observations $\mathcal{F}_1, \mathcal{F}_2, \cdots, \mathcal{F}_n$. However, note that the minimization of this cost function requires that the true conditionals, or that $\boldsymbol{\theta}_0$ be known. In the next section we show that the ARE minimization problem is equivalent to MPLL estimation which allows us to overcome this problem.

MAXIMUM PARTIAL LIKELIHOOD ESTIMATION

The optimal network parameters $\boldsymbol{\theta}_0$ have the fundamental information theoretic interpretation that they minimize the Kullback-Leibler information given the chosen architecture and the ARE performance measure (3). Thus viewing learning as related to Kullback-Leibler information minimization in this way implies that learning is a *maximum likelihood* statistical estimation procedure for independent observations [10]. Though this may be extended

to dependent data by discounting the dependence structure in some sense, this still requires that the auxiliary information be known in full throughout the period of observation [7]. It is obvious that this requirement can not be satisfied in data communications. *Partial likelihood* (PL) can bypass this problem by allowing inference from the available information.

The distribution learning problem posed in the previous section can be cast as a MPLL estimation problem. To show this, we define

$$r_i = \ln \frac{p_{\theta_0}(x|\mathcal{F}_i)}{p_{\theta}(x|\mathcal{F}_i)} \quad \text{and} \quad \mathcal{J}_n = \sum_{i=1}^{n} Var_{\theta_0}(r_i|\mathcal{F}_i)$$

and based on the theory of partial likelihood [12], show the following:

Theorem 1: If there exist a constant $\delta > 0$, $\alpha_n \uparrow \infty$, continuous functions $f(\cdot)$ and $g(\cdot)$ such that

$$P(\mathcal{I}_n/\alpha_n > \delta) \longrightarrow 1 \quad \text{and} \quad \mathcal{J}_n/\alpha_n \longrightarrow_p 0 \tag{4}$$

then ARE minimization is equivalent to MPLL estimation, i.e.,

$$arg\left(\min_{\theta} \mathcal{I}_n\right) = arg\left(\max_{\theta} \bar{\mathcal{L}}_n\right)$$

where $\mathcal{L}_n = \prod_{i=1}^{n} p_{\theta}(x|\mathcal{F}_i)$ is the partial likelihood function and $\bar{\mathcal{L}}_n = \ln \mathcal{L}_n$ is the partial log-likelihood. (Proof is given in the appendix.)

It then suffices to maximize $\bar{\mathcal{L}}_n$ to estimate the conditional distribution, and the value $\hat{\boldsymbol{\theta}}$, maximizing $\bar{\mathcal{L}}_n$, provides an estimate of the true parameter $\boldsymbol{\theta}_0$. Consistency and asymptotic normality are essential properties to ensure that as the network experience grows, the probability of the network approximation error exceeding any specified level tends to zero. For the parametrized model of (1) we show the following large sample properties of the MPLL estimator:

Theorem 2: For $f(\cdot)$ and $g(\cdot)$ as given in Theorem 1, assume conditions given in (4) hold and that the first and second order derivatives of $f(\cdot)$ and $g(\cdot)$ exist and are continuous. Then if there exist a $\beta_n \uparrow \infty$ and positive definite matrices Q and Q_1 such that

$$\beta_n{}^{-1} U_n(\boldsymbol{\theta}_0) \longrightarrow_p Q_1 \quad \text{and} \quad \beta_n{}^{-1} V_n(\boldsymbol{\theta}_0) \longrightarrow_p Q \tag{5}$$

where

$$U_n(\boldsymbol{\theta}) = \sum_{i=1}^{n} E(\nabla u_i \nabla u_i{}^T), \quad V_n(\boldsymbol{\theta}) = \sum_{i=1}^{n} \nabla u_i \quad \text{and} \quad u_i = \nabla \ln p_{\theta}(x|\mathcal{F}_i) \tag{6}$$

then, we can guarantee that $\hat{\boldsymbol{\theta}}$ is almost surely unique for all sufficiently large n and as $n \rightarrow \infty$,

(i) $\hat{\boldsymbol{\theta}} \rightarrow \boldsymbol{\theta}_0$ in probability,
(ii) $\sqrt{\beta_n}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0) \rightarrow \mathcal{N}[\mathbf{0}, \Lambda]$ in distribution,
where $\Lambda = Q^{-1}Q_1Q^{-1}$ is the information matrix per observation for estimating the true parameter $\boldsymbol{\theta}_0$. (Proof is given in the appendix.)

EXAMPLE: THE BINARY ALPHABET

Consider the adaptive channel equalization problem where the probability that the transmitted signal $x(n) = 1$ from the alphabet $\{0, 1\}$ is to be determined from a training sequence, given the finite past of the received signal: $\mathbf{y}_N(n) = [y(n), y(n-1), ..., y(n-N+1)]$, i.e., $\mathbf{z}_{N+1} = [x(n), \mathbf{y}_N(n)]$. The conditional pmf $p_\theta : \mathbf{R}^N \longrightarrow [0, 1]$ is parametrized such that

$$p_\theta(x(n) = 1|\mathcal{F}_n) = g\left(\boldsymbol{\theta}^T \mathbf{y}_N(n)\right)$$

where $g(\cdot)$ is a differentiable non-linearity such that $g'(s) > 0$ for all s and can be chosen as $g(s) = 1/(1 + e^{-s})$. The pmf is then written as

$$f(\cdot) = g(\cdot)^{x(n)}(1 - g(\cdot))^{1-x(n)}.$$

This is the sigmoidal perceptron model we used in [1].

We can reformulate $f(\cdot)$ as

$$f(\cdot) = \exp(x(n)\gamma_n(\boldsymbol{\theta}) - b_n(\boldsymbol{\theta})) \tag{7}$$

where $\gamma_n(\boldsymbol{\theta}) = \boldsymbol{\theta}^T \mathbf{z}_N(n)$ and

$$b_n = \boldsymbol{\theta}^T \mathbf{z}_N(n) - \ln \frac{1}{1 + \exp(-\boldsymbol{\theta}^T \mathbf{y}_N(n))}.$$

For this exponential model (7), we have

$$r_n(\boldsymbol{\theta}) = -x(n)(\gamma_n(\boldsymbol{\theta}) - \gamma_n(\boldsymbol{\theta}_0)) + b_n(\boldsymbol{\theta}) - b_n(\boldsymbol{\theta}_0)$$

$$E(r_n|\mathcal{F}_n) = b_n(\boldsymbol{\theta}) - b_n(\boldsymbol{\theta}_0) - b'(\boldsymbol{\theta}_0)(\gamma_n(\boldsymbol{\theta}) - \gamma_n(\boldsymbol{\theta}_0))$$

$$Var(r_n|\mathcal{F}_n) = b''(\bar{\boldsymbol{\theta}})(\gamma_n(\boldsymbol{\theta}) - \gamma_n(\boldsymbol{\theta}_0))^2$$

where each $\bar{\theta}_i$ is a value between θ_{0i} and θ_i, and the prime denotes the derivative with respect to θ.

By Lemma 3A [12]:

$$\alpha_n{}^{-2} \sum_{i=1}^{n} (\gamma_i(\theta) - \gamma_i(\theta_0))^2 \rightarrow_p 0.$$

Since $\alpha_n{}^{-1} \sum_{i=1}^{n} (\gamma_i(\theta) - \gamma_i(\theta_0))^2$ is locally uniformly bounded away from zero conditions given in (4) hold and ARE minimization for this problem is

equivalent to MPLL estimation. For this model, the consistency conditions are also satisfied [7].

Dynamics of the LRE algorithm

In [1], we consider the binary case and show that $\tilde{\mathcal{I}}_n(\theta) = -\tilde{\mathcal{L}}_n(\theta)$ where $\tilde{\mathcal{I}}_n$ is the stochastic relative entropy (SRE) cost function which results when we employ first order stochastic approximations for the true conditionals. In this paper, we consider the sigmoidal perceptron as a special case of the new formulation and establish the equivalence of ARE minimization to MPLL estimation under the conditions given in (4). The parameters of the conditional pmf model can be estimated by minimizing the SRE or by maximizing PLL in a number of ways, gradient descent (ascent) learning is one popular alternative. We derive the least relative entropy (LRE) algorithm by gradient descent minimization of the SRE cost function for the single layer perceptron and show that it successfully equalizes multipath channels in [1]. The general formulation for distribution learning with the MLP model is presented in [2] and is applied to the equalization of of nonlinear channels.

The properties of gradient descent learning on the SRE cost function is considered in [1, 2]. Particularly, it is shown that the SRE cost function for single layer perceptron is a *well-formed* cost function in the sense of Wittner and Denker [11] and hence gradient descent learning on this cost function is guaranteed to find a solution. As is well known, there is no such guarantee with the MSE cost function when used on MLP's, even on those without any hidden units. The dynamics of gradient descent learning on the SRE cost function is also studied by considering its parameter updates [2] and it is shown that for LRE updates the backpropagated output error is always a non-vanishing control signal and hence the algorithm can recover from convergence at the wrong extreme while the MSE based MLP can not. In this paper, we present a simulation study to demonstrate this fact.

Consider a binary pulse amplitude modulation (PAM) data transmission system. An abrupt change in the channel response happens during training of the equalizer and causes misclassifications after initial convergence. We model the nonlinear channel as a multipath channel ($H(z) = 1 + 0.5z^{-6} + 0.25z^{-16}$) followed by a nonlinearity $0.5(.)^3$, and the PAM communication system has 8 bits per sample with Nyquist pulse shaping. We implement the LRE algorithm for binary alphabet given in [2] and the gradient descent minimization of the MSE on the same MLP structure for equalization of the given channel. Both algorithms have a 3-8-1 MLP structure. In Figure 1(a), we show the bit error rate (BER) curves for the equalization of this channel which show that both algorithms do an equally good job of partitioning the decision region. What is notable is that when we introduce an abrupt change (an exact sign change) in the channel characteristics after 150 iterations, causing the decision region to rotate suddenly the LRE can very rapidly adapt

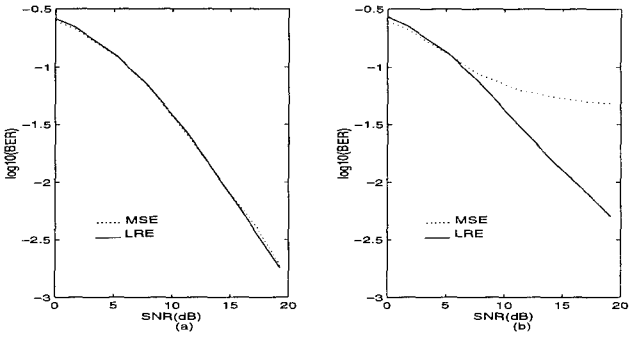


Figure 1: BER Comparison for MSE and LRE MLP Equalizers (a) without (b) with an abrupt change at 150 iterations

to this new operating condition. Starting from the very first iteration after the change it can follow the changes by adapting both its hidden and output layer weights in a few iterations. As we can observe in Figure 1(b), MSE produces many wrong decisions before it can adapt to this new operating condition. In Figure 2(a), we show the transient characteristics of both algorithms with the abrupt change at 150 iterations at a signal to noise ratio (SNR) of 19 dB. As seen in the figure, LRE can recover from convergence at the wrong extreme very effectively whereas MSE based MLP needs a considerable amount of time for the same task. Note that both algorithms have not fully converged at 150 iterations, and if the sudden change causing misclassifications occurs later MSE based MLP might not be able to recover. This is shown in Figure

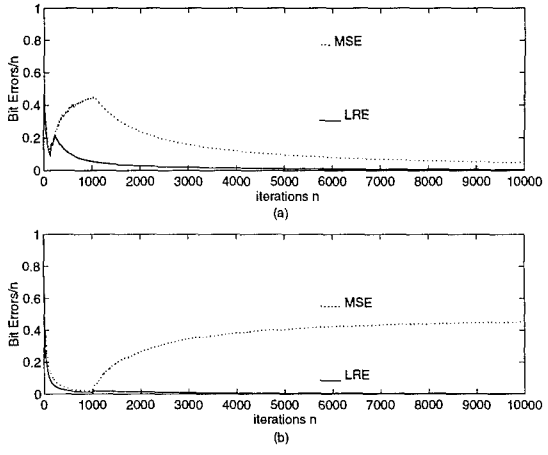


Figure 2: Recovery Characteristics for MSE and LRE MLP Equalizers with an abrupt change at (a) 150 (b) 1000 iterations (SNR =19 dB)

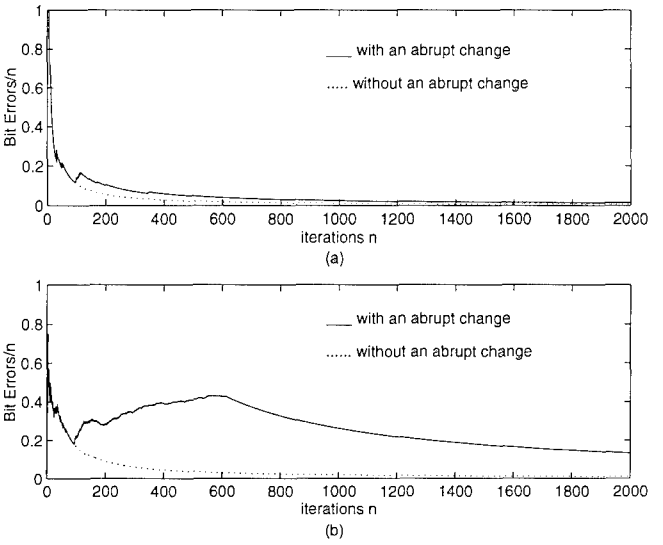


Figure 3: Recovery and Convergence Characteristics for (a) LRE (b) MSE MLP Equalizers (with abrupt change at $n = 100$, SNR =19 dB)

2(b) by introducing the sudden change at iteration 1000. Again LRE can very rapidly adapt to the new operating condition, rapidly recovering from convergence at the wrong extreme. Figures 3 and 4 show the convergence and recovery characteristics of both MLP equalizers (ARE and MSE based) with and without the abrupt change when the change occurs at 100 and 1000 iterations respectively.

APPENDIX

Proof of Theorem 1: Let $\mathcal{R}_n = \sum_{i=1}^{n} r_i$, by Theorem 2A [12]

$$\mathcal{R}_n / \mathcal{I}_n \longrightarrow_p 1.$$

Therefore, $\forall \epsilon > 0$, $\exists N$ for $n \geq N$, for any $\boldsymbol{\theta} \in \boldsymbol{\Theta}$, we have

$$\mathcal{I}_n(\boldsymbol{\theta}_n{}^*)(1-\epsilon) < \mathcal{R}_n(\bar{\boldsymbol{\theta}}_n{}^*) < \mathcal{I}_n(\boldsymbol{\theta}_n{}^*)(1+\epsilon)$$

$$\mathcal{I}_n(\boldsymbol{\theta}_n{}^*)(1-\epsilon) < \mathcal{R}_n(\boldsymbol{\theta}_n{}^*) < \mathcal{I}_n(\boldsymbol{\theta}_n{}^*)(1+\epsilon)$$

where $\boldsymbol{\theta}_n{}^*$ and $\bar{\boldsymbol{\theta}}_n^*$ are the values which minimize $\mathcal{I}_n$ and $\mathcal{R}_n$ respectively. Since ϵ is arbitrary, for sufficiently large N, we have

$$\mathcal{I}_n(\boldsymbol{\theta}_n{}^*) = \mathcal{R}_n(\bar{\boldsymbol{\theta}}_n{}^*) = \mathcal{R}_n(\boldsymbol{\theta}_n{}^*)$$

almost surely on $\boldsymbol{\Theta}$. In addition, we can express $\mathcal{R}_n(\boldsymbol{\theta}_n{}^*)$ as

$$\mathcal{R}_n(\boldsymbol{\theta}_n{}^*) = \bar{\mathcal{L}}_n(\boldsymbol{\theta}_0) - \bar{\mathcal{L}}_n(\boldsymbol{\theta}_n{}^*)$$

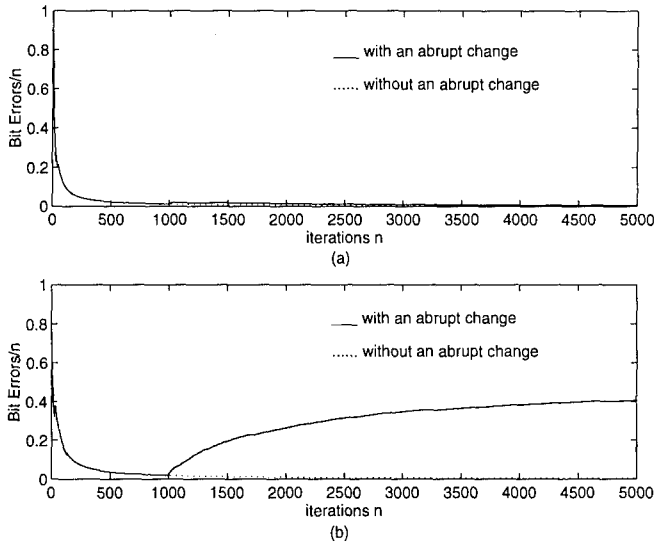


Figure 4: Recovery and Convergence Characteristics for (a) LRE (b) MSE MLP Equalizers (with abrupt change at $n = 1000$, SNR =19 dB)

which implies: $arg\,(\min_\theta \mathcal{I}_n) = arg\,(\max_\theta \bar{\mathcal{L}}_n)$.

Proof of Theorem 2: We have $V_n(\boldsymbol{\theta}) = \nabla\nabla^T \mathcal{L}_n(\boldsymbol{\theta})$ (6). Since conditions given in (5) hold, $\bar{\mathcal{L}}_n(\boldsymbol{\theta})$ is concave with respect to $\boldsymbol{\theta}$ (Theorem 2E [12]). Therefore $\hat{\boldsymbol{\theta}} \rightarrow \boldsymbol{\theta}_0$ in probability.

Let $\mathcal{S}_n(\boldsymbol{\theta}) = \sum_{i=1}^{n} u_i(\boldsymbol{\theta})$, then, we have

$$E(u_i|\mathcal{F}_i) = 0 \quad \text{and} \quad U_n(\boldsymbol{\theta}) = -\sum_{i=1}^{n} E(v_i|\mathcal{F}_i)$$

therefore $\mathcal{S}_n(\boldsymbol{\theta})$ is a martingale. By considering the first two terms in the Taylor expansion of $\nabla\bar{\mathcal{L}}_n(\boldsymbol{\theta}_0)$ we can write

$$0 = \nabla\bar{\mathcal{L}}_n(\boldsymbol{\theta}_0) \approx \mathcal{S}_n(\boldsymbol{\theta}_0) + (\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0)V_n(\boldsymbol{\theta}_0)$$

then

$$\sqrt{\beta_n}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0) \approx \left(\beta_n{}^{-\frac{1}{2}}\mathcal{S}_n(\boldsymbol{\theta}_0)\right)\left(-\beta_n V_n(\boldsymbol{\theta}_0)^{-1}\right).$$

Since the conditions given in (5) are satisfied and that $\mathcal{S}_n(\boldsymbol{\theta})$ is a martingale, by invoking Martingale central limit theorem [3], we have

$$\beta_n{}^{-\frac{1}{2}}\mathcal{S}_n(\boldsymbol{\theta}_0) \rightarrow_D \mathcal{N}[0, Q_1]$$

therefore

$$\sqrt{\beta_n}(\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}_0) \rightarrow_D \mathcal{N}[\mathbf{0}, Q^{-1}Q_1Q^{-1}].$$

REFERENCES

[1] T. Adalı and M. K. Sönmez, "Channel equalization with perceptrons: an information theoretic approach," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Adelaide, Australia), April 1994, vol 3, pp. 297-300.

[2] T. Adalı, M. K. Sönmez, and K. Patel, "On the dynamics of the LRE Algorithm: A distribution learning approach to adaptive equalization," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing* (Detroit, Michigan), May 1995, pp. 929-932.

[3] B. M. Brown, "Martingale central limit theorems," *Ann. Math. Statist.*, vol. 44, pp. 59-66, 1971.

[4] D.R. Cox, "Partial likelihood," *Biometrika*, vol. 62, pp. 69-72, 1975.

[5] G. J. Gibson, S. Siu, and C. F. N. Cowan, "The application of nonlinear structures to the reconstruction of binary signals," in *IEEE Trans. Signal Processing*, pp. 1877-1884, vol. 39, no. 8, Aug. 1991.

[6] G. Kechriotis, E. Zervas, and E. S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalization," *IEEE Trans. on Neural Networks,* vol. 5, no. 2, pp. 267-278, March 1994.

[7] B. Kedem, "Time series analysis by higher order crossings", IEEE press, New York, N.Y., 1994.

[8] L. Kullback, and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics 22*, pp. 79-86, 1951.

[9] M. Meyer and G. Pfeiffer,"Multilayer perceptron based decision feedback equalisers for channels with intersymbol interference, "*IEE Proceedings,* Vol. 140, No.6, pp. 420-424, 1993.

[10] H. White, "Learning in artificial neural networks: A statistical perspective," *Neural Computation,* vol. 1, pp. 425-464, 1989.

[11] B.S. Wittner and J.S. Denker, "Strategies for teaching layered networks classification tasks," *Neural Info. Proc. Systems*, (Denver, CO), p.850-859, 1988.

[12] W. H. Wong,"Theory of partial likelihood," *Ann. Statist.,* 14, pp. 88-123, 1986.

CONSTRUCTIVE NEURAL NETWORK DESIGN FOR THE SOLUTION OF TWO-STATE CLASSIFICATION PROBLEMS WITH APPLICATION TO CHANNEL EQUALIZATION

Catherine Z. W. Hassell Sweatman
Department of Electrical Engineering, University of Edinburgh,
Mayfield Rd, Edinburgh EH9 3JZ, U. K.
Supported by EPSRC Grant GR/J34248

Gavin J. Gibson Bernard Mulgrew

ABSTRACT

We describe a deterministic algorithm for designing a MLP for the solution of a two-state classification problem. The Slab Algorithm was motivated by the problem of reconstructing digital signals which have been passed through a real linear dispersive channel of finite impulse response and corrupted with additive noise. Our algorithm is designed to separate two finite disjoint sets of points by constructing a MLP with one hidden layer and a single output node. In the linearly separable case, no hidden layer is constructed. The parameters of the network are identified by standard linear programming techniques. The performance of the channel equalizer constructed by the Slab Algorithm is compared with that of a Bayesian optimal equalizer.

1 INTRODUCTION

We describe a deterministic algorithm for designing a multi-layer perceptron (MLP) [7] for the solution of a two-state classification problem. This algorithm is illustrated in the context of channel equalization [2], but is more generally applicable.

We consider MLPs constructed from McCulloch-Pitts units [3]. A McCulloch-Pitts unit or node (see Figure 1) accepts a real-valued input $\underline{y} \in \mathbb{R}^m$ and calculates as its output the quantity $f(\underline{y}^T \underline{w} - \theta)$, where f is the Heaviside step function and $\underline{w} \in \mathbb{R}^m$ and $\theta \in \mathbb{R}$ are the weight vector and threshold of the node, respectively. Attention is restricted to MLPs with one hidden layer of nodes and a single output node (see [1] and Figure 1).

0-7803-2739-X/95 $4.00 © 1995 IEEE

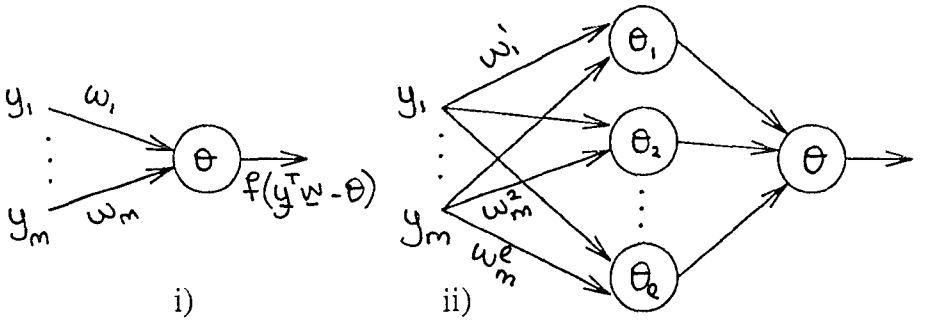


Figure 1: i) A Single Node ii) MLP with One Hidden Layer

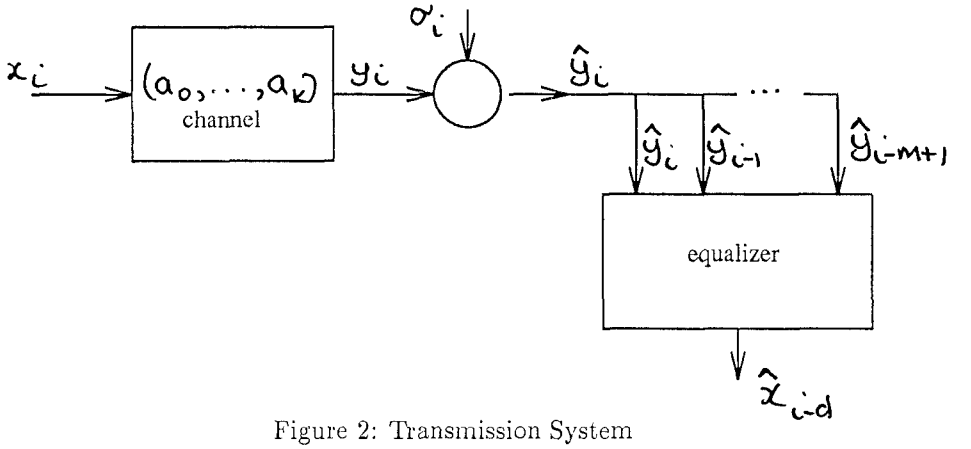


Figure 2: Transmission System

2 CHANNEL EQUALIZATION

The algorithm presented was motivated by the problem of reconstructing digital signals which have been passed through a dispersive channel and corrupted with additive noise as depicted in Figure 2. Explicitly, a random sequence $\{x_i\}$, $x_i \in \{-1, 1\}$, is passed through a real linear dispersive channel of finite impulse response with response function $a(z) = a_0 + a_1 z^{-1} + ... + a_k z^{-k}$, where the coefficients a_j are real, $0 \leq j \leq k$, and a_0 and a_k are non-zero; producing a sequence of outputs, $\{y_i\}$, where $y_i = \Sigma_{j=0}^{k} a_j x_{i-j}$. A term, σ_i, which represents additive noise, is then added to each y_i to produce an observation sequence $\{\hat{y}_i\}$. The task of the equalizer is to use the information represented by the observed channel outputs $\hat{y}_i$, $\hat{y}_{i-1}$, ..., $\hat{y}_{i-m+1}$, to produce an estimate of the input symbol x_{i-d}, where the integer $d \geq 0$ is the *delay* and $m \geq d$. The integer m is the *order* of the equalizer. The input samples x_i are chosen from $\{-1, 1\}$ with equal probability and are assumed to be independent of one another.

In the absence of noise the problem is to separate two sets of points in $\mathbb{R}^m$. These sets of points are the images of the sets

$X_1 = \{(x_0, \ldots, x_{k+m-1})^T : x_i \in \{-1, 1\},\ 0 \leq i \leq k+m-1,\ x_d = 1\}$ and

$X_{-1} = \{(x_0, \ldots, x_{k+m-1})^T : x_i \in \{-1, 1\},\ 0 \leq i \leq k+m-1,\ x_d = -1\}$

under the linear transformation represented by the $m \times (m+k)$ matrix

$$A = \begin{pmatrix} a_0 & a_1 & a_2 & \ldots & a_k & 0 & 0 & \ldots & 0 \\ 0 & a_0 & a_1 & \ldots & a_{k-1} & a_k & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & \ldots & \ldots & \ldots & 0 & a_0 & a_1 & \ldots & a_k \end{pmatrix}.$$

Let $P_{(m,d)}(1) = AX_1$ and $P_{(m,d)}(-1) = AX_{-1}$, as defined in [2].

When noise is present, the observed channel outputs $\underline{\hat{y}} = (\hat{y}_i, \ldots, \hat{y}_{i-m+1})^T$ represent elements of $P_{(m,d)}(\pm 1)$ which are corrupted in each component independently by noise. For low values of the noise variance, each $\underline{\hat{y}}$ is very close to an element of $P_{(m,d)}(1)$ or $P_{(m,d)}(-1)$. The equalizer must represent some function $g : \mathbb{R}^m \rightarrow \{-1, 1\}$ such that $g(A(x_i, x_{i-1}, \ldots, x_{i-(k+m-1)})^T) = x_{i-d}$.

Previous attempts to use the MLP as a channel equalizer are discussed by Gibson *et al.* [2]. In general, $P_{(m,d)}(1)$ and $P_{(m,d)}(-1)$ cannot be separated by a single hyperplane, that is, are not linearly separable, and so a linear transversal equalizer (represented by a single node) will fail to separate them. It has been shown that for $d = 0$, the sets will be linearly separable, for all sufficiently large m, if and only if the channel is minimum phase; that is, if all the roots of $z^k a(z)$ lie strictly within the unit circle in the complex plane [2]. Even if the two sets are linearly separable, the optimal decision boundary is generally non-linear.

Motivated by these difficulties, the MLP was considered. Simulations have shown [2] that a MLP with two hidden layers trained by back propagation [5] can approximate the decision boundary of an optimal equalizer [2] better than can a linear transversal equalizer. In this study, the architecture was chosen by experiment. The training was slow and the decision boundaries obtained were sub-optimal in general. The convergence was often too slow to be of use in the case of time-dependent response function coefficients. In this paper we adopt an alternative approach.

There are basically two strategies for channel equalization; either estimating the channel characteristics from the data and constructing the data equalizer, or estimating the equalizer directly from the data. The former strategy is usually preferred in the case of time-dependent channel characteristics, or when speed is required, as fewer parameters need to be estimated. Methods for estimating the channel are described and assessed by McLaughlin [4].

These observations led us to develop and propose a new algorithm we call the Slab Algorithm to construct a solution in the form of a MLP with one hidden layer and a single output node, assuming knowledge of the channel characteristics.

3 THE SLAB ALGORITHM

Let U and V be the cells of a partition of a finite set of points in $\mathbb{R}^m$. Our aim is to construct a classifier that separates U from V. Let $U_0 = U$ and $V_0 = V$. Iterate the following steps, $p \geq 1$.

1. Find a slab
$$S_p = \{\underline{y} \in \mathbb{R}^m \ : \ a_p \leq \underline{y}^T \underline{w}^p \leq b_p\}$$
where a_p, $b_p \in \mathbb{R}$, $a_p \leq b_p$, $\underline{w}^p \in \mathbb{R}^m$ and $\underline{w}^p \neq \underline{0}$, such that
$$U_{p-1} \subseteq \{\underline{y} \in \mathbb{R}^m \ : \ a_p \leq \underline{y}^T \underline{w}^p\}, \ V_{p-1} \subseteq \{\underline{y} \in \mathbb{R}^m \ : \ \underline{y}^T \underline{w}^p \leq b_p\}$$
and the width of the slab is minimal (in the sense defined below).

2. Let $U_p = U_{p-1} \cap S_p$ and let $V_p = V_{p-1} \cap S_p$. If $U_p = V_p = \emptyset$ then stop. Otherwise, return to step 1, increasing p by one.

We use standard linear programming techniques to identify the slabs. By 'find a slab of minimal width' we mean if U_{p-1} and V_{p-1} are linearly separable, then S_p must be a separating hyperplane such that $(U_{p-1} \cup V_{p-1}) \cap S_p = \emptyset$; otherwise, attempt to minimize the number of elements in $(U_{p-1} \cup V_{p-1}) \cap S_p$.

Knowledge of the slabs S_p, $p \geq 1$, enables a separating network to be specified. If U and V are linearly separable, the Slab Algorithm yields a single separating hyperplane, S_1. A network comprising a single node with weight vector $\underline{w}^1$ and threshold $a_1 = b_1$ will separate U from V.

Otherwise, let $S_1, \ldots, S_{q+1}$, $q \geq 1$, be the slabs calculated. For p, $1 \leq p \leq q$, S_p specifies a pair of hidden layer nodes. The hidden layer nodes are assigned weights $w_1^p, \ldots, w_m^p$ with threshold a_p and weights $-w_1^p, \ldots, -w_m^p$ with threshold $-b_p$. The weights for the output node in the second layer which multiply the outputs from the two nodes corresponding to S_p are $1/2^p$ and $-1/2^p$, respectively. These two nodes combine to contribute $1/2^p$ to the output node from points in $U_{p-1} \backslash S_p$, $-1/2^p$ to the output node from points in $V_{p-1} \backslash S_p$ and zero from points in $(U_{p-1} \cup V_{p-1}) \cap S_p$, $1 \leq p \leq q$. The last node in the hidden layer is specified by the separating hyperplane

$$S_{q+1} = \{\underline{y} \in \mathbb{R}^m \ : \ \underline{y}^T \underline{w}^{q+1} = a_{q+1} = b_{q+1}\} \text{ such that}$$

$U_q \subseteq \{\underline{y} \in \mathbb{R}^m \ : \ \underline{y}^T \underline{w}^{q+1} > b_{q+1}\}$ and $V_q \subseteq \{\underline{y} \in \mathbb{R}^m \ : \ \underline{y}^T \underline{w}^{q+1} < b_{q+1}\}$. The last hidden layer node has weights $w_1^{q+1}, \ldots, w_m^{q+1}$ with threshold b_{q+1}. The corresponding weight for the output node in the second layer is $1/2^{(q+1)}$. This output node is assigned the threshold $1/2^{(q+2)}$. We make use of the geometric series $1/2+1/4+1/8+\ldots$ Note that $1/2 \pm 1/4 \pm 1/8 \pm \ldots \pm 1/2^{q+1} > 1/2^{(q+2)}$. Hence, when $q \geq 1$ and $1 \leq p \leq q$, points in $U_{p-1} \backslash S_p$ and $V_{p-1} \backslash S_p$ are well classified by the pair of nodes corresponding to S_p and will not be misclassified by the addition of further hidden layer nodes.

The number of nodes generated by this process is guaranteed finite because we ensure that at least one point in $U_{p-1} \cup V_{p-1}$ is excluded from the slab S_p at each step.

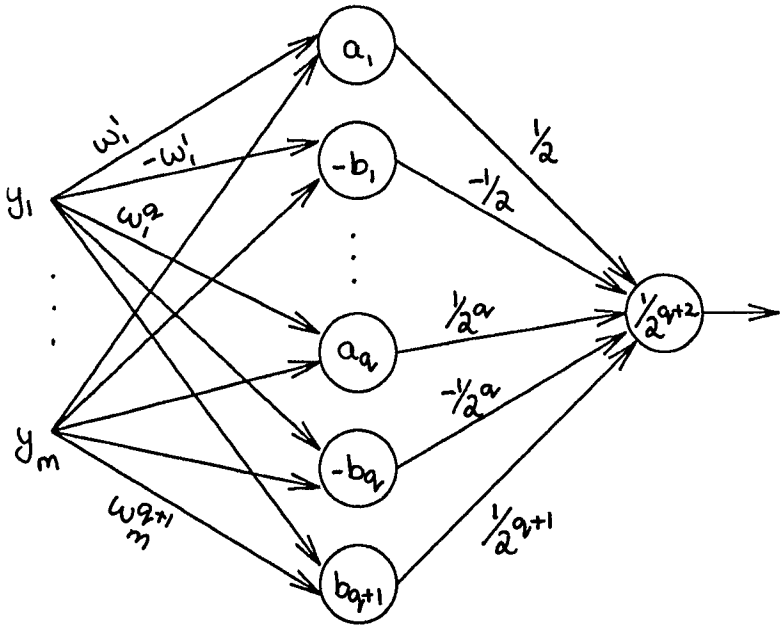


Figure 3: MLP Constructed by the Slab Algorithm

It is possible to replace the p-th pair of hidden layer nodes, $1 \leq p \leq q$, when $q \geq 1$, by a single hidden layer node with two thresholds, a_p and b_p and node activation function $f : \mathbb{R} \to \{-1, 0, 1\}$ defined by $f(x) = -1$ if $x < a_p$, $f(x) = 0$ if $a_p \leq x \leq b_p$ and $f(x) = 1$ if $x > b_p$. The outputs from these nodes (inputs to the output node) are multiplied by the weights $1/2, 1/4, 1/8, \ldots, 1/2^q$, respectively.

The slabs S_p, $1 \leq p \leq q+1$, $q \geq 0$, are determined by linear programming as follows.

1. We first try to separate U_{p-1} from V_{p-1} with a hyperplane

$$S_p = \{\underline{y} \in \mathbb{R}^m \ : \ a_p = \underline{y}^T \underline{w}^p = b_p\}$$

such that

$$U_{p-1} \subseteq \{\underline{y} \in \mathbb{R}^m \ : \ \underline{y}^T \underline{w}^p > a_p\} \text{ and } V_{p-1} \subseteq \{\underline{y} \in \mathbb{R}^m \ : \ \underline{y}^T \underline{w}^p < a_p\}.$$

Linear programming constraints are expressed in the form of linear inequalities or equations, but not as strict inequalities. Hence we seek a separating slab of non-zero thickness

$$S'_p = \{\underline{y} \in \mathbb{R}^m \ : \ c_p \leq \underline{y}^T \underline{w}^p \leq d_p\}$$

where c_p, $d_p \in \mathbb{R}$, $c_p + 1 = d_p$, $\underline{w}^p \in \mathbb{R}^m$, $\underline{w}^p \neq \underline{0}$, such that

$$U_{p-1} \subseteq \{\underline{y} \in \mathbb{R}^m \ : \ \underline{y}^T \underline{w}^p \geq d_p\} \text{ and } V_{p-1} \subseteq \{\underline{y} \in \mathbb{R}^m \ : \ \underline{y}^T \underline{w}^p \leq c_p\}.$$

The constraints which must be satisfied are

(a) $\underline{u}^T \underline{w}^p \geq d_p$ if $\underline{u} \in U_{p-1}$

(b) $\underline{v}^T \underline{w}^p \leq c_p$ if $\underline{v} \in V_{p-1}$ and

(c) $0 \leq d_p - c_p \leq 1$

while minimizing $c_p - d_p$. Begin by setting $\underline{w}^p = \underline{0}$ and $c_p = d_p = 0$. If U_{p-1} and V_{p-1} are linearly separable then, by linear programming, we can find a separating slab S'_p such that $c_p + 1 = d_p$ and $\underline{w}^p \neq \underline{0}$. Set $S_p = \{\underline{y} \in \mathbb{R}^m : \underline{y}^T \underline{w}^p = 1/2 \times (c_p + d_p)\}$. No more slabs are required.

2. Failure to find a suitable $\underline{w}^p$, c_p and d_p as described above indicates that U_{p-1} and V_{p-1} are not linearly separable. In this case, try to find a slab

$$S'_p = \{\underline{y} \in \mathbb{R}^m : a'_p \leq \underline{y}^T \underline{w}^p \leq b'_p\}$$

where $a'_p, b'_p \in \mathbb{R}$, $a'_p + 1 = b'_p$, $\underline{w}^p \in \mathbb{R}^m$, $\underline{w}^p \neq \underline{0}$,

$$U_{p-1} \subseteq \{\underline{y} \in \mathbb{R}^m : \underline{y}^T \underline{w}^p \geq a'_p\} \text{ and } V_{p-1} \subseteq \{\underline{y} \in \mathbb{R}^m : \underline{y}^T \underline{w}^p \leq b'_p\},$$

containing as few elements as possible in $U_{p-1} \cup V_{p-1}$.

The constraints to satisfy are

(a) $\underline{u}^T \underline{w}^p \geq a'_p$ if $\underline{u} \in U_{p-1}$

(b) $\underline{v}^T \underline{w}^p \leq b'_p$ if $\underline{v} \in V_{p-1}$ and

(c) $b'_p - a'_p = 1$

while minimizing $-\Sigma_{\underline{u} \in U_{p-1}} (\underline{u}^T \underline{w}^p - b'_p) + \Sigma_{\underline{v} \in V_{p-1}} (\underline{v}^T \underline{w}^p - a'_p)$. Begin by setting $\underline{w}^p = \underline{0}$ and $a'_p = b'_p = 0$.

This method yields a suitable slab $S'_p = \{\underline{y} \in \mathbb{R}^m : a'_p \leq \underline{y}^T \underline{w}^p \leq b'_p\}$ whose upper bounding hyperplane $\{\underline{y} \in \mathbb{R}^m : \underline{y}^T \underline{w}^p = b'_p\}$ contains elements in V_{p-1} and whose lower bounding hyperplane $\{\underline{y} \in \mathbb{R}^m : \underline{y}^T \underline{w}^p = a'_p\}$ contains elements in U_{p-1}.

With our noisy channel equalization application in mind, shift the slab boundaries outwards. If $U_{p-1} \backslash S_p \neq \emptyset$, let $b''_p = \min_{\underline{u} \in (U_{p-1} \backslash S_p)} \{\underline{u}^T \underline{w}^p\}$ and let $b_p = 1/2 \times (b'_p + b''_p)$. Else, let $b_p = b'_p + 1/2$. If $V_{p-1} \backslash S_p \neq \emptyset$, let $a''_p = \max_{\underline{v} \in (V_{p-1} \backslash S_p)} \{\underline{v}^T \underline{w}^p\}$ and let $a_p = 1/2 \times (a'_p + a''_p)$. Else, let $a_p = a'_p - 1/2$. Define $S_p = \{\underline{y} \in \mathbb{R}^m : a_p \leq \underline{y}^T \underline{w}^p \leq b_p\}$.

Sometimes, one of a pair of hidden layer nodes may be eliminated. If $V_{p-1} \subset S_p$, replace S_p by a single hyperplane, namely $\{\underline{y} \in \mathbb{R}^m : \underline{y}^T \underline{w}^p = b_p\}$, corresponding to a single hidden layer node with weights $w_1^p, \ldots, w_m^p$ and threshold b_p. The corresponding weight for the output node in the second layer is $1/2^{(p)}$. Similarly, if $U_{p-1} \subset S_p$, replace S_p by the hyperplane $\{\underline{y} \in \mathbb{R}^m : \underline{y}^T \underline{w}^p = a_p\}$, corresponding to a single hidden layer node with

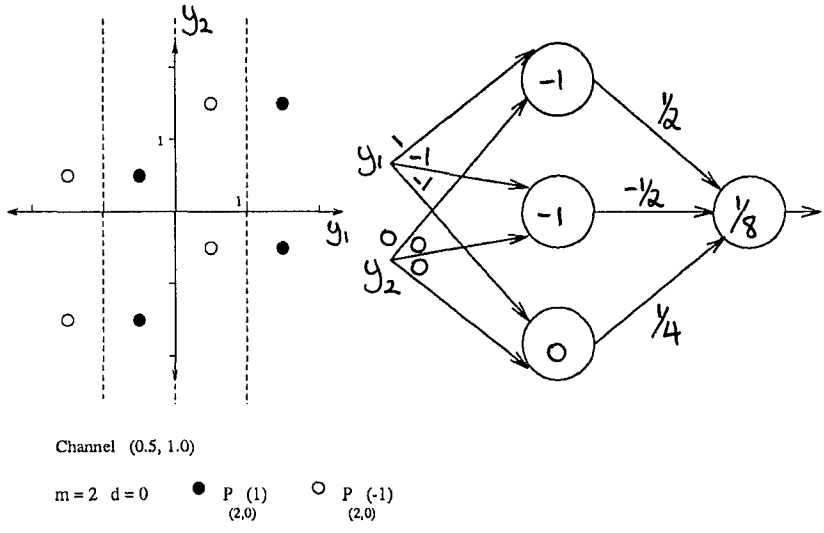


Figure 4: MLP Constructed by the Slab Algorithm for Channel 1

weights $-w_1^p, \ldots, -w_m^p$ and threshold $-a_p$. The corresponding weight for the output node in the second layer is $-1/2^{(p)}$.

In the case of channel equalization, the hyperplanes bounding each slab may be chosen equidistant from the origin and the final separating hyperplane may be chosen to contain the origin. Note that $\underline{y} \in P_{(m,d)}(1)$ if and only if $-\underline{y} \in P_{(m,d)}(-1)$. Taking account of this symmetry reduces the number of parameters to be determined (for a general study see [6]). In a more general case, this symmetry will not be present. We also require $\underline{0} \notin P_{(m,d)}(\pm 1)$.

4 COMPARISONS

Failure rates, called bit error rates (BERs), for the two channels defined by

1. $k = 1$, $m = 2$, $d = 0$, $a_0 = 0.5$, $a_1 = 1.0$ (see Figures 4 and 6) and
2. $k = 2$, $m = 2$, $d = 1$, $a_0 = 0.333$, $a_1 = 0.667$, $a_2 = 1.000$ (see Figures 5 and 7)

were simulated at signal-to-noise ratios between $1dB$ and $20dB$, in steps of $1dB$, using decision boundaries formed by the Slab Algorithm and a Bayesian optimal equalizer (see [2]). It was assumed that the additive noise samples σ_i were chosen randomly and independently from a Gaussian distribution with mean 0 and variance σ^2.

These BERs suggest that the Slab Algorithm may be used to construct a channel equalizer whose performance approximates that of a Bayesian optimal equalizer well, especially at high signal-to-noise ratios.

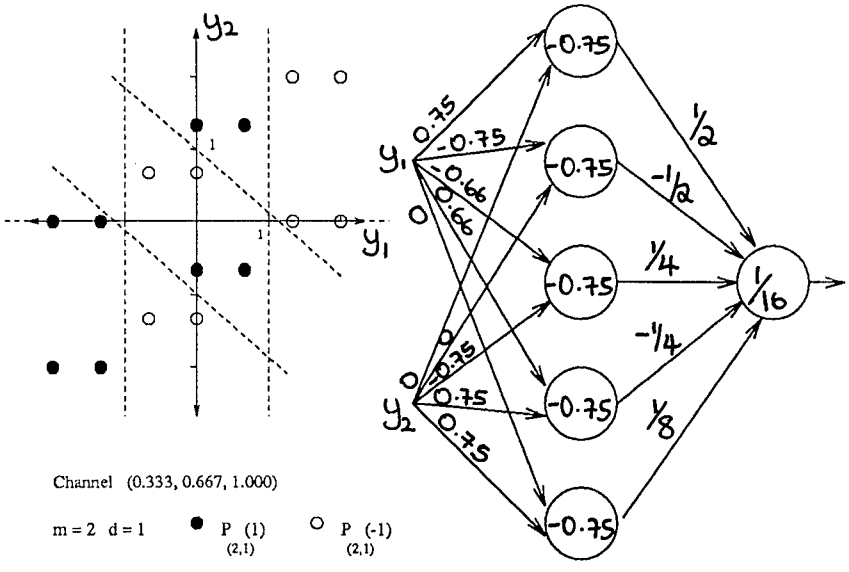


Figure 5: MLP Constructed by the Slab Algorithm for Channel 2

In each case, the sets $P_{(m,d)}(\pm 1)$ are not linearly separable (see Figures 4 and 5). Furthermore, the decision boundary of the Bayesian optimal equalizer is highly non-linear. So, a LTE is totally inadequate, producing a high BER.

5 CONCLUSIONS

The Slab Algorithm will always succeed in separating two finite disjoint sets of points in $\mathbb{R}^m$. It produces a MLP whose design is simple and predictable. If the sets to be separated are linearly separable, it constructs a single separating node. Otherwise, it constructs a MLP with one hidden layer and one output node. In the latter case, the decision boundary is non-linear, formed from hyperplanes.

Our results suggest that the Slab Algorithm may be used to construct a channel equalizer whose performance approximates that of an optimal equalizer well, especially at high signal-to-noise ratios. Our algorithm separates the sets $P_{(m,d)}(\pm 1)$ efficiently, due to the symmetry $\underline{y} \in P_{(m,d)}(1)$ if and only if $-\underline{y} \in P_{(m,d)}(-1)$. Its ability to produce a solution to the equalization problem without the need for long training sequences may mean it is useful in the case of time varying channels.

Our main aim in this paper has been to demonstrate the value of the solutions produced by the Slab Algorithm rather than to optimise its efficiency. Currently, we are investigating alternative methods of identifying the slabs in order to reduce the computational complexity of the algorithm, and to facilitate its implementation in real systems.

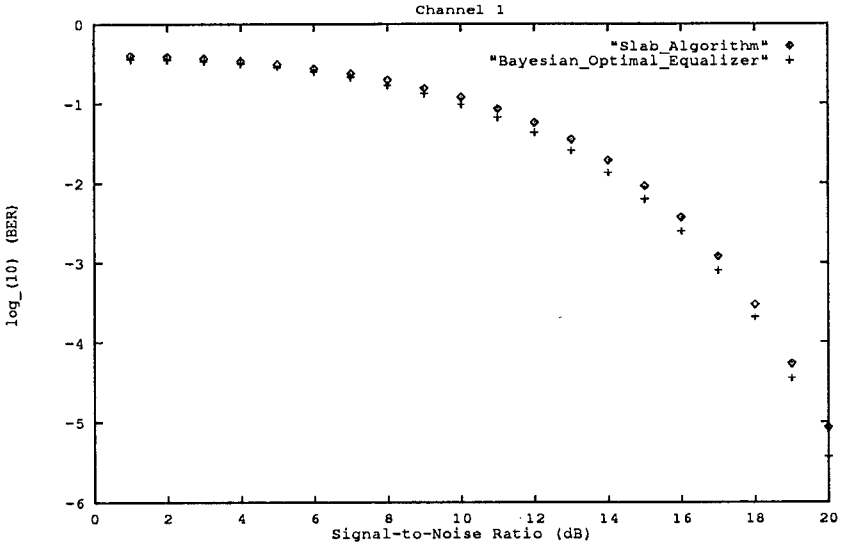


Figure 6: Slab Algorithm and Optimal Equalizer BERs for Channel 1

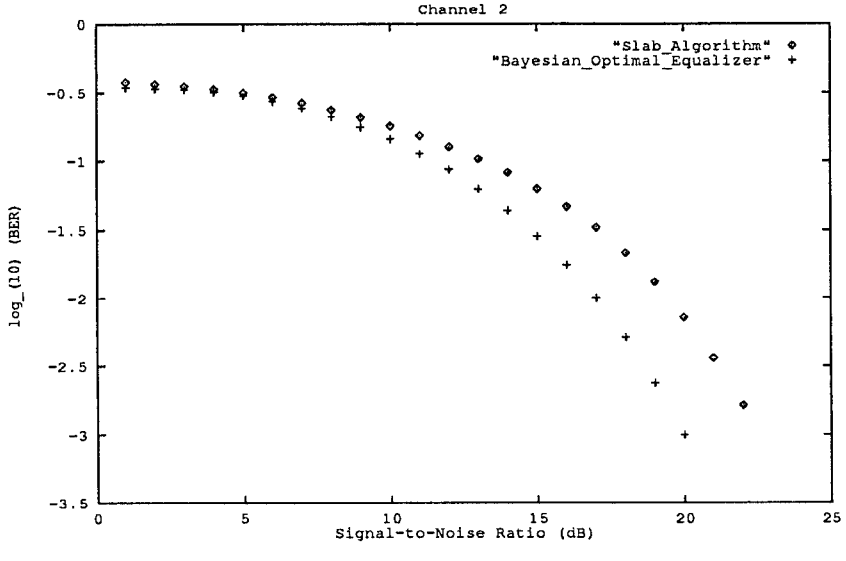


Figure 7: Slab Algorithm and Optimal Equalizer BERs for Channel 2

REFERENCES

[1] Gibson, G. J. (1993). *A Combinatorial Approach To Understanding Perceptron Capabilities* IEEE Transactions on Neural Networks 4(6), 989-992.

[2] Gibson, G. J., Siu, S. and Cowan, C. F. N. (1991). *The Application of Nonlinear Structures to the Reconstruction of Binary Signals* IEEE Transactions on Signal Processing 39(8), 1877-1884.

[3] McCulloch, W. S. and Pitts, W. (1943). *A Logical Calculus of the Ideas Imminent in Nervous Activity* Bulletin of Mathematical Biophysics 5, 115-133.

[4] McLaughlin, S., Mulgrew, B. and Cowan, C. F. N. (1989). *A Performance Study of Three Adaptive Equalisers in the Mobile Communications Environment* IEEE International Conference on Communications, ICC 89, Boston, 193-197.

[5] Rumelhart, D. E., Hinton, G. D. and Williams, R. J. (1986). *Learning Internal Representations by Error Propagation,* Parallel Distributed Processing : Explorations in the Microstructure of Cognition, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA : MIT Press.

[6] Shawe-Taylor, J. (1993). *Symmetries and Discriminability in Feedforward Network Architectures,* IEEE Transactions on Neural Networks 4(5), 816-826.

[7] Widrow, B., Winter, R. and Baxter, R. (1988). *Layered Neural Nets for Pattern Recognition* IEEE Transactions on Acoustics, Speech and Signal Processing 36(7), 1109-1118.

A Parallel Mapping of Backpropagation Algorithm for Mesh Signal Processor

Shoab A. Khan and Vijay K. Madisetti
Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250

Abstract

In this paper, we present a new technique for mapping the backpropagation learning algorithm on a mesh signal processor. The optimal sub-partitioning of computation and communication, and data replication techniques are the key features of our algorithm. Theoretical analysis and simulation results, using the MIT Lincoln Lab simulator, show that our scheme performs better than the other schemes. [1]

1 Introduction

The backpropagation neural networks algorithm (BPN) is one of the most popular neural networks learning algorithms. It has been applied to a number of diverse applications [5]. Because of the high time complexity of the algorithm on a single processor computer, the development of efficient multiprocessor parallel algorithm is very essential. Several architectures, including linear arrays, systolic arrays, meshes, and hypercubes have been explored to implement BPN. Mesh-based machines have been explored by many researchers [6]-[10].

Most of the research in the area of algorithms mapping on an MCC deals with architectures based on simple PEs having small or no local memory. These architectures may contain 100's or 1000's of PEs [11]-[13]. In this case, the processor array needs to be as large as the data structure in order to assign one PE per data element. Therefore, this paradigm poorly performs for a data structure much larger or much smaller than the size of the processor array. Considering the simple architecture of an MCC and the tremendous advancement in the digital signal processor (DSP) technology, where leading off-the-shelf DSPs like C-40 and ADSP21060 also support multiprocessing, an MCC can be easily designed using these DSPs. Our parallel formulation deals with the **M**esh connected computer based on such digital **S**ignal **P**rocessors (MSP), mesh signal processor. In the case of an MSP, the array size is smaller (typically 4×4 or 8×8) due to high computational power and the large local memory of the DSPs. The small size of the processor array and large local memory per PE ensure that this paradigm will perform equally well for large and small data structures.

[1]This research has been supported in part by Analog Devices, Inc, Norwood and Advanced Projects Research Agency (ARPA), under contract 24-6-R8081-0A0, 1994-95.

0-7803-2739-X/95 $4.00 © 1995 IEEE

This paper is organized as follows. Section 2 describes an MSP. Section 3 lists efficient techniques for mapping algorithms on an MSP. Section 4 gives our new parallel formulation. Section 5 shows the software environment for the MIT Lincoln Lab's simulator. Section 6 contains the concluding remarks.

2 Mesh Signal Processor

An example of an MSP is shown in Fig. (1). In an MSP, the most notable features of a DSP are a massive on-chip dual ported static memory and an I/O processor with multi DMA channels. Provision of the dual port memory and I/O processor allows overlap of communication and computation. Both the core processor and the I/O processor have access, via the external port, to the external bus. The DSP includes at least 4 link ports, two bidirectional serial ports, an external port, shared memory, and a broadcast mode to all PEs for multiprocessing support. In an MSP with row broadcasting (MRB), the PEs in the same row are connected to a row bus (RB). In an MSP with row-column broadcasting (MRCB), the PEs in the same row or column are connected to a bus.

3 Our efficient techniques

In this section we describe efficient techniques for designing optimal BPN or other signal processing algorithms on an MSP.

3.1 *Technique 1* Optimal Sub-Partitioning of the Computation

The array size for an MSP is smaller, therefore each PE contains a large amount of data set as opposed to one element per PE. In this situation, execution time can be substantially reduced by subpartitioning computation in each PE. This technique is explained by the following example.

Example 1 In parallel implementations of some of the signal processing algorithms on an MSP it is necessary to find the sum of sub-arrays distributed over rows. Each PE holds a sub-array $\hat{c} \in R^N$ and corresponding elements of $\hat{c}$ are to be added in every row to obtain the resultant array c in each row.

Algorithm I: In a plain mesh, one way to compute the sum is to find the sum of arrays in two adjacent PE and place the result in the left PE and then sum of these two PEs are computed which are now two PE apart, and this pattern is repeated until the final array c is obtained. $P-1$ shifts and $log_2 P$ addition operations are required. Every shift moves N data elements, which is followed by N additions. Therefore

$$t_{com} = N(P-1)t_s, \text{ and } t_p = N(log_2 P)t_a$$

where t_s is the time required to shift one element from one PE to its nearest neighbor, and t_a is the computation time for one addition.

Algorithm II: In an MRB, the RBs are used for reducing the communication time. This is accomplished in two phases [12]. The first phase is like *Algorithm 1*, where the links are used to find the sum of sub-arrays in a block of d adjacent PEs. In second phase of the algorithm the partial sum in the active elements at distance d apart are added using the RBs. This is achieved in $(P/d-1)$ broadcasts, and in every broadcast N data elements are

broadcasted which is followed by N additions. Finally the first PE in each row has the c array for its row. Thus

$$t_{com} = (d-1)Nt_s + \left(\frac{P}{d} - 1\right) Nt_b, \text{ and } t_p = \left(log_2 d + \frac{P}{d} - 1\right) Nt_a$$

were t_b is the time required for shifting one element via bus. The t_{com} will be minimum if $d = \left(\frac{Pt_b}{t_s}\right)^{1/2}$.

Algorithm III **Our Algorithm** An optimal algorithm for the given problem can be designed by sub-partitioning the computations in each PE. Each PE sub-partitions its N elements of $\hat{c}$ into P subgroups of N/P elements each. Here, for simplicity, we assume that P is a multiple of N. In each iteration, every PE shifts only one of the subgroups of N/P elements with starting index $[(i+k) \text{ modulo } P]\frac{N}{P}$, where k is the column location of the PE and $k = 0, 1, \ldots, P-1$, and i is the iteration of the algorithm and $i = 1, 2, \ldots, P-1$. Every PE adds N/P of its corresponding elements into the new N/P elements. This pattern is repeated for $P-1$ iterations. This technique for $N = 4$ and $P = 4$ is shown in Fig. (2). Thus

$$t_{com} = \left(\frac{N}{P}\right)(P-1)t_s, \text{ and } t_p = \left(\frac{N}{P}\right)(P-1)t_a$$

Finally every PE has N/P data elements of the final array c with starting index $k\frac{N}{P}$, Fig. (3) shows that our technique yields the best results without using any broadcasting features of the MSP during its computation. By reversing the communication process, a copy of the resultant array can be stored in every PE of the row.

3.2 *Technique 2* Optimal Partitioning of the Data Set

The large local memory of a DSP in an MSP can be used for further reducing the communication cost of parallel algorithms. A PE may need data stored in other PEs during the execution. The algorithm needs inter-processor communication to acquire this data. This communication can be removed by storing copies of the commonly used data in the corresponding PEs. Further, the data set is partitioned such that the commonly used data is mapped to the PEs sharing the same broadcasting bus. This process eliminates any overhead which may result in the process of replicating data to the PEs. For BPN algorithm the input array is partitioned into P sub-arrays of size N/P each. These P sub-arrays are broadcasted using the CBs. The desired output array is also partitioned into P sub-arrays and broadcasted using the RBs or CBs depending upon the total number of layers in the network. The local memory requirement of the DSP can be reduced by adding a shared memory module to each row of an MRB, and to each row and column of an MRCB [1].

3.3 *Technique 3* Optimal Sub-Partitioning of the Communication

The provision of dual port memory and I/O processor allows overlap of communication and computation. An optimal sub-partitioning of the communication can maximize this overlap. Using this technique, the I/O processor only shifts part of the data required to be moved in each shift operation.

Then, each PE starts its computations on the received part, while the I/O processor shifts the rest of the data. The optimal overlap depends upon the values of t_a and t_s. For instance if $t_a \geq t_s$, for *algorithm III* an optimal sub-partitioning of the communication can be achieved if the I/O processor of each PE only shifts $N/2P$ elements instead of N/P elements. While each PE is adding it's corresponding $N/2P$ elements into the newly arrived elements, the I/O processor shifts the rest of the $N/2P$ elements. When each PE finishes its computation of the first part, the PE starts computing the second part and the I/O processor starts shifting the pre-computed elements from the first part. The communication time in this case is reduced to $\frac{N}{2P}t_s$. This saving in the communication is also shown in Fig. (3).

4 Our Parallel Algorithm

Using techniques developed in section 3, steps in the parallel implementation of BPN algorithm on an MRB/MRCB are as follows:

S0: The first step is for the host to broadcast the input array $\mathbf{x}$ over the MSP using CBs according to the following mapping:

$$x[j] \rightarrow \text{PE}(*, l) \quad \text{for} \quad \begin{array}{l} j = l\frac{N}{P}, \ldots, (l+1)\frac{N}{P} - 1 \\ l = 0, 1, \ldots, P-1 \end{array}$$

as a result, each PE in a row stores a sub-array of size N/P in its local memory, and every PE in the same column has a copy of the sub-array.

S1: The mapping of the desired output array $\mathbf{d}$ depends upon the total number of layers in the network. If the total number of layers is odd the mapping is same as for the input layer given in step S0. In case the total number of layers in the network is even, the $\mathbf{d}$ is broadcasted subject to the following mapping:

$$d[j] \rightarrow \text{PE}(k, *) \quad \text{for} \quad \begin{array}{l} j = l\frac{D}{P}, \ldots, (k+1)\frac{D}{P} - 1 \\ k = 0, 1, \ldots, P-1 \end{array}$$

as a result, every PE in a row has a copy of a sub-array of size D/P.

S2: Initial weights are randomly generated in all DSPs. The mapping for the input layer weights are given below:

$$w_0[i][j] \rightarrow \text{PE}(k, l) \quad \text{for} \quad \begin{array}{l} i = k\frac{N}{P}, \ldots, (k+1)\frac{N}{P} - 1; \\ j = l\frac{M_0}{P}, \ldots, (l+1)\frac{M_0}{P} - 1 \\ k = 0, 1, \ldots, P-1; \\ l = 0, 1, \ldots, P-1 \end{array}$$

The mapping for the weights of any layer is transpose of the mapping for its previous layer. For example the mapping for the first hidden layer weights is transpose of the mapping for the input layer weights and is given by:

$$w_1[i][j] \rightarrow \text{PE}(k, l) \quad \text{for} \quad \begin{array}{l} i = l\frac{N}{P}, \ldots, (l+1)\frac{N}{P} - 1; \\ j = k\frac{M_1}{P}, \ldots, (k+1)\frac{M_1}{P} - 1 \\ k = 0, 1, \ldots, P-1; \\ l = 0, 1, \ldots, P-1 \end{array}$$

S3: Each PE executes Eq. 4 on its input sub-array and weight sub-matrix. This computation results in a *local_net* sub-array in each PE. The corresponding elements of each sub-array are to be added for obtaining the final array *net*. The sum is computed, using *algorithm III* of section 3. As a result hidden layer output array is obtained. The array is distributed over column such that now every slave in a row has a copy of the sub-array. This operation is repeated until the desired output is obtained.

The communication time for one iteration of the algorithm, using computation sub-partitioning and data replication techniques is given by

$$t_{com} = 2\sum_{i=1}^{L-1} \frac{M_i(P-1)}{P} t_s = \frac{2M(P-1)}{P} t_s$$

where M_i is the number of neurons in layer i, M is the total number of neurons in the network, and L is the total number of layers in the network. This communication cost can be substantially reduced by sub-partitioning the communication in each DSP and utilizing I/O processor. Using technique III for computing the array *net* the communication time can be reduced to

$$t_{com} = \frac{M}{P} t_s \tag{1}$$

The overall execution time per iteration per pattern of the BPN algorithm on a single node can be approximated by

$$t_0 = \sum_{i=1}^{L} M_i M_{i-1} t_m \tag{2}$$

where t_m is the time for one multiplication. Using the communication expression in equation (1), the the parallel run-time per iteration per pattern is as follows:

$$t_{\text{MSP}} = \frac{1}{P}\sum_{i=1}^{L} M_i M_{i-1} t_m + \frac{M}{P} t_s \tag{3}$$

Using equations (2) and (3), the expression for speedup, $S_p = \frac{t_0}{t_{\text{MSP}}}$, can be obtained.

With little modification in mapping of the weight sub-matrices, the algorithm can be implemented with the same time complexity on an MRB. The distribution of the input array, the hidden output array, the actual and desired output array and the weights of different layers for a three layers BPN is shown in Fig(4). For training the network the error array is computed by subtracting the actual output from desire output. The array is propagated back through the network, which involves the same computational steps, but in reverse order. The weights are updated using equation (6). The algorithm is terminated when total error is less than a pre-specified number.

4.0.1 The Implementation of Algorithm on ADSP 21060 based MSP

In this section we discuss prototyping the algorithm on a simulator for ADSP-21060 based MSP. The simulator is developed by Ira Gilbert of the MIT Lincoln Labs [2]-[3].

The Processing Element The most notable features of ADSP-21060 are a massive 4-Mbit of on-chip dual ported static memory and an I/O processor with 10 DMA channels [4]. Provision of dual port memory and I/O processor allows overlap of communication and computation. Both the core processor and the I/O processor have access, via the external port, to the external bus. The 21060 includes six 4-bit-wide link ports, two bidirectional serial ports, 48-bit-wide external port, shared memory, and a broadcast mode to all processor for multiprocessing support.

Software Environment We will briefly introduce the software prototyping environment. Two classes of data variables are used in MSP. The *single-valued variable* is restricted to the same value for every slave, and the *multivalued variable* may take on different values in each slave. The *shift* operation is a fundamental operation for inter-processor communication. The macro defined by **Shift**(B, A, x_dim, y_dim, x_shift, y_shift) shifts an $x_dim \times y_dim$ array A, by x_shift in the x direction and y_shift in the y direction; B is the destination array. The BPN algorithm requires to add the corresponding elements of the array *local_net* in each PE of the row. The **Shift** macro is used for shifting these elements from one PE to its neighbor. The code is shown in Fig. (5).

5 Conclusion

Signal processing algorithms including BPN can be efficiently implemented using DSP in a mesh architecture. The paper presented efficient techniques for implementing neural networks and other signal processing algorithm on digital signal processors based SIMD mesh connected computer. Three efficient techniques are given which can reduce the computation and communication time. The first technique reduces computation and communication time by optimally sub-partition the computation in each DSP, the second technique reduces the inter-processor communication by utilizes large local memory of DSP and replicate commonly used data on DSPs sharing the same broadcasting bus. The third technique further reduces the communication cost by sub-partitioning the communication and efficiently utilizing the I/O processor and dual port memory for overlapping communication and computation. The MIT Lincoln Lab simulator does not support DMA programming. We will implement the simulated algorithms on actual MSP, therefore the exact numbers for the saving in communication time using DMA and the speedup will be given in the future work.

Appendix A: Backpropagation Learning Algorithm

The algorithm trains a given feedforward neural network for a given set of learning patterns. The update step at neuron k of layer j is defined as

$$\begin{aligned} x_j[k] &= f_k(net_k) \\ net_k &= \sum_{i=1}^{N} w_{j-1}[k][i] x_{j-1}[i] \end{aligned} \tag{4}$$

where N is the number of nodes on the previous level $(j-1)$, $x_j[k]$ is the output of node j of layer k, $x_{j-1}[i]$ is the value of node i on the previous level $j-1$ and $w_j \quad [k][i]$ is the weight on the connection from node i of layer $j-1$ to node k of layer j, and $f()$ is a nonlinear function. The activation values obtained from Eq. (4) are communicated to the next layer, which repeats the same operation and the actual output is computed. In phase two the error is

propagated back to the network and weights are modified. Let $y[j]$ and $d[j]$ be respectively the actual output and the desired output at neuron k of the output layer. The weight adjustment is based on the following relation:

$$w_j[k][i](t+1) = w_j[k][i](t) + \eta\delta_{j+1}[k]x_j[i] \tag{5}$$

where η is a positive constant referred to as a learning rate and δ is the error signal. For the hidden-to-output neuron level the error signal is given by:

$$\delta_o[k] = y[k](1 - y[k])(d[k] - y[k]) \tag{6}$$

For the hidden layers neuron the error signal is computed using the following relation:

$$\delta_j[k] = x_j[k][1 - x_j[k]) \sum_m \delta_{j+1}[m]w_j[m][k] \tag{7}$$

where m is over all neurons in the layer above the layer of neuron j.

References

[1] S. Khan and V. Madisetti, *Signal and Image Processing on a Mesh-Based SIMD Supercomputer*, first draft.

[2] I. Gilbert, "The Monolithic Synchronous Processor," *Lincoln Laboratory Journal*, Vol. 1, No. 1, 1988.

[3] I. Gilbert and S. Woodworth, *MSP Software Reference*, MIT Lincoln Lab, 1994.

[4] *ADSP 21060 Users Manual*, Analog Devices Inc., Norwood Massachusetts, MA 02062.

[5] F.A. James and M.S. David, *Neural Networks: Algorithms, Applications, and Programming Techniques*, New York: Addison-Wesley, pp. 110-114.

[6] K. W. Przytula, V. K. Prasanna, and W. M. Lin, "Parallel implementation of neural networks," *J. of VLSI Signal Processing*, vol 4, pp 111-122, 1992.

[7] W. M. Lin, V. K. Prasanna, and K. W. Przytula, "Algorithmic mapping of neural network models onto parallel SIMD machines," *IEEE Trans. on Computers*, vol. 40, no. 12, pp. 1390-1401, 1991.

[8] W. Allen and A. Saha, "Parallel neural network simulation using backpropagation for the es-kit environment," in *Proc. 1989 Conf. Hypercubes, Concurrent Computers and Applications*, 1989, pp. 1097-1102.

[9] F. Baiardi, R. Mussard, R. Serr, and G. Valastro, "Feedforward neural networks on message passing parallel computers,"

[10] S. Y. Kung and J. N. Hwang, "A unified systolic architecture for artificial neural networks," *J. Parallel Distrib. Computing*, vol. 6, pp. 358-387, 1989.

[11] Y. C. Chen, W. T. Chen, G. H. Chen, and J. P. Sheu, "Designing efficient parallel algorithms on mesh-connected computers with multiple broadcasting," *IEEE Trans. Parallel Distributed Syst.*, vol. 1, no. 2 pp. 241-245, Apr. 1990.

[12] C. M. Huang, "Efficient image correlation on MCC with multiple global buses architecture," *Proc. Natl. Sci. Counc.*, Vol. 16, No. 3, 1992. pp. 265-269

[13] V. K. Prasanna Kumar and C. S. Raghavendra, "Array processor with multiple broadcasting," *J. Distribut. Comput.*, vol. 2, pp. 173-190, 1987.

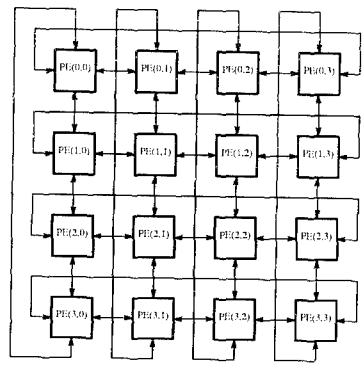


Figure 1: Mesh Signal Processor

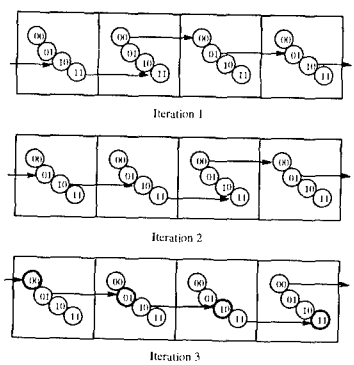


Figure 2: Optimal sub-partitioning of computation for $P = 4$ and $N = 4$ for one row of MSP

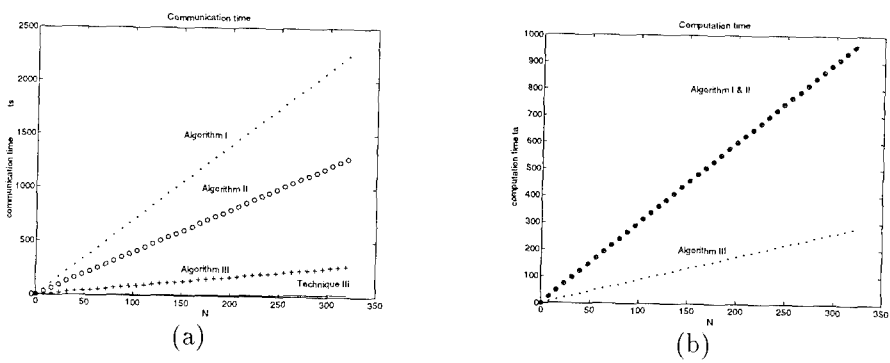


Figure 3: Summation of elements spread over row of 8×8 MSP (a) Communication time (b) Computation time

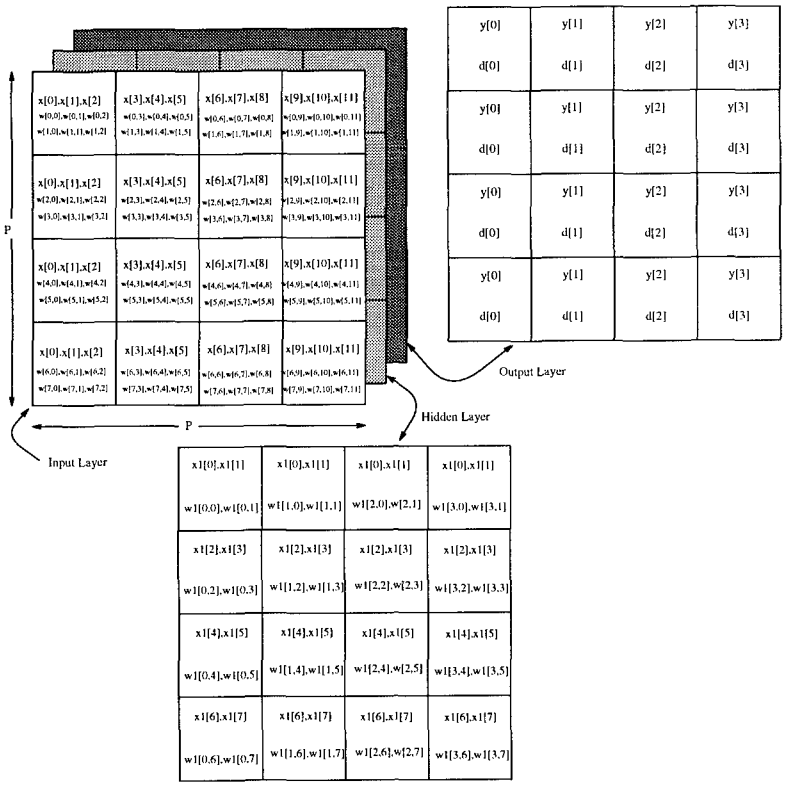


Figure 4: The distribution of neural networks data on MSP.

Function: row_sum

Purpose: This function computes the global sum of the data spread row wise.

Communications: P-1 shifts of the array *local_net* of N/P elements.

```
row_sum(multi int index)
{
int          i, j;
   multi int  shift_net[NbyP], *local_net_ptr;
         for(i=0; i<P - 1; i++)
         {
                local_net_ptr = &local_net[index];
                Shift(shift_net, local_net_ptr, NbyP ,1 , 1, 0);
                for(j=0; j<NbyP; j++)
                {
                    *local_net_ptr  += shift_net[j];
                    local_net_ptr++;
                }
                index = (index - NbyP) % N;
          }
}
```

Figure 5: The Slave DSP program for summing the elements spread over rows of an MSP.

DIGITAL NEUROIMPLEMENTATIONS OF VISUAL MOTION-TRACKING SYSTEMS

Anna Maria Colla°, Luca Trogu*, and Rodolfo Zunino*
° Elsag Bailey - Via Puccini 2 - 16152 Genova - Italy
* DIBE - Genoa University - V.all'Opera Pia 11a - 16145 Genova - Italy

Abstract — **The paper describes the implementation of neural systems for visual motion tracking on a digital neurocomputing platform, i.e., the NLX420 Neural Processing Slice. The chip architecture and the problem model considered greatly facilitate the implementation task, which involves the fulfilment of critical real-time constraints. Experimental results confirm the approach validity in terms of both speed and prediction accuracy; training adjustment techniques are also examined.**

I. RESEARCH BASELINE

The success of neural models in complex applications is mainly based on their ability to face nonlinear problems and on their example-driven training flexibility. In real applications, suitable hardware platforms are often required to fully exploit the benefits of neural systems. Therefore, many researchers recently addressed the development of hardware circuitry for neurocomputing [1,2], but no paradigm (analog or digital) has so far achieved a predominant position in the field. This fact has brought about some odd situations, in which the effectiveness of neural methods has been proved by simulation, whereas a final hardware implementation lacks suitable supporting machinery. This is especially true for those application domains requiring high speed performances to fulfil strict real-time constraints.

In this context, the present paper addresses the problem of hardware implementation of a fully neural system for visual motion tracking. Previous research [3,4] demonstrated the system validity and showed that a simple neural architecture can attain effective tracking ability at a very limited computational cost, thus meeting real-time performance requirements. The proposed method uses a feedforward, back-propagation trained network to infer motion quantities from time-consecutive image frames [3]. The straightforward architecture and its limited computational complexity allow one to privilege simplicity and modularity in the design phase.

For the hardware implementation, we chose a commercial digital chip (NLX420 Neural Processing Slice) suitably developed for neural processing support [5]. The most interesting aspect of such a platform is its "open"

0-7803-2739-X/95 $4.00 © 1995 IEEE

internal architecture, which makes it possible to inspect and adjust all components of the neural model. As a result, a designer can fit the implementation setup to the specific task considered. In addition, the digital representation of numerical quantities facilitates the control of the system accuracy.

The main result of the present research is a method for porting simulated Multi-Layer Perceptrons (MLPs) onto a digital supporting machinery. The methodology involves a simple modification to the back-propagation (BP) [6] training procedure for the network to take into account the finite precision of the supporting architecture. The approach validity was confirmed experimentally by the satisfactory results obtained in the visual application domain. Accuracy has been evaluated by comparing floating-point simulations with actual chip results; high computation speed allows the system integration into technical, target-application setups.

II. DIGITAL NEUROIMPLEMENTATION

II.1 - The NLX420 Neural Processing Slice

The NeuraLogix NLX420 Neural Processing Slice (NPS) [5] is an 84-pin chip designed as a digital building block for Neural Networks, and working with 16-, 8-, 4- or 1-bit precision. In addition to timing and control signals, the chip provides 16 pins for inputs, 16 for outputs and 16 for synaptic weights, which are stored in an external RAM. As shown by the schematic diagram in Fig.1, 16 Processing Elements (PEs) compute the inner products of the synaptic inputs, $\mathbf{x}_j$, and weights, $\mathbf{w}_j$. If the number of network inputs is larger than 16, the process is iterated, and each PE has a 32-bit register where intermediate results, r_j, accumulate:

$$r_j = \mathbf{x}_j \bullet \mathbf{w}_j \; ; \quad \text{j=1,...,16} \tag{1}$$

When all neuron inputs have been processed, the dot products enter the transfer function block, where an arbitrary nonlinear function (e.g., sigmoid, hyperbolic tangent, step) is approximated:

$$o_j = f\left(r_j\right) \; ; \quad \text{j=1,...,16} \tag{2}$$

The results are stored in 16 output registers. The nonlinear function $f()$ can be implemented by a number of segments dependent on the precision mode selected. With 16-bit precision, the range between -32768 and 32767 is divided into 16 contiguous user-specified regions; each region is characterized by a specific slope and a specific offset. Among the other features, the chip supports both synchronous and asynchronous timings. A loopback connection and a standard 8-bit bus interface are also available. The NLX420 NPS operates as a sort of neural coprocessor that must be driven by a host CPU. It is characterized by low cost and good performance, though the maximum clock rate is 20 Mhz, as it exploits the parallelism

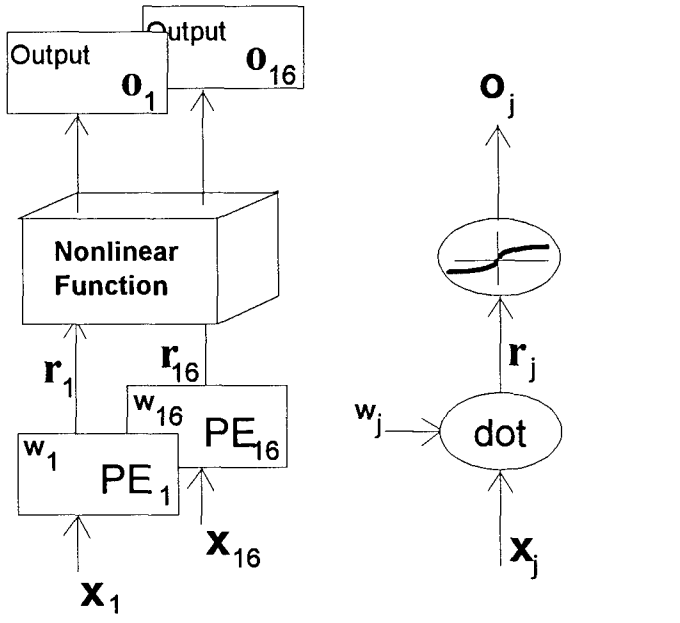


Fig.1 - Functional schema of the NLX420

implicit in neural algorithms. On the other hand, the use of integer arithmetic and the approximate transfer function, if not suitably tuned, may affect the precision of calculations.

The results described in this paper have been obtained by using a board equipped with 4 NPS and as many 32K X 16-bit Weight RAM; in this configuration, synaptic inputs and control signals are common to the four processors, thus a layer of 64 neurons is obtained.

II.2- Implementation of MLPs on the NLX 420 NPS

The use of the described NPS to implement a Multi-Layer Perceptron follows a straightforward approach. After setting the Weight RAM pointer to select the desired bank of neurons, the host CPU submits the inputs to the NPS; when the outputs are ready, they are buffered. If a layer has more than 64 neurons, this operation is repeated until the whole layer is processed; the outputs may then be used as inputs to the next layer.

As the NPS does not provide hardware support for network training, weights are imported from software simulations of the back-propagation algorithm. This requires floating-point-into-integer conversion; the 16-bit mode should always be chosen to preserve an acceptable precision. Network inputs are typically normalized to either the interval (0,1) or (1,1), whereas weight values may extend to some units. In this situation, a good multiplicative factor for both quantities has been found to be 2^{12}; smaller factors would result in unacceptable losses in accuracy, and larger ones might lead to an overflow. According to this choice, the value 2^{24} in an

accumulator, r_j, corresponds to a floating-point value 1.0. As the transfer function supports only 16-bit input values, the inner product can be fetched by the external host CPU, right-shifted by 12 positions, and eventually supplied to the transfer function. This method works well, although processing time increases because the shift operation is not supported by the hardware.

II.3 - Techniques for Improving Training

The adjustment of the BP algorithm to take into account the actual hardware-supported transfer function has yielded interesting results. The basic idea is to modify the training process so as to take into account the finite-precision representation of the nonlinearity of the hardware-supported neurons. A straightforward approach is to replace the ideal sigmoidal nonlinearity with its on-chip piecewise approximation for the duration of the whole training process. Such a procedure, however, proves inadequate mainly because the quantization of the cost function often traps the gradient descent in "spurious" local minima, especially at early training steps. According to a more effective procedure, training starts by using an ideal nonlinearity (either a sigmoid or a hyperbolic tangent) to best follow the shape of the cost function; when the gradient descent requires a fine adjustment, the training procedure switches to the piecewise transfer function (and its derivatives). This reduces the quantization effects brought about by the final porting phase. The switching from the ideal nonlinearity, $f(x)$, to its quantized version, $g(x)$, is ruled by a threshold-based criterion:

1) back-propagation training proceeds using $f(x)$ up to a satisfactory solution (e.g., for which the output mean square error is below a given threshold);
2) the nonlinear function switches to $g(x)$; if the output MSE for the new function exceeds the cost threshold, BP training restarts until threshold-based convergence is attained again.

However, other, more refined schemata for the switching process can be envisioned. A general solution involves a smoother iterative process; at each iteration, the nonlinear function will be given by:

$$h^{(i)} = \alpha_i \cdot f(x) + (1-\alpha_i) \cdot g(x) \tag{3}$$

where $i=1,\ldots,N$; $\alpha_i < \alpha_{i+1}\ \forall i$; $\alpha_1 = 1$; $\alpha_N = 0$. The parameter α modulates the transition between the analog and the hardware-supported activation functions. Crucial parameters characterizing this algorithm include: the number of steps, N; the actual sequence of decreasing factors, α_i; finally, the mechanism ruling the function switching at each step.

Some elementary considerations may help choose suitable settings. Fast reduction rates of α_i increase the likelihood for gradient descent to be trapped in a local minimum; moreover, each switching brings about some perturbation in the cost function and leads to an immediate, apparent deterioration of evaluated output error. For these reasons, an initial value $\alpha_1 = 1$ (involving a truly analog simulation) should be maintained until the

optimization process has approached a final solution. Then, once started the sequence of decreasing α values (3), for each fixed α_i, the optimization process should be allowed to reach proper termination. The number of steps, N, can be increased to attenuate the negative effects of each switching; in fact, very fine discretizations of the transition parameter (virtually simulating a continuous variation of α) did not yield remarkable improvements. The values of α_i can be selected accordingly: among others, the most natural choice seems to be:

$$\alpha_i = \frac{i-1}{N-1} \tag{4}$$

For evaluation purposes, this technique has been applied to the well-known nested-spirals benchmark. In this strongly nonlinear domain, we considered the NLX 420 implementation of a reference MLP structure [9] having topology 2-5-5-5-1; the training set included 194 patterns. In a first experiment, the weights resulting from software (ideal) simulations were directly ported onto the hardware platform; this allowed the network to classify correctly only 93.3% of the training samples; these errors were due to the actual use of a piecewise transfer function. Then we used the training-adjustment technique described in (3) and (4), and set N= 6 (i.e., 6 different values for α). At each i-th step (i=1,...,6) we enabled switching to the next step only after achieving a 0.0% error; when porting the obtained weights onto the NPS board, the NLX 420 implementation recognized correctly the whole training set. Figure 2 presents a comparison between an ideal sigmoid simulation and a piecewise neuroimplementation trained with the modified technique described above. Experimental evidence indicates a satisfactory performance of the overall method, especially when considering that results display generalization performance.

Fig. 2 - Generalization results between SW and HW
Software simulation NLX neuroimplementation

III. NEURAL TARGET TRACKING

The basic approach to neural target tracking is outlined in [3] and evaluated in [4]. Although the method is quite general, the implementation described here considers only the tracking of one object that translates in a scene. The camera is supposed to be initially centered on the object, and the system's task is to maintain the camera center aligned with the object's position.

III.1 - Message Generation

The high data dimensionality of visual information inhibits a direct application of a neural network to sensor signals, mainly due to generalization constraints [7]. Therefore, in neural target tracking, sensorial inputs first undergo a dimension-reduction process that works out a pair of lower-dimensional vectors ("messages") from input images. The only requirement message generation must meet is to preserve the motion-related differential information. In the implementation described here, such a process only involves row-wise and column-wise summations of image pixels; a sample functional diagram is shown in Fig.3. Other similar mechanisms can be used for rotation tracking; the method's inherent scale-invariance compensates for radial motion [4].

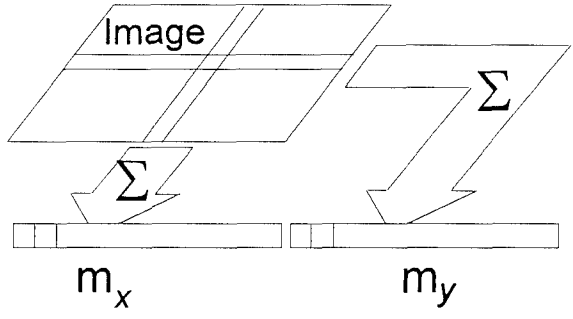


Fig.3 - Message generation

III.2 - Single-network Motion Estimation

The basic idea underlying neural tracking is to let a network infer a target's motion parameters from the differential information contained in a sequence of time-consecutive frames. Due to the aforesaid generalization constraints, the neural system is supplied with differences between time-consecutive messages; two different networks provide horizontal and vertical motion estimates, respectively. The functional schema of the tracking mechanism for one coordinate axis is outlined in Fig.4.

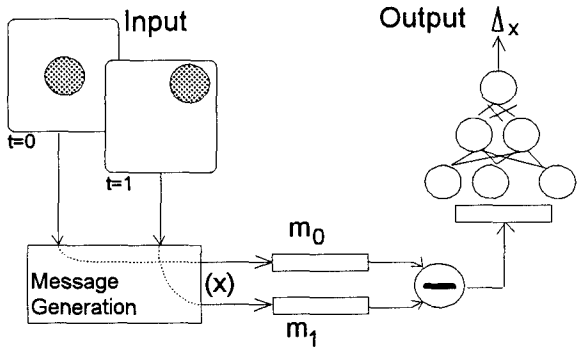


Fig.4 - Neural-tracking schema

A training set includes a set of differences in messages related to several shifts and several objects; the desired outputs are the actual amplitudes of such shifts on the image plane. Experimental evidence shows that the system's run-time performance can become independent of the shapes of training objects. Roughly speaking, proper training lets the neural network learn the differential relationship beween object motion and messages.

III.3 - Multinetwork Motion Estimation

Local accuracy has been evaluated by comparing the prediction by each network with actual object shifts. A different problem lies in *global* stability, i.e., the correctness of the system's closed-loop operation. A system may be locally accurate (i.e., individual prediction may be below a tolerance threshold) but may prove globally unstable. In practice, error may accumulate and cause divergence, so that the camera progressively "loses" the moving object.

An increase in stability on a theoretical basis can be attained by an integrated approach involving network ensembles [8]. Multinetwork evaluation combines the results of several independently trained estimators to work out the system's prediction. The theoretical basis of this schema is that integrating independent estimators reduces error variance, thus removing local "spike" errors. For the neural tracking system, averaging individual estimates can be written as

$$\Delta v_{TOT} = \frac{1}{K}\sum_{k=1}^{K} \Delta v_k \qquad (5)$$

where K indicates the number of involved estimators, and v_k represents the estimation result of the i-th network along the v axis ($v \in \{x,y\}$).

From a circuit-implementation perspective, the multinetwork approach proves extremely efficient, as its structure provides inherent parallelism; therefore, it can support multichip hardware implementation.

IV. EXPERIMENTAL RESULTS

IV.1 - Tracking Stability

The method's effectiveness (in terms of both accuracy and stability) has been verified in previous research, and will be briefly reviewed here. The visual testbed for network *training* involved elementary, binary geometrical shapes (128x128 pixel) belonging to five classes (circles, squares, L-shapes, diamonds, hexagons, butterflies). Each shape class included 16 different scale factors, but only three of them were used to build the actual training set. A preliminary *test* phase of the system's performance involved shapes different from training ones at all 16 possible scales. This analysis made it possible to verify the method's effectiveness in terms of prediction accuracy. A further test phase checked the method stability; test shapes moved along a random trajectory in order to verify that error do not accumulate in closed-loop operation. Figure 5 presents a sample result; the continuous line plots the actual object movements, whereas dots mark the positions of the camera under the control of the tracking system. Other tests involving pictorial images with moving backgrounds confirmed such stability and shape-invariance results.

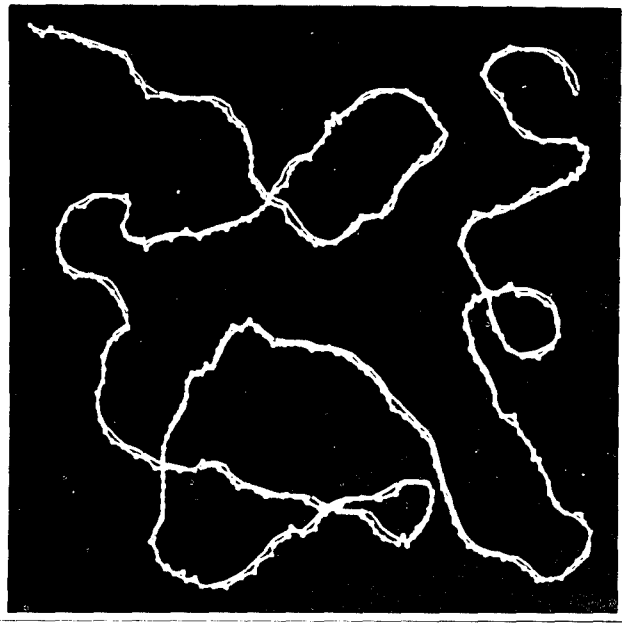


Fig.5 - Stability of the tracking system

IV.2 - Implementation Consistency

This section analyzes the consistency of the digital neuroimplementation of the tracking system. To evaluate the proposed method, we compared the results of software simulations with those obtained by the hardware implementation. The large number of experiments performed made it possible to verify the notable accuracy of the hardware predictions, which always confirmed global stability. A sample of the obtained results on hardware consistency is given in Fig.6, which presents the normalized predictions of a single-network estimator and those of a three-network ensemble. The coordinate axis marks different experiments, whereas the y axis gives the normalized object shifts (in the graph scale, for example, a y value of 0.9 corresponds to a vertical object shift of 30 pixels). The graphs include the results for three different set of experiments with varying shift amplitudes, covering 10-, 20- and 30-pixel shifts, respectively.

The displayed curves demonstrate the correctness of the hardware model; incidentally, the graphs also demonstrate the better performance of the multinetwork estimator, as the three-network curve appears less affected by

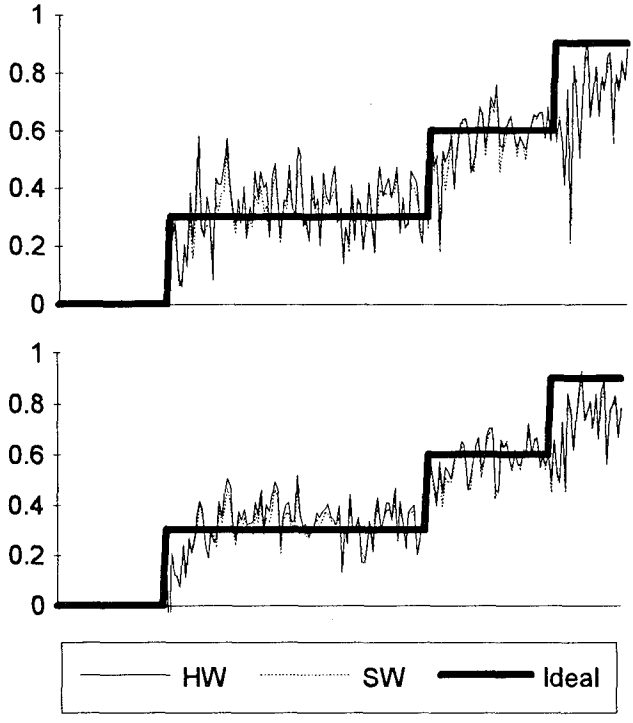


Fig.6 - Comparative tracking results
Top: single-network estimation
Bottom: three-network ensemble estimation

spikes and suggests a more stable behaviour in following the actual shift curve.

The system's timing performance ranged from about 1 msec (single-network setup) to about 2.2 msec (three-network predictor). This value is not proportional to the former because the available boards made it possible to exploit some parallelism. Anyway, this performance makes a technical application to real domains feasible. To sum up, results showed that the proposed hardware-porting method can support neural target tracking effectively.

Current research aims to optimize both neural modelling (better training algorithms) and hardware implementation (efficient architecture design). In particular, a complete automated environment is being set up, which will make it possible to include and test additional features, such as low-pass filtering of local estimates to compensate for local oscillations, and second-order estimation to take into account accelerated motion.

REFERENCES

[1] J.C.Lee, B.J.Shen, W.C.Fang, R.Chellappa "VLSI neuroprocessors for video motion detection" **IEEE Trans. on Neural Networks**, vol.4, no.2 pp.178-191, March 1993.

[2] **IEEE Trans. on Neural Networks**, Spec. issue on Neur.Netw. Hardware, vol.4, no.3, May 1993

[3] D.Anguita, F.Passaggio, R.Zunino "Visual motion tracking: a neural approach" **Proc. World Congress on Neural Networks WCNN'94**, 1994, vol.IV, pp.321-326

[4] D.Anguita, G.C.Parodi, R.Zunino "Neural structures for visual motion tracking" **Machine Vision and Applications**, in press 1995

[5] NeuraLogix, **NLX420 Data Sheet**, June 1992

[6] D.E.Rumelhart, J.L.McClelland (Eds) **Parallel Distributed Processing voll.1-2**, MIT Press, 1986

[7] E.B.Baum, D.Haussler "What size net gives valid generalization?" **Neural Computation**, no.1, pp.151-160, 1989

[8] M.P.Perrone, L.N.Cooper "Learning from what's been learned: supervised learning in multi-neural network systems" **Proc. World Congress on Neural Networks WCNN'93**, 1993, vol.III, pp.354-357.

[9] S.E.Fahlman, C.Lebiere "The cascade-correlation learning architecture" in D.Touretzky (Ed.) **Advance in Neural Information Processing systems 2**, Morgan Kaufman, San Mateo 1990, pp.524-532

LEVEL CROSSING TIME INTERVAL CIRCUIT FOR MICRO-POWER ANALOG VLSI AUDITORY PROCESSING

Nagendra Kumar, Gert Cauwenberghs, Andreas G. Andreou
Department of Electrical and Computer Engineering
The Johns Hopkins University
Baltimore, MD 21218

Abstract

A low-power circuit for level crossing interval measurements on continuous-time auditory signals, as obtained from the outputs of an analog cochlear filter bank, has been designed and fabricated. Experimental results from a fabricated array of 9 level crossing transducers demonstrate frequency-to-voltage conversion over a range covering the audio band. The power consumption is less than 20 μW per cell from a 5 V supply, for the speech frequency range.

1 INTRODUCTION

The location of level crossings of a signal in the time domain reveals much of its characteristics in the spectral domain. In particular, the average time interval between consecutive zero crossings yields the inverse of a **dominant frequency** present in the signal [1]. Neural models of auditory processing in the inner-hair cells using level-crossing signal representations have been formulated [2],[3]. In Ghitza's model [2], sensory outputs from time interval measurements between level crossings are aggregated across cochlear channels to produce an Ensemble Interval Histogram (EIH) spectral measure. EIH has been experimentally shown to provide robust features for speech recognition in the presence of noise. However it is computationally expensive to implement signal processing schemes based on level-crossings on a digital machine [4].

VLSI signal processing systems are often classified as analog, digital or mixed–mode. With a recent emphasis on low power VLSI systems, there has been discussion as to how much processing should be done in analog and how much in digital to achieve optimum performance in terms of power dissipation [5]. Missing in most of these discussions is the fact that it is the design of the algorithm that gives an advantage to either analog or digital implementation.

0-7803-2739-X/95 $4.00 © 1995 IEEE

In this paper, we present a low power mixed-mode circuit cell, designed on the basis of analog VLSI neuromorphic principles [6],[7], for accurate and fast on-line measurement of level crossing time intervals. The cell is suitable for integration with analog VLSI cochlear implementations [8, 9, 10]. It produces a voltage proportional to the time interval between consecutive upward crossings of the input signal with respect to a reference level, sampled at the end of every interval. An array of nine such cells has been fabricated and tested.

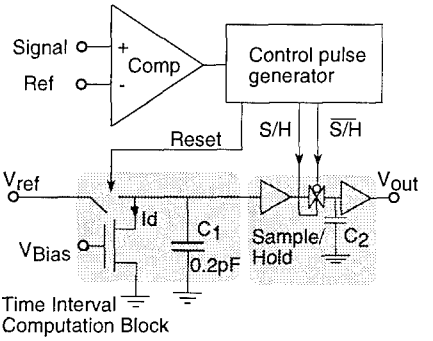


Figure 1: Circuit cell block diagram

2 CIRCUIT DESCRIPTION

The basic functional elements of the circuit cell are shown in Figure 1. Essentially, the cell integrates a constant supplied current I_d onto capacitor C_1 over the time interval between consecutive upward level crossings, and samples the capacitor voltage onto the output at the end of the interval, while almost simultaneously resetting the capacitor voltage for integration in the next cycle. Thus, at any time the output voltage (V_{out}) is a measure of the most recent level crossing interval. A comparator serves to indicate the location in time and the polarity of the level crossings. The most complicated part of the circuitry is the self-timed control logic to ensure the capacitor voltage is reset *after* the output is sampled and held. All this needs to occur in a very short time at every upward level crossing. The time needed to sample and reset the capacitor voltage will ultimately limit the resolution of time-to-voltage conversion. The details of operation are outlined below.

2.1 The Comparator Circuit

The comparator circuit is shown in Figure 2(a). The comparator consists of a standard two-stage differential CMOS amplifier, without frequency compensation. While this circuit topology is typically used in above threshold MOS

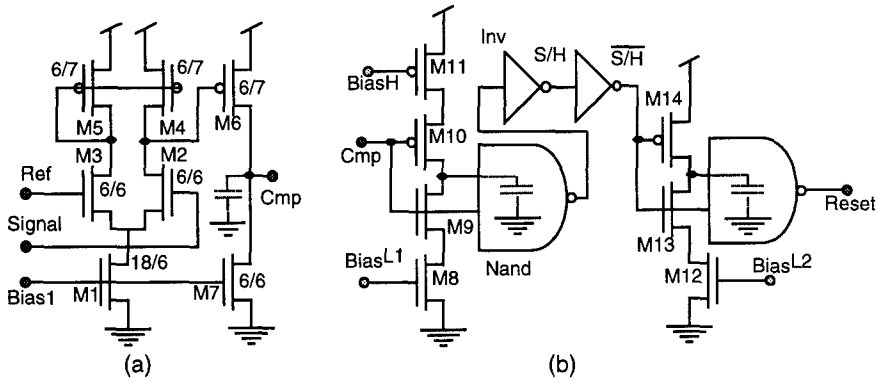


Figure 2: (a) Comparator Circuit, (b) Control pulse generator

operation, transistors M_1 and M_7 are here biased in sub-threshold. Small signal analysis of the comparator, using device sizing of Figure 2 and typical values for the Early voltage in a 2 μm process, yields a DC voltage gain of the order of 3×10^5. However, a large signal analysis reveals a strong asymmetry in the transient behavior. A much larger output current results from M_6 which is driven above threshold when the signal is high, than the fixed sub-threshold current supplied by M_7. This means that the rise time of the comparator is orders of magnitude smaller than its fall time, which is slew-rate limited. This asymmetry is used to create a refractory period in the response of the comparator, during which no second positive transition can be registered. This plays an equivalent role to refractory period in a neuron, avoiding spurious transitions due to noise. The duration of this period can approximately be written as

$$T_{refract} = \frac{C_{load}(V_{DD} - V_{SS})}{2I_{M7}} \tag{1}$$

where C_{load} is the parasitic load capacitance at the output of the comparator, V_{DD} and V_{SS} are the supply voltages, and I_{M7} is the drain current in the transistor M_7. The value for the refractory period cannot exceed the least time separation between consecutive positive transitions. For audio applications, the least time interval of interest is of the order of 25 μsec. With an estimated parasitic load capacitance of 40 fF on the Cmp output node, it would require the drain current of M_7 to be at least 8 nA. This value of current corresponds to sub-threshold operation for a square MOS device in 2 μm technology.

2.2 Self-timed Control Pulse Generator

The circuit for generating the sample-and-hold (S/H) and reset pulses is shown in Figure 2(b). A time delay element is realized using the transistors M_8 to M_{11}. Transistors M_8 and M_{11} are biased in sub-threshold, and can slow down the output voltage swing from a fraction of a microsecond to several microseconds. Bias voltages Bias_H and Bias_{L1} independently control the rise time and fall time respectively. The NAND gate is configured such that it is active low only during the upward transition of the input *Cmp*. At that instant, a pulse is generated of width approximately equal to the fall-time delay in the delay element.

As noted above, the comparator implementation shown in Figure 2(a) has a very short rise-time but a fairly long fall time. The rise time of the delay element is chosen to be longer than the comparator fall time, in order to avoid a spurious pulse at the output of the NAND at the falling edge of the comparator output. The fall time, which is controlled by Bias_{L1} at the gate of M_8 sets the width of the S/H pulse.

The circuit for generating the reset pulse is essentially identical to that just described for generating the S/H pulse, with the exception that the second delay element is configured for no delay in the upward transition of the input. This has been done to ensure that a discharge pulse is generated even for very short duration sample-and-hold pulses. The bias voltage Bias_{L2} at the gate of the transistor M_{12} controls the width of the discharge pulse.

2.3 Sample-and-Hold

The sample-and-hold (S/H) circuit in Figure 1 is implemented in standard form, using an input voltage buffer, a switch, a hold capacitor C_2, and an output voltage buffer. The dummy-compensated complementary switch is driven by the S/H and $\overline{\text{S/H}}$ control signals. Presently, the buffers are implemented as differential transconductance amplifiers with unity-gain feedback, and are hence slew-rate limited. For a non-stationary input the output voltage not only depends on the voltage on C_1 when the S/H pulse arrives, but also on the slew rate of the first buffer in the presence of the load C_2. In case the interval between level-crossings changes significantly from period to period, this may be a limiting factor in the performance of the circuit. In particular, the maximum change in output voltage between consecutive periods is given by

$$\Delta V_{\text{out}}^{\max} = T_{\text{S/H}} \frac{I_{\text{Bias}}}{C_2} \tag{2}$$

where $T_{\text{S/H}}$ is the duration of the sample-and-hold pulse, and I_{Bias} is the bias current in the S/H transconductance buffer.

For a quasi-periodic input, the steady state output voltage V_{out}, relative to the reset voltage V_{ref}, is given by the discharge current I_d integrated on

capacitor C_1 over the level crossing time interval ΔT. Ignoring the conductance of the current source I_d and the finite width of the discharge pulse, V_{out} is approximately given by

$$V_{\text{out}} = V_{\text{ref}} - \Delta T \frac{I_d}{C_1} + V_{\text{off}} \tag{3}$$

where V_{off} is the offset voltage of the sample-and-hold output circuit. In the current design, I_d is the drain current of a MOS transistor in sub-threshold. The value of the current I_d is controlled by the gate voltage V_{Bias} of that transistor.

One useful characterization of the circuit is the maximum bandwidth F_B(in decades) for which the cell produces a useful output. The constraints result from power supply limitations and the resolution of the measurement equipment. Ignoring V_{off} in equation (3), one can write

$$\Delta T = \frac{(V_{\text{ref}} - V_{\text{out}})C_1}{I_d} \tag{4}$$

Taking frequency as reciprocal of the period, and dividing (4) for the maximum and minimum values of V_{out} gives

$$F_B = \log \frac{f_{max}}{f_{min}} = \log \frac{V_{\text{ref}} - V_{\text{out}}^{min}}{\max[(V_{\text{ref}} - V_{\text{out}}^{max}), V_\Delta]} \tag{5}$$

where the minimum and maximum output voltages V_{out}^{min} and V_{out}^{max} are determined by the the power supply and the range of operation for the S/H, and

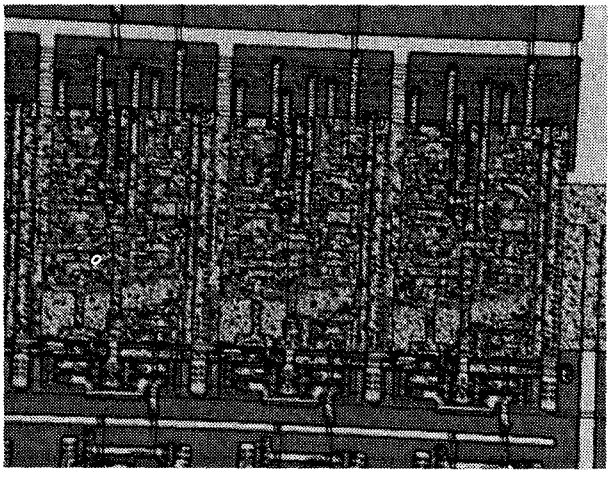


Figure 3: Micro-graph of three cells on the chip. Cell dimensions are $120\mu m \times 227\mu m$ in $2\mu m$ CMOS

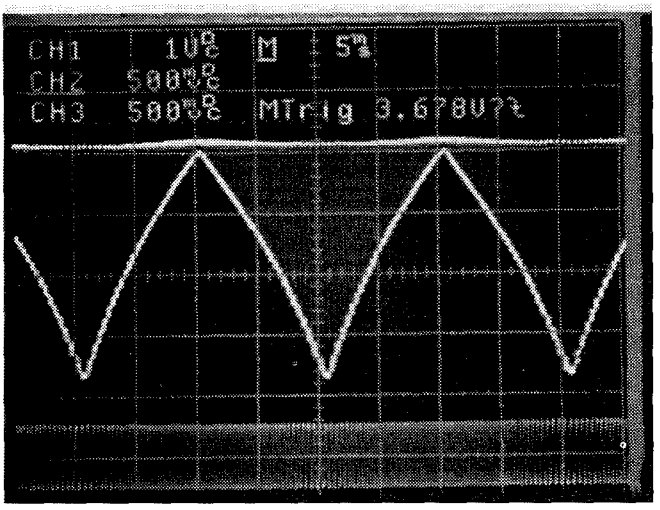


Figure 4: Output waveform for a frequency modulated input - The lower waveform is the frequency modulated input, upper waveform (the straight line and thin vertical lines) show the resetting and discharging of the capacitor, and the triangular wave is the final output

V_Δ is the resolution limit due to the noise in the circuit and the measurement equipment.

3 EXPERIMENTAL RESULTS

An array of nine circuit cells has been fabricated through MOSIS. A micrograph of the chip, depicting 3 cells of dimension $120\mu m \times 227\mu m$ each, fabricated in N-well $2\mu m$ CMOS technology, is shown in Figure 3. Common control signals are provided to all cells, except for the bias voltage V_{Bias}.

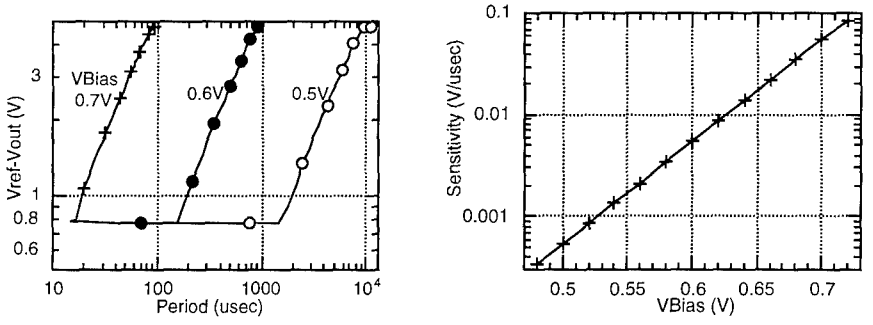


Figure 5: (a) Output voltage versus input signal period, at three different discharge current bias settings. (b) Output sensitivity versus bias voltage

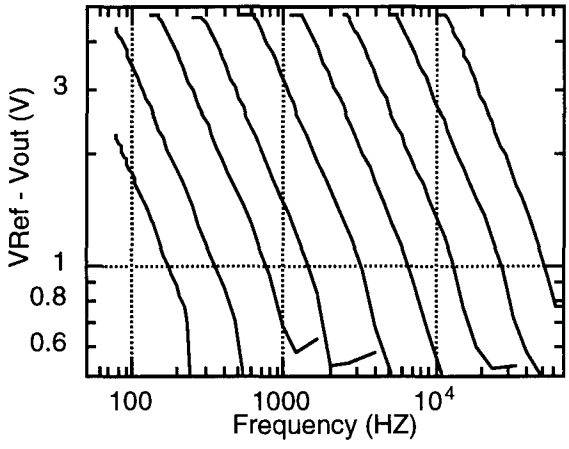


Figure 6: Frequency-to-voltage characteristics of the transducer array over the audio frequency range

Bias voltages are tapped from equally spaced points on a resistive polysilicon wire which is used as a resistive voltage divider. Voltages at the two ends of the polysilicon wire may be controlled externally, thus locally controlling the discharge current I_d in the individual cells.

3.1 Temporal Response

Figure 4 illustrates the response of a single cell when a frequency modulated signal is applied to the input. In this example an input with minimum frequency of 1kHz and a maximum frequency of about 4kHz is used, with a triangular modulation at 50Hz. It can be seen that the output voltage is nearly triangular, with a value proportional to the input frequency.

3.2 Tuning Characteristics

From (3), the relationship between ΔT and V_{Bias} is expected to be linear, proportional to the discharge current I_d. Plots of the output voltage V_{out} versus the period ΔT of a periodic input signal are shown in Figure 5(a), for three different values of V_{Bias}. As shown, the bias voltage setting allows to scale the linear frequency versus output voltage response over a wide range of frequencies. We define as the output sensitivity S as the change in output voltage V_{out} in response to a unit change in period ΔT. From (3), the quantity S is given as I_d/C_1, in the linear range of operation. The recorded output sensitivity as a function of bias voltage V_{Bias} is shown in Figure 5(a). The exponential relationship between the S and V_{Bias} derives from the fact that the in the sub-threshold region of operation, the saturation drain current of a MOS transistor I_d of a MOS transistor is exponential in the gate voltage

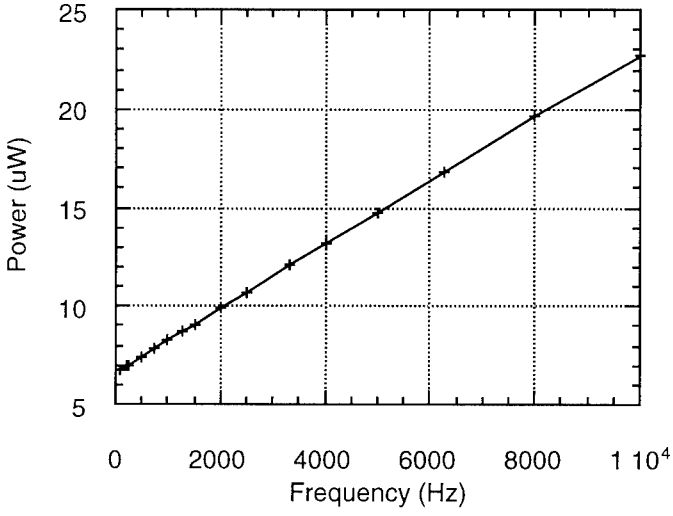


Figure 7: Power consumption of the fabricated cell with a 5V power supply

V_{Bias}.

$$I_d \simeq I_0 e^{\kappa(V_{\text{Bias}}/V_T)} \tag{6}$$

3.3 As an Array of Processors

By applying a uniform linear voltage gradient across the polysilicon wire that provides V_{Bias}, an exponential distribution in bias current I_{Bias} of the cells is obtained. This allows each cell to be tuned to a particular range of frequencies in the corresponding input signal, whereby the center frequencies of the cells are spaced uniformly in the log frequency domain. The motivation is to interface the array of level crossing cells directly with outputs from a cochlear filter bank with matched center frequencies. As a proof of concept, a linear gradient in V_{Bias} is constructed by tapping the bias voltages on equally spaced points along a resistive polysilicon line. The maximum and the minimum values of V_{Bias}, defining the corner frequencies, are applied at the ends of the resistive line. A frequency sweep in the audio range is input to the whole array. The resulting outputs are shown in Figure 6. The useful frequency range of each cell is about a decade, suitable for use with cochlear filter banks which typically have bandwidths less than an octave per channel.

3.4 Power Consumption

The total power consumption can be broadly divided into two separate components. The first component is that which is fixed. It is due to the constant bias currents in the comparator, the time-interval computation block, and the sample-and-hold. The value for these bias currents is determined using

the criterion that the circuit should function in the worst case scenario. In our case we require that the circuits should continue to function properly for input frequencies up to 40 kHz. The bias current in the S/H is set to allow instantaneous changes in level-crossing-intervals to be recorded. The second component is due to the switching in the control-pulse generator. Since the amount of switching is proportional to the input frequency, this component is directly proportional to the average input frequency. The experimental data for power consumption is shown in Figure 7. For the speech input frequency range (less than 8 kHz) the power consumption is less than $20\mu W$. Note that it is possible to further reduce the power consumption by reducing the power supply voltage.

4 CONCLUSION

Power consumption and robustness are key issues in the development of speech-processing applications for personal digital assistants. This compact and low-power VLSI circuit implementation exploits the robustness [1] of zero-crossing-based signal processing at a very low power cost. Sub-threshold exponential characteristics of the MOS transistor and continuous operation make arrays of cells particularly suitable to multi-resolution signal processing such as in a mammalian cochlea. The power consumption of the circuit is small due to two reasons. First, since the application involves audio-frequencies, is has been possible to use sub-threshold CMOS circuits that consume very little current. Second, due to the asynchronous self-timed design, unnecessary switching has been eliminated.

REFERENCES

[1] B. Kedem, "Spectral analysis and discrimination by zero-crossings," *Proceedings of the IEEE*, vol. 74, pp. 1477–1493, Nov. 1986.

[2] O. Ghitza, "Auditory nerve representation as a front-end for speech recognition in a noisy enviornment," *Computer Speech and Language*, vol. 1, pp. 109–130, 1986.

[3] K. Wang, S. Shamma, and W. J. Byrne, "Noise robustness in the auditory representation of speech signals," in *ICASSP*, vol. 2, pp. 335–338, 1993.

[4] C. R. J. Jr., "A comparison of auditory models for automatic speech recognition," Master's thesis, M.I.T., May 1992.

[5] E. Vittoz, "Analog VLSI signal processing: why, where and how?," *Journal of Analog Integrated Circuits and Signal Processing*, pp. 27–44, June 1994.

[6] C. A. Mead, *Analog VLSI and Neural Systems.* Reading, MA: Addison-Wesley, 1989.

[7] A. G. Andreou, K. A. Boahen, P. O. Pouliquen, A. Pavasović, R. E. Jenkins, and K. Strohbehn, "Current-mode subthreshold MOS circuits for analog VLSI neural systems," *IEEE Trans. Neural Networks*, vol. 2, pp. 205–213, March 1991.

[8] W. Liu, A. G. Andreou, and M. G. Goldstein, "Voiced speech representation by an analog silicon model of the auditory periphery," *IEEE Transactions on Neural Networks*, vol. 3, pp. 477–487, May 1992.

[9] L. Watts, D. A. Kerns, R. F. Lyon, and C. A. Mead, "Improved implementation of the silicon cochlea," *IEEE Journal of Solid State Circuits*, vol. 27, pp. 692–700, May 1992.

[10] J. Lazzaro, J. Wawrzynek, M. Mohwald, M. Sivilotti, and D. Gillespie, "Silicon auditory processors as computer peripherals," *IEEE Transactions on Neural Networks*, vol. 4, pp. 523–528, May 1993.

Communications

OPTIMUM LAG AND SUBSET SELECTION FOR A RADIAL BASIS FUNCTION EQUALISER

E.S.CHNG †, B.MULGREW †, S.CHEN ‡ and G.GIBSON+

† *Dept. of Electrical Eng., The University of Edinburgh, U.K.*
‡ *Dept. of Electrical and Electronic Eng., The University of Portsmouth, U.K*
+ *Scottish Agricultural Statistics Service, The University of Edinburgh, U.K*

Abstract

This paper examines the application of the radial basis function (RBF) network to the modelling of the Bayesian equaliser. In particular, we study the effects of delay order d on decision boundary and attainable bit error rate (BER) performance. To determine the optimum delay parameter for minimum BER performance, a simple BER estimator is proposed.

The implementation complexity of the RBF network grows exponentially with respect to the number of input nodes. As such, the full implementation of the RBF network to realise the Bayesian solution may not be feasible. To reduce some of the implementation complexity, we propose an algorithm to perform subset model selection. Our results indicate that it is possible to reduce model size without significant degradation in BER performance.

Indexing Term: Bayesian equaliser, neural networks, RBF network, BER.

1 Introduction

It is well-known that the performance of neural network (NN) equaliser is superior to the conventional linear equaliser for the digital communication symbol-by-symbol equalisation problem [1–3]. The superiority of the NN structure is due to its ability to model the optimum Bayesian decision boundary better than the conventional linear systems. In many practical equalisation problems, the Bayesian decision boundary is often highly nonlinear, and in some cases, not linearly separable. It is thus not surprising that NN techniques, which are capable of modelling any nonlinear decision boundaries, have become very popular in equalisation problems. This paper continues this theme and investigates the application of the radial basis function (RBF) network to realise the Bayesian equaliser.

The paper is organised as follows: In Sec. 2.1, we extend the work reported in [1, 2] to show the effects of delay order on the Bayesian equaliser's decision boundary and BER performance. Our analysis show that the equaliser achieves different attainable BER performance when different delay order

0-7803-2739-X/95 $4.00 © 1995 IEEE

is applied under the same signal to noise (SNR) operating condition. To determine the optimum delay order, a simple BER estimate for the equaliser is proposed in Sec. 3. The implementation complexity of the RBF equaliser is also discussed, and an algorithm to select small-sized RBF models which approximate the Bayesian solution is presented in Sec.4

2 Implementing the Bayesian equaliser

An established model of a digital communication channel subjected to inter-symbol interference (ISI) for a multi-level pulse amplitude modulation (2-ary PAM) scheme is described by the following equation [2,4]:

$$r(k) = \sum_{i=0}^{n_a-1} s(k-i)a(i) + e(k) \tag{1}$$

where $r(k)$ is the received signal at time k, $s(k)$ is an independently identically distributed (i.i.d) transmitted symbol with symbol constellation defined by the set $\{\pm 1\}$, $a(i)$ are the channel impulse response coefficients with the length of the impulse response n_a, and $e(k)$ is the additive white Gaussian noise $e(k)$ of zero mean and variance σ_e^2 [2,4]. The equaliser uses an array of received signal

$$\mathbf{r}(k) = [r(k), \cdots, r(k-m+1)]^T \tag{2}$$

to estimate the transmitted symbol $s(k-d)$, i.e. $\hat{s}(k-d)$. The integers m and d are known as the feedforward order and delay order respectively.

The transmitted symbols that affect the input vector $\mathbf{r}(k)$ is the transmit sequence $\mathbf{s}(k) = [s(k), \cdots, s(k-m-n_a+2]^T$. There are $N_s = 2^{m+n_a-1}$ possible combinations of these input sequences, i.e. $\{\mathbf{s}_j\}, 1 \leq j \leq N_s$ [2]. In the absence of noise, there are N_s corresponding received sequences $C_d = \{\mathbf{c}_j\}, 1 \leq j \leq N_s$, which are also referred to as channel states. The subscript d in C_d denotes the delay order used. The values of the channel states are defined by the following equation,

$$\mathbf{c}_j = F[\mathbf{s}_j] \quad 1 \leq j \leq N_s \tag{3}$$

where the matrix $F \in R^{m \times (m+n_a-1)}$ is

$$F = \begin{bmatrix} a(0) & a(1) & \dots & a(n_a-1) & 0 & \dots & \dots & 0 \\ 0 & a(0) & a(1) & \dots & a(n_a-1) & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & a(0) & a(1) & \dots & a(n_a-1) \end{bmatrix} \tag{4}$$

When noise is present, the received vector $\mathbf{r}(k)$ has a Gaussian distribution with expected values corresponding to the respective $\mathbf{c}_j$.

The set of channel states $\{\mathbf{c}_j\}, 1 \leq j \leq N_s$ can be partitioned according to the value of $s(k-d)$, i.e., channel states associated with $s(k-d) = +1$

belong to the class $C_d^{(+)}$, and channel states associated with $s(k-d) = -1$ belong to the class $C_d^{(-)}$. The response of the Bayesian equaliser prior to the slicer is [2],

$$f(\mathbf{r}(k)) = \sum_{\mathbf{c}_i \in C^{(+)}} p_i (2\pi\sigma_e^2)^{-m/2} \exp(-\|\mathbf{r}(k) - \mathbf{c}_i\|^2 / 2\sigma_e^2) - \sum_{\mathbf{c}_j \in C^{(-)}} p_j (2\pi\sigma_e^2)^{-m/2} \exp(-\|\mathbf{r}(k) - \mathbf{c}_j\|^2 / 2\sigma_e^2) \quad (5)$$

where p_i and p_j are the *a priori* probabilities of occurrence for the respective channel states. In the case of i.i.d transmitted symbols, $p_i = p_j = 1/N_s$. The output of the Bayesian equaliser $\hat{s}(k-d)$ is $\text{sgn}(f(\mathbf{r}(k)))$, where sgn(.) is the signum function.

From Eq. 5, it is obvious that the structure of the RBF network is identical to the Bayesian equaliser [2], and that the RBF network realises precisely the Bayesian solution when the weights, centres and the nonlinearity of hidden units are set accordingly.

2.1 Effects of delay order on decision boundaries

The set $\{\mathbf{r}(k) | f(\mathbf{r}(k)) = 0\}$ defines the Bayesian decision boundary and is dependent on the channel state values and the delay order parameter [1, 2]. The channel states are determined by the channel impulse response and the equaliser feedforward order. The channel states however do not uniquely define the decision boundary. Given a set of channel states, the decision boundary can be changed by using different delay orders.

As an example, the Bayesian decision boundaries realised by a RBF equaliser with feedforward order $m = 2$ for channel $H(z) = 0.5 + 1.0z^{-1}$ is examined. Fig 1a lists all the 8 possible combinations of the transmitted signal sequence $\mathbf{s}(k)$ and the corresponding channel states $\mathbf{c}_i$. Fig. 1b depicts the corresponding decision boundaries for the different delay orders. Note the dramatic change in the shape of the decision boundaries for different delay orders.

The use of different delay orders also results in different limits of BER performance. To determine the optimum delay order, a computationally simple method to estimate the BER of the Bayesian equaliser is presented in Sec. 3.2.

3 Probability of mis-classification

This section presents the analysis of probability of mis-classification of the Bayesian equaliser.

S/No	Transmitted symbols			Channel State	
i	$\mathbf{s}(k)$			$\mathbf{c}_i$	
	[$s(k)$	$s(k\text{-}1)$	$s(k\text{-}2)$]	[$r(k)$	$r(k\text{-}1)$]
1	1	1	1	1.5	1.5
2	1	1	-1	1.5	-0.5
3	1	-1	1	-0.5	0.5
4	1	-1	-1	-0.5	-1.5
5	-1	1	1	0.5	1.5
6	-1	1	-1	0.5	-0.5
7	-1	-1	1	-1.5	0.5
8	-1	-1	-1	-1.5	-1.5

Fig (a) : State Table

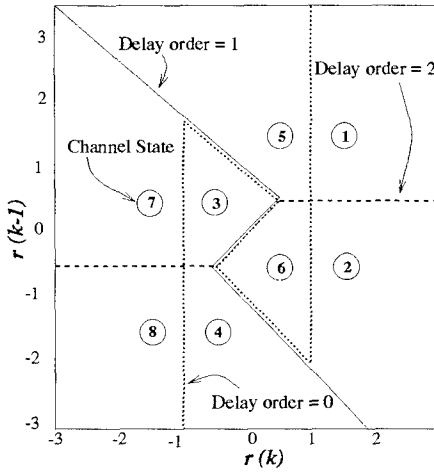


Fig (b) : Decision boundaries

Figure 1: (a) Input and desired channel states for channel $H(z)$,
(b) Bayesian decision boundaries for channel $H(z)$.

3.1 Evaluating the probability of error

We define $Z^+ \subset R^m$ to be the region of $\mathbf{r}(k)$ classified as $+1$ and $Z^- \subset R^m$ to be the region classified as -1. The probability of making a wrong decision P_e is

$$P_e = \sum_{\mathbf{c}_i \in \mathbf{C}^+} p_i \int_{\mathbf{r} \in Z^-} f_{\mathbf{r}|\mathbf{c}_i}(\mathbf{r}) d\mathbf{r} + \sum_{\mathbf{c}_j \in \mathbf{C}^-} p_j \int_{\mathbf{r} \in Z^+} f_{\mathbf{r}|\mathbf{c}_j}(\mathbf{r}) d\mathbf{r} \qquad (6)$$

where $f_{\mathbf{r}|\mathbf{c}_l}(\mathbf{r})$ is the probability density function (pdf) of the noisy received vector $\mathbf{r}$ conditioned on the received channel state being $\mathbf{c}_l$,

$$f_{\mathbf{r}|\mathbf{c}_l}(\mathbf{r}) = (2\pi\sigma_e^2)^{-m/2} \exp(-\|\mathbf{r} - \mathbf{c}_l\|^2/(2\sigma_e^2)) \qquad (7)$$

Because the symbol constellation is symmetric, equation (6) can be reduced to

$$P_e = 2 \sum_{\mathbf{c}_i \in \mathbf{C}^+} p_i \int_{\mathbf{r} \in Z^-} f_{\mathbf{r}|\mathbf{c}_i}(\mathbf{r}) d\mathbf{r} \qquad (8)$$

3.2 Estimating the probability of error

The evaluation of BER using Eq. 8 involves evaluating m-dimensional integrals over the error region Z^-. As a closed-form solution for the expression does not exist, one must resort to numerical methods. This option however

is un-attractive for large m. As our requirement to find BER performance is only one of comparing relative performance for equalisers using different delay orders, a simple approximation may be used to estimate the BER. The probability of mis-classifications, P_e, can be expressed as

$$P_e = \frac{1}{N_s} \sum_{i=1}^{N_s} P_e(\mathbf{c}_i) \tag{9}$$

where $P_e(\mathbf{c}_i)$ is the probability of mis-classification conditioned on the noise-free channel state being $\mathbf{c}_i$. It can be shown that in the case when SNR $\rightarrow \infty$, $P_e(\mathbf{c}_i)$ can be reduced to the minimum distance bound [5], i.e.,

$$P_e(\mathbf{c}_i) = Q(|\zeta_i|/\sigma_e) = \int_{|\zeta_i|}^{\infty} (2\pi\sigma_e^2)^{-1/2} \exp(-x^2/(2\sigma_e^2)) dx \tag{10}$$

where $|\zeta_i|$ is the absolute minimum Euclidean distance of $\mathbf{c}_i$ to the decision boundary.

Although Eq. 10 is only valid for very high SNR, it can be applied with Eq. 9 to evaluate a rough estimate of the BER performance. Our simulation results however indicate that the proposed estimator gives good BER estimates even for low SNRs.

3.3 BER estimate : Some simulation results

Simulations were conducted to compare the BER results obtained using Eqs. 9 and 10 with those obtained by the Monte Carlo (MC) simulations. The following channels which have the same magnitude but different phase response were used,

$$H1(z) = 0.8745 + 0.4372z^{-1} - 0.2098z^{-2} \tag{11}$$

$$H2(z) = 0.2620 - 0.6647z^{-1} - 0.2623z^{-2} \tag{12}$$

For the experiment, the equaliser's feedforward order was chosen to be 4 with the transmit symbol alphabet $\{\pm 1\}$. Fig 2 compares the BER estimates of Eqs. 9 and 10 with those of MC simulations for the two channels using different delay orders. The results show that the proposed BER estimate is very accurate. To illustrate the strong dependence of the equaliser's performance with respect to the delay order, we plot the performance of the equaliser using the delay parameter as the horizontal axis in Fig 3.

4 Selecting subset RBF model

The implementation of the full RBF solution requires the use of all N_s channel states. In some cases, equivalent Bayesian solution may be realised by using a subset of the full model. For example, the decision boundaries of delay

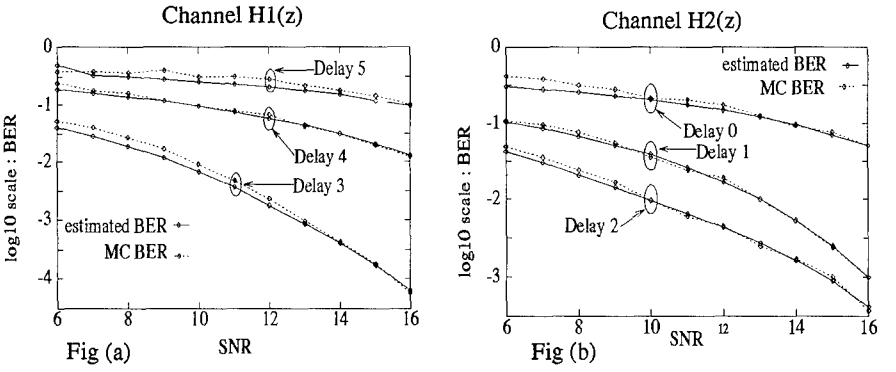


Figure 2: Estimated and MC simulations of BER vs SNR for $H1(z)$ (Fig a) and $H2(z)$ (Fig b).

order 1 and 2 (Fig 1b) can be realised by a RBF model with centres $\mathbf{c}_i$ from $\{\mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$ (Fig. 4a,b).

In many cases, we have observed that it is possible to find small subset RBF model to approximate the full model's solution when the decision boundary is linearly separable. The task is however much more difficult when the decision boundary is nonlinearly separable.

This section examines subset model selection algorithms to reduce implementation complexity of the RBF equaliser. The objective is to find a smaller-sized, in terms of number of centres, RBF model to realise or to approximate the same Bayesian solution as the full model. To understand how centres affect boundary, we analyse the effects of centre positions on decision boundary when $\sigma_e \rightarrow 0$. Defining the points on the boundary as $\mathbf{r}_0$, i.e. $\{\mathbf{r}_0 | f(\mathbf{r}_0) = 0\}$, Eq. 5 becomes

$$\sum_{\mathbf{c}_i \in C_d^{(+)}} p_i (2\pi\sigma_e^2)^{-m/2} \exp(-\|\mathbf{r}_0 - \mathbf{c}_i^+\|^2/2\sigma_e^2) \quad = \quad \sum_{\mathbf{c}_j \in C_d^{(-)}} p_j (2\pi\sigma_e^2)^{-m/2} \exp(-\|\mathbf{r}_0 - \mathbf{c}_j^-\|^2/2\sigma_e^2) \qquad (13)$$

When $\sigma_e \rightarrow 0$, the sum on the l.h.s. of Eq. 13 becomes dominated by the closest centres to $\mathbf{r}_0$, i.e.

$$U^+ = \min_{\mathbf{c}_k \in C_d^{(+)}} \{\|\mathbf{r}_0 - \mathbf{c}_k\|\} \qquad (14)$$

This is because the contribution of centres $\mathbf{c}_k \notin U^+$ converges much more quickly to zero when $\sigma_e \rightarrow 0$ than centres belonging to U^+. Similarly, the sum on the r.h.s of Eq. 13 becomes dominated by the closest centres, U^-. This

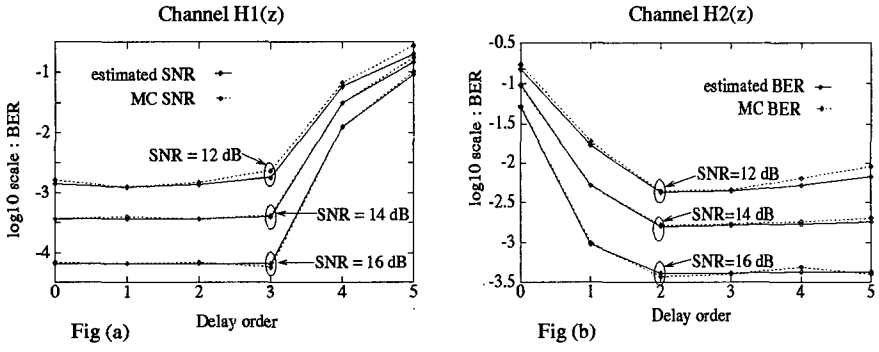


Figure 3: Estimated and MC simulations BER vs delay order for SNR 12dB, 14dB and 16dB for $H1(z)$ (Fig a) and $H2(z)$ (Fig b).

implies that the asymptotic decision boundaries are hyper-planes between pairs of U^+, U^- and that the set of all U^+, U^- defines the asymptotic decision boundaries. The following algorithm may be employed to find the set of all U^+, U^-.

Algorithm 1 : Finding U^+, U^-

$$\vdots$$

For $\mathbf{c}_i \in \mathbf{c}^+$

For $\mathbf{c}_j \in \mathbf{c}^-$

$\mathbf{r} = \mathbf{c}_i + (\frac{\mathbf{c}_j - \mathbf{c}_i}{2})$

$$\text{if} \left[\begin{array}{l} f(\mathbf{r}) = 0 \text{ and} \\ \mathbf{c}_i = \min_{\mathbf{c}_k \in C_d^{(+)}} \{\|\mathbf{r} - \mathbf{c}_k\|\} \end{array} \right] \tag{15}$$

$\mathbf{c}_i \rightarrow U^+, \mathbf{c}_j \rightarrow U^-$

next $\mathbf{c}_j$,

next $\mathbf{c}_i$.

Algorithm 1 was tested to find subset models from the full RBF model (Sec. 2.1) used on channel $H(z) = 0.5 + 1.0z^{-1}$. As expected, when delay order 0 was used, all the centres, $\{\mathbf{c}_1, \ldots, \mathbf{c}_8\}$ were picked to form the subset model (Fig. 1b). For the case of using delay order 1, the selected subset model consisted of centres $\{\mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$ These results can be easily verified by visual inspection of the boundary formation as illustrated in Figs. 4a and 1b.

Although algorithm 1 works, the selection process is not optimum in the sense that redundant centres may be included to form the subset model. To illustrate, consider the selected subset model when delay order 2 was used.

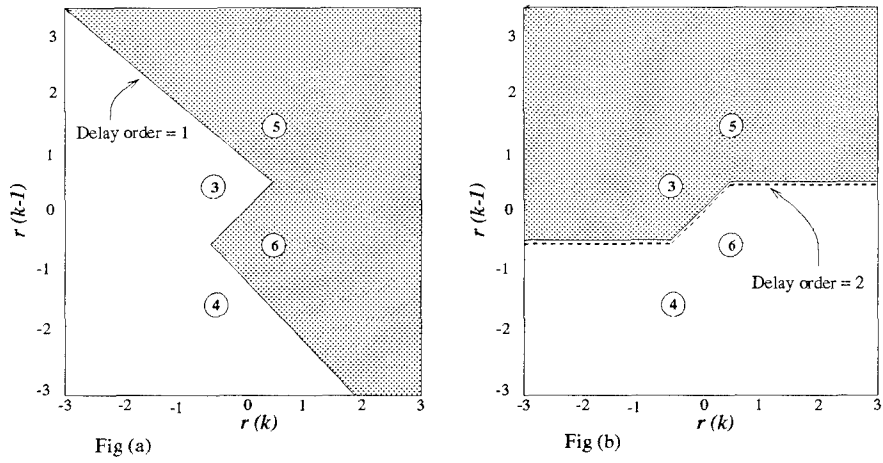


Figure 4: Realisation of decision boundary using subset RBF model.
Decision boundary for (Fig a) delay $d = 1$, (Fig b) delay $d = 2$.

By visual inspection of Figs. 4b and 1b, it is clear that the subset model with centres $\{\mathbf{c}_3, \mathbf{c}_4, \mathbf{c}_5, \mathbf{c}_6\}$ is sufficient to realise the Bayesian boundary. Algorithm 1, however, picked all the centres to form the subset model. The reason for including centres $\{\mathbf{c}_1, \mathbf{c}_2\}$ and $\{\mathbf{c}_7, \mathbf{c}_8\}$ is that these two pairs of centres satisfy Eq. 15 in algorithm 1 and thus also define the asymptotic decision boundary. They are however unnecessary because the decision boundary formed using centres $\{\mathbf{c}_3, \mathbf{c}_4\}$ and $\{\mathbf{c}_5, \mathbf{c}_6\}$ are the same.

To minimise the inclusion of redundant centres, an additional condition is introduced in Eq. 15 to verify if the new centres under consideration affect decision boundary. If the decision boundary changes with the inclusion of the new centres, they will be accepted, otherwise ignored. By adding this condition, some redundant centres will not be included in the selected subset model. The algorithm for the improved version is as follows:

Algorithm 2 : Finding U^+, U^-

$$\vdots$$

$$\mathbf{r} = \mathbf{c}_i + (\frac{\mathbf{c}_j - \mathbf{c}_i}{2})$$

$$\text{if} \quad \left[\begin{array}{l} f(\mathbf{r}) = 0 \text{ and} \\ \mathbf{c}_i = \min_{\mathbf{c}_k \in \mathbf{c}^+} \{\|\mathbf{r}_0 - \mathbf{c}_k\|\} \text{ and} \\ f_s(\mathbf{r}) \neq 0 \end{array} \right]$$

$$\mathbf{c}_i \rightarrow U^+, \ \mathbf{c}_j \rightarrow U^-$$

$$f_s = \text{RBF model formed using } U^+, U^- \text{ as centres.}$$

$$\vdots$$

4.1 Subset model selection : some simulation results

Simulations were conducted to select subset models from the full model used on channels $H1(z)$ and $H2(z)$. The feed forward order used was $m = 4$, resulting in a full model with $N_s = 2^{m+n_a-1} = 64$ centres. Using SNR condition at 16dB, simulations were conducted to evaluate the performance of the subset RBF, full RBF and the linear Wiener equalisers for the two channels. The results are tabulated in Table 1a and 1b; The first column of each table indicates the delay order parameter, the second column shows the size of the subset model used while the third, fourth and fifth columns list the BER performance of the respective equalisers and the last column indicates if the decision boundary is linearly or not-linearly separable.

Our results indicate that the full RBF models' BER performance, for cases when the decision boundary is linearly separable, are normally better than those when the decision boundary is not linearly separable. This is not surprising since decision boundaries which are not linearly separable tend to be much more complicated and have more centres with different decision outputs near to each other. It was also observed that smaller-sized RBF subset models can be found for the case when the boundary is linearly separable, and their performance not significantly poorer than the full model's performance.

Delay	Subset Size	Subset BER	Full-model BER	Linear-Eq BER	Decision Boundary
0	56	-4.09	-4.09	-3.44	Linear Sep.
1	57	-4.14	-4.14	-3.07	Linear Sep.
2	32	-4.11	-4.12	-2.36	Linear Sep.
3	32	-4.11	-4.12	-1.84	Linear Sep.
4	48	-1.91	-1.91	-0.59	Not-Linear Sep.
5	64	-0.97	-0.97	-0.37	Not-Linear Sep.

Table a : Channel H1(z)

Delay	Subset Size	Subset BER	Full-model BER	Linear-Eq BER	Decision Boundary
0	56	-0.80	-1.30	-0.37	Not-Linear Sep.
1	46	-2.99	-2.99	-1.61	Linear Sep.
2	38	-3.38	-3.38	-2.67	Linear Sep.
3	56	-3.43	-3.43	-1.94	Linear Sep.
4	55	-3.32	-3.32	-1.16	Not-Linear Sep.
5	54	-3.41	-3.41	-0.80	Not-Linear Sep.

Table b : Channel H2(z)

Table 1: Comparing the performance of the full-size (64 centres) RBF equaliser, subset RBF equaliser and the Wiener equaliser for Channel $H1(z)$ (Table 1a) and Channel $H2(z)$ (Table 1b) at SNR=16db.

5 Conclusions

This paper discusses the implementation of the RBF equaliser to realise the Bayesian solution. In particular, the effects of the delay order parameter on decision boundaries and BER performance is highlighted. We have showed that the attainable BER performance depends strongly on the delay order parameter and can be significantly different for various values of the delay order. To determine the optimum operating delay order parameter, a simple BER estimator for the RBF equaliser is proposed.

The implementation complexity of the RBF equaliser to realise the Bayesian solution is also discussed. To reduce some of the implementation complexity, we have introduced an algorithm to select subset model from the full RBF implementation. Our results indicate that that good subset models with no significant degradation in BER performance may be found.

Acknowledgement

We would like to thank Achilleas G Stogioglou of Edinburgh University for his many helpful comments and ideas during the development of this work.

References

[1] G.J.GIBSON, S.SIU, and C.F.N.COWAN, "The application of nonlinear structures to the reconstruction of binary signals", *IEEE Trans. Signal Processing*, vol. 39, no. 8, pp. 1877–1884, 1991.

[2] S.CHEN, B.MULGREW, and P.M.GRANT, "A clustering technique for digital communications channel equalization using radial basis function networks", *IEEE Trans. Neural Networks*, vol. 4, no. 4, pp. 570–579, 1993.

[3] C.P.CALLENDER and C.F.N.COWAN, "A comparison of six different non-linear equalisation techniques for digital communication systems", *Proc. of the Seventh European Signal Processing Conference*, pp. 1524–1527, 1994.

[4] S.U.H.QURESHI, "Adaptive equalization", *Proc. IEEE*, vol. 73, no. 9, pp. 1349–1387, 1985.

[5] B.HUGHES, "On the error probability of signals in additive white Gaussian noise", *IEEE Trans. Information Theory*, vol. 37, no. 1, pp. 151–155, 1991.

CHANNEL EQUALIZATION BY FINITE MIXTURES AND THE EM ALGORITHM[1]

Lei Xu
Dept. of Computer Science
The Chinese University of Hong Kong, Shatin, Hong Kong

ABSTRACT

The model of finite mixtures and the EM learning algorithm have been applied to the task of channel equalization in communication problems for the channels that may vary its properties between a number of different modes. Computer experiments have also been given to show that the proposed approach work well with a promising potential for applications.

1. INTRODUCTION

The finite mixture model is an important parametric model in the family of unsupervised learning models. It is popular in statistics and pattern recognition [3]. The model is used for a variety of problems, including clustering analysis, estimation of densities of multiple populations, and unsupervised classifier design. The finite mixture model and the EM algorithm have gained an increasing attention in neural network literature[1][4-8][11-14].

In this paper, we consider the application of the finite mixture model and its modification to the task of channel equalization in communication problems for the channels that may vary its properties between a number of different modes.

2. CHANNEL EQUALIZATION

In communication channels, dispersion causes interference between successive samples and greatly complicates reliable transmission and reception of signals. Presently, the problem is usually solved by a technique called *Adaptive Channel Equalization.* An adaptive equalizer is installed in the receiving end. It is usually an adaptive transversal filter (LMS filter) which forms an inverse transfer function for the channel within the channel passband so that the cascade transfer function is essentially flat (constant) in magnitude and is essentially linear in phase. This adaptive equalizer works quite well, when the transfer function of the channel is not too complicated and can be approximately regarded as a linear transversal filter. However, in some communication tasks, the transfer function of the channel is actually much complicated. For example, in radio communication and mobile phones, the transfer characteristics of the channel will vary from time to time and place to place. It is often quite difficult for a simple LMS filter to cope with these complicated cases. In this section, we propose some techniques for tackling

[1]This research was supported by the HK RGC Earmarked Grant CUHK250/94E.

0-7803-2739-X/95 $4.00 © 1995 IEEE

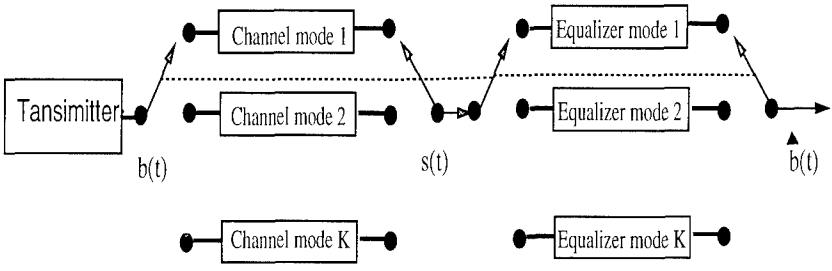


Figure 1: A complicated communication channel and the proposed multi-mode equalizer.

this problem based on the finite mixture model and the EM algorithm.

We assume that the channel varies its transfer characteristics between a number of different modes. More precisely, as shown in Fig.1, we imagine that there is a synchronous random switch in both the sending end and the receiving end. During the communication, the switch randomly chooses a channel mode j with a prior probability α_j and then keeps using this channel mode for a random period T. During the channel mode j is under using, we have the receiving signal given by

$$s(t) = H_j(b(t), \cdots, b(t - p_j + 1), \theta_j) + \varepsilon(t). \tag{1}$$

where $b(t)$ is the signal at the sending end, and $H_j(.)$ is the transfer function of the channel in mode-j. Particularly, we consider the linear transversal filter as follows

$$\begin{aligned} s(t) &= c_{j,0} + \sum_{\tau=1}^{p_j} c_{j,\tau} b(t - \tau + 1) + \varepsilon(t), \\ thus \quad \theta_j &= [c_{j,0}, c_{j,1}, \cdots, c_{j,p_j}]. \end{aligned} \tag{2}$$

In eq.(1), $\varepsilon(t), t = 1, 2, \cdots$, are i.i.d. noise samples with mean $E\varepsilon(t) = 0$ and variance $Var\varepsilon(t) = \sigma_j^2$. We consider the case that $\varepsilon(t)$ is from Gaussian distribution although the results are easily to be extended to any member of the exponential family distributions. Thus, we have

$$\begin{aligned} & P(\varepsilon(t)|\theta_j, \sigma_j) = P(s(t), b(t), \cdots, b(t - p_j + 1)|\theta_j, \sigma_j) \\ &= \frac{1}{\sqrt{2\pi}\sigma_j} exp\{-\frac{1}{2\sigma_j^2}[s(t) - H_j(b(t), \cdots, b(t - p_j + 1), \theta_j)]^2\}. \end{aligned} \tag{3}$$

Since the switch chooses channel mode j with probability α_j, and the receiving signal $s(t)$ and its corresponding noise $\varepsilon(t)$ can be described by the finite mixture model

$$P(\varepsilon(t)|\Theta) = \sum_{j=1}^{K} \alpha_j P(\varepsilon(t)|\theta_j, \sigma_j), \quad \alpha_j > 0, \quad \sum_{j=1}^{K} \alpha_j = 1. \tag{4}$$

with $P(\varepsilon(t)|\theta_j, \sigma_j)$ being given by eq.(3). Since $\varepsilon(t), t = 1, 2, \cdots$, are i.i.d. samples, the parameters $\Theta = \{\{\theta_j\}_1^K, \{\sigma_j\}_1^K, \{\alpha_j\}_1^K\}$ is estimated by Maximizing the Likelihood(ML):

$$L = \sum_{t=1}^{N} \ln P(\varepsilon(t), \Theta). \tag{5}$$

We use the popular iterative EM algorithm to do this ML. Following Dempster, Lair& Rubin [2] and Jordan & Xu [7] and after some mathematical work, we get

The E-step: Compute

$$h_j^{(k)}(\varepsilon(t)) = P(j|\varepsilon(t)) = \frac{\alpha_j^{(k)} P(\varepsilon(t)|\theta_j^{(k)}, \sigma_j)}{\sum_{i=1}^{K} \alpha_i^{(k)} P(\varepsilon(t)|\theta_i^{(k)}, \sigma_j)},$$

$$Q(\Theta|\Theta^{(k)}) = \sum_{t=1}^{N} \sum_{j=1}^{K} h_j^{(k)}(\varepsilon(t)) \ln[\alpha_j^{(k)} P(\varepsilon(t)|\theta_j^{(k)}, \sigma_j)]. \tag{6}$$

The M-step: Update $\Theta^{(k)}$ into $\Theta^{(k+1)}$ by maximing $Q(\Theta|\Theta^{(k)})$. That is, from $\frac{\partial Q}{\partial \alpha_j} = 0$, $\frac{\partial Q}{\partial \theta_j} = 0$, $\frac{\partial Q}{\partial \sigma_j} = 0$ and $\sum_{j=1}^{K} \alpha_j = 1$, we obtain

$$\begin{aligned}
\alpha_j^{(k+1)} &= \frac{1}{N} \sum_{t=1}^{N} h_j^{(k)}(\varepsilon(t)), \\
\theta_j^{(k+1)} &= (R_j^{(k)})^{-1} C_j^{(k)}, \\
C_j^{(k)} &= \sum_{t=1}^{N} h_j^{(k)}(\varepsilon(t)) s(t) B_{p_j} \\
B_{p_j} &= [1, b(t), \cdots, b(t - p_j + 1)]^T, \\
R_j^{(k)} &= \sum_{t=1}^{N} h_j^{(k)}(\varepsilon(t)) B_{p_j} B_{p_j}^T, \\
(\sigma_j^2)^{(k+1)} &= \frac{1}{\sum_{t=1}^{N} h_j^{(k)}(\varepsilon(t))} \sum_{t=1}^{N} h_j^{(k)}(\varepsilon(t)) \times \\
&\times \quad [s(t) - H_j(b(t), \cdots, b(t - p_j + 1), \theta_j)]^2
\end{aligned} \tag{7}$$

where the updating formula for $\theta_j^{(k+1)}$ holds only when $H_j(.)$ is linear as given by eq.(2). When $H_j(.)$ is nonlinear, we can generalize it by the EIRLS method[7].

However, our main purpose here is not the estimation of these channel parameters. What we want is to get an estimate $\hat{b}(t)$ such that it is as close as possible to the original $b(t)$. As shown in Fig.1, we try to design an equalizer which has also k modes. The change between the modes is controlled also by a random switch which is synchronous to the switch that selects between the channel modes.

During the equalizer mode j is chosen, we have the signal $\hat{b}(t)$ given by

$$\hat{b}(t) = E_j(s(t), \cdots, s(t - p_j + 1), \phi_j). \tag{8}$$

where $E_j(.)$ is the transfer function of the equalizer in mode-j. Particularly, we consider the linear transversal filter as follows

$$\begin{aligned} \hat{b}(t) &= d_{j,0} + \sum_{\tau=1}^{p_j} d_{j,\tau} s(t - \tau + 1), \\ thus\ \phi_j &= [d_{j,0}, d_{j,1}, \cdots, d_{j,p_j}]. \end{aligned} \tag{9}$$

In the sequel, we propose two designs for such a equalizer.

The first design is a technique we called *Weighted Least Squares*. It consists of two stages. In the first stage, we use a training set of samples $\{s(t), b(t), t = 1, \cdots, N\}$ to get the transfer function H_j of each channel mode by eqs.(6) & (7), and then put the obtained parameters of the channel modes into eq.(3). In the second stage, we design each equalizer mode by solving ϕ_j that minimizes the weighted least squares error

$$\sum_{t=1}^{N} h_j(\varepsilon(t))(b(t) - \hat{b}(t))^2 = \sum_{t=1}^{N} h_j(\varepsilon(t))[b(t) - E_j(s(t), \cdots, s(t - p_j + 1), \phi_j)]^2. \tag{10}$$

where $h_j(\varepsilon(t))$ is computed by eq.(6) and eq.(3). Particularly, when E_j is linear as given by eq.(9), we can get

$$\begin{aligned} \phi_j &= R_{e_j}^{-1} C_{e_j}, \\ C_{e_j} &= \sum_{t=1}^{N} h_j(\varepsilon(t)) b(t) S_{p_j}, \\ S_{p_j} &= [1, s(t), \cdots, s(t - p_j + 1)]^T, \\ R_{e_j} &= \sum_{t=1}^{N} h_j(\varepsilon(t)) S_{p_j} S_{p_j}^T. \end{aligned} \tag{11}$$

We can also modify eq.(11) into an adaptive algorithm as follows:

$$\begin{aligned} \phi_j^{(t+1)} &= \bar{R}_{e_j}^{(t+1)} C_{e_j}^{(t+1)}. \\ C_{e_j}^{(t+1)} &= C_{e_j}^{(t)} + \lambda_f h_j(\varepsilon(t)) b(t) S_{p_j}, \end{aligned}$$

$$\bar{R}_{e_j}^{(t+1)} = \bar{R}_{e_j}^{(t)} - \frac{\bar{R}_j^{(t)} S_{p_j} S_{p_j}^T \bar{R}_{e_j}^{(t)}}{\frac{1}{\lambda_f h_j(\varepsilon(t))} + S_{p_j}^T \bar{R}_{e_j}^{(t)} S_{p_j}}. \quad (12)$$

where $0 < \lambda_f < 1$ is a forgetting factor. $\bar{R}e_j$ is the inverted Re_j defined in eq.(11). $\bar{R}e_j(0)$ is initialized by an arbitrary positive definite matrix. Furthermore, we can even use Widrow's simple LMS[10] to replace eq.(12). That is, we update with a learning stepsize $0 < \lambda < 1$.

$$\phi_j^{(t+1)} = \phi_j^{(t)} + \lambda h_j(\varepsilon(t))(b(t) - \hat{b}(t)) S_{p_j}, \quad (13)$$

After training, we use the obtained parameters to computer the switching signal $h_j(\varepsilon(t))$ for every received signal sample point $s(t)$ by

$$h_j(\varepsilon(t)) = \frac{\alpha_j P(\varepsilon(t)|\theta_j, \sigma_j)}{\sum_{i=1}^{K} \alpha_i P(\varepsilon(t)|\theta_i, \sigma_j)}. \quad (14)$$

with which the output of the equalizer mode $j*$ as $\hat{b}(t)$ such that

$$h_{j^*}(\varepsilon(t)) = \max_j h_j(\varepsilon(t)), \quad (15)$$

we call this manner the *hard switching*. An alternative is the *soft switching* as follows

$$\hat{b}(t) = \sum_{j=1}^{K} h_j(\varepsilon(t)) E_j(s(t), \cdots, s(t - p_j + 1), \phi_j). \quad (16)$$

The second design is an one package approach. We do not solve the transfer functions of each channel mode by eq.(6) and eq.(7). Instead, during the switch chooses the equalizer mode j, we assume that

$$b(t) - \hat{b}(t) = b(t) - E_j(s(t), \cdots, s(t - p_j + 1), \phi_j) = \varepsilon(t)$$

for $t = 1, 2, \cdots$, are i.i.d. Gaussian samples with mean $E\varepsilon(t) = 0$ and variance $Var\varepsilon(t) = \sigma_j^2$. That is, we have

$$P(\varepsilon(t)|\phi_j, \sigma_j) = \frac{1}{\sqrt{2\pi}\sigma_j} exp\{-\frac{1}{2\sigma_j^2}[b(t) - E_j(s(t), \cdots, s(t - p_j + 1), \phi_j)]^2\}, \quad (17)$$

By considering all the modes, we see that $\varepsilon(t)$ can again be described by the finite mixture model eq.(4), but now with $P(\varepsilon(t)|\theta_j, \sigma_j)$ being replaced by $P(\varepsilon(t)|\phi_j, \sigma_j)$ given by eq.(17). Moreover, eq.(5) applies too. Therefore, the learning of the parameters of each equalizer mode can be implemented by an EM algorithm with

The E-step: Given by eq.(6) again.

The M-step: Be a slight modification of eq.(7), given by

$$\alpha_j^{(k+1)} = \frac{1}{N} \sum_{t=1}^{N} h_j^{(k)}(\varepsilon(t)),$$

$$
\begin{aligned}
\phi_j^{(k+1)} &= (R_{e_j}^{(k)})^{-1} C_{e_j}^{(k)}, \\
C_{e_j}^{(k)} &= \sum_{t=1}^{N} h_j^{(k)}(\varepsilon(t)) b(t) S_{p_j} \\
S_{p_j} &= [1, s(t), \cdots, s(t-p_j+1)]^T, \\
R_{e_j}^{(k)} &= \sum_{t=1}^{N} h_j^{(k)}(\varepsilon(t)) S_{p_j} S_{p_j}^T, \\
(\sigma_j^2)^{(k+1)} &= \frac{1}{\sum_{t=1}^{N} h_j^{(k)}(\varepsilon(t))} \sum_{t=1}^{N} h_j^{(k)}(\varepsilon(t)) \times \\
&\times [b(t) - E_j(s(t), \cdots, s(t-p_j+1), \phi_j)]^2, \qquad (18)
\end{aligned}
$$

where the updating formula for $\phi_j^{(k+1)}$ holds only when $E_j(.)$ is linear as given by eq.(9). When $H_j(.)$ is nonlinear, we can use the EIRLS method[7].

One adaptive version of eq.(18) is given as follows

$$
\begin{aligned}
h_j(\varepsilon(t)) &= \alpha_j^{(t)} P(\varepsilon(t)|\phi_j^{(t)}) / \sum_{i=1}^{K} \alpha_i^{(t)} P(\varepsilon(t)|\phi_i^{(t)}), \\
\theta_j^{(t+1)} &= \bar{R}_{e_j}^{(t+1)} C_{e_j}^{(t+1)}, \\
C_{e_j}^{(t+1)} &= C_{e_j}^{(t)} + \lambda_f h_j(\varepsilon(t)) b(t) S_{p_j} \\
S_{p_j} &= [1, s(t), \cdots, s(t-p_j+1)]^T, \\
\bar{R}_{e_j}^{(t+1)} &= \bar{R}_{e_j}^{(t)} - \frac{\bar{R}_{e_j}^{(t)} S_{p_j} S_{p_j}^T \bar{R}_{e_j}^{(t)}}{\frac{1}{\lambda_f h_j(\varepsilon(t))} + S_{p_j}^T \bar{R}_{e_j}^{(t)} S_{p_j}}, \\
\lambda_t &= t/(t+1), \quad \alpha_j^{(t+1)} = \lambda_t \alpha_j^{(t)} + \lambda_f (1-\lambda_t) h_j(\varepsilon(t)), \\
S_{h_j}(0) &= 0, \quad S_{h_j}(t+1) = S_{h_j}(t) + h_j(\varepsilon(t)), \\
\lambda_h &= S_{h_j}(t)/S_{h_j}(t+1), \\
(\sigma_j^2)^{(t+1)} &= \lambda_h (\sigma_j^2)^{(t)} + \lambda_f (1-\lambda_h) h_j(\varepsilon(t)) \times \\
&\times [b(t) - E_j(s(t), \cdots, s(t-p_j+1), \phi_j)]^2. \qquad (19)
\end{aligned}
$$

where $\bar{R}e_j$ is the inverted Re_j defined in eq.(18). $\bar{R}e_j(0)$ is initialized by an arbitrary positive definite matrix.

We can get the estimated $\hat{b}(t)$ by either the *hard switching* eq.(15) or the *soft switching* eq.(16) proposed in the first design.

Before closing this section, we should point out that for all the studies in this section, we assume that a set of training samples for $b(t), t = 1, \cdots, N$ are available. In practice, this is possible by either sending a known signal sequence periodically or by using some specific estimation measure. We will not discuss this issue in detail. The reader is referred to Widrow & Stearns[10].

3. COMPUTER EXPERIMENTS

We use the signal shown in Fig.(2a) (top) as the sending signal $b(t)$, which passes through a two-mode channel and becomes a output signal $s(t)$ in the

receiving end, as shown in Fig.(2a) (bottom), with

$$s(t) = \begin{cases} b(t) - 1.98b(t-1) + 0.9b(t-2), & \text{for channel mode-1} \\ b(t) + 1.6b(t-1) + 0.9b(t-2), & \text{for channel mode-2} \end{cases} \quad (20)$$

Both the two channel modes happen in equal probability. The lasting period T of each mode is a uniform random variable from the interval $[30, 130]$. In comparison with the sending signal and receiving signal given in Fig.(2a). We can observe the serious distortions in the receiving signal $s(t)$ due to the filtering of the channel.

We design a two-mode channel equalizer as follows:

$$\hat{b}(t) = \begin{cases} c_{1,1}s(t) + c_{1,2}s(t-1) + c_{1,3}s(t-2), & \text{for equalizer mode-1} \\ c_{2,1}s(t) + c_{2,2}s(t-1) + c_{2,3}s(t-2), & \text{for equalizer mode-2} \end{cases} \quad (21)$$

We use the batch way EM algorithm eq.(18) to train the two-mode channel equalizer. The learning converges about only 8 iterations. The obtained parameters for eq.(21) are

$$[c_{1,1}, c_{1,2}, c_{1,3}] = [-17.7150, 5.5391, 2.4743]$$
$$[c_{2,1}, c_{2,2}, c_{2,3}] = [0.3506, -0.0008, -0.0677].$$

Fig.(2b) is the obtained switch for control the selection of the two equalizer modes. We see that the controlling process for the equalizer mode switch obtained by our approach is almost exactly the same as the unknown controlling process of the channel mode switch. This synchronous switch makes the two-mode equalizer recover the sending signal very nicely. Fig.(3a) gives a comparison of the recovered signal $\hat{b}(t)$ with the sending signal $b(t)$. We can see that the recovery is very good except some very small distortions.

To get some further understanding on the advantage of the proposed channel equalizer. We also use the conventional adaptive LMS channel equalizer on the same problem. The equalizer is also of order-3:

$$\hat{b}(t) = c_1 s(t) + c_2 s(t-1) + c_3 s(t-2), \quad (22)$$

Fig.(2d) are the results of the LMS channel equalizer. The LMS learning converges after about 5000 steps to $[c_1, c_2, c_3] = [0.1111, 0.3532, -0.1892]$. However, it follows from Fig.(2d) that the recovered signal $\hat{b}(t)$ by the LMS channel equalizer is very poor, refer to Fig.(3b) for a comparison.

4. CONCLUSION

Based on the model of finite mixtures and the EM learning algorithm, we have proposed new approach for channel equalization. In comparison with the standard adaptive LMS channel equalizer, The experiments have shown a significant improvement obtained by the proposed approach for the complicated communication channel that varies between different modes.

REFERENCES

[1] S. Amari, "Information geometry of the EM and em algorithms for neural networks", *Neural Networks*, in press.

[2] A.P.Dempster, N.M.Laird, and D.B. Rubin, "Maximum- likelihood from incomplete data via the EM algorithm", *J. of Royal Statistical Society, B39*, pp1-38, 1977.

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

[4] Z.Ghahramani and M.I. Jordan, "Function approximation via density estimation using the EM approach", *Advances in NIPS 6*, eds., Cowan, J.D., Tesauro, G., and Alspector, J., San Mateo: Morgan Kaufmann Pub., CA, 1994, 120-127.

[5] R.A.Jacobs, M.I.Jordan, S.J.Nowlan and G.E.Hinton, "Adaptive mixtures of local experts", *Neural Computation, 3*, pp 79-87, 1991.

[6] M.I.Jordan & R.A.Jacobs, "Hierarchical mixtures of experts and the EM algorithm", *Neural Computation 6*, 181-214, 1994.

[7] M.I.Jordan & L.Xu, "Convergence results for the EM approach to mixtures-of-experts architectures", *Neural Networks*, in press.

[8] R. N.Neal & G. E.Hinton, " A new view of the EM algorithm that justifies incremental and other variants", to appear.

[9] V.Tresp, S. Ahmad, and R. Neuneier,"Training neural networks with deficient data", *Advances in NIPS 6*, eds., Cowan, J.D., Tesauro, G., and Alspector, J., San Mateo: Morgan Kaufmann Pub., CA, 1994, 128-135.

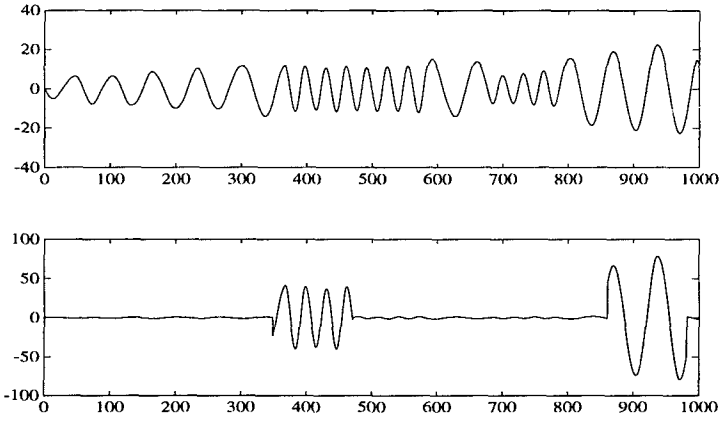
[10] B. Widrow and S.D. Stearns, *Adaptive Signal Processing*, Prentice-Hall, Inc., 1985.

[11] L.Xu and M.I.Jordan, "On Convergence Properties of the EM Algorithm for Gaussian Mixtures", *Neural Computation*, in press.

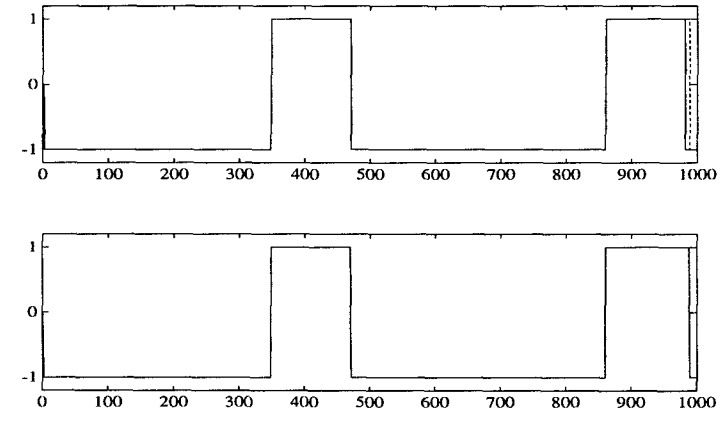
[12] L.Xu and M.I.Jordan, "EM Learning on A Generalized Finite Mixture Model for Combining Multiple Classifiers", Proc. of WCNN'93, Portland, OR, Vol. IV, 227-230, 1993.

[13] L.Xu , M.I.Jordan and G.E.Hinton, "A Modified Gating Networks for the Mixtures of Experts Architecture", *Proc. of WCNN'94*, SanDiego, Vol. 2, pp405-410, 1994.

[14] A. L.Yuille, P. Stolorz, & J. Utans, "Statistical Physics, Mixtures of Distributions and the EM Algorithm", *Neural Computation 6*, 334-340, 1994.

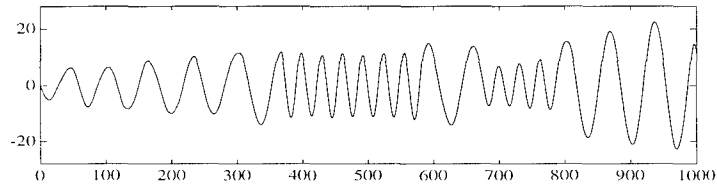


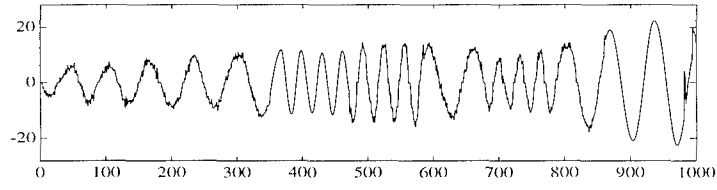
(a)



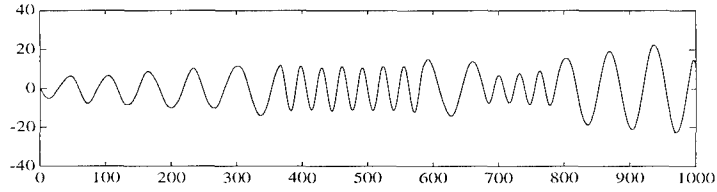
(b)

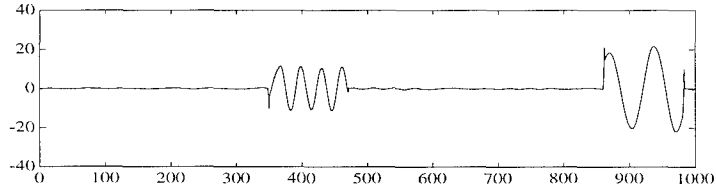
Figure 2: The experiments for channel equalization. (a) The sending signal $b(t)$ (top) and the receiving signal $s(t)$ (bottom). This $b(t)$ passes through a channel with two modes that randomly engage in. After filtered by this channel, the receiving signal $s(t)$ becomes considerably distorted. (b) The original switch control for the selection of the two equalizer modes (top), and the estimated switch control for the selection of the two equalizer modes (bottom).





(c)





(d)

Figure 3: The experiments for channel equalization. (c) The estimated signal $\hat{b}(t)$ (bottom) after channel equalization by our proposed approach in comparison with the sending signal $b(t)$ (top). (d) The estimated signal $\hat{b}(t)$ (bottom) after the channel equalization by the standard adaptive LMS approach in comparison with the sending signal $b(t)$ (top).

Comparison of a Neural Network based receiver to the Optimal and Multistage CDMA Multiuser Detectors

George I. Kechriotis† Elias S. Manolakos ‡[1]

† THINKING MACHINES INC., 245 First Street, Cambridge, MA 02142

‡ COMMUNICATIONS AND DIGITAL SIGNAL PROCESSING (CDSP)
Center for Research and Graduate Studies
Electrical and Computer Engineering Department
Northeastern University, Boston, MA 02115

Abstract – In this paper we compare the Hopfield Neural Network (HNN) based CDMA Multiuser Detection introduced in [1, 2] to the Optimal Multiuser Detector (OMD) [3] and the suboptimal Multistage Detector (MSD) [4]. It is shown that the HNN receiver is a powerful generalization of the MSD. Moreover, it is shown that by fine tuning the parameters of the HNN it is possible to achieve BER performance that exceeds that of the MSD and gets very close to the optimal.

1. Introduction

Spread Spectrum Code-Division-Multiple-Access (CDMA) is an emerging standard for high-bandwidth wireless communications. With the CDMA technique a number of active simultaneously transmitting users share a Gaussian channel by modulating different *signature* waveforms. At a receiver end, the incoming signal is the superposition of *all* transmitted signals and Gaussian channel noise. In many cases (e.g. hubs, satellite networks) a *centralized* receiver is required to resolve *all* active users sharing the channel. The conventional *multi-user* detector for this case is a generalization of the single-user detector, i.e. it consists of a bank of filters, matched to the signature waveforms of each individual transmitter, followed by a set of thresholding devices. The major inherent limitation of the conventional multi-user detector is the so called *near-far* problem; i.e. when the power contributions of the individual transmitters to the overall received signal are very dissimilar (e.g. some users are stronger than others or distant) the Bit-Error-Rate (BER) performance at the receiver degrades severely. To overcome this limitation a number of approaches have been proposed. Power-control, in which the transmitters are adjusting their power so that at the receiver-end their contributions are approximately equal, is the one most frequently used. However, when more than one receivers are present

[1] This work is partially supported by a grant from the Advanced Research Project Agency of the Depatment of Defense, under contract MDA-972-93-1-0023. E-mail: elias@cdsp.neu.edu, Web: http://www.cdsp.neu.edu/info/manolakos.html

0-7803-2739-X/95 $4.00 © 1995 IEEE

(as in the case of a wireless network) power-control is difficult to implement and alternative methods have to be found.

It has been proven by Verdu et al. [3] that the problem of *Optimal Multi-user Detection* can be formulated as a quadratic integer optimization problem that is however NP-hard. Many suboptimal solutions with good near-far resistance properties have therefore been proposed, including the *decorrelating* detector and the *Multistage* detector [4]. Recently, the interest in neural-network based solutions to the Optimal Multiuser CDMA detection problem has grown considerably [1, 2, 5, 6, 7]. In [6] it has been shown that multi-layer perceptrons can approximate more accurately than other signal processing schemes the decision region of the optimal detector. Furthermore, with the recent advances in analog VLSI technology, practical analog neural network based receivers may soon become available and deliver demodulation times per symbol that are much smaller than those of today's digital microprocessor based implementations.

In this paper we show that the recently proposed by the authors [1, 2, 5, 7] Hopfield recurrent Neural Network (HNN) based multi-user detector is indeed a powerful generalization of a well known and extensively studied Multistage detector (MSD). We show that an infinite number of stages MSD is functionally equivalent to a particular instance of an HNN. Furthermore we show that using the additional design flexibility that the HNN approach offers leads to better BER performance. Taking into account that the HNN detector may be complemented by powerful signal pre-processing stages [2] that can reduce the network's size and computational cost per bit demodulation and the BER, we believe that recurrent neural network structures may provide a new framework for the design of high speed and performance CDMA multiuser receivers.

2. Hopfield Neural Networks for CDMA Multiuser Detection

Assume that K active transmitters share the same Gaussian channel at a given time instance. A signature waveform $s_k(t)$, time limited in the interval $t \in [0,\ T]$, is assigned to each transmitter. Let us denote the i^{th} information bit of the k^{th} user as $b_k^{(i)} \in \{+1,\ -1\}$. In a general CDMA system, the signal at a receiver is the superposition of K transmitted signals and additive channel noise:

$$r(t) = \sum_{i=-P}^{P} \sum_{k=1}^{K} b_k^{(i)} s_k(t - iT - \tau_k) + n(t), \quad t \in \mathcal{R} \tag{1}$$

In (1) $\tau_k \in [0,\ T)$ denote the relative time delays between the users, and $2P+1$ the packet size. In case that the stations cooperate to maintain synchronism, it holds that $\tau_k = 0, \quad k = 1, \ldots, K$. As we mentioned in the previous section, the Conventional Detector (CD) consists of a bank of filters matched to the signature waveforms of each user, and a simple thresholding

device that produces an estimate $\hat{b}_k^{(i)}$ for the i^{th} information bit of the k^{th} user according to the sign of the i^{th} output of the k^{th} matched filter:

$$\begin{array}{lcl} y_k^{(i)} & = & \int_{iT-\tau_k}^{(i+1)T-\tau_k} r(t)s_k(t-iT-\tau_k)dt \\ \mathbf{b}_{CD}^{(i)} & = & sign(\mathbf{y}^{(i)}) \end{array} \quad (2)$$

where $\mathbf{y}^{(i)} = [y_0^{(i)} \;\; y_1^{(i)} \;\; \ldots y_{K-1}^{(i)}]^T$.

On the other hand, the *Optimal Multiuser Detector* (OMD) produces an estimate for the information vector transmitted at the discrete time instant i, based on the maximization of the logarithm of the likelihood function. In the *synchronous* CDMA transmission case it holds [3]:

$$\mathbf{b}_{OMD}^{(i)} = \arg \max_{\mathbf{b}\in\{+1,\,-1\}^K} \{2\mathbf{y}^{(i)^T}\mathbf{b} - \mathbf{b}^T\mathbf{H}\mathbf{b}\} \quad (3)$$

where $\mathbf{H} \in \mathcal{R}^{\mathcal{K}\times\mathcal{K}}$ is the matrix of signal waveform cross-correlations.

Because of the exponential growth of the computational complexity of the OMD with the number of active users, suboptimal multi-user detection schemes have been proposed. The *Multistage Detector* (MSD) introduced by Varanasi and Aazhang [4] is such a scheme having a number of attractive properties including relatively low computational complexity, near-optimal BER performance, and insensitivity to near-far problems. The MSD consists of a collection of cascaded stages $m = 1, 2, \ldots$ each producing an estimate $\mathbf{b}_{MSD}^{(i)}(m)$ as follows :

$$\mathbf{b}_{MSD}^{(i)}(m+1) = sign\left(\mathbf{y}^{(i)} - (\mathbf{H}-\mathbf{E})\mathbf{b}_{MSD}^{(i)}(m)\right) \quad (4)$$

where $\mathbf{E}$ is a diagonal matrix with elements $e_{ii} = \int_0^T s_i^2 dt$ (signal energies). The output of the first stage ($m = 1$) is initialized to the estimate of the CD. In the next section we will show that an MSD with an infinite number of stages may converges to a *local minimum* of the OMD objective function.

Hopfield Neural Networks (HNN) [8], are single layer networks with output feedback consisting of simple processors (neurons) that can collectively provide good solutions to difficult optimization problems. A connection between two processors in established through a conductance T_{ij} which transforms the voltage outputs of neuron j to a current input for neuron i. Externally supplied bias currents I_i are also present in every processor i. Overall, neuron i receives a weighted sum of the activations all other neurons in the network, and updates its activation according to the rule:

$$V_i = g(U_i) = g(\sum_{j\neq i} T_{ij}V_j + I_i) \quad (5)$$

where $g(U_i)$ can be either a binary or antipodal thresholding function. For the case of the McCulloch-Pitts neurons $V_i = g(U_i) = sign(U_i)$ or any

monotonically increasing nonlinear function. One example of such a nonlinear function often used in simulations is the *sigmoid* function:

$$V_i = g(U_i) = sigm(\alpha U_i) = \frac{1 - e^{-\alpha U_i}}{1 + e^{-\alpha U_i}} \tag{6}$$

where α is a positive constant that controls the slope of the nonlinearity. In particular, when $\alpha \rightarrow \infty$, then $g(U_i) \rightarrow sign(U_i)$.

It has been shown by Hopfield that in the case of symmetric connections ($T_{ij} = T_{ji}$), the equations of motion for the activation of the neurons of a HNN *always* lead to convergence to a *stable state*, in which the output voltages of all the neurons remain constant. Also, when the diagonal elements (T_{ii}) are zero and the width of the amplifier gain curve is narrow, (i.e. the nonlinear activation function $g(\cdot)$ approaches the antipodal thresholding function), the stable states of a network with N neuron units are the *local* minima of the quantity (*energy function*):

$$E = -\frac{1}{2}\sum_{i=1}^{K}\sum_{j=1}^{K} T_{ij} V_i V_j - \sum_{i=1}^{K} V_i I_i \tag{7}$$

The *equations of motion* for the i^{th} neuron may be described in terms of the energy function (7) as follows:

$$\frac{dU_i}{dt} = -\frac{\partial E}{\partial V_i} - \frac{U_i}{\tau} = -\frac{U_i}{\tau} + \sum_{i \neq j} T_{ij} V_j + I_i \tag{8}$$

where $\tau = RC$ is the *time constant* of the RC circuit connected to neuron i. With the exception of pathological cases, when the matrix $\mathbf{T}$ is negative or positive definite, networks with vanishing diagonal elements have minima *only* at the *corners* of the K-dimensional hypercube $[-1 \quad +1]^K$.

It is apparent from Eqn. (3) that the OMD objective function has a very similar form with an HNN energy function. The cross-correlation matrix $\mathbf{H}$ is symmetric since $h_{kl} = \int_0^T s_k(t)s_l(t)dt = \int_0^T s_l(t)s_k(t)dt = h_{lk}$. Moreover, Eqn. (3) can be rewritten as:

$$\begin{aligned} \mathbf{b}_{OMD}^{(i)} &= \arg\min_{\mathbf{b}\in\{+1,\,-1\}^K}\{-\mathbf{y}^{(i)^T}\mathbf{b} + \tfrac{1}{2}\mathbf{b}^T\mathbf{H}\mathbf{b}\} = \\ &= \arg\min_{\mathbf{b}\in\{+1,\,-1\}^K}\{-\mathbf{y}^{(i)^T}\mathbf{b} + \tfrac{1}{2}\mathbf{b}^T(\mathbf{H}-\mathbf{E})\mathbf{b}\} \end{aligned} \tag{9}$$

since $\mathbf{b}^T\mathbf{E}\mathbf{b}$ is always a positive number. The matrix $-(\mathbf{H}-\mathbf{E})$ is symmetric, and has zero diagonal elements since $h_{ii} = \int_0^T s_i^2 dt = e_{ii}$. Therefore, the OMD objective function can be directly translated into the energy function of a Hopfield Neural Network (7)) with weight matrix $\mathbf{T} = -(\mathbf{H} - \mathbf{E})$ and biases $\mathbf{I} = \mathbf{y}^{(i)}$.

The obvious advantage of the HNN detector over other suboptimal receivers lies in its fast convergence. The main part of the receiver can be implemented by relatively simple analog VLSI hardware with convergence times in the order of a few nanoseconds, while other detectors implemented using digital microprocessors or ASICs are much slower and consume a lot more power [9].

3. Relation of the HNN to the Optimal and Multistage Detectors

High speed and low power consumption are not the only advantages of the HNN detector. We will show that the well-known Multistage Detector (MSD) is in fact a special case of a discrete-time approximation of the proposed HNN detector. In particular, under certain conditions, the HNN detector corresponds to an infinite number of stages MSD, thus sharing the same near-far resistance and near-optimal performance characteristics. Moreover, the additional flexibility that the HNN detector allows for, in terms of controlling the RC constants of the analog circuits and the slopes of the nonlinearities, facilitates the design of HNN receivers capable of delivering near-optimal performance in regions determined by the relative energies and cross-correlation values of the user signature waveforms.

Consider the discrete-time approximation of Eqn. (8) that describes the equations of motion of the i^{th} neuron of the HNN: $\Delta U_i = -\frac{U_i}{\tau} + \sum_{i \neq j} T_{ij} V_j + I_i$. If the activation function of the neurons is the $sign(\cdot)$ function, the dynamics of the i^{th} neuron at the discrete-time instant $t = m+1$, are described by the following equation:

$$\begin{aligned} V_i(m+1) &= g(U_i(m+1)) = sign(U_i(m) + \Delta U_i(m)) = \\ &= sign(U_i(m) - \tfrac{U_i(m)}{\tau} + \textstyle\sum_{i \neq j} T_{ij} V_j(m) + I_i) \end{aligned} \tag{10}$$

By setting $\tau = 1$ and substituting in Eqn. (10) for the values of $\mathbf{T}$ and $\mathbf{I}$ for the proposed HNN detector, (10) becomes
$V_i(m+1) = sign(y_i - \sum_{i \neq j} h_{ij} V_j(m))$, which can be written in matrix form as

$$\mathbf{V}(m+1) = sign\left(\mathbf{y} - (\mathbf{H} - \mathbf{E})\mathbf{V}(m)\right) \tag{11}$$

Now by comparing Eqn. (11) and Eqn. (5) we see that for the special case in which the RC constant of the HNN circuit is equal to $\tau = 1$ and $g(\cdot) = sign(\cdot)$, the estimate of the $(m+1)^{th}$ stage of a MSD coincides with the output of a discrete-time approximation of this HNN at time instant $t = m+1$. Moreover, since the update of the estimate of each MSD stage is being performed synchronously, an infinite number of stages MSD is essentially equivalent to a discrete-time HNN operating in synchronous (fully parallel) updating mode, which may lead to limit cycles of length 2 as shown by Bruck [10]. An analog HNN with continuous updating, or an asynchronously updated discrete-time HNN, does not exhibit oscillations, and thus in principle it can outperform even an infinite number of stages MSD. In the sequel we will demonstrate that the HNN detector can be designed to outperform the MSD, and that its performance can be further improved and approach the optimal by selecting the appropriate values for the RC constant and the nonlinearity slope of the neuron units.

4. Comparison - Regions of Failure

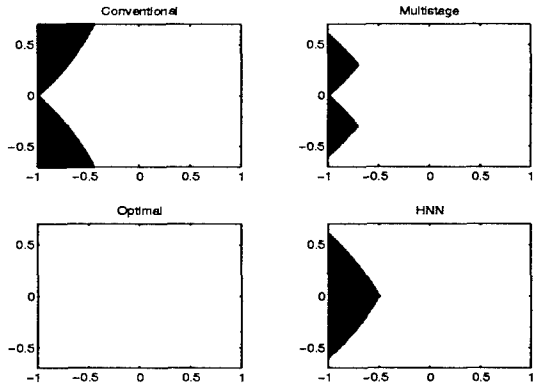


Figure 1: Regions of Failure (dark regions) for CD, 2-stage MSD, OMD, and randomly initialized HNN: Vertical axis = log_{10}(near-far-ratio); Horizontal axis = normalized signature waveforms cross-correlation.

Let us first consider a $K = 2$ synchronous CDMA users noiseless case. Assume that at a given time instant the symbols transmitted by the two users are $\mathbf{b}_{sent} = [b_0 \;\; b_1]^T$, and that the energy of the second user remains constant while the energy of the first user is varying. The cross correlation matrix can be expressed as:

$$\begin{aligned} \mathbf{H} &= \mathbf{E}^{1/2}\mathbf{H}_{norm}\mathbf{E}^{1/2} = \\ &= \begin{bmatrix} e^{1/2} & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & h \\ h & 1 \end{bmatrix} \cdot \begin{bmatrix} e^{1/2} & 0 \\ 0 & 1 \end{bmatrix} \end{aligned} \tag{12}$$

where, $h \in (-1, 1)$ is the normalized cross-correlation of the users signature waveforms and $e \in \mathcal{R}$ is the near-far-ratio which coincides with the energy of the first user. In the noiseless case, the sampled outputs of the matched filter-bank can be expressed as $\mathbf{y} = \mathbf{H} \cdot \mathbf{b}_{sent}$. By evaluating the outputs of a multiuser detector for various values of h and e and comparing them with the originally transmitted symbols $\mathbf{b}_{sent}$, we can estimate its *Region Of Failure* (ROF). In Figure 1 we compare the Conventional (CD), 2-stage MSD, OMD, with a *randomly* initialized HNN detector by plotting their ROFs (dark regions) assuming that the originally transmitted information vector was $\mathbf{b}_{send} = [b_0 \;\; b_1]^T = [-1 \;\; -1]^T$. In each of the plots, the vertical axis corresponds to the decimal logarithm of the near-far ratio ($log_{10}\, e$), and the horizontal to the value of the cross-correlation h. For each point in the ROF, $\mathbf{b}_{detected} \neq \mathbf{b}_{sent}$.

As we can see from Figure 1, for heavily correlated signature waveforms the CD exhibits the worst performance producing the correct estimate only when the energies of the users are close to each other. The 2-stage MSD performs well when one of the users is much stronger than the other (since the strong user can be estimated more accurately), but fails in a region where the energy of one user is larger than that of the other but not enough

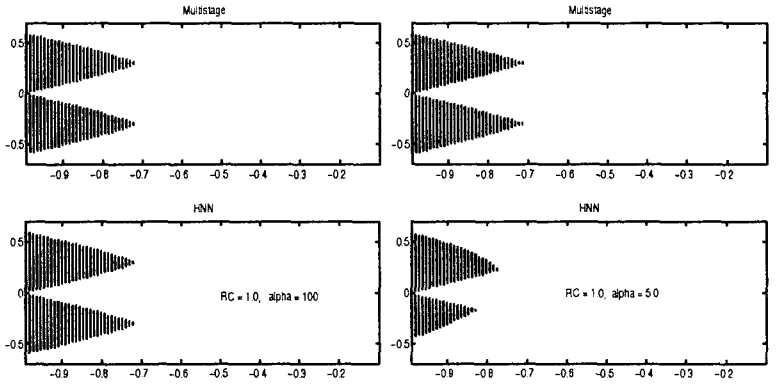


Figure 2: Regions of Failure of the HNN detector initialized with the CD estimates $\mathbf{U}(0) = \mathbf{y}$. *Left:* $\alpha = 100$, $\tau = RC = 1$. ROF of HNN almost same as for the 2-stage MSD with the same initialization. *Right:* $\alpha = 5.0$, $\tau = RC = 1$. ROF of HNN becomes smaller when using a smoother non-linearity.

to allow its correct estimation. For the noiseless case and range of h and e shown in Figure 1, the OMD never fails to produce the correct estimate for the transmitted symbols. The *randomly* initialized HNN exhibits a much larger ROF than the 2-stage MSD. In particular, since a randomly initialized HNN is guaranteed to converge only to a local minimum of the OMD's objective function, its ROF corresponds to objective functions with either two local minima or a unique global minimum that is however different from the originally transmitted symbols.

However, as we have shown in the previous section, an HNN detector initialized with the outputs of the CD (matched filter bank) $\mathbf{y}$ corresponds, under certain conditions, to an infinite-number-of-stages long MSD. The particular conditions, are that the neurons' nonlinearity is steep (α is large in Eqn. (6)) such that it approximates the $sign(\cdot)$ function, and that the $\tau = RC$ constant of the neuron units is equal to one. In Figure 2(left) we plot the ROFs for such an HNN detector with $\tau = RC = 1.0$ and $\alpha = 100$ and we compare it to the ROF of the 2-stage MSD. As we can see, the ROF of the HNN detector roughly coincides with that of the 2-stage MSD. On the other hand, from the right part of the same Figure we can see that the ROF of the HNN can be made smaller by making the nonlinearity of the neurons smoother ($\alpha = 5.0, \tau = 1.0$). Finally, in Figure 3 we plot the ROF of the HNN for $\alpha = 5.0$ and several values of the RC constant. It is evident from Figure 3, that by increasing slightly the value of τ, the ROF of the HNN can be made much smaller than that of the MSD, and can approach that of the OMD.

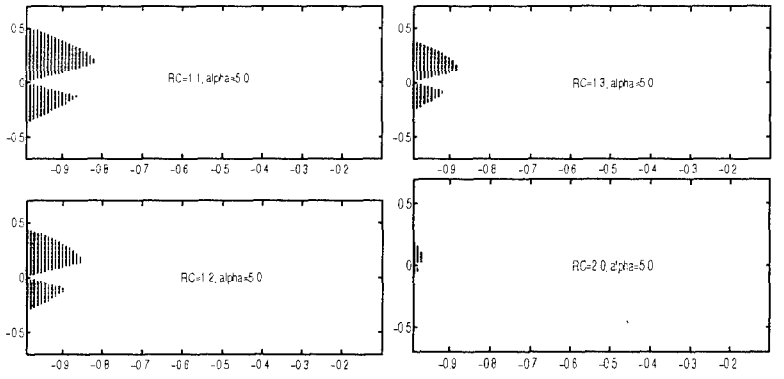


Figure 3: HNN detector with $g(x) = sigm(5.0 \cdot x)$, and $\mathbf{U}(0) = \mathbf{y}$: Regions of Failure versus the value of the $\tau = RC$ for fixed $\alpha = 5.0$. As RC increases, the HNN's ROF approaches that of the OMD.

5. Experimental Results

In Figure 4, we compare the CD, 2-stage MSD, HNN and OMD detectors by evaluating their error probabilities in a 2 users' synchronous CDMA case with noise. The decimal logarithm of the Bit-Error-Rates (BER) are plotted versus the value of the near-far-ratio (e_2/e_1 in decibel) for $h = 0.7$. The Signal-to-Noise-Ratio of user 1 (SNR_1) is fixed at 8dB, and the BER of user 1 is being evaluated. The 2-stage MSD detector exhibits worst performance when the energies of the two users are not very dissimilar, as expected, whereas the HNN's and OMD's performance are practically indistinguishable.

In another experiment $K = 5$ synchronous users employ spreading of length $L = 4$ derived according to the binary representation of the identifying user index with the addition of a leading $+1$. The weak users 2,4,5 transmit with unit normalized energy, whereas the energy of user 1 is 30 times larger and the energy of user 3 is 10 times larger than that of the other users. In Table 1 we show the *cumulative* (calculated over all the active users) BER that the CD, 10-stage MSD, HNN and Optimal multi-user detectors achieve. The HNN detector used had parameters $\alpha = 1.0$ and $\tau = 1.0$. In the same setting we investigated how varying the time-constant of the HNN circuit (τ) and the slope of the HNN neuron's nonlinearity (α) affects the BER performance. The same set of $K = 5$ active users as above was used for this experiment. First we evaluated the performance variations due to the slope α changes for fixed $\tau = 1.0$ and $\text{SNR}_2 = 8$ db (w.r.t. the weak users). Then we evaluated the BER performance for fixed $\alpha = 1.0$ and varying τ. The results are summarized in Table 2.

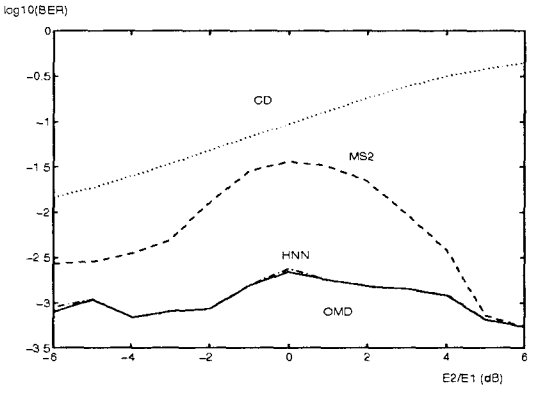


Figure 4: Error probability comparison of the CD, 2-stage MSD, HNN and OMD detectors for $K = 2$, $h = 0.7$ and $\text{SNR}_1 = 8$ dB, vs. the ratio e_2/e_1. HNN parameters: $\alpha = 5.0$, $RC = 2.5$.

SNR	CD	MSD-10	HNN	OMD
7	-0.717526	-1.52563	-1.78888	-1.90427
8	-0.723444	-1.69699	-1.92942	-2.38535
9	-0.728022	-1.78329	-2.34044	-2.56229
10	-0.745812	-1.84727	-2.52396	-3.30211
11	-0.751903	-1.89566	-2.81458	-3.41664

Table 1: Comparison of CD, MSD-10, HNN and OMD detectors. $K = 5, L = 4, E_1 = 30, E_3 = 10, E_2 = E_4 = E_5 = 1$. Cumulative BER vs. SNR (w.r.t. the weak users).

α	BER
1	-1.92942
5	-2.26717
10	-2.31641
20	-1.93785
25	-1.74430

$\tau = RC$	BER
.6	-1.80811
.9	-1.91424
1	-1.92942
10	-2.27184
100	-1.84510

Table 2: $K = 5$ synchronous CDMA users; *Left:* BER vs. α of the HNN nonlinearity, for $\tau = 1.0$; *Right:* BER vs. $\tau = RC$ of HNN, for $\alpha = 1.0$.

6. Conclusions

The relation of HNN based receivers to the Optimal (but infeasible) CDMA Multiuser Detector (OMD) and the well-known and extensively studied sub-optimal Multistage Detector (MSD) was investigated. There is range for the slope of the neuron nonlinearities (α) where the BER performance of the HNN approaches that of the optimal detector. This was demonstrated for the 2-user case by comparing the corresponding Regions-Of-Failure and has been verified for a larger number of users by simulations. In a similar fashion, the value of the $\tau = RC$ constant of the neural circuit affects the performance of the HNN. It has been shown that for a particular set of values for parameters α and $\tau = RC$, a synchronously updated discrete-time HNN is equivalent to an infinite-number-of-stages long MSD. In addition, the design and engineering flexibility that the HNN approach offers as well as the almost instantaneous convergence when implemented in analog VLSI hardware, makes it attractive for real time multi-user demodulation.

References

[1] G. Kechriotis and E. S. Manolakos. Implementing the Optimal CDMA Multiuser Detector with Hopfield Neural Networks. *Int'l Workshop on Applications of Neural Networks to Telecomm.*, pp. 60-67, 10/1993.

[2] G. Kechriotis and E. S. Manolakos. A Hybrid digital computer–Hopfield Neural Network Spread-Spectrum CDMA detector for Real-Time Multiuser Demodulation. *1994 IEEE Workshop on Neural Networks for Signal Processing IV*, pp. 545-554, 9/1994.

[3] S. Verdu. Computational Complexity of Optimum Multiuser Detection. *Algorithmica*, 4:303–312, 1989.

[4] M. K. Varanasi and B. Aazhang. Multistage Detection in Asynchronous Code-Division Multiple Access Communications. *IEEE Trans. on Comm.*, 38:509–519, April 1990.

[5] G. Kechriotis and E. S. Manolakos. Hopfield Neural Network Implementation of the Optimal Multiuser CDMA Detector. *IEEE Trans. on Neural Networks*, 1995, to appear.

[6] B.-P. Paris B. Aazhang and G. Orsak. Neural Networks for Multi-user Detection in CDMA Communication. *IEEE Trans. on Comm.*, 40:1212–1222, July 1992.

[7] G. Kechriotis. *Feedback Neural Networks in Digital Communications: Algorithms, Architectures and Applications.* PhD thesis, Northeastern University, 8/1994.

[8] J. J. Hopfield. Neural Networks and Physical Systems with Emerging Collective Computational Abilities. *Proc. Natl. Acad. Sci. USA*, 79:2554–2558, 1982.

[9] A. Jayakumar and J. Alspector. An Analog Neural-Network Co-processor System for Rapid Prototyping of Telecommunications Applications. *Int. Workshop on Appl. of Neural Networks to Telecomm.*, pp. 13-19, 10/1993.

[10] J. Bruck. On the Convergence Properties of the Hopfield Model. *Proceedings of the IEEE*, 78:1579–1585, October 1990.

A
Adali, T. 541
Adelson, E.H. 293
Alexopoulos, V.N. 407
Amaldi, E. 303
Anderson, C. W. 475
Andreou, A.G. 581
Andrianasy, F. 371

B
Bacchiani, M. 67
Bachmann, C. M. 361
Back, A.D. 191
Bajaria, P. M. 494
Birkett, A.N. 449
Blekas, K. 153
Bourbakis, N.G. 504

C
Campa, A. 181,
Capobianco, E. 87
Carrato, S. 388,
Cauwenberghs, G. 581
Chan, N. 201
Chen, S. 593
Chng, E-S 593,
Clothiaux, E. E. 361
Colla, A. M. 571
Cunningham, R. K. 351

D
Del Giudice, P. 181
Devulapalli, S.V. 475
Drzewiecki, K.T. 484

E
Engel, R.R. 465

F
Fang, M. 323
Farrell, K.R. 243
Furlanello, C. 233

G
Gersho, A 58
Ghosh, J. 135
Gibson, G. J. 551, 593
Girosi, F. 201
Giuliani, D. 233
Goubran, R.A. 449
Grossberg, S. 313

H
Hanes, D M. 380
Hansen, L.K. 30, 439, 484
Hassell Sweatman, C.Z.W. 551
Hintz-Madsen, M. 484
Holm, S. 439
Hood, C. S. 521
Hu, Y.H. 459,
Hwang, J-N 253

J
Ji, C. 521

K
Kadirkamanathan, V. 171
Kangas, J. 343
Kasper, C. 272
Katagiri, S. 48, 77
Kathmann, N. 465
Kechriotis, G. 613
Khachikyan, L. 21
Khan, S. A. 561
Kollias, S.D. 407
Kumar, N. 581
Kung, S.Y. 323, 337

L
Larsen, J. 30, 484
Lee, J. 380
Li, Q. 125
Likas, A. 153
Lin, L-J 337

Lin, S-H 323, 337
Liou, H-S 213
Liu, X. 541
Luong, D.Q. 361

M
Ma, K. 223
Madisetti, V. K. 561
Makhoul, J. 263
Mammone, R. 213, 282
Manolakos, E. S. 613
Manry, M. 105
Marsi, S. 388,
Martinez , D. 31
Matsugu, M. 11
Mattavelli, M. 303
Mhaskar, H.N. 21
Milgram, M. 371
Miller, D. 58
Mingolla, E. 313
Moon, S. 253
Moore, J. W. 361
Morch, N. 439
Morgan, N. 223
Mulgrew, B. 551, 593

N
Nadal, J-P 181
Niyogi, P. 40

O
Olesen, E. 484
Opris, I. 162

P
Paliwal, K.K. 67
Palreddy, S. 459
Parga, N. 181
Paulson, O. 439
Picard, R.W. 417
Poggio, T. 398
Poon, C-S 115
Principe, J. C. 11, 380

R
Rao, A. 58,
Reininger, H. 272
Ressler, E. K. 427
Rose, K 58
Ruda, H. 513

S
Sagisaka, Y. 67
Schwartz, R. 263
Segee, B. E. 494
Sharma, M. 282
Sherstinsky, A. 417
Shoop, B. L. 427
Snorasson, M. 513
Sonmez, K. 541
Sroka, J. 263
Stafylopatis, A. 153
Stiles, B. W. 135
Stolc, E. A. 475
Sung, K-K 40, 398
Svarer, C. 439

T
Tacsillo, A. 504
Tacsillo, M. 504
Tigges, P. K. 465
Toda, N. 145
Tompkins, W. J. 459
Tresp, V. 1
Trogu, L. 571
Tsoi, A.C. 191
Tufts, D.W. 125

U
Usui, S. 145

V
Van Hulle, M. M. 95

W
Wang, L. 11
Watanabe, H. 48, 77
Waxman, A. M. 351
Weiss, Y. 293
Williamson, J. 313
Wolf, D. 272
Wust, H. 272

X
Xu, L. 603

Y
Yamaguchi, T. 48
Yang, W. 531

Z
Zhao, Y. 263
Zunino, R. 571