MSL-96

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

Proceedings of the Third International Workshop on

Multistrategy Learning

May 23-25
Harpers Ferry, WV

Edited by Ryszard S. Michalski and Janusz Wnek

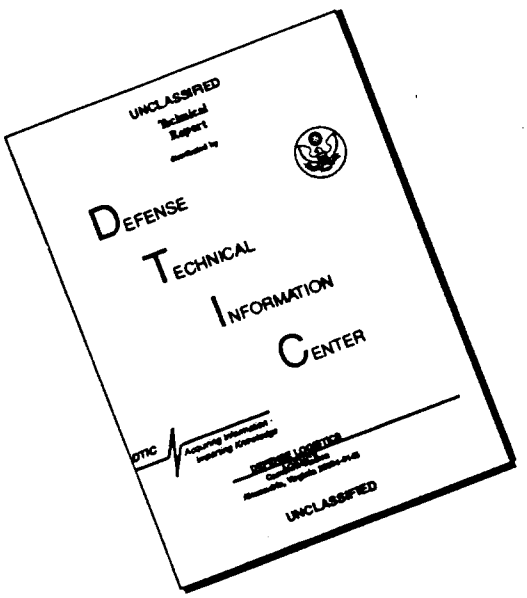
Organized by the
Machine Learning and Inference Laboratory
George Mason University with the collaboration of
International Joint Conferences on Artificial Intelligence, Inc.

Sponsored by the
National Science Foundation
Office of Naval Research
American Association for Artificial Intelligence

19961023 258

DTIC QUALITY INSPECTED

DISCLAIMER NOTICE



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

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 16, 1996	Final 05/01/96 - 07/31/96

4. TITLE AND SUBTITLE

Third International Workship on Multistrategy Learning

5. FUNDING NUMBERS

N 00014-96-1-0859

6. AUTHOR(S)

Ryszard S. Michalski, Janusz Wnek, (eds.)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

George Mason University
4400 University Drive
Fairfax, VA 22030

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
800 North Quincy Street
Building - One
Arlington, VA 22217-5660

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

The preprints of the Workshop Proceedings can be obtained from Machine Learning and Inference Laboratory at George Mason University (admin@aic.gmu.edu). The book form workshop proceedings will soon be published by the AAAI Press. A special issue of Machine Learning journal is being prepared that will include updated versions of selected papers.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The Third International Workshop on Multistrategy Learning (MSL-96), held in Harpers Ferry, WV, May 23-25, 1996, attracted leading researchers in this area from Australia, Austria, Belgium, France, Germany, Italy, Japan, New Zealand, Poland, and the United States.

The workshop covered theoretical and empirical issues in the development of learning systems that employ multiple inferential and/or computational strategies. The study of such systems draws upon the achievements in all subareas of machine learning, and constitutes a major new research direction in the field.

Major topics of the workshop included: the study of interrelationships among learning strategies and paradigms, cognitive models of learning processes and their relationships to methods and paradigms of machine learning, the development of multistrategy learning systems, and their practical applications. Among major application areas presented at the workshop were data mining and knowledge discovery in large databases, intelligent text retrieval, flight simulation, robot navigation robot control, planning, stock market analysis, world wide web searches, and molecular biology.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Multistrategy learning, Machine Learning, Inductive inference, Knowledge discovery in databases.	one
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

Proceedings of the Third International Workshop on

MULTISTRATEGY LEARNING (MSL-96)

May 23-25, 1996
Harpers Ferry

Edited by Ryszard S. Michalski and Janusz Wnek

Organized by Machine Learning and Inference Laboratory at George Mason University with the collaboration of International Joint Conferences on Artificial Intelligence, Inc.

Sponsored by the National Science Foundation, Office of Naval Research, and American Association for Artificial Intelligence

Organizing Committee

Ryszard S. Michalski, General Chairman
George Mason University

Janusz Wnek, Program Chairman
George Mason University

Program Committee

John Anderson, *Carnegie Mellon University*
Francesco Bergadano, *University of Torino*
Jaime Carbonell, *Carnegie Mellon University*
Hugo De Garis, *ATR, Kansai Science City*
Luc De Raedt, *Catholic University of Leuven*
Marie desJardains, *SRI International*
Diana Gordon, *Naval Research Laboratory*
Kenneth Haase, *Massachusetts Institute of Technology*
Heedong Ko, *Korea Institute of Technology*
Yves Kodratoff, *University of Paris-South*
Stan Matwin, *University of Ottawa*
Doug Medin, *Northwestern University*
Raymond Mooney, *University of Texas at Austin*
Stephen Muggleton, *Oxford University*
Michael Pazzani, *University of California at Irvine*
Ashwin Ram, *Georgia Institute of Technology*
Lorenza Saitta, *University of Torino*
Claude Sammut, *University of New South Wales*
Jude Shavlik, *University of Wisconsin*
Derek Sleeman, *University of Aberdeen*
Gheorghe Tecuci, *George Mason University and Romanian Academy*

Local Arrangements

Abhay Kasera
George Mason University

Auxiliary Reviewers

Gilles Bisson
Hendrik Blockeel
Eric Bloedorn
Luc Dehaspe
Denise Gurer
Reuven Karni
Ken Kaufman
Mark Maloof
Herve Mignot
Giancarlo Ruffo
Michele Sebag
Haleh Vafaie
Qi Zhang

Additional copies of preprints of MSL-96 Proceedings, as well as MSL-91 and MSL-93 Proceedings, can be obtained from:

Administrative Assistant, Machine Learning and Inference Laboratory, M.S. 4A5
George Mason University, 4400 University Dr, Fairfax, VA 22030-4444, USA
Email: admin@aic.gmu.edu Tel: 703 993-1719 Fax: 703 993-3729

Copies of MSL-96 Proceedings can be obtained from: AAAI Press, P.O. Box 60036, Palo Alto, CA 94306

Copyright ©1996, Machine Learning and Inference Laboratory
All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means - electronic, mechanical, recording or otherwise, without the permission of the publisher.

Foreword

This volume contains papers presented at the Third International Workshop on Multistrategy Learning (MSL-96), held in Harpers Ferry, WV, May 23-25, 1996. The workshop was organized by the Machine Learning and Inference Laboratory at George Mason University, with the collaboration of the International Joint Conferences on Artificial Intelligence, Inc. Support for the workshop was provided by the National Science Foundation, Office of Naval Research and American Association for Artificial Intelligence.

The theme of the workshop, multistrategy learning, concerns theoretical and empirical issues in the development of learning systems that employ multiple inferential and/or computational strategies. The study of such systems draws upon the achievements in all other research subareas of machine learning, and constitutes a major new research challenge for this field. As humans are multistrategy learners, multistrategy learning has a natural connection to cognitive studies of learning, and provides an excellent opportunity for cross-fertilization of these two areas. Due to their versatility and the ability to integrate complementary strategies, multistrategy learning systems have a potential for solving more complex learning problems than monostrategy systems, which have so far been the main focus of machine learning research. Multistrategy learning workshops serve as a forum for presenting and discussing research progress in this area. MSL-96 is a sequel to the previous workshops, MSL-91 and MSL-93, also organized by the GMU Machine Learning and Inference Laboratory.

Papers in this volume present a sample of the recent research on multistrategy learning conducted at major research laboratories in Australia, Austria, Belgium, France, Germany, Italy, Japan, New Zealand, Poland, and the United States. This indicates a truly international research interest in this area. Major topics of the workshop include: the study of interrelationships among learning strategies and paradigms, cognitive models of learning and their relationships to methods and paradigms of machine learning, the development of multistrategy learning systems, and their practical applications. The papers have been grouped into four categories, according to their primary themes:

Theoretical Issues
Cognitive Models
Methods and Systems
Special Topics and Applications.

The organizers thank all of the individuals and organizations for making this workshop possible, in particular:

Larry H. Reeker, Program Director of the Knowledge Models and Cognitive Systems Program at the National Science Foundation, and Michael O. Shneier, Program Director of Artificial Intelligence at the Office of Naval Research, for their interest and support that was indispensable to the organization of this workshop.

Ray Perrault, President of the International Joint Conferences on Artificial Intelligence, Inc. for his interest and collaboration.

Carol Hamilton and Mike Hamilton from American Association for Artificial Intelligence, for their help in the preparation of the workshop proceedings.

Program Committee members John Anderson, Francesco Bergadano, Jaime Carbonell, Hugo De Garis, Luc De Raedt, Marie desJardains, Diana Gordon, Kenneth Haase, Heedong Ko, Yves Kodratoff, Stan Matwin, Doug Medin, Raymond Mooney, Stephen Muggleton, Michael Pazzani, Ashwin Ram, Lorenza Saitta, Claude Sammut, Jude Shavlik, Derek Sleeman,

Gheorghe Tecuci, and the auxiliary reviewers, Gilles Bisson, Hendrik Blockeel, Eric Bloedorn, Luc Dehaspe, Denise Gurer, Reuven Karni, Ken Kaufman, Mark Maloof, Herve Mignot, Giancarlo Ruffo, Michele Sebag, Haleh Vafaie, and Qi Zhang, for their careful and timely reviews of the submitted papers. Their assistance was essential for ensuring the high quality of the contributions.

Invited speakers Jaime Carbonell, Hugo De Garis, Luc De Raedt, Kenneth Haase, Larry Hunter, Doug Medin, Katharina Morik, Michael Pazzani, Paul Rosenbloom, Lorenza Saitta, Benjamin Smith, and Claude Sammut.

Abhay Kasera, Grant Manager at the GMU Machine Learning and Inference Laboratory, and Local Arrangements Chair, who so diligently directed and executed many organizational aspects of the workshop.

Eric Bloedorn, Ken Kaufman, Mark Maloof, Haleh Vafaie and Qi Zhang, Graduate Research Assistants in the GMU Machine Learning and Inference Laboratory, for skillfully handling many organizational chores. They were a great and reliable team, whose help cannot be overstated.

Yves Kodratoff, Stan Matwin and Gheorghe Tecuci for their contribution to the initiation and development of the Multistrategy Learning Workshop series.

Last but not least, our deep and special gratitude goes to the Defense Advanced Research Projects Agency, the National Science Foundation and the Office Naval Research for supporting research in the Machine Learning and Inference Laboratory. Their continuous support over the years has enabled us to develop a critical mass of researchers, and to build at George Mason University a research program in machine learning and inference. The MSL series of workshops is one of its byproducts.

We present these Proceedings to the reader with the hope that they will contribute in a meaningful manner to the dissemination of ideas and further progress in this highly challenging and important research direction.

Ryszard S. Michalski

Janusz Wnek

Contents

I. Theoretical Issues

Multistrategy Learning: When, How and Why: 3
Lorenza Saitta

Using Background Knowledge to Build Multistrategy Learners 11
Claude Sammut

A Multistrategy Approach to Relational Knowledge Discovery in Databases 17
Katharina Morik and Peter Brockhausen

Induction in Logic 29
Luc De Raedt

Induction as Knowledge Integration 39
Benjamin D. Smith and Paul S. Rosenbloom

Fusing the Results of Diverse Algorithms 53
John F. Elder IV

Learning Synthesis Schemes in Intelligent Systems 57
Lech Polkowski and Andrzej Skowron

II. Cognitive Models

The Basic Level and Privilege in Relation to Goals, Theories, and Similarity 71
Douglas L. Medin, Elizabeth B. Lynch, John D. Coley, and Scott Atran

Coevolution Learning: 85
Synergistic Evolution of Learning Agents and Problem Representations
Lawrence Hunter

A Comparison of Action Selection Learning Methods 95
Diana Gordon and Devika Subramanian

A Cognitive Modeling Approach to Learning of Ill-defined Categories 103
Mukesh Rohatgi

III. Methods and Systems

Multistrategy Task-adaptive Learning Using Dynamically Interlaced Hierarchies: 115
A Methodology and Initial Implementation of INTERLACE
Nabil W. Alkharouf and Ryszard S. Michalski

Combining Symbolic and Numeric Methods for Learning to Predict Temporal Series 125
Marco Botta and Attilio Giordana

An Empirical Study of Computational Introspection: 135
Evaluating Introspective Multistrategy Learning in the Meta-AQUA System
Michael T. Cox

From Instances to Rules: A Comparison of Biases 147
Pedro Domingos

Multistrategy Learning to Apply Cases for Case-Based Reasoning 155
David B. Leake, Andrew Kinley and David Wilson

Inductive Logic Programming + Stochastic Bias = Polynomial Approximate Learning 165
Michele Sebag, Celine Rouveirol and Jean-Francois Puget

Theory Restructuring: Coarse-grained Integration of Strategies for Induction 177
and Maintenance of Knowledge Bases
Edgar Sommer

Decision Combination Based on the Characterization of Predictive Accuracy 191
Kai Ming Ting

A Multistrategy Learning System for Planning Operator Acquisition 203
Xuemei Wang

On-line Metalearning in Changing Contexts: METAL(B) and METAL(IB) 217
Gerhard Widmer

Learning Weighted Prototypes Using Genetic Algorithms 229
Jianping Zhang and Qiu Fan

IV. Special Topics and Applications

Revising User Profiles: The Search for Interesting Web Sites 239
Daniel Billsus and Michael Pazzani

CAM-BRAIN: ATR's Billion Neuron Artificial Brain Project 251
Hugo De Garis

Integrating EBL and ILP to Acquire Control Rules for Planning 271
Tara A. Estlin and Raymond J. Mooney

Forecasting of Options in the Car Industry Using a Multistrategy Approach 281
Stefan Ohl

How to Predict It: Inductive Prediction by Analogy Using Taxonomic Information 295
Takashi Ishikawa and Takao Terano

Addressing Knowledge Discovery Problems in a Multistrategy Framework 305
Kenneth Kaufman

Automated Extraction of Expert System Rules from Databases Based on Rough Set Theory 313
Shusaku Tsumoto and Hiroshi Tanaka

Comparative Analysis of Amino-acid Sequences Based on Rough Set Theory 325
and Change of Representation
Shusaku Tsumoto and Hiroshi Tanaka

Application of Multistrategy Learning in Finance 333
Martin Westphal and Gholamreza Nakhaeizadeh

Author Index 339

I. Theoretical Issues

Multistrategy Learning: When, How and Why

Lorenza Saitta

Università di Torino, Dipartimento di Informatica
Corso Svizzera 185, 10149 Torino (Italy)
saitta@di.unito.it

Abstract

In this paper we describe our experience in designing, implementing and using multistrategy approaches to machine learning. In particular, the system SMART+, WHY and REGAL, each one interpreting "multistrategy" in a different way, will be briefly described, and the lessons learned from their realization commented upon.

1 Introduction

With the growing complexity of the applications of machine learning systems, there has been an increasing need of exploiting the power of multiple (or hybrid) approaches to their development, and several projects have been devoted to this research line. Even though there is no general agreement on what *multistrategy* really means, efforts have been made to find out an adequate definition; for example, Michalski (1991) proposes to classify learning methods according to a set of "knowledge transmutation" processes.

Independently of a precise definition, a wide variety of learning approaches and systems can be considered included in the category; they differ for what they mean for "strategy" or for the implemented type of interaction among strategies. However, they all share the basic fact that at some architectural level different components cooperate to achieve a learning task.

In a learning system, multiple strategies can cooperate according to a variety of configurations and interactions. One of the simplest ways to achieve cooperation is the *Toolbox* idea. In Figure 1 the *Selection* case is described: a number of "strategies" S_i ($1 \le i \le M$) work on (or are related to) the same learning problem (aspect) P. The result R_j is the "best", according to a given criterion, among the results given by each one of the strategies. The choice can be done before using the S_i's (*ex ante* selection), if a-priori information is available, or after using the S_i's (*ex post* selection).

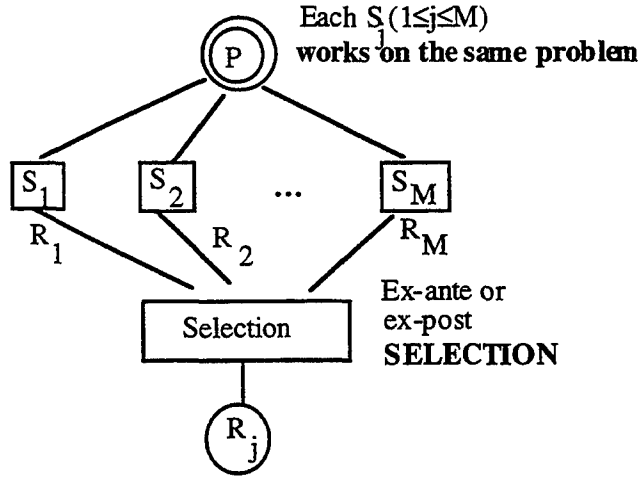


Figure 1 – Toolbox (Selection). Only one among the strategies S_i ($1 \le i \le M$) has to be chosen. This selection can be done before or after the strategies have been used.

A toolbox with selection approach is described in (Kodratoff et al., 1992), where a number of algorithms/systems, oriented to learn rules, concepts, or classes, or combinations of these, are offered; the choice is based, among other parameters, on the type of sought knowledge representation and on the target learning task. In this case, the strategies correspond to algorithms/systems.

A modification of the above schema is the *Toolbox* with *Integration*, represented in Figure 2: the "strategies" S_i ($1 \le i \le M$) again work on (or are related to) the same learning problem (aspect) P. The result R is a function $f(R_1, ..., R_M)$ of the results obtained by all the strategies.

An example of this kind of toolbox integration is given by Drobnic and Gams (1993), who try to combine the results of alternative classification systems to obtain a single final decision.

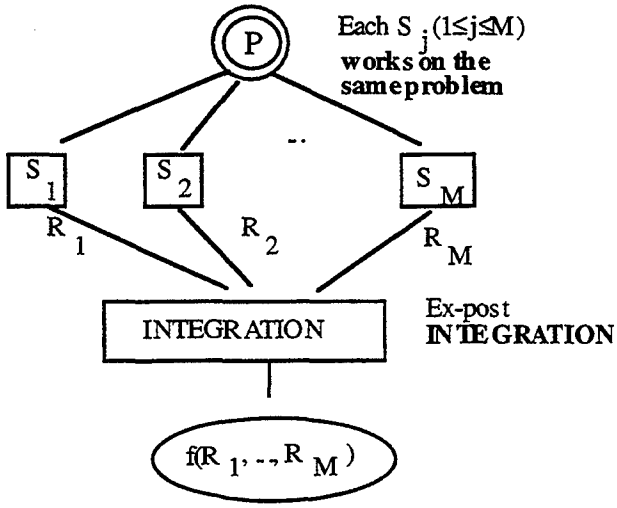


Figure 2 – Toolbox (Integration). The global results is a combination of the results obtained by all the strategies S_i ($1 \le i \le M$).

In the toolbox approach, the strategies S_i work independently, in the sense that each one of them can achieve the common goal by itself, and all of them could be possibly run in parallel. A tighter model of cooperation is *Sequential Cooperation,* represented in Figure 3. According to this schema, each strategy works (possibly iteratively) during a different phase Ph_i of the problem solution, according to a fixed temporal/logical order. The final result is the result provided by the last strategy in the chain.

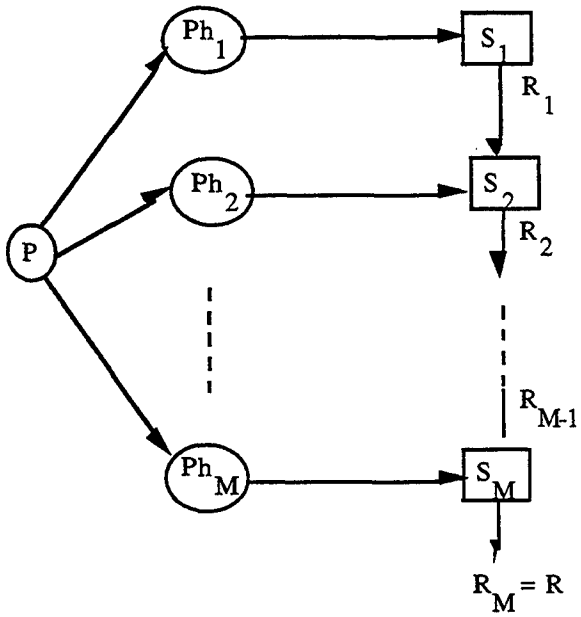


Figure 3 – Sequential Cooperation. The global results is supplied by the last strategy, S_M, in the chain.

Finally, when different phases or aspects of subproblems cannot be isolated, all the strategies work together, as reported in Figure 4, and this cooperation results in a *Tight Integration.*

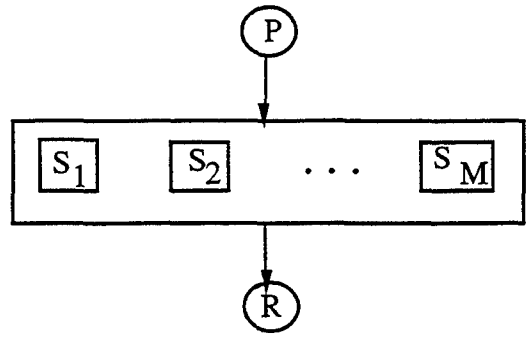


Figure 4 – Tight Integration. The result is provided by the joint work of all the strategies.

Also a mixture of the described schemes is possible. For instance, the *boosting* algorithm AdaBoost (Freud & Schapire, 1995; Dietterich, Kearns & Mansour, 1996) combines the sequential cooperation and toolbox with integration schemes, in the special case in which all the strategies are the same one, consisting of a run of a (weak) learning algorithm; each run is made on a modification of the original training examples distribution, combining then the hypotheses generated at each run through a threshold majority voting.

2 Three Multistrategy Learning Systems

In this section we briefly describe the learning systems SMART+, WHY and REGAL, and discuss the reasons underlying the choice of a multistrategy approach for each one of them.

2.1 The system SMART+

SMART+ (Botta & Giordana, 1993) is a multistrategy learning tool derived from the learning system ML-SMART (Bergadano, Giordana & Saitta, 1991). The system top-down learns concepts, expressed as a set of classification rules in First Order Logic, starting from a set of training examples and a background theory.

Multistrategy occurs in SMART+ at several levels. First of all, different inference schemes are integrated: induction, deduction and abduction. Given a domain theory and a definition of the target concept, deduction is first used as much as possible, in an EBL manner, to set up the frontier of a search tree, whose nodes are then expanded by induction until a halt condition on the quality of the leaves is reached.

Induction adds to the hypotheses under construction subformulas in the specified hypothesis language. During this process, the domain theory may also be used to guide the addition of subformulas, taking into consideration their occurrence in the theory. Let, for instance, the domain theory **T** contain the two rules describing the stability of a cup:

$$\mathbf{T} = \{\text{stable}(x) \leftarrow \text{part-of}(x,y) \wedge \text{flat}(y) \wedge \text{bottom}(y);$$
$$\text{stable}(x) \leftarrow \text{part-of}(x,y) \wedge \text{support}(y) \wedge \text{part-of}(x,w) \wedge \text{body}(w) \wedge \text{above}(w, y)\}$$

and let

$$h = \text{light}(x) \wedge \text{flat}(y) \wedge \text{bottom}(y)$$

be the current hypothesis, to be further specialized. Then, using the theory in an abductive way, we can suppose that the subformula g = flat(y) ∧ bottom(y) occurring in h suggests the presence of the concept stable(x), which can be substitute to g in h. If the resulting formula is too general, then stable(x) can be expanded with the second of the two clauses, obtaining thus the new hypothesis:

$$h' = \text{light}(x) \wedge \text{part-of}(x,y) \wedge \text{support}(y) \wedge \text{part-of}(x,w) \wedge \text{body}(w) \wedge \text{above}(w, y).$$

This process may substantially reduce the number of trials of adding new predicates one at a time.

When expanding the search tree, alternative search strategies are available: best-first, deep-first with backtracking or beam search (hill climbing as a special case). These strategies are in alternative, and only one of them can be selected for a given run.

Also the criterion for evaluating a hypothesis can be selected among a set of available ones: information gain, a statistical evaluation of the hypothesis' completeness and consistency, abducibility in the domain theory (i.e., ratio between the number of literals in the hypothesis body and number of them occurring in the theory), and, finally, a combination of the preceding ones.

Considering the kind of data that SMART+ can handle, examples can be described by both categorical and numerical attributes, so as to combine a numerical and symbolic approach to learning. In particular, numerical attributes are discretized by means of fuzzy sets, in order to avoid singularities in the behavior at the discretization points. Finally, the fuzzy sets introduced for each numerical attribute may contain numerical parameters, determining their position and span. The values of these parameters are optimized in two phases: during the hypothesis construction, based on information local to the current node of the search tree, and after a complete hypothesis has been found. At this point, a genetic algorithm further optimizes the parameter values, taking into consideration global information.

By summarizing, SMART+ exploits multistrategy at the level of reasoning strategies, by means of a tight integration scheme. The same kind of integration exists between the numerical and symbolic approaches. On the other hand, SMART+ behaves as a toolbox with selection with respect to the choice of a search strategy and the hypothesis evaluation criterion. Finally, sequential cooperation exists between the symbolic paradigm and the genetic one.

At the beginning, the system ML-SMART only exploited induction with depth-first search strategy, and had a fixed evaluation criterion; moreover, the fuzzy sets for numerical attributes were defined manually by an expert. For the first applications handled, such as recognition of simple images and of speech sounds, the facilities offered by the system were sufficient. As the range and complexity of the applications increased, new requirements were imposed by the end user.

One of the most important was the possibility of explaining the learned knowledge. In order to offer this possibility, the learning system must use a-priori knowledge of the domain, so that inference mechanisms, suitable to exploit this knowledge, have to be incorporated in the system. The availability of domain knowledge, if effectively used, can also provide a strong reduction in the amount of search to be done, focalizing the search towards promising regions.

The need of integration between the numerical and symbolic approaches has its root in the kind of data to be handled in real-world domains, where this co-occurrence is quite frequent. As continuous and categorical features may interact, it is necessary that the same learning algorithm can treat both at the same time, introducing the necessity of some type of discretization. Integration between numerical and symbolic is realized, in SMART+, through a sequential integration type, as the definition of the fuzzy sets (apart from their optimization) precedes the search of the hypothesis space.

Regarding the search strategies and the evaluation criteria, the possibility of offering a choice was motivated by the desire to render the system as flexible as possible, at a low programming expense. Moreover, the multiplicity of possibilities let SMART+ be able to simulate other existing approaches, such as the type of search implemented in decision trees construction.

The reasons of using a genetic algorithm to perform the final optimization are the same that were at the basis of the system REGAL, and will be discussed later.

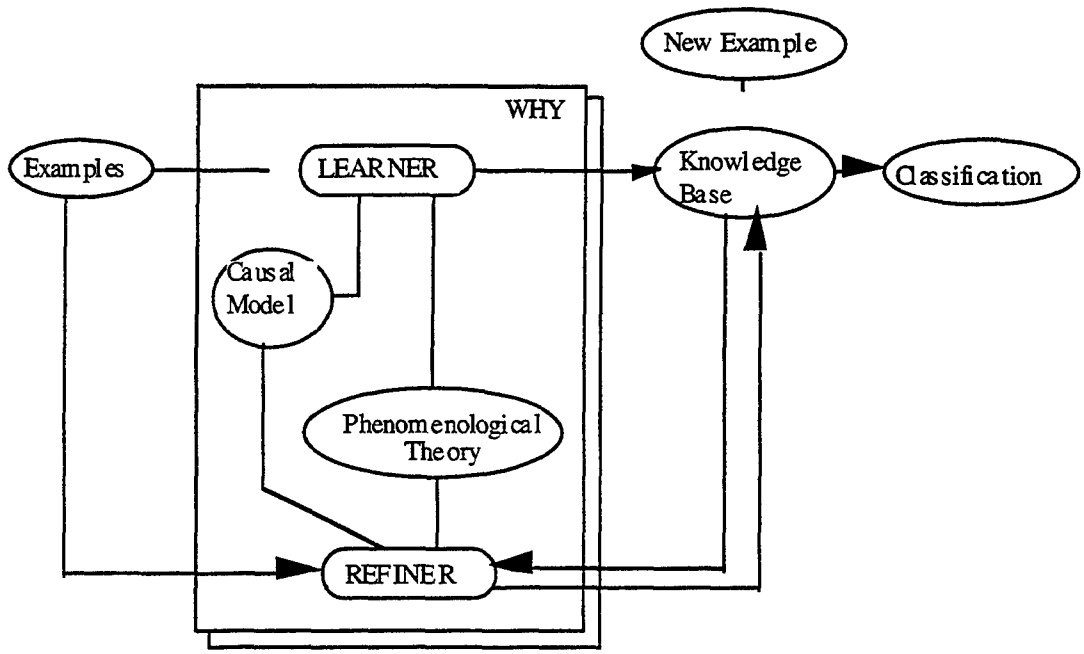


Figure 5 – WHY's architecture. The system is provided with a causal model of the domain and a phenomenological theory and is constituted by two modules: the LEARNER and the REFINER. The LEARNER interprets and explains a set of classified examples, and generates a knowledge base of heuristic rules, allowing classification of new examples to be made upon recognition of specific premises. The REFINER individuates errors and incompleteness both in the knowledge base and in the causal and phenomenological theories.

2.2 The system WHY

WHY is a system that learns and refines a knowledge base for classification tasks, using a *causal model* of the domain, C, a set of examples and a *phenomenological* theory, P, describing concrete manifestations of abstract concepts occurring in C. The system, whose architecture is described in Figure 5, can also be used in a semi-automated way, allowing a direct interaction with a human user (Baroglio, Botta & Saitta, 1994; Saitta, Botta & Neri, 1993).

The representation of all kinds of knowledge used by WHY is based on a First Order Logic language L. The causal model C is represented, at the logical level, as a network. Nodes in the network are either *primary* (ellipses) or *accessory* (rectangles and clouds). Primary nodes correspond to processes or system states and a subset of these nodes contains the *first causes* (shaded nodes). Accessory nodes represent either *constraints* on causal links (rectangles) or *contexts* (clouds), i.e., conditions to be satisfied by the environment. In Figure 6 a causal network describing heat transfer phenomena is represented.

The phenomenological theory P contains structural information, definitions of ontologies, general knowledge and a set of rules describing manifestations associated to abstract concepts. The theory P contains also the links between the causal network C and the classes, which the examples belong to. Examples of pieces of information contained in P are the following ones:

$$\text{CONDUCTION}(x,y) \Leftarrow \text{OBJ}(x) \wedge \text{OBJ}(y) \wedge \text{CONTACT}(x,y)$$
$$\text{HEAT-SOURCE}(x) \Leftarrow \text{fire}(x)$$
$$\text{GOOD-HEAT-COND}(x) \Leftarrow \text{METAL}(x)$$
$$\text{CONTACT}(x,y) \Leftarrow \text{adjacent}(x,y) \vee \text{part-of}(x,y) \vee \text{inside}(x,y)$$
$$\text{METAL}(x) \Leftarrow \text{lead}(x) \vee \text{tin}(x) \vee \text{gold}(x) \vee \text{iron}(x)$$

The target knowledge base K consists of a set of classification rules, derived from the causal network. For instance, from C, the following rule can be learned (x and y denote objects, while T and U denote temperatures):

$$\forall x,y \; [\exists T,U\{\text{OBJ}(x) \wedge \text{OBJ}(y) \wedge \text{diff}(x,y) \wedge$$

$$T \; \text{HERMAL-GAP}(x,y,T,U) \wedge \neg \text{HEAT-SOURCE}(x) \wedge \text{GOOD-HEAT-COND}(y) \wedge \neg \text{HEAT-SOURCE}(y) \wedge \neg\text{CHANGING-STATE}(x) \wedge \neg\text{CHANGING-STATE}(y) \Rightarrow \text{THERMAL-EQUILIBRIUM}(x,y)]$$

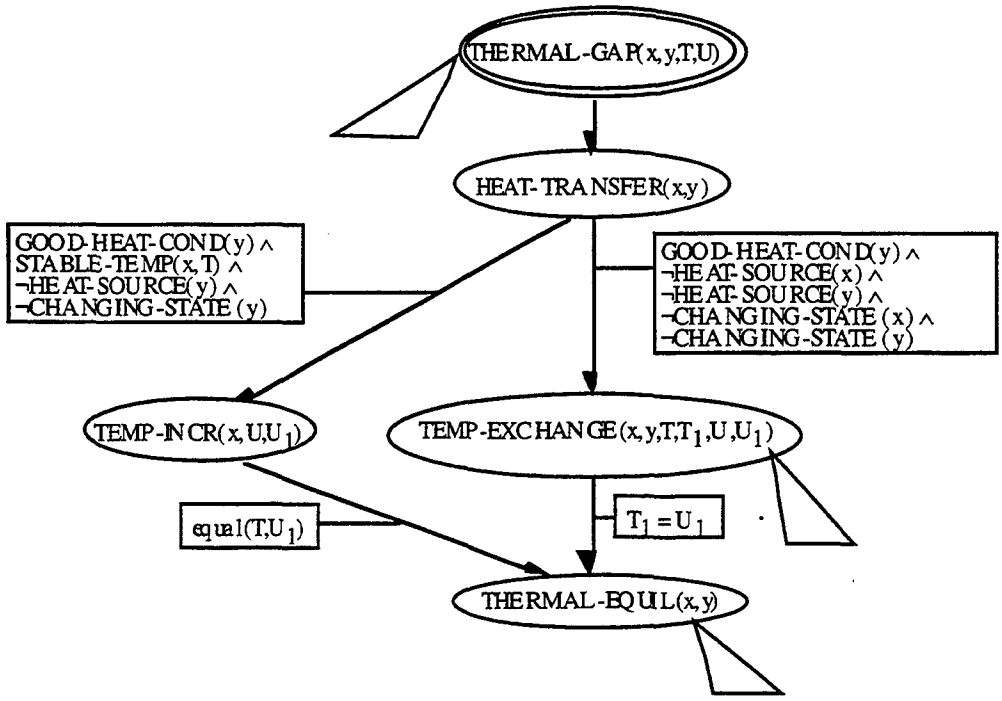


Figure 6 – A causal model C describing heat transfer phenomena.

Four basic reasoning mechanisms are integrated into the system WHY: induction, deduction, abduction and prediction. *Induction* is used when the domain theory is incomplete, and is performed by a call to the system SMART+. Deduction is used to build up a data structure called the *justification forest,* containing links between concrete manifestations and abstract predicates; this forest is a generalization of the explanation tree in EBL. The causal model can be searched either from effects to causes (search for an explanation via *abduction*) or from causes to effects (*prediction* of consequences via deduction).

The system can learn both in one step, from a set of examples, or incrementally, by analyzing one example at a time and updating correspondingly the target knowledge, if necessary. It can also refine a knowledge base, acquired either by itself or supplied by an external source, for instance by an expert. Finally, a semi-automated theory revision is possible, via an interaction with a human expert. WHY was designed as an alternative improvement of ML-SMART, originated by an explicit request by the end user of a complex application of mechanical diagnosis (Giordana et al., 1993). In fact, it seemed important, for both prognosis and repair, not to limit the system answer to the location and classification of faults, but to extend it to include an explanation of why the fault occurred and the identification of its possible causes. In the field of mechanical functioning, qualitative causal models were available. However, in order to effectively exploit these models, the complexity of the system's reasoning scaled up an order of magnitude. Actually, WHY is not a typical learning system, but is rather a general reasoner that extracts from its a-priori knowledge as much information as possible, and tries to drastically reduce the amount of inductive search to be performed.

From the point of view of its structure as a multistrategy system, WHY has the characteristics of a toolbox with integration, with respect to the use of induction and the other reasoning mechanisms. In this case, the "strategies" are whole learning module/systems. Another sense in which WHY is a multistrategy system, using the toolbox with integration scheme, is the cooperation between the learner and the performance module, which exploits both the target heuristic knowledge and the causal network, in case of a difficult classification, not covered by the learned heuristic rules.

The peculiar features of WHY allows tasks, that are difficult to be perform with other learning systems, to be realized. For instance, it was used to investigate the effects of example order presentation on the learning behavior. The presence of a causal model allowed the kind of examples best suited to improve learning in both speed and quality to be predicted, without relying an experiments only based on random order presentations.

Moreover, the "human-like" nature of WHY's learning methods make the system particularly well suited to model simple aspects of human learning. As an example, we are using the system to model conceptual changes in young students learning elementary physics.

2.3 The system REGAL

The last system we consider is REGAL, a learner of First Order Logic concepts based on a genetic algorithm (Neri & Giordana 1995; Neri & Saitta, 1995; Giordana & Neri, 1996). REGAL learns disjunctive concept descriptions by exploiting the theory of niches and species formation (Mahfoud, 1995). Individuals in the population encode partial solutions, i.e. conjunctive formulas, and the whole population is a redundant set of these partial solutions, a subset of which has to be chosen.

Species formation is achieved, in REGAL, by means of two basic mechanisms: on one hand, the selection process is modified by introducing a novel selection operator, called *Universal Suffrage Operator,* and, on the other, the normal population evolution is controlled by a long-term focusing strategy, handled by a special "supervisor" process.

The language for describing the hypotheses is a subset of First Order Logic, namely a Horn clause language, in which terms can be variables or disjunctions of constants (see, for instance, internal disjunction in Induce (Michalski, 1983)), and negation occurs in a restricted form. The set of admissible formulas (hypotheses) is specified by a *Language Template,* which is mapped onto a fixed-length bit string.

The fitness function, used to evaluate individuals, takes into account consistency and simplicity of the hypotheses only, because completeness is taken in care by the Universal Suffrage selection operator.

In REGAL there are four types of crossovers: the *two-point* (De Jong, 1975) and *uniform* (Syswerda, 1989) crossovers, and the *generalizing* and *specializing* crossovers, specifically designed for the task at hand. Finally, a *seeding* operator, reminiscent of the use of a "seed" in Induce (Michalski, 1983), returns a formula covering a given positive instance.

REGAL is a distributed system, working in a computational environment consisting of a network of interconnected subpopulations, exchanging individuals at each generation. The distributed model envisages a SUPERVISOR processor, coordinating a set of *Nodal Genetic Algorithms*, each one searching a subset of the hypothesis space, biased by a different subset of the learning set. The SUPERVISOR realizes an explicit long-term strategy aimed at forming a classification theory (i.e., a complete concept description), with the help of parallel evolving populations. The SUPERVISOR performs two other fundamental tasks: it periodically extracts a classification theory from the global population and, when it finds a satisfactory one, halts the whole system, according to a chosen criterion.

The system REGAL show a tight integration between two learning paradigm: symbolic learning and genetic search. The basic idea motivating the development of this system starts from the consideration that a most critical issue in learning is computational complexity. Instead of reducing the hypothesis space through biases, as is done in most learning systems, we have chosen to leave the search space (and then the representation language) as wide as possible, while increasing, on the contrary, the computational resources dedicated to the search, by exploiting parallelism.

3 Conclusions

In order to face different classes of learning applications, we have developed a number of systems with different characteristics and architectures. The experience gained so far seems to indicate that the complexity of real-world tasks requires a matching complexity in the learning system, in terms of cooperation among components, in such a way that the system results more flexible and adaptable to the application needs. On the other hand, designing more complex systems is a costly process, both in terms of resources required for their realization, and also in terms of speed and ease of use of the learned knowledge. This is especially true when a large amount of a-priori knowledge is to be encoded, so that the improvements in the system outcomes should compensate the greater costs and efforts.

The toolbox approach, however, does not require a large overhead for the integration. On the other hand, it introduces a new problem, namely the problem of the selection. Given a task, there is up to now no guidance (except for a few "rules of thumb" only little less than obvious) to decide what kind of approach or what system is likely to give the "best" performances, according to given criteria.

For multistrategy systems, deciding what components, at what level of the learning architecture should be put together is even more difficult, without a clear understanding of the way the single components are related to the features of the task at hand.

References

Baroglio, C., Botta, M., and Saitta, L. 1994. WHY: A System that Learns from a Causal Model and a Set of Examples. In R. Michalski & G. Tecuci (Eds.), *Machine Learning: A Multistrategy Approach, Vol IV,* 319-348. Los Altos, CA: Morgan Kaufmann.

Bergadano, F., Giordana, A., and Saitta, L. 1991. *Machine Learning: A General Framework and its Applications,* Chichester, UK: Ellis Horwood Ltd..

Botta, M., and Giordana, A. 1993. SMART+: a Multi Strategy Learning Tool. In Proc. 13th Int. Joint Conference on Artificial Intelligence, 937-943. San Mateo, CA: Morgan Kaufmann.

De Jong, K. A. 1975. Analysis of the Behaviour of a Class of Genetic Adaptive Systems, Doctoral Dissertation, Dept. of Computer and Communication Sciences, University of Michigan, Ann Arbor, MI.

Dietterich, T., Kearns, M., and Mansour, Y. 1996. In Proceedings of the 13th Int. Conf. on Machine Learning. San Francisco: Morgan Kaufmann. Forthcoming.

Drobnic, M., and Gams, M. 1993. Multistrategy Learning: An Analytical Approach. In R. Michalski & G. Tecuci (Eds.), Proc. of the Second Multistrategy Learning Workshop, 31-41. Fairfax, VA: George Mason University, Center for Artificial Intelligence.

Freud, Y., and Schapire, R. 1995. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In Proceedings of Second European Conf. on Computational Learning Theory, 23-37. Springer Verlag.

Giordana, A., and Neri, F. 1996. Search-Intensive Concept Learning. *Evolutionary Computation.* Forthcoming.

Giordana, A., Saitta, L., Bergadano, F., Brancadori, F., and De Marchi D. 1993. ENIGMA: A System that Learns Diagnostic Knowledge. *IEEE Transactions on Knowledge and Data Engineering,* KDE-5: 15-28.

Kodratoff, Y., Sleeman, D., Uszynski, M., Causse, K., and Craw, S. 1992. Building a Machine Learning Toolbox. In L. Steels and B. Lepape (Eds.), *Enhancing the Knowledge-Engineering Process - Contributions from Esprit,* Elsevier Publ. Co.

Mahfoud, S. W. 1995. *Niching Methods for Genetic Algorithms,* Ph.D. Thesis, University of Illinois, Urbana-Champaign, IL.

Michalski, R. 1983. A Theory and Methodology of Inductive Learning. In R. Michalski, J. Carbonell & T. Mitchell (Eds.), *Machine Learning: An AI Approach, Vol. I,* 83-134. Los Altos, CA: Morgan Kaufmann.

Michalski, R. 1991. Inferential Learning Theory as a Basis for Multistrategy Task-Adaptive Learning. In R. Michalski & Y. Kodratoff (Eds.), Proc. of the First Multistrategy Learning Workshop, 3-18. Fairfax, VA: George Mason University, Center for Artificial Intelligence.

Neri, F., and Saitta, L. 1995. Analysis of Genetic Algorithms Evolution under Pure Selection. In Proc. of 6-th Int. Conf. on Genetic Algorithms, 32-41. San Francisco: Morgan Kayfmann.

Neri, F., and Giordana, A. 1995. A Parallel Genetic Algorithm for Concept Learning. In Proc. of 6-th Int. Conf. on Genetic Algorithms, 436-443. San Francisco: Morgan Kaufmann.

Saitta, L., Botta, M., and Neri, F. 1993. Multistrategy Learning and Theory Revision. *Machine Learning,* 11: 153-172.

Syswerda, G. 1989. Uniform Crossover in Genetic Algorithms. In Proc. 3th Int. Conf. on Genetic Algorithms, 2-9. San Mateo, CA: Morgan Kaufmann.

Using Background Knowledge to Build Multistrategy Learners

Claude Sammut

School of Computer Science and Engineering
University of New South Wales
Sydney Australia 2052
claude@cse.unsw.edu.au

Abstract

This paper discusses the role that background knowledge can play in building flexible multistrategy learning systems. We contend that a variety of learning strategies can be embodied in the background knowledge provided to a general purpose learning algorithm. To be effective, the general purpose algorithm must have a mechanism for learning new concept descriptions which can refer to knowledge provided by the user or learned from some other task. The method of knowledge representation is a central problem in designing such a system since it should be possible to specify background knowledge in such a way that the learner can apply its knowledge to new information.

Introduction

There are many reasons why one may wish to combine a variety of learning strategies in one system. Many experiments have confirmed that we are yet to find a single learning algorithm that does best in all circumstances (for example, Michie, Spiegelhalter & Taylor, 1994). It is often more effective to use different algorithms at different times than to try to use one algorithm all the time. This approach raises several questions, including: under which circumstances is a particular method appropriate and how large does our library of algorithms have to be?

Often, multistrategy learning systems have a fixed set of algorithms to draw upon and their interaction is predetermined by the programmer. In other words, they are often designed to be specific to a particular type of learning problem. In this paper, we suggest that general purpose multistrategy learners can be constructed by allowing the learner to make use of complex background knowledge.

We contend that a great deal of flexibility can be gained by a machine learning system which uses a concept representation language that permits references to user defined background knowledge as well as knowledge derived from other learning tasks. First, we illustrate the power of complex background knowledge with an example of numerical reasoning in inductive logic programming. Next, we describe a machine learning system that encourages the use of learned as well as predefined background knowledge. Finally, we discuss the connection between background knowledge and multistrategy learning.

An Example: Numerical Reasoning in ILP

The effectiveness of background knowledge as a means of providing a variety of learning strategies can best be seen through an example. Sammut, Hurst, Kedzier and Michie (1992) describe a learning problem where a human pilot is required to fly an aircraft in a flight simulator. During the flight the actions of the pilot are logged, along with the situation in which the action is performed. Flight logs are used as input to an induction program which generates rules for an autopilot.

The task of piloting an aircraft through a complete flight plan is a complex task involving a number of stages each of which is defined by a particular goal. For example, the pilot may be told to take off; climb to a particular altitude; turn to some heading; maintain straight and level flight until a certain marker is reached, etc. For each stage of a flight, we build a separate set of control rules. Further complexity is added by the fact that an aircraft has a number of controls, eg. elevators, ailerons, rudder, flaps, throttle, etc. Within each stage we build rules for each control available to the pilot. The result is a two-level controller where the top level is an "executive" which invokes low level agents to perform a particular task. Thus, "Learning to Fly" applies machine learning in a number of ways to build a control system that has different strategies for different situations. While the result is a multistrategy controller, our original experiments only involved a single learning algorithm., C4.5 (Quinlan, 1993).

Although these experiments resulted in working autopilots capable of flying the aircraft from take-off to landing, the rules that were generated were often large, difficult to read and not always robust under different flight conditions. One reason for some of these difficulties is that only the raw data from the simulator were presented to the learning algorithm. The data obtained from the simulator include the position, orientation and velocities of the aircraft as well as the control settings. While these data are complete in the sense that they contain all the information necessary to describe the state of the system, they are not necessarily presented in the most convenient form. For

Table 1. Background Predicates

pos(P, T)	position, *P*, of aircraft at time, *T*.
before(T1, T2)	time, *T1*, is before time, *T2*.
regression(ListY, ListX, M, C)	least-square linear regression, which tries to find $Y = M \times X + C$ for the list of X and Y values.
linear(X, Y, M, C)	linear(X, Y, M, C) :- Y is M*X+C.
circle(P1, P2, P3, X, Y, R)	fits a circle to three points, specifying the centre (X, Y) and radius, R
=<, >=, abs	Prolog built-in predicates

example when a pilot is executing a constant rate turn, it makes sense to talk about trajectories as arcs of a circle. Induction algorithms, such as C4.5, can deal with numeric attributes to the extent that they can introduce in equalities, but they are not able to recognise trajectories as arcs or recognise any other kind of mathematical property of the data.

Srinivasan and Camacho (forthcoming) have shown how such trajectories can be recognised with the aid of a sophisticated mechanism for making use of background knowledge. This mechanism is part of the inductive logic programming system Progol (Muggleton, 1995). The feature of Progol that most interests us here is that the learning algorithm is embedded in a custom built Prolog interpreter. Concepts learned by Progol are expressed in the form of Prolog Horn clauses and background knowledge is also expressed in the same form. Moreover, the Prolog interpreter's built-in predicates may appear in either learned or background clauses.

Like Marvin (Sammut, 1981; Sammut & Banerji, 1986), Progol begins by taking the description of a single positive example and *saturates* the description by adding to it literals derived from background knowledge. These new literals result from predicates in the background knowledge being satisfied by data in the example. Suppose the example contained the numbers 2 and 3 somewhere in the example description. Also suppose that the built-in predicate '<' were supplied as background knowledge. Saturation might add a new literal, $X < Y$, to the description of the example. X has been substituted for 2 and Y for 3. Progol permits *mode* declarations to restrict the application of predicates to avoid an explosion in the number of literals that the saturation procedure may try. Note that *any* of Prolog's built-in predicates can be declared as background knowledge and any user defined Prolog program may also be declared as background knowledge. Furthermore, since Progol concept representation language is Horn clauses, any learned concepts may also be entered as background knowledge.

When saturation is completed, Progol has a *most specific clause* to serve as a bound on a general-to-specific search. Beginning with the most general clause, ie. one with an empty body, Progol tries to find a subset of the most specific clause that satisfies a minimum description length criterion for the best clause. Muggleton (1995) describes the search procedure.

Progol's ability to use background knowledge can be used in a multistrategy approach to learning to pilot an aircraft. Srinivasan and Camacho applied Progol to the problem of learning to predict the roll angle of an aircraft during a constant rate turn at a fixed altitude. To do this effectively, the target concept must be able to recognise the trajectory as an arc of a circle. The predicates shown in Table 1 are included in the background knowledge.

The *pos* predicate is the 'input' to the learner since it explicitly describes the trajectory of the aircraft as a sequence of points in space. These points are derived from flight logs. The *before* predicate imposes an ordering on the points in the trajectory. The *regression* predicate provides the key to finding the relationship between the roll angle and the radius of the turn. The mode declaration for *regression* specifies that the first two arguments are lists which described the sequence of pairs of coordinates for the aircraft during the turn. The mode declaration causes Progol to generate these lists and invokes *regression* which performs a least-square regression to find the coefficients of the linear equation which relates roll angle and radius. *Regression* must be accompanied by another background predicate, *linear*, which implements the calculation of the formula.

The regression predicate is an intermediate relationship that does not appear in the final description of the learned concept. During saturation, *regression* recognises the relationship between the angle and radius, given the sequence of aircraft positions. Once the coefficients of the linear equation are available, Progol can generate a reference to *linear*. Thus the theory produced is[1]:

```
roll_angle(T1, Angle) :-
   pos(P1, T1), pos(P2, T2), pos(P3, T3),
   before(T1, T2), before (T2, T3),
   circle(P1, P2, P3, _, _, Radius),
   linear(Angle, Radius, 0.043, -19.442).
```

The *circle* predicate recognises that P1, P2 and P3 fit a circle of radius, *Radius* and regression finds a linear approximation for the relationship between *Radius* and *Angle* which is:

$$Angle = 0.043 \times Radius - 19.442$$

[1] The theory as been siplified slightly to avoid lengthy explanations of details not relevant to this discussion.

The '_' arguments for *circle* are "don't cares" which indicate that, for this problem, we are not interested in the centre of the circle.

Machine learning algorithms generally tend to be poor at numerical reasoning. This is usually left to scientific discovery systems. However, the flight trajectory example illustrates why, for many domains, it is important to incorporate numerical reasoning into induction. One way to do this is by adding background knowledge for a variety of equation solvers.

Because Progol permits the use of arbitrary Prolog code in its representation of concepts, it can invoke a variety of methods for fitting data which go beyond Progol's own ILP style of learning. Linear regression is just one example of a statistical method for fitting data. A library of such predicates gives a system like Progol the ability to use different strategies depending on the type of data available. Since the components of the library are just Prolog programs, the learning algorithm does not have to be modified to permit different data fitting methods to be added. Furthermore, the library may even include other learning algorithms. In the following section, we describe an ILP system that has such a library of learning algorithms.

Using Induction to Build Background Knowledge

An experimental ILP system is currently being developed which is actually a Prolog interpreter with a variety of machine learning and statistical algorithms included as built-in predicates. The algorithms included so far are: Aq (Michalski, 1973; 1986), ID3 with pruning (Quinlan, 1979; 1993), Induct-RDR (Gaines, 1989), naive Bayes, regression trees (Breiman et al, 1984), linear discriminant, DUCE (Muggleton, 1987). All of these algorithms can be invoked as predicates which are available as background knowledge to an ILP system similar to Progol.

First, we give a brief example of the individual use of some these algorithms and then we describe how they may be combined in a multistrategy learner. Like most attribute/value based systems, a description of the attributes and their legal values is required. Following the example of Cendrowksa (1987):

```
mode lens(
   age(young, pre_presbyopic, presbyopic),
   prescription(myope, hypermetrope),
   astigmatism(not_astigmatic, astigmatic),
   tear_production(reduced, normal),
   lens(hard, soft, none)
).
```

The task here is to learn to predict whether a person should wear a hard or soft contact lens or wear no contact lens. The class value is always the last attribute in the list. Examples are entered as ground unit clauses in Prolog's database. A small sample for the problem described above is:

```
lens(young, myope, not_astigmatic, reduced, none).
lens(young, hypermetrope, not_astigmatic, reduced, none).
lens(young, hypermetrope, not_astigmatic, normal, soft).
lens(pre_presbyopic, myope, astigmatic, reduced, none).
lens(pre_presbyopic, myope, astigmatic, normal, hard).
lens(presbyopic, myope, not_astigmatic, reduced, none).
lens(presbyopic, myope, not_astigmatic, normal, none).
lens(presbyopic, hypermetrope, astigmatic, normal, none).
```

To run an induction algorithm, we simply invoke it as a procedure call in Prolog. Thus, to build a decision tree, the following call is executed:

```
id(lens)?
```

The output of all the induction algorithms is a Prolog clause which captures, the decision tree or set of rules. Since the output is in standard Prolog form, it can be asserted into the database and used as an ordinary program.

```
lens(Age, Prescription, Astigmatism, TearProduction, Lens) :-
   (TearProduction = reduced -> Lens = none
   |TearProduction = normal ->
      (Astigmatism = not_astigmatic -> Lens = soft
      |Astigmatism = astigmatic ->
         (Prescription = myope -> Lens = hard
         |Prescription = hypermetrope ->
            |Age = pre_presbyopic -> Lens = none
            |Age = presbyopic -> Lens = none))))).
```

ILP learning systems are typically used in domains where relational information is important. These are domains in which the common attribute/value representation of traditional induction algorithms makes representation of examples and concepts difficult. However, when a relational representation is not essential, propositional learning algorithms are often still preferable because of their speed and ability to handle noise and numeric data.

We have seen how Srinivasan and Camacho used linear regression to fit numeric data. In the same way, it is possible to use regression tree methods as background knowledge to construct a function. The following data are samples from a flight log. Each clause gives the position of the aircraft and the pitch angle in 10ths of degrees.

```
pitch(500, -2461, -977, -7).
pitch(498, -2422, -955, -17).
pitch(489, -2385, -932, -20).
pitch(413, -2219, -801, -20).
pitch(397, -2190, -771, -18).
pitch(369, -2131, -709, -19).
pitch(265, -1925, -494, -15).
pitch(252, -1895, -463, -13).
pitch(231, -1835, -400, -12).
pitch(211, -1771, -339, -10).
```

```
pitch(202, -1737, -311, -8).
pitch(195, -1702, -283, -7).
```

Suppose we wish to predict the pitch of the aircraft, based on its current position. This is a gross simplification of what we would really want to know since pitch is usually not a function of position alone.

Running the CART regression algorithm, we obtain the following regression tree[1]:

```
pitch(Alt, Dist, X, Pitch) :-
    (Alt <= 216 ->
        (Alt <= 21.5 ->
            Pitch = -3.86;
            (Alt <= 83.5 ->
                Pitch = -7.70;
                Pitch = -6.11));
        Pitch = -17.18).
```

As with linear regression, regression trees can be built during saturation. Two background predicates are required to achieve this. Like linear regression, one predicate is needed to invoke the regression tree algorithm during saturation and the second predicate is needed for execution of the regression tree, eg. pitch. In this case, however, the second predicate is built by the first predicate during saturation. Note that when predicates of this form are built, the system is performing quite sophisticated constructive induction.

Structured Induction

The present solution to the "Learning to Fly" task is an example of *structured induction*. Shapiro (1987) introduced the idea of structured induction as a means of simplifying a learning task and for making the result of learning more human readable. His experiments were conducted in the familiar domain of chess end-games. Quinlan (1979) had previously demonstrated that it was possible to induce decision trees for this domain. However, they were usually large and end-game experts could find little in them that corresponded to their own intuition. To overcome this problem, Shapiro obtained the help of a chess master who was able to describe high-level features that players looked for.

Armed with this knowledge, Shapiro induced decision trees for each of the high-level features and organised the whole knowledge base as a tree of trees. The top-level tree was hand-crafted from the knowledge obtained by the chess expert. The subtrees were built by induction. The result was an accurate solution which also made sense to chess experts.

Shapiro used as uniform representation and the same type of induction algorithm throughout his analysis. This is also true of the original flight control system. All the control agents were synthesised using the same learning algorithm. However, we have seen that different algorithms may be used when one is more appropriate than another, thus extending the notion of structured induction.

It should be noted that, multistrategy learning in the context described here is not fully automatic, but rather, is a collaboration between human and machine. In principle, it is possible to give systems like Progol very little guidance about what kinds of predicates to try to use in the saturated clause. In practice, however, the run time for an unconstrained search is prohibitive for large problems. Thus, the user normally provides mode declarations to specify the type of predicate to look for.

Background Knowledge and Generalisation

Having seen some specific examples of the use of background knowledge in multistrategy learning, what lesson can be learned?

Often when one sees many specialised strategies doing subtly different things, one is tempted to ask if there is some underlying principle that is common to all cases.

To illustrate this, let us consider Michalski's (1983) categorisation a number of generalisation operations. For example, the "climbing a generalisation tree" is given as a distinct generalisation operator. In the following, we define a simple type hierarchy in terms of Horn clauses.

```
living_thing(X) :- plant(X).
living_thing(X) :- animal(X).

animal(X) :- mammal(X).
animal(X) :- fish(X).

mammal(X) :- elephant(X).
mammal(X) :- whale(X).
```

Suppose 'fred' is an example of the concept 'p' which we wish to learn. Fred has the property that it is an elephant.

```
p(fred) :- elephant(fred).
```

Saturation proceeds as follows, *mammal(fred)* is true since elephant(fred) is given. Having concluded *mammal(fred)*, *animal(fred)* follows and consequently so does *living_thing(fred)*. A system like Progol would construct a *most specific clause* of the form:

```
p(fred) :-
        elephant(fred),
        mammal(fred),
        animal(fred),
        living_thing(fred).
```

It then becomes a matter for the search procedure to determine which subset of literals in the *most specific clause* is the target concept. Thus appropriate background

[1] Prolog's standard notation for the logial 'or' is ';'. We use this symbol and '|' interchangably.

knowledge gives us generalisation by climbing a type hierarchy. In what way does this differ from the following implementation of the "closing the interval" rule?

The background knowledge consists of the clauses:

X in L..H :- X =< L, L =< H.

and the example,

p(2, 1, 3).

generalises to:

p(N, X, Y) :- N in X..Y.

since *2 in 1..3* is satisfied. Thus the "closing the interval" rule can also be obtained from background knowledge.

In fact, linear regression and regression trees can also be seen as generalisation rules since they are implemented in exactly the same way as these "basic" generalisation rules are. So the main lesson to be learned is that while different forms of generalisation may be appropriate for different types of data, these forms can be described using a uniform representation. This means that they can all share a common mechanism for applying them. This, in turn, leads to a very flexible framework for learning.

Scientific Discovery

Induction and scientific discovery are often thought of as related but separate sub-fields of machine learning. This need not be the case. It is clear from the use of linear regression that finding models for numerical data can be accommodated in an induction setting. Indeed, a model finding algorithm becomes another generalisation operation. The search for combinations of variables to create formulae can be guided by background knowledge to indicate what classes of formulae to look for.

Srinivasan (personal communication) has used a scheme similar to the one described for flight trajectories to learn the equations of motion for a pole and cart system. A simulation of a pole and cart was run with random actions applied to push the cart left or right. A trace of the systems behaviour was input to Progol with the result that equations of motion predicting the next state from the current state were synthesised.

Discussion

This discussion as taken place in the setting of inductive logic programming. However, the issues raised here go beyond a debate about the suitability or otherwise of logic as a representation language. The key issue are that:

- the background knowledge and the concept description language should be the same and
- the expressions in the language should be executable as programs.

Uniformity of representation ensures that knowledge can be reused. That is, learned concepts can be added to background knowledge for further learning. The user can also provide pre-defined background knowledge and the system will not have to know what is user defined and what has been learned because both are treated the same way.

If the representation language is executable and therefore can be used to write programs, the background knowledge can describe very powerful concepts, including other learning and data modelling algorithms.

It happens that Horn clause logic has these features. For all the limitations of this form of representation, it does provide an excellent framework for building flexible learning systems.

Acknowledgments

Thanks to Ashwin Srinivasan and Mike Bain for their assistance in understanding the intricacies of Progol.

References

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. 1984. *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group.

Cendrowkska, J. 1987. An algorithm for Inducing Modular Rules. *International Journal of Man-Machine Studies*, 27(4): 349-370.

Gaines, B. R. 1989. An ounce of knowledge is worth a ton of data. In A. M. Segre (Ed.), *Proceedings of the Sixth International Workshop on Machine Learning*, 156-159. Ithaca, New York: Morgan Kaufmann.

Michalski, R. S. 1973. Discovering Classification Rules Using Variable Valued Logic System VL1. In *Third International Joint Conference on Artificial Intelligence*, 162-172.

Michalski, R. S. 1983. A Theory and Methodology of Inductive Learning. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine Learning: An Artificial Intelligence Approach*. Palo Alto: Tioga.

Michalski, R. S., Mozetic, I., Hong, J., & Lavrac, N. 1986. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of AAAI-86*, Philadelphia: Morgan Kaufmann.

Michie, D., Spiegelhalter, D. J., & Taylor, C. C. (Ed.). 1994. *Machine Learning, Neural and Statistical Classification*. Elis Horwood.

Muggleton, S. 1987. Duce, An oracle based approach to constructive induction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 287-292. Milan: Morgan Kaufmann.

Muggleton, S. 1995. Inverse entailment and Progol. *New Generation Computing*. 13:245-286.

Quinlan, J. R. 1979. Discovering rules by induction from large collections of examples. In D. Michie (Eds.), *Expert Systems in the Micro-Electronic Age*. Edinburgh: Edinburgh University Press.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Sammut, C. A. 1981. Concept Learning by Experiment. In *Seventh International Joint Conference on Artificial Intelligence*. Vancouver.

Sammut, C. A., & Banerji, R. B. 1986. Learning Concepts by Asking Questions. In R. S. Michalski Carbonell, J.G. and Mitchell, T.M. (Eds.), *Machine Learning: An Artificial Intelligence Approach, Vol 2*. 167-192. Los Altos, California: Morgan Kaufmann.

Sammut, C., Hurst, S., Kedzier, D., & Michie, D. 1992. Learning to Fly. In D. Sleeman & P. Edwards (Ed.), *Proceedings of the Ninth International Conference on Machine Learning*, Aberdeen: Morgan Kaufmann.

Shapiro, A. 1987. *Structured Induction in Expert Systems*. Addison-Wesley.

Srinivasan, A & Camacho, R. Numerical Reasoning in ILP. In S. Muggleton, K. Furukawa & D. Michie (Ed.), *Machine Intelligence 15*. Oxford University Press. Forthcoming

A Multistrategy Approach to Relational Knowledge Discovery in Databases

Katharina Morik and **Peter Brockhausen**
Univ. Dortmund, Computer Science Department, LS VIII
D–44221 Dortmund
{morik, brockh}@ls8.informatik.uni-dortmund.de

Abstract

When learning from very large databases, the reduction of complexity is of highest importance. Two extremes of making knowledge discovery in databases (KDD) feasible have been put forward. One extreme is to choose a most simple hypothesis language and so to be capable of very fast learning on real-world databases. The opposite extreme is to select a small data set and be capable of learning very expressive (first-order logic) hypotheses. A multistrategy approach allows to combine most of the advantages and exclude most of the disadvantages. More simple learning algorithms detect hierarchies that are used in order to structure the hypothesis space for a more complex learning algorithm. The better structured the hypothesis space is, the better can learning prune away uninteresting or losing hypotheses and the faster it becomes.

We have combined inductive logic programming (ILP) directly with a relational database. The ILP algorithm is controlled in a model-driven way by the user and in a data-driven way by structures that are induced by three simple learning algorithms.

Introduction

Knowledge discovery in databases (KDD) is an application challenging machine learning because it has high efficiency requirements with yet high understandability and reliability requirements. First, the learning task is to find all valid and non–redundant rules (rule learning). This learning task is more complex than the concept learning task as was shown by Uwe Kietz (Kietz 1996). To make it even worse, second, the data sets for learning are very large.

Two extremes of making KDD feasible have been put forward. One extreme is to choose a most simple hypothesis language and to be capable of very fast learning on real-world databases. Fast algorithms have been developed that generalize attribute values and find dependencies between attributes. These algorithms are capable of directly accessing a database, i.e. the representation language $\mathcal{L}_\mathcal{E}$ is the language of the database. The APRIORI and APRIORITID algorithms find *association rules* that determine subsets of correlated attribute values. Attribute values are represented such that each attribute value becomes a Boolean attribute, indicating whether the value is true or false for a certain entity (Agrawal *et al.* 1996). Rules are formed that state

If a set of attributes is true, then also another set of attributes is true.

As all combinations of Boolean attributes have to be considered, the time complexity of the algorithm is exponential in the number of attributes. However, in practice the algorithm takes only 20 seconds for 100 000 tuples [1].

Other fast learning algorithms exploit given hierarchies and generalize attribute values by climbing the hierarchy (Michalski 1983), merging tuples that become identical, and drop attributes with too many distinct values that cannot be generalized. The result are rules that characterize all tuples that have a certain value of attribute A in terms of generalized values of other attributes (Cai, Cercone, & Han 1991). Similarly, the KID3 algorithm discovers dependencies between values of two attributes using hierarchies from background knowledge (Piatetsky-Shapiro 1991). The result is a set of rules of the form

$A = a' \rightarrow cond(B)$

where a' is a generalized attribute value (i.e,. it covers a set of attribute values) of attribute A.

It is easy to see that more complex dependencies between several attributes cannot be expressed (and, hence, cannot be learned) by these fast algorithms. In particular, relations between attribute values of different attributes cannot be learned. For instance, learning a rule stating that

If the value of $A \leq$ *the value of* B
then the value of $C = c$ (1)

[1] In (Agrawal *et al.* 1996) the authors present a series of experiments with the algorithms and give a lower bound for finding an association rule.

requires the capability of handling relations. Hence, these fast learning algorithms trade in expressiveness for the capability of dealing with very large data sets.

The other extreme of how to make KDD feasible is to select a small subset from the data set and learn complex rules. This option is chosen by most algorithms of inductive logic programming (ILP), which are applied to the KDD problem. The rule learning task has been stated within the ILP paradigm by Nicolas Helft (Helft 1987) using the logic notion of minimal models of a theory $\mathcal{M}^+(Th) \subseteq \mathcal{M}(Th)$:
Given observations $\mathcal{E}$ in a representation language $\mathcal{L}_\mathcal{E}$ and background knowledge $\mathcal{B}$ in a representation language $\mathcal{L}_\mathcal{B}$,
find the set of hypotheses $\mathcal{H}$ in $\mathcal{L}_\mathcal{H}$, which is a (restricted) first-order logic, such that

(1) $\mathcal{M}^+(\mathcal{B} \cup \mathcal{E}) \subseteq \mathcal{M}(H)$

(2) for each $h \in \mathcal{H}$ there exists $e \in \mathcal{E}$ such that $\mathcal{B}, \mathcal{E} - \{e\} \not\models e$ and $\mathcal{B}, \mathcal{E} - \{e\}, h \models e$ (necessity of h)

(3) for each $h \in \mathcal{L}_\mathcal{H}$ satisfying (1) and (2), it is true that $\mathcal{H} \models h$ (completeness of $\mathcal{H}$)

(4) $\mathcal{H}$ is minimal.

This learning task is solved by some systems (e.g., RDT (Kietz & Wrobel 1992), CLAUDIEN (De Raedt & Bruynooghe 1993)), LINUS (Lavrac & Dzeroski 1994) and INDEX (Flach 1993)). For the application to databases the selected tuples are re-represented as (Prolog) ground facts. In general, ILP algorithms trade in the capability to handle large data sets for increased expressiveness of the learning result.

Given the trade-off between feasibility and expressiveness, we propose a multistrategy approach. The idea is to combine different computational strategies for the same inferential strategy (here: induction) [2]. The overall learning task is decomposed into a sequence of learning tasks. Simpler subtasks of learning can then be performed by simpler (and faster) learning methods. The simpler algorithms induce hierarchies of attributes and attribute values that structure the hypothesis space for the ILP learner. The ILP learning algorithm uses this structure for its level-wise refinement strategy. The architecture of our MSL-system for KDD is shown in figure 1.

The learning algorithms and their transmutations (Michalski 1994) are:

[2]According to (Michalski 1994) the *computational strategy* means the type of knowledge representation and the associated methods for modifying it in the process of learning. The *inferential strategy* means the primary type of inference underlying a learning process.

FDD : learning functional dependencies between database attributes by an association transmutation. The functional dependencies constitute a *more general* relationship of database attributes. This relationship determines a sequence of more and more detailed hypothesis languages.

NUM_INT : learning a hierarchy of intervals of linear (numerical) attribute values by an agglomeration transmutation. From this hierarchy, the user selects the most interesting intervals which are then introduced as more abstract attribute values into the hypothesis language.

STT : learning a hierarchy of nominal attribute values from background knowledge by an agglomeration transmutation. More abstract attribute values are introduced into the database (database aggregation).

RDT/DB : learning rules in a restricted first-order logic by a generalization transmutation. RDT/DB searches in a hypothesis space that is tailored to the application by the user and the preceding three algorithms.

The paper is organized as follows. First, the RDT algorithm is summarized and it is shown how we enhanced it to become RDT/DB which directly accesses relational databases via SQL. The time complexity of RDT/DB is presented in terms of the size of its hypothesis space. The analysis of the hypothesis space indicates, where to further structure the hypothesis space in order to make learning from very large data sets feasible. Second, the algorithms that preprocess data for RDT/DB are characterized, namely FDD, NUM_INT, STT. Third, we discuss our approach with respect to related work and applications. We argue that multistrategy learning is important for KDD applications.

Applying ILP to Databases

ILP rule learning algorithms are of particular interest in the framework of KDD because they allow the detection of more complex rules such as the one presented above. Until now, however, they have not been applied to commonly used relational database systems. Since the demand of KDD is to analyze the databases that are in use, we have now enhanced RDT to become RDT/DB, the first ILP rule learner that directly interacts with a database management system.

RDT/DB

RDT/DB uses the same declarative specification of the hypothesis language as RDT does in order to restrict

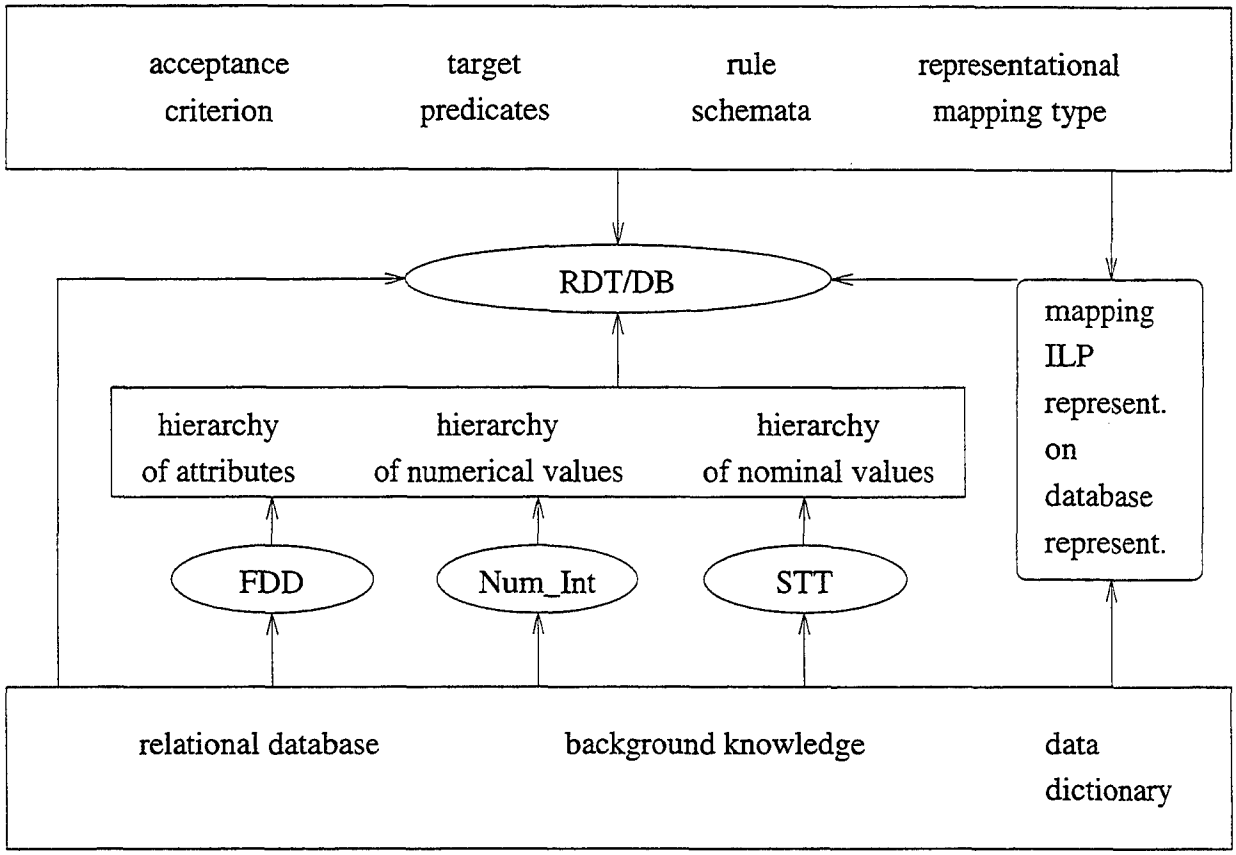


Figure 1: A multistrategy framework for KDD

the hypothesis space (see for details (Kietz & Wrobel 1992)). The specification is given by the user in terms of rule schemata. A rule schema is a rule with predicate variables (instead of predicate symbols). In addition, arguments of the literals can be designated for learning constant values. A simple example of a rule schema is:
$mp_two_c(C, P1, P2, P3)$:
$P1(X1, C)\&P2(Y, X1)\&P3(Y, X2) \rightarrow P1(X2, C)$

Here, the second argument of the conclusion and the second argument of the first premise literal is a particular constant value that is to be learned.

For hypothesis generation, RDT/DB instantiates the predicate variables and the arguments that are marked for constant learning. A fully instantiated rule schema is a rule. An instantiation is, for instance,
$region(X1, europe)\&licensed(Y, X1)\&produced(Y, X2)$
$\rightarrow region(X2, europe)$

In the example, it was found that the cars which are licensed within Europe have also been produced within Europe.

The rule schemata are ordered by generality: for every instantiation of a more general rule schema there exist more special rules as instantiations of a more special rule schema, if the more special rule schema can be instantiated at all. Hence, the ordering reflects the generality ordering of sets of rules. This structure of the hypothesis space is used when doing breadth-first search for learning. Breadth-first search allows to safely prune branches of sets of hypotheses that already have too few support in order to be accepted.

Another user-given control knowledge is the acceptance criterion. The user composes it of $pos(H)$, the number of supporting tuples, $neg(H)$, the number of contradicting tuples, and $concl(H)$, the number of all tuples for which the conclusion predicate of the hypothesis holds. The user can enforce different degrees of reliability of the learning result, or, to put it the other way around, tolerate different degrees of noise. A typical acceptance criterion is, for instance, $(pos(H)/concl(H) - (neg(H)/concl(H)) \geq 0.8$.

For RDT/DB we have developed an interaction model between the learning tool and the ORACLE database system. The data dictionary of the database system informs about relations and attributes of the database. This information is used in order to automatically map database relations and attributes to predicates of

RDT's hypothesis language. Note, that only predicate names and arity are stored in RDT/DB as predicate declarations, not a transformed version of the database entries! Hypothesis generation is then performed by the learning tool, instantiating rule schemata top-down breadth-first. For hypothesis testing, SQL queries are generated by the learning tool and are sent to the database system. For instance, the number of supporting tuples, $pos(H)$, for the rule above is determined by the following statement:

```
SELECT COUNT (*)
       FROM  vehicles veh1, vehicles veh2,
             regions reg1, regions reg2
       WHERE reg1.place = veh1.produced_at
             and veh1.ID = veh2.ID
             and veh2.licensed = reg2.place
             and reg1.region = 'europe'
             and reg2.region = 'europe';
```

The number of contradicting tuples, $neg(H)$, is determined by the negation of the condition which correspond to the rule's conclusion:

```
SELECT COUNT (*)
       FROM  vehicles veh1, vehicles veh2,
             regions reg1, regions reg2
       WHERE reg1.place = veh1.produced_at
             and veh1.ID = veh2.ID
             and veh2.licensed = reg2.place
             and reg2.region = 'europe'
             and not reg1.region = 'europe';
```

The database with two relations being:
vehicles:

ID	produced_at	licensed
fin_123	stuttgart	ulm
fin_456	kyoto	stuttgart
...	...	...

regions:

place	region
stuttgart	europe
ulm	europe
kyoto	asia

The counts for $pos(H), neg(H)$, and $concl(H)$ are used for calculating the acceptance criterion for fully instantiated rule schemata.

Analysis of the Hypothesis Space

The size of the hypothesis space of RDT/DB does not depend on the number of tuples, but on the number of rule schemata, r, the number of predicates that are available for instantiations, p, the maximal number of literals of a rule schema, k. For each literal, all predicates have to be tried. Without constant learning, the number of hypotheses is $r \cdot p^k$ in the worst case. As k is usually a small number in order to obtain understandable results, this polynom is acceptable. Constants to be learned are very similar to predicates. For each argument marked for constant learning, all possible values of the argument (the respective database attribute) must be tried. Let i be the maximal number of possible values of an argument marked for constant learning, and let c be the number of different constants to be learned. Then the hypothesis space is limited by $r \cdot (p \cdot i^c)^k$.

The size of the hypothesis space determines the cost of hypothesis generation. For each hypothesis, two SQL statements have to be executed by the database system. These determine the cost of hypothesis testing.

Here, the size of the hypothesis space is described in terms of the representation RDT/DB uses for hypothesis generation. The particular figures for given databases depend on the mapping from RDT/DB's representation to relations and attributes of the database. An immediate mapping would be to let each database relation become a predicate, the attributes of the relation becoming the predicate's arguments.

Mapping 1: For each relation R with attributes $A_1, \ldots, A_n$, a predicate $r(A_1, \ldots, A_n)$ is formed, r being the string of R's name.

Learning would then be constant learning, because it is very unlikely that a complete database relation determines another one. Hence, p would be the number of relations in the database. This is often a quite small number. However, c would be the maximal number of attributes of a relation! All constants in all combinations would have to be tried. Hence, this first mapping is not a good idea.

If we map each attribute of each relation to a predicate, we enlarge the number of predicates, but we reduce constant learning.

Mapping 2: For each relation R with attributes $A_1, \ldots, A_n$, where the attributes $A_j, \ldots, A_l$ are the primary key, for each $x \in [1, \ldots, n] \backslash [j, \ldots, l]$ a predicate $r_AX(A_j, \ldots, A_l, A_x)$ is formed, where AX is the string of the attribute name.

If the primary key of the relation is a single attribute, we get two–place predicates. The number of predicates is bounded by the number of relations times the maximal number of attributes of a relation (subtracting the number of key attributes). Since the number of constants to be learned cannot exceed the arity of predicates, and because we never mark a key attribute

for constant learning, c will be at most 1. This second mapping reduces the learning task to learning k–place combinations of constant values.

A third mapping achieves the same effect. Attribute values of the database are mapped to predicates of RDT/DB.

Mapping 3: For each attribute A_i which is not a primary key and has the values $a_1, \ldots, a_n$ a set of predicates $r_AI_ai(A_j, \ldots, A_l)$ are formed, $A_j, \ldots, A_l$ being the primary key.

It becomes clear that the restriction of the hypothesis space as it is given by RDT/DB's hypothesis generation can lead to very different hypothesis spaces depending on the mapping from database attributes to predicates. Only when reducing the learning task from finding all valid combinations of attribute values to finding k–place combinations, learning becomes feasible on large databases.

The analysis of the hypothesis space gives good hints for how to write rule schemata: the most general ones should have only one premise literal and the most special ones not more than 3 as this keeps k small; there should be as few schemata as possible in order to keep r small; at most one constant should be marked for learning in order to keep c small.

Further Control of Complexity

Given the declarative syntactic bias in terms of rule schemata, the hypothesis space in a real-world application can still be very high, because the number of attributes (determining p) and attribute values i is usually high. This leads to the demand of restricting the number of attributes and attribute values used for learning.

Some attributes are of particular interest for the user of the KDD system. For instance, in an analysis of warranty cases, the attribute that expresses, whether an item was a warranty case or not, is the most relevant one. The user can specify this attribute as the target for learning. However, the attributes that determine a warranty case cannot be specified by the user. It is the task of learning to identify them! This is the point where FDD comes into play. FDD learns a partial generality ordering on attributes. A sequence of learning passes of RDT/DB is started, each pass using only the most general and not yet explored attributes of the attribute hierarchy for characterizing the user-given target attributes. The sequence is stopped as soon as hypotheses are found that obey the acceptance criterion. That means, after successfully learning in language $\mathcal{L}_{\mathcal{H}_i}$, no new learning pass with $\mathcal{L}_{\mathcal{H}_{i+1}}$ is started. Of course, the user may start RDT/DB with the next language and continue the sequence of learning passes. Most important is, however, that the user gets an output of learned rules, in time, because p decreases remarkably. Note, that the representational bias in terms of the sequence of $\mathcal{L}_{\mathcal{H}_i}$ is a *semantic* declarative bias (as opposed to the syntactic rule schemata or the language sequences of CLINT (De Raedt 1992)). It is domain-dependent. Using the output of FDD, RDT/DB adapts its behavior to a new data set.

Reducing the number of attribute values of an attribute can be done by climbing a hierarchy of more and more abstract attribute values. If this background knowledge is not available, it has to be learned. For numerical values, NUM_INT finds a hierarchy of intervals. Since this is a learning result in its own right, it is presented to the user who selects relevant intervals. These are transformed by RDT/DB into predicates that are used instead of the ones that would have been formed on the basis of the original database attribute values (according to the third mapping). Hence, p slightly increases, but i decreases a lot.

For nominal values, any fast learning algorithm that is capable of finding clusters within the values of one attribute (i.e. finds sets of attribute values) could be plugged into our multistrategy framework. The clusters are named and these names become more abstract attribute values replacing the original values. In our current application (see below), we have chosen a different approach. We have background knowledge about three different aspects of the attribute values of a given attribute A in the database. The background knowledge is used for learning a graph, where the nodes contain attribute values of the given attributes and the links establish a subset ordering. Note, that the graph forms clusters that combine two different aspects. Since the sets of attribute values are meaningful for the user, he can select a level of the graph. Each node of the chosen level becomes a binary attribute of the database. The new attributes replace the original database attribute A in $\mathcal{L}_{\mathcal{H}_i}$. Again, p slightly increases, but i decreases a lot.

Detecting Functional Dependencies — FDD

In the following we assume some familiarity with definitions of relational database theory (see, e.g., (Kanellakis 1990)). Capital letters like A, B, C denote attributes and X, Y, Z sets of attributes. A functional dependency (FD) $X \rightarrow Y$ is valid, if every pair of tuples, which agrees in its X values, also agrees in its Y values. According to Armstrongs's Axioms (Ullman 1988) and without loss of generality we only regard FDs with one attribute on the right hand side. The

discovery of FDs may be visualized as a search in semi lattices. The nodes are labeled with data dependencies and the edges correspond to the *more general than* relationship as in (Savnik & Flach 1993), which implies the partial ordering.

Definition 1 *(More general FD) Let X and Y be sets of attributes such that $X \subseteq Y$, then the FD $X \rightarrow A$ is more general than the dependency $Y \rightarrow A$, or $Y \rightarrow A$ is more specific than $X \rightarrow A$.*

In contrast to the notion of a minimal cover in database theory, the algorithm FDD computes a *most general cover*. The difference is shown by the following example: The set $\{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ is most general in our sense, but not minimal according to database theory, since the transitivity rule is applicable.

Definition 2 *(Most general cover) The set of functional dependencies F of relation R ($R \models F$) is a most general cover, if for every dependency $X \rightarrow A \in F$, there does not exist any Y with $Y \subset X$ and $Y \rightarrow A \in F$.*

The hypothesis generation is a top–down, breadth–first search through the semi lattice imposed by the *more general than* relationship as in (Mannila & Räihä 1994). In order to speed up computation, we use a subset of the complete axiomatization of functional and unary inclusion dependencies and independencies (Bell 1995). Furthermore, FDD computes a partition of all attributes into three classes. The first class contains candidate key attributes, the second contains attributes with null values and only the attributes of the third class (the rest) are needed for discovery. Before actually starting the top–down search, the most specific hypothesis will be tested. If this hypothesis does not hold, then the whole semi lattice can be discarded.

FDD uses the interaction model that was described above, i.e. FDD generates SQL queries (i.e. hypotheses) and the database system computes the answer (i.e. tests the hypothesis). Figure 2 lists this kind of statements and the condition which must hold. The clue is the GROUP BY instruction. The computational costs of this operation are dependent on the database system, but it can be done in time $O(m * \log m)$.

We define a hierarchy on the involved attributes of one–place FDs in the following way:

Definition 3 *(More general attribute) Given a set of one–place functional dependencies F. The attribute C is more general than A, if $A \rightarrow C$ is an element of the transitive closure of F and $C \rightarrow A$ is not an element of the transitive closure of F.*

- ```
  SELECT MAX (COUNT (DISTINCT B))
  FROM R
  GROUP BY A1, ..., An                    =:a1
  ```
- $a_1 = 1 \Rightarrow A_1 \ldots A_n \rightarrow B$

Figure 2: A SQL query for the computation of functional dependencies, ($B \notin \{A_1 \ldots A_n\}$)

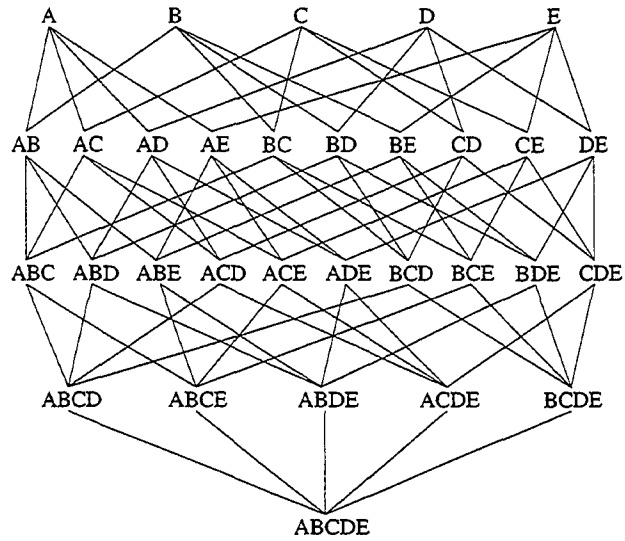


Figure 3: Search lattice for FDD.

We present a simple example to illustrate this. Let the only valid FDs in R be the following: $\{A \rightarrow B, B \rightarrow C\}$. Then we will get a hierarchy of attributes, where C is more general than B, and B more general than A. Since the three attributes are from the same relation and the FDs hold, there must be more tuples with the same value for C than for B. This follows immediately from the definition of FDs. The same is true for B and A. Furthermore, if there are cycles in the transitive closure of these one–place FDs, then all attributes within the cycle are of equal generality. Attributes entering the cycle are more general than the ones within it. Attributes leaving the cycle are more specific than attributes within it.

Although the time complexity of FDD is exponential in the worst case (Beeri *et al.* 1984), in practice, FDD successfully learned from several databases of the size of 65 000 tuples, up to 18 relations, and up to 7 attributes each. Even without any depth restriction – although most of the FDs actually were one-place dependencies – FDD takes 3.5 minutes for learning 113 FDs from 18 relations with up to 6 attributes each.
  ```

We exploit this *more general* relationship on attributes for the development of a sequence of hypothesis languages. Each stays within the same syntactical structure as given by the rule schemata, but has only a subset of the attributes. Given, for instance, the FDs $\{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$, the first set of predicates in $\mathcal{L}_{\mathcal{H}_1}$ includes C and neither A nor B, $\mathcal{L}_{\mathcal{H}_2}$ includes B and neither A nor C, and $\mathcal{L}_{\mathcal{H}_3}$ includes A and neither B nor C. As a result, we have a level–wise refinement strategy as in (Mannila & Toivonen 1996), that means, we start the search with hypotheses consisting of most general attributes. If these hypotheses are too general, continue with more specific attributes only.

Discretization of Numerical Attributes — NUM_INT

Numerical values offer an ordering that can be exploited. Hence, these values can be grouped with less complexity than nominal values, even if no classification is available. The idea behind NUM_INT is to order the numbers and to search for "gaps" in the stream of values. The biggest gap is supposed to be the best point for splitting up the initial interval [min, max]. The next gaps are taken to continue splitting in a top–down fashion. The result is a tree with the initial interval [min, max] as the root, split intervals as inner nodes and unsplit intervals as leaves. The depth of this tree is determined by a parameter (d) which is set by the user.

The result is obtained by three conceptual steps: First, order the numbers (using the statement "select ...order by ..." via embedded SQL); second, fetch tuple by tuple (via embedded SQL) gathering all information needed for splitting; and third, build up the tree of intervals. The complexity of the three steps is as follows: Step (1) should be $\mathcal{O}(N \log N)$, N being the number of tuples, because we select only one attribute (this is done by the database system and therefore beyond our control). In step (2) each gap has to be sorted into an array of depth d which leads to $\mathcal{O}(N \cdot d)$. Finally in step(3) we have to insert $\mathcal{O}(d)$ values into an ordered array of depth $\mathcal{O}(d)$ resulting in complexity of $\mathcal{O}(d^2)$.

Most time is consumed by simply fetching the tuples one by one via SQL. We tested NUM_INT on a database containing about 750.000 tuples: the algorithm ran 35 minutes on a SUN SPARC for a depth of 100: 2 minutes were taken for ordering, 8 minutes for internal processing and about 25 minutes for waiting on the database system to deliver the tuples. This also shows that it is essential that we touch each tuple only once and collect all information (min, max, found intervals, number of tuples in each interval) "on the fly".

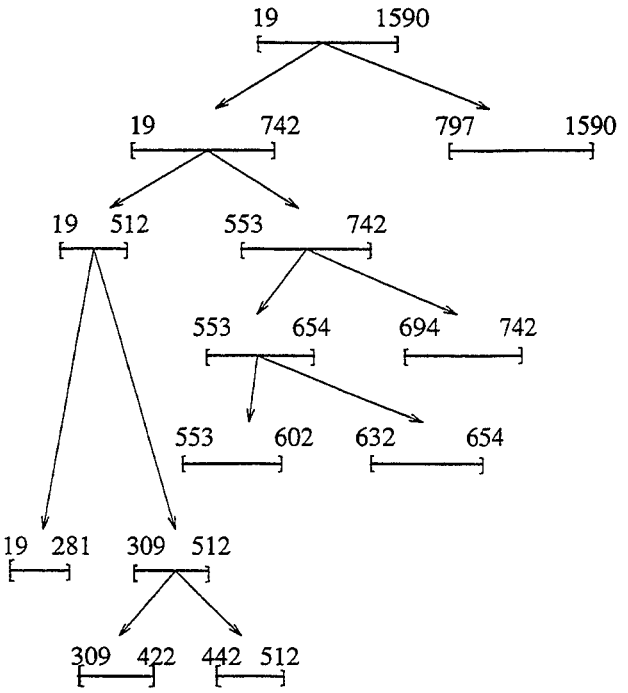


Figure 4: Hierarchy of intervals found by NUM_INT

Of course we are aware of the fact that this "gap"–approach is a quite simple one and that more sophisticated approaches for learning intervals are known (e.g., (Wettscherek & Dietterich 1995), (Pazzani 1995)). However, these algorithms are either more complex (in particular, clustering algorithms although being much more elegant are too complex to be applicable), or require a classification.

Pazzani, for instance, presented an iterative improvement approach for finding discretizations, i.e. intervals, of numeric attributes. His algorithm starts with a partition into a small number of seed intervals with equal size, and then iteratively uses split or merge operations on the intervals based on error or misclassification costs. In most cases, an appropriate number of intervals is unknown in advance, resulting in some loops of the algorithm and therefore he has to reconsider all values of the attribute. This is unfeasible in large databases. However, these algorithms are either more complex (in particular, clustering algorithms although being much more elegant are too complex to be applicable), or require a classification.

Forming a hierarchy of nominal attribute values – STT

Background knowledge is most often used in a KDD framework in order to structure sets of attribute values, that is, the background knowledge offers a hierarchy of

more and more abstract attribute values. However, background material is often unstructured. In this case, it needs some structuring before it can be used for learning from the database. For this task, we use STT, a tool for acquiring taxonomies from facts (Kietz 1988)[3]. For all predicates it forms sets for each argument position consisting of the constant values which occur at that position. The subset relations between the sets is computed. It may turn out, for instance, that all values that occur as second argument of a predicate p_1 also occur as first argument of predicate p_2, but not the other way around. In this case, the first set of values is a subset of the second one. We omit the presentation of other features of STT. Here, we apply it as a fast tool for finding sets of entities that are described by several predicates in background knowledge.

We have represented textual background material as one–ary ground facts. The predicates express independent aspects of attribute values of an attribute of the database. These attribute values are at argument position. Different predicates hold for the same attribute value. For 738 predicates and 14 828 facts, STT delivered a graph with 273 nodes[4]. The graph combines the different aspects. Selecting a layer with 4 nodes, we introduced 4 new Boolean predicates into the database. This increases p by 3, but decreases i by almost 10 000, since we disregard the original database attribute in $\mathcal{L}_{\mathcal{H}}$.

Discussion

The rule learning algorithm RDT is particularly well suited for KDD tasks, because its complexity is not bound with reference to the number of tuples but with reference to the representation of hypotheses. Its top-down, breadth-first search allows to safely prune large parts of the hypothesis space. The declarative syntactic bias is extremely useful in order to restrict the hypothesis space in case of learning from very large data sets. In order to directly access database management systems, RDT was enhanced such that hypothesis testing is executed via SQL queries. However, the syntactic language bias is not enough for applying RDT to real-world databases without reducing it to the expressiveness of an algorithm such as KID3, for instance. If we want to keep the capability of relational learning but also want to learn from all tuples of a large database, we need more restrictions. They should lead to a reduction of the number p of predicates or the maximal number i of attribute values for an attribute. The restriction should be domain–dependent. The task of structuring the set of attributes as well as the task of structuring sets of attribute values is performed by more specialized learning algorithms. While we are convinced that this work–share between specialized and a more general learning algorithm is a powerful idea, we do not claim that the algorithms for structuring attribute values are in general the best choice. However, our architecture allows to plug in other (better) algorithms, if available.

It is an issue for discussion, whether the user should select appropriate levels from the learned hierarchies of the "service algorithms", or not. We have adopted the position of (Brachman & Anand 1996) that the user should be involved in the KDD process. On the one hand, the selection of one layer as opposed to trying all combinations of all hierarchies makes learning feasible also on very large databases. On the other hand, the user is interested in preliminary results and wants to have control of the data mining process. The user is interested in some classes of hypotheses and does not want to specify this interest precisely as yet another declarative bias. Note, however, that the user in our framework does not need to implement the interaction between the learning result of one algorithm and its impact on the other algorithm. Our multistrategy framework for KDD is a closed-loop learning approach with the user being involved as an oracle. This is a particular difference regarding the KEPLER KDD workbench, which offers a variety of ILP algorithms together with a set–oriented layer for data management (Wrobel *et al.* 1996). KEPLER allows the user to call various learning tools and use the results of one as input to another one. It is a loosely coupled system, where our framework is a tightly coupled one. Another difference is that we have moved to directly accessing databases via SQL.

In the course of an on–going project at Daimler Benz AG on the analysis of their data about vehicles, we have applied our multistrategy learning. The database with all vehicles of a certain type of Mercedes — among them some cases of warranty — is about 2.6 Gigabytes large. It consists of 40 relations with about 40 attributes each. The main topic of interest for the users is to find rules that characterize warranty cases and structure them into meaningful classes. In a monostrategy approach, RDT/DB could well find rules, among them ones that are about 100% correct. However, these rules were not interesting, since they expressed what is known to all mechanics.

[3] A detailed description can be found in (Morik *et al.* 1993).

[4] Since STT took about 8 hours, it cannot be subsumed under the fast algorithms. However, its result is computed only once from the background material which otherwise would have been ignored.

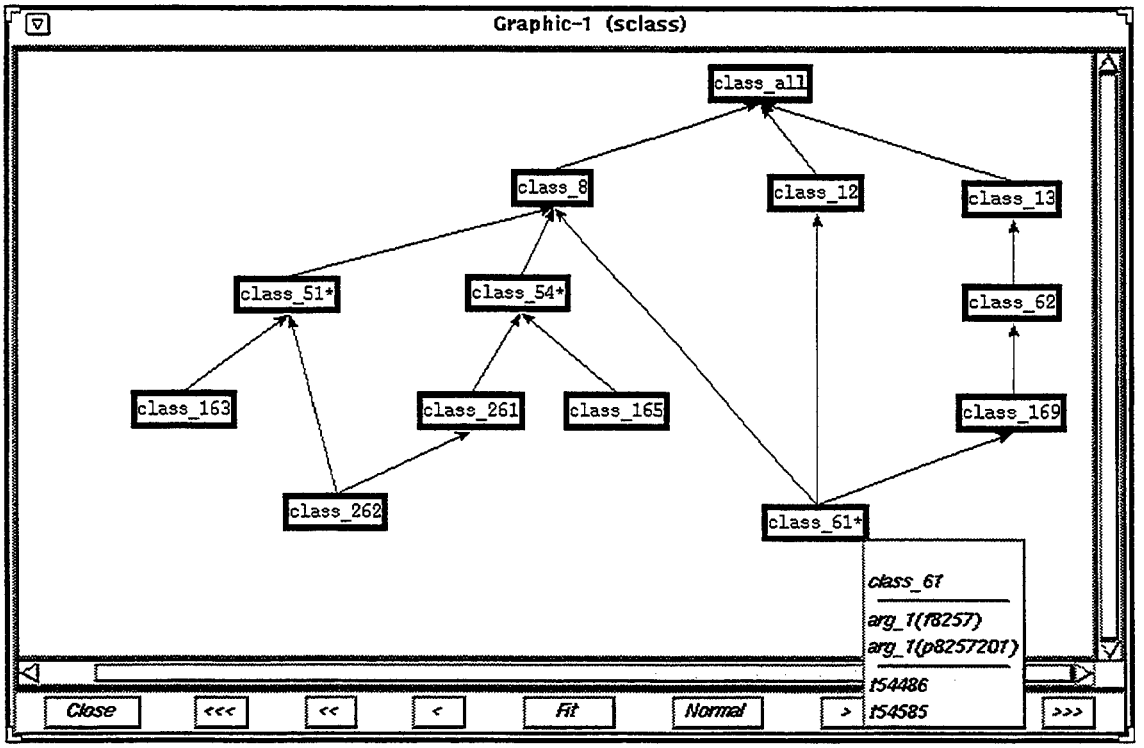


Figure 5: Part of the sort lattice computed by STT.

$engine_variant(VIN, 123) \rightarrow engine_type(VIN, 456)$

More interesting rules could only be found, when preprocessing the data and so focusing RDT/DB on interesting parts of the hypothesis space. Preprocessing achieved even results that were interesting in their own right. FDD found, for instance, that not all variants determine the corresponding type, but only the transmission type determines the transmission variant[5]. Other interesting results are those where the vehicle identification determines a particular vehicle part. Of course, we all know that a car has only one engine. However, for other parts (especially outfit or small technical parts) it is interesting to see, whether they are determined by the vehicle variant, or not.

Also the introduction of the new attributes on the basis of STT's output led to learning more interesting rules. The background material is the mechanic's workbook of vehicle parts, classified by functional (causal) groups of parts, spatial groupings (a part is close to another part, though possibly belonging to another functional group), and possible faults or damages of a part. The vehicle parts are numbered.

[5]As FDs do not allow one single value of the determining attribute to have two values in the determined attribute, the rule found by RDT/DB and shown above is not a FD.

t54486, for instance, is a certain electric switch within the automatic locking device. The functional groups of parts are also numerically encoded. *f8257*, for instance, refers to the locking device of the vehicle. The fact *f8257(t54486)* tells that the switch *t54486* belongs to the locking device. The spatial grouping is given by pictures that show closely related parts. The pictures are, again, numerically encoded. *p8257201*, for instance, refers to a picture with parts of the electronic locking device that are closely related to the injection complex. *p8257201(t54486)* tells that the switch belongs to the spatial group of the injection. Possible damages or faults depend, of course, rather on the material of the part than its functional group. All different types of damages are denoted by different predicates (e.g., *s04* indicates that the part might leak). These three aspects are each represented by several classes and subclasses. Each part belongs at least to three groups (to a functional one, to a spatial one, and a type of possible error), frequently to several subclasses of the same group. The combination of these three aspects has led to surprising classes found by STT. Looking at Figure 5, class_61 comprises two switches, *t54486* and *t54585*. They are the intersection of three meaningful classes:

class_169 : here, several parts of the injection device

are clustered. These are parts such as tubes or gasoline tank. Up in the hierarchy, parts of the functional group of injection and then (class_13) parts of gasoline supply in general are clustered.

class_12 : here, parts of the dashboard are clustered, among them the display of the locking device (protection from theft).

class_8 : here, operating parts of the engine are clustered that serve the injection function.

The illustration shows that mechanics can easily interpret the clusters of parts, and the hierarchy learned by STT is meaningful. The intersection classes are very selective where classes such as, e.g., class_13 cover all parts of a functional group (here: gasoline supply).

The intervals found by NUM_INT in the cost attribute of warranty cases allowed RDT/DB to find 23 rules with an accuracy within the range of 79% to 84%.

Experiments on the data are an ongoing effort [6]. We are planning systematic tests that compare quality and time spent on learning for the various combinations of learning algorithms. Right now, we can state that without using various methods in concert, we achieved valid but not interesting results. Some types of relations could not at all be learned without preprocessing. For instance, no relations with costs of warranty cases could be found before NUM_INT delivered the intervals. Moreover, without the further restrictions by learning results of other learning algorithms, RDT/DB could not use all the tuples. Hence, the advantage of applying the multistrategy approach is not an enhancement of a method that works already, but is that of making relational learning work on real-world databases at all. Since this break-through has been achieved, we can now enhance the algorithms.

Acknowledgments

Work presented in this paper has been partially funded by Daimler–Benz AG, Research Center Ulm, Contract No.: 094 965 129 7/0191. We thank G. Nakhaeizadeh and R. Wirth for indicating practical needs and encouraging us to fulfill these needs using relational learning. Christian Franzel has developed the NUM_INT algorithm. Mark Siebert has acquired background knowledge about vehicle parts and has applied STT to it. We also thank J.–U. Kietz for fruitful discussions.

References

Agrawal, R.; Mannila, H.; Srikant, R.; Toivonen, H.; and Verkamo, A. I. 1996. Fast discovery of association rules. In Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R., eds., *Advances in Knowledge Discovery and Data Mining*, AAAI Press Series in Computer Science. Cambridge Massachusetts, London England: A Bradford Book, The MIT Press. chapter 12, 277–296.

Beeri, C.; Dowd, M.; Fagin, R.; and Statman, R. 1984. On the structure of Armstrong relations for functional dependencies. *Journal of the ACM* 31(1):30–46.

Bell, S. 1995. Discovery and maintenance of functional dependencies by independencies. In *First Int. Conference on Knowledge Discovery in Databases.*

Brachman, R. J., and Anand, T. 1996. The process of knowledge discovery in databases: A human-centered approach. In Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R., eds., *Advances in Knowledge Discovery and Data Mining*, AAAI Press Series in Computer Science. Cambridge Massachusetts, London England: A Bradford Book, The MIT Press. chapter 2, 33–51.

Cai, Y.; Cercone, N.; and Han, J. 1991. Attribute-oriented induction in relational databases. In Piatetsky-Shapiro, G., and Frawley, W., eds., *Knowledge Discovery in Databases.* Cambridge, Mass.: AAAI/MIT Press. 213 –228.

De Raedt, L., and Bruynooghe, M. 1993. A theory of clausal discovery. In Muggleton, S., ed., *Procs. of the 3rd International Workshop on Inductive Logic Programming*, number IJS-DP-6707 in J. Stefan Institute Technical Reports, 25–40.

De Raedt, L. 1992. *Interactive Theory Revision: an Inductive Logic Programming Approach.* Academic Press.

Flach, P. 1993. Predicate invention in inductive data engineering. In Brazdil, P., ed., *Machine Learning - ECML'93*, volume 667 of *Lecture Notes in Artificial Intelligence*, 83–94.

Helft, N. 1987. Inductive generalisation: A logical framework. In *Procs. of the 2nd European Working Session on Learning.*

Kanellakis, P. 1990. *Elements of Relational Database Theory.* Formal Models and Semantics, Handbook of Theoretical Computer Science. Elsevier. chapter 12, 1074–1156.

Kietz, J.-U., and Wrobel, S. 1992. Controlling the complexity of learning in logic through syntactic and task–oriented models. In Muggleton, S., ed., *Inductive Logic Programming.* London: Academic Press. chapter 16, 335–360.

[6] Since the data are strictly confidential, we cannot illustrate the increase of interestingness here.

Kietz, J.-U. 1988. Incremental and reversible acquisition of taxonomies. In *Proceedings of EKAW-88.* chapter 24, 1–11. Also as KIT-Report 66, Technical University Berlin.

Kietz, J. U. 1996. *Induktive Analyse relationaler Daten.* Ph.D. Dissertation. to appear, in german.

Lavrac, N., and Dzeroski, S. 1994. *Inductive Logic Programming - Techniques and Applications.* New York: Ellis Horwood.

Mannila, H., and Räihä, K.-J. 1994. Algorithms for inferring functional dependencies from relations. *Data and Knowledge Engineering* 12:83–99.

Mannila, H., and Toivonen, H. 1996. On an algorithm for finding all interesting sentences. In Trappl, R., ed., *Cybernetics and Systems '96 (EMCSR 1996).*

Michalski, R. S. 1983. A theory and methodology of inductive learning. In *Machine Learning — An Artificial Intelligence Approach.* Los Altos, CA: Morgan Kaufman. 83–134.

Michalski, R. 1994. Inferential theory of learning: Developing foundations for multistrategy learning. In Michalski, R., and Tecuci, G., eds., *Machine Learning - A Multistrategy Approach (IV).* San Francisco: Morgan Kaufmann.

Morik, K.; Wrobel, S.; Kietz, J.-U.; and Emde, W. 1993. *Knowledge Acquisition and Machine Learning - Theory, Methods, and Applications.* London: Academic Press.

Pazzani, M. J. 1995. An iterative improvement approach for the discretization of numeric attributes in Bayesian classifiers. In Fayyad, U. M., and Uthurusamy, R., eds., *The First International Conference on Knowledge Discovery and Data Mining*, 228–233. AAAI Press.

Piatetsky-Shapiro, G. 1991. Discovery, analysis, and presentation of strong rules. In Piatetsky-Shapiro, G., and Frawley, W., eds., *Knowledge Discovery in Databases.* Cambridge, Mass.: AAAI/MIT Press. 229 –248.

Savnik, I., and Flach, P. A. 1993. Bottom-up induction of functional dependencies from relations. In Piatetsky-Shapiro, G., ed., *Proceedings of the AAAI-93 Workshop on Knowledge Discovery in Databases*, 174–185. Menlo Park, California: The American Association for Artificial Intelligence.

Ullman, J. D. 1988. *Principles of database and knowledge–base systems*, volume 1. Rockville, MD: Computer Science Press.

Wettscherek, D., and Dietterich, T. G. 1995. An experimental comparison of the nearest-neighbour and nearest-hyperrectangle algorithms. *Machine Learning* 19(1):5 – 27.

Wrobel, S.; Wettscherek, D.; Sommer, E.; and Emde, W. 1996. Extensibility in data mining systems. submitted paper, available at http://nathan.gmd.de/projects/ml/home.html.

Induction in Logic

Luc De Raedt
Dept. of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A
B-3001 Heverlee, Belgium
Email : Luc.DeRaedt@cs.kuleuven.ac.be

Abstract

Various inductive machine learning approaches and problem specifications are analyzed from a logical perspective. This results in a unifying framework for the logical aspects of inductive machine learning and data mining. The framework explains logical similarities and differences between different machine learning settings, and allows us to relate past and present work on inductive machine learning using logic (such as structural matching, inductive logic programming, data mining, etc.).

Central to the unifying framework are three dimensions: learning from entailment versus learning from interpretations, learning CNF versus DNF, learning characteristic versus discriminant descriptions.

Though the exposition handles both first order and propositional logic, it is meant to be understandable for everyone familiar with propositional logic.

Motivation

Most machine learning and knowledge discovery in databases approaches use logical representation languages to represent data and hypotheses. Despite this uniform representation language, it is often hard to appreciate the differences and similarities in this approaches. In my oppinion this is not only due to the technical nature of some of these papers (in particular in the field of inductive logic programming (Muggleton and De Raedt 1994; Muggleton 1992; De Raedt 1996)), but also to the lack of a unifying framework for these approaches.

In this paper, an attempt is made to build such a unifying framework. The unifying framework builds on past work along at least three different dimensions. The first dimension, learning from entailment versus learning from interpretations (Angluin *et al.* 1992; Frazier and Pitt 1993), determines whether the observations are logical statements that are (resp. are not) logically entailed by the target theory, or are interpretations (or variable-assignments) that satisfy (resp. do not satify) the target theory. Whereas propositional learners learn from interpretations (as for instance in computational learning theory and attribute value learning, (Valiant 1984)), most first order learners learn from entailment (in particular the field of inductive logic programming). Past work along this dimension includes (Angluin *et al.* 1992; Frazier and Pitt 1993; De Raedt and Lavrač 1993). The second dimension distinguishes conjunctive versus disjunctive normal forms, e.g. (Mooney 1995; Haussler 1988; De Raedt and Van Laer 1995). The third dimension is based on the type of learning, i.e. characteristic versus discriminant versus data-mining, and is derived from the work of Michalski (Michalski 1983). Furthermore, the unifying framework allows to give - in logical terms - precise definitions of what is meant by deduction, induction, generalization, specialization, and their relations, not only formalizing these terms but also showing how their relation can be exploited. At the same time, some links between different machine learning settings are formulated in a novel way, in particular inductive logic programming versus structural matching (as studied by e.g. (Vere 1975; Vrain 1990)), and concept-learning versus data-mining.

Though the unifying framework offers some new insights, it extensively borrows from previous work, in particular from (Michalski 1983; 1994) on generalization and specialization and various other aspects, from (Niblett 1988; Kodratoff 1988) on logic and generalization, and from (De Raedt *et al.* 1996; Mooney 1995; De Raedt and Van Laer 1995) on the relation between CNF and DNF.

Though the exposition handles first order as well as propositional logic, it is meant to be understandable for everyone familiar with propositional logic.

This paper is structured as follows: first, a short review of logic is presented, followed by a short review of concept-learning; second, learning from entailment and from interpretations is formalized; third, the relation between generality, induction and logic is discussed, fourth, the differences and relations between CNF and DNF are analysed; fifth, characteristic concept-learning is reformulated as data mining; finally, some conclusions are drawn and relations to

earlier work are presented.

Logic

We first review some notions of logic.

Term A term t is either a constant, a variable or a compound term $f(t_1, ..., t_n)$ composed of a function symbol f, and n different terms t_i.

Atom An atom is a logical formula of the form $p(t_1, .., t_n)$, where p is a predicate symbol and t_i are terms.

Literal A literal is an atom or the negation $\neg A$ of an atom A. Atoms are positive literals, negated atoms are negative literals.

For instance, $pair(card(j, hearts), card(j, diamonds))$, $larger_rank(ace, X)$ and $face_card(j, hearts)$ are atoms, and $\neg face_card(j, hearts)$ is a negative literal.

The main difference between terms and atoms is that one can always assign a truth-value to an atom (not containing variables) whereas a term has no truth-value. In propositional logic, no terms are considered. Thus in propositional logic, only predicate symbols are used in atoms.

CNF A CNF expression is of the form

$$(l_{1,1} \vee ... \vee l_{1,n_1}) \wedge ... \wedge (l_{k,1} \vee ... \vee l_{k,n_k})$$

where all $l_{i,j}$ are propositional literals.

DNF A DNF expression is of the form

$$(l_{1,1} \wedge ... \wedge l_{1,n_1}) \vee ... \vee (l_{k,1} \wedge ... \wedge l_{k,n_k})$$

where all $l_{i,j}$ are propositional literals.

CNF and DNF expressions are well-known in the machine learning literature, as they underly most well-known sytems. They also have a natural upgrade in first order logic, the UCNF and EDNF expressions[1].

UCNF A UCNF expression is of the form

$$(\forall V_{1,1}, ..., V_{1,v_1} : l_{1,1} \vee ... \vee l_{1,n_1})$$
$$\wedge ... \wedge$$
$$(\forall V_{k,1}, ..., V_{1,v_k} : l_{k,1} \vee ... \vee l_{k,n_k})$$

where all $l_{i,j}$ are literals and $V_{i,1}, ..., V_{i,v_i}$ are all variables occurring in $l_{i,1} \vee ... \vee l_{i,n_i}$.

EDNF A EDNF expression is of the form

$$(\exists V_{1,1}, ..., V_{1,v_1} : l_{1,1} \wedge ... \wedge l_{1,n_1})$$
$$\vee ... \vee$$
$$(\exists V_{k,1}, ..., V_{1,v_k} : l_{k,1} \wedge ... \wedge l_{k,n_k})$$

where all $l_{i,j}$ are literals and $V_{i,1}, ..., V_{i,v_i}$ are all variables occurring in $l_{i,1} \wedge ... \wedge l_{i,n_i}$.

[1] In first order logic one might also use CNF or DNF expressions with mixed quantifiers. However, as such prenex normal forms have not yet been considered in the machine learning literature, we will ignore these here.

The symbol $\forall$ reads 'for all' and stands for universal quantification, and $\exists$ reads 'there exists' and stands for existential quantification.

For instance, the formula

$$\exists C, T : triangle(T) \wedge circle(C) \wedge in(C, T)$$

states that there exist a triangle and a circle such that the circle is inside the triangle.

Notice also that UNCF form corresponds to the clausal subset of first order logic. Therefore, we will sometimes write UCNF expressions in clausal notation (where $\leftarrow$ reads 'if' and stands for implication, and each disjunction corresponds to a *clause*). E.g. the UCNF expression :

$$(\forall X : flies(X) \vee \neg bird(X)) \wedge bird(tweety)$$

would be written as:

$$flies(X) \leftarrow bird(X)$$
$$bird(tweety) \leftarrow$$

An interesting subset of clausal logic, is definite clause logic. It consists of UCNF expressions that have exactly one positive literal in each disjunction (or clause). Definite clause logic is the basis of the programming language Prolog and of the machine learning technique known under the name of inductive logic programming.

To reason about the relation among different logical formulae and about what is truth, logicians have developed model theory. Model theory starts with interpretations[2].

Herbrand Interpretation A Herbrand interpretation is a set of ground atoms.

A Herbrand interpretation correponds in the boolean or propositional case to a variable assignment. The meaning of a Herbrand interpretation is that all atoms in the interpretation are true, and all other atoms are false.

Substitution A substitution $\theta = \{V_1 \leftarrow t_1, ..., V_n \leftarrow t_n\}$ is an assignment of terms $t_1, ..., t_n$ to variables $V_1, ..., V_n$.

Applying a substitution The formula $F\theta$ where F is a term, atom, literal or expression, and $\theta = \{V_1 \leftarrow t_1, ..., V_n \leftarrow t_n\}$ is a substitution is the formula obtained by simultaneously replacing all variables $V_1, ..., V_n$ in F by the terms $t_1, ..., t_n$.

By now, we can define truth and falsity of an expression in a Herbrand interpretation.

Literal A ground literal l is true in an interpretation I if and only if l is a positive literal and $l \in I$, or l is a negative literal and $l \notin I$.

[2] For the purposes of this paper, we will only employ Herbrand interpretations, this in order to keep the exposition simple.

UCNF A UCNF expression

$$(\forall V_{1,1},...,V_{1,v_1} : l_{1,1} \vee ... \vee l_{1,n_1})$$

$$\wedge ... \wedge$$

$$(\forall V_{k,1},...,V_{1,v_k} : l_{k,1} \vee ... \vee l_{k,n_k})$$

is true in an interpretation if and only if for all i and for all substitutions θ such that $(l_{i,1} \vee ... \vee l_{i,n_i})\theta$ is ground, at least one of the $l_{i,j}\theta$ is true in I.

EDNF A EDNF expression

$$(\exists V_{1,1},...,V_{1,v_1} : l_{1,1} \wedge ... \wedge l_{1,n_1})$$

$$\vee ... \vee$$

$$(\exists V_{k,1},...,V_{1,v_k} : l_{k,1} \wedge ... \wedge l_{k,n_k})$$

is true in an interpretation I if and only if there exists i and an substitution θ such that $(l_{i,1} \wedge ... \wedge l_{i,n_i})\theta$ is ground, and all $l_{i,j}$ are true in I.

If an expression is true in an interpretation we also say that the interpretation is a model for the expression.

Let us first illustrate this rather complicated definition. It states e.g. $flies \vee \neg bird \vee \neg abnormal$ is true in the interpretations $\{flies\}, \{abnormal\}$ but false in $\{bird, abnormal\}$. Similarly, it allows us to say that

$$\exists C, T : triangle(T) \wedge circle(C) \wedge in(C, T)$$

is true in $\{triangle(t1), circle(c1), in(c1, t1), large(c1), small(t1)\}$ and false in $\{triangle(t1), circle(c1)\}$. Furthermore, $\forall X : polygon(X) \vee \neg square(X)$ is true in $\{square(s1), polygon(s1)\}$ and in $\{circle(c1)\}$ but false in $\{square(s1)\}$.

Model theory is the basis for reasoning about the declarative meaning of logical formulae. It is also used to define logical entailment.

Logical entailment An expression F_1 logically entails an expression F_2 if and only if all models of F_1 are also models of F_2. We will write $F_1 \models F_2$.

Logical entailment is typically verified by theorem provers using the well-known resolution principle, the propositional version of which is defined below.

Resolution - propositional Let $C_1 = l \vee l_1 \vee ... \vee l_n$, $C_2 = l' \vee l'_1 \vee ... \vee l'_m$, C is a resolvent of clauses C_1 and C_2 (denoted by $C \in res(C_1, C_2)$) if and only if $C = l_1 \vee ... \vee l_n \vee l'_1 \vee ... \vee l'_m$ and $l = \neg l'$.

Soundness of resolution For all $C \in res(C_1, C_2)$: $C_1 \wedge C_2 \models C$.

For example, $flies \leftarrow bird \wedge sparrow$ is a resolvent of $flies \leftarrow bird \wedge normal$ and $normal \leftarrow sparrow$.

Inductive Learning

The general definition of *discriminant* inductive concept-learning is the following:

Given:

- a language of hypotheses L_H, which defines the set of a priori acceptable hypotheses
- a language of examples L_e, which defines the set of a priori acceptable examples
- the *covers* relation between L_H and L_e, which defines when an example is covered by a hypothesis, i.e. $covers(H, e)$ is true if and only if the example e is covered by the hypothesis H
- sets of positive and negative examples P and N, expressed in the L_e,

Find: a hypotheses $H \in L_H$ which is

- complete, i.e. covers all positive examples in P
- consistent, i.e. does not cover any of the negative examples in N.

Discriminant learning starts from two classes of examples; its aim is to derive a hypothesis that is able to discriminate the examples as belonging to one of the two classes. In contrast, *characteristic* learning starts from a single class of examples and aims at a maximally specific hypothesis that covers all of the examples in the class. For readability, we formally define this notion as well:

Given:

- a language of hypotheses L_H, which defines the set of a priori acceptable hypotheses
- a language of examples L_e, which defines the set of a priori acceptable examples
- the *covers* relation between L_H and L_e, which defines when an example is covered by a hypothesis, i.e. $covers(H, e)$ is true if and only if the example e is covered by the hypothesis H
- a set of unclassified examples E, expressed in the L_e,

Find: a hypothesis $H \in L_H$ which is

- complete, i.e. covers all examples in E and
- maximally specific in L_H.

The maximally specific hypothesis is the most informative in the sense that it characterizes the examples best, and results in the smallest possible inductive leap.

At the end of this paper, I will argue that the difference between concept-learning and data mining is akin to the difference between discriminant and characteristic induction.

The above two definitions are generally accepted and unify all approaches in concept-learning. This makes them the ideal starting point for a unifying logical framework for induction. As the concept-learning definitions have the representation as a generic parameter, we merely need to instantiate them in order obtain a logical framework.

Induction in Logic

To instantiate the concept-learning framework, we need to make choices for:

- the language of hypotheses, L_H,
- the language of examples, L_e, and
- the *covers* relation.

From the short review of logic, it follows that there are two different choices for the *covers* relation, i.e. entailment and truth in an interpretation. This determines the first dimension.

Learning from interpretations

In this framework the following choices are made: examples are (Herbrand) interpretations and a hypothesis covers an example if the hypothesis is true in the interpretation :

Hypothesis space The hypothesis space is either the set of all UCNF expressions or the set of all EDNF expressions, i.e. $\mathcal{L}_\mathcal{H}$ = UCNF or $\mathcal{L}_\mathcal{H}$ = EDNF.

Example space The example space is the set of all Herbrand interpretations, positive examples are interpretations in which the target concept is true, negative examples are interpretations in which the target concept is false.

Coverage A hypothesis H covers an example e if and only if H is true in e. So, $covers(H, e)$ is true if and only if H is true in e.

Notice that these notions apply to characteristic as well as discriminant concept-learning.

Historically speaking, learning from interpretations (or a slight variant of it) has been employed by researchers such as (Winston 1975; Vere 1975; Hayes-Roth and McDermott 1978; Ganascia and Kodratoff 1986), who worked on structural matching and learned EDNF expressions, (Valiant 1984; Haussler 1988; Natarajan 1991), who worked on computational learning theory within propositional logic (i.e. without using quantifiers), and (De Raedt and Džeroski 1994; De Raedt and Van Laer 1995), who upgraded Valiant's results on CNF to UCNF, and (Michalski 1983), who – in a sense – learns DNF expressions in this framework.

We now illustrate this framework, first using an example in UCNF, and then using an example from structural matching (EDNF).

Example 1 *An UCNF learning task Consider the following family descriptions:*

$$p_1 = \{human(jef), male(jef), female(an), human(an)\}$$

$$p_2 = \{human(paul), male(paul)\}$$

$$p_3 = \{human(mary), female(mary)\}$$

$$n_1 = \{female(tweety), bird(tweety)\}$$

$$n_2 = \{male(dracula), bat(dracula)\}$$

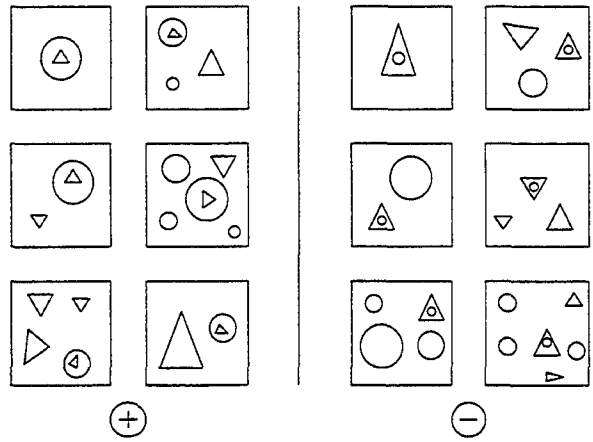


Figure 1: A Bongard Problem

In this example, the p_i are positive examples, the n_j negative ones. Furthermore, assume $\mathcal{L}_\mathcal{H}$ = UCNF. A possible solution for this concept-learning task is then:

$$H = (\forall X : human(X) \vee \neg male(X)) \wedge$$

$$(\forall X : human(X) \vee \neg female(X))$$

This learning problem could be handled by the recent ICL system of (De Raedt and Van Laer 1995).

Example 2 *An EDNF learning task Consider the Bongard problem shown in Figure 1. Some of the examples are represented as follows.*

$$p_1 = \{triangle(t1), circle(c1), in(t1, c1)\}$$

$$p_2 = \{circle(c1), circle(c2), triangle(t1), triangle(t2),$$
$$small(c1), large(c2), small(t1), large(t2), in(t1, c2)\}$$

$$\ldots$$

$$n_1 = \{triangle(t1), circle(c1), in(c1, t1)\}$$

$$n_2 = \{triangle(t1), triangle(t2), circle(c1), circle(c2),$$
$$in(c1, t2), small(c1)\}$$

$$\ldots$$

The target hypothesis can then be formulated as follows:

$$\exists T, C : triangle(T) \wedge circle(C) \wedge in(T, C)$$

This learning problem could be handled by techniques known as structural matching, cf. above.

Learning from entailment

An alternative logical formulation of concept-learning is used in inductive logic programming. In contrast to learning from interpretations, it employs the single-representation trick. In logical approaches to learning, this is often achieved by using the clausal logic subset of first order logic (i.e. UCNF[3].), which also forms the basis of logic programming and the programming language Prolog.

Hypothesis space The hypothesis space is the set of all clausal theories. In practice, one restricts it to definite clause theories.

Example space The example space is the set of all clauses; positive examples are logically entailed by the target hypothesis, negatives ones are not. In practice, one again restricts the example space to definite clauses.

Coverage A hypothesis H covers an example e if and only if H logically entails e. So, $covers(H, e)$ is true if and only if $H \models e$.

Note that the coverage relation in this framework is only semi-decidable. This may lead to various computational properties.

Learning from entailment is the main setting studied in the fashionable inductive logic programming paradigm. It is illustrated in Example 3.

Example 3 *Learning from entailment. Consider the following examples.*

$$p_1 = grandfather(leo, luc) \leftarrow father(leo, rose)$$
$$\wedge mother(rose, luc)$$
$$p_2 = grandfather(rene, lieve) \leftarrow father(rene, william)$$
$$\wedge father(william, lieve)$$
$$n_1 = grandfather(alice, luc) \leftarrow mother(alice, rose)$$
$$\wedge mother(rose, luc)$$
$$n_2 = grandfather(X, X) \leftarrow$$

The following hypothesis is complete and consistent with regard to the positive examples p_1 and p_2 and the negative ones n_1 and n_2.

$$grandfather(X, Y) \leftarrow father(X, Z) \wedge parent(Z, Y)$$
$$parent(X, Y) \leftarrow mother(X, Y)$$
$$parent(X, Y) \leftarrow father(X, Y)$$

This example (or variants thereof) could be handled by inductive logic programming systems, such as e.g. (Muggleton 1995; Quinlan 1990).

[3]Instead of using UCNF, one might also want to use the EDNF representation. However, then it is more natural to reverse the coverage relation, i.e. to define $covers(H, e) = e \models H$. This has for instance been considered in structural matching, as studied by (Kodratoff 1988). However, we do not elaborate on this any further here.

Instead of learning from entailment, we might as well call this framework *open* because – in contrast to learning from interpretations which could be called *closed* – it does not assume complete knowledge about the examples. Indeed, in the first example it may be the case that *leo* is married to *alice* but this need not be specified. This difference can best be illustrated on a small example. Suppose there is a bird called tweety, and we know that tweety does not fly, is an ostrich, is black, but we do not know whether tweety is normal. In the closed framework, this cannot be represented because complete knowledge about tweety is assumed. There are two possible interpretations which are in agreement with our knowledge and this is not permitted[4]. Using clauses, the above example can naturally be represented as the negative example $flies(tweety) \leftarrow bird(tweety) \wedge ostrich(tweety) \wedge black(tweety)$. Moreover in learning from entailment it would be possible to include in the induced hypothesis the clause $abnormal(X) \leftarrow ostrich(X)$. Including this clause in the hypothesis would realize an inductive leap *on the example*, as using this clause, one can write the negative example as $flies(tweety) \leftarrow bird(tweety) \wedge ostrich(tweety) \wedge black(tweety) \wedge abnormal(tweety)$, making additional assumptions about tweety. This is impossible when learning from interpretations. This difference between the closed and open framework has great impact on the complexity of the induction problem. The reason is that in learning from entailment different clauses may make contradicting assumptions about particular examples. Therefore learning from entailment is computationally harder than learning from interpretations. In learning from interpretations, it is true that clauses $c_1 \wedge c_2$ cover an example if and only if clause c_1 and clause c_2 cover the example. In learning from entailment this need not be the case. For instance, $(flies \leftarrow bird \wedge abnormal) \wedge (abnormal \leftarrow ostrich)$ covers $flies \leftarrow bird \wedge ostrich$ whereas the individual clauses do not.

A further difference between the two frameworks concerns the use of implication. Indeed, as examples in learning from entailment are implications each example can be understood in causal terms. For instance, the properties of a bird cause (or imply) that the bird flies (or does not fly). As such learning from entailment promotes certain predicates as the target predicates, which are to be learned. In learning from interpretations all predicates are equal. Therefore, it is easy to represent problems of learning from intepretations as learning from entailment, but not the other way around. For instance, consider the Bongard problem of Figure 1. It can be modelled as a learning from entailment problem as follows :

[4]For completeness sake, we should mention that there exist in the literature few approaches and proposals to cope with such situations (see e.g. (Helft 1989)). However, as these have not received a lot of attention, we will not consider these here.

Example 4 *Reconsider Figure 1 and Example 2. The examples can be represented as follows:*

$$p_1 = \oplus \leftarrow triangle(t1) \wedge circle(c1) \wedge in(t1, c1)$$

$$p_2 = \oplus \leftarrow circle(c1) \wedge circle(c2) \wedge triangle(t1) \wedge$$
$$triangle(t2) \wedge small(c1) \wedge large(c2) \wedge small(t1) \wedge$$
$$large(t2) \wedge in(t1, c2)$$

...

$$n_1 = \oplus \leftarrow triangle(t1) \wedge circle(c1) \wedge in(c1, t1)$$

$$n_2 = \oplus \leftarrow triangle(t1) \wedge triangle(t2) \wedge circle(c1) \wedge$$
$$circle(c2) \wedge in(c1, t2) \wedge small(c1)$$

...

Notice that the negative examples are represented also as clauses for the predicate $\oplus$. However, in contrast to the positive examples, negative examples are false and should not be entailed by the target theory. The target hypothesis can then be formulated as follows:

$$\oplus \leftarrow triangle(T) \wedge circle(C) \wedge in(T, C)$$

Implicit in learning from interpretations is that all facts not stated are false. To model this in learning from entailment one should also add negated atoms in the condition part of the examples. However, in the current state of the art - due to the use of definite clauses in inductive logic programming, this is not considered here.

Using background knowledge

This section may be skipped by the casual reader, less interested in technical aspects. Up till now we have completely ignored the use of background knowledge. Here, we show how the two frameworks can be adapted to take into account background knowledge. In both frameworks we will assume that background knowledge is specified as a definite clause theory B.

Learning from interpretations In learning from interpretations, background knowledge is used to expand the examples into an interpretation. This is usually done by taking the least Herbrand model of the background theory and example. The least Herbrand model of a definite theory consists of all ground facts constructed with the predicate, constant and functor symbols of the theory that are logically entailed by the theory. This can be formalized as follows:

Examples An example is a set of definite clauses. Often one will only use facts.

Coverage A hypothesis H covers an example e w.r.t. a definite clausal background theory B if and only if H is true in $M(B \wedge e)$.

Learning from entailment In learning from entailment, background knowledge can easily be integrated in the induction process as follows.

Coverage A hypothesis H covers an example e w.r.t. background theory B if and only if $B \wedge H \models e$.

Deduction and induction

Now that we have seen how the problem of concept-learning can be formulated in terms of logic, the question arises as how well-known concept-learning techniques and algorithms can be adapted to use logic as their representation language. This question will be answered by first studying the generality relation for the two frameworks, which will then motivate the introduction of logical notions of induction.

Generality and Entailment

It is well-known in the literature (cf. (Mitchell 1982; Michalski 1983; De Raedt and Bruynooghe 1992), that the generality relation is crucial for developing concept-learning algorithms. The generality relation is typically defined in terms of coverage. Indeed, one concept is more general than another concept if the first concept covers all examples that are covered by the second concept. Reformulating this in terms of learning interpretations leads to:

Generality - learning from interpretations
A hypothesis G is more general than a hypothesis S if and only if $S \models G$.

Because of the definition of generality in terms of coverage and the definition of logical entailment, generality and logical entailment coincide when learning from interpretations.

This notion of generality also means that the most specific statement, one can make is $\square$, the inconsistent theory which does not have any model, and the most general statement is the empty hypothesis, for which all interpretations are a model.

Let us now investigate whether this notion of generality also applies to learning from entailment.

It turns out that the generality relation in learning from entailment is the inverse as in learning from interpretations.

Generality - learning from entailment
A hypothesis G is more general than a hypothesis S if and only if $G \models S$.

This property follows from the transitivity of logical entailment. Again we see that – as in learning from interpretations – generality and logical entailment coincide. However, this time entailment is reversed. This means that the most general hypothesis is $\square$, the inconsistent theory, because it logically entails everything. The most specific hypothesis then is the empty theory. Notice also that it is this view of generality that machine learning research has typically adopted (due to its use in the inductive logic programming paradigm).

	generalization	specialisation
interpretations	deduction	induction
entailment	induction	deduction

Induction and deduction

Because the logical framework employed determines whether logical entailment and generality coincide or whether the inverse of logical entailment and generality coincide, it will turn out useful to abstract away from this using deduction and induction:

Deduction - induction equation Let F_1 and F_2 be two logical formulae. If $F_1 \models F_2$ we say that F_2 follows deductively from F_1 or equivalently that F_1 follows inductively from F_2.

As we have seen in the previous section, generalization corresponds to induction when learning from entailment and to deduction when learning from interpretations. Dually, specialisation corresponds to deduction when learning from entailment and to induction otherwise. For instance, let $F_1 = \mathit{flies} \leftarrow \mathit{bird}$, $F_2 = \mathit{flies} \leftarrow \mathit{bird} \wedge \mathit{normal}$. Then $F_1 \models F_2$ and F_1 is a generalization of F_2 when learning from entailment and a specialisation when learning from interpretations. These facts are summarized in the above Table.

The reader may notice that in the induction-deduction equation we view induction as the inverse of deduction. Whereas this viewpoint may appear to be controversial, especially for cognitive scientists and philoshophers, it will prove to be a very operational framework for concept-learning operators.

The question now is how to exploit these dual notions in order to obtain generalisation and specialisation operators for use in concept-learning algorithms such as those presented by (Mitchell 1982; De Raedt and Bruynooghe 1992). As the meaning of deduction (and induction) with regard to the generality relation depends on which logical framework is used, we will talk about deductive and inductive operators rather than about generalisation and specialisation operators. The equation also shows that one deductive operator can be used to perform both generalisation as well as specialisation, depending on the framework. This should clarify terms such as inductive specialisation and deductive generalisation.

Deductive operator A deductive operator ρ maps a formula F_1 onto a set of formulae $\rho(F_1)$ such that for all $F_2 \in \rho(F_1) : F_1 \models F_2$.

Inductive operator An inductive operator ρ maps a formula F_1 onto a set of formulae $\rho(F_1)$ such that for all $F_2 \in \rho(F_1) : F_2 \models F_1$.

When looking at these definitions, it should be clear that deductive operators are well-known in the literature on theorem proving. Many different deductive operators have been studied; usually they are denoted by the symbol $\vdash$ as they implement $\models$ in one way or another. As there exist many deductive operators, and deduction is just the opposite of induction, one can obtain inductive operators by inverting deductive ones. Furthermore, as some of these deductive operators restrict hypotheses in one way or another, the inductive ones will have to work within the same restrictions. Restrictions that are often applied are restrictions on clauses being definite, and also on the number of clauses in the hypotheses.

Using this idea, one can clarify the three main different notions of entailment (i.e. of $\vdash$) used as the basis of operators in inductive logic programming (Muggleton and De Raedt 1994). These are θ-subsumption (Plotkin 1970) which works on single clauses, (inverse) implication (Lapointe and Matwin 1992; Muggleton 1994; Idestam-Almquist 1993) which works on single clauses and inverse resolution (Muggleton and Buntine 1988) which works on sets of (definite) clauses.

θ-subsumption Let c and c' be two clauses. Clause c *θ-subsumes* c' (i.e. $c \vdash c'$) if there exists a substitution θ, such that $c\theta \subseteq c'$[5].

Resolution cf. above.

Implication Let c and c' be two clauses. Then c implies c' (i.e. $c \vdash c'$) if and only if $c \models c'$.

These different options for $\vdash$ result in different deductive and inductive operators used within inductive logic programming. For more information, we refer to (Muggleton and De Raedt 1994).

UCNF and EDNF

We now show that UCNF and EDNF are dual representations for concept-learning. The duality can be exploited at the level of operators, and sometimes as well at the level of algorithms.

UCNF and EDNF operators

To show the duality at the first level, consider a deductive operator ρ for UCNF. We will now show that this operator also can be used as an inductive operator for EDNF.

Indeed, let F_1, F_2 be two UCNF formulae such that $F_2 \in \rho(F_1)$, i.e. $F_1 \models F_2$. Because of the properties of logical entailment this is equivalent to $\neg F_2 \models \neg F_1$. Let

$$F_1 = (\forall l_{1,1} \vee ... \vee l_{1,n_1}) \wedge ... \wedge (\forall l_{r,1} \vee ... \vee l_{r,n_r})$$

$$F_2 = (\forall k_{1,1} \vee ... \vee k_{1,n'_1}) \wedge ... \wedge (\forall k_{r',1} \vee ... \vee k_{r',n'_{r'}}).$$

$$\text{Then} \neg F_1 = \neg\{(\forall l_{1,1} \vee ... \vee l_{1,n_1}) \wedge ... \wedge (\forall l_{r,1} \vee ... \vee l_{r,n_r})\} =$$

$$\neg(\forall l_{1,1} \vee ... \vee l_{1,n_1}) \vee ... \vee \neg(\forall l_{r,1} \vee ... \vee l_{r,n_r}) =$$

$$(\exists \neg l_{1,1} \wedge ... \wedge \neg l_{1,n_1}) \vee ... \vee (\exists \neg l_{r,1} \wedge ... \wedge \neg l_{r,n_r}).$$

[5] In this definition of θ-subsumption, we consider a clause $h_1 \vee ... \vee h_n \leftarrow b_1 \wedge ... \wedge b_m$ as the set of literals $\{h_1, ..., h_n, \neg b_1, ..., \neg b_m\}$.

One can apply the same steps for F_2. This results in :

$$(\forall l_{1,1} \vee ... \vee l_{1,n_1}) \wedge ... \wedge (\forall l_{r,1} \vee ... \vee l_{r,n_r}) \models$$
$$(\forall k_{1,1} \vee ... \vee k_{1,n'_1}) \wedge ... \wedge (\forall k_{r',1} \vee ... \vee k_{r',n'_{r'}})$$

if and only if

$$(\exists \neg k_{1,1} \wedge ... \wedge \neg k_{1,n'_1}) \vee ... \vee (\exists \neg k_{r',1} \wedge ... \wedge \neg k_{r',n'_{r'}}) \models$$
$$(\exists \neg l_{1,1} \wedge ... \wedge \neg l_{1,n_1}) \vee ... \vee (\exists \neg l_{r,1} \wedge ... \wedge \neg l_{r,n_r})$$

For instance,

$$(\forall X : flies(X) \vee \neg normal(X)) \wedge$$
$$(\forall Y : normal(Y) \vee \neg sparrow(X)) \models$$
$$\forall Z : flies(Z) \vee \neg sparrow(Z).$$

Therefore

$$\exists Z : \neg flies(Z) \wedge sparrow(Z) \models$$
$$(\exists X : \neg flies(X) \wedge normal(X)) \vee$$
$$(\exists Y : \neg normal(Y) \wedge sparrow(X))$$

What does this mean ? It means that any deductive operator on UCNF can be mapped into an inductive operator on EDNF (and dually an inductive UCNF on a deductive EDNF, or also dually an inductive EDNF on a deductive UCNF, ...) as follows. First map the EDNF formula onto an UCNF by negating all literals, changing all $\wedge$ into $\vee$, all $\vee$ into $\wedge$, and all $\forall$ into $\exists$, then apply the operator onto the obtained UCNF formula, and then map the resulting UCNF formulae back to EDNF by again negating all literals, changing all $\wedge$ into $\vee$, all $\vee$ into $\wedge$, and all $\forall$ into $\exists$. Formally speaking, this yields:

Mapping UCNF operator on EDNF Let ρ be a deductive (resp. inductive) operator on UCNF. Then ρ^d is an inductive (resp. deductive) operator on EDNF where $\rho^d = f^{-1} \; o \; \rho \; o \; f$, where $f((\exists k_{1,1} \wedge ... \wedge k_{1,n'_1}) \vee ... \vee (\exists k_{r',1} \wedge ... \wedge k_{r',n'_{r'}})) =$ $(\forall \neg l_{1,1} \vee ... \vee \neg l_{1,n_1}) \wedge ... \wedge (\forall \neg l_{r,1} \vee ... \vee \neg l_{r,n_r})$

The definition of the mapping from EDNF to UCNF is left to the reader as an exercise.

Let us illustrate this on an example. Suppose we want to apply induction on the formula

$$(\exists X : \neg flies(X) \wedge normal(X)) \vee$$
$$(\exists Y : \neg normal(Y) \wedge sparrow(X))$$

Then we first use the mapping f, which yields:

$$(\forall X : flies(X) \vee \neg normal(X)) \wedge$$
$$(\forall Y : normal(Y) \vee \neg sparrow(X))$$

Then we apply a deductive operator (in this case resolution) on the UCNF formula, which yields :

$$\forall Z : flies(Z) \vee \neg sparrow(Z)$$

Finally, applying f^{-1} then results in the desired formula

$$\exists Z : \neg flies(Z) \wedge sparrow(Z).$$

Because of this duality between UCNF and EDNF, it suffices to analyse operators for UCNF. Transformers for EDNF can then be obtained by the above mappings. This shows that the operators used within the field of inductive logic programming directly apply to structural matching as well. Using this technique, (De Raedt *et al.* 1996) have modified Plotkin's well-known least general generalization and relative least general generalization operators for use in structural matching. The resulting operator solves some of the problems with existing ones for structural matching.

UCNF and EDNF algorithms

There exists also a more direct mapping between UCNF and EDNF, which is well-known in computational learning theory (cf. (Haussler 1988) and which makes that one should only develop one type of algorithm for *learning from interpretations*. The property is:

$$(\exists l_{1,1} \wedge ... \wedge l_{1,n_1}) \vee ... \vee (\exists l_{k,1} \wedge ... \wedge l_{k,n_k})$$

is a solution to an EDNF concept-learning task with as positives P and as negatives N if and only if

$$(\forall \neg l_{1,1} \vee ... \vee \neg l_{1,n_1}) \wedge ... \wedge (\forall \neg l_{k,1} \vee ... \vee \neg l_{k,n_k})$$

is a solution to the UCNF concept-learning task with as positives N and as negatives P.

This property holds because e is covered by H if and only if e is not covered by $\neg H$, and the negation of a UCNF is an EDNF, and vice versa.

The main implication of this property is that a EDNF learner can be used to learn UCNF, and vice versa. One only has to switch to a dual problem formulation where the positives and the negatives are inverted, and the result is negated.

This has implications e.g. for the study of (Mooney 1995) who developed two algorithms, one for learning CNF and one for DNF, as well as for appraoches to structural matching and inductive logic programming, which can be used to address the opposite task. E.g. the ICL system of (De Raedt and Van Laer 1995) can easily be used to address structural matching (and EDNF).

Data Mining and Concept-Learning

Data mining and knowledge discovery (Mannila 1995; Fayyad *et al.* 1995) have recently enjoyed a lot of attention. It has its roots in machine learning, data bases and statistics. Nevertheless, the relation to classical concept-learning techniques is often unclear, as the aim in data mining is to discover regularities (often rules or clauses) that are valid in the data, rather than develop hypotheses with the aim of classification.

Using the logical tools developed above, it is possible to analyse and relate both problems from a logical perspective. In this analysis, we will focuss on the UCNF

or clausal representation mostly employed in data mining.

We start by looking at the specification of the problem as recently formulated by Heikki Mannila (Mannila 1995). He views data mining as the process of a constructing a theory $Th(L_S, r, q)$, where L_S is a set of sentences to consider, r the data(base), and q the quality criterion. The aim then is to find all sentences ϕ in the language L_S that satisfy the quality criterion w.r.t. the data r, i.e.

$$Th(L_S, r, q) = \{\phi \in L_S \mid q(r, \phi) \textit{ is true.}\}$$

In the practice of data mining, Th typically consists of a set of rules or clauses, i.e. Th is in a kind of $UCNF$ format, and each ϕ in L_S thus corresponds to a single clause. Furthermore, the aim is to find all possible $\phi \in L_S$ that are valid. Also, validity roughly corresponds to certain relaxations of ϕ being true in r.

Under these conditions, the problem of data mining can roughly be formulated as that of characteristic learning of UCNF expressions from interpretations. Indeed, let r contain a set of unclassified interpretations (positive examples only), let L_S contain all clauses allowed in UCNF expressions (when characteristic learning), and let $q(r, \phi) = \textit{covers}(\phi, r) = r$ is a model for ϕ. The conjunction of the clauses in $Th(L_S, r, q)$ then denotes the solution to the characteristic concept-learning task, i.e. the maximally specific hypothesis covering all positives. From this it follows that, the main difference between data mining and characteristic learning of interpretations in UCNF form lies in the relaxation of the quality-criterion, which typically requires rules to be approximately true (according to user-specified criteria) instead of completely true on the data.

This view of data mining has been exploited in the Clausal Discovery Engine of (De Raedt and Bruynooghe 1993; De Raedt and Dehaspe 1995), which addresses characteristic learning of interpretations using UCNF representations. It is probably the first data mining system working with full clausal representations.

Related Work and Conclusions

Various researchers have addressed similar issues as I have. For instance, (Michalski 1983; 1994) has studied induction, deduction, generalization and specialisation. My contribution here is to relate this discussion to the two standard notions of coverage employed in machine learning, and to observe that generalization and entailment always coincide (in one direction or another). Following (Niblett 1988), various logical notions of generalization have been analyzed and their use has been outlined. The relation between UCNF and EDNF is based on work by (Haussler 1988), though it was upgraded here to first order logic. Also, learning from entailment and learning from interpretations has been studied in computational learning theory, where their relation has been (partly) studied for propositional logic (Angluin *et al.* 1992; Frazier and Pitt 1993). More novel aspects of the current work include: the relation between data mining and characteristic concept-learning, the transformation of operators from UCNF to EDNF, the relation between inductive logic programming and structural matching, and the integration of these aspects. There are also several remaining questions such as: what is the relation between learning from entailment and learning from interpretations, and what is the relation between UCNF and EDNF (when learning from entailment) ?

Acknowledgements

Luc De Raedt is supported by the Belgian National Fund for Scientific Research, and by the ESPRIT IV project no. 20237 on Inductive Logic Programming II (ILP^2). He is grateful to Maurice Bruynooghe, Peter Flach, Wim Van Laer, Hendrik Blockeel, Luc Dehaspe, and especially to Nada Lavrac for many inspiring discussions.

References

D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of horn clauses. *Machine Learning*, 9:147–162, 1992.

L. De Raedt and M. Bruynooghe. A unifying framework for concept-learning algorithms. *The Knowledge Engineering Review*, 7(3):251–269, 1992.

L. De Raedt and M. Bruynooghe. A theory of clausal discovery. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1058–1063. Morgan Kaufmann, 1993.

L. De Raedt and L. Dehaspe. Clausal discovery. Technical Report KUL-CW, Department of Computer Science, Katholieke Universiteit Leuven, 1995. submitted.

L. De Raedt and S. Džeroski. First order jk-clausal theories are PAC-learnable. *Artificial Intelligence*, 70:375–392, 1994.

L. De Raedt and N. Lavrač. The many faces of inductive logic programming. In J. Komorowski, editor, *Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1993. invited paper.

L. De Raedt and W. Van Laer. Inductive constraint logic. In *Proceedings of the 5th Workshop on Algorithmic Learning Theory*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1995.

L. De Raedt, P. Idestam-Almquist, and G. Sablon. θ-subsumption for structural matching. Technical Report KUL-CW, Department of Computer Science, Katholieke Universiteit Leuven, 1996. draft.

L. De Raedt, editor. *Advances in Inductive Logic Programming*, volume 32 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 1996.

U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, editors. *Advances in Knowledge Discovery and Data Mining*. The MIT Press, 1995.

M. Frazier and L. Pitt. Learning from entailment. In *Proceedings of the 9th International Conference on Machine Learning*, 1993.

J.G. Ganascia and Y. Kodratoff. Improving the generalization step in learning. In R.S Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: an artificial intelligence approach*, volume 2, pages 215–241. Morgan Kaufmann, 1986.

D. Haussler. Quantifying inductive bias : AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36:177 – 221, 1988.

F. Hayes-Roth and J. McDermott. An interference matching technique for inducing abstractions. *Communications of the ACM*, 21:401–410, 1978.

N. Helft. Induction as nonmonotonic inference. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 149–156. Morgan Kaufmann, 1989.

P. Idestam-Almquist. Generalisation under implication using or-introduction. In *Proceedings of the 6th European Conference on Machine Learning*, volume 667, pages 56–64. Lecture Notes in Artificial Intelligence, 1993.

Y. Kodratoff. *Introduction to Machine Learning*. Pitman, 1988.

S. Lapointe and S. Matwin. Sub-unification: a tool for efficient induction of recursive programs. In *Proceedings of the 9th International Workshop on Machine Learning*. Morgan Kaufmann, 1992.

H. Mannila. Aspects of data mining. In Y. Kodratoff, G. Nakhaeizadeh, and G. Taylor, editors, *Proceedings of the MLnet Familiarization Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases*, pages 1–6, Heraklion, Crete, Greece, 1995.

R.S. Michalski. A theory and methodology of inductive learning. In R.S Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning: an artificial intelligence approach*, volume 1. Morgan Kaufmann, 1983.

R.S. Michalski. Inferential theory of learning: developping foundations for multistrategy learning. In R.S. Michalski and G. Tecuci, editors, *Machine Learning: A Multistrategy Approach*, volume 4, pages 3–61. Morgan Kaufmann, 1994.

T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.

R.J. Mooney. Encouraging experimental results on learning cnf. *Machine Learning*, 19:79–92, 1995.

S. Muggleton and W. Buntine. Machine invention of first order predicates by inverting resolution. In *Proceedings of the 5th International Workshop on Machine Learning*, pages 339–351. Morgan Kaufmann, 1988.

S. Muggleton and L. De Raedt. Inductive logic programming : Theory and methods. *Journal of Logic Programming*, 19,20:629–679, 1994.

S. Muggleton, editor. *Inductive Logic Programming*. Academic Press, 1992.

S. Muggleton. Inverting implication. *Artificial Intelligence*, 1994. To appear.

S. Muggleton. Inverse entailment and progol. *New Generation Computing*, 13, 1995.

B.K Natarajan. *Machine Learning : A Theoretical Approach*. Morgan Kaufmann, 1991.

T. Niblett. A study of generalisation in logic programs. In D. Sleeman, editor, *Proceedings of the 3rd European Working Session on Learning*, pages 131–138. Pitman, 1988.

G. Plotkin. A note on inductive generalization. In *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.

J.R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

L. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

S.A. Vere. Induction of concepts in the predicate calculus. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, pages 282–287. Morgan Kaufmann, 1975.

C. Vrain. Ogust: A system that learns using domain properties expressed as theorems. In Y. Kodratoff and R.S. Michalski, editors, *Machine Learning: an artificial intelligence approach*, volume 3, pages 360–381. Morgan Kaufmann, 1990.

P.H. Winston. Learning structural descriptions from examples. In P.H. Winston, editor, *Psychology of Computer Vision*. The MIT Press, 1975.

Induction as Knowledge Integration

Benjamin D. Smith
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive M/S 525-3660
Pasadena, CA 91109-8099
smith@aig.jpl.nasa.gov

Paul S. Rosenbloom
Information Sciences Institute & Computer Science Dept.
University of Southern California
4676 Admiralty Way
Marina del Rey, CA 90292
rosenbloom@isi.edu

Abstract

Two key issues for induction algorithms are the accuracy of the learned hypothesis and the computational resources consumed in inducing that hypothesis. One of the most promising ways to improve performance along both dimensions is to make use of additional knowledge. Multi-strategy learning algorithms tackle this problem by employing several strategies for handling different kinds of knowledge in different ways. However, integrating knowledge into an induction algorithm can be difficult when the new knowledge differs significantly from the knowledge the algorithm already uses. In many cases the algorithm must be rewritten.

This paper presents KII, a Knowledge Integration framework for Induction, that provides a uniform mechanism for integrating knowledge into induction. In theory, arbitrary knowledge can be integrated with this mechanism, but in practice the knowledge representation language determines both the knowledge that can be integrated, and the costs of integration and induction. By instantiating KII with various set representations, algorithms can be generated at different trade-off points along these dimensions.

One instantiation of KII, called RS-KII, is presented that can implement hybrid induction algorithms, depending on which knowledge it utilizes. RS-KII is demonstrated to implement AQ-11 (Michalski 1978), as well as a hybrid algorithm that utilizes a domain theory and noisy examples. Other algorithms are also possible.

Introduction

Two key criteria for evaluating induction algorithms are the accuracy of the induced hypothesis and the computational cost of inducing that hypothesis. One of the most powerful ways to achieve improvements along both of these dimensions is by integrating additional knowledge into the induction process. Knowledge consists of examples, domain theories, heuristics, and any other information that affects which hypothesis is induced—that is, knowledge is examples plus biases.

A given single-strategy learning algorithm can utilize some knowledge very effectively, others less effectively, and some knowledge not at all. By using multiple strategies, an induction algorithm can make more effective use of a wider range of knowledge, thereby improving performance. However, even a multi-strategy learning algorithm can only make use of knowledge for which its strategies are designed.

In order to utilize new kinds of knowledge, the knowledge must either be recast as a kind for which the algorithm already has a strategy—for example, integrating type constraints into FOIL by casting them as pseudo negative examples (Quinlan 1990)—or the algorithm must be rewritten to take advantage of the new knowledge by adding a new strategy or modifying an existing one. The first approach—recasting knowledge—is limited by the expressiveness of the knowledge already used by the algorithm. If the new knowledge cannot be expressed in terms of the existing kinds of knowledge, then the new knowledge cannot be utilized. The second approach—rewriting an algorithm to utilize a new kind of knowledge—is difficult. It also fails to solve the underlying problem—if yet another kind of knowledge is made available, the algorithm may have to be modified once again.

What is needed is an easier way to integrate knowledge into induction. One approach for doing this exploits the observation that a knowledge fragment plus a strategy for using that knowledge constitutes a bias, since together they determine which hypothesis is induced. These biases can be expressed uniformly in terms of constraints and preferences on the hypothesis space. The induced hypothesis is the most preferred hypothesis among those that satisfy the constraints. New knowledge and strategies are integrated into induction by combining their constraints and preferences with those previously integrated.

This approach is formalized in a framework called KII. This framework represents constraints and preferences as sets, and provides set-based operations for integrating knowledge expressed in this way, and for inducing hypotheses from the integrated knowledge. Converting knowledge into constraints and preferences is handled by *translators* (Cohen 1992), which are written by the user for each knowledge fragment, or class of related knowledge fragments.

Since KII is defined in terms of sets and set operations, some set representation must be specified in order for KII to be operational. The set representation determines the kinds of knowledge that can be expressed, and also determines the computational complexity of integration and induction. Each set representation yields an instantiation of KII at a different trade-off point between expressiveness and computational complexity.

This approach is most similar to that of Russell and Grosof (Russell & Grosof 1987), in which biases are represented as determinations, and the hypothesis is deduced from the determinations and examples by a theorem prover. As in KII, the inductive leaps come from biases, which may be grounded in supposition instead of fact. A major difference between this system and KII is KII's ability to select different set representations, which allows different trade-offs to be made between expressiveness and cost. Determinations, by contrast, are at a fixed trade-off point, although one could imagine using restricted logics.

One advantage of KII's formal relationship between the set representation and the cost/expressiveness trade-off is that it allows formal analysis of these trade-offs. In particular, an upper limit can be established on the expressiveness of the set representations for which induction is even computable. This sets a practical limit on the kinds of knowledge that can be utilized by induction.

Among the set representations below this limit, there are a number that generate useful instantiations of KII. Most notably, Incremental Version Space Merging (Hirsh 1990) can be generated by using a boundary set representation for constraints (i.e., version spaces), and an empty representation for preferences; and an algorithm similar to Grendel (Cohen 1992) can be instantiated from KII by representing sets as antecedent description grammars (essentially context free grammars). These will be discussed briefly. A new algorithm, RS-KII, is instantiated from KII by representing sets as regular grammars. This algorithm seems to strike a good balance between expressiveness and complexity.

RS-KII can use a wide range of knowledge, and combine this knowledge in a number of ways. This makes it a good multi-strategy algorithm. RS-KII can use the knowledge and strategies of at least two existing algorithms, the Candidate Elimination Algorithm (Mitchell 1982) and AQ-11 with a beam width of one (Michalski 1978). It can also utilize additional knowledge, such as a domain theory and noisy examples. Although space limits us from discussing all of these in detail, the translators needed to implement AQ-11 are demonstrated, as well as those for the domain theory and noisy examples. When utilizing only the AQ-11 knowledge, RS-KII induces the same hypotheses as AQ-11 with a beam width of one, with a computational complexity that is only a little worse. When RS-KII utilizes the translators for the additional knowledge, RS-KII induces a more accurate hypothesis than AQ-11, and in much less time. RS-KII looks able to express and integrate other common knowledge sources and strategies as well, though this is an area for future research.

The Knowledge Integration Framework

This section formally describes KII, a Knowledge Integration Framework for Induction. The combination of a knowledge fragment and a strategy for using that knowledge can be considered a bias, which is expressed in terms of constraints and preferences over the hypothesis space. For instance, a positive example and a strategy that assumes the target concept is strictly consistent with the examples, would be translated as a constraint that is satisfied only by hypotheses that cover the example. A strategy that assumed noisy examples might be expressed as a preference for hypotheses that were most consistent with the example, but does not reject inconsistent hypotheses outright.

The biases are *integrated* into a single composite bias by combining their respective constraints and preferences. The composite bias, which includes the examples, wholly determines the selection of the induced hypothesis. If there are several hypotheses which the bias finds equally acceptable, any one may be selected arbitrarily as the target concept. This set is called the *solution set*. In this view, integration precedes induction, rather than being part of it. This separation makes it easier to integrate knowledge into induction, since the effects of each process are clearer.

KII formalizes these ideas as follows. Each bias is expressed as a triple of three sets, $\langle H, C, P \rangle$, where H is the hypothesis space, C is the set of hypotheses that satisfy the constraints of all the biases, and P is a set of hypothesis pairs, $\langle x, y \rangle$, such that x is less preferred than y by at least one of the biases. The solution set, from which the induced hypothesis is selected arbitrarily, is the set of most preferred hypothesis among those that satisfy the constraints—namely, the hypotheses in C for which no other hypothesis in C is preferable, according to P. Formally, $\{x \in C \mid \forall y \in C \ \langle x, y \rangle \notin P\}$.

KII provides several operations on knowledge expressed in this representation: *translation*, *integration*, *induction* (selecting a hypothesis from the solution set), and solution set *queries*. These operations, as well as the solution set itself, are defined in terms of set operations on H, C, and P. These operators are described in detail below.

Translation Knowledge is converted from the form in which it occurs (its *naturalistic representation* (Rosenbloom *et al.* 1993)) into $\langle H, C, P \rangle$ triples by *translators* (Cohen 1992). Since knowledge is translated into constraints and preferences over the hypothesis space, the implementation of each translator depends on both the hypothesis space and the knowl-

edge. In the worst case, a different implementation is required for each pair of knowledge fragment and hypothesis space. Since there are a potentially infinite number of translators, they are not provided as part of the KII formalism, but must be provided by the user as needed.

Fortunately, closely related pairs of hypothesis space and knowledge often have similar translations, allowing a single translator to be written for all of the pairs. One such translator, which will be described in detail later, takes as input an example and a hypothesis space. The example can be any member of the instance space, and the hypothesis space is selected from a family of languages by specifying the set of features. The same translator works for every pair of example and hypothesis language in this space.

Integration Translated knowledge fragments are *integrated* by composing their $\langle H, C, P \rangle$ triples. A hypothesis can only be the induced hypothesis if it is accepted by the constraints of all of the knowledge fragments, and if the combined preferences of the knowledge fragments do not prefer some other hypothesis. That is, the induced hypothesis must satisfy the conjunction of the constraints, and be preferred by the disjunction of the preferences. This reasoning is captured in the following definition for the integration of two tuples, $\langle H, C_1, P_1 \rangle$ and $\langle H, C_2, P_2 \rangle$. The hypothesis space is the same in both cases, since it is not clear what it means to integrate knowledge about target hypotheses from different hypothesis spaces.

$$\text{Integrate}(\langle H, C_1, P_1 \rangle, \langle H, C_2, P_2 \rangle) = \langle H, C_1 \cap C_2, P_1 \cup P_2 \rangle \quad (1)$$

The integration operator assumes that the knowledge is consistent. That is, C_1 and C_2 are not mutually exclusive, and that $P_1 \cup P_2$ does not contain cycles (e.g., $a < b$ and $b < a$). Although such knowledge can be integrated, the inconsistencies will not be dealt with in any significant fashion. Mutually exclusive constraints will result in an empty solution set, and cycles are broken arbitrarily by assuming every element of the cycle is dominated. Developing more sophisticated strategies for dealing with contradictions is an area for future research.

Although KII does not deal with contradictory knowledge, it can deal with uncertain knowledge. For example, noisy examples and incomplete domain theories can both be utilized in KII. Translators for these knowledge sources are described later.

Induction and Solution Set Queries The integrated knowledge is represented by a single tuple, $\langle H, C, P \rangle$. The target concept is induced from the integrated knowledge by selecting an arbitrary hypothesis from the solution set of $\langle H, C, P \rangle$. KII also supports queries about the solution set, such as whether it is empty, a singleton, contains a given hypothesis, or is a subset of some other set. These correspond to the operations that have proven empirically useful for version spaces (Hirsh 1992), which can be thought of as solution sets for knowledge expressed as constraints.

It is conjectured that these four queries plus the ability to select a hypothesis from the solution set are sufficient for the vast majority of induction tasks. Most existing induction algorithms involve only the enumeration operator and perhaps an *Empty* or *Unique* query. The Candidate Elimination algorithm (Mitchell 1982) and Incremental Version Space Merging (IVSM) (Hirsh 1990) use all four queries, but do not select a hypothesis from the solution set (they return the entire set).

The queries and selection of a hypothesis from the solution set can be implemented in terms of a single *enumeration* operator. The enumeration operator returns n elements of a set, S, where n is specified by the user. It is defined formally as follows.

$$\begin{aligned} &\mathit{Enumerate}(S, n) \rightarrow \{h_1, h_2, \ldots h_m\} \\ &\quad \text{where} \\ &\quad m = \min(n, |S|), \{h_1, h_2, \ldots h_m\} \subseteq S \end{aligned}$$

Normally, S is the solution set of $\langle H, C, P \rangle$. It can sometimes be cheaper to compute the first few elements of the solution set from $\langle H, C, P \rangle$ than to compute even the intensional representation of the solution set from $\langle H, C, P \rangle$. Therefore, the S argument to the enumeration operator can be either a $\langle H, C, P \rangle$ tuple, or a set expression involving an $\langle H, C, P \rangle$ tuple and other sets. This allows the enumeration operator to use whatever optimizations seem appropriate. A different implementation of the enumerate operator is needed for different set representations of S, H, C, and P.

A hypothesis is induced by selecting a single hypothesis from the solution set. This is done with a call to $\mathit{Enumerate}(\langle H, C, P \rangle, 1)$. The emptiness and uniqueness queries are implemented as shown below, where S is the solution set of tuple $\langle H, C, P \rangle$, A is set of hypotheses in H, and h is a hypothesis in H.

- $\mathit{Empty}(S) \Leftrightarrow \mathit{Enumerate}(\langle H, C, P \rangle, 1) = \emptyset$
- $\mathit{Unique}(S) \Leftrightarrow |\mathit{Enumerate}(\langle H, C, P \rangle, 2)| = 1$
- $\mathit{Member}(h, S) \Leftrightarrow \mathit{Enumerate}(\langle H, C, P \rangle \cap \{h\}, 1) \neq \emptyset$
- $\mathit{Subset}(S, A) \Leftrightarrow \mathit{Enumerate}(\langle H, C, P \rangle \cap \overline{A}, 1) = \emptyset$

An Example Induction Task

An example of how KII can solve a simple induction task is given below. Sets have been represented extensionally in this example. Although this is not the only possible set representation, and is generally a poor one, it is the simplest one for illustrative purposes.

The Hypothesis Space The target concept is a member of a hypothesis space in which hypotheses are described by conjunctive feature vectors. There are three features `size`, `color`, and `shape`. The values for these features are `size` $\in$ {`small`, `large`, `any-size`}, `color` $\in$ {`black`, `white`, `any-color`}, and

- *TranPosExample*$(H, \langle z, c, s\rangle) \rightarrow \langle C, \{\}\rangle$ where
  $C = \{x \in H \mid x \text{ covers } \langle z, c, s\rangle\}$
  $= \{z, \texttt{any-size}\} \times \{c, \texttt{any-color}\} \times \{s, \texttt{any-shape}\}$
- *TranNegExample*$(H, \langle z, c, s\rangle) \rightarrow \langle C, \{\}\rangle$ where
  $C = \{x \in H \mid x \text{ does not cover } \langle z, c, s\rangle\}$
  $=$ complement of
  $\{z, \texttt{any-size}\} \times \{c, \texttt{any-color}\} \times \{s, \texttt{any-shape}\}$
- *TranPreferGeneral*$(H) \rightarrow \langle H, P\rangle$ where
  $P = \{\langle x, y\rangle \in H \times H \mid x \text{ is more specific than } y\}$
  $= \{\langle sbr, ?br\rangle, \langle sbr, s?r\rangle, \langle sbr, ??r\rangle, \langle swr, ?wr\rangle, \ldots\}$

Figure 1: Translators.

shape $\in$ {`circle`, `rectangle`, `any-shape`}. Hypotheses are described as 3-tuples from `size`×`color`×`shape`. For shorthand identification, a value is specified by the first character of its name, except for the any values which are represented by a "?". So the hypothesis ⟨`any-size`, `white`, `circle`⟩ would be written as *?wc*.

Instances are the "ground" hypotheses. An instance is a tuple $\langle size, color, shape\rangle$ where $color \in$ {`black`, `white`}, $size \in$ {`small`, `large`}, and $shape \in$ {`circle`, `rectangle`}.

Available Knowledge The available knowledge consists of three examples (classified instances), and an assumption that accuracy increases with generality. There are three examples, two positive and one negative. The two positive examples are $e_1 = swc$ and $e_2 = sbc$. The negative example is $e_3 = lwr$. The target concept is *s??*. That is, `size` = `small`, and color and shape are irrelevant.

Translators The first step is to translate the knowledge into constraints and preferences. Three translators are constructed, one for each type of knowledge: the positive examples, negative examples, and the generality preference. These translators are shown in Figure 1. Since the hypothesis space is understood, $\langle H, C, P\rangle$ tuples will generally be referred to as just $\langle C, P\rangle$ tuples for the remainder of this illustration.

The examples are translated in this scenario under the assumption that they are correct; that is, the target concept covers all of the positive examples and none of the negatives. Positive examples are translated as constraints satisfied only by hypotheses that cover the example. Negative examples are translated similarly, except that hypotheses must *not* cover the example. The bias for general hypotheses is translated into a $\langle C, P\rangle$ pair where C is H (it rejects nothing), and $P = \{\langle x, y\rangle \in H \times H \mid x \text{ is more specific than } y\}$. Hypothesis x is more specific than hypothesis y if x is equivalent to y, except that some of the values in y have been replaced by "any" values. For example, *swr* is more specific than *?wr*, but there is no ordering between *?wc* and *swr*.

Integration and Induction Examples e_1 and e_2 are translated by *TranPosExample* into $\langle H, C_1, \emptyset\rangle$ and $\langle H, C_2, \emptyset\rangle$, respectively. Example e_3 is translated by *TranNegExample* into $\langle H, C_3, \emptyset\rangle$. The preference for general hypotheses is translated into $\langle H, H, P_4\rangle$. These tuples are integrated into a single tuple, $\langle H, C, P\rangle = \langle H, C_1 \cap C_2 \cap C_3 \cap H, \emptyset \cup \emptyset \cup \emptyset \cup P_4\rangle$. This tuple represents the combined biases of the four knowledge fragments.

A hypothesis is induced by selecting one arbitrarily from the solution set of $\langle H, C, P\rangle$. This is accomplished by calling *Enumerate*$(\langle H, C, P\rangle, 1)$. The solution set consists of the undominated elements of C with respect to the dominance relation P. C contains three elements, *s??*, *??c* and *s?c*. P prefers both *s??* and *??c* to *s?c*, but there is no preference ordering between *s??* and *??c*. The undominated elements of C are therefore *s??* and *??c*. One of these is selected arbitrarily as the induced hypothesis.

Instantiating KII

In order to implement KII, specific set representations for H, C, and P are necessary. These representations can be as simple as an extensional set, or as powerful as arbitrary Turing machines. However, some representation is needed. The representation determines which knowledge can be expressed in terms of $\langle H, C, P\rangle$ tuples and integrated. It also determines the computational complexity of the integration and enumeration operations, which are defined in terms of set operations. By instantiating KII with different set representations, algorithms can be generated at different tradeoff points between cost and expressiveness.

The space of possible set representations maps onto the space of grammars. Every computable set is the language of some grammar. Similarly, every computable set representation is equivalent to some class of grammars. These classes include, but are not limited to, the classes of the Chomsky hierarchy (Chomsky 1959)—regular, context free, context sensitive, and recursively enumerable (r.e.). The complexity of set operations generally increases with the expressiveness of the language class.

Allowing H, C, and P to be recursively enumerable (i.e., arbitrary Turing machines), would certainly provide the most expressiveness. Although $\langle H, C, P\rangle$ tuples with r.e. sets can be expressed and integrated, the solution sets of some such tuples are uncomputable, and there is no way to know which tuples have this property. This will be discussed in more detail below. Since it is impossible to enumerate even a single element of an uncomputable set, it is impossible to induce a hypothesis by selecting one from the solution set. There is clearly a practical upper limit on the expressiveness of the set representations.

It is possible to establish the most expressive languages for C and P that guarantee a computable solution set. This establishes a practical limit on the knowledge that can be integrated into induction.

By definition, the solution set is computable if and only if it is recursively enumerable. The solution set can always be constructed by applying a formula of set operations to C and P, as will be shown below. The most restrictive language in which the solution set can be expressed can be derived from this formula and the set representations for C and P by using the the closure properties of these set operations. Inverting this function yields the most expressive C and P representations for which the solution set is guaranteed to be at most recursively enumerable.

The solution set can be computed from C and P according to the equation $\overline{first((C\times C)\cap P)}\cap C$. The derivation is shown in Equation 2, below. In this definition, the function $first(\{\langle x_1, y_1\rangle, \langle x_2, y_2\rangle, \ldots\})$ is a projection returning the set of tuple first-elements, namely $\{x_1, x_2, \ldots\}$.

$$\begin{aligned} SolnSet(\langle H, C, P\rangle) &= \{x \in C \mid \forall_{y\in C}\ \langle x, y\rangle \notin P\} \\ &= \overline{\{x \in H \mid (x \in C \text{ and } \exists_{y\in C}\langle x, y\rangle \in P) \text{ or } x \notin C\}} \\ &= \overline{\{x \in H \mid x \in C \text{ and } \exists_{y\in C}\langle x, y\rangle \in P\} \cup \overline{C}} \\ &= \overline{first(\{\langle x, y\rangle \in C\times C \mid \langle x, y\rangle \in P\})}\cap C \\ &= \overline{first((C\times C)\cap P)}\cap C \end{aligned} \quad (2)$$

The least expressive representation in which the solution set can be represented can be computed from the closure of the above equation over the C and P set representations. To do this, it helps to know the closure properties for the individual set operations in the equation: intersection, complement, Cartesian product, and projection (first). The closure properties of intersection and complement are well known for most language classes, although it is an open problem whether the context sensitive languages are closed under complementation (Hopcroft & Ullman 1979). The closure properties of projection and Cartesian product are not known as such, but these operations map onto other operations for which closure properties are known.

The Cartesian product of two grammars, $A\times B$, can be represented by their concatenation, AB. The tuple $\langle x, y\rangle$ is represented by the string xy. The Cartesian product can also be represented by interleaving the strings in A and B so that $\langle x, y\rangle$ is represented by a string in which the symbols in x and y alternate. Interleaving can sometimes represent subsets of $A\times B$ that concatenation cannot, depending on the language in which the product is expressed. The closure properties of languages under Cartesian product depends on which approach is used. The following discussion derives limits on the languages for $C\times C$ and P. When the language for C is closed under Cartesian product, then the limits on $C\times C$ also apply to C, since both can be expressed in the same language. Otherwise, the limits on C have to be derived from those on $C\times C$ using the closure properties of the given implementation of Cartesian product. However, when C is not closed under Cartesian product, the language for C is necessarily less expressive than that for $C\times C$. The expressiveness limits on $C\times C$ therefore provide a good upper bound on the expressiveness of C that is independent of the Cartesian product implementation.

Regardless of the representation used for Cartesian product, projection can be implemented as a *homomorphism* (Hopcroft & Ullman 1979), which is a mapping from symbols in one alphabet to strings in another. Homomorphisms can be used to erase symbols from strings in a language, which is exactly what projection does—it erases symbols from the second field of a tuple, leaving only the symbols from the first field. A more detailed derivation of the properties for projection and Cartesian product can be found in (Smith 1995).

The closure properties of languages under projection, intersection, intersection with a regular grammar, and complement are summarized in Table 1. It should be clear that the solution set, $\overline{first((C\times C)\cap P)}\cap C$, is r.e. when $(C\times C)\cap P$ is at most context free, and uncomputable when it is any more expressive than that. For example, if $(C\times C)\cap P$ is context sensitive, then $first((C\times C)\cap P$ is r.e. The complement of a set that is r.e. but not recursive is uncomputable (Hopcroft & Ullman 1979), so the solution set, $\overline{first((C\times C)\cap P}$, is uncomputable. A complete proof appears in (Smith 1995).

There are several ways to select C, P, and the implementation of Cartesian product, such that $(C\times C)\cap P$ is at most context free. The expressiveness of both C and P can be maximized by choosing one of C and P to be at most regular, and the other to be at most context free. This is because CFLs are closed under intersection with regular sets, but not with other CFLs. Regular sets are closed under all implementations of Cartesian product (both concatenation and arbitrary interleaving), and context free sets are closed under concatenation but only some interleavings. So if C is regular, any implementation of Cartesian product can be used, but if C is context free, then the choices are more restricted.

As a practical matter, C should be closed under intersection and P under union in order to support the integration operator. This effectively restricts C to be regular and P to be at most context free. This also maximizes the choices of the Cartesian product implementation. However, it is possible for C to be context free and P to be regular if the C set of at most one of the $\langle H, C, P\rangle$ triples being integrated is context free and the rest are regular. This follows from the closure of context free languages under intersection with regular grammars.

Other ways of selecting C and P are summarized in Table 2. This table assumes that C is closed under Cartesian product. As one interesting case, if the representation for P can express only the empty set, then the solution set is just C, so C can be r.e. The

Operations	Language					
	Regular	DCFL	CFL	CSL	recursive	r.e.
$\cap$	$\surd$			$\surd$	$\surd$	$\surd$
$\cap R$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$	$\surd$
complement	$\surd$	$\surd$		?	$\surd$	
projection (homomorphisms)	$\surd$		$\surd$			$\surd$

Table 1: Closure Under Operations Needed to Compute the Solution Set.

restriction that $(C \times C) \cap P$ be at most context free is still satisfied, since $(C \times C) \cap P$ is always the empty set, and therefore well within the context free languages.

RS-KII

Instantiating KII with different set representations produces algorithms with different computational complexities and abilities to utilize knowledge. One instantiation that seems to strike a good balance between computational cost and expressiveness represents H, C, and P as regular sets. This instantiation is called RS-KII.

RS-KII is a good multi-strategy algorithm, in that it can utilize various knowledge and strategies, depending on what knowledge is integrated, and how it is translated. Existing algorithms can be emulated by creating translators for the knowledge and strategies of that algorithm, and integrating the resulting $\langle H, C, P \rangle$ tuples. Hybrid multi-strategy algorithms can be created by translating and integrating additional knowledge, or by integrating novel combinations of knowledge for which translators already exist.

Creating algorithms by writing translators for individual knowledge fragments and integrating them together can be easier than writing new induction algorithms. Algorithms can be constructed modularly from translators, which allows knowledge fragments to be easily added or removed. By contrast, modifications made to an algorithm in order to utilize one knowledge fragment may have to be discarded in order to utilize a second fragment.

The remainder of this section demonstrates how RS-KII can emulate AQ-11 with a beam width of one (Michalski 1978), and how RS-KII can integrate additional knowledge, namely an overgeneral domain theory and noisy examples, to create a hybrid algorithm. AQ-11 with higher order beam widths is not demonstrated, since it is not clear how to express the corresponding bias as a regular grammar. This bias may require a more powerful set representation.

When using only the AQ-11 knowledge, RS-KII induces the same hypotheses as AQ-11, albeit at a slightly worse computational complexity. When utilizing the additional knowledge, RS-KII induces a more accurate hypothesis than AQ-11, and does so more quickly.

RS-KII translators can be written for other knowledge as well, though space restrictions prevent any detailed discussion. Of note, RS-KII translators can be constructed for all biases expressible as version spaces (for certain classes of hypothesis spaces) (Smith 1995). It also looks likely that RS-KII translators can be constructed for the knowledge used by other induction algorithms, though this is an area for future research.

Translators for AQ-11 Biases

The biases used by AQ-11 are strict consistency with the examples, and an user-defined *lexicographic evaluation function* (LEF). The LEF totally orders the hypotheses according to user-defined criteria. The induced hypothesis is one that is consistent with all of the examples, and is a (possibly local) maximum of the LEF. A translator is demonstrated in which the LEF is an information gain metric, as used in algorithms such as ID3 (Quinlan 1986).

Hypotheses are sentences in the VL_1 language (Michalski 1974). There are k features, denoted f_1 through f_k, where feature f_i can take values from the set V_i. A hypothesis is a disjunction of terms, a term is a conjunction of selectors, and a selector is of the form $[f_i \; rel \; v_i]$, where v_i is in V_i and rel is a relation in $\{<, \leq, =, \neq, \geq, >\}$. A specific hypothesis space in VL_1 is specified by the list of features and their values, and is denoted $VL_1(\langle f_1, V_1 \rangle, \ldots, \langle f_k, V_k \rangle)$.

An instance is a vector of k values, $\langle x_1, x_2, \ldots, x_k \rangle$, where x_i is a value in V_i. A selector $[f_i \; rel \; v_i]$ is satisfied by an example if and only if $x_i \; rel \; v_i$. A hypothesis *covers* an example if the example satisfies the hypothesis.

Strict Consistency with Examples A bias for strict consistency with a positive example can be expressed as a constraint that the induced hypothesis must cover the example. Similarly, strict consistency with a negative example constrains the induced hypothesis not to cover the example. Each of these constraints is expressed as a regular grammar that only recognizes hypotheses that satisfy the constraint. The regular expression for the set of VL_1 hypotheses covering an example, $Covers(H, e)$ is shown in Figure 2. The sets of values in COVERING-SELECTOR are all regular sets. For example, the set of integers less than

C	P	$(C \times C) \cap P$	$\overline{first((C \times C) \cap P)} \cap C$
$\leq$ regular	$\leq$ regular	$\leq$ regular	$\leq$ regular
$\leq$ regular	$\leq$ CFL	$\leq$ CFL	$\leq$ recursive
$\leq$ CFL	$\leq$ regular	$\leq$ CFL	$\leq$ recursive
$\geq$ CFL	$\geq$ CFL	$\geq$ CSL	uncomputable
		$>$ CFL	uncomputable

Table 2: Summary of Expressiveness Bounds.

$$TranPosAQExample(VL_1(\langle f_1, V_1\rangle, \ldots, \langle f_k, V_k\rangle), \langle x_1, x_2, \ldots, x_k\rangle) \rightarrow \langle H, C, \{\}\rangle$$

where

$H = VL_1(\langle f_1, V_1\rangle, \ldots, \langle f_k, V_k\rangle)$

$C = Covers(VL_1(\langle f_1, V_1\rangle, \ldots, \langle f_k, V_k\rangle), \langle x_1, \ldots, x_k\rangle)$

$$TranNegAQExample(VL_1(\langle f_1, V_1\rangle, \ldots, \langle f_k, V_k\rangle), \langle x_1, x_2, \ldots, x_k\rangle) \rightarrow \langle H, C, \{\}\rangle$$

where

$H = VL_1(\langle f_1, V_1\rangle, \ldots, \langle f_k, V_k\rangle)$

$C = Excludes(VL_1(\langle f_1, V_1\rangle, \ldots, \langle f_k, V_k\rangle), \langle x_1, \ldots, x_k\rangle)$

Figure 3: Example Translators for VL_1.

100 is $(0-9)|((1-9)(0-9))$. There is an algorithm that generates each of these sets given the relation and the bounding number, but it is omitted for brevity. The complement of $Covers(H, e)$ is $Excludes(H, e)$, the set of hypotheses in H that do not cover example e. These two regular grammars implement the translators for positive and examples in the VL_1 hypothesis space language, as shown in Figure 3. The translator takes as input the list of features and their values, and the example.

The LEF AQ-11 performs a beam search of the hypothesis space to find a hypothesis that maximizes the LEF, or is at least a good local approximation. AQ-11 returns the first hypothesis visited by this search that is also consistent with the examples. This is a bias towards hypotheses that come earlier in the search order.

This bias can be expressed as an $\langle H, C, P\rangle$ tuple in which $C = H$ (i.e., no hypotheses are rejected), and P is a partial ordering over the hypothesis space in which $\langle a, b\rangle$ is in P if and only if hypothesis a comes after hypothesis b in the search order (i.e., a is less preferred than b).

The search order of a beam search is difficult, and perhaps impossible, to express as a regular grammar. However, with a beam width of one, beam search becomes hill climbing, which can be expressed as a regular grammar.

In hill climbing, single selector extensions of the current best hypothesis are evaluated by some evaluation function, f, and the extension with the best evaluation becomes the next current best hypothesis. Given two terms, $t_1 = a_1 a_2 \ldots a_n$ and $t_2 = b_1 b_2 \ldots b_m$, where a_i and b_i are selectors, t_1 is visited before t_2 if the first $k-1$ extensions of t_1 and t_2 are the same, but on the k^{th} extension, either t_1 has a better evaluation than t_2, or t_1 has no more selectors. Formally, there is either some extension $k \leq \min(m, n)$ such that for all $i < k$, $a_i = b_i$ and $f(a_1 \ldots a_k) > f(b_1 \ldots b_k)$, or $m < n$ and the first m selectors of t_1 and t_2 are the same.

This is equivalent to saying that the digit string $f(a_1) \cdot f(a_1 a_2) \cdot \ldots \cdot f(a_1 a_2 \ldots a_n)$ comes before the digit string $f(b_1) \cdot f(b_1 b_2) \cdot \ldots \cdot f(b_1 b_2 \ldots b_n)$ in dictionary (lexicographic) order. This assumes that low evaluations are best, and that the evaluation function returns a unique value for each term—that is, $f(a_1 \ldots a_m) = f(b_1 \ldots b_m)$ if and only if $a_i = b_i$ for all i between one and m. This can be ensured by assigning a unique id to each selector, and appending the id for the last selector in the term to the end of the term's evaluation. The evaluations of two terms are compared after each extension until one partial term either has a better evaluation, or terminates.

A regular grammar can be constructed that recognizes pairs of hypotheses, $\langle h_1, h_2\rangle$, if h_1 is visited before h_2 in the search. This is done in two steps. First, a grammar is constructed that maps each hypothesis onto digit strings of the kind described above. The digit strings are then passed to a regular grammar that recognizes pairs of digit strings, $\langle d_1, d_2\rangle$, such that d_1 comes before d_2 in dictionary order. This is equivalent to substituting the mapping grammar into the dictionary ordering grammar. Since regular grammar are closed under substitution, the resulting grammar is also regular (Hopcroft & Ullman 1979).

The digit string comparison grammar is the simpler of the two, so it will be described first. This grammar recognizes pairs of digit strings, $\langle x, y\rangle$, such that x comes before y lexicographically. A special termination symbol, #, is appended to each string, and the resulting strings are interleaved so that their symbols alternate. The interleaved string is given as input to the grammar specified by the regular expression EQUAL* LESS-THAN ANY*, where EQUAL = (00|11|##), LESS-THAN = (01|#0|#1) and ANY = (0|1|#). This expression assumes a binary digit string, but can be easily

$$
\begin{array}{l}
\mathit{Covers}(VL_1(\langle f_1, V_1\rangle, \langle f_2, V_2\rangle, \ldots, \langle f_k, V_k\rangle), \langle x_1, x_2, \ldots, x_k\rangle) \rightarrow G \text{ where} \\
\quad \text{G = (ANY-TERM or)}^* \text{ COVERING-TERM (or ANY-TERM)}^* \\
\quad \text{ANY-TERM = SELECTOR}^+ \\
\quad \text{COVERING-TERM = COVERING-SELECTOR}^+ \\
\quad \text{SELECTOR = } \text{“[”} \ f_1 \ (< \mid \leq \mid = \mid \neq \mid \geq \mid >) \ V_1 \ \text{“]”} \ \mid \\
\qquad\qquad\qquad\quad \text{“[”} \ f_2 \ (< \mid \leq \mid = \mid \neq \mid \geq \mid >) \ V_2 \ \text{“]”} \ \mid \\
\qquad\qquad\qquad\qquad\qquad \vdots \\
\qquad\qquad\qquad\quad \text{“[”} \ f_k \ (< \mid \leq \mid = \mid \neq \mid \geq \mid >) \ V_k \ \text{“]”} \\
\quad \text{COVERING-SELECTOR} = \{[f_i \ \# \ v] \mid x_i \ \# \ v \text{ and } \# \in \{<, \leq, =, \neq, \geq, >\}\} \\
\quad = \begin{array}{llll}
\text{“[”} \ f_1 < \{v \in V_1 \mid v \geq x_1\} \ \text{“]”} \mid & \ldots \mid & \text{“[”} \ f_k < \{v \in V_k \mid v \geq x_k\} \ \text{“]”} \mid \\
\text{“[”} \ f_1 \leq \{v \in V_1 \mid v > x_1\} \ \text{“]”} \mid & \ldots \mid & \text{“[”} \ f_k \leq \{v \in V_k \mid v > x_k\} \ \text{“]”} \mid \\
\text{“[”} \ f_1 = x_1 \ \text{“]”} \mid & \ldots \mid & \text{“[”} \ f_k = x_k \ \text{“]”} \mid \\
\text{“[”} \ f_1 \neq (V_1 - \{x_1\}) \ \text{“]”} \mid & \ldots \mid & \text{“[”} \ f_k \neq (V_k - \{x_k\}) \ \text{“]”} \mid \\
\text{“[”} \ f_1 \geq \{v \in V_1 \mid v < x_1\} \ \text{“]”} \mid & \ldots \mid & \text{“[”} \ f_k \geq \{v \in V_k \mid v < x_k\} \ \text{“]”} \mid \\
\text{“[”} \ f_1 > \{v \in V_1 \mid v \leq x_1\} \ \text{“]”} \mid & \ldots \mid & \text{“[”} \ f_k > \{v \in V_k \mid v \leq x_k\} \ \text{“]”}
\end{array}
\end{array}
$$

Figure 2: Regular Expression for the Set of VL_1 Hypotheses Covering an Instance.

extended to handle base ten numbers.

The mapping of a hypothesis onto a digit string is accomplished by a Moore machine—a DFA that has an output string associated with each state. Recall that the digit string for a term, $a_1 a_2 \ldots a_m$, is $f(a_1) \cdot f(a_1 a_2) \cdot \ldots \cdot f(a_1 a_2 \ldots a_m)$). The machine takes a hypothesis as input. After reading each selector, it outputs the evaluation string for the current partial term. So after seeing a_1, it prints $f(a_1)$. After seeing a_2 it prints $f(a_1 a_2)$, and so on until it has printed the digit string for the term. When the end of the term is encountered (i.e., an *or* symbol is seen), the DFA returns to the initial state and repeats the process for the next term. The evaluation function must return a fixed-length string of digits.

A Moore machine can only have a finite number of states. It needs at least one state for each selector. It must also remember enough about the previous selectors in the term to compute the term's evaluation. Since terms can be arbitrarily long, no finite state machine can remember all of the previous selectors in the term. However, the evaluation function can often get by with much less information.

For example, when the evaluation function is an information metric, the evaluation of a partial term, $a_1 a_2 \ldots a_k$, depends only on the number of positive and negative examples covered by the term. This can be represented by 2^n states, where n is the number of examples. In this case, a state in the Moore machine is an n digit binary number, where the i^{th} digit indicates whether or not the example is covered by the term. In the initial state, all of the examples are covered. When a selector is seen, the digits corresponding to examples that are not covered by the selector are turned off. The binary vector for the state indicates which examples are covered, and the output string for the state is the information corresponding to that coverage of the examples.[1] When an *or* is seen, the DFA prints a zero to indicate end-of-term, and returns to the initial state.

This Moore machine is parameterized by the list of examples and the evaluation function f. This machine is substituted into the regular expression for comparing digit strings. The resulting DFA takes recognizes a pair of hypotheses, $\langle h_1, h_2\rangle$, if and only if h_1 comes before h_2 in the hill climbing search.

Although the machine has an exponential number of states, they do not need to be represented extensionally. All that must be maintained is the current state (an n digit binary number). The next state can be computed from the current state and a selector by determining which examples are not covered by the selector, and turning off those bits. This requires at most $O(n)$ space and $O(mn)$ time to evaluate a hypothesis, where n is the number of examples, and m is the number of selectors in the hypothesis.

The translator for this knowledge source takes as input the hypothesis space, the list of examples, and an evaluation function, f. The function f takes as input the number of covered and uncovered examples, and outputs a fixed length non-negative integer. The translator returns $\langle H, H, P\rangle$, where P is the grammar described above. $\langle H, H, P\rangle$ prefers hypotheses that are visited earlier by hill climbing with evaluation function f. This kind of bias is used in a number of induction algorithms, so this translator can be used for them as well.

Although the logic behind the LEF translator is

[1] Since information is a real between -1 and 1, and the output must be a fixed-length non-negative integer, the output string for a state is the integer portion of $(info + 1.0) * 10^6$, where *info* is the information of the example partitioning represented by the n digit number for that state.

rather complex, the translator itself is fairly straightforward to write. The Moore machine requires only a handful of code to implement the next-state and output functions, and the digit-string comparison grammar is a simple regular expression. The design effort also transfers to other biases. The evaluation function can be changed, so long as it only needs to know which examples are covered by the current term, and the basic design can be reused for translators of similar biases.

Some of the difficulty in designing the LEF translator may be because the bias is designed for use in a hypothesis space search paradigm, and does not translate well to RS-KII. Bear in mind that the beam-search is an approximation of another bias, namely that the induced hypothesis should maximize the LEF. Finding a maximal hypothesis is intractable, so AQ-11 approximates it with a beam search. This particular approximation was chosen because it is easy to implement in the hypothesis-space search paradigm. However, RS-KII uses a different paradigm, so a different approximation of the "maximize the LEF" bias that is easier to express in RS-KII may be more appropriate.

Translators for Novel Biases

The following translators are for biases that AQ-11 does not utilize, namely consistency with one class of noisy examples, and an assumption that the target hypothesis is a specialization of an overgeneral domain theory.

Noisy Examples with Bounded Inconsistency Bounded inconsistency (Hirsh 1990) is a kind of noise in which each feature of the example can be wrong by at most a fixed amount. For example, if the width value for each instance is measured by an instrument with a maximum error of ±0.3mm, then the width values for these instances have bounded inconsistency.

The idea for translating examples with bounded inconsistency is to use the error margin to work backwards from the noisy example to compute the set of possible noise-free examples. One of these examples is the correct noise-free version of the observed example, into which noise was introduced to produce the observed noisy example. The target concept is strictly consistent with this noise-free example.

Let e be the noisy observed example, E be the set of noise-free examples from which e could have been generated, and let e' be the correct noise-free example from which e was in fact generated. Since it is unknown which example in E is e', a noisy example is translated as $\langle H, C, \emptyset \rangle$, where C is the set of hypotheses that are strictly consistent with one or more of the examples in E. Hypotheses that are consistent with none of the examples in E are not consistent with e', and therefore not the target concept. This is the approach used by Hirsh (Hirsh 1990) in IVSM to translate noisy examples with bounded inconsistency.

$$\begin{aligned} &\mathit{TranPosExampleBI}(H, \langle \delta_1, \delta_2, \ldots, \delta_k \rangle, \\ &\qquad \langle x_1, x_2, \ldots, x_k \rangle) \rightarrow \langle H, C, P \rangle \\ E &= [x_1, \pm\delta_1] \times [x_2, \pm\delta_2] \times \ldots [x_k, \pm\delta_k] \\ &\quad \text{where } [x_i, \pm\delta_i] = \{v \mid x_i - \delta_i \leq v \leq x_i + \delta_i\} \\ C &= \bigcup_{e_i \in E} C_i \text{ s.t. } \langle C_i, \emptyset \rangle = \mathit{TranPosAQExample}(H, e_i) \end{aligned}$$

Figure 4: RS-KII Translator for Positive Examples with Bounded Inconsistency.

This suggests the following RS-KII translator for examples with bounded inconsistency. The set of possible noise-free examples, E, is computed from the noisy examples and the error margins for each feature. Each example, e_i, in this set is translated using one of the RS-KII translators for noise-free examples—either $\mathit{TranPosAQExample}(H, e_i)$ or $\mathit{TranNegAQExample}(H, e_i)$—which translates example e_i into $\langle H, C_i, \emptyset \rangle$. C_i is the set of hypotheses that are strictly consistent with e_i. The translator for the bounded inconsistent example returns $\langle C = \bigcup_{i=1}^{|E|} C_i, \emptyset \rangle$. C is the set of hypotheses consistent with at least one of the examples in E.

The set E is computed from the observed example, $\langle x_1, x_2, \ldots, x_k \rangle$, and the error margins for each feature, $\pm\delta_1$ through $\pm\delta_k$, as follows. If the observed value for feature f_i is x_i, and the error margin is $\pm\delta_i$, then the correct value for feature f_i is in $\{v \mid x_i - \delta_i \leq v \leq x_i + \delta_i\}$. Call this set $[x_i, \pm\delta_i]$ for short. Since instances are ordered vectors of feature values, E is $[x_1, \pm\delta_1] \times [x_2, \pm\delta_2] \times \ldots \times [x_k, \pm\delta_k]$.

A translator for examples with bounded inconsistency based on this approach is shown in Figure 4. It takes as input a VL_1 hypothesis space (H), the error margin for each feature ($\pm\delta_1$ through $\pm\delta_k$) and an instance. Negative examples are translated similarly, except that $\mathit{TranNegAQExample}(H, e_i)$ is used.

Domain Theory A domain theory encodes background knowledge about the target concept as a collection of horn-clause inference rules that explain why an instance is a member of the target concept. The way in which this knowledge biases induction depends on assumptions about the correctness and completeness of the theory. Each of these assumptions requires a different translator, since the biases map onto different constraints and preferences.

A translator for a particular overgeneral domain theory is described below. The theory being translated is derived from the classic "cup" theory (Mitchell, Keller, & Kedar-Cabelli 1986; Winston *et al.* 1983), and is shown in Figure 5. It expands into a set of sufficient conditions for `cup(X)`, as shown in Figure 6. The translator assumes that the target concept is a specialization of the theory. In this case, the actual target concept

```
cup(X)          :-  hold_liquid(X), liftable(X),
                    stable(X), drinkfrom(X).
hold_liquid(X)  :-  plastic(X) | china(X) | metal(X).
liftable(X)     :-  small(X), graspable(X).
graspable(X)    :-  small(X), cylindrical(X) |
                    small(X), has_handle(X).
stable(X)       :-  flat_bottom(X).
drinkfrom(X)    :-  open_top(X).
```

Figure 5: CUP Domain Theory.

```
1.  cup(X)  :-  plastic(X), small(X), cylindrical(X),
                flat_bottom(X), open_top(X).
2.  cup(X)  :-  china(X), small(X), cylindrical(X),
                flat_bottom(X), open_top(X).
3.  cup(X)  :-  metal(X), small(X), cylindrical(X),
                flat_bottom(X), open_top(X).
4.  cup(X)  :-  plastic(X), small(X), has_handle(X),
                flat_bottom(X), open_top(X).
5.  cup(X)  :-  metal(X), small(X), has_handle(X),
                flat_bottom(X), open_top(X).
6.  cup(X)  :-  china(X), small(X), has_handle(X),
                flat_bottom(X), open_top(X).
```

Figure 6: Sufficient Conditions of the CUP Theory.

is "plastic cups without handles," which corresponds to condition one, but this information is not provided to the translator. All the translator knows is that the target concept can be described by a disjunction of one or more of the sufficient conditions in the cup theory.

The translator takes the theory and hypothesis space as input, and generates the tuple $\langle H, C, \{\}\rangle$, where C is satisfied by hypotheses equivalent to a disjunct of one or more of the theory's sufficient conditions. In general, the hypothesis space language may differ from the language of the conditions, making it difficult to determine equivalence. However, for the VL_1 language of AQ-11, the languages are similar enough that simple syntactic equivalence will suffice, modulo a few cosmetic changes. Specifically, the predicates in the sufficient conditions are replaced by corresponding selectors. All disjuncts of the resulting conditions are VL_1 hypotheses. The mappings are shown in Figure 7. In general, the predicates are Boolean valued, and are replaced by Boolean valued selectors. To show that other mappings are also possible, the predicate `small(x)` is replaced by the selector `[size` $\leq$ `5]`.

The grammar for C is essentially the grammar for the cup theory, with a few additional rules. First, the cup theory is written as a context free grammar that generates the sufficient conditions. If the grammar does not have certain kinds of recursion, as is the case in the CUP theory, then it is in fact a regular grammar. In this case, the grammar for C will also be regular. Otherwise, the grammar for C will be context free. This limits the theories that can be utilized by RS-KII. However, RS-KII could be extended to utilize

```
plastic(x)      →  [plastic = true]
china(x)        →  [china = true]
metal(x)        →  [metal = true]
has_handle(x)   →  [has_handle = true]
cylindrical(x)  →  [cylindrical = true]
small(x)        →  [size ≤ 5]
flat_bottom(x)  →  [flat_bottom = true]
open_top(x)     →  [open_top = true]
```

Figure 7: Selectors Corresponding to Predicates in CUP Theory.

```
C               →  TERM | C or TERM
TERM            →  CONDITION
CONDITION       →  CUP(X)

PLASTIC(X)      →  [plastic = true]
CHINA(X)        →  [china = true]
METAL(X)        →  [metal = true]
HAS_HANDLE(X)   →  [has_handle = true]
CYLINDRICAL(X)  →  [cylindrical = true]
SMALL(X)        →  [size ≤ 5]
FLAT_BOTTOM(X)  →  [flat_bottom = true]
OPEN_TOP(X)     →  [open_top = true]
```

Figure 8: Grammar for VL_1 Hypotheses Satisfying the CUP Theory Bias.

a context free theory by allowing the C set of at most one $\langle H, C, P\rangle$ tuple to be context free. This would be a different instantiation of KII, but still within the expressiveness limits discussed in the previous section.

Once the theory has been written as a grammar, rewrite rules are added that map each terminal predicate (those that appear in the sufficient conditions) onto the corresponding selector(s). This grammar generates VL_1 hypotheses equivalent to each of the sufficient conditions. To get all possible disjuncts, rules are added that correspond to the regular expression CONDITION (*or* CONDITION)*, where CONDITION is the head of the domain-theory grammar described above.

The grammar for C discussed above is shown in Figure 8. This grammar is a little less general than it could be, since it does not allow all permutations of the selectors within each term. However, the more general grammar contains considerably more rules, and permuting the selectors does not change the semantics of a hypothesis. In the following grammar, the nonterminal CUP(X) is the head of the cup domain-theory grammar, which has the same structure as the theory shown in Figure 5.

Enumerating the Solution Set

The solution set is a regular grammar computed from C and P, as was shown in Equation 2. A regular grammar is equivalent to a deterministic finite automaton

(DFA). One straightforward way to enumerate a string from the solution set is to search the DFA for a path from the start state to an accept state. However, the DFA computed by the solution-set equation from C and P can contain *dead* states, from which there is no path to an accept state. These dead states can cause a large amount of expensive backtracking.

There is a second approach that can reduce backtracking by making better use of the dominance information in P. The solution set consists of the undominated strings in C, where P is the dominance relation. Strings in this set can be enumerated by searching C with branch-and-bound (Kumar 1992). The basic branch-and-bound search must be modified to use a partially ordered dominance relation rather than a totally ordered one, and to return multiple solutions instead of just one. These modifications are relatively straightforward, and are described in (Smith 1995).

Although the worst-case complexity of branch-and-bound is the same as a blind search of the solution-set DFA, the complexity of enumerating the first few hypotheses with branch-and-bound can be significantly less. Since for most applications only one or two hypotheses are ever needed, RS-KII uses branch-and-bound.

Results

By combining biases, different induction algorithms can be generated. AQ-11 uses the biases of strict consistency with examples, and prefers hypotheses that maximize the LEF. When using only these biases, both RS-KII and AQ-11 with a beam width of one induce the same hypotheses, though RS-KII is slightly more computationally expensive. The complexity of AQ-11 with a beam-size of one is $O(e^4k)$, where e is the number of examples and k is the number of features. The complexity of RS-KII when using only AQ-11 biases is $O(e^5k^2)$. These derivations can be found in (Smith 1995), and generally follow the complexity derivations for AQ-11 in (Clark & Niblett 1989). RS-KII is a little more costly because it assumes that the LEF bias, encoded by P, is a partial order, where it is in fact a total order. This causes RS-KII to make unnecessary comparisons that AQ-11 avoids. One could imagine a version of RS-KII which used information about whether P was a total order or a partial order.

RS-KII's strength lies in its ability to utilize additional knowledge, such as the domain theory and noisy examples with bounded inconsistency. When the domain theory translator is added, RS-KII's complexity drops considerably, since the hypothesis space is reduced to a relative handful of hypotheses by the strong bias of the domain theory. The concept induced by RS-KII is also more accurate than that learned by AQ-11, which cannot utilize the domain theory. When given the four examples of the concept "plastic cups without handles," as shown in Table 3, AQ-11 learns the overgeneral concept

ID	class	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8
e_1	+	t	f	f	f	t	5	t	t
e_2	+	t	f	f	f	t	3	t	t
e_3	−	f	t	f	f	t	4	t	t
e_4	−	t	f	f	t	t	1	t	t

f_1	plastic	f_5	has_handle
f_2	china	f_6	size
f_3	metal	f_7	flat_bottom
f_4	cylindrical	f_8	open_top

Table 3: Examples for the CUP Task.

```
[plastic = true] [cylindrical = true]
```

which includes many non-cups, whereas RS-KII learns the correct concept:

```
[plastic = true] [cylindrical = true]
[size ≤ 5] [flat_bottom = true]
[open_top = true]
```

The additional bias from the domain theory makes this the shortest concept consistent with the four examples.

RS-KII can also handle noisy examples with bounded inconsistency. For the cup domain, assume that the size can be off by at most one. Let the size feature of example e_2 be six instead of five. AQ-11 would fail to induce a hypothesis at all, since there is no hypothesis consistent with all four examples. When using the bounded-inconsistency translator for examples, RS-KII *can* induce a hypothesis, namely the same one learned above with noise-free examples. In general, noisy examples introduce uncertainty, which can increase the size of the solution set and decrease the accuracy of the learned hypothesis. Additional knowledge may be necessary to mitigate these effects. In this case, however, the domain theory bias is sufficiently strong, and the noise sufficiently weak, that no additional knowledge is needed.

The ability to utilize additional knowledge allows RS-KII to induce hypotheses in situations where AQ-11 cannot, and allows RS-KII to induce more accurate hypotheses. RS-KII can also make use of knowledge other than those shown here by writing appropriate translators.

Precursors to KII

KII has its roots in two knowledge integration systems, Incremental Version Space Merging (Hirsh 1990), and Grendel (Cohen 1992). These systems can also be instantiated from KII, given appropriate set representations. These systems and their relation to KII are described below.

IVSM. Incremental Version Space Merging (IVSM) (Hirsh 1990) was one of the first knowledge integration

systems for induction, and provided much of the motivation for KII. IVSM integrates knowledge by translating each knowledge fragment into a version space of hypotheses consistent with the knowledge, and then intersecting these version spaces to obtain a version space consistent with all of the knowledge. Version spaces map onto $\langle H, C, P \rangle$ tuples in which C is a version space in the traditional $[S, G]$ representation, and P is the empty set (i.e., no preference information).

KII expands on IVSM by extending the space of set representations from the traditional [S,G] representation—and a handful of alternative representations (e.g., (Hirsh 1992; Smith & Rosenbloom 1990; Subramanian & Feigenbaum 1986))—to the space of all possible set representations. KII also expands on IVSM by allowing knowledge to be expressed in terms of preferences as well as constraints, thereby increasing the kinds of knowledge that can be utilized. KII strictly subsumes IVSM, in that IVSM can be cast as an instantiation of KII in which C is a version space one of the possible representations, and P is expressed in the *null representation*, which can only represent the empty set.

Grendel. Grendel (Cohen 1992) is another cognitive ancestor of KII. The motivation for Grendel is to express biases explicitly in order to understand their effect on induction. The biases are translated into a context free grammar representing the biased hypothesis space.[2] This space is then searched for a hypothesis that is strictly consistent with the examples, under the guidance of an information gain metric. Some simple information can also be encoded in the grammar.

Grendel cannot easily integrate new knowledge. Context free grammars are not closed under intersection (Hopcroft & Ullman 1979), so it is not possible to generate a grammar for the new knowledge and intersect it with the existing grammar. Instead, a new grammar must be constructed for all of the biases. KII can use set representations that are closed under intersection, which allows KII to add or omit knowledge much more flexibly than Grendel. KII also has a richer language for expressing preferences. Grendel-like behavior can be obtained by instantiating KII with a context free grammar for C.

Future Work

One prime area for future work is constructing RS-KII translators for other biases and knowledge sources, especially those used by other induction algorithms. This is both to extend the range of knowledge available to RS-KII, and to test the limits of its expressiveness with respect to existing algorithms.

A second area is investigating the naturalness of the $\langle H, C, P \rangle$ representation. In RS-KII, some of the knowledge in AQ-11 is easy to express as $\langle H, C, P \rangle$ tuples, but some, such as the LEF, is more awkward. Others, such as the beam search bias, cannot be expressed at all in RS-KII. One approach is to replace this hard-to-express knowledge with knowledge that achieves similar effects on induction, but is easier to express. Similar approaches are used implicitly in existing algorithms for knowledge that cannot be easily used by the search. For example, AQ11 approximates a bias for the best hypothesis with a beam search that finds a locally maximal hypothesis.

Finally, the space of set representations should be investigated further to find representations that will yield other useful instantiations of KII. In particular, it would be worth identifying a set representation that can integrate n knowledge fragments and enumerate a hypothesis from the solution set in time polynomial in n. This would provide a tractable knowledge integration algorithm. Additionally, the set representation for the instantiation effectively defines a class of knowledge from which hypotheses can be induced in polynomial time. This would complement the results in the PAC literature, which deal with polynomial-time learning from examples only (e.g.,(Vapnik & Chervonenkis 1971), (Valiant 1984), (Blummer *et al.* 1989)).

Conclusions

Integrating additional knowledge is one of the most powerful ways to increase the accuracy and reduce the cost of induction. KII provides a uniform mechanism for doing so. KII also addresses an apparently inherent trade-off between the breadth of knowledge utilized and the cost of induction. KII can vary the trade-off by changing the set representation. RS-KII is an instantiation of KII with regular sets that shows promise for being able to integrate a wide range of knowledge and related strategies, thereby creating hybrid multi-strategy algorithms that make better use of the available knowledge. One such hybridization of AQ-11 was demonstrated. Other instantiations of KII may provide similarly useful algorithms, as demonstrated by IVSM and Grendel.

Acknowledgments

Thanks to Haym Hirsh for many helpful discussions during the formative stages of this work. This paper describes work that was supported by the National Aeronautics and Space Administration (NASA Ames Research Center) under cooperative agreement number NCC 2-538, and by the Information Systems Office of the Advanced Research Projects Agency (ARPA/ISO) and the Naval Command, Control and Ocean Surveillance Center RDT&E Division (NRaD) under contract number N66001-95-C-6013, and partially supported by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

[2] More precisely, they are are expressed as an antecedent description grammar.

References

Blummer, A.; Ehrenfect, A.; Haussler, D.; and Warmuth, M. 1989. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery* 36(4):929–965.

Chomsky, N. 1959. On certain formal properties of grammars. *Information and Control* 2.

Clark, P., and Niblett, T. 1989. The CN2 induction algorithm. *Machine Learning* 3(?):261–283.

Cohen, W. W. 1992. Compiling prior knowledge into an explicit bias. In Sleeman, D., and Edwards, P., eds., *Machine Learning: Proceedings of the Ninth International Workshop*, 102–110.

Hirsh, H. 1990. *Incremental Version Space Merging: A General Framework for Concept Learning.* Boston, MA: Kluwer Academic Publishers.

Hirsh, H. 1992. Polynomial-time learning with version spaces. In *AAAI-92: Proceedings, Tenth National Conference on Artificial Intelligence*, 117–122.

Hopcroft, J. E., and Ullman, J. D. 1979. *Introduction to Automata Theory, Languages, and Computation.* Reading, MA: Addison-Wesley.

Kumar, V. 1992. Search, branch and bound. In *Encylopedia of Artificial Intelligence.* John Wiley & Sons, Inc., second edition. 1000–1004.

Michalski, R. 1974. Variable-valued logic: System VL_1. In *Proeedings of the Fourth International Symposium on Multiple-Valued Logic.*

Michalski, R. 1978. Selection of most representative training examples and incremental generation of VL_1 hypotheses: The underlying methodology and the descriptions of programs ESEL and AQ11. Technical Report 877, Department of Computer Science, University of Illinois, Urbana, Illinois.

Mitchell, T.; Keller, R.; and Kedar-Cabelli, S. 1986. Explanation-based generalization: A unifying view. *Machine Learning* 1:47–80.

Mitchell, T. 1982. Generalization as search. *Artificial Intelligence* 18(2):203–226.

Quinlan, J. 1986. Induction of decision trees. *Machine Learning* 1:81–106.

Quinlan, J. R. 1990. Learning logical definitions from relations. *Machine Learning* 5:239–266.

Rosenbloom, P. S.; Hirsh, H.; Cohen, W. W.; and Smith, B. D. 1993. Two frameworks for integrating knowledge in induction. In Krishen, K., ed., *Seventh Annual Workshop on Space Operations, Applications, and Research (SOAR '93)*, 226–233. Houston, TX: Space Technology Interdependency Group. NASA Conference Publication 3240.

Russell, S., and Grosof, B. 1987. A declarative approach to bias in concept learning. In *Sixth national conference on artificial intelligence*, 505–510. Seattle, WA: AAAI.

Smith, B., and Rosenbloom, P. 1990. Incremental non-backtracking focusing: A polynomially bounded generalization algorithm for version spaces. In *Proceedings of the Eighth Naional Conference on Artificial Intelligence*, 848–853. Boston, MA: AAAI.

Smith, B. 1995. *Induction as Knowledge Integration.* Ph.D. Dissertation, University of Southern California, Los Angeles, CA.

Subramanian, D., and Feigenbaum, J. 1986. Factorization in experiment generation. In *Proceedings of the National Conference on Artificial Intelligence*, 518–522.

Valiant, L. 1984. A theory of the learnable. *Communications of the ACM* 27(11):1134–1142.

Vapnik, V., and Chervonenkis, A. 1971. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications* 16(2):264–280.

Winston, P.; Binford, T.; Katz, B.; and Lowry, M. 1983. Learning physical descriptions from functional definitions, examples, and precedents. In *Proceedings of the National Conference on Artificial Intelligence*, 433–439. Washington, D.C.: AAAI.

Fusing the Results of Diverse Algorithms

John F. Elder IV, Ph.D.

Elder Consulting
1006 Wildmere Place
Charlottesville VA 22901
elder@stat.rice.edu

Abstract

Structurally adaptive methods -- from decision trees and polynomial networks, to projection pursuit models, additive networks, and cascade correlation neural networks -- iteratively add components in an attempt to construct models with complexity appropriate to the data. The diverse basis functions and search strategies employed usually lead to a distribution of results (with rankings hard to predict *a priori*) yet, robustly combining the output estimates can achieve a consensus model with properties often superior to the best of the individual models. Additionally, other information learned by some of the modeling techniques can be shared to the benefit of the fused system, including identification of key variables to employ and outlying cases to ignore.

This paper describes the fused model developed for a small but challenging classification dataset (where one infers the species of a bat from its chirps) and introduces a robust method of combining the outputs of diverse models.

Inductive Modeling Toolbox

Though new inductive methods are introduced continually, they appear to be motivated by only a handful of key underlying ideas, and it is likely that a suite of methods can be identified which will essentially span "method space". In a readable empirical survey, Michie, Spiegelhalter and Taylor (1994)[1] suggested such a sufficient toolbox would include linear discrimination, decision trees, and *k*-nearest neighbors, as well as the more recent methods of projection pursuit and radial basis functions. Elder and Pregibon (1996) further suggest adding polynomial networks and (perhaps) adaptive splines and neural networks to the set of covering methods, due to some relatively unique properties of those algorithms. Four of these methods -- decision trees, *k*-nearest neighbors, neural networks, and polynomial networks -- were selected to be used, in a multi-stage procedure, to address a challenging classification problem.

[1] Reviewed by Elder (1996).

Application Example: Identifying Bat Species

Tracking populations of potentially endangered bats can be simplified if similar species can be distinguished using features of their sophisticated echolocation signals. Data recently collected and analyzed by Kaefer et al. (1996) demonstrated that the necessary information is present in the signals, but that a fused model (described here) seems required to extract it. The data was obtained by capturing and labeling bats, recording several of their in-flight biosonar calls, extracting Fourier and time-frequency features from such "chirps", and iterating the signal feature extraction phase after analyzing early returns from individual induction algorithms charged with constructing discriminating classifiers. The data consists of 93 cases representing 18 different bats (with 3-8 signals per individual) from 5 species (some quite related physically)[2] all of which emit calls in the FM range (a spectrum area previously unexplored for the purposes of discrimination).

Because there were multiple signals from each individual, a voting mechanism could be employed to arrive at a final classification for each set (bat). That is, a bat was counted as correctly classified if a plurality of its acceptable signals were correctly identified. (The acceptance process is defined below.)

For each algorithm, cross-validation accuracy was measured for both case-wise and bat-wise identification. That is, each method trained a model using 17 of the bat signal sets, then tested it on the 18th set (with this repeated 18 times so each set could be the hold-out sample in turn) and the collection of test results accumulated. Thus, the results reflect the accuracy of each algorithm (represented by a bundle of models) when applied to *new* individuals, as is required to estimate potential field performance.

Decision Trees

Decision tree algorithms (e.g., CART; Breiman, et al., 1984) try to discriminate between classes by recursively

[2] Physically distinguishing one pair of species, for example, apparently requires examination of the underside of the bat's ear.

partitioning the data until the resulting groups are as pure as can be sustained. They start with all the training data and find the single (typically univariate) threshold split which divides the sample into two maximally pure parts. Each part is then similarly recursively subdivided until either no splits are possible, the classification is perfect, or the partitions reach some minimum size. As this maximal tree is typically too complex to be useful on new cases (it overfits), CART then prunes back the tree, using cross-validation to decide how much to simplify. (Other methods use a complexity penalty to determine a forward stopping point.)

Tree methods use a "greedy'" search scheme, optimizing each successive step. Yet, it is well known that greedy algorithms may not result in the best classifier. The TX2step decision tree algorithm (Elder, 1994) enhances this aspect of decision trees by looking two steps ahead in choosing the split, often thereby improving performance (though at the cost of squaring the *KN* runtime for *K* variables and *N* cases).

As expected, the 2-step trees were more accurate on training and usually more complex). Yet they also fared better on evaluation, where the one-step trees got 46% of the cases correct, and only 7 of the 18 bats. The 2-step trees correctly identified 58% of the cases and 11 of 18 bats.

Backpropagation Neural Networks

Decision trees form sharp decision boundaries, transitioning immediately from one class to another. Smoother, more probabilistic, transitions can be more robust, however. Backpropagation Neural Networks (BNNs) are nonlinear estimators which can be applied to the problem of classification, typically by defining a network with an output node representing each class, an input node for each input variable, and one or more intermediate (hidden) layers. The nodes are nonlinear -- weighted sums followed by a logistic transformation. Using the training data, the weights in the network are adjusted according to a type of gradient search to cause the correct output node to take on a value near unity and the other output nodes to approach zero. When classifying an unknown case, its input variable values are run through the network, and the class with the largest output node value is selected. (Note that, though not used here, "optimal scoring" -- a superior way to use the output of estimative methods for classification, has recently been introduced by Hastie, Tibshirani, & Buja, 1994.) Though quite slow to train, BNNs do have the ability to construct nonlinear classification functions of greater variety than simpler methods such as linear regression or decision trees.

We[1] employed networks with three layers and varying *K*, with the number of nodes in the hidden layer proportional to *log(K)*. (The internal network training parameters were set after some experimentation to be: learning rate = 0.3, momentum = 0.8, tolerance = 0.1, iterations = 40,000.) As the logistic activation function for each node does not reach its extreme values of 1 or 0 without infinitely large weights, we instead set targets of 0.9 and 0.1.

The first set of BNNs trained using the 35 original variables was 52% accurate on the cases, and showed signs of overfit. Therefore, correlation and visualization information was employed to reduce the number of input variables to 17, improving the case-wise cross-validation (CV) accuracy to 63%. Further reducing the input set to the 8 variables selected by the decision tree methods boosted the BNN results to 69%. After voting, this last BNN set got 14 of 18 bats correct.

Nearest Neighbors

Exemplar-based statistical techniques, such as nearest neighbors and kernel methods, attempt classification by more direct comparison of test data with the training data itself. In *k*-nearest neighbors (*k* -NN), a new case is labeled with the class of the majority of the *k* closest examples in the training set, usually using Euclidean distance as the metric. This simple non-parametric method is very flexible and straightforward; furthermore, it has been proven that, given enough training data, the error rate is less than twice that of the theoretical optimum (Cover and Hart, 1967).

However, moreso than with methods which select variables, *k*-NN is sensitive to the dimensions chosen, as any unnecessary one contributes to the distance calculations and thereby obscures the true relationships. So, the 8 dimensions found useful by the tree methods were employed, and a 5-NN algorithm was 70% accurate on the cases and hit 14.5 of 18 bats (counting ties as half right). Further, when all subsets of these 8 variables were exhaustively explored, a set of 5 dimensions were found with 80% accuracy, indicating the importance of dimension selection for the algorithm. (However, it is not valid to use this performance number, as the CV accuracy was maximized through a search; another level of CV would be required to get an honest estimate of performance on truly new data.)

Although the input set and the resulting case accuracy of the *k*-NN and BNN methods are equivalent, they disagreed on 35% of the case labels. (Also, the *k*-NN took almost no time to run, whereas the Matlab implementation of the BNNs suite required about a day of processing power.) This degree of difference suggests that some cases are easier for one method to handle than the other and that combining them (below) would likely be fruitful.

Polynomial Networks

Another powerful adaptive modeling method escapes the limitations of linearity through employing compositions of nonlinear polynomials. Polynomial networks (PNs) (e.g., GMDH, Farlow 1984; ASPN, Elder & Brown, 1992) use linear least squares regression to build nodes (groups of nonlinear terms) from combinations of the inputs. ASPN

[1] Neural network runs were performaed by Oliver Kaefer and Doug Jones of the Univ. Illinois Urbana/Champaigne EE Dept.

selects the best several nodes for membership in the current layer using a criterion which penalizes complexity as well as error (to avoid overfit). Within each node, reverse elimination is used to pare away terms not contributing sufficiently to its accuracy. The building of layers continues until the modeling criterion can no longer be improved. A great strength of the method is that the structure of the network (its inputs, connections, and terms) does not have to be pre-specified, as is required with neural networks, but instead adapts automatically toward the structure of data. The resulting multiple layers of diverse polynomial nodes, each with several terms, is a flexible compound function which smoothly interpolates the sample region but, because of its use of polynomials, extrapolates poorly beyond the data boundaries.

The suite of PNs was 62% accurate on the cases but actually performed best of all individual methods on getting bats right: 15 of 18, *when outliers were ignored*, as described below.

Case Elimination

Three of the methods -- decision trees, BNNs, and *k*-NN -- all degrade rather gracefully compared to the last; since PNs employ global polynomials, they can easily "explode" on extrapolated cases. This is normally considered a great negative (e.g., Friedman, 1991) but can, in a voting situation, be turned into an important advantage. With nonlinear PNs, cases sufficiently distant from the convex hull of the training data space can have estimates many orders of magnitude more positive or negative than the target values [0-1]. (Experiments showed that a few orders of magnitude off could be tolerated.) When this occurs, it can signal the system that a "don't know" answer is more appropriate than the typical approach of limiting the response to the nearest boundary.

Further, when all the output nodes are close to a zero value, it is generally better to leave the case out of the voting scheme, than to "split hairs" among uncertain estimates. Here, a PN output summation threshold of 0.1 eliminated 12% of the cases. When this extra information from the PN was employed within the *other* methods, the results of each improved slightly.

Combining Outputs

It is an ancient statistical principle[1] that merging a set of independent estimates can reduce the variance of the output. First, to smooth the contributions of the boundary-based methods, the leaf nodes of the trees were labeled not with the winning class, but with the sample distribution. Similarly for the 5-NN method; each sample neighbor contributed a mass of 0.2 to the distribution. (Late in the experiment it was discovered that simply averaging the 5-NN distributions over a set of signals from a single individual and *then* taking the maximum worked quite well, getting 16 of 18 bats correct. However, for the following experiments, the outputs were treated like the other methods, labeling each case with its maximum, then voting among the accepted cases making up one set.)

Simply averaging the outputs of the four techniques before voting did improve the CV case accuracy to 74%. When the weights for each method's outputs were made proportional to their individual performance, this increased to 78%. Then, when the polynomial network was allowed to identify and remove outliers for all four techniques, 82% accuracy was obtained.

Advisor Perceptrons

Weighted sums can be sensitive to outlying estimates which can dominate any combination. To bound the influence of each method, the technique of "advisor perceptrons" (AP) -- originally developed to induce binary models (Elder, 1992) -- was employed. APs are motivated by the scenario of a decision maker faced with a binary choice (e.g., buy/sell, class 1/not class 1) who had d advisors offering binary recommendations. It was shown that there are a finite number of possible weight sets which lead to distinct decision maps, and that a linear program could generate the list. For example, with four advisors, there are only 12 sets to enumerate: corresponding to giving double weight to one advisor (2111, 1211, 1121, 1112), ignoring one advisor (0111, 1011, 1101, 1110), or employing only one advisor (1000, 0100, 0010, 0001). (For five inputs the list grows to a size of 81, and is in the billions by $d = 9$.)

Recently, Lee (1996) employed APs to combine five methods -- logistic regression, decision trees, projection pursuit regression, BNNs, and learning vector quantization -- to achieve results superior to averaging on each of four diverse data sets. This is somewhat surprising, as one is throwing away some probability information by discritizing each estimate. However, the added robustness can be quite useful on new data.

Here, there were three perceptron weight sets with similar strong performance (correct on 16 of 18 bats), corresponding to, respectively: doubling the BNN influence, or removing either the tree or PN contributions to the output estimation. (These results were not subject to CV checking, so must be relied on with caution. They are essentially equivalent though to the performance obtained by optimizing the combination weights in hindsight, which suggests that the upper limit of accuracy on this data is hitting 16.5 bats out of 18.)

Conclusions

Diverse induction methods have different strengths which can often be tapped by combining their information. Such *fusion* requires merging their outputs, either say, by weighting (based on individual performance) or using the robust technique of "advisor perceptrons". But it also

[1] Proverbs 24:6b: "... in a multitude of counselors there is safety."

involves sharing information learned about the data between techniques. Here, on a challenging species classification problem, the methods which select variables (decsion trees and polynomial networks) were able to improve the cross-validation performance of those which cannot (backpropagation neural networks, and k-nearest neighbors). Also, a hitherto negative property of polynomial networks -- their tendancy to "explode" in extrapolation situations -- was harnessed to identify cases which should not contribute to the vote among a set, and thereby turned into a benefit.

References

Breiman, L., J. H. Friedman, R. A. Olshen, & C. J. Stone 1984. *Classification and Regression Trees*. Wadsworth & Brooks, Pacific Grove, CA.

Cover, T. M., & P. E. Hart 1967. Nearest Neighbor Pattern Classification. *IEEE Transactions on Information Theory* 13: 21-27.

Elder, J. F. IV 1992. Optimal Discrete Perceptrons for Graded Learning, *International Systems, Man, and Cybernetics Conf,* Chicago, IL, Oct. 18-21.

Elder, J. F. IV 1994. Inducing Models less Greedily, *International Systems, Man, and Cybernetics Conf.,* San Antonio, TX, Oct. 2-5, pp. 908-912.

Elder, J. F. IV 1996. A review of *Machine Learning, Neural and Statistical Classification*, (eds. Michie, Spiegelhalter & Taylor; Ellis Horwood, 1994), *J. American Statistical Assoc.* 91, no. 433: 436-437.

Elder, J. F. IV & D. E. Brown 1992. Induction and Polynomial Networks, Univ. VA Tech. Rpt. IPC-TR-92-9. (Forthcoming as Chapter 3 in *Advances in Control Networks and Large Scale Parallel Distributed Processing Models* (Vol. 2), Ablex: Norwood, NJ.

Elder, J. F. IV & D. Pregibon 1996. A Statistical Perspective on Knowledge Discovery in Databases, Chapter 4 in *Advances in Knowledge Discovery and Data Mining*, eds. U. M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, AAAI/MIT Press.

Farlow, S. J., ed. 1984. *Self-Organizing Methods in Modeling: GMDH Type Algorithms.* Marcel Dekker, NY.

Friedman, J. H. 1991. Multiple Adaptive Regression Splines (with discussion). *Annals of Statistics* 19: 1-141.

Hastie, T., R. Tibshirani, & A. Buja 1994. Flexible Discriminant Analysis by Optimal Scoring. *J. American Statistical Assoc.* 89, no. 428: 1255-1270.

Kaefer, O., J. F. Elder IV, C. J. Condon, K. R. White, A. S. Feng, & D. L. Jones 1996. Classification and Discrimination of the Echolocation Signals of Six Species of Vespertilionid Bats from Illinois. Forthcoming.

Lee, S. S. 1996. Combining Neural and Statistical Classifiers Via Perceptron, *Proc. 2nd Annual Knowledge Discovery in Databases Conf.*, Aug. Forthcoming.

Michie, D., D. J. Spiegelhalter, & C. C. Taylor eds. 1994. *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, NY.

Learning Synthesis Schemes in Intelligent Systems

Lech Polkowski
Institute of Mathematics
Warsaw University of Technology
Pl. Politechniki 1, 00-650 Warsaw, Poland
polk@mimuw.edu.pl

Andrzej Skowron
Institute of Mathematics
Warsaw University
Banacha 2, 02-097 Warsaw, Poland
skowron@mimuw.edu.pl

Abstract

We present a settting in which one can discuss problems of design, synthesis, analysis and control of complex systems by adaptive teams of intelligent distributed agents. We point to learning problems of this approach related to the necesity of extracting from the empirical data of constructs allowing the agents to negotiate their co - operative actions. We put our analysis into the framework of multistrategy learning (Michalski 1994) which combines empirical induction, abduction and reasoning by analogy in a hierarchical setting (Michalski 1994).

Introduction

We propose a discussion of synthesis schemes for synthesis of complex objects defined by uncertain or incompletely understood requirements. Our synthesis schemes are constructed over sets of intelligent autonomous agents which co - operate and negotiate their goals towards synthesis of an appropriate artifact from some simple prescribed inventory objects. A requirement which approximately specifies a complex object comes to the agents from the external world and can be expressed in a language not fully understandable by the agents. See in this respect Example 1 below. The main result of the uncertainty caused by the language is that the agents are able to describe the complex object in their internal language approximately only; however, they strive to produce a complex artifact which satisfies the external requirement in satisfactory degree i.e. the external customer will accept this artifact as satisfying the requirement. To achieve this end, the agents form teams which co - operate towards creating the final artifact. The links among local teams of agents are formed as a result of negotiations among agents which result in a synthesis scheme; the scheme is locally robust i.e. each team of agents in the scheme has certain bounds of uncertainty within which to operate. The adaptiveness of the scheme is the result of its forming technique: when the external conditions change, the agents may re - negotiate their local uncertainty bounds as well as links among local teams which may lead to a new scheme with new boundary conditions. One may also admit the possibility of re -negotiating local goals which may lead to a new class of schemes.

Our principal aim is to give a formal model for accounting for uncertainty and approximative character of our schemes. We propose to this end to apply our recently developed (Polkowski & Skowron 1994a), (Polkowski & Skowron 1994b), (Skowron & Polkowski 1995), (Polkowski & Skowron 1995), (Polkowski & Skowron 1996a), (Polkowski & Skowron 1996b), theory of rough mereology. Rough mereology is a theory of a relation of being a part in a degree which can be traced back to mereology of Leśniewski (Leśniewski 1992). This theory allows for introducing a logic of approximative reasoning about complex objects (Komorowski, Polkowski, & Skowron 1996) in which, in particular, we are able to express the fact that a given artifact satisfies a given property in certain degree. Let us stress also that rough mereology encompasses fuzzy set theory: the notions of an element and the subset coincide in rough mereology (Polkowski & Skowron 1994), (Polkowski & Skowron 1996a) hence the relation of being a part in a degree can be regarded as a fuzzy membership function.

In our approach rough mereologies are assigned to individual agents in a distributed way: to each agent a rough inclusion (see below) is assigned which permits the agent to measure degrees of similarity among objects manipulated by it, and local mereologies of co - operating agents are related by means of propagating functors (mereological connectives) extracted (learned) from data.

Forming a synthesis scheme proceeds in two steps; first - the external requirement is absorbed as a constraint in a language of agents and it is decomposed into simpler constraints; any constraint is associated with an agent or a local team of agents which are able to fulfill it. The process of decomposition involves negotiations among agents which result in assignment to the agents of local constraints, assembling operations (decomposition rules) as well as uncertainty bounds. Constraints may be expressed in an informal language and they are rendered in logical languages of agents as approximate formulas in the form of a pair (Φ, ε)

where Φ is a predicate in the logic of the agent and ε is an uncertainty bound. In positive cases constraints reach the level of inventory (primitive) agents which are able to convert their approximate formulas into objects satisfying them.

The second step consists in assembling a complex artifact from primitive objects by means of negotiated operations.

Our approach is analytic: all constructs are learned from data tables which constitute the agents knowledge.

We present in what follows: preliminaries on mereology and rough mereology, logic for approximative reasoning (satisfactory fulfillment of a requirement), design and synthesis schemes, and the learning aspects of our approach. We conclude with a discussion of potential applications to problems of design, synthesis and control of complex systems.

Preliminaries: Mereology of Leśniewski

We recall here the basic notions of the mereological system of Leśniewski (Leśniewski 1992); in the next section the mereological system of Leśniewski will be extended to the system of approximate mereological calculus called *rough mereology*.

We consider a finite non - empty set U. A binary relation π on the set U will be called the *relation of being a (proper) part* in the case when the following conditions are fulfilled

(P1) (*irreflexivity*) for any $x \in U$, it is not true that $x\pi x$;

(P2) (*transitivity*) for any triple $x, y, z \in U$, if $x\pi y$ and $y\pi z$, then $x\pi z$.

It follows obviously from (P1) and (P2) that the following property holds

(P3) for any pair $x, y \in U$, if $x\pi y$ then it is not true that $y\pi x$.

In the case when $x\pi y$ we say that the object x is a (*proper*) *part* of the object y. The notion of being (possibly) an improper part is rendered by the notion of an ingredient; for objects $x, y \in U$, we say that the object x is a π-*ingredient* of the object y when either $x\pi y$ or $x = y$. We denote the relation of being a π-ingredient by the symbol $ingr(\pi)$; hence we can write

(I1) for $x, y \in U$, $x\ ingr(\pi)\ y$ iff $x\pi y$ or $x = y$.

It follows immediately from the definition that the relation of being an ingredient has the following properties:

(I2) (*reflexivity*) for any $x \in U$, we have $x\ ingr(\pi)\ x$;

(I3) (*weak antisymmetry*) for any pair $x, y \in U$, if $x\ ingr(\pi)\ y$ and $y\ ingr(\pi)\ x$ then $x = y$;

(I4) (*transitivity*) for any triple $x, y, z \in U$, if $x\ ingr(\pi)\ y$ and $y\ ingr(\pi)\ z$ then $x\ ingr(\pi)\ z$.

We will call any pair (U, π) where U is a finite set and π a binary relation on the set U which satisfies the conditions (P1) and (P2) a *pre-model of mereology*.

We now recall the notions of a set of objects and of a class of objects. For a given pre-model (U, π) of mereology and a property m which can be attributed to objects in U, we will say that an object x is an object m (x *object* m, for short) when the object x has the property m. The property m will be said to be non-void when there exists an object $x \in U$ such that x *object* m. Consider a non-void property m of objects in a set U where (U, π) is a pre-model of mereology.

An object $x \in U$ is said to be a *set of objects with the property* m when the following condition is fulfilled:

(SETm) for any $y \in U$, if y *object* m and $y\ ingr(\pi)\ x$ then there exist $z, t \in U$ with the properties: $z\ ingr(\pi)\ y$, $z\ ingr(\pi)\ t$, $t\ ingr(\pi)\ x$ and t *object* m.

We will use the symbol x *set* m to denote the fact that an object x is a set of objects with the property m.

Assume that x *set* m; if, in addition, the object x satisfies the condition

(CLm) for any $y \in U$, if y *object* m then $y\ ingr(\pi)\ x$ then we say that the object x is *a class of objects with the property* m and we denote this fact by the symbol x *class* m. We will say that a pair (U, π) is a model of mereology when the pair (U, π) is a pre-model of mereology and the condition

(EUC) for any non-void property m of objects in the set U, there exists a unique object x such that x *class* m holds.

The notions of a set and a class permit to regard collections of objects as objects; the intuitive, mathematical idea related to them is that of a set theoretical union. Our application of these notions is explained below, e.g., in design theory.

Rough mereology

An approximate mereological calculus called rough mereology has been proposed (Polkowski & Skowron 1994), (Polkowski & Skowron 1996a) as a formal treatment of the hierarchy of relations of being a part in a degree. We begin with an exposition of rough mereological calculus in the form of a logic L_{rm}.

Syntax of L_{rm}

It will be the standard syntax of the predicate calculus in which we will have the following basic ingredients:

Variables: $x, x_1, x_2, ...,\ y, y_1, y_2, ...,\ z, z_1, z_2, ...$ of type *set_element* and $r, r_1, r_2, ..., s, s_1, s_2, ...$ of type *lattice_element*;

Constants: ω of type *lattice_element*;

Predicate symbols, function symbols: $\leq$ of type (*lattice_element, lattice_element*) and μ of type (*set_element, set_element, lattice_element*);

Auxiliary symbols: propositional connectives: $\vee$, $\wedge$, $\Longrightarrow$, $\neg$, quantifier symbols: $\forall$, $\exists$ and commas, parentheses.

Formulae: atomic formulae are of the form $\mu(x,y,r)$, $s \leq r$ and formulae are built from atomic formulae as in the predicate calculus.

Axioms: the following are axioms of L_{rm}

(A1) $\forall x.\mu(x,x,\omega)$;

(A2) $\forall x.\forall y.\{\mu(x,y,\omega) \implies \forall s.\forall r.\forall z.[\mu(z,x,s) \wedge \mu(z,y,r) \implies (s \leq r)]\}$;

(A3) $\forall x.\forall y.\{\mu(x,y,\omega) \wedge \mu(y,x,\omega) \implies$

$\forall s.\forall r.\forall z.[\mu(x,z,s) \wedge \mu(y,z,r) \implies (s \leq r)]\}$;

(A4) $\exists x.\forall y.\mu(x,y,\omega)$;

(A5) $\forall x.\forall y.\{[\forall z.[[\exists u.\neg(\mu(z,u,\omega)) \wedge \mu(z,x,\omega)] \implies$

$\exists t.(\exists w.(\neg\mu(t,w,\omega)) \wedge \mu(t,z,\omega) \wedge \mu(t,y,\omega)] \implies$

$\mu(x,y,\omega)\}$;

and the axiom schemata $(A6)_n$ for n = 2,3,.... where

$(A6)_n$

$\forall x_1.\forall x_2....\forall x_n.\exists y.(\alpha_n(x_1,x_2,...,x_n,y) \wedge$

$\beta_n(x_1,x_2,...,x_n,y) \wedge \gamma_n(x_1,x_2,...,x_n,y))$

where

$\alpha_n(x_1,x_2,..,x_n,y)$:

$\forall z.\{[\exists t.(\neg\mu(z,t,\omega)) \wedge \mu(z,y,\omega)] \implies$

$\exists x_i.\exists w.[(\exists u.(\neg\mu(w,u,\omega))) \wedge \mu(w,z,\omega) \wedge \mu(w,x_i,\omega)]\}$;

$\beta_n(x_1,x_2,..,x_n,y)$:

$\mu(x_1,y,\omega) \wedge \mu(x_2,y,\omega) \wedge ... \wedge \mu(x_n,y,\omega)$;

$\gamma_n(x_1,x_2,..,x_n,y)$:

$\forall z.\{[\alpha_n(x_1,x_2,..,x_n,z) \wedge \beta_n(x_1,x_2,..,x_n,z)] \implies$

$\mu(y,z,\omega)\}$.

The formal introduction to rough mereology is expanded below when we discuss rough inclusions.

Semantics of L_{rm}

We will call an *interpretation* of L_{rm} a triple $M = (U^M, L^M, F^M)$ where U^M is a finite set, L^M is a (complete) lattice with the lattice partial order $\leq^M$ and with the greatest element Ω^M and F^M is a mapping which assigns to constants and predicate symbols of L_{rm} their denotations in M in the following manner: $F^M(\omega) = \Omega^M$, $F^M(\leq) = \leq^M$ and $F^M(\mu) = \mu^M \subseteq U^M \times U^M \times L^M$, where the relation $\mu^M \subseteq U^M \times U^M \times L^M$ is a function i.e. $\mu^M : U^M \times U^M \longrightarrow L^M$.

An M-value assignment g is a mapping which assigns to any variable x of L_{rm} of type *set_element* the element $g(x) \in U^M$ and to any variable r of L_{rm} of type *lattice_element* the element $g(r) \in L^M$. For an M-value assignment g, a variable x of L_{rm} of type *set_element* and an element $u \in U^M$, we denote by the symbol $g[u/x]$ the M-value assignment defined by the conditions: $g[u/x](v) = g(v)$ in case $v \neq x$ and $g[u/x](x) = u$; the same convention will define $g[p/r]$ in case of a variable r of type *lattice_element* and $p \in L^M$.

For a formula α of L_{rm}, we denote by the symbol $[\alpha]^{M,g}$ the meaning of the formula α in the model M relative to an M-value assignment g by the following conditions

(M1) $[\mu(x,y,r)]^{M,g} = true$ iff $\mu^M(g(x),g(y)) = p$ for some $p \geq^M g(r)$;

(M2) $[s \leq r]^{M,g} = true$ iff $g(s) \leq^M g(r)$;

(M3) $[\alpha \vee \beta]^{M,g} = true$ iff $[\alpha]^{M,g} = true$ or $[\beta]^{M,g} = true$;

(M4) $[\neg\alpha]^{M,g} = true$ iff $[\alpha]^{M,g} = false$;

(M5) $[\exists x.\alpha]^{M,g} = true$ iff there exists $u \in U^M$ such that $[\alpha]^{M,g[u/x]} = true$;

(M6) $[\exists r.\alpha]^{M,g} = true$ iff there exists $p \in L^M$ such that $[\alpha]^{M,g[p/r]} = true$.

It follows that the intended meaning of a formula $\mu(x,y,r)$ is that "*the object x is a part of the object y in degree at least r*".

A formula α is *true in an interpretation* M iff α is M,g-true (i.e. $[\alpha]^{M,g} = true$) for any M-value assignment g. An interpretation M is a *model* of L_{rm} iff all axioms (A1)-(A6) are true in M.

Rough inclusions. A real function $\mu^M(X,Y)$ on a universe of objects U^M with values in the interval $[0,1]$ is called *a rough inclusion.* It satisfies the following conditions which are translations of respective axioms $(A1) - (A6)_n$.

(A) $\mu^M(X,X) = 1$ for any X ;

(B) $\mu^M(X,Y) = 1$ implies that $\mu^M(Z,Y) \geq \mu^M(Z,X)$ for any triple X,Y,Z;

(C) there is N such that $\mu^M(N,X) = 1$ for any X. An object N satisfying (C) is a μ-null object: such objects are excluded in mereology of Leśniewski.

We let $X =_\mu Y$ iff $\mu^M(X,Y) = 1 = \mu^M(Y,X)$ and $X \neq_\mu Y$ iff $non(X =_\mu Y)$.

We have other conditions for rough inclusion:

(D) **if** objects X,Y have the property :

if $Z \neq_\mu N$ and $\mu^M(Z,X) = 1$

then there is $T \neq_\mu N$

with $\mu^M(T,Z) = 1 = \mu^M(T,Y)$

then it follows that: $\mu^M(X,Y) = 1$.

(D) is an inference rule: it is applied to infer the relation of being a part from the relation of being a subpart.

(E) For any collection Γ of objects there is an object X with the properties:

(i) if $Z \neq_\mu N$ and $\mu^M(Z,X) = 1$ then there are $T \neq_\mu N, W \in \Gamma$ such that

$$\mu^M(T,Z) = \mu^M(T,W) = \mu^M(W,X) = 1;$$

(ii) if $W \in \Gamma$ then $\mu^M(W,X) = 1$;

(iii) if Y satisfies the above two conditions then $\mu^M(X,Y) = 1$.

(E) will be applied below to show the existence and uniqueness of classes of objects.

We now outline the way in which mereology of Leśniewski follows out of rough mereology.

Rough inclusions: reduced models and Leśniewski's mereology

Given a model M of L_{rm}, $M = (U^M, L^M, F^M)$, we will call the function $\mu^M : U^M \times U^M \longrightarrow L^M$ the M-rough inclusion. We define a relation $congr(\mu^M)$ on the set U^M by letting for $u, w \in U^M$: $u\ congr(\mu^M)\ w$ iff $\mu^M(u,w) = \Omega^M = \mu^M(w,u)$. The following proposition, whose proof follows immediately by (A2) and (A3) and is therefore omitted, establishes the basic properties of the relation $congr(\mu^M)$ and demonstrates it to be a μ^M−congruence.

Proposition 1 *The relation $congr(\mu^M)$ is an equivalence relation on the set U^M and we have*
(i) if $u\ congr(\mu^M)\ w$ then $\mu^M(v,w) = \mu^M(v,u)$;
(ii) if $u\ congr(\mu^M)\ w$ then $\mu^M(u,v) = \mu^M(w,v)$
for any triple $u, v, w \in U^M$.

□

We denote by u_μ the class of $congr(\mu^M)$ which contains u. It follows that the rough inclusion can be factored throughout the relation $congr(\mu^M)$ i.e. we define the quotient set $U^M_\mu = U^M/congr(\mu^M)$ and the quotient function

$$\mu^M_\sim : U^M_\mu \times U^M_\mu \longrightarrow L^M$$

by letting $\mu^M_\mu(u_\mu, w_\mu) = \mu^M(u,w)$; clearly, the pair $(U^M_\mu, \mu^M_\sim)$ introduces a model $M_\sim$ of L_{rm}. In the sequel we will always work with a fixed reduced model $M_\sim$. We denote by the symbol n_μ the *null object* i.e. the object existing in virtue of (A4) and such that $\mu^M_\sim(n_\mu, w_\mu) = \Omega^M$ for any $w_\mu \in U^M_\mu$. We will write $u_\mu \neq_\mu n_\mu$ to denote the fact that the object u_μ is not the null object. Let us recall that the existence of a null object in a model of mereology of Leśniewski reduces the model to a singleton, as observed by Tarski. In the sequel, for simplicity of notation, we will write μ in place of $\mu^M_\sim$, U in place of U^M_μ, u in place of u_μ etc. We will call the rough inclusion μ a *strict rough inclusion* when it satisfies the condition $\mu(x,n) = 0$ for any non-null object x; we observe that any standard rough inclusion is strict.

We now show how the rough inclusion μ introduces in U a model of mereology of Leśniewski. To this end, we define a binary relation $part(\mu)$ on the set U by letting

$u\ part(\mu)\ w$ iff $\mu(u,w) = \Omega^M$ and it is not true that $\mu(w,u) = \Omega^M$.

Then we have the following proposition whose straightforward proof is omitted.

Proposition 2. (i) the relation $part(\mu)$ satisfies the conditions (P1) and (P2);

(ii) the relation $ingr(\ part(\mu)\ )$ satisfies the following for any pair $u, w \in U$:

$$u\ ingr(part(\mu))\ w \text{ iff } \mu(u,w) = \Omega^M.$$

□

We now define in the model $M_\sim$ for any collection Ψ of objects in U, the notions of a set of objects in Ψ and of a class of objects in Ψ. We will say then that $u \in U$ is a *set of objects in* Ψ, *u set Ψ* for short, when

(S1) for any $w \neq_\mu n$ such that $w\ ingr(part(\mu)\ )\ u$ there exist $v \neq_\mu n$ and $t \in \Psi$ such that $v\ ingr(part(\mu)\ )\ w$, $v\ ingr(part(\mu))\ t$, $t\ ingr(part(\mu)\ )\ u$;

if in addition, we have

(S2) $t\ ingr(part(\mu)\ )\ u$ for any $t \in \Psi$;
(S3) for any t, if t satisfies (S1) and (S2) with Ψ then $u\ ingr(part(\mu)\ )\ t$

then we say that u is a *class of objects in* Ψ, *u class Ψ* for short. It follows from (A6) that for any collection Ψ there exists a unique object u such that *u class Ψ* and there exists objects of the form *set Ψ*. We have therefore

Proposition 3. The pair $(\ U - \{n\}, part(\mu) \restriction (U - \{n\}) \times (U - \{n\})$ is a model of mereology.

□

We apply the above theoretical scheme to the task of formalization of design, synthesis and control of complex systems on the basis of knowledge learned from data tables.

Approximate logic of design and synthesis

Design agents. Requirements. The designer operates on the set Des_Ag of design agents; any design agent dag is equipped with an information system $\mathbf{A}_{dag} = (U_{dag}, A_{dag})$ and a rough inclusion μ_{dag}. The table $\mathbf{A}_{dag}$ describes objects (possibly complex) from a universe U_{dag} in the language of attributes in A_{dag}. The variable b_{dag} runs over objects in U_{dag}. A valuation v_X where X is a set of design agents is a function which assigns to any b_{dag} for $dag \in X$ an element $v_X(b_{dag}) \in U_{dag}$. A *(designer) requirement* $\Psi(ag)$ at ag is a formula in the logic $C(A_{dag}, V)$ of conditional attributes (Skowron 1995); the symbol Ψ will denote a requirement at some design agent. The symbol $x \models_d \Psi_d$ will denote that x satisfies Ψ.

Synthesis agents. Any synthesis agent ag is assigned a *label*

$$lab(ag) = \{U(ag), \pi(ag, dag), D(ag), St(ag), L(ag), \mu_o(ag), F(ag)\}$$

where

$U(ag)$ is the universe of objects at ag;

for any synthesis agent ag there exists a design agent dag and a mapping $\pi(ag, dag) : U(ag) \longmapsto U_{dag}$;

$D(ag) = \{U(ag), A(ag), d(ag)\}$ is the decision system (Pawlak 1991) of ag; $A(ag)$ is the set of conditional attributes of ag. The value $d(ag)(x)$ is a designer requirement Ψ satisfied by $\pi(ag, dag)(x)$;

$St(ag) \subseteq U(ag)$ is the set of standard objects (standards) at ag;

$L(ag)$ is a set of unary predicates at ag (specifying properties of objects in $U(ag)$). Predicates of $L(ag)$ are constructed as formulas in $C(A(ag), V)$;

$\mu_o(ag) \subseteq U(ag) \times U(ag) \times [0,1]$ is a pre-rough inclusion at ag (Polkowski & Skowron 1996a); usually it is defined from the information system of the agent ag by the formula

$$\mu^o(x,y) = \frac{cardinality\{a \in A(ag) : a(x) \neq a(y)\}}{cardinality A(ag)};$$

$F(ag)$ is a set of *mereological connectives at ag* (Polkowski & Skowron 1995), (Skowron & Polkowski 1995), (Polkowski & Skowron 1996b).

Mereology μ_{DS}. Synthesis agents classify their objects by means of pre - rough inclusions; any pre - rough inclusion $\mu_o(ag)$ can be extended (Polkowski & Skowron 1996a) to a rough inclusion $\mu(ag)$ on the set $2^{U(ag)}$ of subsets of $U(ag)$. The $D_S - mereology$ μ_{DS} (the *design - synthesis mereology*) is a family $\{\mu(ag) : ag \in Ag\}$ where $\mu(ag)$ is a fixed extension of $\mu_o(ag)$ for any $ag \in Ag$. Mereology μ_{DS} defines design objects (categories of real objects): a *design object* x is a $class(\mu(ag))\Gamma$ where $\Gamma \subseteq U(ag)$. Thus, *design objects* are *mereological classes of synthesis objects.* On these classes the mereologies of design agents act decomposing them into simpler classes.

The communication between synthesis spaces and design spaces is provided by mappings $\pi(ag, dag)$: for $x \in U(ag)$, the value $\pi(ag, dag)(x) \in U(dag)$ is a design object . Let us observe that while $\pi(ag, dag)(x)$ is unique, the object x may belong to more than one design object. This causes the ambiguity in communication among design agents and synthesis agents: while a synthesis agent ag classifies x as $cat(x) = \pi(ag, dag)(x)$, the design agent dag can regard x as an element of a category $cat'(x)$ distinct from $\pi(ag, dag)(x)$. However, categories $cat(x)$, $cat'(x)$ containing x should be regarded as similar. We express this similarity on the higher level of requirements by means of a tolerance relation Des_sat.

Satisfactory satisfiability of requirements. The requirements of the designer specify classes of ideal objects but the ultimate purpose of design is a real object whose category would satisfy Ψ. It can happen that an object whose category satisfies a requirement $\Psi' \neq \Psi$ is accepted as satisfying Ψ in a degree satisfactory to the designer. We denote by Des_req the set of designer requirements; the vagueness of designer requirements will be formalized by means of a tolerance relation des_sat on the set Des_req; for $\Psi, \Psi' \in Des_req$, Ψ des_sat Ψ' will read "any object satisfying Ψ' (resp. Ψ) satisfies Ψ (resp. Ψ') in degree satisfactory to the designer". We denote by the symbol $[\Psi]$ the tolerance class $des_sat(\Psi)$. We denote by the symbol $\Psi^\vee$ the disjunction of formulas in $[\Psi]$ i.e. $\Psi^\vee$ is $\vee\{\Psi' : \Psi' \in [\Psi]\}$. Clearly, if $x \models_d \Psi^\vee$ then x satisfies Ψ in the satisfactory degree.

Mereological compatibility of requirements. We assume that μ_{dag} is compatible with des_sat i.e. **if** $x = class(\mu_{dag})\ \{x_1, x_2, ..., x_k\}$, $x_i \models_d \Psi_i$, $x \models_d \Psi$, $y_i \models_d \Psi_i^\vee$, $y = class(\mu_{dag})\ \{y_1, y_2, .., y_k\}$ **then** $y \models_d \Psi^\vee$. The compatibility condition means that the designer schemes are designed as insensitive to local communication ambiguities.

Approximate logic of synthesis. The symbol b_{ag} will denote the variable which runs over objects in U_{ag}. A valuation v_X where X is a set of synthesis agents is a function which assigns to any b_{ag} for $ag \in X$ an element $v_X(b_{ag}) \in U_{ag}$. The symbol v_{ag}^x denotes $v_{\{ag\}}$ with $v_{\{ag\}}(b_{ag}) = x$.

We now define syntax of a logic of approximate formulas L (Komorowski, Polkowski, & Skowron 1996). The *atomic formulas* of L are of the form $< st(ag), \Phi(ag), \varepsilon(ag) >$ where $st(ag) \in St(ag), \Phi(ag) \in L(ag), \varepsilon(ag) \in [0,1]$ for $ag \in Ag$. The *formulas* of L are built from atomic formulas by means of classical propositional connectives $\neg, \vee, \wedge, \Rightarrow, \Leftrightarrow$ and of modal unary propositional connective $\Diamond$ (cf. Sect.4 for semantics of $\Diamond$). The *semantics* of L is defined as follows. For $ag = Root(C)$, $v \in V_C$, we say that v satisfies a formula $\alpha =< st(ag), \Phi(ag), \varepsilon(ag) >$, symbolically $v \models \alpha$, in case $E_{\mu(ag)}(v(b_{ag}), st(ag)) \geq \varepsilon$ and $v(b_{ag}) \in \Phi(ag)_{D(ag)}$ (see Appendix). We let $v \models \alpha \wedge \beta$ in case $v \models \alpha$ and $v \models \beta$; $v \models \neg\alpha$ in case $non(v \models \alpha)$. For a formula $< st(ag), \Phi(ag), \varepsilon(ag) >$ of L, we write $x \models < st(ag), \Phi(ag), \varepsilon(ag) >$ iff $v_{ag}^x \models < st(ag), \Phi(ag), \varepsilon(ag) >$.

Decision rules of synthesis agents. From the triple $(D(ag), L(ag), \mu(ag))$, the agent ag generates by the standard techniques (Skowron 1995) the decision rules of the form

$$< st(ag), \Phi(ag), \varepsilon(ag) > \Longrightarrow \Psi^\vee.$$

The meaning of this rule is:

$$(< st(ag), \Phi(ag), \varepsilon(ag) > \Longrightarrow \Psi^\vee) \text{ is } true$$
$$\text{iff}$$
$$(\pi(ag, dag)(st(ag)) \models_d \Psi) \wedge [\, x \models$$
$$< st(ag), \Phi(ag), \varepsilon(ag) > \Rightarrow \pi(ag, dag)(x) \models_d \Psi^\vee].$$

Informally, this means that the design class of the standard $st(ag)$ satisfies the requirement Ψ and design

classes of objects satisfactory close to $st(ag)$ (closeness measured by $\varepsilon(ag)$ satisfy $\Psi^\vee$ i.e. they satisfy Ψ in a satisfactory degree.

Synthesis schemes

Local decomposition schemes. By means of the rough inclusion μ_{dag} the agent dag induces on the universe U_{dag} the mereological relation $part_{dag}$ in the sense of Leśniewski (Leśniewski 1992). The relation $part_{dag}$ establishes a *local decomposition scheme* of some complex objects in the universe U_{dag} into simpler objects i.e. the relation $x\ class(\mu_{dag})\ \{x_1, x_2, .., x_k\}$ means that x is designed (built) from parts $x_1, x_2, .., x_k$. In this way the designer establishes a hierarchy $\{part_{dag}\}$ of local decomposition schemes on the set of possible complex objects. Local decomposition schemes can be composed in the sense that whenever the relation $part_{dag}$ expresses an object x as built from parts $x_1, x_2, .., x_k$ and the relation $part_{dag'}$ expresses, say, x_1 as built from parts $x_{11}, x_{12}, .., x_{1m}$ then the composition $part_{dag} \circ part_{dag'}$ expresses x as built from parts $x_{11}, x_{12}, ..., x_{1m}, x_2, .., x_k$.

The designer language $Link_d$. The designer task is now to establish for a given complex object x, specified by a requirement Ψ, a *scheme of design agents* (for simplicity we assume this scheme to be a tree) such that the object x can be decomposed over the scheme by means of a composition of local decomposition schemes into primitive objects which belong to the universes of leaf agents of the scheme. We define a language $Link_d \subseteq Des_Ag^+$ where Des_Ag^+ is the set of all finite non-empty strings over Des_Ag. For a string $\mathbf{dag} = dag_1 dag_2 ... dag_k dag$, where dag is the *root agent* and $dag_1, dag_2, .., dag_k$ are *leaf agents*, we have $\mathbf{dag} = dag_1 dag_2 ... dag_k dag \in Link_d$ iff there exist $x, x_1, x_2, .., x_k$, x such that $x\ class(part_{dag})\ \{x_1, x_2, .., x_k\}$ where $x_i \in U_{dag_i}$ for $i \leq k$, $x \in U_{dag}$. We let $set(\mathbf{dag}) = \{dag_1, dag_2 ..., dag_k, dag\}$. For $L \subseteq Link_d$, we define $Des_Ag(L) = \cup\{set(\mathbf{dag}) : \mathbf{dag} \in L\}$ and we denote by $\leq$ a relation on $Des_Ag(L)$ defined by: $dag \leq dag'$ if and only if there exists $\mathbf{dag} \in L$ such that $dag, dag' \in set(\mathbf{dag})$ and dag' is the **dag**-target. A set $L \subseteq Link_d$ is *a construction support* in case $(Des_Ag(L), \leq)$ is a tree. From now on L will denote a construction support.

Constructibility mapping and design operations. For $\mathbf{dag} = dag_1 dag_2 . .. dag_k dag \in L$, we define the **dag**-*constructibility relation* $\rho(\mathbf{dag}) \subseteq U(dag_1) \times U(dag_2) \times ... \times U(dag_k) \times U(dag)$ by letting $(x_1, x_2, ..., x_k, x) \in \rho(\mathbf{dag})$ iff $x\ class(part_{dag})\{x_1, x_2, .., x_k\}$.

The *constructibility mapping* $con(\mathbf{dag})$, associated with the relation $\rho(\mathbf{dag})$, will be defined by $con(\mathbf{dag})(x_1, x_2, ..., x_k) = x$ iff $(x_1, x_2, ..., x_k, x) \in \rho(\mathbf{dag})$.

The existence of the mapping $con(\mathbf{dag})$ follows from the uniqueness of classes in mereology generated by the relation $part_{dag}$. The constructibility mapping $con(\mathbf{dag})$ is in general a many - to -one mapping. We select from $con(\mathbf{dag})$ one -to -one mappings called design operations defined as follows: a *design operation* $o(\mathbf{dag})$ associated with **dag** is a one - to -one partial mapping such that if $o(\mathbf{dag})(x_1, x_2, .., x_k) = x$ then $con(\mathbf{dag})(x_1, x_2, .., x_k) = x$ and $range\ o(\mathbf{dag}) = range\ con(\mathbf{dag})$.

We denote by the symbol $O(\mathbf{dag})$ the collection of design operations associated with **dag**. We will write $o(dag)$ instead of $o(\mathbf{dag})$, where $dag = root(\mathbf{dag})$.

We observe that the relation $\rho(\mathbf{dag})$, the mapping $con(\mathbf{dag})$ and design operations $o(\mathbf{dag})$ are compatible with requirements with respect to the tolerance des_sat viz. the condition

$(x_1, x_2, ..., x_k, x) \in \rho(\mathbf{dag})$, $x_i \models_d \Psi_i$, $x \models_d \Psi$, $y_i \models_d \Psi_i^\vee$ and $(y_1, y_2, ..., y_k, y) \in \rho(\mathbf{dag})$

implies $y \models_d \Psi^\vee$ and similar conditions hold for $con(\mathbf{dag})$ and $o(\mathbf{dag})$.

Design schemes. Now, given a requirement Ψ (a formula in the conditional logic of the designer), the designer initiates the communication and negotiation process among design agents. This process results eventually in a design scheme which is able to design an object satisfying Ψ in the satisfactory degree. We describe this initializing process in the following steps.

Step 1. The designer selects a design agent dag_o such that there exists an object $x_o \in U_{dag_o}$ which satisfies the requirement $\Psi^\vee$ and there exists $\mathbf{dag}_o = dag_{o,1} dag_{o,2} ... dag_{o,k} dag_o \in Link_d$ such that the condition

$$con(\mathbf{dag}_o)(x_1, x_2, ..., x_k) = x$$

holds for some vector argument $(x_1, x_2, ..., x_k)$.

Step 2. Agents $dag_o, dag_{o,1}, dag_{o,2}, .., dag_{o,k}$ negotiate the choice of requirements $\Psi^\vee_{o,1}, \Psi^\vee_{o,2}, . .., \Psi^\vee_{o,k}$ and a vector argument $(x_1, x_2, ..., x_k)$ such that any x_i satisfies the requirement $\Psi^\vee_{o,i}$; this choice determines the design operation $o(\mathbf{dag}_o)$ with the property that $o(\mathbf{dag}_o)(x_1, x_2, .., x_k) = x$. The negotiated condition is the following: if $(y_1, y_2, .., y_k) \in domain\ o(\mathbf{dag}_o)$ and any y_i satisfies the requirement $\Psi^\vee_{o,i}$ then $o(\mathbf{dag}_o)(y_1, y_2, .., y_k)$ satisfies the requirement $\Psi^\vee$.

Factors involved in negotiations and influencing them may be e.g. the complexity of $o(\mathbf{dag})$, the cost of assembling via $o(\mathbf{dag})$, the accessibility of $x_1, .., x_k$ etc.

Next, the designer repeats steps 1 and 2 with the agents $dag_{o,1}, dag_{o,2}, .., dag_{o,k}$ and the respective requirements $\Psi^\vee_{o,1}, ..., \Psi^\vee_{o,k}$.

The negotiation process continues with new requirements. The process stops when all branches terminate with strings in $Link_d$ whose all leaf agents can satisfy the chosen requirements with inventory objects.

We observe that the result of negotiations for any non-inventory agent dag in L can be described by means of the pair $lab(dag) = (\Psi^\vee(dag), o(dag))$; the singleton $lab(dag) = (\Psi^\vee(dag))$ summarizes the negotiation outcome for any inventory agent dag in L. We will call the tuple $lab(dag)$ the *design label* of the agent dag. The construction support L along with the set $\{lab(dag)\}$ of design labels of its agents will be called the *design scheme* and denoted by $D_s = (L, \{lab(dag) : dag \in L\})$. The agent dag_o will be called the *root agent* of D_s; $dag(1), .., dag(m)$ are leaf agents of D_s. We write $D_s(\mathbf{dag}_o, \mathbf{dag}_1, .., \mathbf{dag}_m)$ when $\mathbf{dag}_o, \mathbf{dag}_1, .., \mathbf{dag}_m$ are all strings used to construct D_s. $D_s(\mathbf{dag}_i)$ is called *an elementary design scheme.*

The *operation term* of D_s, denoted by the symbol $T(D_s)$ will be defined by induction on the number of strings from $Link$ in L:

(i) **if** $L = \{\mathbf{dag}_o\}$ **then** we let $T(D_s) = o(dag_o)(b_{dag_{o,1}}, b_{dag_{o,2}}, ..., b_{dag_{o,k}})$;

(ii) **if** $L = \{\mathbf{dag}_o, \mathbf{dag}_1, ..., \mathbf{dag}_k\}$,
an agent dag is a leaf agent of D_s,
$\mathbf{dag}_{k+1} = dag_1 dag_2 ... dag_n dag$,
$T(D_s) = o(dag_o)(...(...(..., b_{dag}, ...)...)$
then D_s' results from D_s by attaching $\mathbf{ag}_{k+1}$ to D_s at dag i.e.

$$T(D_s)' = o(dag_o)(...(...(..., o(dag)(b_{dag_1}, b_{dag_2}, ..., b_{dag_n}), ...)...).$$

The term $T(D_s)$ represents the composition of operations along the construction support L. This composition is a global operation which assigns to argument vectors of objects at leaf agents of D_s the value which is an object in the universe of the root agent of D_s.

The following proposition resumes the upshot of the negotiation process.

Proposition 4. For any valuation v_X on the set X of leaf agents $dag(1), .., dag(m)$ of the design scheme D_s where $v(b_{dag(i)})$ satisfies $\Psi^\vee(dag(i))$ and $lab(ag))$ $= (\Psi^\vee(dag_i))$ for $i = 1, 2, ..., m$, the uniquely defined object

$$T(D_s)(v(b_{dag(1)}), v(b_{dag(2)}), ..., v(b_{dag(k)})) \in U_{dag_o} \text{ satisfies } \Psi^\vee(dag_o)$$

where $lab(dag_o) = (\Psi^\vee(dag_o), o(dag_o))$.

□

The projection

$$Dg = (\Psi^\vee(ag_1), \Psi^\vee(ag_2), .., \Psi^\vee(ag_k), \Psi^\vee(ag_o))$$

of the design scheme D_s onto the requirement space will be called a *designer goal*; the designer goal Dg expresses the existence of a design scheme which can design an object satisfying $\Psi^\vee(ag_o)$ from objects satisfying $\Psi^\vee(ag_i), i = 1, 2, .., k$, respectively.

In the synthesis stage the synthesis agents are organized into a synthesis scheme modelled on a given design scheme.

Synthesis (of approximate reasoning scheme)

We begin with a set Ag of *synthesis agents* (called simply *agents* in what follows) and a set I of primitive objects (the *inventory*). Elements of $lab(ag)$ constitute the basic knowledge of a synthesis agent ag; other elements are worked out in negotiations among synthesis agents and in their interactions with the design agents. We now describe this process.

The synthesis language $Link$. We recall that the designer creates a language $Link_d \subseteq Des_Ag^+$; the language $Link_d$ is a pattern according to which a language $Link \subseteq Ag^+$ is defined in the process of communication among synthesis agents. To this end, the agents $ag_1, ag_2, .., ag_k, ag_o$ form the string $\mathbf{ag} = ag_1 ag_2 .. ag_k ag_o \in Link$ if and only if there exist design agents $dag_1, dag_2, ..., dag_k$, dag_o such that

$$\mathbf{dag} = dag_1 dag_2 ... dag_k dag_o \in Link_d,$$

and $\pi(ag_i, dag_i)$ is defined for each i,
and there exist objects $x_1 \in U(ag_1), ..., x_k \in U(ag_k)$, $x \in U(ag_o)$
such that

$$(\pi(ag_1, dag_1)(x_1), .., \pi(ag_k, dag_k)(x_k), \pi(ag_o, dag_o)(x)) \in \rho(\mathbf{dag}).$$

For $\mathbf{ag} = ag_1 ag_2 ... ag_k ag_o \in Link$, an *operation* $o(\mathbf{ag})$ is a partial one-to-one mapping with *domain* $o(\mathbf{ag}) \subseteq U(ag_1) \times U(ag_2) \times ... \times U(ag_k)$ and *range* $o(\mathbf{ag}) \subseteq U(ag_o)$ such that
if $o(ag_o)(x_1, x_2, .., x_k) = x$ **then**

$$o_d(\mathbf{dag})(\pi(ag_1, dag_1)(x_1), .., \pi(ag_k, dag_k)(x_k)) = \pi(ag_o, dag_o)(x)$$

for some unique design operation $o_d(\mathbf{dag})$; let us observe that to a design operation $o_d(\mathbf{dag})$ there may correspond more than one operation $o(\mathbf{ag})$. We let $\mathbf{dag} = \Pi(\mathbf{ag})$ and $o_d(\mathbf{dag}) = \Pi(o(\mathbf{ag}))$. The operation Π extends in the natural way to compositions of operations.

Elementary constructions. We define *an elementary construction* c: if $\mathbf{ag} = ag_1 ag_2 .. ag_k ag_o \in Link$, then an expression

$$c = (\mathbf{ag}, p_sign(ag_1), p_sign(ag_2), . ., p_sign(ag_k), p_sign(ag_o))$$

will be called an *elementary construction* with *presignatures*

$$p_sign(ag_o) = (st(ag_o), \Phi(ag_o), \varepsilon(ag_o), o(ag_o))$$

and $p_sign(ag_i) = (st(ag_i), \Phi(ag_i), \varepsilon(ag_i))$ if there exists a $D_s(\mathbf{dag})$ with
$\mathbf{dag} = dag_1 dag_2 ... dag_k dag_o = \Pi(\mathbf{ag})$,
$o_d(dag_o) = \Pi(o(\mathbf{ag}))$,
$lab(dag_o) = (\Psi^\vee(dag_o), o_d(dag_o))$,
$lab(dag_i) = (\Psi^\vee(dag_i))$

such that $< st(ag_i), \Phi(ag_i), \varepsilon(ag_i) > \Longrightarrow \Psi^{\vee}(ag_i)$ is true for $i = 0, 1, 2, .., k$.

We will say that c *projects onto* D_s. To stress the relationship between $\mathbf{ag}$ and c we will write $c(\mathbf{ag})$ instead of c and $\mathbf{ag}(c)$ instead of $\mathbf{ag}$. We let $ag_o = Root(c), \{ag_1, ag_2, .., ag_k\} = Leaf(c)$, $\{ag_1, ag_2, .., ag_k, ag\} = Ag(c)$.

Constructions. For elementary constructions c, c' with $Ag(c) \cap Ag(c') = \{ag\}$ where $ag = Root(c) \in Leaf(c')$, we define *the ag-composition* $c \star_{ag} c'$ of c and c' with $Root(c \star_{ag} c') = Root(c')$, $Leaf(c \star_{ag} c') = (Leaf(c) - \{ag\}) \cup (Leaf(c'), Ag(c \star_{ag} c') = Ag(c) \cup Ag(c')$. The composition $c \star_{ag} c'$ is called a *construction* if there exists a $D_s(\mathbf{dag}_o, \mathbf{dag}_1)$ such that c projects onto $\mathbf{dag}_o$ and c' projects onto $\mathbf{dag}_1$; we say then that $c \star_{ag} c'$ *projects onto* $D_s(\mathbf{dag}_o, \mathbf{dag}_1)$. A *construction* is any expression C obtained from a set of elementary constructions by applying the composition operation a finite number of times. By V_C we denote the set of partial valuations $v_{Ag(C)}$. By $T(C)$ we denote the unique term composed of operations $o(ag)$ such that $\Pi(T(C)) = T(D_s)$ where C projects onto D_s.

(C, Φ, ε)–schemes. For an elementary construction $c = c(\mathbf{ag})$ as above, with $p_sign(ag) = (st(ag), \Phi(ag), \varepsilon(ag), o(ag))$, we define a $(c, \Phi, \varepsilon) - scheme$ as a pair $(c(\mathbf{ag}), sign(ag))$ where $sign(ag) = p_sign(ag) \cup \{f(ag)\}, \Phi = \Phi(ag), \varepsilon = \varepsilon(ag)$ and $f(ag) \in F(ag)$ satisfies the condition:

(Unc(c))**if** $\mu_o(ag_i)(x_i, st(ag_i)) \geq \varepsilon(ag_i)$ for $i = 1, 2, .., k$
then $\mu_o(ag)(o(ag)(x_1, x_2, .., x_k), st(ag)) \geq f(\varepsilon(ag_1), \varepsilon(ag_2), .., \varepsilon(ag_k)) \geq \varepsilon(ag)$.

The construction c is said to be the *support* of the $(c, \Phi, \varepsilon) - scheme$.

A construction C composed of elementary constructions $c_1, .., c_m, c_o$ with $Root(C) = Root(c_o) = ag_o$ is the support of a $(C, \Phi, \varepsilon) - scheme$ when each c_i is the support of a $(c_i, \Phi_i, \varepsilon_i) - scheme$, where $\Phi_i = \Phi(Root(c_i))$, $\varepsilon_i = \varepsilon(Root(c_i))$, $\Phi = \Phi(ag_o)$ and $\varepsilon = \varepsilon(ag_o)$. For valuations $v = v_{Leaf(C)}$, $v' = v_{Root(C)} \in V_C$, we write $v \longrightarrow_C v'$ iff $T(C)(v) = v'$. We have a proposition.

Proposition 5. *Approximate synthesis theorem* For any valuation v_X on the set X of leaf agents $ag(1), .., ag(m)$ of the (C, Φ, ε)–scheme with $ag_o = Root(C)$ such that

$$v(b_{ag(i)}) \text{ satisfies } < st(ag(i)), \Phi(ag(i)), \varepsilon(ag(i)) > \text{ for } i = 1, 2, ..., m$$

where $p_sign(ag(i)) = (st(ag(i)), \Phi(ag(i)), \varepsilon(ag(i))$, the uniquely defined object $x \in U(ag_o)$ such that $v_X \longrightarrow_C v^x_{ag_o}$ satisfies

$$< st(ag_o), \Phi(ag_o), \varepsilon(ag_o) >$$

where $p_sign(ag_o) = (st(ag_o), \Phi(ag_o), \varepsilon(ag_o), o(ag_o))$.

□

Projections of (C, Φ, ε)-schemes onto design schemes. We say that a $(C, \Phi, \varepsilon) - scheme$ *projects onto a design scheme* D_s when the support C projects onto D_s.

Negotiation (top-down) of a scheme for satisfying a requirement. Consider now a designer goal $Dg = (\Psi^{\vee}(dag_1), \Psi^{\vee}(dag_2), ..., \Psi^{\vee}(dag_k), \Psi^{\vee}(dag_o))$ realized by a design scheme $D_s(\mathbf{dag}_o, \mathbf{dag}_1, ..., \mathbf{dag}_m)$. The realization by synthesis agents of the designer goal Dg must begin with finding a (C, Φ, ε)-scheme S such that S projects onto D_s. We describe this process.

Stage 1. A string $\mathbf{ag}_o = ag_1 ag_2 .. ag_k ag_o \in Ag$ is chosen such that a construction $c(\mathbf{ag}_o)$ projects onto $D_s(\mathbf{dag}_o)$; let the uncertainty coefficients of agents be $\varepsilon'(ag_1), ..., \varepsilon'(ag_k), \varepsilon'(ag_o)$.

Stage 2. Agents $ag_1, ag_2, .., ag_k, ag_o$ negotiate a connective $f(ag_o) \in F(ag_o)$ and uncertainty bounds

$$\varepsilon(ag_1) \geq \varepsilon'(ag_1), .., \varepsilon(ag_k) \geq \varepsilon'(ag_k) .., \varepsilon(ag_o) \geq \varepsilon'(ag_o)$$

such that $(Unc(ag_o))$ is satisfied with $f(ag_o)$ and $\varepsilon(ag_1), .., \varepsilon(ag_k), \varepsilon(ag_o)$.

Stages 1, 2 give, when successful, a $(c(ag_o), \Phi, \varepsilon)$ - scheme which projects onto $D_s(\mathbf{dag}_o)$.

Stages 3 and following. Agents $ag_1, ag_2, .., ag_k$ repeat stages 1,2 with new coefficients $\varepsilon(ag_i)$, and so on.

The successful result of negotiations means that a $(C, \Phi(ag_o), \varepsilon(ag_o))$ - scheme is constructed which projects onto $D_s(\mathbf{dag}_o, \mathbf{dag}_1, ..., \mathbf{dag}_m)$. We state a theorem which follows from Propositions 4 and 5.

Theorem 6. *(the sufficiency criterium of correctness)*

Assume that a (C, Φ, ε)-scheme projects onto a design scheme D_s realizing a designer goal $Dg = (\Psi^{\vee}(dag_1), \Psi^{\vee}(dag_2), .., \Psi^{\vee}(dag_k), \Psi^{\vee}(dag_o))$. Then for any valuation v_X on the set X of leaf agents $ag(1), .., ag(m)$ of the (C, Φ, ε)–scheme with

$$ag_o = Root(C)$$

such that $v(b_{ag(i)})$ satisfies

$$< st(ag(i)), \Phi(ag(i)), \varepsilon(ag(i)) >$$

where $p_sign(ag(i)) =$

$$(st(ag(i)), \Phi(ag(i)), \varepsilon(ag(i), o(ag_o))$$

for $i = 1, 2, ..., m$,
the uniquely defined object $x \in U(ag_o)$ such that $v_X \longrightarrow_C v^x_{ag_o}$ satisfies the condition

$$\pi(ag_o, dag_o)(x) \models_d \Psi^{\vee}(dag_o)$$

i.e. x satisfies the designer requirement Ψ in the satisfactory degree.

□

We extend the satisfiability relation $\models$ to the connective $\diamond$ by $v \models \diamond < st(ag), \Phi(ag), \varepsilon(ag) >$ if there exists C such that

$$v_{\{Root(C)\}} \models < st(ag), \Phi(ag), \varepsilon(ag) >$$

where $v \rightarrow_C v_{\{Root(C)\}}$.

The connective $\diamond$ expresses the existence of a scheme which can satisfy (Φ, ε) over a given input v. In particular, Theorem 6 gives a sufficient condition for finding such C.

Let us emphasize the fact that the functions $f(ag)$, called mereological connectives above, are expected to be extracted from experiments with samples of objects. The above property allows for an easy to justify correctness criterium of a given (C, Φ, ε)-scheme provided that all parameters in this scheme have been chosen properly. The searching process for these parameters and synthesis of an uncertainty propagation scheme satisfying the formulated conditions constitutes the main and not easy part of design and synthesis.

Bottom-up communication. The bottom-up communication process is started by the leaf agents of C; the leaf agents select a valuation v_X compatible with C and any leaf agent ag sends to its parent agent ag_o the object $v_X(b_{ag})$ and the value $E_{\mu(ag)}(v_X(b_{ag}), st(ag))$. This process is repeated with non-leaf agents: any non-leaf agent ag applies the operation $o(ag)$ to the objects $x_1, x_2, ..., x_k$ sent by its children agents, synthesizes the object $x = o(ag)(x_1, x_2, ..., x_k)$ and finds the value $E_{\mu(ag)}(x, st(ag))$. This communication process ends at the root agent of C with the object x_C. In the case when the assembling process proceeds correctly, the object x_C satisfies the requirement Ψ in the satisfactory degree; the correctness of the assembling process is checked by means of the negotiated connectives $f(ag)$ and the found values $E_{\mu(ag)}(x, st(ag))$.

Learning

In synthesis stage discussed above, the crucial factor is presented by mereological connectives (*rmc*) of uncertainty rules. In practical cases, we can infer from the given data tables an approximation f to the appropriate *rmc* F. We include an example which illustrates our approach.

Example 1. *Learning mereological connectives from data tables.* We present a tiny fragment of a distributed system consisting of agents B, L, M. We may imagine thet B, L, M form a part of the assembly line for assembling lego-like toys. The agent B submits bodies and L submits a leg (right) which M fits together to be sent further up. The information system of the agent B is given in Table 1.

	a	b	c	d	e	f	g	h
B1	1	1	1	1	1	1	1	0
B2	1	0	1	0	0	1	0	0
B3	0	0	0	1	0	0	0	0
B4	1	1	1	0	0	1	0	0

where a, b, c, d, e, f, g, h stand for pis, cut, bel, kni, par, co, crl, crr, respectively (these boolean attributes represent here, respectively presence of pistols, a cutlass, a belt, a knife, a parot, a coat, a crutch on left, a crutch on right).

Table 1

Information system of agent L is given in Table 2.

	a	b	c	d	e	f	g	h	i	j
L1	1	1	1	1	1	1	1	1	0	1
L2	1	1	1	1	1	1	1	1	1	0
L3	1	0	1	1	0	0	0	0	1	1

where a, b, c, d, e, f, g, h, i, j stand for fobj, fpa, fsof, mesl, wou, kck, fcol, fwet, frsl, fisl, respectively. The attributes discerning a wooden artificial leg from a "natural" one are like fobj (*feel objects*), fpa (*feel pain*), frsl (*fit the right side*), fisl (*fit the left side*).

Table 2

Information system of agent M is given in Table 3.

	har	mar	lar	crl	crr	par	co
B1L2	1	0	0	1	0	1	1
B2L2	0	1	0	0	0	1	1
B2L3	0	1	0	0	1	1	1
B3L2	0	0	1	0	0	0	0
B3L3	0	0	1	0	1	0	0
B4L2	1	0	0	0	0	0	1
B4L3	1	0	0	0	1	0	1

Attributes here describe the complex object obtained by attaching a leg submitted by L to a body submitted by B. New attributes: har, mar lar, mean rspectively *heavily (medium, lightly) armed.*

Table 3

In the three following tables we present values of similarity functions based on initial rough inclusion

$$\mu^o(x, y) = \frac{cardinality\{a \in A : a(x) \neq a(y)\}}{cardinality(A)}$$

where A is the set of all attributes.

Values of the function μ^0_B of objects at B are given in Table 4.

	B1	B2	B3	B4
B1	1	0.5	0.25	0.62
B2	0.5	1	0.5	0.87
B3	0.25	0.5	1	0.37
B4	0.62	0.87	0.37	1

Table 4

Values of the function μ^0_L at agent L are collected in Table 5.

	L1	L2	L3
L1	1	0.8	0.4
L2	0.8	1	0.4
L3	0.4	0.4	1

Table 5

In Table 6, we collect values of μ^0_M at the agent M.

	a	b	c	d	e	f	g
a	1	0.57	0.43	0.28	0.14	0.71	0.57
b	0.57	1	0.86	0.43	0.14	0.57	0.43
c	0.43	0.86	1	0.28	0.43	0.43	0.57
d	0.28	0.43	0.28	1	0.86	0.57	0.43
e	0.14	0.86	0.43	0.86	1	0.43	0.57
f	0.71	0.57	0.43	0.57	0.43	1	0.86
g	0.57	0.43	0.57	0.43	0.57	0.86	1

where a, b, c, d, e, f, g stand for B1L2, B2L2, B2L3, B3L2, B3L3, B4L2, B4L3, respectively.

Table 6

Now, the agent M synthesizes complex objects of the form XY from objects X sent by B and objects Y sent by L. The mereological connective F proper for this case should satisfy the condition : if $\mu^0_B(X', X) \geq \varepsilon_1$ and $\mu^0_L(Y', Y) \geq \varepsilon_2$ then $\mu^0_M(X'Y', XY) \geq F(\varepsilon_1, \varepsilon_2)$. However, we cannot know F exactly as we do not know all possible objects at agents (at least, in principle, we cannot state so). What we may learn from the data is then a local approximation to F; it turns out in practice that F can be highly inhomogeneous and may merely be a relation which however can be locally approximated by functions. We say therefore that a function f is *an approximation of F at the object X^0Y^0* when the condition holds: if $\mu^0_B(X', X^0) \geq \varepsilon_1$ and $\mu^0_L(Y', Y^0) \geq e_2$ then $\mu^0_M(X'Y', X^0Y^0) \geq f(\varepsilon_1, \varepsilon_2)$. It is not necessary to learn the whole function f: we call a set T of vectors of the form $[\varepsilon_1, \varepsilon_2, \varepsilon]$ the *threshold set of vectors for f at X^0Y^0* whenever T is a minimal set of vectors such that: $f(\varepsilon_1, \varepsilon_2) \geq \varepsilon$ for any $[\varepsilon_1, \varepsilon_2, \varepsilon] \in T$ implies that f is an approximation to F at X^0Y^0.

The following table shows a threshold vector set for an approximation at B1L2.

		ε_1	ε_2	ε
	1	0.25	1	0.28
Table 7	2	0.25	0.4	0.14
	3	0.62	1	0.71
	4	0.62	0.4	0.57

Imagine now that M receives a requirement (in informal, quasi-natural language) for instance: Ψ: *"an artifact is needed which will serve for a toy model of an heavily armed invalid pirate"*. Taking $B1L2$ as a standard, M transforms Ψ into the approximate formula: $(B1L2, (a = 1) \wedge (b = 1) \wedge (c = 1) \wedge (d = 1) \wedge (e = 1) \wedge (f = 1) \wedge (g = 1) \wedge (h = 0), 0.7)$. From Table 7 it follows that $B1L2$ satisfies Ψ. Thus L submits $L2$ and B submits either $B1$ or $B4$.

The learning process is concerned primarily with learning threshold sets of vectors at complex objects. The complexity of this process will be studied elsewhere.

Learning processes on higher level are concerned with learning the dynamics of synthesis schemes. This leads to knowledge concerning the necessity of changing the topology of the scheme i.e. links among local teams or changing the goals of local teams.

Applications: Learning rules of a mereological controller

The approximate specification (Φ, ε) can be regarded as the invariant to be kept over the universe of global states (complex objects) of the distributed system.

The basic problems. Now we can formulate some basic problems related to control in terms of properties of a construction C. The control problems can be divided into several classes depending on the model of controlled object. In this work we deal with the simplest case. In this case, the model of a controlled object is the (C, Φ, ε) -scheme $\mathbf{c}$ which can be treated as a model of the unperturbed by noise controlled object whose states are satisfying the approximate specification (Φ, ε).

The (C, Φ, ε) -scheme defines $\mathbf{c}$ a function $F_{\mathbf{c}}$ called *the output function* of the (C, Φ, ε) -scheme $\mathbf{c}$ given by $F_{\mathbf{c}}(v) = x$ iff $v \longrightarrow_C v^x_{ag_{\mathbf{c}}}$ where $ag_{\mathbf{c}} = Root(C)$. Let $ag_1, ..., ag_r$ be leaf agents of C. Any object x from the set $F_{\mathbf{c}}(U(ag_1) \times ... \times U(ag_r)) \cap \{x \in U(ag) : x \models (st(ag_{\mathbf{c}}), \Phi, \varepsilon)\}$ is called *the (Φ, ε)-invariant object* of $\mathbf{c}$.

We assume the leaf agents of the (C, Φ, ε) -scheme $\mathbf{c}$ are partitioned into two disjoint sets, namely the set $Un_control(\mathbf{c})$ of *uncontrollable (noise) agents* and the set $Control(\mathbf{c})$ of controllable *agents*.

We present now two examples of a control problem for a given (C, Φ, ε) -scheme.

(OCP) OPTIMAL CONTROL PROBLEM:

Input: (C, Φ, ε) -scheme $\mathbf{c}$; information about actual valuation v of leaf agents i.e. the values $v(b_{ag})$ for any $ag \in Control(\mathbf{c})$ and a value ε' such that $F_{\mathbf{c}}(v) \models (st(ag_{\mathbf{c}}), \Phi, \varepsilon')$.

Output: A new valuation v' such that $v'(b_{ag}) = v'(b_{ag})$ for $ag \in Un_control(\mathbf{c})$ and $F_{\mathbf{c}}(v') \models (st(ag_{\mathbf{c}}), \Phi, \varepsilon_o)$

where

$\varepsilon_o = \sup\{\delta : F_{\mathbf{c}}(w) \models (st(ag_{\mathbf{c}}), \Phi, \delta)$
for some w such that
$w(b_{ag}) = v(b_{ag})$ for $ag \in Un_control(\mathbf{c})\}$.

These requirements on the output can be to hard to satisfy directly. One can search for changes of a given uncertainty scheme allowing to construct an object not closest to a given specification Φ but for an object satisfying a given specification in a degree higher than the sum of a given threshold and the degree defined by the current object. In this way we obtain

(CP) ∇-CONTROL PROBLEM

Input: (C, Φ, ε) -scheme $\mathbf{c}$; information about actual valuation v of leaf agents (i.e. the values $v(b_{ag})$

for any $ag \in Control(\mathbf{c})$) and a value ε' such that $F_{\mathbf{C}}(v) \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon')$.

Output: A new valuation v' such that $v'(b_{ag}) = v(b_{ag})$ for $ag \in Un_control(\mathbf{c})$ and $F_{\mathbf{C}}(v') \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon_o)$ where $\varepsilon_o > \varepsilon' + \nabla$ for some given threshold ∇.

The controller. We will describe now the basic idea on which our controllers of complex dynamic objects represented by distributed systems of intelligent agents are built. The main component of controllers are Δ-*rules* describing how local changes (i.e. changes at agents of a given distributed system) $\Delta\varepsilon(ag)$ of uncertainty coefficients $\varepsilon(ag)$ can be compensated by local changes $\Delta\varepsilon(ag_1), ..., \Delta\varepsilon(ag_k)$ of uncertainty coefficients at children $ag_1, ..., ag_k$ of ag for agents in a given scheme. By composing Δ-rules one can calculate the necessary changes of uncertainty coefficients for all agents $ag \in Control(\mathbf{c})$ and in this way to predict possible changes of controllable parameters (i.e. elementary objects being values of b_{ag} for $ag \in Control(\mathbf{c})$).

The Δ-*rules* have the following structure:

$$(\Delta\varepsilon(ag_{i_1}), ..., \Delta\varepsilon(ag_{i_r})) = h(\varepsilon(ag), -\Delta\varepsilon(ag), \varepsilon(ag_1), ..., \varepsilon(ag_k))$$

where $ag_{i_1}, ..., ag_{i_r}$ are all controllable children of ag (i.e. children of ag having descendents in $Control(\mathbf{c})$), $h : R^{k+2} \to R^r$ and R is the set of reals.

The description of the function h is extracted from experimental data. In the process of extracting h from data rough set and boolean reasoning methods (Polkowski & Skowron 1996b) can be applied.

The semantics of the Δ-rule for $\mathbf{ag} = ag_1 ag_2 \ ... \ ag_k ag_o \in Link$ in $\mathbf{c}$ is defined by uncertainty coefficients $\varepsilon(ag), \varepsilon(ag_1), ..., \varepsilon(ag_k)$ attached to agents in $\mathbf{c}$ in the following way: if an object x' issued by the agent ag is satisfying $x' \models (st(ag), \Phi(ag), \varepsilon'(ag))$ where $\varepsilon'(ag) = \varepsilon(ag) + \Delta\varepsilon(ag)$ then if the controllable children $ag_{i_1}, ..., ag_{i_r}$ of ag will issue objects $y_{i_1}, ..., y_{i_r}$ satisfying $y_{i_j} \models (st(ag_{i_j}), \Phi(ag_{i_j}), \varepsilon(ag_{i_j}) + \Delta\varepsilon(ag_{i_j}))$ for $j = 1, ..., r$ where $(\Delta\varepsilon(ag_{i_1}), ..., \Delta\varepsilon(ag_{i_r})) = h(\varepsilon(ag), -\Delta\varepsilon(ag), \varepsilon(ag_1), ..., \varepsilon(ag_k))$ then the agent ag will construct an object y such that $y \models (st(ag), \Phi(ag), \varepsilon)$ where $\varepsilon \geq \varepsilon(ag)$.

In the above formula we assume $\Delta\varepsilon(ag) \leq 0$ and $\Delta\varepsilon(ag_{i_1}) \geq 0, ..., \Delta\varepsilon(ag_{i_r}) \geq 0$. The above semantics covers the case when Δ-rules allow to compensate in one step the influence of a noise. Other cases will be treated in our next paper.

If $ag_1 ag_2 ... ag_k ag_o$, $ag'_1 ag'_2 ... ag'_r ag'_o \in Link$, $\{ag_{i_1}, ..., ag_{i_r}\}$, $\{ag_{j_1}, ..., ag_{j_s}\}$ are disjoint sets of controllable children of ag_o, ag'_o, respectively, then two Δ-rules

$$(\Delta\varepsilon(ag_{i_1}), ..., \Delta\varepsilon(ag_{i_r})) = h_1(\varepsilon(ag_o), \Delta\varepsilon(ag_o), \varepsilon(ag_1), ..., \varepsilon(ag_k))$$
$$(\Delta\varepsilon(ag_{j_1}), ..., \Delta\varepsilon(ag_{j_s})) = h_2(\varepsilon(ag'_o), \Delta\varepsilon(ag'_o), \varepsilon(ag_{j_1}), ..., \varepsilon(ag_{j_s}))$$

can be composed at ag'_o if $ag'_o = ag_{i_t}$ for some t. The result of the composition is defined by

$$(\Delta\varepsilon(ag_{i_1}), ..., \Delta\varepsilon(ag_{i_{t-1}}), \Delta\varepsilon(ag_{j_1}), ..., \Delta\varepsilon(ag_{j_s}), \Delta\varepsilon(ag_{i_{t+1}}), ..., \Delta\varepsilon(ag_{i_r})).$$

Hence the composition of Δ-rules leads to the distribution of changes of uncertainty coefficients among agents.

It is easy to observe that the composition of the Δ-rules leads to a new labelling $\{\varepsilon_{new}(ag)\}$ where for all non-root controllable agents of $\mathbf{c}$ we have $\varepsilon_{new}(ag) = \varepsilon_{new}(ag) + \Delta\varepsilon_{new}(ag)$ and $\Delta\varepsilon(ag)$ is obtained as the result of negotiations among agents about possible changes defined by compositions of Δ-rules. For the root agent of $\mathbf{c}$ and all non-controllable agents in $\mathbf{c}$ we assume $\varepsilon_{new}(ag) = \varepsilon_{new}(ag)$.

We obtain the following proposition establishing a basic property related to the correctness of mereological controllers.

Proposition 7. (*the sufficiency criterium of correctness of the controller*)

Let $F_{\mathbf{C}}(v) \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon(ag_{\mathbf{C}}))$ where v is the valuation of leaf agents of the (C, Φ, ε) -scheme $\mathbf{c}$ and let $F_{\mathbf{C}}(v') \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon'(ag_{\mathbf{C}}))$ where v' is a valuation of leaf agents of $\mathbf{c}$ such that $v'(b_{ag}) = v(b_{ag})$ for $ag \in Control(\mathbf{c})$, $\varepsilon'(ag_{\mathbf{C}}) < \varepsilon(ag_{\mathbf{C}})$. If $\{\varepsilon_{new}(ag)\}$ is a new labelling of agents defined by composition of some Δ-rules such that $\varepsilon_{new}(ag) = \varepsilon(ag)$ for $ag \in Un_control(\mathbf{c})$, $\varepsilon_{new}(ag_{\mathbf{C}}) = \varepsilon(ag_{\mathbf{C}}) = \varepsilon$ and $\{x_{ag} : ag \in Control(\mathbf{c})\}$ is the set of control parameters (inventory objects) satisfying $x_{ag} \models (st(ag), \Phi(ag), \varepsilon_{new}(ag))$ for $ag \in Control(\mathbf{c})$ then for the object $x_{new} = F_{\mathbf{C}}(v_1)$ constructed over the valuation v_1 of leaf agents in $\mathbf{c}$ such that $v_1(b_{ag}) = v'(b_{ag})$ for $ag \in Un_control(\mathbf{c})$ and $v_1(b_{ag}) = x_{ag}$ for $ag \in Control(\mathbf{c})$ holds $x_{new} \models (st(ag_{\mathbf{C}}), \Phi, \varepsilon(ag_{\mathbf{C}}))$.

□

It is not difficult to extract from Tables 1,2,3 of Example 1 the Δ - rules for controlling the system of complex objects at M.

Multistrategy learning in our approach

It has been emphasised (Jenkins 1993), (Michalski 1994) that a multistrategy learning is achieved whenever knowledge sources (agents) are turned into independent learning units (agents). In our approach this effect is achieved by learning appropriate constructs from data tables of independent agents. In the learning process the agents use empirical induction (in order to extract rules from data tables), abduction (to fit objects satisfying the conclusions of the rules when premises are known and selected) and reasoning by analogy (when carrying the synthesis along routes outlined by standards at consecutive agents).

The above approach can be treated as a first step towards modelling complex distributed dynamical systems. We expect that it can be extended to solve control problem for complex dynamical systems i.e. dynamical systems which are distributed, highly nonlinear, with vague concepts involved in their description. It is hardly to expect that the classical methods of control theory can be successfully applied for such complex systems.

Analysis

The process of analysis is split into the design and synthesis spaces. Given an object x, its analysis can be realized in the following sequence of steps.

Step 1. An agent $ag \in Ag$ such that $x \in U(ag)$ is chosen

Step 2. An agent dag such that $\pi(ag, dag)(x)$ is defined chooses its decomposition.

Step 3. The decomposition continues, resulting in the *analysis tree* for $\pi(ag, dag)(x)$.

Step 4. A design scheme for $\pi(ag, dag)(x)$ is designed.

Step 5. A (C, Φ, ε)-scheme for synthesis of x is found.

This constitutes the analysis of constructibility of x.

Further analysis includes e.g. the analysis of stability as well as robustness of (C, Φ, ε).

Acknowledgements

The authors appreciate financial help of the Polish - Japanese Institute of Computer Techniques.

References

Amarel, S. 1991. PANEL on AI and Design. In Proceedings of IJCAI-91; Twelfth Joint International Conference on Artificial Intelligence, 563-565. San Mateo: Morgan Kaufmann.

Dubois, D., Prade, H., and Yager R.R. 1993. *Readings in Fuzzy Sets for Intelligent Systems.* San Mateo: Morgan Kaufmann.

Jenkins, W.T. 1993. INTELOG: A Framework for Multistrategy Learning. In Proceedings MSL-93; The Second International Workshop on Multistrategy Learning, 58-68. Fairfax, VA: Center for Artificial Intelligence, George Masson University.

Komorowski, J.; Polkowski, L.; and Skowron, A. 1996. Towards a rough mereological logic for approximate solution synthesis. *Studia Logica.* Forthcoming.

Leśniewski, S. 1992. Foundations of the general theory of sets. In *Stanislaw Leśniewski, Collected Works,* Surma,S. J., Srzednicki, J. T., Barnett, D. I., and Rickey, V.F., eds., 128-173. Dordrecht: Kluwer.

Michalski, R.S. 1994. Inferential Theory of Learning. In *Machine Learning A Multistrategy Approach.* Vol.IV, 3-61. San Francisco: Morgan Kaufmann.

Pawlak, Z. 1991. *Rough sets: Theoretical Aspects of Reasoning about Data.* Dordrecht: Kluwer.

Pawlak, Z., and Skowron, A. 1994. Rough membership functions. In *Advances in The Dempster - Shafer Theory of Evidence,* Yager, R.R., Fedrizzi, M., and Kacprzyk, J., eds., 251-271. New York: Wiley.

Polkowski L., and Skowron A. 1994a. Rough mereology. In Proceedings ISMIS -94; International Symposium on Methodologies for Intelligent Systems, 85-94. Lecture Notes in Artificial Intelligence 869. Berlin: Springer-Verlag.

Polkowski, L., and Skowron, A. 1994b. Introducing rough mereological controllers: Rough quality control. In Proceedings RSSC-94; The Third International Workshop on Rough Sets and Soft Computing, 78-95. San Jose State University, CA.

Polkowski, L.; Skowron, A. 1996a. Rough mereology: A new paradigm for approximate reasoning. *International Journal of Approximate Reasoning.* Forthcoming.

Polkowski, L., Skowron, A. 1995. Rough mereology and analytical morphology: New developments in rough set theory. In Proceedings of WOCFAI-95; Second World Conference on Fundamentals of Artificial Intelligence, 343-354. Paris: Angkor.

Polkowski, L., Skowron, A. 1996b. Rough mereological approach to knowledge-based distributed AI. In Proceedings of WCES-3; Third World Congress on Expert Systems. New York: Cognizant Communication Corporation.

Skowron, A., and Polkowski, L. 1995. Rough mereological foundations for design, analysis, synthesis and control in distributed systems. In Proceedings of the Second Annual Joint Conference on Information Sciences, Wrightsville Beach, NC.

Skowron, A. 1995. A synthesis of adaptive decision systems from experimental data. In Proceedings of Fifth Scandinavian Conference on AI, SCAI-95, 220-238. Amsterdam: IOS Press.

II. Cognitive Models

The Basic Level and Privilege in Relation to Goals, Theories, and Similarity

Douglas L. Medin, Elizabeth B. Lynch, and John D. Coley

Northwestern University
Department of Psychology
2029 Sheridan Road
Evanston, IL 60208-2710

Scott Atran

CNRS-CREA, Ecole Polytechnique
and University of Michigan

Abstract

It has been argued that some categories are *privileged* in that the correlational structure of features in the environment creates natural chunks that are compelling to learners. If so, then differences in goals should not affect the organization and use of such so-called privileged categories. This paper reports research where the amount and type of expertise is varied for the same domain. Goals exert a major effect on category organization as well as on category-based induction. Even for the case where some categories seemed to be relatively privileged, the basis for this privilege may derive from linguistic cues rather than strictly empirical structure.

Introduction

What are the fundamental principles of category organization and why are some categories more coherent, easy to learn, and useful than other categories? These issues continue to be central to our understanding of concepts and conceptual behavior. In the present paper we focus on two broad questions: 1. How do similarity and theory structure categories? and 2. What makes some categories privileged relative to others? We begin by placing our work in the context of earlier research, next describe some new psychological experiments, and then bring out the implications of our findings for similarity theories and notions of privilege.

A. Similarity versus Theory. Questions about the respective roles of similarity versus theory or explanation serve to organize much of the ongoing work on categorization (Komatsu, 1992; Medin and Heit, in press; Murphy, 1991, 1993a, b; Smith and Heise, 1992 for reviews and Burns, 1992; Fisher, Pazzani, and Langley, 1991; VanMechelen, Hampton, Michalski, and Theuns, 1993; Nakamura, Taraban, and Medin, 1993 for relevant edited volumes). To make similarity meaningful, there must be constraints on what features or aspects enter into a comparison and the weights assigned to them. We are seeing both more sophisticated models of similarity and models of categorization capable of addressing the flexible and dynamic aspects of similarity (e.g. Anderson, 1991; Barsalou, 1991, 1992, 1993; Billman, 1992; Clapper and Bower, 1994; Estes, 1994; Gentner and Markman, 1994; Medin, Goldstone, and Gentner, 1993; Myers, Lohnmeller, and Well, 1994; Nosofsky, Gluck, Palmeri, McKinley, and Glanthier, 1994).

Meanwhile, there has been corresponding progress in understanding the role of knowledge and theories in organizing concepts (Gelman and Wellman, 1991; Keil, 1989; Malt, 1995; Michalski, 1989; Murphy, 1993a,b; Pazzani, 1991; Rips and Collins, 1993; Waldman and Holyoak, 1992; Walker, 1992, and edited volumes by Hirschfeld and Gelman, 1994; Michalski and Tecuci, 1994). Although one might view knowledge-driven learning as an alternative to similarity-based approaches, the scholarly consensus has been in the direction of models which combine similarity-based and explanation-based categorization (to give but a few examples, see Billman and Knutson, 1996; Choi, McDaniel, and Busemeyer, 1993; Fisher, Pazzani, and Langley, 1991; Fisher, and Yoo, 1993; Heit, 1993, in press; Mooney, 1993; Wisniewski and Medin, 1994, a,b).

Although empirical work on similarity-based and knowledge-driven learning has not restricted itself to artificial materials (e.g. Brooks, Norman, and Allen, 1991) neither has it made systematic contact with the kinds of categories associated with theories which emphasize domain-specific categorization principles (e.g. Hirschfeld and Gelman, 1994). The focus of the present project is on one such domain, biological categorization. We shall not argue that categorization processes are necessarily domain-specific, but rather we focus on the point that there are a number of research issues that become more coherent when analyzed within the context of particular domains. Biological reasoning is of particular interest because, in this domain, theorists have laid out distinct positions on the role of mind and world in categorization. Furthermore, the domain of biological kinds seems particularly attractive from the perspective of similarity-based categorization -- surely our perceptual system is an adaptation to the natural world and if similarity models are going to succeed anywhere, it should be here.

B. Mind and world. It is straightforward to discern points on an underlying dimension that represents the extent to which goals and belief systems versus world structure are said to influence organization and reasoning

with biological categories. At one end of the continuum are those who view taxonomies as direct reflections of the structure of the natural world. This view is associated primarily, but not exclusively, with anthropology. On this account folkbiological categories are essentially given by encounters with the environment and virtually "crying out to be named" (Berlin, 1992). The idea is that people are innately prepared by their perceptual apparatus to form categories and little conceptual effort is needed to produce folkbiological classification because in Boster's (1991) terms, "the source of structure in biological similarity judgments is in the world, not in the brain."

A diametrically opposite position is that goals and beliefs determine categorization and reasoning. This view argues that empirical observations underdetermine categories and that the very notion of similarity is theory-laden (Murphy and Medin, 1985; Ellen, 1993). In the words of Gelman and Coley (1991), "to calculate similarity, it is necessary to have a theory of which features are important and how they should be weighted." From this perspective, one might expect differences in biological categorization and reasoning to the extent that associated goals, theories, and belief systems differ.

Although the above two views represent end points on a continuum, they may not be so different as first appears. To begin with, advocates of the "structure in the world" position would concede that structure is relative to a perceptual (and conceptual) apparatus capable of apprehending it. In the same vein, human goals, theories, and interests may have universal components that are not readily separated from world structure. For example, Anderson's (1990) rational model of categorization is premised on a (universal) goal of understanding the world and making predictions about it. Goals and theories must make contact with the world and do not have the luxury of ignoring structure.

In the research motivating the present paper, goals and interests are varied by virtue of the occupational demands and activities associated with different types of expertise for a common biological category, trees. Our experts include taxonomists, landscapers, and parks personnel. Before describing these differences in greater detail, we turn to another dimension relevant to the contributions of mind and world to categories, hierarchical level.

C. The basic level and other privileged categories. Another perspective on the integration of similarity and theory is the view that their relative contributions vary as a function of types or levels of categorization. For example, in a now classic paper Rosch, Mervis, and their associates (Rosch, Mervis, Gray, Johnson, and Boyes-Braem, 1976) argued that "in the real world information-rich bundles of perceptual and functional attributes occur that form natural discontinuities and that basic cuts in categorization are made at these discontinuities" (p. 385). That is, the correlational structure of features of entities in the environment creates natural clusters or basic level categories which play a central role in many cognitive processes associated with categorization. Basic level categories such as Chair, Hammer, and Dog may be contrasted with more general superordinate categories (Furniture, Tool, and Animal) and more specific subordinate concepts (e.g. Recliner, Ballpeen hammer, Labrador retriever). Rosch et al (1976) evaluated a number of criteria associated with category use and they showed a remarkable convergence on the basic level of categorization. Basic level categories are the most inclusive categories that 1 possess numerous common attributes 2 people interact with using similar motor programs 3 have similar shapes and 4 can be identified from averaged shapes of members of the class. Furthermore, basic level categories are preferred in adult naming, first learned by children, and most easily identified (relative to subordinate and superordinate categories). It is as if basic level categories are out in the environment waiting for people to apprehend them.

Although researchers have not tended to employ the variety of converging measures educed by Rosch, the basic level superiority has proven to be quite robust (e.g. see Lassaline, Wisniewski, and Medin, 1992 for a review). There is however, at least one serious puzzle, a puzzle that has recently motivated a significant portion of our research. Rosch et al. (1976) made initial guesses about which level in a hierarchy would prove to be basic and these guesses were generally correct. For biological taxonomies, however, the initial guesses were based on anthropological observations (Berlin, Breedlove, and Raven, 1966, 1973) and in each case these guesses turned out to be wrong. For example, instead of Oak, Trout, and Eagle being basic, Tree, Fish, and Bird met the Rosch et al criteria. The puzzle concerns why the anthropological and psychological criteria did not converge. Berlin (1992) argues that there is one level of categorization of plants and animals that is especially salient but he believes that this privileged level corresponds to the genus level in scientific taxonomy (e.g. Oak). The level that Rosch et al found to be basic corresponds to a much higher level in a scientific taxonomy, Class or even Division. Why the difference?

1. Expertise? One way of reconciling Rosch with Berlin is to suggest that the basic level changes with expertise (Rosch et al., 1976 discussed this possibility; see also Mervis, 1987). People living in urban, high technology settings may be relative novices with respect to living kinds. Surprisingly, there has been only a handful of studies of expertise and categorization (Chi, 1983; Chi, Hutchinson, and Robin, 1989; Chi, Feltovich, and Glaser, 1981; Gobbo and Chi, 1986; Johnson and Mervis, 1994; see also Honeck, Firment, and Case, 1987) and even less work on changes in the basic level with expertise (Tanaka and Taylor, 1991; Johnson, 1992; Mervis, Johnson, and Scott, 1993; Palmer, Jones, Hennessy, Unze, and Dick, 1989). These latter studies reveal changes in naming practices and feature listings with expertise as well as some evidence that subordinate level categorization may come to be as rapid as basic level categorization.

2. Multiple privileged levels? A complementary hypothesis derives from the original Berlin et al folkbiological data (see also Berlin, 1992). Berlin suggested that five or even six levels of taxonomic structure were privileged in the sense of being stable across cultures: 1 unique beginner (plants versus animals), 2 life form (e.g. tree, grass, bush, bird, fish), 3 intermediate (e.g. hooved mammals), 4 folk-generic (usually corresponding to genus), 5 folk-specific (corresponding to species), and 6 varietal (e.g. kinds of corn). Not all of these levels are equally salient. For example, intermediates are less common and the folk-specific and varietal levels are much more likely to be seen in agricultural societies than in foraging cultures. Berlin's analysis raises the possibility that there may be multiple privileged levels in a taxonomic hierarchy. Note that the biological category level that Rosch et al (1976) argued was basic (e.g. bird, fish) corresponds to Berlin's life form level (see also Mandler, Bauer, and McDonough, 1991). It is possible that both the life form and folk-generic levels are psychologically salient but that the subjects tested by Rosch et al were unfamiliar with the particular genera used. The bird watchers studied by Tanaka and Taylor (1991) would have had the requisite experience and their rapid subordinate level categorization is consistent with the folk-generic level being salient (for their dog experts, the subordinate level would correspond to folk-specific or varietal).

Figure 1 shows one way of conceptualizing multiple privileged levels of categorization. There are many candidate metrics for ease of categorization (e.g. Corter and Gluck, 1992; Jones, 1983; Murphy, 1982) but suppose, for illustrative purposes , that categories are salient if the ratio of within category similarity to between category similarity exceeds some criterion. Then (A,B) would be salient because of high within category similarity, (A,B,C,D,E,F,G,H) salient because of low between category similarity, and (A,B,C,D) would be salient for intermediate reasons. (This analysis assumes that similarity is fixed; it is certainly possible that experience and expertise leads to increased sensitivity or sensitivity to new features or dimensions). All three categories should be more salient than the category (A,B,C,D,E,F).

3. Comparability of measures? Rosch et al (1976) pinpointed the basic level with a range of converging measures not seen before or since. Therefore, apparent cross-cultural differences or changes with expertise may reflect changes in some measures but not others. The pattern of convergences and divergences as a function of knowledge may prove to be particularly informative with respect to categorization theories.

D. Goal and theories. Of course, interest in expertise goes considerably beyond possible changes within the basic level. Knowledge and expertise may lead to changes not just in category salience but also in category organization and reasoning with categories (Boster and Johnson, 1989; Carey, 1985; Carey and Spelke, 1994; Johnson, Mervis, and Boster, 1992; Keil, 1994). There is at least suggestive evidence that goals and functions associated with expertise may influence category organization. For example, Boster and Johnson (1989) found that expert fishermen judged similarities among fish on both functional and morphological criteria, while novices used morphological criteria alone. As a result, novice judgments corresponded better to scientific taxonomy than the judgments of experts. Experts were also more variable in their judgments, raising the possibility that they had multiple classification schemes. Boster and Johnson (1989) did not look at hierarchical levels. One implication of the basic level (as given by the environment) perspective is that functions and theories may influence categorization more at the superordinate and subordinate level than at the basic level.

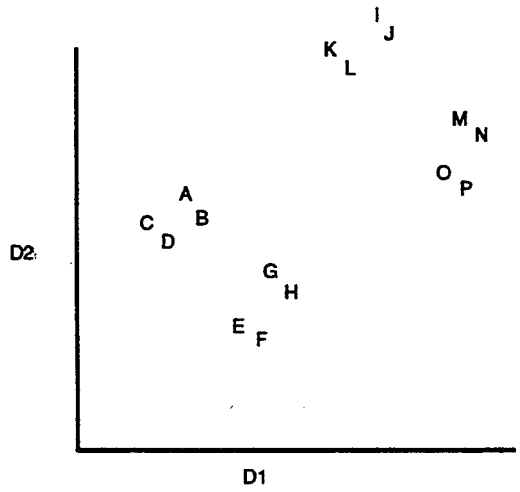


Figure 1. Spatial representation of similarity as the inverse of distance.

E. Overview of present research. Recently, we have been analyzing levels of categorization, category organization, and category-based reasoning as a function of expertise. In one main project, we have been assessing the categorization of and reasoning about trees by taxonomists, landscapers, and parks personnel. As may be obvious from these occupations, a shared ability to identify trees may be coupled with distinctly different goals and interests. Parks personnel spend considerable time planting new trees and culling and pruning diseased, damaged and aging trees. Landscapers focus on aesthetic functions and are only secondarily concerned with maintenance. Taxonomists are involved in a variety of actions ranging from systematics to education and consulting. A key question is the extent to which these subgroups differ in their conceptual behavior. Finally, we include one cross-cultural measure of basicness, comparing Northwestern University undergraduates with a Maya Indian group in Guatemala. In addition to a host of other differences, the Maya are expert folkbiologists, especially in comparison with undergraduates.

Studies of Biological categorization and Reasoning

Our primary aim is to examine category organization as a function of knowledge and type of expertise. How do different goals, theories, and functions affect category organization and do they affect all levels of a taxonomic hierarchy or are some levels (e.g. the basic level) immune from such factors?

A Expert sorting. In the first experiment, we examine the groupings of tree species, and explanations for those groupings, provided by three types of tree experts. Are there differences in spontaneously generated taxonomies? If there are systematic differences, the precise nature of them might specify the ways in which human interests and goals impact folkbiological taxonomic systems. There are several distinct theoretical positions with respect to differences in categorization as a function of type of expertise. We distinguish four.

1. Predictions concerning expertise. One possibility is that, so to speak, all roads lead to Rome--differing goals and concerns may not penetrate the compelling spontaneous categorization that biological kinds afford. But one does not have to be a structure-in-the-world theorist to anticipate the absence of group differences. Goals and associated differences in patterns of interaction may not affect biological classification because experts interact with biological kinds in so many different ways that a general purpose organizational scheme has the greatest utility. In either case we would expect a high correspondence between folk and scientific taxonomy, justifications that are uniform and predominantly taxonomic, and similarly structured taxonomic hierarchies across groups.

Alternatively it may be that goals and activities do affect biological classification but that their influence on categorization is difficult to detect because biological objects have highly correlated features. To the extent that features are correlated within categories, it will increase the likelihood that classifiers with different orientations will create the same categories for different reasons. For example, perhaps feathers are important to Bob because he uses feathers to make pillows. Bob has a category of all the objects that he can use to procure feathers. Alternatively, perhaps wings are important to Jane because she studies flight of vertebrates. Thus Jane has the category "winged vertebrates." Bob and Jane have almost identical categories, but their rationale for placing the objects into these categories differs dramatically. In short, similar categories in terms of content may mask differences in the features relevant to those categories in the minds of the categorizers (see also Boster & D'Andrade, 1989).

Of course there is also psychological research that would, by extension, support more substantial differences as a function of type of expertise. Different goals, activities, and knowledge may well lead to distinct hierarchical structures—fundamentally different ways of organizing nature. This possibility represents the strongest and most pervasive way in which human goals might shape folkbiological taxonomy. This leads to two further possibilities.

Models of categorization allow for selecting weighting of features or dimensions (e.g. Medin and Schaffer, 1978; Nosofsky, 1992; and see Wisniewski and Medin, 1994 for the more radical possibility that theories may influence the construction and construal of features). Distinct types of expertise may lead to different patterns of attention to and weighting of morphological properties of trees. One could think of these differences as yielding a variable stretching and shrinking of dimensions of some multidimensional similarity space (e.g. Boster and D'Andrade, 1989; Nosofsky, 1992). On this account the same general similarity space would describe all experts and any differences might be attributed to variable weighting of a set of common dimensions. For example, taxonomists may differ from landscape and parks maintenance groups in giving extra weight to features associated with reproduction. If differences in the weighting of dimensions were strong enough to overcome the influence of correlated dimensions, one might observe group differences in sorting as well as differences in the justifications.

The final position we shall examine is motivated by Barsalou's analyses of ad hoc or goal-derived categories (Barsalou, 1982, 1983, 1985, 1991). He argues that categories constructed in the service of goals (e.g. things to take on a camping trip) violate the correlational structure of the environment. In addition, he contrasts context-independent properties of concepts (e.g. basketballs are round) and context-dependent properties (basketballs can float) and notes that goals often serve to activate context-dependent properties. Especially important for our purposes, Barsalou suggests that frequently used goal-derived categories may develop well-established category representations much like those of common taxonomic categories (see Barsalou and Ross, 1986).

Barsalou's analysis suggests that extensive use of goal-derived categories may lead to the following patterns of spontaneous sorting: 1 the correspondence between sorting and scientific taxonomic distance should be low, 2 the justifications for sortings should be utilitarian (linked to goals), 3 the folk taxonomic structure may look more like a series of goal-relevant cross-cuts than a true hierarchical taxonomy. Finally, goals do not necessarily partition the full set of entities in a domain. This raises the possibility that sorting may reflect a mixture of goal-derived and taxonomic categories.

The job descriptions of our three sets of experts suggest that landscape personnel are most likely to frequently use goal-derived categories. Because much of their work involves placing the right tree in the appropriate context, utilitarian functions (providing shade, aesthetic qualities, etc.) are salient to landscapers. On the other hand, the day

to day activities of parks maintenance personnel involve caring for an existing population of trees, rather than using trees to achieve particular goals. Finally, we assume that the taxonomists will use scientific, rather than goal-derived categories because most of them frequently use the scientific taxonomy in their work.

In summary, there are theoretical motivations to expect anything from no effect of type of expertise on sorting to a radical restructuring in the service of goals. The justifications for sorts provide data bearing on the idea that the same categories may be created for different reasons. The degree to which expert groups differ in their clustering of tree species should indicate the degree to which human goals and interests influence folkbiological classification.

2. Results. Space does not permit a full analysis of the procedures and results (see Medin, Lynch, Coley, and Atran, in press for details) and we are limited to a summary. Participants were asked to sort 48 tree names into as many categories they wished to create clusters that go together "by nature." In addition the experts were asked to justify or describe the basis for each category. This procedure was repeated to successively lump or split categories to produce a hierarchical taxonomy (the results would be essentially the same if we only considered the initial sorting.

The sorting of the taxonomist group provides a useful comparative standard. Individual taxonomies were combined to produce a consensual taxonomy (again for procedural details and mathematical justification, see Medin et al., in press). For ranks ranging from genus to family up to division, as taxonomic distance increased, mean distance in their consensual clustering increased. Furthermore, the taxonomist's sorting never broke up the genus level. Finally, justifications were almost exclusively in terms of taxonomy and morphology.

The Parks personnel were similar in certain key respects to the taxonomists. Sorting and justifications were primarily in terms of taxonomic and morphological properties. The correlation between scientific taxonomic distance and folk distance was lower, in part because in several cases the morphological properties used in sorting are not necessarily properties given high weight in scientific taxonomy. For example, consensual categories based on type of fruit or nut partially crosscut scientific taxonomy. In general, the greatest correspondence with scientific taxonomy was observed at the genus and family levels. The parks group also differs from the taxonomist group in its inclusion of a clear utilitarian category, weed trees (fast-growing weak-wooded trees that create maintenance problems).

Where does the parks group fall with respect to the question of whether the genus level is privileged in sorting? The picture is mixed but perhaps the best summary statement is that genus is relatively but not absolutely privileged. It is relatively privileged in that the genus level was more likely to be preserved and less likely to be broken up than the family level. In addition, there was a much larger jump in folk distance between the genus level and the family level than between the family level and higher taxonomic ranks. But genus is not absolutely privileged. Even in the most favorable condition where folk terms and science converge on the genus level, the sorting revealed a corresponding category only about half the time.

The pattern of sorting and justification for the landscape group showed the largest influence of utilitarian factors and the largest departure from scientific taxonomy. The correlation of landscapers' folk distance with scientific distance was low and genus-level categories were broken up over half the time. The greatest correspondence between scientific and folk taxonomy appeared to be at the family level though even here it was not strong. The justifications of landscape group members also reflect utilitarian concerns. Weed status, landscape utility, size, and aesthetic value were mentioned more frequently as justifications than morphological properties. It is hard to escape the impression that the landscape sorting is a utilitarian structure imposed on nature rather than a reflection on categories salient in the environment.

Overall, the sorting and justification profile for the landscape group corresponds remarkably well with Barsalou's (1983) analysis of goal-derived categories. The sortings crosscut the correlational structure of the environment (as embodied in scientific taxonomy) were justified in terms of multiple utilitarian factors, and created a hierarchy based on combinations of goal-relevant factors. The modest agreement with science and with the other two groups of experts appears to derive from a combination of goals incompletely partitioning the set of trees and a convergence of morphological and utilitarian criteria on the same categories. Apparently years of experience lead the goal-derived categories of landscape workers to become well-established in memory. The next experiment addresses the question of whether these categories are also used in reasoning.

Summary. Taken together, these comparisons among reveal specific group commonalities and differences. Although there were some clusters of trees that either are similar enough, or convergent enough across goals to be put together regardless of type of expertise, group differences stand out as salient. Most strikingly, what were organized as taxonomic categories by parks and taxonomist groups appeared as goal-derived categories for the landscape group. This does not show that this is the only organization available for the landscape group (it surely is not) but it underlines the accessibility of a utilitarian organization. It remains to be seen whether the nontaxonomic bases for categorization carry over into other conceptual tasks.

B Category-Based Reasoning.

1. Rationale. The second experiment provides a potentially converging measure of category organization in the form of a category-based reasoning task (Rips, 1975). In this experiment, we examine competing bases for extending novel properties from one tree species to another. Experts are told that a tree has some novel

property and then asked which of two alternative trees is most likely to have that property. Alternatives vary in their scientific (and folk) distance from the target. The question of interest is the degree to which different kinds of categories promote inferences about physiological properties. For example, we might say that a new disease has been discovered that Boxelders can get and ask whether it is more likely that a Sugar Maple or a Cottonwood can also get the disease. The Boxelder is a maple (its other common name is Ashleaf Maple) and, therefore, a taxonomically motivated answer would be Sugar Maple. Boxelders and Cottonwoods are both frequently nominated as garbage trees so a functionally motivated answer would be Cottonwood. Alternatively, if such categories only represent a few relevant shared features but fail to capture important underlying similarities, they might not promote inferences as well as scientific categories do; i.e., the expert might still prefer to draw an inference from Boxelder to Sugar maple.

We used three distinct types of so-called "blank" properties (a novel disease, novel physiological property or novel property associated with cross-fertilization), to see if different kinds of properties evoked different forms of category organization. If results converge with those of Experiment 1, taxonomists' responses should closely follow scientific relations, whereas those of landscape and parks personnel should not necessarily do so.

2. Results. Again, taxonomists provide a standard that can be contrasted with the other groups. Taxonomists pick the taxonomically (and presumably morphologically) more similar tree over the alternative more than 90 percent of the time. Landscapers agree more with scientific taxonomy than with their own consensual sorting (83 percent versus 63 percent) whereas parks workers show the opposite pattern (87 percent agreement with their consensual sorting versus 77 percent agreement with taxonomy). At a finer level of analysis, landscapers pick the taxonomic choice 93 percent of the time when the standard and the taxonomic alternative are of the same genus. At levels above the genus the taxonomic alternative is picked only 54 percent of the time. When taxonomy and consensual sorting are in agreement at higher levels (for Landscapers) that alternative is selected 88 percent of the time. Overall, our observations suggest that the genus level is privileged in the reasoning of landscapers.

It is harder to separate scientific from folk taxonomy for the parks personnel but, relative to landscapers, they show a smaller influence of shared genus. For example, on the triad where Boxelder is the base and the alternatives are Cottonwood and Sugar Maple, parks personnel consistently select Cottonwood (a fellow garbage tree) rather than Sugar Maple (a fellow maple). It does seem that the parks subgroup privileges the genus level when it is marked in language (an observation consistent with this is that a number of parks participants grouped the American Mountain Ash with the Green Ash and White Ash in sorting, even though the American Mountain Ash belongs to a different family and order taxonomically). In short, the folk generic level may be privileged for parks personnel but the reason for this may have as much to do with language as with perceptual salience.

For both the parks and taxonomist subgroups one could say that the same category organization revealed by sorting carries over into reasoning. For the taxonomists this is essentially equivalent to scientific taxonomy. Some of their justifications were in terms of known patterns of disease but these are highly correlated with taxonomic distance. For the parks subgroup, the taxonomy guiding reasoning appears to be their consensual folk taxonomy. When folk and scientific taxonomy made different predictions, they were more likely than landscaper personnel to reason in accord with their own consensual taxonomy. When folk and scientific taxonomy conflicted and the scientific relation was above the genus level, parks subgroup responses were consistent with consensual folk taxonomy as a basis for inferences. The correlation of morphological characters with utilitarian function makes it difficult to pinpoint the exact basis for reasoning but the data strongly suggest that at least weed status, and perhaps landscape utility, influenced choices.

In sharp contrast to the other two groups, the landscape subgroup used a reasoning strategy that departed substantially from their consensual sorting. In sorting, landscapers were most likely to cross-classify genus-level groups of trees; in reasoning, they almost never drew inferences on the basis of folk categories when the scientific choice was at the genus level. As mentioned above, when folk and scientific distance conflicted, landscapers were more likely than the parks subgroup to base inferences on science. We hasten to point out that the landscape reasoning data agreed more with parks consensual sorting than with either their own consensual sorting or with science (though in the latter case the differences were small). The fact that parks subgroup sorting and science predicted the landscape subgroup reasoning about equally well makes the basis for landscape reasoning ambiguous. Folk nomenclature had no reliable effect on patterns of reasoning for landscapers. The landscape folk categories weed and ornamental had some effect on landscapers' reasoning, but only at levels above the genus.

Overall, results support the claim that genus-level categories are inductively privileged. For the landscape subgroup this privilege appears for scientific genera regardless of whether folkterminology marks it; for the parks subgroup, privilege extends most clearly to folkgenerics marked by common terms. It remains to be seen whether this difference can be explained in terms of amount of formal education. The next experiment extends our analysis of rank and privilege to other populations.

C. The basic level and Inductive strength: Experiment 3.

1. Rationale. One important function of categories is to promote inferences; if we know a property is true of members of a given category, we can make an educated guess as to whether that property is true of other related categories. This inductive potential should be maximized

at the privileged level of taxonomy. Maximizing induction potential as a single criterion would dictate the most specific categories in that any property true of higher level categories must be true of subordinate categories. Whatever is true of all fish will be true of all trout and all rainbow trout. On the other hand, people encounter more fish than trout and more trout than rainbow trout and induction potential may trade off number inferred properties and confidence in inductions with frequency. In the limiting case of specificity one would not generalize beyond an individual case. The need to draw inferences and make predictions in the face of uncertainty, however, may pick an intermediate level of categorization as maximizing the tradeoff between frequency and accuracy.

In all of the Rosch et al experiments, the logic of locating the basic level was the same; the basic level was the level above which much information was lost, and below which little information was gained. For instance, in a feature-listing task, subjects listed a mean of 3 common features for the superordinate category furniture, a mean of 9 features for basic level furniture categories (e.g., table) and an average of 10.3 features for subordinate furniture categories (e.g., kitchen table). There was a large gain in information when going from the superordinate to the basic level (6 new common features are added, in this example), and only a slight gain going from the basic to the subordinate (1.3 features). The third experiment extends this logic to inductive reasoning; inductive inferences to a privileged category should be significantly stronger than inferences to more general categories and not significantly weaker than inferences to more specific categories. If table (as opposed to kitchen table or furniture) is the basic level, then inferences about tables should be roughly as strong as inferences about kitchen tables, and much stronger than inferences about furniture. Inductive strength should show a sharp drop as one moves above the basic level.

2. Design. Recall that anthropologists studying traditional cultures find that the folk-generic level (e.g., sparrow, trout, oak) is privileged; psychologists, studying mostly urbanized populations, find that the "basic" level for living things maps onto the rank of life-form (e.g., bird, fish, tree). To help shed light on this puzzle, we undertook a series of studies using category-based induction to investigate privileged taxonomic levels. As in the second experiment participants were told that an unfamiliar property was true of all members of a category, but then were asked how likely it was to be true of the members of a more general category. Using Rosch's logic for diagnosing the basic level, inferences to a privileged level should be judged more likely than those to more general levels, and no less likely than those to more specific levels. If Berlin is correct about the primacy of folk-generic categories, this pattern should center on taxa of the folk-generic rank. Specifically, we would expect inferences to folk-generic taxa to be relatively strong, inferences to taxa of lower order only marginally stronger, and inferences to taxa of higher rank to be relatively weak. That is, there should be a breakpoint or elbow between the folk-generic level and higher levels.

The research participants in this study were both Northwestern University undergraduates and Maya from the Peten region of Guatemala (see Coley, Medin, and Atran , submitted, for details). If expertise or contact with a rich natural world is the critical factor in induction, then undergraduates may privilege the life-form level and Maya the folkgeneric level in induction.

3. Results and Discussion. Overall, inferences to folk-generic categories were consistently stronger than inferences to more general categories, and no weaker than inferences to more specific categories (for details, see Coley, et al., submitted). This result held for both the undergraduate and Maya populations. The gain in inductive strength was greatest moving from life-form to folk-generic categories, suggesting that folk-generic categories are psychologically "basic" with respect to induction. For American students, this result is surprising because Rosch's work suggests that "basic object categories" for living things fall at a more general level.

Our results leave us with the two questions about the relations between folkbiological taxonomy and induction. First, why did the privileged level for Americans with respect to induction fail to correspond to Rosch's "basic" level? Second, why were categories at the same level of specificity inductively privileged for both Americans, with little first-hand experience of the natural world, and Maya, who have a great deal?

One possible account for the discrepancy between results of the present experiments and those of Rosch et al. is to propose different functions for within- and between-category similarity in the sets of tasks. On this account, proposed by many researchers since Rosch et al. (see Lassaline et al., 1992, for a review), the inductive strength of a category is driven by within-category similarity alone, whereas status as a "basic object category" results from maximum within-category similarity relative to between-category similarity. For example, trout might be privileged for induction over fish because people perceive all trout to be more alike--and therefore likely to share properties--than all fish; i.e., within-category similarity is higher for trout than fish. In contrast, fish rather than trout might be diagnosed as a "basic level category" because although all trout are a lot alike, they are also fairly similar to other fish. All fish are somewhat similar--less so than trout--but are very different from other animals. So, fish is "basic" because the ratio of within-category similarity to between-category similarity is higher for fish than for trout. On this account, life-form categories (e.g., fish, tree) may be privileged on Roschian tasks because they maximize within-category similarity relative to between-category similarity, and folk-generic categories (e.g., trout, oak) may be privileged in induction because they maximize within-category similarity alone.

Figure 2a illustrates one way in which, on this account, the inductively privileged level and the "basic" level need not coincide. In Figure 2a, the largest drop in within-

category similarity occurs between the folk-generic and the life-form level, predicting an inductive advantage for folk-generic categories. However, the criterion for basic level status--within-category similarity relative to between-category similarity--is maximized at the life-form level (as indicated in Figure2a by the distance between the two lines). Thus, it is possible to conjecture patterns of within- and between-category similarity that yield folk-generic categories privileged for induction but life-form categories privileged for perceptual tasks.

Unfortunately, this account fails when one considers our results, and logical constraints on the relation between within- and between-category similarity. First, according to the pure similarity account, inductive strength should reflect within-category similarity only. We consistently found that the most pronounced dropoff in inductive strength was between the folk-generic and life-form level. This suggests that the largest drop in within-category similarity also occurs between the folk-generic and life-form level. Second, within-category similarity at a given level is logically equivalent to between-category similarity at the immediately subordinate level. For instance, a judgment of within-category similarity for a life-form (e.g., how similar are all fish) is equivalent to aggregated judgments of between-category similarity for folk-generic subordinates of that life-form (e.g., how similar is trout to bass, shark to goldfish, etc.). Therefore, the major breakpoints in within- and between-category similarity cannot both occur between the folk-generic and life-form levels, as required in Figure 2a. On the contrary, within-category similarity at the life-form level must correspond to between-category similarity at the folk-generic level. To suppose otherwise would literally require two different answers to precisely the same question.

Figure 2b is a revision of Figure 2a based on these observations. In accordance with (1), ratings of inductive strength collapsed across studies are substituted in 2b for the hypothetical within-category similarity ratings in 2a. In accordance with (2), between-category similarity at each level in Figure 2b is estimated by projecting the approximations of within-category similarity (i.e., aggregate ratings of inductive strength) to the immediately subordinate level. As Figure 2b indicates, if the major break in within-category similarity occurs between the folk-generic and life-form levels, then the ratio of within- to between-category similarity is maximized at the folk-generic level as well. If these observations are correct, it is impossible to simultaneously observe (1) the largest drop in within-category similarity between folk-generic and life-form categories, and (2) the ratio of within- to between-category similarity maximized at the life-form level. Therefore, a purely similarity-based account such as described above cannot explain both the present results favoring the folk-generic level for induction and Rosch et al.'s findings favoring the life-form level for categorization.

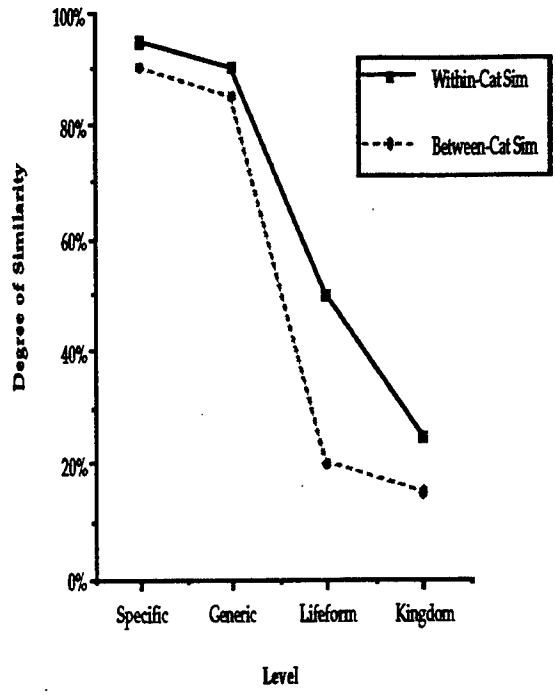


Figure 2a

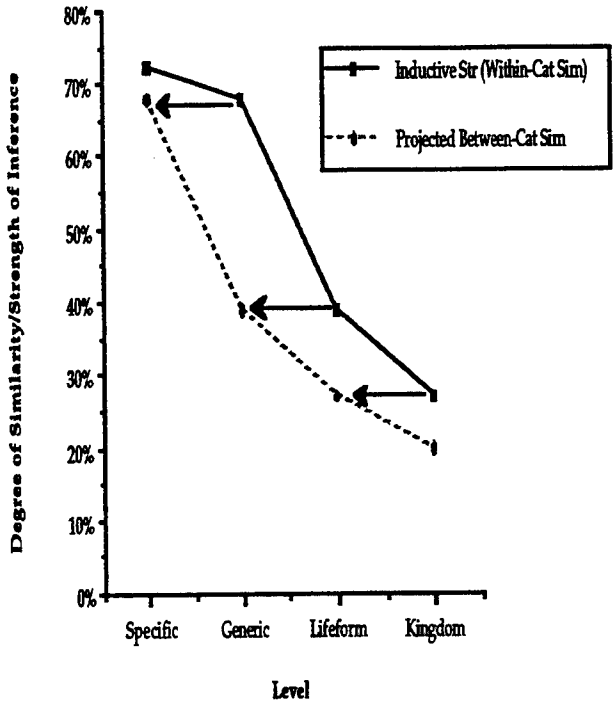


Figure 2b

Although the similarity-based account cannot explain the discrepancy, it could nevertheless be the case that Roschian tasks measuring the "basic level" and our induction task are tapping different competencies. Rosch's results show that (1) the folk-generic level is not the locus of knowledge for urban Americans, and (2) the folk-generic is not the most differentiated level perceptually for Americans. In contrast to Rosch's measures, ours are neither perceptual nor knowledge-dependent. Rather, they are linguistic and expectation-dependent. Rather than having participants list features that they knew to be true of categories, we asked them to project properties chosen expressly for the purpose of being unknown. Instead of testing participants' knowledge, we have tested their expectations. Our results show that despite lack of knowledge and perceptual differentiation, Americans expect that the folk-generic level is most useful for inductive inference.

Why would the most inductively useful level be differ from the "basic" level as indicated by knowledge and perception? Implicitly learning the logic of nomenclatural patterns of inclusion (a red oak is a kind of oak), and explicitly learning inclusion relations of folk-generics under life-forms (an oak is a kind of tree) may be enough to set up a semblance of a folk-biological taxonomy and to flag the folk-generic level despite little experience with members of the categories. But even very limited experience with a subset of biological kinds may be sufficient to fix the locus at the folk-generic level. This may be analogous to what Shipley (1993) refers to an overhypothesis (see also Goodman, 1955). For example, people may know that members of familiar folk-generics are alike in predictable ways, and different from other familiar folk-generics in predictable ways. This knowledge may set up an assumption about novel kinds of animals. For example, if people are told about some unfamiliar mammal, the quagga, they are likely to expect that quaggas will differ from other mammals and be similar to each other in external appearance, in some internal properties, and in a variety of other ways.

The American college students in our study presumably have little first-hand experience of or dependence on the natural world, but they know that elms, maples and oaks are kinds of trees, and expect (perhaps on the basis of nomenclature alone) that red oaks are kinds of oaks, and that northern red oaks are kinds of red oaks. But they may not even know enough to tell the difference between an oak and an elm. There may be universally fixed ranks in folkbiological classification (Berlin, 1992), and the folk-generic level may carry with it the expectation of an inherent physical nature, or underlying essence (Atran, 1987; Medin and Ortony, 1989). We propose that this relatively sophisticated knowledge of nomenclatural patterns in folkbiological taxonomy, coupled with an expectation that (folk-generic?) categories have an essence, would lead to our observed results. Americans may know more features and perceptual affinities at the more abstract level of tree and fish. Nevertheless, they may expect the strongest clusters of properties to cohere at the folk-generic level.

The second question raised by our findings is why we found the same pattern among folkbiologically naive Americans and folkbiologically sophisticated Maya? We have argued that, for Americans, linguistic cues may target an expectation of essence at the folk-generic rank for biological kinds despite lack of familiarity with the target. In situations where folk are likely to be well-acquainted with local living kinds, as in "traditional" societies like the Maya, the weight of ethnobiological evidence argues that perceptual cues converge on folk-generics as being psychologically basic. In fact, the coincidence of domain-general heuristics with domain- specific presumptions may well represent the default case for human understanding of living kinds under normal (evolutionarily-attuned) environmental conditions. This speculation remains to be tested and it certainly is possible that people from traditional societies will perform the same as undergraduates on the sorts of perceptual tasks used by Rosch and her associates. But on the other hand, there is some evidence that the basic level may change with expertise (Tanaka and Taylor, 1991) and people in traditional societies may be experts relative to American undergraduates.

Summary and Conclusions

To what degree do folkbiological conceptual systems reflect universal patterns of feature covariation in the world, or universal habits of mind, and to what degree do they reflect specific goals and needs of the categorizer? In this paper we have focused on the questions of how similarity and theory structure categories and the basis of category privilege. Let's first review differences associated with type of expertise.

Differences in taxonomies. One difference between the groups was in the structure of and justifications for their taxonomies, a difference that may reflect the different purposes for which each group classifies trees. Taxonomists basically reproduced scientific taxonomy. Parks subgroup taxonomies tended to be justified on the basis of morphological and sometimes utilitarian concerns but overall correlated moderately well with science. Finally, the landscapers' used the widest variety of justifications, including many conjunctive justifications ("large native specimen trees"). These findings are consistent with a "taxonomy" based on a number of semi-independent utilitarian dimensions being imposed over the set of trees.

Overall, parks experts can be characterized in terms of heavy emphasis on morphological properties seamlessly integrated with moderate attention to goal-related utilitarian factors and reflecting an influence of common names. The same features, weighted differently (e.g., parks personnel tend to weight features like wood strength and leaf shape more heavily than taxonomists), appear to drive the classification systems of both the taxonomist and

parks subgroups. Likewise, both groups "stuck to their (folk)taxonomic guns" in reasoning, drawing inferences which, for the most part, agreed with their sortings. The landscapers, in contrast, violated the correlational structure of the stimulus set to such a degree that structure-based accounts would be severely strained. No stretching or shrinking of a common space would transform their consensual clustering to conform to that of either of the other two groups (save for the degenerate case of a set of dimensions with zero weighting for various subgroups). Barsalou's analysis of goal-derived categories and his conjecture that frequently used goal-derived categories may become well-established in long-term memory (Barsalou, 1982, 1983, 1991), predicts the pattern of landscape sorting in striking detail.

Indeed, the mismatch of landscapers' sorting and reasoning data provides further support for the above analysis. Assume that the landscapers' consensual taxonomy reflects goal-derived, special-purpose categories rather than general-purpose categories expected to maximally capture similarities in the world. If so, there is no reason to expect these goal-derived categories to support inferences about any properties except those related to the goals. The reproductive, disease, and physiological properties used in Experiment 2 have little relation to the functional, utilitarian goals that seemed to drive the landscapers' sortings from Experiment 1. Therefore, landscapers sensibly abandoned their salient but inductively less useful (in this case, at least) consensual taxonomy in favor of a general-purpose taxonomy (either science or an organization approximating that of the parks workers).

In sum, the first experiment revealed an influence of goals on the structuring of categories and no absolute privilege to the genus level. The second experiment was more supportive of the genus level as privileged but, in the case of the parks subgroup, only if the genus level was marked by language. This suggests that perceived structure is not simply determined by how things are in the world but rather that perceived structure is guided by language. Categorization models can and should be applied to these observations.

The third experiment demonstrated a remarkable convergence across cultures with respect to rank and privilege. Specifically, the folkgeneric (roughly genus) level appeared to be psychologically privileged for both knowledgeable Maya informants and undergraduates who have much less direct contact with biological kinds. Note that these findings also converge with our observations on tree experts whose reasoning was strongly influenced by genus-level matches, particularly when shared genus was marked by language. Although on the surface these results appear to support a "structure in the world" framework, we believe that they are most consistent with the idea that they are driven by language and expectations. As we have seen, the idea of perceptual similarity does not seem capable of simultaneously explaining Rosch's categorization results and our reasoning results. But it's equally clear that ideas about presumption of essence and expectations remain speculative in the absence of a body of evidence associated with more precise predictions. We also need cross-cultural measures of privilege that are more perceptually-oriented.

Although findings that hold across tree experts, undergraduates and Guatemala Maya may have the air of universality, that inductive leap may be premature. For example, we are currently studying ecologists and the structure of biological communities seems to have a lot of action at the species level. For example, the Burr Oak plays a very different role from that of other oaks found in the Savannah. It may be that ecologists will privilege the species level rather than genus. One should also note that horticulturists focus attention on the varietal level and perhaps their inductions would be even more narrowly constrained.

At this point one might even worry that the key notion of a basic, privileged level is suspect. It is not a large step from the idea of multiple privileged levels or different levels being privileged for different conceptual functions to the conclusion that there the very idea of basic or privileged level is not useful. We believe, however, that despair should not be the response to the first indication that categorization principles might not be as parsimonious as one might wish. Indeed there are fascinating questions that remain to be asked and answered concerning expertise, development, and cultural similarities and differences. For example, there is clear evidence that infants learning names for things have clear expectations concerning the extensions of category labels (see Waxman, 1994 for a review). But the question remains whether American infants are surprised to find that things as different as robins and Sparrows are both typically called "birds" or are Maya infants surprised that things as similar as Robins and Sparrows have different names? Or are infants from both cultures ready for either possibility? When we are able to answer questions such as this we will be in a much better position to understand the roles of similarity and theories in the development and expression of conceptual behavior.

References

Anderson, J. R. 1990. *The Adaptive Character of Thought.* Hillsdale, NJ: Erlbaum.

Anderson, J. R. 1991. The adaptive nature of human categorization. *Psychological Review 98*: 409-429.

Atran, S. 1987. Ordinary constraints on the semantics of living kinds: A commonsense alternative to recent treatments of natural-object terms. *Mind and Language 2*: 27-63.

Barsalou, L. W. 1982. Context-independent and context-dependent information in concepts. *Memory and Cognition 10*: 82-93.

Barsalou, L. W. 1983. Adhoc categories. *Memory & Cognition 11*: 211-227.

Barsalou, L. W. 1985. Ideals, central tendency, and frequency of instantiation as determinants of graded structure of categories. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 11: 629-654.

Barsalou, L. W. 1991. Deriving categories to achieve goals. In G. H. Bower (Ed.), *The Psychology of Learning and Motivation: Advances in research and theory, vol. 27.* New York: Academic Press.

Barsalou, L. W. 1992. Flexibility, structure, and linguistic vagary in concepts: Manifestations of a compositional system of perceptual symbols. In A. C. Collins, S. E. Gathercole, M. A. Conway, & P. E. M. Morris (Eds.), *Theories of memories*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Barsalou, L. W. 1993. Structure, flexibility, and linguistic vagary in concepts: Manifestations of a compositional system of perceptual symbols. In A. C. Collins, S. E. Gathercole, & M. A. Conway (Eds.), *Theories of Memory*. London: Erlbaum.

Barsalou, L. W., & Ross, B. H. 1986. The roles of automatic and strategic processing in sensitivity to superordinate and property frequency. *Journal of Experimental Psychology: Learning, Memory, & Cognition 12*: 116-134.

Berlin, B. 1992. *Ethnobiological classification: Principles of categorization of plants and animals in traditional societies*. Princeton, NJ: Princeton University Press.

Berlin, B., Breedlove, D. E. & Raven, P. H. 1966. Folk taxonomies and biological classification. *Science 154*: 273-275.

Berlin, B., Breedlove, D. E. & Raven, P. H. 1973. General principles of classification and nomenclature in folk biology. *American Anthropologist 75*: 214-242.

Billman, D. 1992. Modeling category learning and use: Representation and processing. In B. Burns ed. *Percepts, concepts, and categories: The representation and processing of information.* New York: Elsevier.

Billman, D., & Knutson, J. F. 1996. Unsupervised concept learning and value systematicity: A complex whole aids learning the parts. *Journal of Experimental Psychology: Learning, Memory, and Cognition 22*: 458-475.

Boster, J. S. 1991. The information economy model applied to biological similarity judgment. In L. Resnik, J. Levine, & S. Teasley eds. *Perspectives on socially shared cognition*. Washington, DC: American Psychological Association.

Boster, J. S., & D'Andrade, R. 1989. Natural and human sources of cross-cultural agreement in ornithological classification. *American Anthropologist 88*:569-583.

Boster, J. S., & Johnson, J. C. 1989. Form or function: A comparison of expert and novice judgments of similarity among fish. *American Anthropologist 91*: 866-889.

Brooks, L. R., Norman, G. R., & Allen, S. W. 1991. Role of specific similarity in a medical diagnosis task. *Journal of Experimental Psychology: General 120*: 278-287.

Burns, B. ed. 1992. *Percepts, concepts, and categories: The representation and processing of information.* New York: Elsevier.

Carey, S. 1985. *Conceptual change in childhood.* Cambridge, MA: Bradford Books.

Carey, S., & Spelke, E. 1994. Domain-specific knowledge and conceptual change. In L. A. Hirschfeld & S. A. Gelman eds. *Mapping the mind.* Cambridge: Cambridge University Press.

Chi, M. T. H. 1983. Knowledge-derived categorization in young children. In D. R. Rogers & J. A. Sloboda eds. *The acquisition of symbolic skill*. New York: Plenum Press.

Chi, M. T. H., Feltovich, P., & Glaser, R. 1981. Categorization and representation of physics problems by experts and novices. *Cognitive Science 5*: 121-152.

Chi, M. T. H., Hutchinson, J. E., & Robin, A. F. 1989. How inferences about novel domain-related concepts can be constrained by structured knowledge. *Merrill-Palmer Quarterly 35(1)*: 27-62.

Choi, S., McDaniel, M. A., & Busemeyer, J. R. 1993. Incorporating prior biases in network models of conceptual rule learning. *Memory & Cognition 21*: 413-423.

Clapper, J. P., & Bower, G. H. 1994. Category invention in unsupervised learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition20*: 443-460.

Coley, J. D., Medin, D. L., Atran, S., & Lynch, E. (submitted for publication). Does privilege have its rank? Folkbiological taxonomy and induction in two cultures.

Corter, J. E., & Gluck, M. A. 1992. Explaining basic categories: Feature predictability and information. *Psychological Bulletin 111*: 291-303.

Ellen, R. 1993. *The cultural relations of classification.* Cambridge, MA: Cambridge University Press.

Estes, W. K. 1994. *Classification and Cognition* 161-163. Oxford: Oxford University Press.

Fisher, D. H., Pazzani, M. J., & Langley, P. eds. 1991. *Concept formation: Knowledge and experience in unsupervised learning*. San Mateo, CA: Morgan Kaufman.

Fisher, D., & Yoo, J. P. 1993. Categorization, concept learning, and problem solving: A unifying view. In G. V. Nakamura, R. Taraban, & D. L. Medin eds. *The Psychology of Learning and Motivation: Categorization by*

humans and machines, vol. 29. San Diego: Academic Press, Inc.

Gelman, S. A., & Coley, J. D. 1991. Language and categorization: The acquisition of natural kind terms. In J. P. Byrnes & S. A. Gelman eds. *Perspectives on language and thought: Interrelations in development* (pp. 146-196). Cambridge: Cambridge University Press.

Gelman, S. A., & Wellman, H. M. 1991. Insides and essences: Early understandings of the nonobvious. *Cognition 38*: 213-244.

Gentner, D., & Markman, A. B. 1994. Structural alignment in comparison: No difference without similarity. *Psychology Science 5*: 152-158.

Gobbo, C. & Chi, M. T. H. 1986. How knowledge is structured and used by expert and novice children. *Cognitive Development 1*: 221-237.

Goodman, N. 1955. *Fact, fiction and forecast.* Cambridge, MA: Harvard University Press.

Heit, E. 1993. Modeling the effects of expectations on recognition memory. *Psychological Science 4*: 244-252.

Heit, E. (in press). Models of the effects of prior knowledge on category learning. *Journal of Experimental Psychology: Learning , Memory, and Cognition*.

Hirschfeld, L. A., & Gelman, S. A. eds. 1994. *Mapping the Mind: Domain specificity in cognition and culture*. New York: Cambridge University Press.

Honeck, R. P., Firment, M., & Case, T. J. S. 1987. Expertise and categorization. *Bulletin of the Psychonomic Society 25*: 431-434.

Johnson, K. E. 1992. The effect of expertise on hierarchical systems of categorization. Unpublished doctoral dissertation, Emory University, Atlanta, GA.

Johnson, K. E., & Mervis, C. B. 1994. Microgenetic analysis of first steps in children's acquisition of expertise on shorebirds. *Developmental Psychology 30*: 418-435.

Johnson, K. E., Mervis, C. B., & Boster, J. S. 1992. Developmental changes within the structure of the mammal domain. *Developmental Psychology 28*: 74-83.

Jones, G. V. 1983. Identifying basic categories. *Psychological Bulletin 94*: 423-428.

Keil, F. C. 1989. *Concepts, kinds, and cognitive development*. Cambridge, MA: MIT Press.

Keil, F. C. 1994. The birth and nurturance of concepts by domains: The origins of concepts of living things. In L. A. Hirschfeld & S. A. Gelman eds., *Mapping the mind.* Cambridge: Cambridge University Press.

Komatsu, L. K. 1992. Recent views of conceptual structure. *Psychological Bulletin 112*: 500-526.

Lassaline, M. E., Wisniewski, E. J., & Medin, D. L. 1992. Basic levels in artificial and natural categories: Are all basic levels created equal? In B. Burns ed. *Percepts, concepts, and categories: The representation and processing of information.* New York: Elsevier.

Malt, B. C. 1995. Category coherence in cross-cultural perspective. *Cognitive Psychology 29*: 85-148.

Mandler, J. M., Bauer, P. J., & McDonough, L. 1991. Separating the sheep from the goats: Differentiating global categories. *Cognitive Psychology 23*: 263-298.

Medin, D. L., & Heit, E. J. (in press). Categorization. In D. Rumelhart & B. Martin eds. *Handbook of cognition and perception*. Hillsdale, NJ: Erlbaum.

Medin, D. L., & Ortony, A. 1989. Psychological essentialism. In S. Vosniadou & A. Ortony eds. *Similarity and Analogical Reasoning*.

Medin, D. L. & Schaffer, M. M. 1978. Context theory of classification learning. *Psychological Review 85*: 207-238.

Medin, D. L., Goldstone, R. L., & Gentner, D. 1993. Respects for similarity. *Psychological Review 100*: 254-278.

Medin, D. L., Lynch, E. B., Coley, J. D., Atran, S. (in press). Categorization and reasoning among tree experts: Do all roads lead to Rome? *Cognitive Psychology*.

Mervis, C. B., Johnson, K. E., & Scott, P. 1993. Perceptual knowledge, conceptual knowledge, and expertise: *Comment on Jones and Smith. Cognitive Development 8*: 149-155.

Michalski, R. S. 1989. Two-tiered concept meaning, inferential matching, and conceptual cohesiveness. In S. Vosniadou & A. Ortony eds. *Similarity and analogical meaning*. New York: Cambridge University press.

Michalski, R. S., & Tecuci, G. Eds. 1994. *Machine learning: A multistrategy approach*, vol. 4. San Francisco: Morgan Kaufman.

Mooney, R. J. 1993. Integrating theory and data in category learning. In G. V. Nakamura, R. Taraban, & D. L. Medin eds. *The Psychology of Learning and Motivation: Categorization by humans and machines*, vol. 29. San Diego: Academic Press, Inc.

Murphy, G. L. 1982. Cue validity and levels of categorization. *Psychological Bulletin 91*: 174-177.

Murphy, G. L. 1991. Parts in object concepts: Experiments with artificial categories. *Memory & Cognition, 19*: 423-438.

Murphy, G. L. 1993a. Theories and concept formation. In I. V. Mechelen, J. Hampton, R. Michalski, & P. Theuns eds. *Categories and concepts: Theoretical views and inductive data analysis*. London: Academic Press.

Murphy, G. L. 1993b. A rational theory of concepts. In G. V. Nakamura, R. Taraban, & D. L. Medin eds. *The Psychology of Learning and Motivation: Categorization by humans and machines*, vol. 29. San Diego: Academic Press, Inc.

Murphy, G. L., & Medin, D. L., 1985. The role of theories in conceptual coherence. *Psychological Review* 92: 289-316.

Myers, J. L., Lohmeier, J. H., & Well, A. D. 1994. Modeling probabilistic categorization data: Exemplar meaning and connectionist nets. *Psychological Science 5:* 83-89.

Nakamura, G. V., Taraban, R., & Medin, D. L. eds. 1993 *The Psychology of Learning and Motivation: Categorization by humans and machines*, vol. 29. San Diego: Academic Press, Inc.

Nosofsky, R. M. 1992. Exemplar-based approach to relating categorization, identification, and recognition. In F. G. Ashby ed. *Multidimensional models of perception and cognition.* Hillsdale, NJ: Erlbaum.

Nosofsky, R. M., Gluck, M. A., Palmeri, T. S., McKinley, S. C., & Glauthier, P. 1994. Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland, and Jenkins 1961. *Memory & Cognition 22*: 352-369.

Palmer, C. F., Jones, R. K., Hennessy, B. L., Unze, M. G., & Pick, A. D. 1989. How is a trumpet known? The "basic object level" concept and perception of musical instruments. *American Journal of Psychology 102*: 17-37.

Pazzani, M. J. 1991. Influence of prior knowledge on concept acquisition: Experimental and computational results. *Journal of Experimental Psychology: Learning, Memory, and Cognition 17*, 416-432.

Rips, L. J. 1975. Inductive judgements about natural categories. *Journal of Verbal Learning and Verbal Behavior 14*: 665-681.

Rips, L. J., & Collins, A. 1993. Categories and resemblance. *Journal of Experimental Psychology: General 122*: 468-486.

Rosch, E., Mervis, C. B., Gray, W., Johnson, D., & Boyes-Braem, P. 1976. Basic objects in natural categories. *Cognitive Psychology 8*: 573-605.

Shipley, E. F. 1993. Categories, hierarchies, and induction. In D. L. Medin ed. *The Psychology of Learning and Motivation, Vol. 30*. New York: Academic Press.

Smith, L. B., & Heise, D. 1992. Perceptual similarity and conceptual structure. In B. Burns ed *Percepts, concepts, and categories: The representation and processing of information.* New York: Elsevier.

Tanaka, J. W., & Taylor, M. 1991. Object categories and expertise: Is the basic level in the eye of the beholder? *Cognitive Psychology 23*: 457-482.

VanMechelen, I., Hampton, J., Michalski, R., & Theuns, P. eds. 1993. *Categories and Concepts: Theoretical Views and Inductive Data Analysis*. London: Academic Press.

Waldman, M. R., & Holyoak, K. J. 1992. Predictive and diagnostic learning within causal models: Asymmetries in cue competition. *Journal of Experimental Psychology: General 121*: 222-236.

Walker, S. J. 1992. Supernatural beliefs, natural kinds, and conceptual structure. *Memory & Cognition 20*: 655-662.

Waxman, S. R. 1994. The development of an appreciation of specific linkages between linguistic and conceptual organization. *Lingua 92*:229-257.

Wisniewski, E. J., & Medin, D. L. 1994. On the interaction of theory and data in concept learning. *Cognitive Science 18*:221-281.

Wisniewski, E. J., & Medin, D. L. 1994a The fiction and nonfiction of features. In R. S. Michalski & G. Tecuci eds. *Machine Learning Vol. 4*, San Mateo, CA: Morgan Kaufmann.

Wisniewski, E. J., & Medin, D. L. 1994b. On the interaction of theory and data in concept learning. *Cognitive Science 18*: 221-281.

Coevolution Learning:
Synergistic Evolution of Learning Agents and Problem Representations

Lawrence Hunter

National Library of Medicine
8600 Rockville Pike
Bethesda, MD 20894
hunter@nlm.nih.gov

Abstract

This paper describes exploratory work inspired by a recent mathematical model of genetic and cultural coevolution. In this work, a simulator implements two independent evolutionary competitions which act simultaneously on a diverse population of learning agents: one competition searches the space of free parameters of the learning agents, and the other searches the space of input representations used to characterize the training data. The simulations interact with each other indirectly, both effecting the fitness (and hence reproductive success) of agents in the population. This framework simultaneously addresses several open problems in machine learning: selection of representation, integration of multiple heterogeneous learning methods into a single system, and the automated selection of learning bias appropriate for a particular problem.

Introduction

One clear lesson of machine learning research is that problem representation is crucial to the success of all inference methods (see, e.g. (Dietterich, 1989; Rendell & Cho, 1990; Rendell & Ragavan, 1993)). However, it is generally the case that the choice of problem representation is a task done by a human experimenter, rather than by an automated system . Also significant in the generalization performance of machine learning systems is the selection of the inference method's free parameter values (e.g. the number of hidden nodes in an artificial neural network, or the pruning severity in a decision tree induction system), which is also a task generally accomplished by human "learning engineers" rather than by the automated systems themselves.

The effectiveness of input representations and free parameter values are mutually dependent. For example, the appropriate number of hidden nodes for an artificial neural network depends crucially on the number and semantics of the input nodes. This paper describes exploratory work on a method for simultaneously searching the spaces of representations and parameter settings, using as inspiration a recent mathematical model of the coevolution of genetics and culture from anthropology (Durham, 1991). An important additional feature of this work is that is provides a simple, seamless and effective way of synergistically combining multiple inference methods in an integrated and extensible framework. When only a single inference method is used, this framework reduces to a variant on constructive induction. However, with multiple and diverse learning agents, the system is able to generate and exploit synergies between the methods and achieve results that can be superior to any of the individual methods acting alone.

The work presented here fits into a growing body of research on these issues. The importance of bringing learning engineering tasks within the purview of the theory and practice of automated systems themselves was described at length in (Schank, et al., 1986). Some effective computational methods for aspects of this task have been reported recently. (Kohavi & John, 1995,) describes an automated method for searching through the space of possible parameter values for C4.5. Genetic algorithms have also been used to search the space of parameter values for artificial neural networks (Yao, 1993). There has also been recent work on selecting appropriate ("relevant") subsets of features from a superset of possible features for input representation in machine learning ((Langley, 1994) is a review of 14 such approaches), as well as a long history of work on constructive induction ((Wnek & Michalski, 1994) includes a review, but see also (Wisniewski & Medin, 1994) for critical analysis of this work. This paper describes an exploratory approach that appears to have promise in addressing these issues in an integrated way.

Background

The idea of coevolution learning is to use a kind of genetic algorithm to search the space of values for the free parameters for a set of learning agents, and to use a system

metaphorically based on recent anthropological theories of cultural evolution to search through the space of possible input representations for the learning agents. This section provides some brief background on these ideas.

Evolutionary algorithms are a weak search method related to, although clearly distinguishable from, other weak methods (e.g. beam search). They are based on a metaphor with naturally occurring genetic evolution. In general, evolution has three components: inheritance, or the passage of traits from one individual to another; variation, of the generation of novel traits or combinations of traits; and selection, a competition between individuals based on their traits that effects the probability that an individual will have descendants that inherit from it. There are many ways of building computational models that evolve, including genetic algorithms, genetic programming and evolutionary strategies; see (Angeline, 1993) for an excellent survey and analysis of the approach.

Anthropological theories of culture have been converging for some time on what are now known as "ideational" theories. They hold that culture "consists of shared ideational phenomena (values, beliefs, ideas, and the like) in the minds of human beings. [They refer] to a body or 'pool' of information that is both public (socially shared) and prescriptive (in the sense of actually or potentially guiding behavior)." (Durham, 1991), p. 3. Note that this view is different from one that says culture *is* some particular set of concrete behavior patterns; it instead suggests that culture is just one of the factors that shapes behavior. An individual's phenotype (here, its behavior) is influenced by its genotype, its individual psychology and its culture. Durham summarizes "the new consensus in anthropology regards culture as systems of symbolically encoded conceptual phenomena that are socially and historically transmitted within and between populations" (p. 8-9). This historically rooted, social transmission of ideational elements can be analyzed as an evolutionary process. The characterization of that evolutionary process (and its relationship to genetic evolution) is the subject of Durham's book, and also the subject of a great deal other research dating back at least one hundred years (much of which is surveyed by Durham).

There are several significant differences between cultural and genetic evolution. Cultural traits are transmitted differently than genetic ones in various ways. It is possible to transfer cultural traits to other members of your current generation, or even to your cultural "parents." It is also possible for one individual to pass cultural traits to very many more others than he or she could genetically. The selection pressure in the competition among cultural entities is not based on their reproductive fitness as with genetic evolution, but on the decisions of individuals to adopt them, either through preference or imposition. And most importantly for the purposes of this work, cultural evolution involves different sources of variation than genetic evolution. Rather than relying on mutation or sexual recombination of genomes to provide novel genetic variants, culture relies on individual's own discovery and synthesis as the source of novelty. By providing a computational model in which individuals are able to learn from their experiences and share what they have learned, it becomes possible to simulate a cultural kind of evolution.

A key aspect of any model of cultural evolution is the specification of the smallest unit of information that is transmitted from one agent to another during cultural transmission, christened the "meme" by Richard Dawkins. Although there is a great deal of argument over what memes are (ideas, symbols, thoughts, rules patterns, values, principles, postulates, concepts, essences and premises have all been suggested), and a great deal of theoretical analysis describing how memes compete, are transformed, interact with genes, etc., I am aware of no attempts to operationalize the term so that it would be possible to build computational simulations of populations of memes.

A meme must play several roles in a simulation of cultural inheritance. First, it must be able to have an effect on the behavior of the individuals in simulation. Second, individuals must be able to create new memes as a result of their experiences (e.g. by innovation, discovery or synthesis). Third, it must be possible for other individuals in the simulation to evaluate memes to determine whether or not they will adopt a particular meme for its own use. In the sections below, I will show how the input representation used by a machine learning system can be used to meet these requirements.

A Formal Definition of Coevolution Learning

A coevolution learning system functions by evolving a population of learning agents. The population is defined by a classification task **T**, a set of classified examples of that task expressed in a primitive representation $\mathbf{E_p}$, a fitness function for agents f_A , and a set of learning agents **A**:

$$P_{coev} \equiv \{T, E_p, f_A, A\}$$

A fitness function for agents maps a member of the set **A** to a real number between 0 and 1. For convenience, it is useful to define $\mathbf{P_L}$ to be the subset of $\mathbf{P_{coev}}$ where all the agents use learning method **L** (see below).

Each learning agent **A** is defined by a learning method **L**, a vector of parameter values **v**, a fitness function for memes f_m, and an ordered set of problem representation transformations **R**:

$$A_i \equiv \{L, v, f_m, R\}$$

The vector of parameter values may have different length for different learning methods. Each member of the set of problem representation transformations $\mathbf{R_i}$ is a mapping from an example ($\mathbf{e_p} \in \mathbf{E_p}$) to a value which is a legal element of an input representation for **L** (e.g. a real number, nominal value, bit, horn clause, etc.). The individual mappings are called *memes*. The ordered set of memes ($\mathbf{R_i}$) is called the agent's *memome*, and the vector of parameter values is called its *genome*. The fitness function for memes $\mathbf{f_m}$ is a function that maps a member of the set $\mathbf{R_i}$ to a real number between 0 and 1.

A transformed example e_i is the result of sequentially applying each of the problem representation transformations $r_i \in R_i$ to an example $e_p \in E_p$. The application of the set of problem representation transformations to the set of examples in primitive representation results in a set of transformed examples, which is called E_i. When $E_i = E_p$, the transformation is said to be the identity.

Given a population as defined above, the process of coevolution is defined in the pseudocode in figure 1. The creation of the next generation of agents is a minor variant on traditional genetic algorithms. Instead of having the genome be "bits" it is a vector of parameter values. The crossover, mutation and fitness proportional reproduction functions are all identical with the related operations on bit vectors in genetic algorithms. It is, of course, possible to transform parameter vectors into bitstring representations themselves, if it were desirable. In addition to using parameter vectors instead of bits, the other difference is that each subpopulation using a particular learning method (the P_L's) has its own type of parameter vector, since the free parameters and their legal values vary among learning methods. These parameter vectors may be of different sizes, so crossover can only be applied with members of the same subpopulation.

The main difference between coevolution learning and other evolutionary methods derives from the creation and exchange of memes. The meme creation process takes the output of learning agents that have been applied to a particular problem, and identifies combinations of the input features that the learner determined were relevant in making the desired distinction. The process of parsing output and creating new memes is specific to each learning method. For example, a program that learns rules from examples might create new memes from the left hand sides of each of the induced rules. Or, a program that learned weights in a neural network might create new memes that were the weighted sum of the inputs to each of its hidden nodes (perhaps thresholded to remove marginal contributions). Specific meme generation methods are discussed in more detail in the implementation section, below.

Initialize the population with random legal parameter vectors and the identity representation transformation.

* **Determine the phenotype** of each agent *i* by applying learning algorithm L_i to transformed examples E_i using parameter values v_i using K-way cross-validation. The phenotype is an ordered set of: the output of the learner's cross-validation runs, the cross validation accuracies and the learning time.

Determine the fitness of each agent by applying f_A to the phenotype of each agent.

Create new memes by parsing the output in each learner's phenotype and extracting important feature combinations. The union of all memes generated by all members of the population is called the meme pool.

Exchange memes. For each agent i, apply its meme fitness function F_m to elements of the meme pool. Select the agent's target number of memes (a free parameter) from the meme pool with a probability proportional to the fitness of the memes.

Repeat from * meme-transfers-per-generation times

Determine phenotype of the agents with their new memes

Determine the fitness of the agents

Create the next generation of agents by mutation, crossover and fitness proportional reproduction. Agents whose genomes are mutated keep their memomes intact. For each pair of agents whose genomes are crossed over to create a new pair of agents, the memomes of the parents are arbitrarily assigned to the offspring.

Repeat from * until a member of the population can solve the problem

Figure 1: Pseudocode of coevolution algorithm

An Implementation of a Coevolution Learner

COEV (short for "the beginning of coevolution") is a simple implementation of the coevolution learning framework, written in Carnegie Mellon Common Lisp 17f and the PCL Common Lisp Object System, running on a Silicon Graphics Indigo^2 workstation. The current implementation includes the C4.5 decision tree induction system and C4.5rules rule extraction tool (Quinlan, 1991), the LFC++ constructive induction program (Rendell & Ragavan, 1993; Vilalta, 1993) and the conjugate gradient descent trained feedforward neural network (CG) from the UTS neural network simulation package (van Camp, 1994). Each learning method is associated with an object class which defines how to execute it, how to parse the results returned, and what the vector of free parameters is for that type of learner.

Most of the documented parameters for each of type of learning program is included in the free parameter vector for that system, along with a specification of either an upper and lower bound for the parameter's values or a list of possible parameter values. For example, the parameter vector for the UTS conjugate gradient descent learner includes the number of hidden nodes in the network, the maximum number of iterations before halting, the output tolerance (specifying how close to 1 or 0 an output has to be to count as true or false), a flag for whether or not to use competitive learning on the output, two nominal parameters specifying the kind of line search and direction finding method to use, and three parameters specifying the maximum number of function evaluations per iteration, the minimum function reduction necessary to continue the search and the maximum slope ratio to continue the search. Theses parameters are explained more fully in the documentation available with the source code.

Each learner must also have a method for extracting new features from its output. LFC++, like other constructive induction programs, specifically defines new features as part of its output. For C4.5, the output of the C4.5rules tool was parsed so that each left hand side of each rule was reified into a new feature definition. For the neural network learner, a new feature was created for each hidden node defined by the weighted sum of the inputs to that node, with any input whose contribution to that sum was less than the threshold of the node divided by the number of inputs removed from the sum. These extracted features are added to the meme pool.

Memes (feature combinations) must be assigned fitnesses by agents. It is possible for each agent to have its own method of determining meme fitness. However, a simplified method is used in COEV. Whenever a program generates a new meme, its fitness is defined to be the average "importance" of the feature combination, weighted by the accuracy of the learners that used the meme. In Durham's terms, the inventor of the new meme "imposes" its belief in its value on others. In addition, receivers of memes have a parameter which determines how much weight they put on memes they generate themselves versus memes generated by others. This mechanism could be straightforwardly generalized to weight the fitnesses of memes generated by different classes of agents differently. Importance is defined differently for different learning methods. For C4.5 and LFC++, the importance of a feature combination is the ratio of correctly classified examples that triggered the rule containing the feature to the total number of occurrences of the feature. So, for example, if "A and not B" is a feature extracted from a single C4.5 learner that had a 80% cross-validation accuracy, and 9 of the 10 examples the feature appeared in were classified correctly by the rule containing the feature, it's importance would be 9/10 and its fitness would be 0.9 * 0.8 = 0.72. For UTS, the importance of a feature is the absolute value of the ratio of the weight from the hidden node that defined the feature to the threshold of the output unit. If there is more than one output unit, it is the maximum ratio for any of the output units. Features that play a significant role in learners that are accurate have higher fitness than features that do not play as much of a role or appear in learners that are less accurate. It is possible for a feature to have relatively low prevalence in the population and yet still be important if it tends to generate correct answers when it is used.

In addition to defining the fitness of features, COEV must define the fitness of learners themselves for the evolution of parameter values. The fitness function for learners was selected to evolve learners that are accurate, robust and fast:

$$f(A_i) = C(A_i) - \sqrt{S(A_i)} - k\left(\frac{t_{A_i} - t_A}{S(t_A)}\right)$$

where $C(A_i)$ is the cross-validation accuracy of the agent A_i, $S(A_i)$ is the standard deviation of that accuracy, t_{A_i} is the time it took for that agent to learn, t_A is the mean execution time for that type of agent, $S(t_A)$ is the standard deviation of that mean, and k is a constant that trades off the value of accuracy versus that of learning time. Execution time is measured as the number of standard deviations from the mean learning time for that class of learners so that classes of learners with different training times can coexist in the same population. In the COEV system, k was selected to be 3 and accuracies and standard deviations are measured in percentage points. So, a learner that had a mean accuracy of 80%, a standard deviation of 9% and took 1 standard deviation less than the mean training time for that class of learner would get a fitness score of $80 - \sqrt{9} - (-1) = 78$.

Several other aspects of the framework must be specified in order to implement the system. Meme definitions must be stored in a canonical form so that it is possible to detect when two or more generated memes are in effect identical and should have their fitness scores combined. Memes in COEV are represented as boolean combinations of primitives or mathematical formula over primitives, which

can be straightforwardly compared, although the extension to first order predicate calculus would make this a more difficult problem. The number of memes that an agent uses (i.e. the dimensionality of its input representation) is treated as a free parameter of the agent, and allowed to vary from two to twice the number of primitives. In addition, agents are allowed to prefer memes of their own creation to memes created by other learners. A single parameter, ranging over [0,1] specifies the internal meme preference of each agent.

Simulation Results on a Simple Test Problem

The COEV system was applied to an artificial problem designed to be moderately difficult for the machine learning programs included in the system. The problem was to identify whether any three consecutive bits in a nine bit string were on, e.g. 011011011 is false, and 001110011 is true. This problem is analogous both to detecting a win in tic-tac-toe and to solving parity problems, which are known to be difficult for many types of learning systems. There are 512 possible examples, and the problem is simple enough so that the machine learning systems used (even the neural network) run reasonably quickly. This is important, since each program is executed a large number of times (see the computational complexity section, below).

This problem was run on a system that used only LFC++ and the CG learners. The population consisted of 10 LFC++ learners and 10 CG learners, and was run for 10 generations. Six fold cross-validation was used, and there was one meme exchange per generation. This rather modest problem therefore required 2400 learner executions, taking more than four days of R4400 CPU time. (The CPU time use was dominated by the 1200 neural network training runs.) Despite the large amount of computation required for this simple problem, it is reasonable to hope that the amount of computation required will grow slowly with more difficult problems (see the computational complexity section, below).

This simulation illustrated three significant points. First, the coevolution learning system was able to consistently improve as it evolved. The improvement was apparent in both the maximum and the average performance in the population, and in both the cross-validation accuracies of the learning agents and in their fitnesses (i.e. accuracy, speed and robustness). Figure 2 shows these results. In ten generations, the average fitness of the population climbed from 51%% to 75%, and the standard deviation fell from 21% to 12.5%. Looking at the best individual in the population shows that the maximum fitness climbed from 84% to 94%, and the maximum cross-validation accuracy climbed from 82.3% to 95%. Average execution time fell by more than 25% for both classes of learners (it fell somewhat more sharply, and in much greater absolute magnitude for the neural network learners.)

Figure 3 illustrates the separate contributions of the genetic evolution component and the memetic evolution component compared to coevolution in the performance of the best individual learning agent in the population. Consider first the purely genetic evolution of the free parameter vectors. When there is no memetic component, there is no interaction between the different classes of learners, since crossover and other sources of genetic variation apply only to a specific type of learner. Using genetic search to find the best free parameter values for a neural network is known to be slow (Yao, 1993). In this genetic-only neural network simulation, only a small improvement was found in the fifth generation. Similarly for the genetic evolution of LFC++'s free parameters, a modest difference was found in the sixth generation. Memetic evolution alone (without changing any of the free parameter values) showed a more significant effect. Because both the neural network and the constructive induction program make contributions to memetic evolution, any synergies between them will appear in the memetic-only curve (see discussion of figure 4, below). However, the steady rise of the coevolution curve, compared to the memetic-evolution only curve suggests that some adjustment of the free parameter values to match the representation changes induced by memetic transfer may have a positive effect. At the end of 10 generations, the maximum accuracy of the coevolved population is more than 5 percentage points higher than the memetic only population. However, it is worth noting that the memetic only population equaled that performance figure earlier in the simulation, and then drifted away.

Figure 4 illustrates the synergy that coevolution finds between the CG agent population and the LFC++ agent population. When run with only a single type of learner, the memetic evolution part of COEV becomes a kind of constructive induction program, where the output of the learner is feed back as an input feature in the next iteration. Since LFC++ is already a constructive induction program, it is somewhat surprising to note that LFC++ agents coevolving with each other still manage a small amount of improvement. Perhaps this is due to the fact that the appropriate features for this problem have three conjuncts, and LFC++ under most free parameter values tends to construct new features from pairs of input features. CG alone does not do very well on this problem. Even when coevolved with itself, its most accurate agent improves only from about 60% to a bit over 70% in ten generations. However, when these two learning methods are combined in a single coevolution learner, the results are clearly better than either of the methods used alone. The gap between the best performance of the pair of methods and the best single method tends to be about 5 percentage points in this simulation, and the difference tends to be larger as the population evolves.

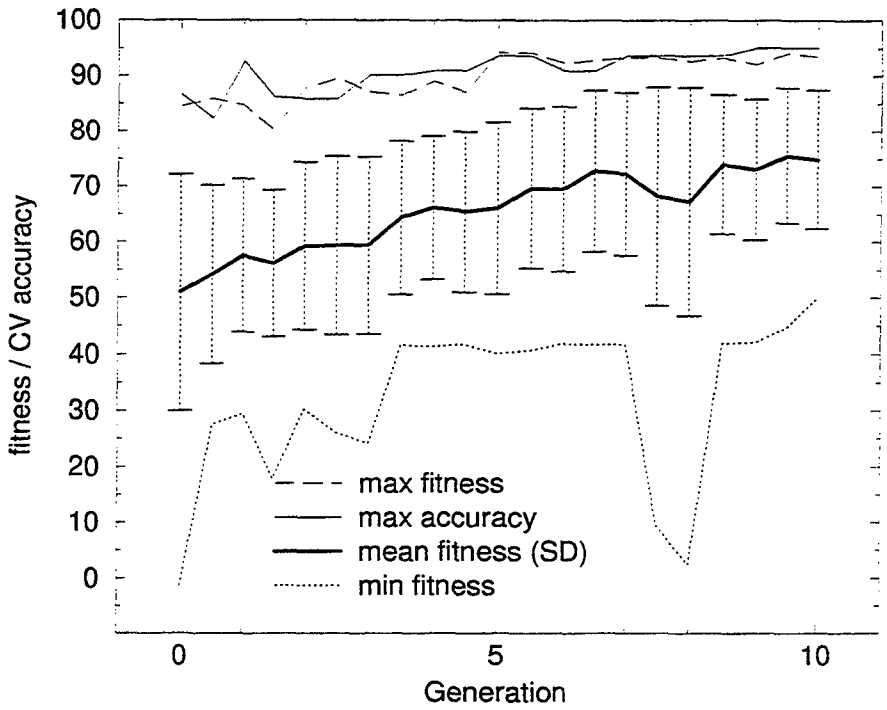


Figure 2: Fitness of best, worst and population average (with S.D.) fitness over 10 generations.

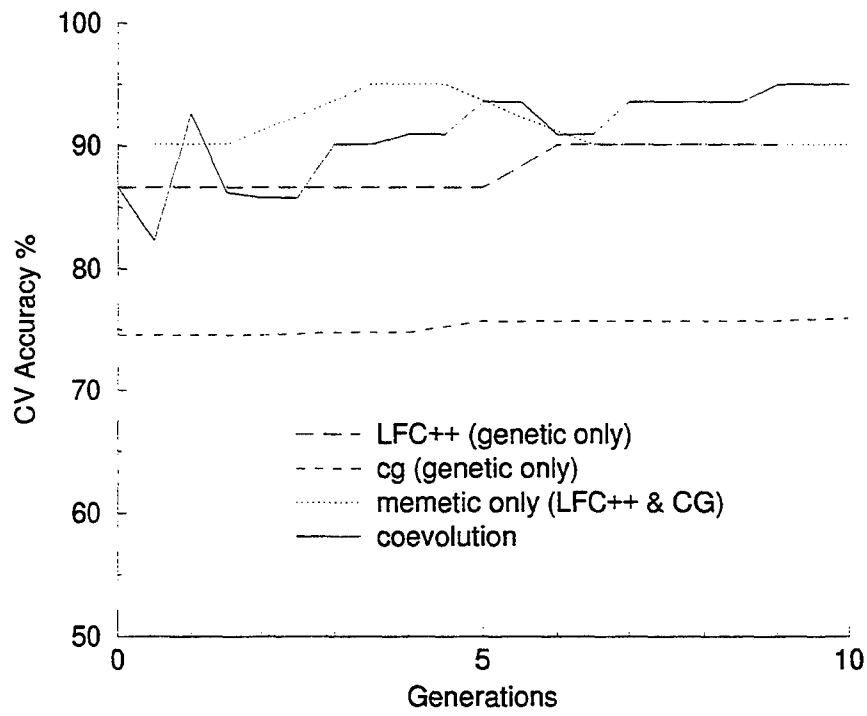


Figure 3: Genetic evolution only, memetic evolution only and coevolution

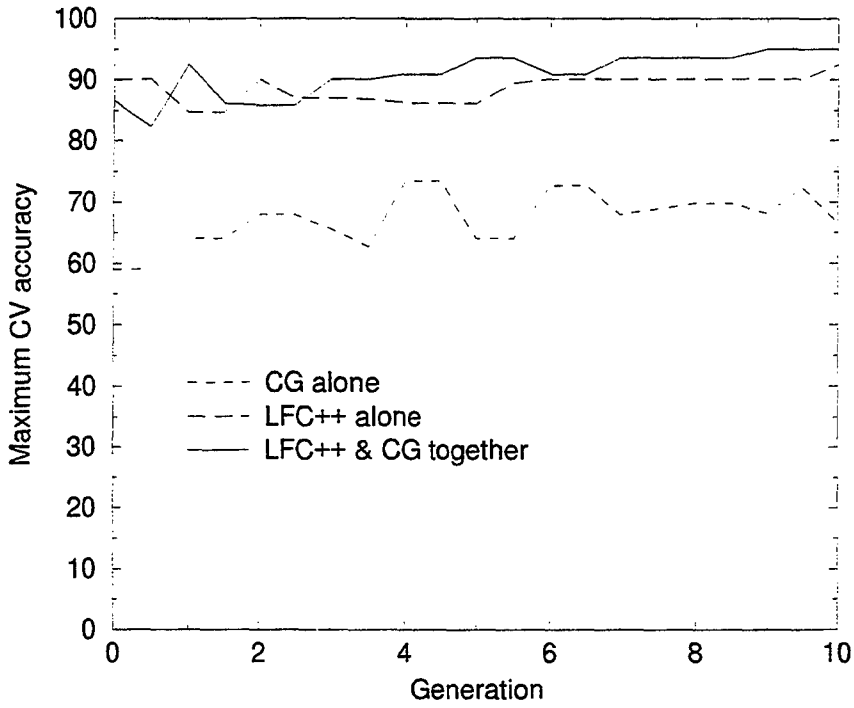


Figure 4: Synergistic interaction of the different types of learners

Why does Coevolution Appear to Work?

This is intended to be an exploratory paper, describing a novel approach to learning. It does not present any proofs nor any detailed empirical evaluations. However, it is possible to speculate about why the method appears to work, and this speculation may be useful in guiding future research in this area.

Constructive induction systems have not been able to solve problems that are a great deal more difficult than their root form of induction, even though they automatically generate relevant feature combinations to use in representation. One possible explanation is that there features are constructed with the same learning biases as the method that uses those features to build concepts, limiting the value of composing these actions. Since the learning agents in coevolution learning express a variety of biases, it may be that features discovered by one learner can more significantly facilitate learning by another learner which embodies a different bias.

Coevolution learning has a different advantage over most other forms of multistrategy learning. In these other forms, the relationships between different learning methods tend to be predefined by the learning researcher (see the introduction and many of the chapters in (Michalski & Tecuci, 1994)). For example, the KBANN system (Towell, et al., 1990) uses a domain theory to specify a neural network architecture, then trains the net and extracts a revised domain theory. Flexibility in the type of learning used and the order in which learning agents are executed may make possible important synergies that are difficult to capture with a predefined relationship.

My own earlier work was dedicated to building a planner which could select among and combine multiple learning methods based on reasoning about explicit goals for knowledge (Hunter, 1989; Hunter, 1990; Hunter, 1992) . However, first principles planning for achieving learning goals is even more difficult than planning for physical goals. Due to the large inherent uncertainty in predicting the outcome of the execution of any particular learning method, it is extremely difficult to reason about the likely outcome of any significant composition of learning actions. I also pursued the use of plan skeletons, templates and strategies, which are the usual response to the difficulties of first principles planning, but these run into the same problems in maintaining flexibility as other predefined approaches to multistrategy learning. Coevolution learning is able to flexibly combine learning methods (even in novel ways) without having to reason ahead of time about what the outcome of that combination will be.

The reason that coevolution appears to work better than just memetic evolution alone may be the two way linkage between parameter values and representations. Not only does it appear to be the case that which parameter values produce the best outcome depends on the input representation used, but it also appears that the selection of parameter values has a significant effect on what new

features are constructed. This seems especially relevant when considering that two of the parameters in COEV specify how many input features each learner will use, and what the propensity of each learner is for taking those features from external sources is.

Computational Complexity and Parallelization Issues

One of the drawbacks to coevolution learning is the large amount of computation required. Doing an evolutionary search is slow in general, and when the evaluation of a single individual can take minutes or hours (as training a neural network can) then the problem is exacerbated. There are three reasons to be optimistic that this approach can be used to solve large scale problems.

First, the fitness function for the population includes a time factor, which tends to increase the speed of learning as the evolution progresses. Average running time of both LFC++ and CG was substantially lower at the end of ten generations than it was at the beginning. The search through the space of parameter values not only finds values that provide good results, but also ones that provide them quickly.

Second, as better input representations are found, most learning methods increase their learning speed. Since the concepts learned from good representations tend to be simple (that's what makes a representation good), and most machine learning methods have a computational cost that is dependent on the complexity of the learned concept, as the system evolves better representations, the speed of learning should go up.

Finally, coevolution learning is well suited to parallelization. The vast majority of the computation is done by the learning agents, which are entirely independent of each other during learning. The only communication required is at the time of meme and gene exchange, where all agents must register their fitnesses and the fitnesses of their memes. It is also possible to relax even that requirement, so that agents use only a sample of the fitnesses from other agents to determine who to exchange memes and genes with. Since the selection of memes and genetic partners is probabilistic anyway, sampling will be indistinguishable from exhaustive search if the sample size is large enough. By using sampling, it is also possible to do asynchronous updating of the population.

Future directions

There are three current research goals for coevolution learning currently being pursued. The first is to empirically demonstrate its effectiveness on realistic problems. In particular, we are hoping to demonstrate that this approach solves problems that are very difficult for other machine learning or statistical approaches. Solving real world problems is likely to require additional learning methods, larger agent populations and longer evolution times.

The second goal is to add several new learning techniques to the system. In particular, we are interested in adding:

- a function finding system that can generate memes that identify mathematical relationships between features;
- a feature relevance metric (e.g. (Kira & Rendell, 1992)) that can be added to the meme fitness function;
- a relation learning system (e.g. (Quinlan, 1990) or one of the ILP learners) that can help the system transcend the limits of boolean feature combinations

Each of these desired additions presents challenges to the design of the current system that must be addressed.

Our third current goal is to add at least some coarse grained parallelism to the system so that it can be run on a network of workstations. Due to the nature of the computations, we expect nearly speedup nearly linear with the number of workstations used, up to the point where there is one workstation per agent.

Conclusion

Coevolution provides a powerful metaphor for the design of a machine learning system. Models of genetic evolution have already driven significant advances in machine learning. When nature added cultural evolution to genetic evolution, the result was spectacular. The computational model of culture proposed in this paper is tremendously empovershed in comparison to people. Nevertheless, the ability of even this drastically simplified model of coevolution to synergize the abilities of disparate learners appears promising.

References

Angeline, P. J. (1993) Evolutionary Algorithms and Emergent Intelligence. Ph.D. diss, Ohio State University,

Dietterich, T. (1989). Limitations on Inductive Learning. In Proceedings of Sixth International Workshop on Machine Learning, (pp. 125-128). Ithaca, NY: Morgan Kaufman.

Durham, W. H. (1991). *Coevolution: Genes, Culture and Human Diversity*. Stanford, CA: Stanford University Press.

Hunter, L. (1989) Knowledge Acquisition Planning: Gaining Expertise Through Experience. PhD diss, Yale University, Available as YALEU/DCS/TR-678.

Hunter, L. (1990). Planning to Learn. In Proceedings of The Twelveth Annual Conference of the Cogntive Science Society, (pp. 26-34). Boston, MA:

Hunter, L. (1992). Knowledge Acquisition Planning: Using Multiple Sources of Knowledge to Answer Questions in Biomedicine. *Mathematical and Computer Modelling,* 16(6/7), 79-91.

Kira, K., & Rendell, L. (1992). The Feature Selection Problem: Traditional Methods and a New Algorithm. In Proceedings of National Conference on AI (AAAI-92), (pp. 129-134). San Jose, CA: AAAI Press.

Kohavi, R., & John, G. (1995). Automatic Parameter Selection by Minimizing Estimated Error. In Proceedings

of ECML-95: The Eighth European Conference on Machine Learning.

Langley, P. (1994). Selection of Relevant Features in Machine Learning. In Proceedings of AAAI Fall Symposium on Relevance. New Orleans, LA: AAAI Press.

Michalski, R. S., & Tecuci, G. (Ed.). (1994). *Machine Learning IV: A Multistrategy Approach*. San Mateo, CA: Morgan Kaufman Publishers.

Quinlan, J. R. (1990). Learning Logical Definitions from Relations. *Machine Learning*, 5(3), 239-266.

Quinlan, J. R. (1991). C4.5. In Sydney, Australia: Available from the author: quinlan@cs.su.oz.au.

Rendell, L., & Cho, H. (1990). Empirical Learning as a Function of Concept Character. *Machine Learning*, 5(3), 267-298.

Rendell, L., & Ragavan, H. (1993). Improving the Design of Induction Methods by Analyzing Algorithm Functionality and Data-based Concept Complexity. In Proceedings of IJCAI, (pp. 952-958). Chambery, France:

Schank, R., Collins, G., & Hunter, L. (1986). Transcending Inductive Category Formation In Learning. *Behavioral and Brain Sciences*, 9(4), 639-687.

Towell, G. G., Shavlik, J. W., & Noordewier, M. O. (1990). Refinement of Approximate Domain Theories by Knowledge-Based Artificial Neural Networks. In Proceedings of Seventh International Conference on Machine Learning, .

van Camp, D. (1994). UTS/Xerion. In Toronto, Canada: Available by anonymous file transfer from ftp.cs.toronto.edu:/pub/xerion/uts-4.0.tar.Z.

Vilalta, R. (1993). LFC++. In Available from the author: vilalta@cs.uiuc.edu.

Wisniewski, E., & Medin, D. (1994). The Fiction and Nonfiction of Features. In R. Michalski & G. Tecuci (Eds.), *Machine Learning IV: A Multistrategy Approach* San Francisco, CA: Morgan Kaufmann.

Wnek, J., & Michalski, R. (1994). Hypothesis driven constructive induction in AQ17: A method and experiments. *Machine Learning* 14:139-168.

Yao, X. (1993). Evolutionary artificial neural networks. International Journal of Neural Systems, 4(3), 203-221.

A Comparison of Action Selection Learning Methods

Diana F. Gordon
Naval Research Laboratory, Code 5510
4555 Overlook Avenue, S.W.
Washington, D.C. 20375
gordon@aic.nrl.navy.mil

Devika Subramanian
Department of Computer Science
Rice University
Houston, TX 77005
devika@cs.rice.edu

Abstract

Our goal is to develop a hybrid cognitive model of how humans acquire skills on complex cognitive tasks. We are pursuing this goal by designing hybrid computational architectures for the NRL Navigation task, which requires competent sensorimotor coordination. In this paper, we empirically compare two methods for control knowledge acquisition (reinforcement learning and a novel variant of action models), as well as a hybrid of these methods, with human learning on this task. Our results indicate that the performance of our action models approach more closely approximates the rate of human learning on the task than does reinforcement learning or the hybrid. We also experimentally explore the impact of background knowledge on system performance. By adding knowledge used by the action models system to the benchmark reinforcement learner, we elevate its performance above that of the action models system.

Introduction

Our goal is to develop a hybrid cognitive model of how humans acquire skills by explicit instruction and repeated practice on complex cognitive tasks. We are pursuing this goal by designing hybrid (multistrategy) computational architectures for the NRL Navigation task, which requires sensorimotor coordination skill. In this paper, we develop a novel method based on parametric action models for actively learning visual-motor coordination. Although similar to previous work on action models, our method is novel because it capitalizes on available background knowledge regarding sensor relevance. We have confirmed the existence and use of such knowledge with extensive verbal protocol data collected from human subjects. In our action models approach, the agent actively interacts with its environment by gathering *execution traces* (time-indexed streams of visual inputs and motor outputs) and by learning a compact representation of an effective policy for action choice guided by the action model.

This paper begins by describing the NRL Navigation task, as well as the types of data collected from human subjects performing the task. Then, two learning methods are described: our model-based method and a benchmark reinforcement learning algorithm that does not have an explicit model. Prior results reported in the literature of empirical comparisons of action models versus reinforcement learning are mixed (Lin, 1992; Mahadevan, 1992); they do not clearly indicate that one method is superior. Here we compare these two methods empirically on the Navigation task using a large collection of execution traces. Our primary goal in this comparison is to determine which performs more like human learning on this task. Both methods include sensor relevance knowledge from the verbal protocols. The results of this empirical comparison indicate that our action models method more closely approximates the time-scales and trends in human learning behavior on this task. Nevertheless, neither algorithm performs as well as the human subject.

We next explore a multistrategy variant that combines the two methods for the purpose of better approximating the human learning, and present empirical results with this method. Although the multistrategy approach is unsuccessful, an alternative is highly successful. This alternative consists of modifying the architecture of the reinforcement learner to incorporate knowledge used by the action models method.

The NRL Navigation and Mine Avoidance Domain

The NRL navigation and mine avoidance domain, developed by Alan Schultz at the Naval Research Laboratory and hereafter abbreviated the "Navigation task," is a simulation that can be run either by humans through a graphical interface, or by an automated agent. The task involves learning to navigate through obstacles in a two-dimensional world. A single agent controls an autonomous underwater vehicle (AUV) that has to avoid mines and rendezvous with a stationary target before exhausting its fuel. The mines may be stationary, drifting, or seeking. Time is divided into episodes. An episode begins with the agent on one side of the mine field, the target placed randomly on the other side of the mine field, and random

mine locations within a bounded region. An episode ends with one of three possible outcomes: the agent reaches the goal (success), hits a mine (failure), or exhausts its fuel (failure). Reinforcement, in the form of a binary reward dependent on the outcome, is received at the end of each episode. An episode is further subdivided into decision cycles corresponding to actions (decisions) taken by the agent.

The agent has a limited capacity to observe the world it is in; in particular, it obtains information about its proximal environs through a set of seven consecutive sonar segments that give it a 90 degree forward field of view for a short distance. Obstacles in the field of view cause a reduction in sonar segment length (segment length is proportional to obstacle distance); one mine may appear in multiple segments. The agent also has a range sensor that provides the current distance to the target, a bearing sensor (in clock notation) that indicates the direction in which the target lies, and a time sensor that measures the remaining fuel. A human subject performing this task sees visual gauges corresponding to each of these sensors. The turn and speed actions are controlled by joystick motions. The turn and speed chosen on the *previous* decision cycle are additionally available to the agent. Given its delayed reward structure and the fact that the world is presented to the agent via sensors that are inadequate to guarantee correct identification of the current state, the Navigation world is a partially observable Markov decision process (POMDP).

An example of a few snapshots from an execution trace (with only a subset of the sensors shown) is the following:

time	range	bearing	sonar1	turn	speed
4	1000	1	220	32	20
5	1000	12	220	-32	20
6	1000	11	220	0	20
7	1000	11	90	0	20

A trace file records the binary success/failure for each episode.

Data from Human Subjects

In the experiments with humans, seven subjects were used, and each ran for two or three 45-minute sessions with the simulations. We instrumented[1] the simulation to gather execution traces for subsequent analysis (Gordon *et al.*, 1994). We also obtained verbal protocols by recording subject utterances during play and by collecting answers to questions posed at the end of the individual sessions.

[1]Note that although human subjects use a joystick for actions, we do not model the joystick but instead model actions at the level of discrete turns and speeds (e.g., turn 32 degrees to the left at speed 20). Human joystick motions are ultimately translated to these turn and speed values before being passed to the simulated task. Likewise, the learning agents we construct do not "see" gauges but instead get the numeric sensor values directly from the simulation (e.g., range is 500).

Methods for Modeling Action Selection Learning

Our goal is to build a model that most closely duplicates the human subject data in learning performance. In particular, subjects become proficient at this task (assuming no noise in the sensors and only 25 mines) after only a few episodes. Modeling such an extremely rapid learning rate presents a challenge. In developing our learning methods, we have drawn from both the machine learning and cognitive science literature. By far the most widely used machine learning method for tasks like ours is reinforcement learning. Reinforcement learning is mathematically sufficient for learning policies for our task, yet has no explicit world model. More common in the cognitive science literature are action models, e.g., (Arbib, 1972), which require building explicit representations of the dynamics of the world to choose actions.

Reinforcement learning

Reinforcement learning has been studied extensively in the psychological literature, e.g., (Skinner, 1984), and has recently become very popular in the machine learning literature, e.g., (Sutton, 1988; Lin, 1992; Gordon & Subramanian, 1993). Rather than using only the difference between the prediction and the true reward for the error, as in traditional supervised learning, (temporal difference) reinforcement learning methods use the difference between successive predictions for errors to improve the learning. Reinforcement learning provides a method for modeling the acquisition of the policy function:

$$F : \text{sensors} \rightarrow \text{actions}$$

Currently, the most popular type of reinforcement learning is *q-learning*, developed by Watkins, which is based on ideas from temporal difference learning, as well as conventional dynamic programming (Watkins, 1989). It requires estimating the q-value of a sensor configuration s, i.e., $q(s, a)$ is a prediction of the utility of taking action a in a world state represented by s. The q-values are updated during learning based on minimizing a temporal difference error. Action choice is typically stochastic, where a higher q-value implies a higher probability that action will be chosen in that state.

While q-learning with explicit state representations addresses the temporal credit assignment problem, it is standard practice to use input generalization and neural networks to also address the structural credit assignment problem, e.g., (Lin, 1992). The q-value output node of the control neural network corresponding to the chosen action a is given an error that reflects the difference between the current prediction of the utility,

$q(s_1, a_i)$, and a better estimate of the utility (using the reward) of what this prediction should be:

$$error_i =$$

$$\begin{cases} (r + \gamma\ max\{q(s_2, k) | k \in A\}) - q(s_1, a_i) & \text{if } a_i = a \\ 0 & \text{otherwise} \end{cases}$$

where r is the reward, A is the set of available actions, a is the chosen action, s_2 is the state achieved by performing action a in state s_1, i indexes the possible actions, and $0 \leq \gamma < 1$ is a discount factor that controls the learning rate. This error is used to update the neural network weights using standard backpropagation. The result is improved q-values at the output nodes.

We selected q-learning as a benchmark algorithm with which to compare because the literature reports a wide range of successes with this algorithm, including on tasks with aspects similar to the NRL Navigation task, e.g., see (Lin, 1992). Our implementation uses standard q-learning with neural networks. One network corresponds to each action (i.e., there are three turn networks corresponding to turn left, turn right, and go straight; speed is fixed at a level frequently found in the human execution traces, i.e., 20/40). Each turn network has one input node for every one of the 12 sensor inputs (e.g., one for bearing, one for each sonar segment, etc.), one hidden layer[2] consisting of 10 hidden units, and a single output node corresponding to the q-value for that action. A Boltzmann distribution is used to stochastically make the final turn choice:

$$probability(a|s) = e^{q(s,a)/T} / \sum_i e^{q(s,a_i)/T} \qquad (1)$$

where s is a state and the temperature T controls the degree of randomness of action choice.

We use a reward r composed of a weighted sum of the sensor values.[3] Our reward models a special type of sensor relevance information derived from the human subject data we collected — it represents knowledge of the relative importance of the various sensory inputs in determining action. The verbal protocols reveal that the sonar and bearing sensors appear to be critical for action selection. This is logical: after all, the sonar shows mines which you need to avoid, and the bearing tells you whether you are navigating toward or away from the target. We have implemented a reward function that weights the bearing and sonar equally and gives 0 weight to the other sensors. Thus, if the bearing shows the target straight ahead and the sonar segments show no obstacles, then the reward is highest. Our subjects appeared to learn relevance knowledge and action selection knowledge simultaneously. Here, we assume the relevance is known. Future work will involve methods for acquiring relevance knowledge.

The verbal protocols also indicate heuristics for focusing attention on different sensors at different times. This knowledge is implemented in our novel variant of action models, described next. Nevertheless it is not implemented in the q-learner because to do so would require a departure from the standard q-learning architecture reported in the literature, with which we wish to compare initially as a benchmark. Later, we describe modifications to the q-learner to include this focusing knowledge.

Learning action models

One of the more striking aspects of the verbal protocols we collected was that subjects exhibited a tendency to build internal models of actions and their consequences, i.e., *forward models* of the world. These expectations produced surprise, disappointment, or positive reinforcement, depending on whether or not the predictions matched the actual results of performing the action. For example, one subject had an expectation of the results of a certain joystick motion: "Why am I turning to the left when I don't feel like I am moving the joystick much to the left?" Another expressed surprise: "It feels strange to hit the target when the bearing is not directly ahead." Yet a third subject developed a specific model of the consequences of his movements: "One small movement right or left seems to jump you over one box to the right or left," where each box refers to a visual depiction of a single sonar segment in the graphical interface.

Action models (i.e., forward models) have appeared in multidisciplinary sources in the literature. Arbib (1972) and Drescher (1991) provide examples in the psychological literature, STRIPS (Nilsson, 1980) is a classic example in the AI literature, and Sutton uses them in DYNA (Sutton, 1988). The *learning* of action models has been studied in the neural networks (Moore, 1992), machine learning (Sutton, 1990; Mahadevan, 1992), and cognitive science (Munro, 1987; Jordan & Rumelhart, 1992) communities.

Our algorithm uses two functions:

$$M : \text{sensors} \times \text{actions} \rightarrow \text{sensors}$$
$$P : \text{sensors} \rightarrow \Re$$

M is an action model, which our method represents as a decision tree. The decision trees are learned using Quinlan's C4.5 system (Quinlan, 1986).[4] P rates the

[2] We ran initial experiments to try to optimize the reinforcement learning parameters. For the neural networks, the chosen learning rate is 0.5, momentum 0.1, 10 hidden units, and 10 training iterations for the neural networks and a discount factor of 0.9.

[3] Ron Sun suggested a reward of sensor values for this task (personal communication). Our choice of sensor weights for the reward is 30 for bearing and 10 for each of the seven sonar segments, and the scale for the reward is between -1.0 and 0.

[4] We are not claiming humans use decision trees for action models; however, we use this implementation because it appears to have a computational speed that is needed for

desirability of various sensor configurations. P embodies background (relevance) knowledge about the task. For sonars, high utilities are associated with large values (no or distant mines), and for the bearing sensor high utilities are associated with values closer to the target being straight ahead. Currently, P is supplied by us. At each time step, actions are selected using P and M by performing a 1-step lookahead with model M and rating sensory configurations generated using P. The action models algorithm has the same action set as the q-learning algorithm, i.e., turn right, turn left, or go straight at a fixed speed (20/40).

First, our algorithm goes through a training phase, during which random turns are taken and the execution traces saved as input for C4.5. C4.5 models the learning of the function M. In particular, it constructs two decision trees from the data: one tree to predict (from (s, a)) the *next* composite value of the sonar segments (prediction choices are no-mines, mine-far, mine-mid, mine-fairly-close, or mine-close, where these nominal values are translations from the numeric sonar readings) and one tree to predict the bearing on the next time step. Note that the choice of these two trees employs the same relevance information used in the reinforcement learning reward function, namely, that the sonar and bearing are the relevant sensors. The training phase concludes after C4.5 constructs these two decision trees.

During the testing phase, these trees representing the world dynamics (M) are consulted to make predictions and select turns. Given the current state, a tree is chosen. The tree selection heuristic for focus of attention (hereafter called the *focus heuristic*) states: if all the sonar segments are below a certain empirically determined threshold (150/220), the sonar prediction tree selects the *next* turn. Otherwise, the bearing prediction tree selects the next turn. To make a prediction, the agent feeds the current sensor readings (which include the *last* turn and speed) and a candidate *next* turn to the decision tree and the tree returns the predicted sonar or bearing value. The agent chooses the next turn which maximizes P.[5]

It is unlikely that humans recompute the consequences of actions when the current state is similar to one seen in the past. Therefore, our future work will address memorizing cases of successful action model use so that memory can be invoked, rather than the trees, for some predictions.

modeling human learning. We are also investigating connectionist models as in Jordan & Rumelhart (1992). Currently, C4.5 learning is in batch. To more faithfully model human learning, we are planning to use an incremental version of decision tree learning in future implementations.

[5]If the next turn is considered irrelevant by the decision tree, a random action choice is made.

Empirical Comparison of the Two Methods

To make our comparisons fair, we include a training phase for the reinforcement learner with Boltzmann temperature at 0.5, which results in random actions.[6] A testing phase follows in which the turn with the best q-value is selected deterministically at each time step. In summary, the reinforcement learner takes random actions and learns its q-values during training.[7] It uses these learned q-values for action selection during testing. The action models method takes the same random actions as the q-learner during training (i.e., it experiences exactly the same sensor and action training data as the q-learner), and then from the execution trace training data it learns decision tree action models. The focus heuristic uses the learned trees for action selection during testing. Neither of the two methods learns during testing. Both methods have the same knowledge regarding which sensors are relevant, i.e., the bearing and sonar sensors.

In our experimental tests of all hypotheses, the training phase length is varied methodically at 25, 50, 75, and 100 episodes. The testing phase remains fixed at 400 episodes.[8] Each episode can last a maximum of 200 time steps, i.e., decision cycles. In all experiments, the number of mines is fixed at 25, there is a small amount of mine drift, and no sensor noise. These task parameter settings match exactly those used for the human subject whose learning we wish to model.[9]

Recall that the outcome of every episode is binary (success/failure) and that success implies the AUV avoids all the mines and reaches the target location. The performance measure we use is the percentage of test episodes in which the AUV succeeds. This information is obtained from the trace file. Performance is averaged over 10 experiments because the algorithms are stochastic during training, and testing results depend upon the data seen during training. Graphs show mean performance, and error bars denote the standard deviation.

We denote the q-learning scheme described above as Q and the action model scheme with decision trees described above as A. Our goal is to find an algorithm that most closely approximates the human learning. We start with the basic algorithms (Q and A), then

[6]We also tried an annealing schedule but performance did not improve.

[7]Arbib (1972) provides convincing cognitive justification for the role of random exploration of actions in the acquisition of motor skill.

[8]We experimented with the number of episodes and chose a setting where performance improvement leveled off for both algorithms.

[9]Both algorithms go straight (0 turn) for the first three time steps of every episode during training. This not only matches performance we observed in the execution traces from human subjects, but also aids the learning process by quickly moving the AUV into the mine field.

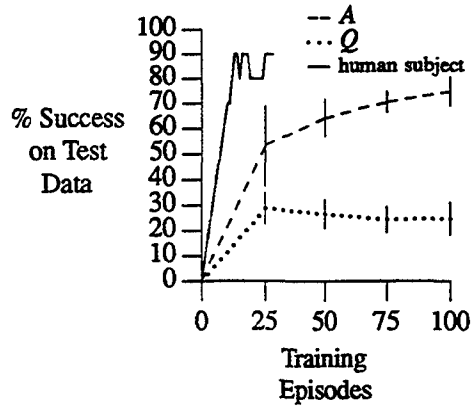


Figure 1: The graph represents the learning curves for A, Q, and the human subject.

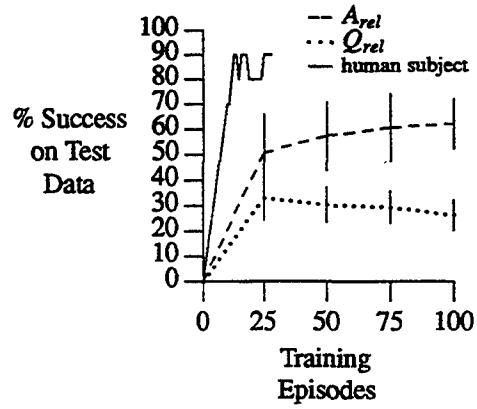


Figure 2: The graph represents the learning curves for A_{rel}, Q_{rel}, and the human subject.

make changes as needed to more closely approximate the human learning.

We begin by empirically testing the following hypothesis:

- *Hypothesis 1:* The slope of A's learning curve is closer than Q's to the slope of the human learning curve, for the Navigation task.

The justification for Hypothesis 1 is that our action models method uses an action choice policy, including a focus heuristic, specially designed to capitalize on sensor relevance knowledge.

To test Hypothesis 1, we used data from a typical (the variance between subjects was surprisingly low) subject for a single 45-minute session. Note that we cannot divide the human learning into a training phase and a testing phase during which the human stops learning. Therefore, we have averaged performance over a sliding window of 10 previous episodes. We considered averaging performance over multiple subjects, but that would entail significant information loss.

Figure 1 shows the results of testing Hypothesis 1. A outperforms Q at a statistically significant level (using a paired, two-tailed t-test with $\alpha = 0.05$). Thus, Hypothesis 1 is confirmed.[10] Apparently, our novel method for coupling action models with an action choice policy that exploits sensor relevance has tremendous value for this task.

Among the most plausible explanations for the power of our action models approach over the q-learner for this task are: (1) A's focus of attention heuristic, (2) use of an action model *per se*, and (3) the decision tree representation, e.g., Chapman and Kaelbling (1991) discuss how decision trees can improve over neural networks with backpropagation for reinforcement learning.

[10]It is unclear why the performance of the q-learner drops slightly with more training episodes, though perhaps overfitting explains this.

Although A performs more like the human than Q, the most pressing question is why neither performs as well the human. We next add more knowledge in order to gain a better approximation of the curve of the human learner. A more careful examination of the verbal protocols indicates that subjects not only attended to the sonar sensor, but some subjects mentioned focusing almost exclusively on the middle three sonar segments. This makes sense if you consider the middle three sonar segments show mines straight ahead, which is critical information for avoiding immediate collisions. We have have altered Q's reward function to weight the bearing equally to the middle three sonar segments and to give all other sensors zero weight. Thus, if the bearing shows the target straight ahead and the middle three sonar segments show no obstacles ahead, then the reward is highest. A was also modified to include this more specific relevance knowledge, i.e., the modified version only predicts the values of the middle three sonar segments rather than all, and the focus heuristic checks the threshold for only those three segments. We denote the modified q-learning scheme Q_{rel} and the modified action model scheme A_{rel}.

We empirically test the following hypothesis:

- *Hypothesis 2:* The slope of A_{rel}'s learning curve (respectively, Q_{rel}) is higher than that of A (respectively, Q), for the Navigation task.

The justification for Hypothesis 2 is that the addition of relevance knowledge should improve performance.

Figure 2 shows the results of testing Hypothesis 2. If we compare Figure 2 with Figure 1, we see that Q_{rel} performs slightly better than Q. The performance difference is not statistically significant ($\alpha = 0.10$) for training lengths of 25, 50, and 100, but is significant ($\alpha = 0.10$) with a training length of 75. The surprise is that A_{rel} performs worse than A. The differences are statistically significant at $\alpha = 0.05$ for training lengths of 75 and 100, but only at $\alpha = 0.20$ for training lengths of 25 and 50. Our results refute Hypothesis 2.

(Although these results refute Hypothesis 2, they provide further confirmation of Hypothesis 1 because the performance improvement of A_{rel} over that of Q_{rel} is statistically significant with $\alpha = 0.05$).

Apparently, adding this relevance knowledge does not help A, which is the better performer. We conjecture the reason is that although the middle three sonar segments may be the most relevant, all the sonar segments are in fact relevant for action choice. For example, if there is a mine ahead as well as one to the left, then the best action to take to avoid the mines is to turn right. With knowledge only of what is straight ahead, turning right or left would seem equally valid. The subjects were most likely using this knowledge, but their verbal recollections were probably incomplete accounts of the full knowledge that they actually used for decision-making.

A Multistrategy Approach

The addition of knowledge about which sensors are relevant does not improve our algorithms. Therefore, we are still left with the question: why are *both* methods slower learners than the human?[11] While Q in principle could match A's performance, the search that it embodies has to be rescued from the performance plateau we find in Figures 1 and 2, in order to match human learning rates. Q possesses the ability to implicitly acquire the function P as well as the focus heuristic from task interaction. Therefore we examine the question: is it possible to combine algorithms A and Q to create a hybrid method that outperforms both, and which more closely matches the human learning curve on this task?

We are in the preliminary stages of investigating this question. The idea is to use A's superior performance after the same level of training to get Q out of its performance plateau and then use Q to refine the knowledge it acquired from A's focus heuristic and sensor evaluation function P. We have implemented the first part of our hybrid scheme in the algorithm MSL, described below.

MSL performs a random walk in the space of actions for the first 25 episodes, i.e., it experiences exactly the same training data as A and Q for the first 25 episodes. The A component of MSL then builds the decision trees. Next, MSL uses the focus heuristic with the learned decision trees to select actions from episodes 25 to 100. Concurrently, action choices and rewards are used to refine the q-values for Q. Since we have empirically established that the action choices of A are significantly better than those of Q after 25 episodes, we expect algorithm Q to find itself in a better region of the policy space during learning episodes 25 to 100. After the 100th episode, we turn off learning and test MSL for 400 episodes. In the testing phase, MSL chooses actions based on the learned q-values alone. The results of this experiment (shown below) are quite surprising. The performance of MSL turns out to be not statistically significantly different from that of Q ($\alpha = 0.20$). This experiment highlights some of the difficulties in constructing hybrid algorithms for complex tasks.

Algorithm	Training Length	Mean	Std
Q	100	24.92	6.62
MSL	100	27.07	8.56
A	25	53.50	15.81

It seems the multistrategy approach is not rescuing Q to make the combined system perform more like the human. In the next section, we try an alternative tactic, namely, that of adding what we consider to be one of the most useful elements of A into Q. Perhaps by adding elements of A one by one into Q we can eventually develop a system whose learning rate more closely approximates that of the human than either A or Q because it would include the best elements of both systems.

Adding the Focus Heuristic to Q

As mentioned above, the focus heuristic, the use of action models, and the use of decision trees are all candidate explanations for A's superior performance over that of Q. We begin by considering the impact of the focus heuristic. This heuristic separates sonar and bearing information and therefore is able to take advantage of knowledge regarding which sensors are relevant *when*. Because the q-learning method lumps all knowledge into one reward it probably needs longer training to do likewise. Here, we explore the impact of adding the focus heuristic to Q. Future work will investigate adding other elements of A to Q one by one.

A new version of Q, called Q_{focus}, has been created by generating two sets of neural networks, one for selecting the actions for making choices to improve the bearing and another set for improving the sonar. The first set receives the bearing component of the reward and the second receives the sonar component (i.e., these are the respective elements of the previous reward, which is a weighted sum). Arbitration between the two sets of networks is done by the same focus heuristic used by the action models approach, namely, use the sonar networks to select a turn (based on Boltzmann selection using the q-values) if the sonar values are below the threshold; otherwise use the bearing networks to select.

We empirically test the following hypothesis:

- *Hypothesis 3:* The slope of Q_{focus}'s learning curve is closer to that of the human's than Q's is.

[11] It may also be the case that their asymptotic performance is lower than that of humans, although A does show signs of a steady increase and the slope is positive even at 100 training episodes.

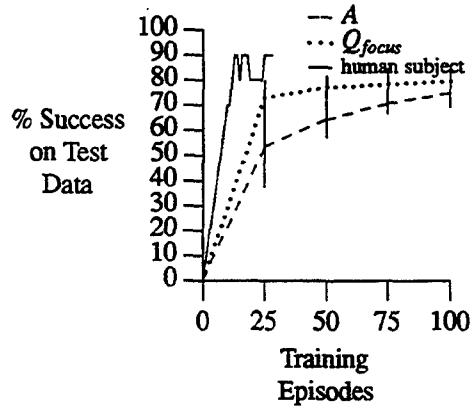


Figure 3: The graph represents the learning curves for A, Q_{focus}, and the human subject.

The results of testing Hypothesis 3 are in Figure 3. The hypothesis is confirmed, but what is unexpected is that Q_{focus} outperforms A and comes significantly closer to the human learning curve. The differences between the curves for Q_{focus} and A are statistically significant at all points ($\alpha = 0.05$). Apparently the focus heuristic plays a major role in the advantage of A over Q, and by adding that knowledge to generate Q_{focus}, we have generated a system that most closely approximates the human learner of any system we have investigated so far.

Discussion

Our results indicate that relevance knowledge, especially knowledge regarding which sensors are relevant when, is one of the keys to better performance on this domain. Future work will explore adding other elements of A to Q. The rate of human learning on this task is a function of both the amount of strategic knowledge that humans possess on related navigation tasks, as well as the finely honed sensorimotor procedural machinery that makes *effective* use of this information. To make our algorithms competitive with human learning, we are making explicit other forms of navigational knowledge brought to bear by humans and by refining our algorithms for making use of this knowledge. We hope to thereby achieve our goal of developing an algorithm that models human learning on the Navigation task.

Acknowledgements

This research was sponsored by the Office of Naval Research N00014-95-WX30360 and N00014-95-1-0846. We thank Sandy Marshall and William Spears for their helpful comments and suggestions, and especially William Spears for his suggestion to divide the q-learner into two data structures in order to use the focus heuristic. Thanks also to Helen Gigley and Susan Chipman for their encouragement and support, to Jim Ballas for the joystick interface, and to Alan Schultz for all the work he did to tailor the simulation to both machine learning and human experimental needs.

References

Arbib, M.A. 1972. *The Metaphorical Brain.* NY: Wiley and Sons Publishers.

Breiman, L., Friedman, J.H. Olshen, R.A.. & Stone, C.J. 1984. *Classification and Regression Trees.* Belmont, CA: Wadsworth International Group Publishers.

Chapman, D. & Kaelbling, L. 1991. Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. In *Proceedings of Twelfth International Joint Conference on Artificial Intelligence* (pp. 726–731). San Mateo, CA: Morgan Kaufmann Publishers.

Drescher, G.L. 1991. *Made-Up Minds.* Cambridge, MA: MIT Press.

Gordon, D., Schultz, A., Grefenstette, J., Ballas, J., & Perez, M. 1994. *User's Guide to the Navigation and Collision Avoidance Task.* Naval Research Laboratory Technical Report AIC-94-013.

Gordon, D. & Subramanian, D. 1993. A Multistrategy Learning Scheme for Agent Knowledge Acquisition. *Informatica*, 17, 331–346.

Jordan, M.I. & Rumelhart, D.E. 1992. Forward models: supervised learning with a distal teacher. *Cognitive Science*, 16, 307–354.

Lin, L. 1992. Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Machine Learning*, 8, 293–321.

Mahadevan, S. 1992. Enhancing transfer in reinforcement learning by building stochastic models of robot actions. In *Proceedings of the Ninth International Conference on Machine Learning* (pp. 290–299). San Mateo, CA: Morgan Kaufmann Publishers.

Moore, A. 1992. Fast, robust adaptive control by learning only forward models. In *Proceedings of the International Joint Conference on Neural Networks*, 4, (pp. 571–578). San Mateo, CA: Morgan Kaufmann.

Munro, P. 1987. A dual back-propagation scheme for scalar reward learning. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society* (pp. 165–176). Hillsdale, NJ: Erlbaum.

Nilsson, N. 1980. *Principles of Artificial Intelligence.* Palo Alto, CA: Tioga Publishing Company.

Quinlan, J.R. 1986. Induction of decision trees. *Machine Learning*, 1, 81–107.

Rissanen, J. 1983. *Minimum Description Length Principle.* Report RJ 4131 (45769), IBM Research Laboratory, San Jose.

Rumelhart, D.E. & McClelland, J.L. 1986. *Parallel Distributed Processing : Explorations in the Microstructure of Cognition.* Cambridge, MA: MIT Press.

Skinner, B.F. 1984. Selection by consequences. *The Behavior and Brain Sciences*, 7, 477–510.

Sutton, R. 1988. Learning to Predict by the Methods of Temporal Differences. Machine Learning, 3, 9–44.

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

Watkins, C. 1989. Learning from Delayed Rewards. Doctoral dissertation. Cambridge, England: Cambridge University.

A Cognitive Modeling Approach to Learning of Ill-defined Categories

Mukesh Rohatgi

Information Systems and Decision Sciences Department
Old Dominion University
Norfolk, VA 23529
E-mail: mxr100f@economy.bpa.odu.edu

Abstract

A vast majority of concepts learned by human beings are ill-defined. Such concepts elude precise definitions and are represented by nonverbalizable descriptions. Literature from cognitive psychology provides an insight into processes used by human beings for learning ill-defined categories. This paper describes the implementation of an Adaptive Concept Learning (ACL) system designed to learn ill-defined categories by simulating human learning behavior. The contents of this paper stand in contrast to the extant concept learning systems described in machine learning literature. The extant systems are generally designed to learn well defined concepts that can be represented with the help of verbalizable rules. In contrast, the system described in this paper is designed to learn ill-defined categories. Therefore, the implementation of ACL is a more realistic simulation of concept learning behavior observed in human beings.

Introduction

Concept learning, also referred to as learning by examples or inductive learning, has received much attention within the area of machine learning (Winston 1975, Mitchell 1982, Michalski 1983, Dietterich & Michalski 1983, Quinlan 1986). A typical concept learning system uses generalization to generate a concept description from specific instances that belong to a predefined category. The purpose of the learning system is to induce a maximally-specific or a complete and consistent structural description of the category from the exemplar set. Concept learning systems also use counterexamples to prevent overgeneralization. In such cases, system input consists of positive and negative examples and the goal is to induce a concept description which covers all positive examples and excludes all counterexamples.

As pointed out by Schank et al. (Schank, Collins, & Hunter 1986), the typical learning system, as described above, does not relate to what people do in real life. The main problem highlighted by Schank et al. was the usage of well defined concepts in machine learning systems. A well defined concept is characterized by defining features that form necessary and sufficient conditions for category membership. This view of category membership, referred to as the classical view by cognitive psychologists, is currently being faced with the following problems (for a detailed review of criticisms against the classical view see Smith & Medin 1981):

(a) the failure of experts to specify defining properties for most real-world object concepts;
(b) its inability in explaining the typicality effects exhibited by category members; and
(c) use of nonnecessary features by human beings in the determination of category membership.

Psychological Evidence for Learning Ill-Defined Categories

The failure of classical view in adequately explaining the basis of concept formation has led the way towards empirical demonstrations that natural objects form categories with ill-defined boundaries (Mervis & Rosch 1981). The current thinking on learning concept descriptions for ill-defined categories is dominated by two model types. The first type includes prototype-based models subsumed under the probabilistic viewpoint and the second type includes exemplar-based models conforming to the exemplar viewpoint (Smith & Medin 1981, Barsalou 1990). Comparisons between prototype-based and exemplar-based models have shown that latter fare better in explaining experimental data on categorization (Medin & Florian 1992), but as discussed in Homa, Sterling, & Trepel (1981), the exemplar models may be faring better because of simplistic laboratory conditions being used during evaluation.

The general issue that concerns us is the nature of information acquired by people when they experience several instances of ill-defined categories, i.e., categories that do not reduce to a simple or easily specified rule (Elio & Anderson 1981). Studies conducted by Posner and Keele (Posner & Keele 1968, 1970) have shown that in case of ill-defined categories, human beings tend to abstract or summarize the description of category

members in form of a prototype. The category prototype does not represent a set of necessary and sufficient features but captures the central tendency of the properties associated with the experienced exemplars (Reed 1972). Additional empirical evidence obtained by cognitive psychologists (Barsalou 1990, Breen & Schvaneveldt 1986, Homa, Sterling, & Trepel 1981, Homa & Vosburgh 1976, Homa & Chambliss 1975, Homa et al. 1973) also seems to suggest that situations that deal with multiple, highly variable, and large sized ill-defined categories favor a prototype-based model for concept learning. These empirical studies have also shown that an abstracted prototype should be viewed as an evolving, dynamic construct that undergoes repeated transformations with continuing exposure to category instances. In addition to the above, these studies have also shown that the retention of abstracted information is superior to the retention of individual exemplars.

Research Objectives

As pointed out earlier, extant systems designed for concept learning tend to ignore ill-defined categories. Also, the central idea embodied in these systems is to generate a verbalizable "decision rule" as the concept description with the help of generalization heuristics. In our research, we are interested in designing a system that is capable of learning concept descriptions of ill-defined categories as opposed to well defined categories. Furthermore, we want to use category prototypes as the basis for representing concept descriptions. The use of prototypes for concept representation is motivated by the psychological evidence outlined in the previous section.

The objective of this paper is to describe the construction and performance of a learning system that has been designed to implement the human learning processes believed to be responsible for learning prototype-based concept descriptions of ill-defined categories. System functions include acquisition and storage of object descriptions (category exemplars); derivation of a prototype from observed exemplars; and dynamic (continuous) modification of prototype as a function of category experience. The following section will describe the organization and implementation of the concept learning system. After describing the system construction, we evaluate its performance with respect to two classification experiments. Finally we present the conclusions along with our agenda for future work.

Description of the Learning System

The learning system, henceforth referred to as the Adaptive Concept Learner (ACL), was implemented under version 2.1 of ART-IM (Automated Reasoning Tool) for MS-DOS. Since a considerable amount of research in cognitive psychology has assumed category prototypes or concept descriptions to be feature-based mental constructs, it was deemed appropriate to use frames as the knowledge representation scheme for the learning system.

As mentioned previously, the overall goal of the learning system is to incrementally learn concept descriptions in form of category prototypes. To achieve its goal, the learning system utilizes multiple learning processes integrated hierarchically in a closed-loop configuration. Each learning process makes use of one or more knowledge transmutation operators (see Michalski 1994) in response to external inputs or to outputs from a lower level learning process. The transmutation operators used by the ACL learning process are shown in Table 1.

Learning Processes	Knowledge Operators	Affected / Resulting Knowledge Structures
Object Learning	Insertion (memorization), and Deletion	Instance Descriptions
Exemplar Learning	Agglomeration	Reduced Exemplar-Based Representations
Prototype Learning	Abstraction, Selection, and Characterization	Category Prototypes
Failure-Based Classification Learning	Characterization	Multiple Category Prototypes

Table 1. Knowledge Transmutation operators used by the learning processes of the ACL System

The learning cycle is initiated with the invocation of object learning process. At this stage, the goal of the learning system is to acquire input knowledge through rote learning. The learning task is accomplished in a supervised learning environment and results in the acquisition of category instances described by a user or a teacher. Besides the acquisition of input knowledge, the object learning process is also responsible for maintaining the background knowledge used by the system to facilitate the user-system interaction.

Descriptions of category instances acquired through the object learning process are then agglomerated into reduced exemplar-based representations (Barsalou 1990) by the exemplar learning process. It also maintains several forms of frequency counts that are used at later stages of the learning cycle for assigning weights to the attribute-value pair included in the category prototype.

The reduced exemplar-based representations serve as inputs to the prototype learning process. The goal of the prototype learning process is to derive a category prototype through abstraction and assign weights to the attribute-value pairs included in the prototype description. A category prototype serves as the characteristic description of an object class during classification. Category prototypes are updated if the learning system is exposed to additional category instances. Updating also results in periodic weeding of idiosyncratic information present in the prototype description. The weeding of idiosyncratic information is equivalent to the selection of relevant attributes for a category prototype.

The prototype learning process generates a single prototype for each category. For highly variable categories, the cognitive economy provided by creating category prototypes may prove to be detrimental. To protect itself from such contingencies, the ACL system relies on a failure-based classification learning process to generate multiple category prototypes for capturing the category variability. Hence, the classification process used by ACL has been designed to perform a dual role. In its first role, it acts as a passive similarity-based classifier which is responsible for the determination of category membership when provided with an object description without the knowledge of its class affiliation. In its second role, the classification process is responsible for generating multiple prototypes for a single category. Prototypes generated through failure-based classification should be considered as abstract representations of subclusters within a highly variable class.

The following sections will describe the details of the learning processes used by the ACL system.

Object Learning Process

The object learning process functions as a surrogate for the perceptual and sensory capabilities present in natural systems. In reality, it is the user-interface which allows a teacher to interact with the learning system. The role of object level processes is to facilitate the teacher-system interaction and acquire object descriptions provided by a teacher through rote learning.

The system starts its learning cycle with zero knowledge about the domain in which it is operating. Knowledge acquisition at object level is concerned with the learning of an object description in terms of attribute-value pairs. The system expects the user to provide it with a label, which it assumes to be the name of an object instance belonging to a basic level category (for the definition of basic level categories see Rosch 1978). After a label has been provided as an input to the system, the user is asked to indicate the object type of the label. The ACL expects the object type to be either a physical object, a conceptual object, a physical action, or a mental action.

To facilitate the user-system interaction, we have provided the learning system with a certain amount of metaknowledge about the typical attributes that can be used to describe a given object type. At this point, it should be understood that the ACL is capable of learning this metaknowledge by itself. The metaknowledge has been preloaded to reduce excessive user-system interaction.

Based on its metaknowledge, the system prompts for a value of an attribute along with a displayed list of known values associated with the attribute. A user can either pick a value from the displayed list or opt to provide a new value for the attribute by choosing the "other" option in the displayed list. The user may also decide not to provide a value for the prompted attribute.

A user is not limited to the attributes that he or she is prompted for. After the system has exhausted the known list of attributes, it displays a menu through which a user can either: (a) add or delete an attribute in the current object description; (b) add or delete a value of an attribute in the current object description; (c) provide multiple values for an attribute; or (d) take no action and quit the menu. The add-delete menu provides the user with an opportunity to define new attributes for the category instance currently being described or change the earlier responses if necessary. For example, if the user wants to specify whether an instance of a ball is solid or hollow, then by using the menu options he or she can define an additional attribute and give it a name of his or her choice, e.g., solid-or-hollow. Once an additional attribute is defined, it becomes a permanent part of the system's metaknowledge. Therefore, the next time around, if a user is describing another instance of a ball, then he or she is also asked to provide a value for the solid-or-hollow attribute. The system also keeps track of the data type (symbolic or numeric) for each attribute. Information about attribute data type enables the system to simultaneously work with both symbolic and numeric attributes.

The above described process of acquiring an object description is linear in nature, i.e., knowledge acquisition proceeds from label to type and then to associated attribute-value pairs. This linear execution process is only possible if no new symbol, either in the form of an attribute or an attribute value, is mentioned to the system.

If during the acquisition of an object description, the user mentions a new symbol to the system, the focus of the inquiry changes and the system attempts to learn more about the new symbol by treating it as a new label. The recursion process is infinite because it is being assumed that the system cannot learn an object description in terms of symbols that have not been learned previously. If the user attempts to use circular descriptions (e.g., if the label currently being described is property, and the value for the context attribute of property is specified to be property), the system suspends the recursion and starts unwinding. The recursion process is also terminated, if the user informs the system that not enough information is known about the new symbol at the present time. In such cases the system memorizes the symbol and avoids the recursion process. Additional information about the memorized symbols can be provided to the system at a later time.

The user-system interaction during knowledge acquisition is further facilitated by providing the user with an online display of brief descriptions that serve as explanatory notes describing the scope and meaning of each symbol being used to name an attribute or a value associated with the attribute. Therefore, when the system is informed of a new attribute or a new value of an attribute, it also asks the user to provide a brief description that can be attached to the new symbol. The word description should not be confused with object descriptions in form of attribute-value pairs. Here the word description refers to a string of characters used as explanatory statements.

The object learning process of the ACL has been designed to adapt its interaction according to the object type being described. For example, the ACL will change the focus of its metaknowledge if a physical action is being described instead of a physical object. In the case of a physical action, the user is prompted for an agent executing the action, object on which action is being performed, effect of the action, context of the action, etc. This implies that the user interface is controlled by the object type that is currently being described.

Exemplar Learning Process

The exemplar learning process is responsible for collecting object descriptions into exemplar-based categories. Object instances sharing the same symbol for their name are grouped into a category. The shared name also becomes the category label. Category attributes at the exemplar level are determined by creating a union of attributes associated with each member object. A category representation similar to the one described above has been termed as the exemplar view of concepts (see Smith & Medin 1981).

Three forms of statistics are computed while organizing the described objects into a category. The first statistic maintains the total number of exemplars in a given category. An attribute count is maintained with the help of the second statistic, while the third statistic is used to maintain the attribute value count. For example, if ten instances of blue balls, five instances of green balls, and five instances of red balls were organized into a category named ball, then the exemplar count is equal to 20. Similarly, attribute value counts for blue, green and red values of the color attribute for category ball are 10, 5, and 5 respectively. The attribute count for color is 20 because the color attribute appears in the description of all 20 exemplars representing the category ball. The attribute value count is maintained only for symbolic attributes. For numeric attributes with continuous values the exemplar learning process maintains the minimum and maximum values of the attribute observed among the category exemplars. The minimum and maximum values are used for the computation of similarity evidence during classification. All statistics are updated with additional exposure to new exemplars belonging to the category.

Two pieces of information are lost by creating a union of object descriptions to form a category representation. First, multiple objects collected in a category lose their individual identities; and second, because of the identity loss, correlation among category attributes is also lost. Loss of identity can be assumed to represent the forgetting of individual stimuli, a common occurrence in human beings. Also, due to the loss of individuality by category exemplars, the exemplar view is not the correct term for the category representation used by the exemplar learning process. As stated by Barsalou (1990), such representations are subsumed under reduced exemplar models for concept formation.

Prototype Learning Process

The prototype learning process is activated immediately after the assimilation of an object description into the reduced exemplar-based representation of a category. This process is responsible for generating a prototype-based concept description for each object label encountered by the system. Category prototypes are derived from exemplar level knowledge structures through abstraction. The abstraction process captures the central tendency of the attributes present in the reduced exemplar structure. A modal value is used for symbolic attributes and a mean value is used for numeric attributes. Symbolic attributes with more than one modal value function as multivalued attributes in the prototype description.

Besides creating a prototype-based concept description, the prototype creation process also computes the attribute value salience and attribute relevancy for each symbolic

attribute present in the category prototype. Only attribute relevancy is computed for numeric attributes. Attribute value salience is an estimate of the probability of occurrence of an attribute value with respect to a given category. Similarly, attribute relevancy is an estimate of the probability of occurrence of an attribute for a given category. For example, assume that 20 instances of category ball were experienced by the ACL system, and the attribute color was used to describe 10 of them. If the modal value of the attribute color happens to be red, which was observed in 5 instances, then the salience of the value red for the color attribute is estimated to be 0.50 (10 divided by 5). Based on the same data attribute relevancy for the color attribute will be estimated as 0.5 (20 divided by 10). Hence, the attribute value salience and attribute relevancy are computed by using the exemplar count, attribute count and attribute value count statistics maintained at the exemplar level. The salience and relevancy values are used for similarity computations during classification. Attribute relevancy is also used to eliminate idiosyncratic information from a category prototype.

The category prototype description is modified as and when necessary. With the acquisition of additional information at the object level, it is possible to experience a shift in the central tendency of the category attributes. Therefore, the prototype description along with the salience and relevancy values is updated continuously. The update process propagates in a bottom up fashion, i.e., additional information at the object level results in an update at the exemplar level which in turn affects the prototype level.

As mentioned earlier, the category prototype inherits all attributes from the exemplar level. The exemplar level in turn inherits all attributes used in the description of member instances at the object level. This attribute preserving process leads to the inclusion of two types of attributes in the category prototype. The first type represents the set of characteristic attributes, (i.e., attributes often used to describe category instances) and the second set represents the idiosyncratic information or information associated with only a few exemplars of the category. In order to facilitate the evolution of category prototypes towards a predominance of characteristic attributes, the prototype learning process has been designed to eliminate any idiosyncratic information from the prototype description.

The idiosyncratic attributes are eliminated (or deleted) on the basis of attribute relevancy. As explained earlier, an estimate of attribute relevancy is computed by dividing the attribute count with the exemplar count. In order to eliminate attributes with low relevancy, a system defined parameter representing a threshold value (currently set at 0.1) is used. If the attribute relevancy is greater than the threshold parameter, then the attribute is retained as a part of the prototype description; otherwise, it is eliminated. The elimination process is executed whenever exemplar count mod 20 is equal to zero.

Categorization Process

A plausible way to evaluate the quality of concept descriptions generated by a learning system is to evaluate their usefulness in a classification task. The ability of a concept description to produce correct classifications (defined as accuracy by Bergadano et al. 1988) can be used as a surrogate measure of concept quality. Although the form and content of category descriptions is critical for accurate classification, the role of decision rule used in the categorization task should also be taken into account.

A two stage categorization process is used by the ACL system for the purpose of classification. In the first stage, the categorization process attempts to generate multiple prototypes for each object class currently represented in the knowledge base. Multiple category prototypes generated at this stage can be considered as abstract representations of possible subclusters in a highly variable class. The subclusters are identified through failure-based classification of training instances. Classification of test instances comprises the second stage of the categorization process.

Since the categorization process uses multiple prototypes to represent a category, it is also referred to as the Multiple Prototype Classification Model. The following steps illustrate the categorization process used by the ACL system (see Rohatgi 1994, for a complete account of Multiple Prototype Classification model):

1. Before invoking the categorization process, use the mechanisms of object learning, exemplar learning, and prototype learning processes to generate a single prototype for each object class present in the training data set.
2. Use the following procedure to classify instances in the training set:
   a. Create a frame-based representation of the instance that needs to be classified. Therefore, I (instance to be classified) = $\{(a_1, v_1), (a_2, v_2),..., (a_n, v_n)\}$, where a_i is the ith attribute and v_i is the value for the ith attribute in the instance description.
   b. Compute $S_i = \sum_j EVD(I_j, P_{ij})$ for each i, where i = 1...K, K being the number of categories in the knowledge base, I being the instance, P_i being the prototype for ith category, j being the sum of common and distinct attributes present in the prototype and the instance descriptions, EVD being the evidence for similarity, and S_i being the cumulative similarity evidence for the ith category.

The Following rules should be used to compute similarity evidence along each feature dimension:

Similarity evidence for symbolic attributes:

= (attribute value salience x attribute relevancy), if the attribute and attribute values match.

= -(attribute value salience x attribute relevancy), if the attributes match but there is a value mismatch.

= -1.0, if the attribute is present in the instance and is absent in the prototype.

= -(attribute relevancy), if the attribute is present in the prototype and absent in the instance.

Similarity evidence for numeric attributes (e.g. diameter):

$$= 1 - \frac{|\text{instance value} - \text{prototype value}|}{(\text{Max} - \text{Min} + \text{Delta}) / 2.0}$$

Delta = arbitrary small number (0.05). It is being used as a mathematical convenience to avoid dividing by zero.

Note: Min and Max values are identified at the exemplar level, while the attribute value salience and attribute relevancy are computed at the prototype level during the training phase.

c. Compute $C = \text{Max}\{ S_i \}$.
d. Classify the instance as belonging to the category i for which $\sum_j \text{EVD}(I_j, P_{ij}) = C$.

3. A misclassification in step 2 highlights the inadequacy of a single category prototype for accurate classification. The misclassified instance is used as the basis for creating a new subcluster. A prototype for the new subcluster is also generated. At this point, the prototype is based on a single member of the subcluster.

4. Repeat steps 2 and 3 for all training instances (prototypes generated in step 3 also participate in the categorization task after their initial creation). If a subcluster gets assigned as the possible category by the classification procedure in step 2 then the prototype representing the subcluster and the subcluster exemplar count are updated. The exemplar count of the subcluster (number of instances in the reference set of the subcluster) divided by the category exemplar count tabulated in step 1 (number of instances in the entire category) estimates the span of a subcluster in terms of fractional representation. No action is taken if a category prototype generated in step 1 is ruled as most similar to the instance under consideration. It should be noted that the presentation order of training instances may affect the number and quality of subclusters identified in step 3.

5. After exhausting all training instances, the categorization process uses a truncation parameter to drop certain subclusters. The truncation parameter represents a threshold value that ranges between 0.0 and 1.0. Any subcluster with a fractional representation smaller than the truncation parameter is deleted.

6. Classify test instances by using the classification rule outlined in step 2. The object classes present in the training set are assumed to be represented by the multiple prototypes retained after truncation in step 5. If the classification rule picks a subcluster as the possible category then it is assumed that the test instance belongs to the parent class of the subcluster. A match between the class predicted by the classification rule and the true class of the test instance is considered a hit.

To achieve computational economy, the ACL system has been designed to accept a user-defined truncation parameter. If the user is interested in an optimal value of the truncation parameter, then the ACL system can be asked to progressively drop each subcluster starting from the one having the least value in terms of fractional representation. The classification procedure in step 6 is repeated after the elimination of each subcluster. The truncation parameter value which results in most accurate classification is considered as the optimal value of the parameter.

The optimality of the truncation parameter is confined to a given domain and its optimal value will change from one domain to another. Furthermore, the truncation parameter can be understood as an estimate of the abstraction bias in the overlapping area between classes. The significance of the truncation parameter value is discussed in the next section along with the analysis of classification experiments performed on the ACL system.

Execution Summary. The execution sequence of the above described learning processes can be summarized in the following algorithmic form:

```
/* Use Training Data Set */

While Object Instances are Available to be Described

  Symbol = Get_Object_Label

  If not a Previously Known Symbol Then
        Get_Object_type (Symbol);
  End-If

  Object_Description =
  Get_Object_Description (Symbol)

  /* If a previously unknown symbol is used while
  describing an object instance then use recursion and
  invoke Get_Object_Type (New_Symbol) */

  Object_Description =
  Add_Delete_Menu (Object_Description)

  /* If a previously unknown symbol is used while
  modifying an object description then use recursion and
  invoke Get_Object_Type (New_Symbol) */

  If Exemplar Representation does not Exist Then
      Reduced_Exemplar_Structure =
      Create_New_Exemplar_Representation
      (Object_Description)
  Else
      Reduced_Exemplar_Structure =
      Update_Exemplar_Representation
      (Object_Description)
  End-If

  If Prototype Representation does not Exist Then
      Prototype = Create_New_Prototype_Representation
      (Reduced_Exemplar_Structure)
  Else
      Prototype = Update_Prototype_Representation
      (Reduced_Exemplar_Structure)
  End-If

End While loop

/* Invoke Classification Process */

Generate_Multiple_Prototypes (Training_Set)

/* Use test data set */

Perform_Classification (Test_Data)
Print_Classification_Statistics

End Program
```

Evaluation

As mentioned previously, classification accuracy can be used as a surrogate measure for concept quality. Therefore, to measure the quality of concept descriptions generated by the ACL, the prototype system was used to perform two classification tasks.

Description of Data Sets

A summarized description of data sets used for the classification tasks is shown in Table 2. The detailed description is available on the internet node (ics.uci.edu) which functions as the secondary source for data sets used in machine learning experimentation. We chose these previously used data sets to compare the concept learning performance of the ACL prototype system with the classification results reported by other studies (See Weiss & Kulikowski 1991 and Michalski 1989). The breast cancer data provides an opportunity for the ACL system to work with symbolic attributes while the Iris data set tests the capability of ACL to work with numeric attributes. Furthermore, while working with the cancer data, the ACL system is also challenged with classes exhibiting a high degree of overlap.

Experiments with Breast Cancer Data

The data set for evaluating the prognosis of breast cancer reoccurrence was used for the first classification task. 70% of the test cases were randomly selected for learning concept descriptions and the remaining 30% were used for testing. This process was repeated four times to create four data sets. The technique used to create the training and test data sets is the same as described in Michalski (1989). This was done to replicate the experimental conditions of the studies performed by Michalski. Since other researchers have only reported the average results of the four data sets, the following discussion will also refer to the average performance of the multiple prototype classification model.

As shown in Table 3, the multiple prototype model has a better classification accuracy--0.73 compared to 0.68 for AQ15 and 0.72 for ASSISTANT tree--when truncation was used for all three methods. On the other hand, without truncation, the multiple prototype model performs very poorly. In fact, the system performance goes down with the increase in the number of prototypes used for representing a fixed number of classes. Similarly, performance also deteriorates if only one prototype is used to represent each class. This is because a single prototype represents too much abstraction, while too many prototypes for overlapping classes (like the two classes in the breast data) confounds the classification process.

Table 4 (source: Weiss & Kulikowski 1991) shows classification error results for several different

classification methods. In comparison to other methods, the multiple prototype model ranks third behind the PVM rule and the CART tree. It is also interesting to note that the classification accuracy of human experts for the breast cancer data is only 0.64 and that most classification methods, including the Multiple Prototype model, can deliver a better performance.

Experiments with the Iris Data

The Iris data set was used for the second classification task. As opposed to the train and test procedure used for breast cancer data, the cross-validation method was used for arriving at the classification accuracy. This was done to be consistent with other classification studies that have used the Iris data set.

A classification accuracy of 0.94 or an error rate of 0.06 places the multiple prototype model as sixth among the classification models shown in Table 4. From the results shown in Table 4, it is clear that the performance of multiple prototype classification model is clearly not among the best.

Analysis

The iris and breast cancer data sets were chosen to study the classification performance of ACL prototype on numeric and symbolic attribute types. The comparative performance data shown in Tables 4 indicate that the classification performance of ACL is better for symbolic data, and average for numeric data. This implies that compared to numeric concept descriptions, the quality of symbolic concept descriptions learned by the ACL prototype system is higher.

For best classification results, a suitable value for the truncation parameter used by multiple prototype model must be found. Higher values of the truncate parameter indicate that it may be possible to use abstraction for maintaining information about exemplars belonging to category areas that overlap with another category in a multidimensional space. On the other hand, lower values of the truncate parameter imply that category instances in the overlap area may have to be remembered as individual exemplars with little scope of abstraction.

Classification performance of ACL for a set of user specified truncation parameter values is shown in Table 5. On the basis of these results it may be concluded that a classification model using a hybrid of prototype and exemplar representations may eventually prove to be the best. This viewpoint is similar to the decentralized view of concept learning proposed by Brooks (1987).

It is worth mentioning that attributes associated with each decision class in the cancer or the iris data set are the same. Under these circumstances, a categorization process is limited to the exploitation of similarities and differences among attribute values for predicting a decision class. This is generally true for objects at the subordinate level in an object hierarchy consisting of superordinate, basic, and subordinate levels (see Rosch 1978). On the contrary, at the basic and superordinate levels, attributes associated with each object class form the basis for classification rather than attribute values. Unfortunately, most data sets for which published classification results are available tend to use attribute values as discriminants for decision classes. Due to lack of comparative data, the classification behavior of the multiple prototype model for basic level objects was not investigated.

Comparison of ACL with Related Research

The following salient characteristics of ACL stand in contrast with respect to existing work in the area of concept learning and classification:

1. The concept learning process used by ACL is based on abstraction instead of the traditional inductive generalization techniques (Michalski 1983) used by most of the existing systems (e.g., see Dietterich and Michalski 1983).
2. ACL subscribes to the probabilistic view of concept learning instead of the classical view adopted by concept learning systems based on inductive generalization techniques.
3. ACL uses prototype-based concept representations as opposed to exemplar-based representations used in machine learning systems like PROTOS (Bareiss 1989) and CYRUS (Kolodner 1984), which follow the case-based reasoning paradigm.
4. The technique used for generating multiple prototypes through failure-based classification is quite similar in spirit to the growth algorithm presented in Kibler and Aha (1987). Although similar, the two approaches differ in their output because the growth algorithm retains its classification failures as exemplars whereas ACL attempts to create prototype-based abstractions from its classification failures.
5. Clustering programs like COBWEB (Fisher 1987) and UNIMEM (Lebowitz 1987) are capable of generating a concept hierarchy while ACL at present is limited to a single level of clustering. ACL uses a truncation parameter in conjunction with classification accuracy to optimize the number of resulting clusters. Unfortunately, ACL like COBWEB does not use any operators to mitigate the effect of instance ordering.
6. The ACL system is not limited to either symbolic or numeric attributes but can function with both attribute types simultaneously.

Data Set Name	Decision Classes	Number of Instances	Attributes per Instance	Values / Attribute	Attribute Type
Breast Cancer	2	286	9	5.8	Symbolic
Iris	3	150	4	n/a	Numeric

Table 2. Description of Data Sets

Method		Complexity	Accuracy
		Domain : Breast Cancer	
AQ15			
	Coverage = Complete	Complexes = 41 Selectors = 160	0.66
	Truncation, unique > 1	Complexes = 32 Selectors = 128	0.68
	Top Rule	Complexes = 2 Selectors = 7	0.68
ASSISTANT Tree			
	No Pruning	Nodes = 120 Leaves = 63	0.67
	With Pruning	Nodes = 16 Leaves = 9	0.72
ACL			
	Coverage = Maximal[1]	Prototypes = 23	0.38
	Truncation parameter = 0.20	Prototypes = 3.5	0.73
Human Experts		-	0.64

Table 3 Comparison of Multiple Prototype Classification Model with AQ15 and ASSISTANT (descendant of ID3)[2]

Notes for Table 3:
1. Maximal is equivalent to no truncation
2. Performance data source for AQ15 and Assistant: Michalski (1989)

Method	Err_{cv}^{*} (Iris Data)	Err_{Test}^{*} (Cancer Data)
Linear	0.020	0.294
Quadratic	0.027	0.344
Nearest Neighbor	0.040	0.347
Bayes Independence	0.067	0.282
Bayes 2nd Order	0.160	0.344
Neural Net (BP)	0.033	0.285
Neural Net (ODE)	0.027	0.276
PVM rule	0.040	0.229
Optimal Rule Size 2	0.020	-
CART Tree	0.047	0.229
ACL (Truncation Parameter)	0.060 (0.025)	0.270 (0.200)

*Represents the error rate or rate of misclassification. Lower values indicate better performance.

Table 4. Comparative Performance of Multiple Prototype Classification Model on Breast Cancer and Iris Data

Truncation Parameter	Classification Accuracy	Average Number of Prototypes
Breast Cancer Data*:		
0.05	0.40	13.75
0.10	0.63	6.00
0.15	0.65	4.25
0.20	0.73	3.50
1.00	0.60	2.00
Maximal**	0.38	23.25
Iris Data:		
0.025	0.94	5.96
1.000	0.94	3.00
Maximal**	0.91	12.92

* Reported classification accuracy is the average performance of Multiple Prototype Classification model on 4 test data sets.

** Maximal is equivalent to no truncation.

Table 5. The Affect of Truncation Parameter on Classification Accuracy

Conclusions and Future Work

A major contribution of this paper is the use of abstraction mechanisms and prototype-based concept representations in concept learning tasks. This is a marked departure from the traditional thinking embodied in the design of extant concept learning systems described in machine learning literature. Furthermore, the system (ACL) described in this paper is not limited to well defined concepts but has been designed to function with ill-defined categories.

The concept learning processes incorporated in the ACL system are nonobservable processes that have been postulated by cognitive psychologists to explain their empirical results obtained from concept learning experiments with human subjects. Experiments conducted with the help of ACL system demonstrate that these learning processes may not be the best option in all circumstances. Results from classification experiments indicate that mathematical formulations may fare better when compared to a simulation of human learning processes. It was also found that concept learning processes used by human beings may be more appropriate in problem domains that are characterized by symbolic attributes.

The conclusions outlined in the previous paragraph are based on a limited set of experiments. Further experimentation with the ACL system is required to substantiate the validity of our results. In the future we plan to conduct few more experiments with ACL to explore its classification behavior in other types of artificial and natural domains. In particular, our plan is to compare the classification performance of ACL with human subjects. We also plan to direct our future effort towards the type of experimentation that will help us in refining the definition of truncation parameter and characterize its role in classification tasks.

References

Bareiss, R. 1989. *Exemplar-Based Knowledge Acquisition.* San Diego, CA: Academic Press.

Barsalou, L. W. 1990. On the Indistinguishability of Exemplar Memory and Abstraction in Category Representation. In T. K. Srull and R. S. Wyer (eds.), *Advances in Social Cognition*, Vol. 3. Hillsdale, New Jersey: Lawrence Erlbaum.

Bergadano, F.; Matwin, S.; Michalski, R. S.; and Zhang, J. 1988. Measuring Quality of Concept Descriptions. In Proceedings of the Third European Working Session on Learning. London: Pitman Publishing.

Breen, T. J. and Schvaneveldt, R. W. 1986. Classification of empirically derived prototypes as a function of Category Experience. *Memory and Cognition* 14(4):313-320.

Brooks, L. R. 1987. Decentralized Control of Categorization: The Role of Prior Processing Episodes. In U. Neisser (ed.), *Concepts and Conceptual Development: Ecological and Intellectual Factors in Categorization.* New York: Cambridge Press.

Dietterich, T. G. and Michalski, R. S. 1983. A Comparative Review of Selected Methods for Learning from Examples. In R. S. Michalski; J. G. Carbonell; and T. M. Mitchell (eds.), *Machine Learning: An Artificial Intelligence Approach*, Vol. 1. Palo Alto, CA: Morgan Kaufmann Publishers.

Elio, R. and Anderson, J. R. 1981. The effects of Category Generalizations and Instance Similarity on Schema Abstraction. *Journal of Experimental Psychology: Human Learning and Memory* 7(6):397-416.

Fisher, D. 1987. Knowledge Acquisition via Incremental Conceptual Clustering. *Machine Learning* 2:139-172.

Homa, D. and Chambliss, D. 1975. The Relative Contributions of Common and Distinctive Information on the Abstraction from Ill-defined Categories. *Journal of Experimental Psychology: Human Learning and Memory* 104(4):351-359.

Homa, D. and Vosburgh, R. 1976. Category Breadth and the Abstraction of Prototypical Information. *Journal of Experimental Psychology: Human Learning and Memory* 2(3):322-330.

Homa, D., Sterling, S. and Trepel, L. 1981. Limitations of Exemplar-Based Generalization and the Abstraction of Categorical Information. *Journal of Experimental Psychology: Human Learning and Memory* 7(6):418-439.

Homa, D.; Cross, J.; Cornell, D.; Goldman, D.; and Shwartz, S. 1973. Prototype Abstraction and Classification of New Instances as a Function of Number of Instances Defining the Prototype. *Journal of Experimental Psychology* 101(1):116-122.

Kibler, D. and Aha, D. W. 1987. Learning Representative Exemplars of Concepts: An Initial Case Study. In Proceedings of the Fifth International Workshop on Machine Learning, 24-30. San Mateo, CA: Morgan Kaufman.

Kolodner, J. L. 1984. *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Lebowitz. M. 1987. Experiments with Incremental Concept Formation: UNIMEM. *Machine Learning* 2:103-138.

Medin, D. L. and Florian, J. E. 1992. Abstraction and Selective Coding in Exemplar-Based Models of Categorization. In A. F. Healy; S. M. Kosslyn; and R. M. Shiffrin (eds.), *From Learning Processes to Cognitive Processes*, Vol. 2. Hillsdale, NJ: Lawrence Erlbaum Associates.

Mervis, C. B. and Rosch, E. 1981. Categorization of Natural Objects. *Annual Review of Psychology* 32:89-115.

Michalski, R. S. 1989. Learning Flexible Concepts: Fundamental Ideas and a Method Based on Two-Tiered Representation. In Y. Kodratoff and R. S. Michalski (eds.), *Machine Learning: An Artificial Intelligence Approach*, Vol. 3. Palo Alto, CA: Morgan Kaufmann.

Michalski, R. S. 1994. Inferential theory of Learning: Developing Foundations for Multistrategy Learning. In R. S. Michalski and G. Tecuci (eds.), *Machine Learning: A Multistrategy Approach*, Vol. 4. Palo Alto, CA: Morgan Kaufmann.

Mitchell, T. 1982. Generalization as Search. *Artificial Intelligence* 18:203-226.

Posner, M. I. and Keele, S. W. 1970. Retention of abstract ideas. *Journal of Experimental Psychology* 83:304-308.

Posner, M. I. and Keele, S. W. 1968. On the genesis of abstract ideas. *Journal of Experimental Psychology* 77:353-363.

Quinlan, J. R. 1986. Induction of Decision Trees. *Machine Learning* 1:81-106.

Reed, S. 1972. Pattern Recognition and Categorization. *Cognitive Psychology* 3:382-407.

Rohatgi, M. 1994. A human Learning Approach For Designing Adaptive Knowledge-Based Systems. Ph.D. diss., Dept. of Information Systems, Texas Tech University.

Rosch, E. (1978). Principles of Categorization. In E. Rosch and B. B. Lloyd (eds.), *Cognition and Categorization.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R.C.; Collins, G.C.; and Hunter, L.E. 1986. Transcending Inductive category formation in learning. *Behavioral and Brain Sciences* 9:639-686.

Smith, E. E. and Medin, D. L. 1981. *Categories and Concepts*. Cambridge, MA: Harvard University Press.

Weiss, S. M. and Kulikowski, C. A. 1991. *Computer Systems that Learn.* Palo Alto, CA: Morgan Kaufmann.

Winston, P.H. 1975. Learning structural descriptions from examples. In P.H. Winston (ed.), *The Psychology of Computer Vision.* New York: McGraw Hill.

III. Methods and Systems

Multistrategy Task-adaptive Learning Using Dynamically Interlaced Hierarchies: A Methodology and Initial Implementation of INTERLACE

Nabil W. Alkharouf and Ryszard S. Michalski*

Machine Learning and Inference Laboratory
George Mason University,
Fairfax, Virginia, 22030, USA
{nabil, michalski}@aic.gmu.edu

* Also with GMU Departments of Computer Science and Systems Engineering, and the Institute of Computer Science, Polish Academy of Sciences

Abstract

This research concerns the development of a methodology for representing, planning and executing multitype inferences in a multistrategy task-adaptive learning system. These inferences, defined in the Inferential Theory of Learning as *knowledge transmutations*, are generic types of knowledge operators, and are assumed to underlie all learning processes. The paper shows how several basic knowledge transmutations can be seamlessly integrated using a knowledge representation based on *dynamic interlaced hierarchies* (DIH). The implemented system, INTERLACE, includes an interactive graphical user interface for visualizing knowledge transmutations that are being performed by the system. INTERLACE is illustrated by several examples.

Introduction

The development of a multistrategy learning system capable of integrating different learning strategies according to the needs of a given learning task requires a knowledge representation that facilitates multitype inferences. The Inferential Theory of Learning (Michalski 1993), which provides a theoretical framework for multistrategy learning, views every learning process as a goal-oriented search through a knowledge space. In this search, the operators are instantiations of generic forms inference, called *knowledge transmutations*, the search goal is a specification/characterization of knowledge to be acquired, and the starting point is knowledge the system already possesses. A knowledge transmutation takes some input knowledge (e.g., a training example(s), a general statement, an observation, etc.), a relevant part of the learner's prior knowledge, and produces a new piece of knowledge (e.g., a generalization of examples, a consequence of the given statement, an explanation of the observation, etc.). Knowledge transmutations can be classified on the basis of the type of inference they employ and the type of knowledge they produce.

This paper describes a methodology and an initial implementation of the system INTERLACE, for supporting different knowledge transmutations and applying them according to the *learning task*. A learning task is determined on the basis of the input knowledge provided to the learner, the learner's prior knowledge, and a learning goal. The methodology employs a knowledge representation,, called *Dynamically Interlaced Hierarchies* (DIH), which builds upon research on human plausible reasoning (Collins & Michalski 1989), frame representation (Minsky 1975) and semantic networks (Quillian 1968). The DIH representation supports a seamless execution of a wide range of knowledge transmutations, and provides a backbone for implementing a Multistrategy Task-Adaptive Learning (MTL) system (Michalski 1993).

The DIH representation provides an expressive, modifiable and flexible knowledge representation system. The initial efforts in this direction were described in (Hieb & Michalski 1993). The DIH representation employs *part*, *type* and *precedence* hierarchies for representing hierarchical conceptual structures, and *traces* for representing statements involving concepts from different hierarchies.

INTERLACE combines the DIH representation with a control mechanism for executing knowledge transmutations in a seamless integrated manner, according to a given knowledge goal. A learning process is then a goal-directed

transformation of knowledge that is carried out through inferential processes embodied in knowledge transmutation operators.

The INTERLACE system consists of three interacting modules, a knowledge base for DIH hierarchies, traces and transmutations, a goal-driven planner for generating plausible sequences of transmutations in order to achieve a given goal, and a graphical user interface for visualizing knowledge transmutations. The implemented system includes capabilities for such transmutations as inductive generalization, deductive specialization, abstraction and similization, and their counterparts, deductive generalization, inductive specialization, concretion and dissimilization. It also allows for the graphical creation, modification and deletion of DIH hierarchies and traces.

Related Research

One of the important aspects of the Inferential Theory of Learning (ITL) is that provides an approach to analyzing and describing the *competence* of learning strategies, that is, their logical capabilities. Theory views every learning process as a search through a *knowledge space* defined by the employed knowledge representation, and guided by a learning goal. The search operators, instantiations of transmutations, represent different types of inference or knowledge transformation. Basic knowledge transmutations include generalization, abstraction, explanation, selection, association, similization, agglomeration, characterization, and their counterparts, specialization, concretion, prediction, generation,, disassociation, dissimilization, decomposition and discrimination (Michalski 1994).

According to ITL, to instill learning capabilities in a multipurpose intelligent agent, one needs to implement in it the ability to conduct multiple types of inference (in general, any type of knowledge derivation or transformation) and the ability to store and retrieve knowledge. This observation forms the basic premise of the theory, which can be succinctly stated in the following "equation":

Learning = Inference + Memory

This theory is conceptually close to Newell's ideas of about knowledge-level systems (Newell 1994). ITL abstracts from the medium that transforms knowledge, and concentrates on the type of knowledge transformations that occur in learning processes. Therefore, it can be used, in principle, for analyzing any type of learning. In this paper, ITL is used as a conceptual basis for implementing a *multistrategy task adaptive learning*.

Research on multistrategy task-adaptive learning (MTL) systems aims at synergistically integrating a wide range of learning strategies in order to perform different learning tasks. A learning task is defined by the available input to the learning system, learner's prior knowledge, nd a learning goal. The proposed system is called task-adaptive, because it is supposed to dynamically combine different learning strategies so that it can perform a given learning task (Michalski 1993, 1994; Michalski & Ram 1995). The system is inherently goal-dependent, because it does not assume that the learning process is just a search for a generalization of examples, as often done in machine learning, but can, in principle, search for any kind knowledge that is desirable by the learner (Michalski 1994).

Research on task-adaptive learning is conceptually related to research on integrated learning, (e.g., Bergadano, Giordana & Saitta 1991), and goal-driven learning (e.g., Hunter 1990; Leake & Ram 1995; Ram & Leake 1995). In general, multistrategy learning systems range from *loosely coupled* systems that consist of several learning modules coordinating to achieve the learning task to *tightly coupled* systems that deeply and synergistically integrate inferential learning strategies in order to achieve the desired learning performance. Examples of tightly ("deeply") integrated learning systems include the discussed learning system (MTL-DIH) and a related system based on plausible justification trees (Tecuci 1993).

Figure 1 presents a general schema of a multistrategy task-adaptive learning system based on DIH knowledge representation (a MTL-DIH system). The learning task is defined by the input DIH traces, relevant background knowledge in the form of DIH hierarchies and existing traces through the hierarchies, and a learning goal (in general, it can be a set of goals). A multitype inference engine utilizes DIH transmutations as knowledge generation operators under the supervision of a control and planning subsystem.

Dynamically Interlaced Hierarchies

The idea of the knowledge representation based on Dynamically Interlaced Hierarchies stems from the observation that some knowledge is more or less stable and other knowledge is acquired incrementally using the stable knowledge as hooks. In (Collins & Michalski 1989), this "stable" knowledge is represented by type and part hierarchies. Initial ideas and a method for the implementation of Dynamically Interlaced Hierarchies (DIH) was presented in (Hieb & Michalski 1993). The DIH representation is conceptually related to semantic networks (Quillian 1968), but is has some important distinctive features, designed to facilitate a conceptual simplicy of representation and performing diverse forms of inference in a uniform way.

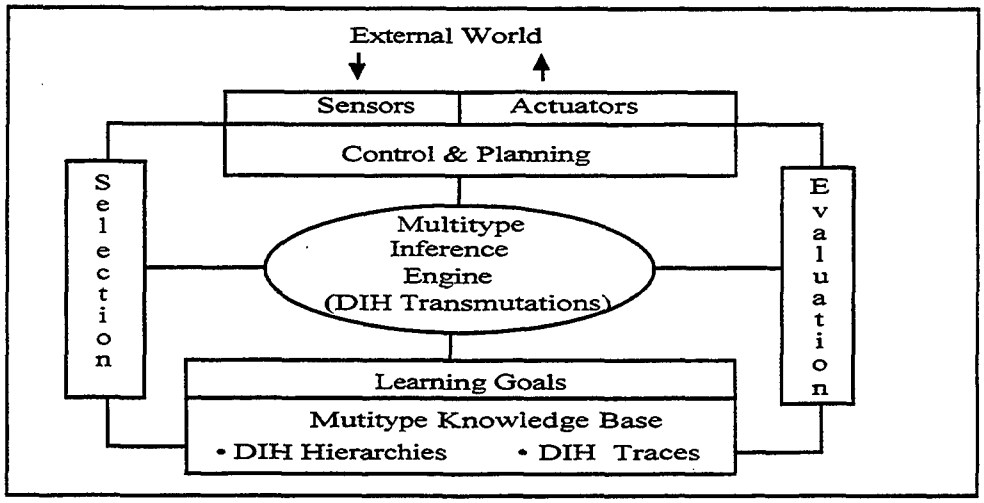


Figure 1. A general schema for a multistrategy task-adaptive learning system based on the DIH knowledge representation.

In DIH knowledge is partitioned into a *static* and *dynamic* part. The static part represents relatively stable knowledge organized into *concept hierarchies* such as part, type and precedence hierarchies, each hierarchy organizes its concept nodes from a certain context or viewpoint, with the possibility of concept nodes participating in more than one hierarchy. The dynamic part represents knowledge that changes relatively frequently. This knowledge is represented by *traces* that are paths linking nodes of different hierarchies in order to represent single statements.

DIH provides a suitable knowledge representation for facilitating multitype inference in multistrategy learning systems that require an expressive, modifiable and flexible representation language. Initial work on the DIH representation was presented in (Hieb & Michalski 1993).

INTERLACE

INTERLACE is the core of the MTL-DIH multistrategy learning system. It consists of three interacting modules — a knowledge base, a goal driven planner and a graphical user interface. To implement INTERLACE, we needed a language that would facilitate the construction of concept hierarchies and making links among them. For this purpose we chose LIFE programming language, originally conceived at MCC.

LIFE is a synthesis of three different programming paradigms: logic programming, functional programming, and object-oriented programming. It is a declarative logic-based language that can be seen as a constraint language. It derives its syntax and resolution method from Prolog. It uses λ-terms as its basic data structures, and unification and matching as its basic operations.

LIFE semantically extends PROLOG in two ways:

1) Herbrand terms are replaced by λ-terms.

2) It includes *call by matching* (Prolog only supports call by unification).

As a programming language LIFE provides clean and elegant solutions to a number of Prolog's deficiencies. It includes functions (correct arithmetic), object-orientation, C-like records, expandable data-structures (arrays and hash-tables), types and multiple inheritance, correct manipulation of cyclic structures, coroutines and constraints, global variables, clean destructive assignment, persistent data structures.

Representation of DIH Hierarchies and Traces

As mentioned earlier, DIH hierarchies represent relatively stable background knowledge. Such knowledge consists of concepts organized into hierarchies of different kinds, such as type, part and precedence hierarchies. A DIH node is represented in LIFE as a sort, an identifier standing for a set of objects. A sort corresponds intuitively to a type in a traditional programming language, or a class in an object-oriented programming language.

Sorts are organized into a hierarchy through a set of sort declarations (using the <| operation). For example, the navy_aircraft type hierarchy in Figure 2 is declared in the following manner:

```
combat_aircraft <| navy_aircraft
fuel_aircraft <| navy_aircraft
service_aircraft <| navy_aircraft
f16 <| combat_aircraft
tomcat <| combat_aircraft
```

DIH traces represent "dynamic" knowledge that changes relatively frequently. A trace roughly stands of a statement, and is a path connecting nodes of different hierarchies. DIH traces are represented using λ-terms, the LIFE's basic data structure. Each λ-term has the following syntax:

root_sort (label_1 => sort, label_2 => sort, ...)

Each sort including the root sort is a *part of* a hierarchy. Each sort can be further refined using the hierarchy that is also a *part of* type. This forms a basis for the functionality of DIH transmutations described below. For example, the DIH trace expressing the statement "Some aircraft carriers in Norfolk have fuel aircraft" is represented as:

aircraft_carrier(quantifier => some, location =>
norfolk, navy_aircraft => fuel),

where *aircraft_carrier* is the trace argument (root sort), *quantifier, location, and navy_aircraft* are viewed as descriptors applied to the trace argument. Adding knowledge to the DIH representation is done by creating hierarchies and specifying traces that express statements involving nodes of different hierarchies.

Representation of DIH Transmutations

We will show now that several basic knowledge generation transmutations, as described in the Inferential Theory of Learning (Michalski 1991, 1992, 1993), can be easily performed in DIH, specifically, just by moving some links. The transmutations include generalization, abstraction, explanation and similization, and their counterparts, specialization, concretion, prediction and dissimilization, respectively. The DIH system currently implements six transmutations:

QGEN: Quantification-based Generalization
QSPEC: Quantification-based Specialization
AGEN: Argument-based Generalization
ASPEC: Argument-based Inductive Specialization
ABST: Abstraction
CONC: Concretion

By repeating these transmutations in different combinations, the system can perform a wide range of inferences. Every transmutation has the following syntax:

TransType(I_TRACE, CONTEXT, O_TRACE),

where I_trace is an input trace, and O_trace is an output trace (a trace generated by a transmutation).

An inference step performed by a transmutation can be depicted graphically as a movement of traces along some nodes of the hierarchies. To illustrate specific transmutations (operators) performed by INTERLACE, suppose that given is an input statement: "Some Norfolk aircraft carriers are fuel carriers."

This statement is represented by a trace:

I_TRACE: aircraft_carrier (quantifier => some, location => norfolk, navy_aircraft => fuel).

The order of arguments for the trace interpretation is defined by a schema hierarchy, SH, shown in Figure 2.

For the given input, the **QGEN** operator (Quantification-based Generalization) produces a trace:

O_TRACE: [aircraft_carrier (quantifier => most, location => norfolk, navy_aircraft => fuel)] (Figure 2)

Given the same input, the **QSPEC** operator (Quantification-based Specialization), produces:

O_TRACE = [aircraft_carrier (quantifier => one, location => norfolk, navy_aircraft => fuel)].

Again, given the same input, the **AGEN** transmutation (Argument-based Generalization) produces:

O_TRACE=[navy_battleship(quantifier=>some, location => united_states, navy_aircraft => fuel)] (Figure 2).

ASPEC (Argument-based specialization can produce one of several alternatives, depending on *merit parameters* (Collins & Michalski 1989). One of them (Figure 3):

O_TRACE = [iowa_class(quantifier => some, location => united_states, navy_aircraft => fuel)].

ABST (Abstraction) produces:

O_TRACE = [aircraft_carrier(quantifier => some, location => united_states, navy_aircraft => fuel)] (Figure 4).

CONC (Concretion), another form of inductive inference, produces (among other plausible alternatives):

O_TRACE = [aircraft_carrier(quantifier => some, location => united_states, navy_aircraft => b677) (Figure 5).

In principle, all the above transmutaions can represent deductive or inductive inference, depending on the combination of nodes in a trace and on what nodes are moved (Michalski 1994).

Goal-Driven Learning based on DIH

The idea behind the DIH-based goal-driven learning is to apply a sequence of transmutations that transfer a given input (one or more traces) into one or more output traces that satisfy a learning goal. A learning goal is specification of knowledge that the learner wants to aquire. A learning goal is generated by a performance system, for example, as a request for some knowledge, as a result of an impass in problem solving (Newell 1994), etc. A goal specification is expressed as a trace. A goal trace may include high level nodes whose children are to be learned, or nodes indicating the type of knowledge to be acquired. Learning goals can be classified into two categories: domain specific and domain independent (Michalski & Ram 1995). Learning in INTERLACE system is designed to handle both domain specific and domain independent goals.

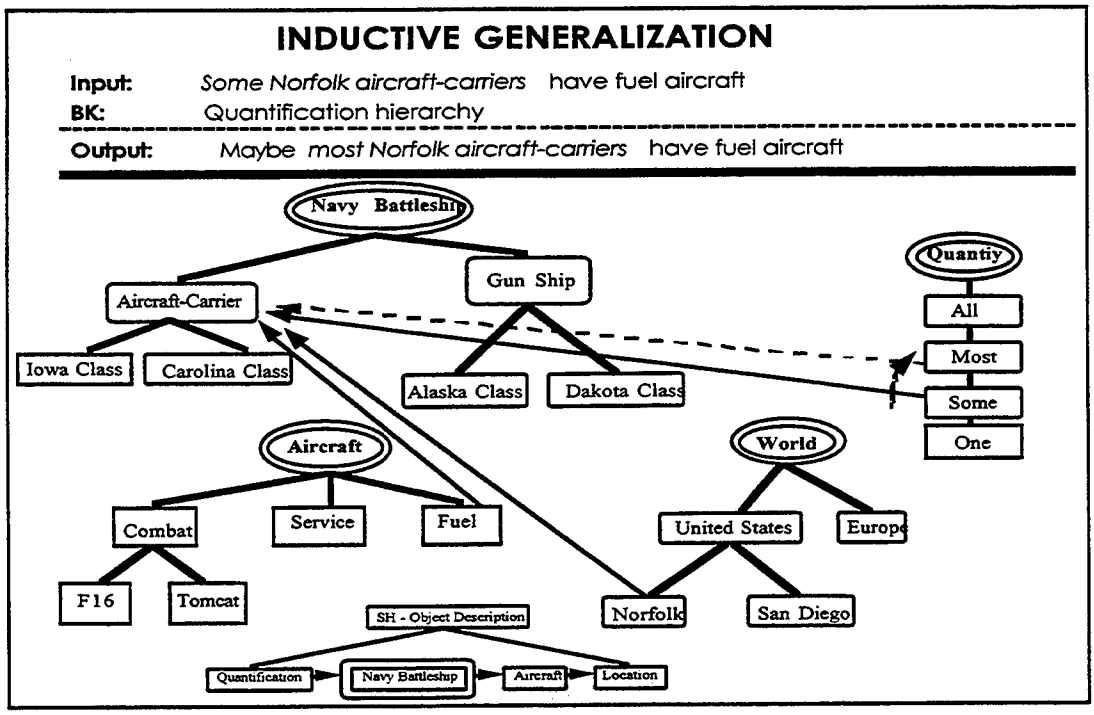


Figure 2. An example of a quantification-based inductive generalization.

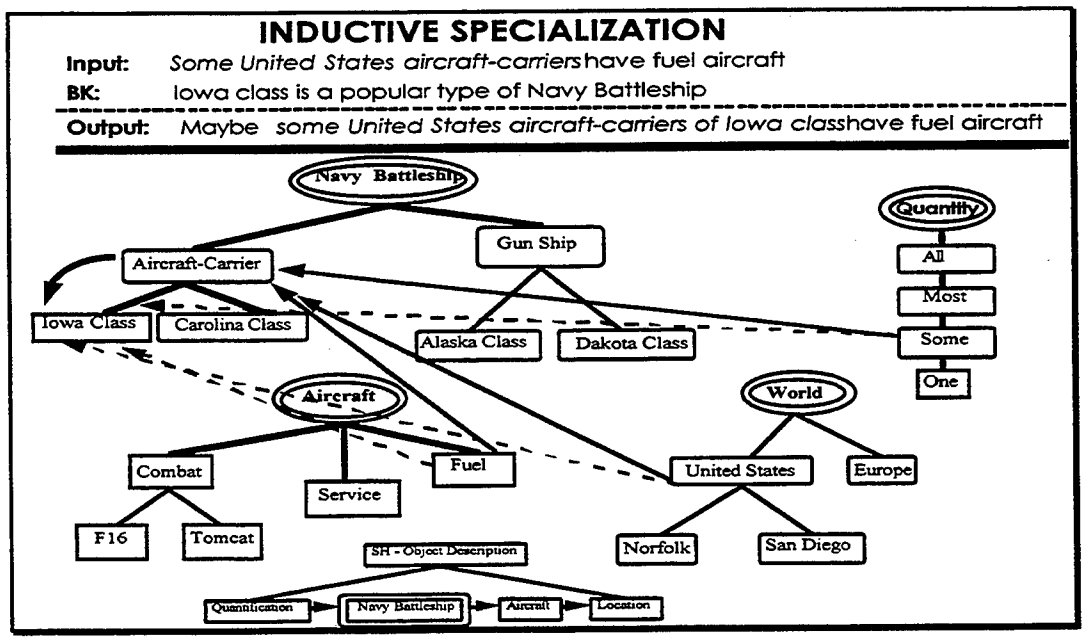


Figure 3. An example of an argument-based inductive specialization.

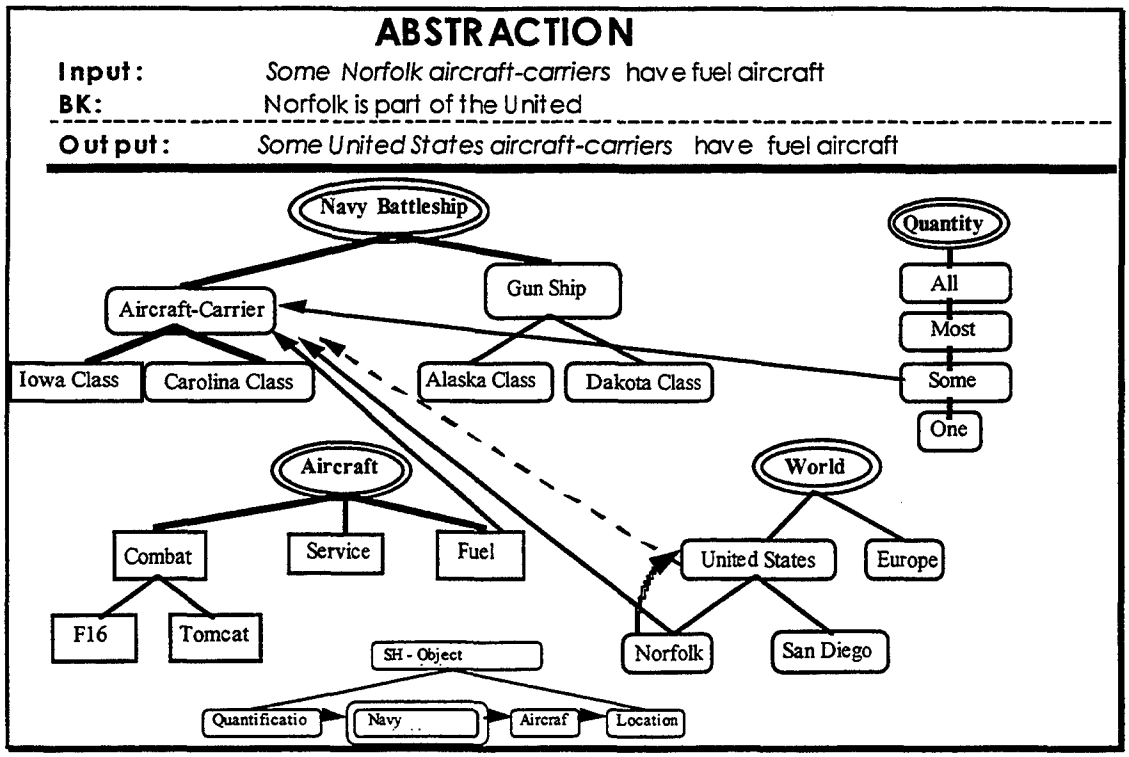


Figure 4. An example of abstraction transmutation.

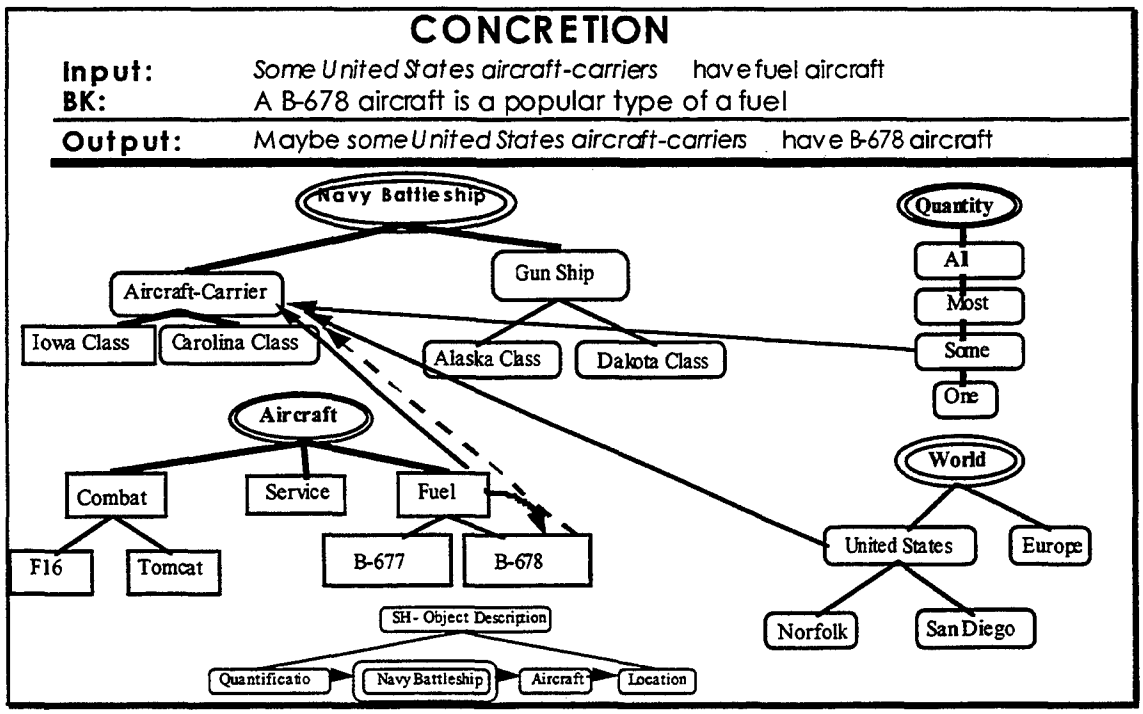


Figure 5. An example of concretion transmutation.

Domain Specific Goals. Domain specific goals define knowledge that is relevant in the given domain. The task of the planner is to generate a sequence of transmutations that produce knowledge that satisifies the given goal.

Such a process is invoked in INTERLACE by:

Plan(I_TRACES, G_TRACES, CONTEXT, PLAN),

where

I_TRACES — a list of relevant input traces

G_TRACES — a goal trace (or a list of traces).

CONTEXT — a context for the process, specified by a list of hierarchies within which the transmutations are to be performed)

PLAN — a list of plausible transmutations that proof the G_TRACES.

The planner determines which transmutation is appropriate for a given goal in the given contex Deductive transmutations are favored over inductive transmutations during the planning process.

To illustrate the process, let us consider: I_TRACES = [Iowa_class (quantifier => one, location => United_States, aircraft_type => tomcat)], and G_TRACES = [aircraft_carrier (quantifier => most general, location => norfolk, aircraft_type => combat)]. The planning process produces the following sequence of plausible transmutations to achieve the goal: argument-based deductive deneralization from Iowa_class to aircraft_carrier, abstraction from tomcat to combat, two inductive quantification-based generalizations — from one to some, and then to most, and an inductive concretion from United_states to Norfolk.

Domain Independent Goals. Domain independent goals require the system to determine some general type of knowledge, e.g., a generalization of examples, a hierarchical classification of given facts, an explanation for an observation, etc.). Such goals are handled by designated procedures. In the absence of a more specific goal, the system executes the universal learning goal — derive all plausible knowledge that can be learned from a given input. The universal goal is executed by analyzing the relationship between an input trace and background knowledge, and performing steps dependent on the result of this analysis, as illustrated in Figure 6 (Michalski 1994).

INTERLACE Graphical Interface

An interactive graphical interface using OSF MOTIF (Johnson 1993; Young 1994) was utilized for the creation, modification and deletion of DIH hierarchies and traces, all actions performed by the user in the graphical interface are reflected in the underlying knowledge base. The interactive graphical interface has the following features:

Hierarchy Creation. Create a node anywhere on the screen. Two types of arcs are defined: Undirected and Directed. Each of the above arcs can be in solid line style or on-off dash style. Hierarchies are created by using undirected solid line style arcs. By a repeated application of "create an arc" between two nodes, one can create a hierarchy. The system allows a user to create multiple hierarchies within a screen.

Trace Creation. Traces are created by using directed on-off dash arcs connecting nodes from separate hierarchies. The tip of the directed on-off dash arc designates the trace argument. The tail of the directed on-off dash arc designates the referent.

Hierarchy Movement and Orientation

- Hierarchies can be viewed vertically and horizontally.
- Hierarchies can be centered on the screen
- Hierarchies can be selected and moved around on the screen
- A single node within a hierarchy can be moved around.
- Arcs can be moved to reflect new relationships between two nodes.
- Subhierarchies can be selected and moved around.

Hierarchy and Trace Deletion

- Delete a single selected node, this in turn deletes any arcs associated with the deleted nodes
- Delete a set of selected nodes, this in turn deletes any arcs associated with the deleted nodes
- Delete an arc
- Delete selected arcs
- Delete the whole hierarchy

Conclusion

This paper described an ongoing research on the design and implementation of a task-adaptive multistrategy learning system based on the DIH knowledge representation. The DIH representation was shown to be very useful for performing a wide range of knowledge generation transmutations that are required in multistrategy task-adaptive learning.

Among the most important objectives for future research is how to represent different kinds of goals, and how to determine most appropriate transmutations in pursuing the given learning goal.

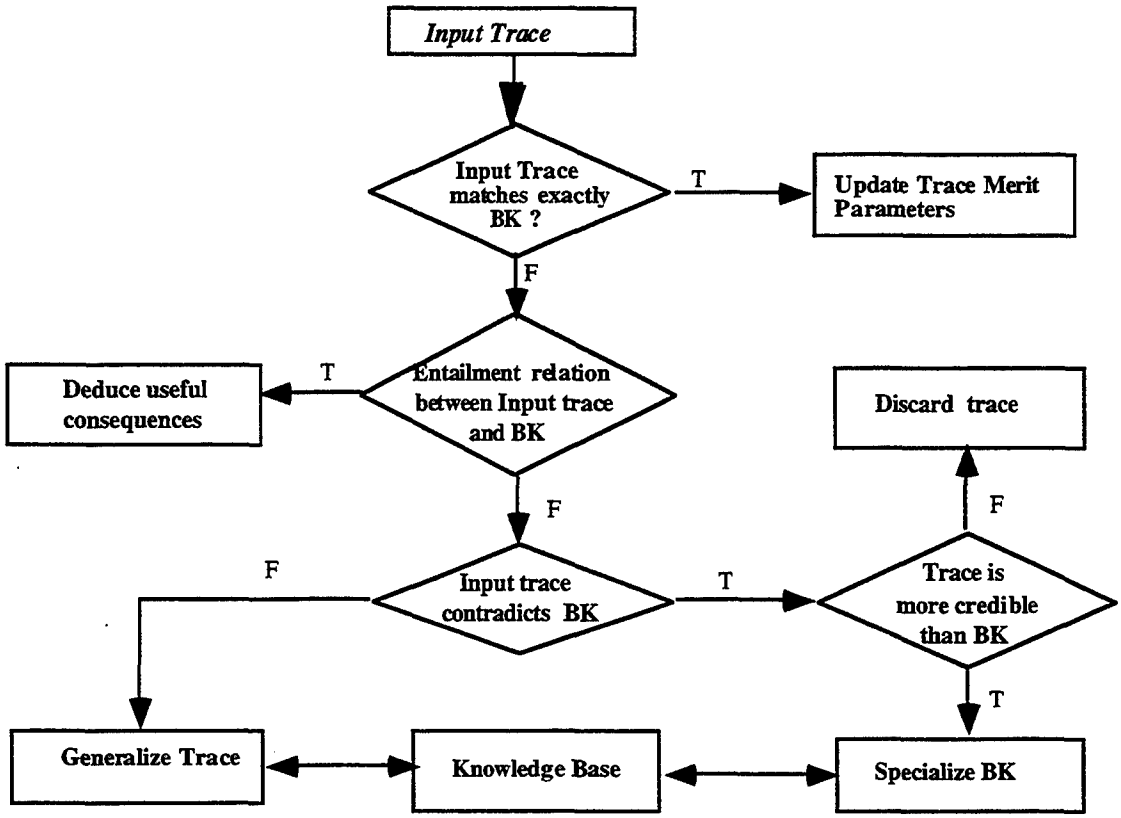


Figure 6. A simplified schema for multistrategy learning pursuing the universal learning goal.

Acknowledgments

This research was done in the Machine Learning and Inference Laboratory at George Mason University. Authors thank members of the laboratory for valuable discussions and criticism.

The Laboratory's research is supported in part by the National Science Foundation under grants DMI-9496192 and IRI-9020266, in part by the Office of Naval Research under grant N00014-91-J-1351, in part by the Defense Defense Advanced Research Projects Agency under grant No. N00014-91-J-1854 administered by the Office of Naval Research, and in part by the Defense Advanced Research Projects Agency under grants F49620-92-J-0549 and F49620-95-1-0462, administered by the Air Force Office of Scientific Research.

References

Bergadano, F., Giordana, A. and Saitta 1991. *Machine Learning: An integrated Framework and its Applications*, Ellis Horwood.

Collins A. and Michalski, R.S. 1989. Logic of Plausible Reasoning: A Core Theory, Cognitive Science, vol. 13.

Hieb, M., Michalski, R.S. 1993. Knowledge Representation for Multistrategy Task-Adaptive Learning: Dynamically Interlaced Hierarchies, In Michalski R.S. and Tecuci, G. (Eds.), *Proceedings of the Second International Workshop on Multistrategy Learning*, published and distributed by Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA.

Johnson-Laird, P. 1983, *Mental Models*, Harvard University Press.

Combining Symbolic and Numeric Methods for Learning to Predict Temporal Series

M. Botta and A. Giordana
Dipartimento di Informatica, Università di Torino,
C.so Svizzera 185, 10149 Torino, Italy
e-mail:{botta,attilio}di.unito.it

Abstract

Radial Basis Function Networks are universal function approximators which can be easely constructed from rule sets learned by a symbolic learner. This paper proposes the use of a more expressive concept description language, based on first order logics, and of a learning system (FLASH) working in such an environment, in order to incorporate the feature selection phase into the learning process. The main advantage of the first order description language used by FLASH, is that it allows to define and manipulate a large number of features while keeping simple the description of instances. The method is tested on two non trivial problems of functional regression represented by the Mackey-Glass temporal series and by the control function for a manipulation robot. The obtained results demonstrate the high accuracy of the proposed method.

Introduction

After an initial period, in which the symbolic and the connectionist approaches to machine learning have been seen in counterposition, several researchers begun to work in the direction of an integrated approach that would take advantage of the specific benefits that each one of them offers. On one hand, the connectionist approach is easier to be implemented, is more suitable to deal with continuous domains and more robust with respect to noisy data. On the other hand, in the original proposal (Rumelhart & McClelland 1986), a neural network is basically a black box that makes it difficult to extract the knowledge it encodes. In this sense, the symbolic approach seems to be better because the learned knowledge is immediately readable and easy to integrate with possibly existing one.

However, in the last years it has been shown that, after all, a neural network is not so opaque as it was assumed to be and, then, it is possible to encode a priori knowledge into a neural net and, viceversa, to extract symbolic knowledge from it. For instance, Shawlik and Towell (Towell, Shavlik, & Noordwier 1990; Towell & Shavlik 1994) provided an algorithm, called KBANN, for transforming a propositional theory into a multilayer perceptron which can be trained by backpropagation. Again Shavlik and Towell (Towell & Shavlik 1993) and, later on, Craven and Shavlik (Craven & Shavlik 1994) presented algorithms for mapping back a multilayer perceptron into a propositional theory. Similar results have been obtained by Tresp (Tresp, Hollatz, & Ahmad 1993) and by Baroglio and others (Baroglio, Giordana, & Piola 1994; Baroglio *et al.* 1996) using another kind of neural network, called Radial Basis Function Networks (RBFNs), introduced by Moody and Darken (Moody & Darken 1988) and mathematically formalized by Poggio and Girosi (Poggio & Girosi 1990). In particular, RBFNs have a structure which immediately matches a flat propositional theory, so that algorithms for mapping symbolic knowledge into a network and viceversa become extremely simple. However, it is worth noting that RBFNs are members of a broader family of networks, including Fuzzy Controllers (Mamdani & Assilian 1975), Probabilistic Neural Networks (Specht 1988) and others, which also benefit from an immediate symbolic interpretation.

The experience with RBFNs, shows that their underlying architecture is excellent for combining symbolic and connectionist learning methods while maintaining the same accuracy available with other kinds of networks, such as multilayer perceptrons. In particular, symbolic algorithms can be advantageously exploited in order to construct a network layout, whereas numeric algorithms, such as the popular Δ-rule, can be used to finely tune numeric coefficients in the network.

This paper continues the research line of Baroglio and others (Baroglio, Giordana, & Piola 1994; Baroglio *et al.* 1996), where decision and regression trees were proposed for constructing RBFN's layouts, and explores a way of using symbolic algorithms designed for learning in first order logics, such as (Michalski 1983; Pazzani & Kibler 1992; Botta & Giordana 1993). The

aim is to incorporate in the learning process large part of the feature selection phase, which, when propositional algorithms are used, takes place in a preprocessing step. As described in the following, this new approach has been applied using a modified version of SMART+ system (Botta & Giordana 1993), called FLASH, and obtained better results than any other method, in learning to predict temporal sequences such as the Mackey-Glass chaotic time series and the control function in a robot control problem. In general, the approach seems feasible and appropriate to regression tasks involving large databases as it frequently happens in data mining applications.

The paper is organized as follows: firstly, we introduce the RBFNs and discuss how they can be constructed using a symbolic algorithm and then refined by performing the error gradient descent. Then, the benefits of using a First Order logic learner are analyzed. Afterwards, we provide some details about the experimental algorithm FLASH and presents the results obtained on the two case studies mentioned above. Finally, some conclusions are drawn.

Learning RBFNs

Given a function $f(\vec{x})$ defined in a vectorial space $\mathbf{X}$, a Radial Basis Function approximates $f(\vec{x})$ as a weighted sum:

$$\hat{f}(\vec{x}) = \frac{\sum_j w_j r_j(\vec{x})}{\sum_j r_j(\vec{x})} \qquad (1)$$

where $r_j(\vec{x})$ is a n-dimensional radial function defined in $X^{(n)}$ and w_j is a real coefficient (weight)[1].

By requiring in (1) that the functions r_j be of the form:

$$r_j(\vec{x}) = \prod_i \mu_{ij}(x_i) \qquad (2)$$

being μ_{ij} unidimensional bell-shaped functions, we obtain a Factorizable Radial Basis Function Network (F-RBFN) as described in (Poggio & Girosi 1990). Through the paper we will make use of RBFNs of this type because better suited to encode a set of propositional rules, as it will be explained in the following.

F-RBFNs can be represented as a 4-layer network (see Figure 1) with two hidden layers.

The neurons in the first hidden layer are feature detectors, each associated to a single unidimensional radial function $\mu_{ij}(x_i)$, and are connected to a single input only. Using Gauss's functions, as it is done through

[1]The canonical form for RBFNs is $\hat{f}(\vec{x}) = \sum_j w_j r_j(\vec{x})$. Nevertheless in this paper we will prefer form (1), also used by some authors, because produces more accurate approximations.

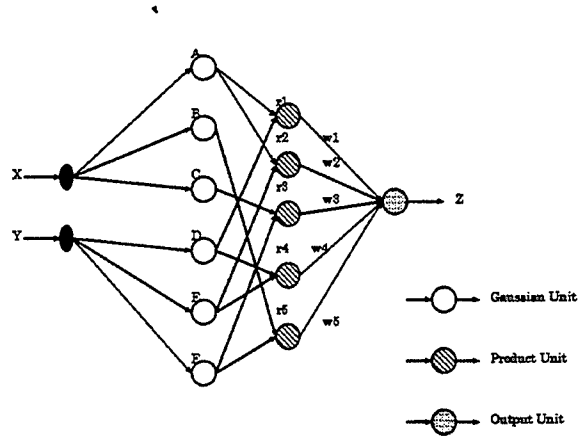


Figure 1: Reference F-RBFN architecture. The first hidden layer units have a unidimensional Gaussian activation function. The second hidden layer units compose the input values computing a product. An output unit performs the normalized weighted sum of the activation values received from product units.

this paper, each neuron μ_{ij} computes the output:

$$\mu_{ij}(x_i) = e^{-\left(\frac{x_i - c_{ij}}{\sigma_{ij}}\right)^2} \qquad (3)$$

where c_{ij} is the center of the Gauss's function along the x_i axis and the σ_{ij} is the width. The neurons in the second hidden layer simply compute a product and construct a multi-dimensional radial function according to expression (2). Finally, the output neuron combines the contributes of the composite functions from the second hidden layer computing expression (1).

The usual methods for learning from examples Radial Basis Function networks (Poggio & Girosi 1990) are based on a two step learning procedure. First a statistical clustering algorithm, such as *k-Means* is used to determine the centers and the amplitude of the activation functions. Then, the weights on the links to the output neuron are determined by computing the coefficients of the pseudo-inverse matrix (Poggio & Girosi 1990) or, alternatively, by performing the error gradient descent by means of the Δ-rule.

Using continuous radial functions that are derivable in the whole parameter space, it is immediate to apply the classical error gradient descent technique in order to finely tune not only the weights w_j but also the parameters c_{ij} and σ_{ij} in the first hidden layer units. Let E be the quadratic error evaluated on the learning set and let, moreover, λ_k indicate a generic parameter in the network, such as, a weight w on a link, the width σ or the center c of a Gauss activation function; all the necessary derivatives can be easely computed, and the

learning rule takes the form:

$$\Delta\lambda_k = -\eta\frac{\partial E}{\partial\lambda_k} \tag{4}$$

The method based on the pseudo-inverse matrix is usually faster than the gradient descent. On the other hand, this last is sometime preferable, because simpler to implement and suitable for on-line learning.

An alternative method for constructing a RBFN is to use a symbolic learning algorithm (Baroglio *et al.* 1996; Tresp, Hollatz, & Ahmad 1993). This becomes particularly simple in the case of F-RBFNs. Approximating each unidimensional Gauss's function μ_{ij} by means of a segment A_{ij}, of length $2\sigma_{ij}$, and interpreting the product in the second layer hidden unit as a logical *AND*, a F-RBFN can be approximated by a set of propositional clauses of the type:

$$\begin{aligned} R_j = &\mathit{member}(x_1, A_{1j}) \wedge \mathit{member}(x_2, A_{2j}) \\ &\wedge \ldots \wedge \mathit{member}(x_n, A_{nj}) \rightarrow w_j \end{aligned} \tag{5}$$

Rules of this type can be easely learned using an algorithm for inducing decision trees, such as ID3 (Quinlan 1979; Sammut *et al.* 1992) or, better an algorithm for inducing regression trees, such as CART (Breiman *et al.* 1984). Owing to the symbolic approximation property, factorizable radial functions as well as the group of corresponding hidden units will be simply referred to as *rules*, in the following.

Approaching Functional Regression in First Order Logics

A regression problem can be easely mapped into a classification problem, by approximating the co-domain of the target function $f(\vec{x})$ with a set W of discrete values. Then, each value $w \in W$ can be considered as a class for which a discriminant definition has to be learned (Baroglio *et al.* 1996). As the learning problem is set in the framework of the propositional calculus, a learner of the same order is sufficient. Nevertheless, we will now propose a new approach based on a First Order Logics learner.

Usually, the input space $\mathbf{X}$ to a RBFN, is not immediately available from the data, but is obtained by applying a transformation $T(D)$, said *feature construction*, on a data space D. When a RBFN is learned using one of the methods mentioned in the previous section, the choice of a proper transformation $T(D)$ is made a priori either on the basis of a domain expert's knowledge, or using feature selection techniques. Nevertheless, this choice should take into account the way the learning algorithm uses the features. Since this is not usually feasible, a $T(D)$ chosen a priori is not optimal, in general. An alternative approach (the compiled approach), which can be followed when the network construction is made using an algorithm for learning decision or regression trees, is to construct a redundant set of features $\mathbf{X_R}$ in order to be sure that the good ones are included. Then, the learning algorithm will select the ones it finds more suitable and so $\mathbf{X}$ will implicitly be chosen out of $\mathbf{X_R}$. Nevertheless, this approach becomes impractical for learning from large learning sets.

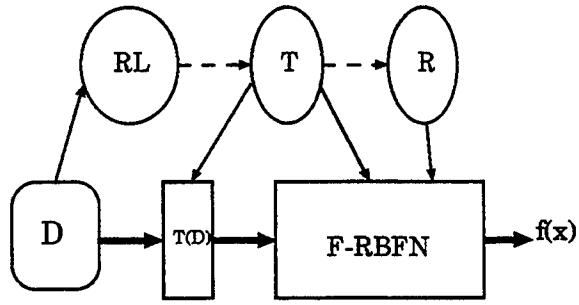


Figure 2: Sequence of steps for learning a F-RBFN. D = Learning set; RL = Rule Learning; T = Translation from rules to F-RBFN; R = Refinement by means of the Δ-Rule.

Let us consider a typical regression problem, frequently found in data mining, consisting in learning to predict a value $f(d_{t+\tau})$ associated to an event $d_{t+\tau}$ on the basis of the N events occurred in a time window going from event d_t back to d_{t-N+1}. Let, moreover, $\mathcal{F}$ be a set of functions assigned in order to construct the transformations $T(D)$ on the data space D. Every function $f_i(y_0, y_1, \ldots, y_k) \in \mathcal{F}$ can potentially define a new feature for each one of the N^k possible bindings between the arguments $y_0, y_1, \ldots, y_k$ and the N events in the temporal window. Then, the construction of all possible features, as required for an unrestricted compiled approach, would lead to an extremely complex learning set description (if $\mathcal{F}$ contains many operators and the temporal window ranges on many events).

The alternative approach, we propose here, is to introduce a mechanism that allows for the construction of features on demand of the learning algorithm. This can be realized by adopting a learning algorithm for learning relations extended with a mechanism similar to the one used in databases for supporting *computable constraints*. In particular, we will introduce a concept description language L in which predicate semantics is defined by means of a boolean function built on top of a feature constructor in the set $\mathcal{F}$. For instance, the predicate $\mathit{older}(x_0, x_1)$ stating that the event bound to x_0 occurred before the one bound to x_1, can be defined as:

$$\mathit{older}(x_0, x_1) :: \delta(x_0, x_1) > 0 \tag{6}$$

where $\delta(x_0, x_1) \in \mathcal{F}$ and computes the time interval between x_0 and x_1. When the learning algorithm uses predicate $older(x_0, x_1)$, its truth will be evaluated on the items bound to its variables. Therefore, computable predicates do not change the basic learning strategy of the algorithm but only delay the construction of the corresponding feature until it is used.

A first benefit is that memory requirements for the learning set is strongly reduced. However, this is balanced by a potential increase in the computational complexity, because a feature needs to be constructed again every time the learning algorithm tries to use a predicate depending on it. Fortunately, this does not happen in general, because only a small subset of the possible bindings is explored by a learning algorithm so that the replicated feature constructions are widely compensated by the strong reduction of the instantiations actually explored. It is worth noting that computable predicates have already been used in ML-smart (Bergadano, Giordana, & Saitta 1988) and in Smart+ (Botta & Giordana 1993).

Then, a rule set, learned in this way, has to be transformed into a feature constructor set $T(D)$ and a F-RBFN, which will be refined by performing the error gradient descent (see Figure 2). The set $T(D)$ can be easely obtained by collecting all the instances of the functions defining computable predicates occurring in the rule set. However, this translation can only be done if, for each learned rule $\varphi \rightarrow w$, the variables occurring in φ are bound to events occurring in the same position in every temporal window where φ is verified. As it will be described in the next section this condition can be enforced by adopting a specific control strategy in the learning algorithm.

The Symbolic Learning Algorithm

The FLASH (Fuzzy Logic Approximation SHaper) system, used for learning the layout of a RBFN, is a direct descendent of Smart+ (Botta & Giordana 1993) from which it inherits the general architecture. Nevertheless, several fundamental novelties have been introduced in order to handle the specific task we are addressing. FLASH shares with Smart+, as well as other learning relations algorithms, a general-to-specific search strategy for inductive learning, but it has a richer set of specializing operators, and it uses search strategies specific for regression tasks. FLASH uses two main formalisms for representing its knowledge: a relational database is used to store the training examples and partial hypothesis extensions; a first order logic language is used to represent the background and the acquired knowledge. A training example is represented as a set of distinct elements, called *objects*, each described by a set of properties, called *basic attributes* (Botta, Giordana, & Saitta 1992). Each object is typed and can be described by its own set of basic attributes. The main advantage of this representation scheme is its flexibility: the set of basic attributes can be easily extended; as well, the number of component objects can change from instance to instance and it naturally reduces to attribute-valued representation when all instances are made of only one object. Furthermore, this representation scheme is suited to a more flexible definition of the concept description language. In a regression problem, this representation can be exploited in the following way: let us consider the function plotted in Figure 3. An instance presented to FLASH is generated by taking into account a window of N temporal events and by defining an object for each one of them, described by two basic attributes: the relative (to the window) position of the event and the value of the function for that event. The instance is then labeled by the value that the function takes on in the event we have to predict. FLASH concept description language L is a first order Horn clause language, in which a clause body is built up by a conjunction of predicates in a set P. For each predicate $p \in P$, a semantic function S_p, is defined to instruct the system how to evaluate the truth of p on the learning events, by correlating terms in the predicate p to objects and attribute values in the instances. In general, S_p evaluates the truth μ_p of a numerical expression f_p as in the following example:

$$S_p(x_1, \ldots, x_n) :: member(f_p(x_1, \ldots, x_n), I_p) \quad (7)$$

being I_p an interval (segment) defining the range on the feature f_p where the predicate p is considered true. The numerical expression f_p can be any C language expression combining attribute values, C library mathematical functions and user-defined functions. This intensional definition of semantic functions usually results in a more concise representation, leads to a computationally faster implementation and allows *predicate schema*, that depend on run-time numerical constants, to be defined. By instantiating these constants with values in a specified range, FLASH can select the best instantiations in the context of the current hypothesis. For example, the semantic function S_δ, for a predicate schema $\delta(x_1, x_2, k_1, k_2)$ can be defined as follows:

$$\begin{array}{c} S_\delta(x_1, x_2, k_1, k_2) :: member(f_\delta, I_\delta) \\ f_\delta :: |h(x_2) - h(x_1)| \quad I_\delta = (k_1 - k_2, k_1 + k_2) \\ k_1 = [0.0 \div 1.0, 0.05] \qquad k_2 = [0.1 \div 0.2, 0.05] \end{array} \quad (8)$$

where I_δ is a segment with center k_1, that can assume the values 0.0, 0.05, 0.1, ..., 1.0, and width k_2, that

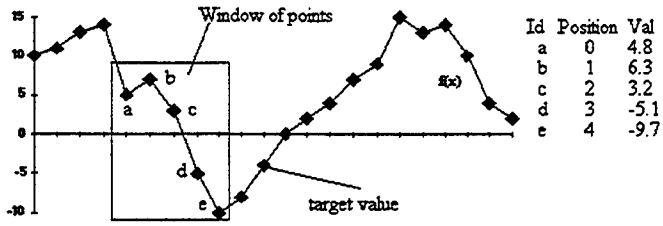


Figure 3: Example of function to be approximated.

can assume the values 0.1, 0.15, 0.2, and $h(x)$ is a function that returns the value of attribute h of objects x. Each numerical expression f_p can be thought of as a descriptive feature of the instances.

FLASH learning duty also comprises the selection of the suitable features for the task at hand, and, as a result, the learned knowledge will specify not only a set of implication rules, but also the set of features that are to be used to represent the instances. We will return to this point later on. FLASH can use background information in the form of necessity constraints on the applicability of predicates and on the acceptability of classification rules. The former are statements of the type:

$$p(x_1, x_2, \ldots, x_n) \; needs \; \varphi(y_1, y_2, \ldots, y_m) \qquad (9)$$

where $\varphi(y_1, y_2, \ldots, y_m)$ is a formula belonging to the concept description language. Expression (9) states that $p(x_1, x_2, \ldots, x_n)$ can be true only if $\varphi(y_1, y_2, \ldots, y_m)$ has been proven true. The induction space can be arbitrarily constrained by properly defining a set of necessity constraints. For instance, the following constraint states that predicate $\delta(x_1, x_2)$ can be added to a formula φ only if predicate $height(x_1)$ has already been applied (is part of the formula):

$$CR1(\varphi) :: \delta(x_1, x_2) \; needs \; height(x_1) \qquad (10)$$

These necessity constraints are used by FLASH to collect applicable predicates at each step of the search process. The second type of necessity constraints affects the structure of the classification rules the system should generate:

$$\psi(x_1, x_2, \ldots, x_n) \; needs \; \varphi(y_1, y_2, \ldots, y_m) \qquad (11)$$

where ψ and φ are formulas of the concept description language L. Expression (11) means that formula ψ can be accepted as a classification rule only if it contains φ. For instance, according to control rule CR2, a formula ψ can be accepted as a classification rule only if it contains predicates $height(x_1)$ and $\delta(x_1, x_2)$:

$$CR2(\psi) :: \psi \; needs \; height(x_1) \wedge \delta(x_1, x_2) \qquad (12)$$

By specifying which predicates must be present in a formula, it is possible to achieve a complete characterization of the input region associated to the value represented by the target class. Control rules of this type are applied at each step of the learning process to check if a formula needs further specialization (independent of its classification accuracy) or can be considered as a classification rule (evaluated for classification accuracy). One specific necessary constraint has been defined for the approximation task: as FLASH learns first order rules that are to be translated into a propositional network, classification rules must have a unique model on the data, in such a way that a direct mapping is feasible. This is assured by the following constraint:

$$CR(\psi) :: \psi(x_1, \ldots, x_n) \; needs \; isobj(x_1, k_1) \wedge \ldots \wedge isobj(x_n, k_n) \qquad (13)$$

where $isobj(x, k)$ is a predicate scheme which binds a variable x to a specific object in a sequence:

$$S_{isobj} :: position(x) = k \qquad (14)$$

FLASH is configured to learn a concept at a time by using a best first search coupled with a covering strategy: at each step, the most promising formula is selected for specialization and applicable predicate schema are collected according to the necessary constraints; then, for every predicate schema, numerical parameters are instantiated, the predicate is tested on the data and scored according to an information gain function. The best $n \geq 1$ predicate schema so obtained are used to expand a search tree. The found formulas are tested for classification accuracy and against necessary constraints; those that are accepted as classification rules are saved and instances are declared covered, so that FLASH can focus its attention on the remaining training examples.

1. Start with an "or" tree T containing the formula "True", or previously initialized with a set of formulas predicted by a theory (or by an expert).
2. Insert in the list $OPEN$ all the leaves currently existing in T.

3. Select from $OPEN$ the formula ϕ having the best ranking $\nu(\phi)$.

4. Determine the set P_A of applicable predicates to ϕ according to the predicate constraints.

5. For each predicate $p(\vec{x}, \vec{K}) \in P_A$ find the assignment $\vec{k}$ for the parameters $\vec{K}$ in p, which maximizes $\nu(\phi \wedge p(\vec{x}, \vec{k}))$, and if $\nu > 0$ put $\phi \wedge p(\vec{x}, \vec{k})$ in the list $TODO$ sorted according to decreasing values of ν.

6. Then, expand T with the $n \geq 1$ first formulas in $TODO$ and put them in $OPEN$.

7. If now $OPEN$ contains some classification rule ψ (according to an assigned criterion), go to step 1; otherwise go to step 3.

This cycle is repeated until all instances have been covered, or there is no possibility to find other classification rules (e.g., because allocated computational resources have been exhausted). The function ν is a measure of the quality of an inductive hypothesis ϕ, and has been chosen to be the function

$$if(v \geq v_0)\ then\ \nu = v \cdot u\ else\ \nu = 0 \qquad (15)$$

being $0 \leq v \leq 1$ and $0 \leq u \leq 1$ the proportion of positive examples covered by ϕ and the correctness of ϕ, respectively. The condition $v \leq v_0$ prevents the generation of too specific classification rules, in order to control the overfitting.

Inductive hypotheses are accepted as classification rules also if they are slightly inconsistent. In particular, a classification rule ϕ is allowed to cover some examples of the two classes corresponding to the two values in W adjacents to the target class, provided that its correctness doesn't drop below an assigned threshold u_0. The tuning of u_0 and v_0 is left to the user.

FLASH learns rules of the following type:

$$p_1(\vec{x}, k_{11}, k_{12}) \wedge \ldots \wedge p_n(\vec{x}, k_{n1}, k_{n2}) \rightarrow w \qquad (16)$$

in which all parameters k_{ij}s have been instantiated, there are as many $isobj$ predicates as variables in the rule, and w is one of the values of the function to be approximated belonging to the set W. Each rule is translated into a second hidden layer neuron connected to the output neuron with a link having weight equal to w. A predicate $p_i(x, k_{i1}, k_{i2})$ in the antecedent of a rule can be translated into a first hidden layer neuron with gaussian activation function, derived by the segment defined by k_{i1} and k_{i2} according to the transformation described in the previous sections, and linked to the corresponding second hidden layer neuron and to an input neuron.

Evaluation of the Method

The described methodology has been tested on two approximation problems, bearing quite different characteristics: the first testbed is a classical approximation problem of a continuous function generated by a well known chaotic series (Mackey-Glass), which has been addressed using other techniques and allows comparisons to be made, whereas the second is concerned with the approximation of a control function in a robotics environment.

The Makey-Glass chaotic series is generated by the following differential equation:

$$\dot{x} = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \qquad (17)$$

with $x(t < 0) = 0$, $x(0) = 1.2$ and $\tau = 17$. The classical learning problem is defined as follows: given the series values at instants t, t-6, t-12, t-18, forecast the series value at t+84. Table 1 shows some of the best results presented in the literature, according to the NDEI (Non-Dimensional Error Index) error index, that is the ratio between the root average square error and the standard deviation. MLP refers to the result obtained by a multi-layer perceptron; ANFIS (Jang 1993) is a fuzzy inference system implemented as an adaptive neural network, whose structure has been manually defined. FC stands for a fuzzy controller built from symbolic knowledge learned by CART (Breiman *et al.* 1984) and Smart+ (on a propositional definition of the problem) and refined for 5000 epochs. $RBFN_{Smart+}$ is the result obtained by using a RBFN on the same symbolic knowledge used by FC, refined for 5000 epochs.

Table 1: Comparison between the best results on the Mackey-Glass learning problem.

Method	Learning Instances	NDEI
MLP	500	0.05
$ANFIS$	500	0.036
$FC_{CART} + R$	1000	0.037
$FC_{Smart+} + R$	1000	0.032
$RBFN_{Smart+} + R$	1000	0.032

As it can be noted, the Radial Basis Function Networks obtained a result that compares favorably to the others, being only better at the 6th decimal digit than the one obtained with FC. In order to test FLASH ability to automatically select which input features are relevant to the problem, a different formulation of the above problem has been used in two new experiments: instead of using only the four above mentioned values as input features, all the values of the series corresponding to instants from t-18 to t have been considered. In particular, in the first experiment, we used a

propositional definition of the problem, by defining the learning problem in terms of 37 numerical attributes, 19 of which refers to the values at instants t-18, ...t, and the remaining 18 refers to the differences between the values of two consecutive instants. The concept description language consists of 37 predicate schemas, whose semantics functions compute the membership of an attribute value in an interval. In the second experiment, we used a first order representation of the instances: each instance consists of 19 objects described by two attributes, the position inside the window and the corresponding series value at that point. In this second case, we defined only two predicate schemas, which refer to the series value and the difference between the values at instants t-x, t-y, $x \neq y \in [0,18]$, respectively. In all experiments, we used the same learning set, consisting of 1000 instances, and test set, of 500 instances, as the ones used in the reported experiments, labeled according to 12 classes of values. For the sake of comparison, CART has been run on the propositional definition of the problem and the learned knowledge translated in RBFN and refined for 5000 epochs. The results on the test set are reported in Table 2. As can be noted, the approximator constructed with the described methodology obtained better performances than any other method analyzed. What is significant in this experiment is not only the smallest error index obtained, but the fact that the first order representation language used allowed FLASH to analyze a large number of possible features (190) and to select the most relevant ones (46), which resulted in a very good network layout.

Table 2: Results obtained on the new problem definition.

Method	Learning Instances	NDEI
$RBFN_{CART} + R$	1000	0.049
$RBFN_{FLASH_{prop}} + R$	1000	0.046
$RBFN_{FLASH} + R$	1000	0.024

In the second testbed, we considered an approximation problem in a robotics environment: the goal was to learn a control function of a robot arm performing the peg-into-hole task, in such a way that it can be reproduced at a higher speed. Both the peg and the hole are of circular shape. Input values are forces and torques along the three directions, axial, radial and vertical, measured by sensors placed on the robot arm (Figure 6 reports plots of these values), whereas output values are the corresponding velocity in the three directions the robot arm should move at (Figure 7 reports a plot of the output values).

For every output velocity, a learning problem has been defined according to the methodology previously described: in particular, each instance presented to FLASH consists of 8 objects, each described by 6 attributes (three force values and three torque values) and corresponding to sensory readings at successive time points t-7, t-6, ..., t, and it is labeled according to the velocity value at time point t+1 (14 classes of values). Furthermore, we defined three sets of predicate schemas (each consisting of 6 predicate schemas) in order to account for the fact that controllers used in this task are usually Integral-Differential controllers: one predicate schema refers to the values of an attribute at a given time point; a second predicate schema refers to the difference between attribute values at time points t and t-x, $\forall x \in [1,7]$, that should capture properties of the derivative of the signal; a third one refers to the sum of attribute values between any time point t-x, $x \in [1,7]$, and t, that should capture properties of the integral of the signals.

Table 3: Statistical comparative results for V_z, the most critical output control signal for the peg-into-hole task; all numbers are percentages (%).

Method	Epochs	V_z $Err_{avg} \pm Std.dev.$
CART (alone)	0	1.38 ± 5.01
FC_{CART}	0	11.84 ± 20.92
$FC_{CART} + R$	5000	0.45 ± 2.75
FC_{SMART+}	0	2.12 ± 4.94
$FC_{SMART+} + R$	5000	1.04 ± 4.18
$RBFN$	10000	3.3 ± 3.83
$TDNN$	10000	2.7 ± 2.30
MLP	10000	4.5 ± 3.84
$FLASH$ (alone)	0	3.35 ± 5.26
$RBFN_{FLASH}$	0	10.21 ± 22.15
$RBFN_{FLASH} + R$	3000	0.43 ± 2.74

As shown in Table 3, several learning algorithm have been compared, by using the same learning (four insertion sequences) and test sets (three insertion sequences). In particular, we reported results obtained by using CART (Breiman *et al.* 1984) and Smart+ (Botta & Giordana 1993) to initialize a fuzzy controller (FC), implemented according to the schema reported in Figure 1, on a propositional problem definition: each instance is described by 24 attributes, corresponding to force and torque values measured in the last four time points. The number of neurons used in the fuzzy controller varies from 84 to 168 in the first hidden layer with unidimensional gaussian activation function, and from 46 to 85 in the second hidden layer. RBFN has been initialized by using a clustering algo-

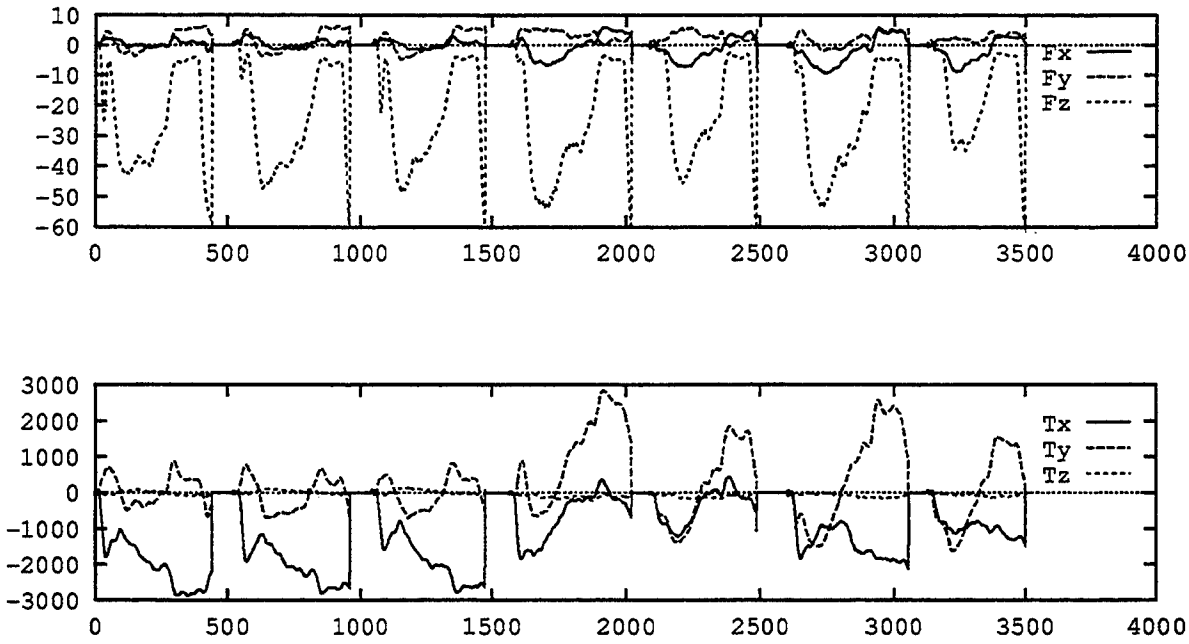


Figure 4: Input signals for seven insertion sequences: Fx, Fy e Fz correspond to the force sensors along the three coordinates while Tx, Ty e Tz correspond to the torque sensors. The values have been sampled at 10ms.

rithm (Musavi *et al.* 1992) that built a network with a number of neurons in the hidden layer that varies from 30 to 60, but with a 6-dimensional gaussian activation function. Moreover, we reported results obtained by other researchers (Kaiser 1994) with two other kind of neural networks, TDNN (time-delay neural network) and MLP (multi-layer perceptron). The layout of these networks has been manually chosen by performing several experiments with different configuration, by varying the number of neurons in the hidden layer and the number of delay units (for TDNN). The reported values correspond to the configuration that obtained the best results. Finally, the last three rows contain results obtained by the described methodology at each step: by using FLASH classifier, by using RBFN without refinement and after refinement. The number of neurons in the first hidden layer varies from 31 to 88 whereas, in the second hidden layer, it varies from 23 to 36. The first point to deserve attention is that the presented methodology builds up networks that need less training to reach equivalent or even better performances. This is mainly due to the compactness of the network produced, that, if compared to the fuzzy controller case, needs about 50% neurons, so having much less parameters to be tuned.

Figure 6 compares the obtained approximation with the original curve of velocity Vz (the most critical for a good insertion operation) in one of the experiments performed. After refinement (Figure 6c), an almost complete identity in the resulting behavior can be observed.

Conclusion

A new method based on an algorithm for learning relations in First Order Logics (FOL) has been presented in order to learn RBFN layouts from examples. The experimental results presented above show that the method can construct very accurate approximators in regression problems.

However, beyond the specific algorithms, the novelty presented in this paper is in the way FOL has been used: not as a target, but as an intermediate step in the construction of a knowledge base which still belongs to the propositional logics. In particular, the ability at exploring many alternative bindings offered by a FOL learner has been exploited in order to search the feature space usable for constructing a RBFN.

In our opinion, this is a new way of looking at FOL which up to now has not been considered and that deserves attention.

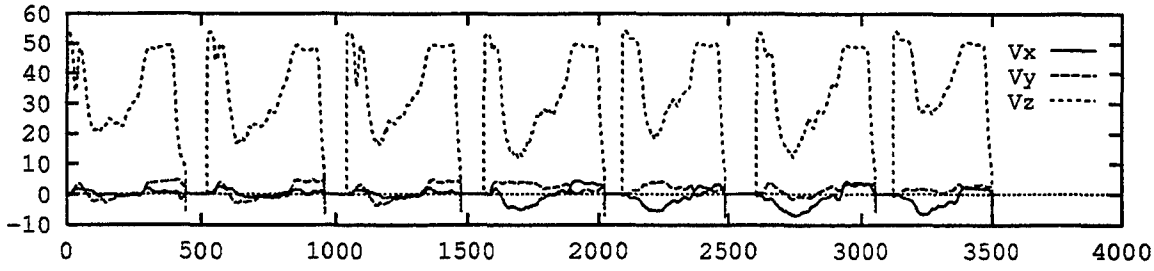


Figure 5: Velocity signals computed by PID-controller from the signals reported in the previous pictures.

References

Baroglio, C.; Giordana, A.; Kaiser, M.; Nuttin, M.; and Piola, R. 1996. Learning controllers for industrial robots. *Machine Learning.*

Baroglio, C.; Giordana, A.; and Piola, R. 1994. Learning control functions for industrial robots. In *Proceedings of the "Workshop on Robotics", Machine Learning Conference.*

Bergadano, F.; Giordana, A.; and Saitta, L. 1988. Learning concepts in noisy environment. *IEEE Transaction on Pattern Analysis ans Machine Intelligence* 555–578.

Botta, M., and Giordana, A. 1993. SMART+: A multi-strategy learning tool. In *IJCAI-93, Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, volume 2.

Botta, M.; Giordana, A.; and Saitta, L. 1992. Learning in uncertain environments. In Yager, R., and Zadeh, L., eds., *An Introduction to Fuzzy Logic Applications in Intelligent Systems.* Kluver Academic Publishers.

Breiman, L.; Friedman, J.; Ohlsen, R.; and Stone, C. 1984. *Classification And Regression Trees.* Pacific Grove, CA: Wadsworth & Brooks.

Craven, M., and Shavlik, J. 1994. Using sampling and queries to extract rules from trained neural networks. In *Proceedings of the Eleventh International Conference on Machine Learning*, 37–45.

Jang, J. 1993. ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Transactions on Systems, Men and Cybernetics* SMC-23(3):665–687.

Kaiser, M. 1994. Time-delay neural networks for control. In *Proceedings of the Symposium on Robot Control '94 (SYROCO '94).*

Mamdani, E., and Assilian, S. 1975. An experiment in linguistic synthesis with a fuzzy logic controller. *International Journal of Man-Machine Studies* 7(1):1–13.

Michalski, R. 1983. A theory and methodology of inductive learning. In Michalski, R.; Carbonell, J.; and Mitchell, T., eds., *Machine Learning: An Artificial Intelligence Approach*, 83–134. Los Altos, CA: Morgan Kaufmann.

Moody, J., and Darken, C. 1988. Learning with localized receptive fields. In Sejnowski, T.; Touretzky, D.; and Hinton, G., eds., *Connectionist Models Summer School.*

Musavi, M.; Ahmed, W.; Chan, K.; Faris, K.; and Hummels, D. 1992. On the training of radial basis function classifiers. *Neural Networks* 5:595–603.

Pazzani, M., and Kibler, D. 1992. The utility of knowledge in inductive learning. *Machine Learning* 9:57–94.

Poggio, T., and Girosi, F. 1990. Networks for approximation and learning. *Proceedings of the IEEE* 78(9):1481–1497.

Quinlan, J. 1979. Induction over large data bases. Technical Report HPP-79-14, Heuristic Programming Project, Stanford University.

Rumelhart, D. E., and McClelland, J. L. 1986. *Parallel Distributed Processing : Explorations in the Microstructure of Cognition, Parts I & II.* Cambridge, Massachusetts: MIT Press.

Sammut, C.; Hurst, S.; Kedzier, D.; and Michie, D. 1992. Learning to fly. In Sleeman, D., and Edwards, P., eds., *Machine Learning - Proceedings of the Ninth International Workshop (ML92)*, 385–393. Morgan Kaufmann.

Specht, D. 1988. Probabilistic neural networks for classification mapping, or associative memory. In

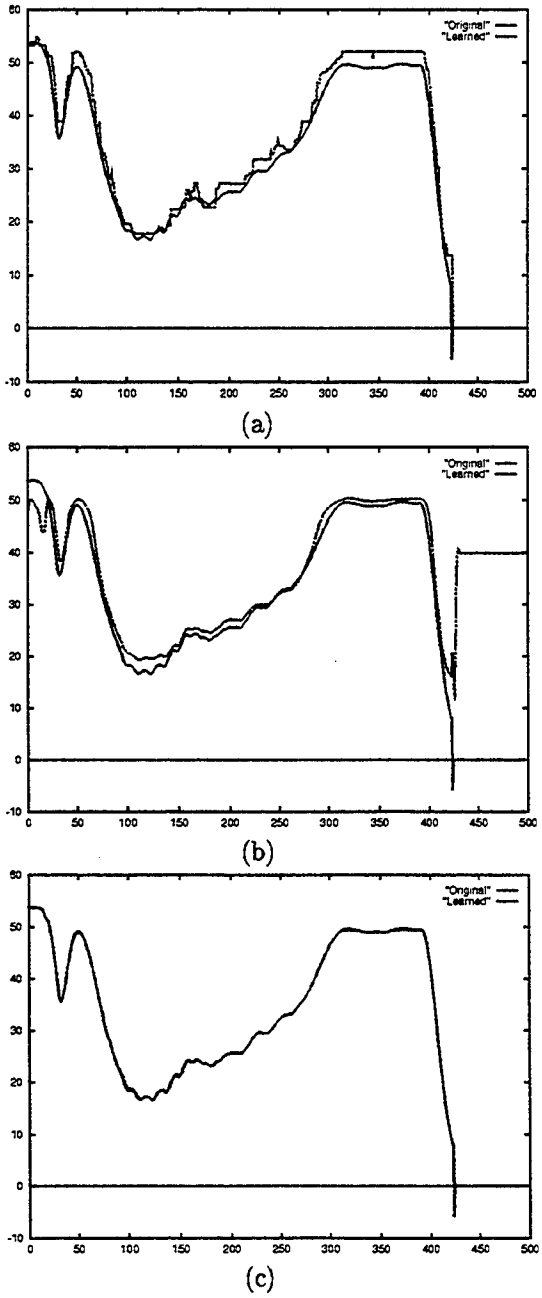


Figure 6: Approximation of the control function V_z obtained using: (a) FLASH only; (b) RBFN before refinement; (c) RBFN after refinement.

IEEE International Conference on Neural Networks, volume 1, 525–532.

Towell, G., and Shavlik, J. 1993. Extracting refined rules from knowledge-based neural networks. *Machine Learning* 13(1):71–101.

Towell, G., and Shavlik, J. 1994. Knowledge based artificial neural networks. *Artficial Intelligence* 70(4):119–166.

Towell, G.; Shavlik, J.; and Noordwier, M. 1990. Refinement of approximate domain theories by knowledge-based neural networks. In *Proceedings of the 8th National Conference on Artificial Intelligence AAAI'90*, 861–866.

Tresp, V.; Hollatz, J.; and Ahmad, S. 1993. Network structuring and training using rule-based knowledge. In *Advances in Neural Information Processing Systems 5 (NIPS-5)*.

An Empirical Study of Computational Introspection: Evaluating Introspective Multistrategy Learning in the Meta-AQUA System

Michael T. Cox
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213-3891
mcox+@cs.cmu.edu

Abstract

The theory of introspective multistrategy learning proposes that three transformations must occur to learn effectively from a performance failure in an intelligent system: Blame assignment, deciding what to learn, and learning-strategy construction. The Meta-AQUA system is a multistrategy learner that operates in the domain of story-understanding failures and is designed to evaluate this learning approach. We discuss experimental results supporting the hypothesis that introspection facilitates learning in a multistrategy environment. In an empirical study, Meta-AQUA performed significantly better with a fully introspective mode than with a reflexive mode in which learning goals were ablated. In particular, the results lead to the conclusion that the process which posts learning goals (deciding what to learn) is a necessary transformation if negative interactions between learning methods are to be avoided and if learning is to remain effective. Moreover, we show that because learning algorithms can negatively interact, the arbitrary ordering of learning methods can actually lead to worse system performance than no learning at all. The goals of this research are to better understand the interactions between learning algorithms, to determine the role of introspective mechanisms when integrating them, and to more firmly establish the conditions under which such an approach is warranted (and those under which it is not).

Introduction

From the very early days of AI, researchers have been concerned with the issues of machine self-knowledge and introspective capabilities (e.g., McCarthy 1959; Minsky 1954/1965), yet few have quantitatively evaluated the trade-offs involved with much care or investigated the nature of the role introspection assumes when learning. The research presented here uses computational introspection to assist in the choice and sequencing of learning algorithms within a multistrategy framework. Yet open questions exist as to whether introspection is worth the computational overhead and in exactly what ways it facilitates the learning process. This paper begins to investigate these research questions empirically.

The theory of *introspective multistrategy learning* (IML) proposes that three transformations must occur to learn effectively from performance failures. First, given a trace of the performance failure, a learner must perform *blame assignment* by mapping the symptoms of the failure to a causal explanation of the failure. Secondly, the learner must use this explanation to *decide what to learn* by posting explicit learning goals to achieve desired changes in its background knowledge. Thirdly, the learner can use these goals for *learning-strategy construction* by treating the learning task as a nonlinear planning problem. That is, the learner constructs a partially-ordered plan to repair the background knowledge by sequencing calls to standard learning algorithms. The Meta-AQUA system (Cox 1996; Ram & Cox 1994) is a multistrategy learner that operates in the domain of story understanding failures and is designed to evaluate this learning approach.

Section 2 briefly presents the Meta-AQUA system by describing the story generation module with which experimental trials are generated and by providing a brief explanation of the performance and learning tasks. Section 3 provides a computational evaluation of the hypothesis that introspection facilitates learning using data obtained from the Meta-AQUA system. Section 4 summarizes the results and concludes with a short discussion.

Meta-AQUA

Meta-AQUA is a learning system that chooses and combines multiple learning methods from a toolbox of algorithms in order to repair faulty components responsible for failures encountered during the system's performance task. The system architecture and flow of information within Meta-AQUA is shown in Figure 1. The problem generation module outputs a story to the story-understanding performance system with the initial goal to understand the input. The performance module uses schemas from its background knowledge (BK) to explain the story and to build a representation for it in its foreground knowledge (FK). If this task fails, then a trace of the reasoning that preceded the failure is passed to the learning subsystem. A case-based reasoning (CBR) (Kolodner, 1993) subsystem within the learner uses past cases of introspective reasoning from the BK to explain the comprehension failure and to generate a set of learning goals. These goals, along with the trace, are then passed to a nonlinear planner. The planner subsequently builds a learning strategy from its toolbox of learning methods. The learning plan is passed to an execution system that examines and changes items in the BK. These changes enable improved future performance.

The above conceptualization of learning is consistent with both Michalski's (1994) Inferential Learning Theory that decomposes a learning task into an input, the BK, and a learning goal and Carbonell (1986) and Veloso's (1992) emphasis on reasoning from a trace of the derivation of a solution rather than from solutions themselves. Although the algorithms and knowledge structures used by Meta-AQUA have been reported elsewhere (e.g., Cox 1994, 1996; Cox & Ram 1995; Ram & Cox 1994; Ram, Cox, & Narayanan 1995), this section outlines the system in order to provide a context for understanding the evaluation.

The Input: Elvis World and Tale-Spin

To support large data collection, the Tale-Spin story generation program[1] provides a potentially infinite number of input variations that test Meta-AQUA's ability to learn from explanation failure. Given a main character and a problem, Tale-Spin simulates the actions that would be necessary for the character to achieve goals stemming from the problem. For example if a character is bored, Tale-Spin assigns the character an initial goal to remove the state of boredom. The character can achieve the goal by convincing a friend to play, finding a ball, going outside, and then batting the ball back and forth (see Figure 2). For each event in the story, the generator adds any associated causal results. These results change the world and enable further actions by characters in

[1]Tale-Spin (Meehan 1981) was obtained from the UC Irvine repository. Pazzani (1994) used it to evaluate the OCCAM multistrategy learning system.

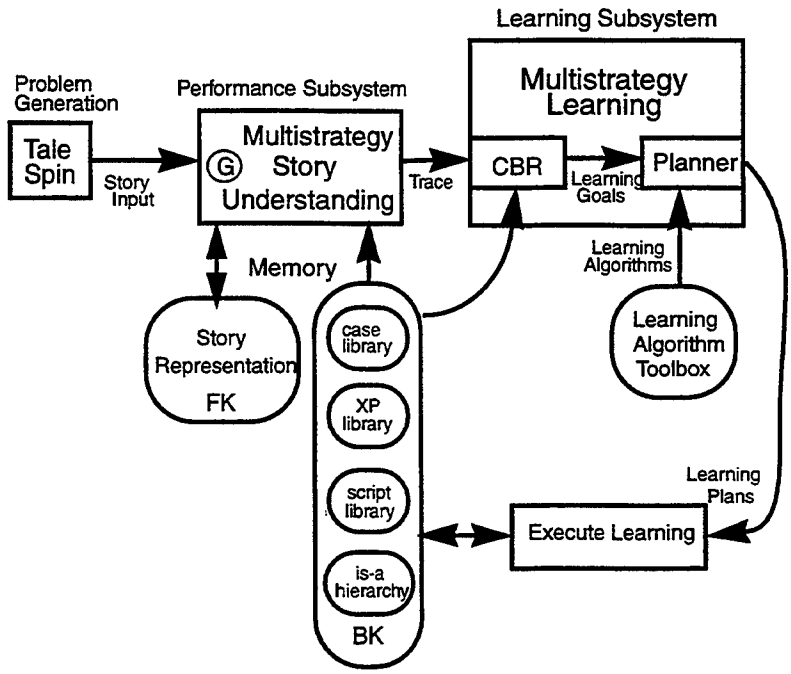


Figure 1. Detailed Meta-AQUA system architecture

the story. For example, the act of getting the ball and going outside enables the hitting of the ball which results in the ball's movement between the characters. In turn, these actions remove the boredom. The story terminates when the goals and subgoals of the main character have been achieved or when all possible plans to achieve them have been exhausted.

```
Elvis asked Lynn, "Would you push the
ball2 to me away from you?" Lynn went to
the garage. She picked up the ball2. She
had the ball2. She went to outside. He
went to outside. He played with the
ball2. She hit the ball2. She hit the
ball2 because she wanted to move the
ball2 to him. He hit the ball2. He hit the
ball2 because he wanted to move the ball2
to her. He played with the ball2 because
he didn't want to be bored.

--- The End ---
```

Figure 2. Sample Elvis World story

Among the changes to Tale-Spin, we added a musician named Elvis and a police officer to the cast of characters. Elvis is temporarily boarding with Mom, Dad and their daughter Lynn, whereas the officer occasionally visits the house, presumably because of neighborhood complaints of loud music and raucous behavior. Furthermore, the police officer often (but not always) brings a drug-detection dog along with him. We also removed Karen from the cast of main characters available as a protagonist and the state of hunger from the possible initial problem states. Thus, Tale-Spin now generates a more uniform distribution of situations.

We also added two new problem types to the original problems of thirst and boredom. Characters may now be *jonesing*[2] for drugs. In Elvis' case, he sometimes smokes marijuana to relieve his jones, whereas Dad occasionally smokes a pipe with tobacco. Lynn has also been given a tobacco habit. The police officer has the problem of being *concerned* about the law. The police officer's state of being concerned is relieved if he can either locate contraband or arrest criminals.[3] We also reprogrammed Tale-Spin to hide the marijuana during story initialization in different locations (e.g, in the cupboard, refrigerator, and under the carpet), so the officer's task varies depending on entry conditions (i.e., at what point in the story the officer arrives on the scene and whether the dog accompanies him), the initial location of the pot, and the actions of the characters in the story.

Finally, to facilitate the performance task, we modified the Tale-Spin program so it generates explanations of key events in the stories. The resolution of all anomalies are thereby incorporated within each story. For example, Tale-Spin includes a reason why Lynn strikes the ball in the story above because it knows that Meta-AQUA will find the action anomalous and thus try to explain it. Although in an ideal implementation, the understanding process should be able to make independent inferences that confirm explanations of the input, Meta-AQUA depends on the story to provide explanations for this confirmation. The implementation concentrates on the learning task rather than the understanding task.

The Performance and Learning Tasks: Story Understanding, Explanation, and Repair

The Meta-AQUA system learns about drug-smuggling and sports activities, given its prior experience with stories about terrorists and its general knowledge of physical causality. The systems' performance task is to "understand" stories by building causal explanations that link the individual events into a coherent whole. The performance sub-system uses a multistrategy approach to understanding. Thus, the top-level goal is to choose a comprehension method (e.g., script processing, case-based reasoning, or explanation pattern application) by which it can understand the input. When an anomalous or otherwise interesting input is detected, the system builds an explanation of the event, incorporating it into the preexisting model of the story in the FK.

[2]In the vernacular, a "jones" is a drug habit accompanied by withdrawal symptoms. The verb "to jones" is to be going through a state of withdrawal.

[3]Unlike the UC Irvine version of Tale-Spin in which characters and their goals did not interact, we modified the program so that the police officer is a competing character with his own problem and goal. Because the officer confiscates the marijuana when found and arrests Elvis, such events may preempt the enabling conditions of actions Elvis had planned to perform. For instance, if Elvis is thirsty but the officer arrests him, this condition restricts his freedom of movement so that he cannot go to the faucet for water. Therefore, the story can end with Elvis still having the problem with which he began (i.e, thirst).

In the story from Figure 2 for example, Meta-AQUA finds it unusual for a person to strike a ball because its conceptual definition of the "hit" predicate constrains the object attribute to animate objects. It tries to explain the action by hypothesizing that Lynn tried to hurt the ball (an abstract explanation pattern, or XP, retrieved from the BK instantiates this explanation). In the following, sentence, however, the story specifies an alternate explanation (i.e., the hit action is intended to move the ball to the opposing person). This input causes an expectation failure because the system had expected one explanation to be true, but another proved true instead.

When the Meta-AQUA system detects an explanation failure, the performance module passes a trace of the reasoning to the learning subsystem. At this time, the learner needs to explain why the failure occurred (assign blame) by applying an introspective explanation to the trace. A meta-explanation (Meta-XP) is retrieved using the failure symptom as a probe into memory. Meta-AQUA instantiates the retrieved meta-explanation and binds it to the trace of reasoning that preceded the failure. The resulting structure is then checked for applicability. If the Meta-XP does not apply correctly, then another probe is attempted. An accepted Meta-XP either provides a set of learning goals (determines what to learn) that are designed to modify the system's BK or generates additional questions to be posed about the failure. Once a set of learning goals are posted, they are passed to the nonlinear planner for building a learning plan (strategy construction).

Figure 3 lists the major state transitions that the three learning processes produce. The learning plan is fully ordered to avoid interactions. For example, the abstraction step must precede the other steps. A knowledge dependency exists between the changes on the hit concept as a result of the abstraction and the use of this concept by both generalization and the indexing.[4] After the learning is executed and control returns to sentence processing, subsequent sentences concerning the hit predicate causes no anomaly. Instead, Meta-AQUA predicts the proper explanation.[5]

Symptoms:
- Contradiction between input and background knowledge
- Contradiction between expected explanation and actual explanation

Faults:
- Incorrect domain knowledge
- Novel situation
- Erroneous association

Learning Goals:
- Reconcile input with conceptual definition
- Differentiate two explanations

Learning Plan:
- Abstraction on concept of hit
- Generalization on hit explanation
- Index new explanation
- Mutually re-index two explanations

Figure 3. Learning from Explanation Failure

Computational Evaluation

This section presents the results of computational studies performed with Meta-AQUA to test the hypothesis that introspection facilitates learning. The methodology below not only tests our hypothesis, but also it more directly supports the position that a loose coupling of blame-assignment and repair (via learning goals) is preferred to a tight coupling approach. But perhaps more importantly, this methodology also scrutinizes the claim that the second phase of learning, deciding what to learn, is *necessary* for effective learning. IML theory is the only learning theory that makes such a strong claim. Few computational systems other than Meta-AQUA include an explicit calculation of a goal to learn and then use that goal to influence learning. Converging with the arguments and hand-coded examples from previous research that favor this position (e.g., Cox 1994; Cox & Ram 1995),

[4]During mutual re-indexing, the explanations are differentiated based on the object attribute-value of the hit. However, the abstraction transmutation changes this attribute. The generalization method applied to the new explanation also uses this attribute. See Cox & Ram (1995) for a more complete analysis.

[5]As pointed out by an anonymous reviewer, it would be nice for the system to use ontological knowledge to infer that the inanimate objects cannot feel pain. At the current time, however, the system possess neither the bias to make a proper inductive leap during learning nor the prerequisite knowledge to make the inference. Indeed, the system has but a primitive causal understanding of the mechanics of pain.

this paper presents quantitative evidence that supports the utility of this stage.

The Hypothesis

Generally, we claim that introspection facilitates learning. More specifically, we assert that the rate of improvement in story understanding with learning goals exceeds that of story understanding without learning goals holding all other factors constant. Our approach is to perform a kind of ablation study. Surgically removing the learning goals eliminates part of the system's mechanism responsible for introspection. The intention of this manipulation is to show different empirical learning curves with and without introspection as a function of the number of inputs.

Introspective learning is a computational process with the decomposition as shown in the upper portion of Figure 4. *Fully introspective multistrategy learning* consists of examining one's own reasoning to explain where the reasoning fails. It consists further of knowing enough about the self and one's own knowledge that the reasoner can explicitly decide what needs to be learned. Introspection amounts to performing blame assignment and subsequently posting explicit goals to learn. Learning amounts to the construction of a learning plan designed to change the reasoner's knowledge and thereby to achieve the learning goals.

Removing the goals from the introspective process above, leaves a more reflexive activity we call *semi-introspective multistrategy learning*[6] (see the lower portion of Figure 4). Instead of using the explanation of failure created during the blame-assignment phase to post a set of learning goals that then direct the construction of a learning plan, the explanation can directly determine the choice of repair methods. System performance under both conditions can then be compared with Meta-AQUA under a no learning situation.

Independent and Dependent Variables

Learning rates relative to a baseline no-learning condition are compared between the fully introspective and a semi-introspective version of Meta-AQUA. The independent variable that effects this change is the presence and influence of learning goals. The first experimental condition is referred to as the learning goal (LG) condition, and represents Meta-AQUA as described in Ram & Cox (1994). Under the LG condition, the system builds a learning strategy. This construction is guided by the learning goals spawned by the Meta-XPs that explain the failure. Hence, this condition represents a loose coupling approach between fault (failure cause) and repair (learning).

The second condition is called the random learning (RL) condition. Given the explanation of the causes of failure the system can directly assign calls to particular learning algorithms for each fault. The construction of the learning plan is then performed by a random ordering of these function calls, rather than by non-linear planning to achieve the learning goals. The RL condition represents a tight coupling

[6]It is semi-introspective because, although part of the introspective process has been removed, the introspective mechanics of blame-assignment remain. Future research remains to test the performance with blame assignment removed and learning goals present.

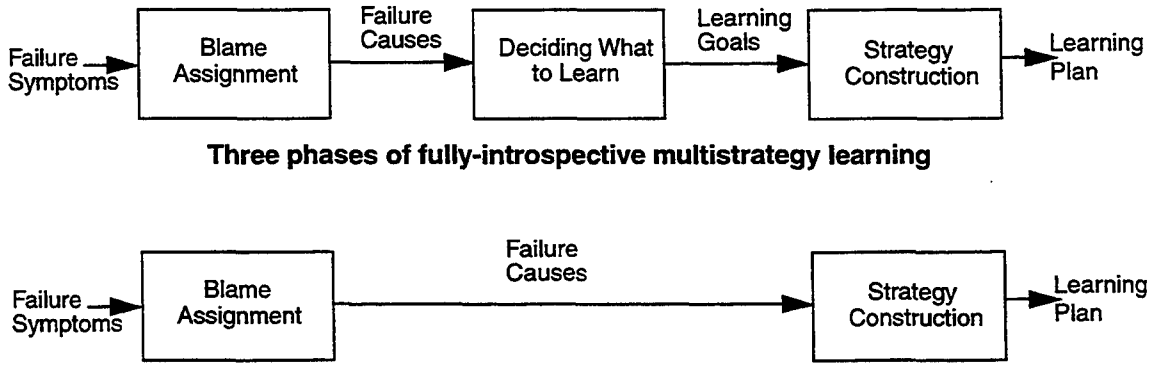


Figure 4. Learning goal ablation

approach (i.e., direct mapping from fault, or failure cause, to repair).

The final condition is called the no learning (NL) condition in which Meta-AQUA performs story understanding, but if a failure exists, the system constructs no learning strategy. This condition represents the baseline performance from which both the LG and RL conditions can be compared. Holding all parameters constant except the independent variables, Meta-AQUA is given input from the Tale-Spin problem generator and the dependent variable is measured.

The dependent variable must measure system performance (i.e., story understanding and explanation). In previous research, paraphrase and question answering tasks have been used as this measure (e.g., Lehnert, Dyer, Johnson, Yang, & Harley 1983; Schank & Riesbeck 1981; Wilensky 1978). If a reader sufficiently understands a body of text, the reader should be able to summarize the central points of the story and list the major events within it. If the story is well understood, then the reader can answer questions concerning the events and relationships within the story.

With story understanding programs such as BORIS (Lehnert et al. 1983), the researchers pose questions to the system and subjectively evaluate the answers to determine text comprehension effectiveness. One can count the number of questions answered correctly to ascertain an "absolute" measure of performance, but this is misleading. In contrast to externally posed questions, Chi (1995, Chi et al. 1989) reports that improved learning is correlated with human subjects who generate their own questions and explain the answers themselves. Being able to recognize that a gap exists in one's own knowledge, and thus to ask the question "Why don't I understand this?" (Ram 1991), is the first step to improving understanding. To pose self-generated questions thus indexes understanding and simultaneously reduces the probability of asking only easy questions. So, to evaluate the ability of the Meta-AQUA system, credit is given for simply posing a question that deserves asking.

Moreover, humans who are asked questions on reading tests are sometimes given points for partial answers. Unlike questions that have provably correct answers, answers to explanatory questions are difficult to judge in an absolute sense. So to be more realistic, the evaluation criterion in Meta-AQUA assigns credit for providing any answer to a question. Therefore, the full evaluation metric is as follows. For each anomalous or interesting input in a story, a point is given for posing a question, an additional point is given for providing any answer whatsoever, and a third point is assigned for answering what the researcher judges correct. The sum represents the dependent variable.

The Empirical Data

To serve as experimental trials and to minimize order effects, Tale-Spin generated six random sequences of Elvis-World stories. On each of these runs, Meta-AQUA processes a sequence three times, once for each experimental manipulation. The system begins all runs with the same initial conditions. For a given experimental condition, it processes all of the stories in the sequence while maintaining the learned knowledge between stories. At the end of the sequence, the system resets the BK. The input size for a run varies in length, but averages 27.67 stories per run.[7] The corpus for the six runs includes 166 stories, comprising a total of 4,884 sentences. The stories vary in size depending on the actions of the story and Tale-Spin's randomness parameters (e.g., the probability that a character will stop throwing an object on the current toss), but average 29.42 sentences.

Run Number Four. Run number four is particularly interesting because the greatest number of learning interactions occur in this set. The input to run four consists of 24 stories as enumerated in Table 1. The stories contain a total of 715 sentences, and the average number of sentences per story is 29.8. Each numeric entry in Table 1 contains a triple of the form <LG, RL, NL>. For example, the sixth column represents the number of learning episodes for each trial and for each condition. Note that the third element of each triple in this column is zero since learning is disabled in the NL condition. The fifth column (Question Points) contains the values for the dependent variable. These values represent the sums of triples from the second, third and fourth columns (Posed Questions, Answered Questions and Correct Answers, respectively).

In this run, random drug busts occur 11 times (5 with the canine squad and 6 with a lone police officer). Also, Dad is the most common protagonist, while Elvis, Officer1, and Lynn are tied for the least common. Furthermore, boredom is the major problem encountered by the main characters, although considering the number of random drug busts, the household can hardly be classified as sedate. The main characters solve (or attempted to solve) seven of these boredom problems by playing with one of three balls and solve three by playing with balloons. The state of being concerned is the least recurrent problem exhibited in the run.

---

[7]The reason that each run varies in length is that, after generating around 600,000 gensyms, Meta-AQUA will use all available swap space on the Symbolics and thus inadvertently halt the underlying LISP system. We then discard the story which is being processed at the time of the crash. The data from the remaining stories constitute the results of the run.

Table 1: Results from run number four

Story Number (sentences)[a]	Questions Posed (LG RL NL)	Answered Questions (LG RL NL)	Correct Answers (LG RL NL)	Question Points (LG RL NL)	Learning Episodes (LG RL NL)	Protagonist and Problem[b]
1 (26)	1 1 1	0 0 0	0 0 0	1 1 1	1 1 0	Mom bored (balloon)
2 (19)	3 3 3	3 2 2	1 0 0	7 5 5	2 3 0	Mom bored (ball)
3 (38B)	1 1 1	0 0 0	0 0 0	1 1 1	1 1 0	Elvis jonesing
4 (51b)	1 1 1	1 0 0	1 0 0	3 1 1	0 1 0	Dad jonesing
5 (21)	1 1 1	1 0 0	1 0 0	3 1 1	0 1 0	Mom bored (ball)
6 (13)	1 1 1	1 0 0	1 0 0	3 1 1	0 1 0	Officer1 concerned
7 (13)	1 1 1	1 0 0	1 0 0	3 1 1	0 1 0	Dad bored (ball)
8 (21)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Dad thirsty
9 (44B)	2 2 2	2 1 1	1 0 0	5 3 3	1 2 0	Dad thirsty
10 (51B)	3 3 3	2 1 1	2 1 0	7 5 4	0 1 0	Dad bored (balloon)
11 (11)	2 2 1	1 1 1	1 0 0	4 3 2	1 2 0	Lynn bored (ball)
12 (3)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Officer1 concerned
13 (47b)	2 2 1	1 1 0	1 1 0	4 4 1	0 0 0	Mom thirsty
14 (15)	4 4 4	4 2 3	4 0 0	12 6 7	0 4 0	Mom bored (ball)
15 (28)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Lynn jonesing
16 (42B)	2 2 2	2 1 1	2 1 0	6 4 3	0 1 0	Dad jonesing
17 (45b)	2 2 1	1 1 0	1 1 0	4 4 1	0 0 0	Elvis jonesing
18 (21)	2 2 2	2 1 1	2 1 0	6 4 3	0 1 0	Officer1 concerned
19 (20)	0 0 0	0 0 0	0 0 0	0 0 0	0 0 0	Dad jonesing
20 (52b)	2 2 1	1 0 0	1 0 0	4 2 1	0 1 0	Dad bored (balloon)
21 (39b)	2 2 1	1 1 1	1 1 1	4 4 3	1 1 0	Lynn jonesing
22 (17)	2 2 2	2 1 1	2 0 0	6 3 3	0 2 0	Dad bored (ball)
23 (40B)	2 2 2	1 1 1	1 1 0	4 4 3	1 1 0	Elvis thirsty
24 (38b)	2 2 1	1 1 0	1 0 0	4 3 1	0 1 0	Mom bored (ball)
Total 715	38 38 32	28 15 13	25 7 1	91 60 46	8 26 0	

a. The letter "B" means that the story contains an attempted drug bust by the police canine squad, whereas the letter "b" means that the officer entered the house alone to attempt a bust.

b. Items in parentheses represent games played to dispel boredom.

Table 2: Summary of results from run four

Learning Condition	Questions Posed	Answered Questions	Correct Answers	Total Question Points	Learning Episodes
LG	38	28	25	91	8
RL	38	15	7	60	26
NL	32	13	1	46	0

Table 2 summarizes the totals from Table 1.The dependent variable (column 5) shows that Meta-AQUA's performance under the LG condition is significantly greater than the performance under the RL condition. In turn, Meta-AQUA performance in the RL condition far exceeded the performance under the NL condition.

Alternatively, if only absolute performance (column 4) is considered, the differential is even greater. By this measure, the LG condition is more than three times the value of the RL condition, whereas, the performance of the NL condition is insignificant. By looking at column three, however, the numbers of questions answered in some way (right or wrong), are roughly equivalent in the RL and NL conditions, whereas the ratio of the LG condition to either of the other two is 2:1. Finally, the number of questions posed are virtually equal across all three conditions.

In contrast to these differences, Meta-AQUA attempts to learn from failure more than three times as often under the RL condition as under the LG condition. That is, learning is more *effective* with learning goals than without. In the RL condition, learning does not increase performance as much as does the LG condition, while concurrently, it leads Meta-AQUA to expend more resources by increasing the amount of learning episodes. Thus, the system works harder and gains less under RL than under LG.

Figure 5 graphs the accumulation of question points across trials (i.e., stories).[8] The behavior of the system as measured by the dependent variable is greatest under the LG condition, next best under RL, and worst under the NL condition. But, the trend does not hold for each trial. Figure 6 shows raw scores indicating that the NL condition actually outperforms the RL condition on trial number 14. The reason for this effect is that under worse-case conditions, if the interactions present between learning methods are negative, the performance may actually degrade. As a result, randomly ordered learning may be worse than no learning at all.

The differences as a function of the independent variable are even more pronounced if only accuracy (the number of correct answers) is examined and partial credit ignored. Figure 7 shows that under the RL condition, Meta-AQUA did not answer a question correctly until trial number 10, whereas under the NL condition, it did not perform correctly until trial 21. On the other hand, because under the LG condition the system learned a new explanation early in trial number 1, it was able to answer a question by trial number two. This striking result was facilitated by the random order of input (i.e., the second trial happened to be about the same problem as the first) as well as by computational introspection.

Overall Results. Table 3 summarizes the evaluation data from the six program runs. As is evident across all runs, the LG condition consistently outperforms the RL condition in the total cumulative question points. In turn, the RL condition outperforms the NL condition, despite the occasional poor performance due to negative interactions. As indicated by the standard deviations, the amount of differences between and within conditions exhibit high variability across runs.

Given these totals, the percent improvement for either learning condition over the NL base condition is simply the ratio of the difference in the base performance score and either score to the base score itself. Thus for run one, the ratio of the difference between the LG and NL conditions (35 points) to the NL condition (50 points) is .7, or 70 percent. Again, the improvement in performance for the LG condition is consistently higher than that of the RL condition. This difference is calculated in the final column. The differential is the percent improvement of the LG condition over the RL condition and is computed by the same measure as was the improvements in the individual learning conditions. That is, the differential is the ratio of the difference between the two improvements to the lower rate.[9] Thus, the differential between the LG rate of learning in run number one and that of the RL condition is the ratio of the difference (8 percentage points) to the RL percentage (62). Hence, the ratio is .129, or an improvement of nearly 13 percent.

Although the average differential between the two learning conditions (i.e., between fully-introspective and semi-introspective multistrategy learning) is more than 106 percent with a large standard deviation, this figure still overstates the difference. The expected gain in learning is more conservative. The differential between the average LG improvement (102.70) and the average RL improvement (65.67) is a 56.38 percent difference. That is, across a number of input conditions, the use of learning goals to order and combine learning choices should show about 1.5 times the improvement in performance than will a straight mapping of faults to repairs when interactions are present.

---

[8]Note that the final extent of all three curves reach the value of the triple in the totals column for column five.

[9]Note that this ratio can also be calculated as the difference between the performance scores of the learning conditions to the difference between the performance score of the RL and NL conditions. In other words, the ratio (LG-RL) / (RL-NL).

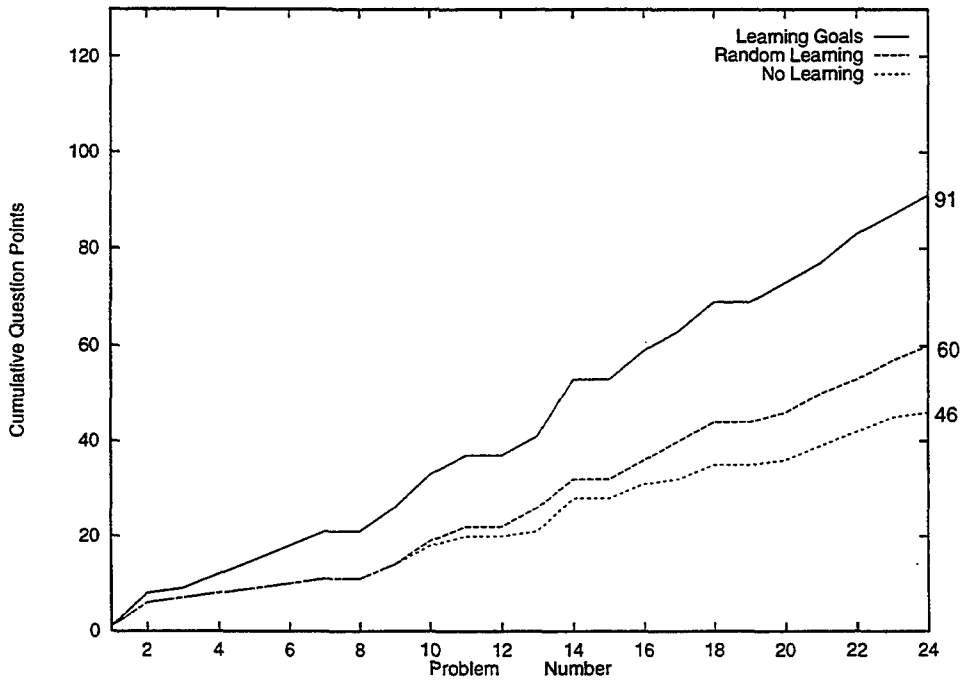


Figure 5. Run 4, cumulative question points as a function of the number of problems

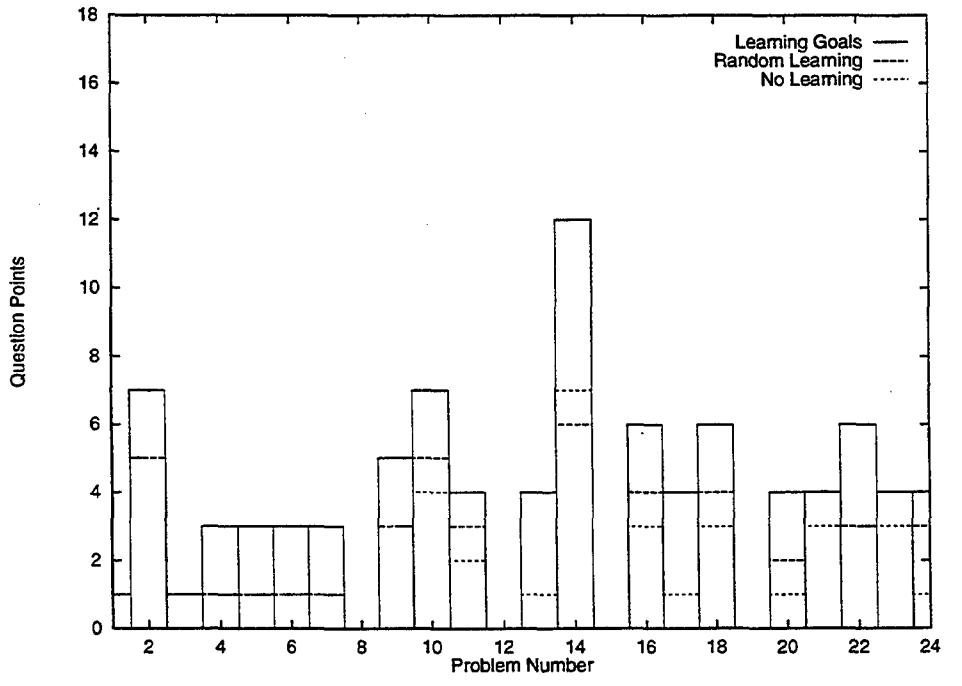


Figure 6. Run 4, question points histogram

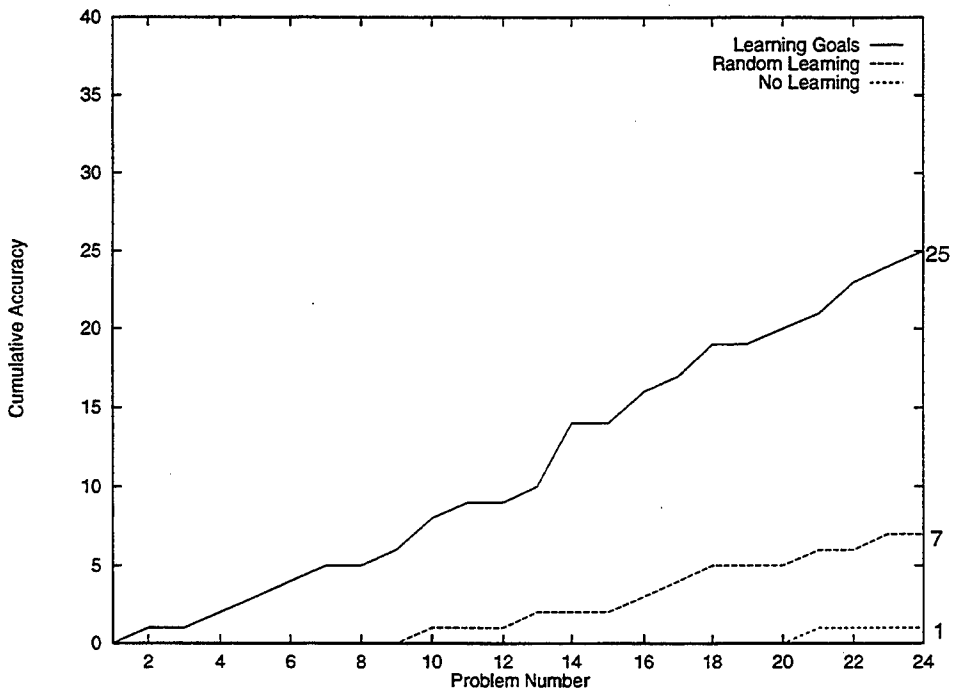


Figure 7. Run 4, cumulative correct answers (accuracy) as a function of the number of problems

Table 3: Summary of cumulative results

Run Number[a]	Cumulative Question Points			% LG Improved	% RL Improved	Improvement Differential %
	LG	RL	NL			
Run 1 (34)	85	81	50	70.00	62.00	12.90
Run 2 (30)	106	98	43	146.51	127.91	14.55
Run 3 (28)	120	102	60	100.00	70.00	42.86
Run 4 (24)	91	60	46	97.83	30.43	221.43
Run 5 (22)	57	49	27	111.11	81.48	36.36
Run 6 (28)	103	66	54	90.74	22.22	308.33
Averages	93.67	76.00	46.67	102.70	65.67	106.07
Std. Dev.	21.72	21.31	11.34	25.43	38.17	126.59

a. Amounts in parentheses indicate total number of stories in each run.

Summary and Discussion

The experiments reported in this paper provide a number of results that support the hypothesis that computational introspection facilitates multistrategy learning. Meta-AQUA expended more learning resources and induced less performance improvement without learning goals than it did under a condition that included them. Moreover, we have shown that because learning algorithms negatively interact, the arbitrary ordering of learning methods (i.e., as under the RL condition) can actually lead to worse system performance than no learning at all. Therefore, an explicit phase to decide exactly what to learn (i.e., to spawn learning goals or an equivalent mechanism) is *necessary* to avoid these interactions and to maintain effective learning in multistrategy environments. The paper also provided a novel quantitative measure with which to evaluate the comprehension process. As dependent variable, this partial credit metric provides rewards for both posing questions autonomously and giving some type of answer, as well as for getting answers correct.

Because of the considerable computational overhead involved in maintaining a reasoning trace, performing blame-assignment, spawning learning goals, and constructing a plan with which to pursue such goals, the benefits of using introspection must be substantial to justify the costs.[10] Furthermore, under extremely complex situations or in informationally impoverished circumstances, deciding on an optimal learning goal is certainly intractable. In such situations, it may be more beneficial to proceed without further reasoning, rather than to attempt to understand the exact causes of the failure. Knowing when a learning task is worth pursuing is itself an important skill to master for an intelligent system. Identifying the most appropriate conditions for the use of an introspective approach is therefore a desirable research goal. To establish only that introspection facilitates learning and that the model of introspection has some quality of reasonableness is not satisfactory. Although further inquiry into these conditions is left for future research, a number of remarks can be made at this time.

If the distributions of the kinds of failures generated by the performance task change the nature of the differences in the learning curves generated in the experiments used to establish the hypothesis, then applicability conditions can be established that predict when the utility of introspection exceeds its cost. The space of applicability conditions for introspection is expected to emerge from the taxonomy of causal factors presented in Ram, Cox, and Narayanan (1995). It has already been shown through the existing implementation that introspection in certain circumstances is tractable. Thus, a lower bound is already available. It is clearly not possible to reason in any effective manner if all possible failures occur at once or given an overly-sparse BK. So an analysis of the interaction of the taxonomized causal factors should result in a set of complex failures that can be programmed into Tale-Spin in order to produce various distributions of errors. Meta-AQUA is expected to have difficulty learning from some of the failure combinations within these error distributions. As with the ablation study, measures with and without introspection provide the independent variable for the evaluation of learning. The results should itemize the conjunctions of failure from which it is impossible to recover and those for which a reflexive or tightly coupled approach is more suited.

In the interim, a potential heuristic for deciding when to use an introspective approach is to qualitatively ascertain whether or not interactions between learning mechanisms available to the learner exist. If they exist, then the approach should be applied, otherwise a more reflexive approach is licensed. In speculation, another potential heuristic for determining that introspection is a win is to use a threshold for the number of failure symptoms above which introspection will not be attempted. Through experimentation, this threshold number should be obtained empirically given a distribution of known problem types and a random selection of problems from the distribution. The identification of such heuristics will enable the practical use of introspective methods in systems that cannot afford to squander precious resources with intractable computation.

[10] One must be cautious, however, when summarily dismissing introspection due to computational overhead costs alone. Doyle (1980) warns that to disregard the introspective component and self-knowledge in order to save the computational overhead in space, time, and notation is discarding the very information necessary to avoid combinatorial explosions in search (p. 30).

Acknowledgments

This research is supported by AFOSR under contract #F49620-94-1-0092 and by the Georgia Institute of Technology. I thank Ashwin Ram, Janet Kolodner, Mimi Recker, Kurt Eiselt, Tony Simon, Kenny Moorman, Juan Carlos Santamaria, Tucker Balch, and the anonymous reviewers for comments on various drafts of this paper. I also wish to gratefully acknowledge Mark Devaney's assistance in reprogramming Tale-Spin.

References

Carbonell, J. G. (1986). Derivational Analogy: A theory of Reconstructive Problem Solving and Expertise Acquisition. In R. Michalski, J. Carbonell and T. Mitchell, eds., *Machine learning II: An Artificial Intelligence Approach*, 371-392. San Mateo, CA: Morgan Kaufmann Publishers.

Chi, M. T. H. 1995. Revising the Mental Model as One Learns. Plenary address to the Seventeenth Annual Conference of the Cognitive Science Society. Pittsburgh (July 23).

Chi, M. T. H., Bassok, M., Lewis, M., Reimann, P., and Glasser, R. 1989. Self-Explanations: How Students Study and Use Examples in Learning to Solve Problems. *Cognitive Science* 13:145-182.

Cox, M. T. 1994. Machines that Forget: Learning from Retrieval Failure of Mis-Indexed Explanations. In A. Ram and K. Eiselt, eds., Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society, 225-230. Hillsdale, NJ: Lawrence Erlbaum Associates.

Cox, M. T. 1996. Introspective Multistrategy Learning: Constructing a Learning Strategy under Reasoning Failure. Ph.D. diss., Technical Report, GIT-CC-96-06, College of Computing, Georgia Institute of Technology, Atlanta. Available URL: ftp://ftp.cc.gatech.edu/pub/ai/ram/git-cc-96-06.html

Cox, M. T., and Ram, A. 1995. Interacting learning-goals: Treating learning as a planning task. In J.-P. Haton, M. Keane and M. Manago, eds., *Advances in case-based reasoning*, 60-74. Berlin: Springer-Verlag.

Doyle, J. 1980. A Model for Deliberation, Action, and Introspection. Ph.D. diss., Technical Report, TR-58, Department of Computer Science, Massachusetts Institute of Technology, Cambridge.

Kolodner, J. L. (1993). *Case-Based Reasoning*. San Mateo, CA: Morgan Kaufmann Publishers.

Lehnert, W., Dyer, M. G., Johnson, P., Yang, C., and Harley, S. 1983. BORIS - An Experiment in In-Depth Understanding of Narratives. *Artificial Intelligence* 20(1): 15-62.

McCarthy, J. 1959. Programs with Common Sense. In Symposium Proceedings on Mechanisation of Thought Processes, vol. 1, 77-84. London: Her Majesty's Stationary Office.

Meehan, J. 1981. Talespin. In R. C. Schank and C. Riesbeck, eds., *Inside Computer Understanding: Five Programs Plus Miniatures*, 197-258. Hillsdale, NJ: Lawrence Erlbaum Associates.

Michalski, R. S. 1994. Inferential theory of learning: Developing Foundations for Multistrategy Learning. In R. Michalski and G. Tecuci, eds., *Machine learning IV: A Multistrategy Approach*, 3-61. San Francisco: Morgan Kaufmann.

Minsky, M. L. 1965. Matter, Mind, and Models. In Proceedings of the International Federation of Information Processing Congress 1965, vol. 1, 45-49. (Original work from 1954)

Pazzani, M. 1994. Learning Causal Patterns: Making a Transition from Data-Driven to Theory-Driven Learning. In R. Michalski and G. Tecuci, eds., *Machine learning IV: A Multistrategy Approach*, 267-293. San Francisco: Morgan Kaufmann.

Ram, A. 1991. A Theory of Questions and Question Asking. *The Journal of the Learning Sciences* 1(3&4): 273-318.

Ram, A., and Cox, M. T. 1994. Introspective Reasoning Using Meta-Explanations for Multistrategy Learning. In R. S. Michalski and G. Tecuci, eds., *Machine learning IV: A Multistrategy Approach*, 349-377. San Francisco: Morgan Kaufmann.

Ram, A., Cox, M. T., and Narayanan, S. 1995. Goal-Driven Learning in Multistrategy Reasoning and Learning Systems. In A. Ram and D. Leake, eds., *Goal-Driven Learning*, 421-437. Cambridge, MA: MIT Press/Bradford Books.

Schank, R. C., and Riesbeck, C. eds. 1981. *Inside Computer Understanding: Five Programs Plus Miniatures*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Veloso, M. (1994). *Planning and Learning by Analogical Reasoning*. Berlin: Springer-Verlag.

Wilensky, R. 1978. Understanding Goal-Based Stories. Ph.D. diss., Technical Report, 140, Department of Computer Science, Yale University, New Haven, CT.

From Instances to Rules: A Comparison of Biases

Pedro Domingos

Department of Information and Computer Science
University of California, Irvine
Irvine, California 92717, U.S.A.
pedrod@ics.uci.edu
http://www.ics.uci.edu/~pedrod

Abstract

RISE is an algorithm that combines rule induction and instance-based learning (IBL). It has been empirically verified to achieve higher accuracy than state-of-the-art representatives of its parent approaches in a large number of benchmark problems. This paper investigates the conditions under which RISE's bias will be more appropriate than that of the pure approaches, through experiments in carefully controlled artificial domains. RISE's advantage compared to pure rule induction increases with increasing concept specificity. RISE's advantage compared to pure IBL is greater when the relevance of features is context-dependent (i.e., when some of the features used to describe examples are relevant only given other features' values). The paper also reports lesion studies and other empirical observations showing that RISE's good performance is indeed due to its combination of rule induction and IBL, and not to the presence of either component alone.

Introduction

Rule induction (either performed directly (Michalski 1983) or by means of decision trees (Quinlan 1993a)) and instance-based learning (Aha, Kibler, & Albert 1991) (forms of which are also known as case-based, memory-based, exemplar-based, lazy, local, and nearest-neighbor learning) constitute two of the leading approaches to concept and classification learning. Rule-based methods discard the individual training examples, and remember only abstractions formed from them. At performance time, rules are typically applied by logical match (i.e., only rules whose preconditions are satisfied by an example are applied to it). Instance-based methods explicitly memorize some or all of the examples; they generally avoid forming abstractions, and instead invest more effort at performance time in finding the most similar cases to the target one.

The two paradigms have largely complementary strengths and weaknesses. Rule induction systems often succeed in identifying small sets of highly predictive features, and, crucially, these features can vary from example to example. However, these methods can have trouble recognizing exceptions, or in general small, low-frequency sections of the space; this is known as the "small disjuncts problem" (Holte, Acker, & Porter 1989). Further, the general-to-specific, "separate and conquer" search strategy they typically employ causes them to suffer from the "fragmentation problem": as induction progresses, the amount of data left for further learning dwindles rapidly, leading to wrong decisions or insufficient specialization due to lack of adequate statistical support. On the other hand, IBL methods are well suited to handling exceptions, but can be very vulnerable to irrelevant features. If many such features are present in the example descriptions, IBL systems will be confused by them when they compare examples, and accuracy may suffer markedly. Unsurprisingly, in classification applications each approach has been observed to outperform the other in some, but not all, domains.

We believe that rule induction and instance-based learning have much more in common than a superficial examination reveals, and can be unified into a single, simple and coherent framework for classification learning, one that draws on the strengths of each to combat the limitations of the other. This unification rests on two key observations. One is that an instance can be regarded as a maximally specific rule (i.e., a rule whose preconditions are satisfied by exactly one example). Therefore, no syntactic distinction need be made between the two. The second observation is that rules can be matched approximately, as instances are in an instance-based classifier (i.e., a rule can match an example if it is the closest one to it according to some similarity-computing procedure, even if the example does not logically satisfy all of the rule's preconditions; see (Michalski *et al.* 1986)). A rule's extension, like an instance's, then becomes the set of examples that it is the most similar rule to, and thus there is also no necessary semantic distinction between a rule and an instance.

The RISE algorithm (Domingos 1995b) is a practical, computationally efficient realization of this idea.[1]

[1] Obviously, it is not the only possible approach to uni-

RISE starts with a rule base that is simply the training set itself, and gradually generalizes each rule to cover neighboring instances, as long as this does not increase the rule base's error rate on the known cases. If no generalizations are performed, RISE acts as a pure instance-based learner. If all cases are generalized and the resulting set of rules covers all regions of the instance space that have nonzero probability, it acts as a pure rule inducer. More generally, it will produce rules along a wide spectrum of generality; sometimes a rule that is logically satisfied by the target case will be applied, and in other cases an approximate match will be used. RISE's bias, which is in effect intermediate between that of pure rule inducers and that of pure instance-based learners, has been observed to lead to improvements in accuracy in a large number of domains from the UCI repository (Murphy & Aha 1995), resulting in significantly better overall results than either "parent" bias (with C4.5RULES (Quinlan 1993a) and CN2 (Clark & Boswell 1991) being used as representatives of rule induction, and PEBLS (Cost & Salzberg 1993) as a representative of IBL). RISE is described in greater detail in the next section.

The question now arises of exactly what factors RISE's comparative advantage is due to, and thus of when it will be appropriate to apply this algorithm instead of a pure IBL or a pure rule induction one. This will first be approached by showing through lesion studies that RISE's strength derives from the simultaneous presence of the two components, and not from either one alone. We will then consider rule induction and IBL in turn, formulating hypotheses as to the factors that favor RISE over the "atomic" approach, and testing these hypotheses through empirical studies in artificial domains, where these factors are systematically varied.

The RISE Algorithm

RISE's learning and classification procedures will be considered in turn. More details can be found in (Domingos 1995b; 1995a).

Representation and Search

Each example is a vector of attribute-value pairs, together with a specification of the class to which it belongs; attributes can be either nominal (symbolic) or numeric. Each rule consists of a conjunction of antecedents and a predicted class. Each antecedent is a condition on a single attribute, and there is at most one antecedent per attribute. Conditions on nominal attributes are equality tests of the form $a_i = v_j$, where a_i is the attribute and v_j is one of its legal values. Conditions on numeric attributes take the form of allowable intervals for the attributes, i.e., $a_i \in [v_{j1}, v_{j2}]$, where v_{j1} and v_{j2} are two legal values for a_i. Instances (i.e., examples used as prototypes for classification) are viewed as maximally specific rules, with conditions on all attributes and degenerate (point) intervals for numeric attributes. A rule is said to *cover* an example if the example satisfies all of the rule's conditions; a rule is said to *win* an example if it is the nearest rule to the example according to the distance metric that will be described below.

fying the two paradigms (cf. (Branting & Porter 1991; Golding & Rosenbloom 1991; Quinlan 1993b), etc.).

Table 1: The RISE algorithm.

Input: ES is the training set.

Procedure RISE (ES)

Let RS be ES.
Compute $Acc(RS)$.
Repeat
  For each rule R in RS,
    Find the nearest example E to R not already covered by it (and of the same class).
    Let R' = Most_Specific_Generalization(R, E).
    Let RS' = RS with R replaced by R'.
    If $Acc(RS') \geq Acc(RS)$
    Then Replace RS by RS',
      If R' is identical to another rule in RS,
      Then delete R' from RS.
Until no increase in $Acc(RS)$ is obtained.
Return RS.

The RISE algorithm is summarized in Table 1. RISE searches for "good" rules in a specific-to-general fashion, starting with a rule set that is the training set of examples itself. RISE looks at each rule in turn, finds the nearest example of the same class that it does not already cover (i.e., that is at a distance greater than zero from it), and attempts to minimally generalize the rule to cover it. The generalization procedure is outlined in Table 2. If the change's effect on global accuracy is positive, it is retained; otherwise it is discarded. Generalizations are also accepted if they appear to have no effect on accuracy; this reflects a simplicity bias. This procedure is repeated until, for each rule, attempted generalization fails.

A potential difficulty is that measuring the accuracy of a rule set on the training set requires matching all rules with all training examples, and this would entail a high computational cost if it was repeatedly done as outlined. Fortunately, at each step only the *change* in accuracy needs to be computed. Each example memorizes the distance to its nearest rule and its assigned class. When a rule is generalized, all that is necessary is then to match that single rule against all examples, and check if it wins any that it did not before, and what its effect on these is. Previously misclassified examples that are now correctly classified add to the

Table 2: Generalization of a rule to cover an example.

Inputs: $R = (a_1, a_2, \ldots, a_A, c_R)$ is a rule,
$E = (e_1, e_2, \ldots, e_A, c_E)$ is an example.
a_i is either True, $x_i = r_i$, or $r_{i,lower} \leq x_i \leq r_{i,upper}$.

Function Most_Specific_Generalization (R, E)

For each attribute i,
If a_i = True then Do nothing.
Else if i is symbolic and $e_i \neq r_i$ then a_i = True.
Else if $e_i > r_{i,upper}$ then $r_{i,upper} = e_i$.
Else if $e_i < r_{i,lower}$ then $r_{i,lower} = e_i$.

accuracy, and previously correctly classified examples that are now misclassified subtract from it. If the former are more numerous than the latter, the change in accuracy is positive, and the generalization is accepted. With this optimization, RISE's worst-case time complexity has been shown to be quadratic in the number of examples and the number of attributes, which is comparable to that of commonly-used rule induction algorithms (Domingos 1995b).

Classification

At performance time, classification of each test example is performed by finding the nearest rule to it, and assigning the example to the rule's class. The distance measure used is a combination of Euclidean distance for numeric attributes, and a simplified version of Stanfill and Waltz's value difference metric for symbolic attributes (Stanfill & Waltz 1986).

Let $E = (e_1, e_2, \ldots, e_A, c_E)$ be an example with value e_i for the ith attribute and class c_E. Let $R = (a_1, a_2, \ldots, a_A, c_R)$ be a rule with class c_R and condition a_i on the ith attribute, where a_i = True if there is no condition on i, otherwise a_i is $x_i = r_i$ if i is symbolic and a_i is $r_{i,lower} \leq x_i \leq r_{i,upper}$ if i is numeric. The distance $\Delta(R, E)$ between R and E is then defined as:

$$\Delta(R, E) = \sum_{i=1}^{A} \delta^2(i) \quad (1)$$

where the component distance $\delta(i)$ for the ith attribute is:

$$\delta(i) = \begin{cases} 0 & \text{if } a_i = \text{True} \\ SVDM(r_i, e_i) & \text{if } i \text{ is symbolic} \wedge a_i \neq \text{True} \\ \delta_{num}(i) & \text{if } i \text{ is numeric} \wedge a_i \neq \text{True} \end{cases} \quad (2)$$

$SVDM(r_i, e_i)$ is the simplified value difference metric, defined as:

$$SVDM(x_i, x_j) = \sum_{h=1}^{C} |P(c_h|x_i) - P(c_h|x_j)| \quad (3)$$

where x_i and x_j are any legal values of the attribute, C is the number of classes, c_h is the hth class, and $P(c_h|x_i)$ denotes the probability of c_h conditioned on x_i. The essential idea behind VDM-type metrics is that two values should be considered similar if they make similar class predictions, and dissimilar if their predictions diverge. This has been found to give good results in several domains (Cost & Salzberg 1993). Notice that, in particular, $SVDM(x_i, x_j)$ is always 0 if $i = j$.

The component distance for numeric attributes is defined as:

$$\delta_{num}(i) = \begin{cases} 0 & \text{if } r_{i,lower} \leq e_i \leq r_{i,upper} \\ \frac{e_i - r_{i,upper}}{x_{max} - x_{min}} & \text{if } e_i > r_{i,upper} \\ \frac{r_{i,lower} - e_i}{x_{max} - x_{min}} & \text{if } e_i < r_{i,lower} \end{cases} \quad (4)$$

x_{max} and x_{min} being respectively the maximum and minimum observed values for the attribute.

The distance from a missing numeric value to any other is defined as 0. If a symbolic attribute's value is missing, it is assigned the special value "?". This is treated as a legitimate symbolic value, and its distance to all other values of the attribute is computed and used. When coupled with SVDM, this is a sensible policy: a missing value is taken to be roughly equivalent to a given possible value if it behaves similarly to it, and inversely if it does not.

When two or more rules are equally close to a test example, the rule that was most accurate on the training set wins. So as to not unduly favor more specific rules, the Laplace-corrected accuracy is used (Niblett 1987):

$$LAcc(R) = \frac{N_{corr}(R) + 1}{N_{won}(R) + C} \quad (5)$$

where R is any rule, C is the number of classes, $N_{won}(R)$ is the total number or examples won by R, $N_{corr}(R)$ is the number of examples among those that R correctly classifies, and C is the number of classes. The effect of the Laplace correction is to make the estimate of a rule's accuracy converge to the "random guess" value of $1/C$ as the number of examples won by the rule decreases. Thus rules with high apparent accuracy are favored only if they also have high statistical support, i.e., if that apparent accuracy is not simply the result of a small sample.

Lesion Studies

Lesion studies were conducted using 30 datasets from the UCI repository (Murphy & Aha 1995). Several aspects of the algorithm's performance were also measured. The results are shown in Table 3. Superscripts indicate significance levels for the accuracy differences between systems, using a one-tailed paired t

Table 3: Results of lesion studies, and performance monitoring. Superscripts denote significance levels: 1 is 0.5%, 2 is 1%, 3 is 2.5%, 4 is 5%, 5 is 10%, and 6 is above 10%.

	Accuracy of subsystem				Match type frequency			
Domain	RISE	IBL	Rules	No tie-b.	No/One	No/Multi	One	Multi
Audiology	77.0	75.8^{5}	55.1^{1}	76.2^{1}	53.6	1.6	43.0	1.8
Annealing	97.4	97.7^{4}	77.7^{1}	97.2^{1}	24.1	0.0	75.6	0.2
Breast cancer	67.7	65.1^{1}	69.0^{4}	68.7^{1}	33.4	1.5	59.1	6.0
Credit screening	83.3	81.3^{1}	66.9^{1}	83.2^{6}	51.9	0.0	46.9	1.1
Chess endgames	98.2	91.9^{1}	91.9^{1}	98.0^{1}	27.6	0.1	71.4	0.9
Pima diabetes	70.4	70.3^{6}	66.2^{1}	70.5^{1}	70.9	0.1	27.4	1.6
Echocardiogram	64.6	59.2^{1}	65.7^{6}	64.5^{6}	70.1	0.1	26.8	3.0
Glass	70.6	68.3^{2}	47.3^{1}	70.4^{2}	71.5	0.0	27.2	1.3
Heart disease	79.7	77.8^{1}	64.7^{1}	79.7^{6}	63.1	0.0	34.8	2.1
Hepatitis	78.3	78.4^{6}	79.6^{5}	78.5^{5}	55.3	0.1	43.1	1.5
Horse colic	82.6	76.6^{1}	79.0^{1}	81.7^{1}	39.7	0.2	55.4	4.6
Thyroid disease	97.5	94.1^{4}	84.8^{1}	97.5^{6}	40.7	0.1	58.3	0.9
Iris	94.0	94.7^{3}	71.0^{1}	94.0^{6}	45.9	0.0	54.0	0.2
Labor neg.	87.2	90.8^{1}	73.7^{1}	87.1^{6}	51.8	0.2	46.6	1.4
Lung cancer	44.7	42.0^{6}	26.5^{1}	44.2^{5}	88.2	0.4	9.1	2.4
Liver disease	62.4	60.9^{4}	62.1^{6}	62.3^{6}	72.7	0.2	24.0	3.0
Contact lenses	77.2	72.5^{1}	64.8^{1}	75.8^{3}	13.2	1.5	83.0	2.2
LED	59.9	55.9^{1}	52.7^{1}	49.7^{1}	14.7	4.5	47.3	33.5
Lymphography	78.7	82.0^{1}	70.1^{1}	78.2^{3}	42.9	0.4	53.9	2.8
Mushroom	100.0	97.5^{6}	100.0^{6}	99.8^{1}	7.3	0.0	92.5	0.2
Post-operative	64.1	59.1^{1}	70.9^{1}	65.9^{1}	36.5	12.2	44.7	6.6
Promoters	86.8	90.6^{1}	68.4^{1}	85.9^{4}	59.7	0.0	35.8	4.5
Primary tumor	40.3	34.3^{1}	33.5^{1}	37.0^{1}	34.2	4.1	41.6	20.1
Solar flare	71.6	71.1^{6}	65.5^{1}	68.1^{1}	16.7	2.9	59.9	20.4
Sonar	77.9	83.8^{1}	52.9^{1}	77.9^{6}	95.6	0.0	4.3	0.1
Soybean	100.0	100.0^{6}	85.1^{1}	100.0^{6}	16.9	0.0	83.1	0.0
Splice junctions	93.1	87.8^{1}	75.9^{1}	92.1^{1}	67.4	0.0	29.9	2.7
Voting records	95.2	94.6^{3}	83.7^{1}	94.2^{1}	7.4	0.1	90.3	2.1
Wine	96.9	95.1^{1}	51.5^{1}	96.9^{6}	78.1	0.0	21.9	0.0
Zoology	93.9	94.5^{4}	81.0^{1}	93.7^{4}	17.5	0.0	81.9	0.5

test.[2] The first four columns compare the full system's accuracy ("RISE") with that obtained using lesioned versions: the IBL component alone ("IBL"), the rule induction component alone ("Rules"), and disabling the tie-breaking procedure ("No tie-b."). The last four columns all refer to the full RISE system, and show, respectively: the percentage of test cases that were not matched by any rule, but had a single closest rule, or for which all equally close rules were of the same class ("No/One"); the percentage not matched by any rule, and for which there were equally close rules of more than one class ("No/Multi"); the percentage matched by only one rule, or rules of only one class ("One"); and the percentage matched by rules of more than one class ("Multi"). These observations aid in interpreting the lesion study results.

These results are more easily understood by summarizing them in a few comparative measures. These are shown in Table 4. The first line shows the number of domains in which RISE achieved higher accuracy than the corresponding system, vs. the number in which the reverse happened. The second line considers only those domains in which the observed difference is significant at the 5% level or lower. The third line shows the global significance levels obtained by applying a Wilcoxon signed-ranks test (DeGroot 1986) to the 30 accuracy differences observed. The average accuracy across all domains is a measure of debatable significance, but it is often reported, and is shown on the last line.

The first specific question addressed was whether

[2] Since, in each case, the goal is to determine whether RISE's accuracy is *higher* than that of the lesioned system (and not just different from it in either direction), a one-tailed test is the appropriate choice (rather than a two-tailed one).

Table 4: Summary of lesion study results.

Measure	RISE	IBL	Rules	No t.-b.
No. wins	-	21-8	25-4	20-4
No. sig. wins	-	16-7	24-2	15-3
Wilcoxon test	-	0.5%	0.1%	0.2%
Average	79.7	78.1	67.9	79.0

there is any gain in the rule induction process (i.e., whether RISE constitutes an improvement over pure instance-based learning). The "IBL" column in Table 3 reports the accuracies obtained by the initial, ungeneralized instance set, and shows that generalization often produces significant gains in accuracy, while seldom having a negative effect.

The converse question is whether the instance-based component is really necessary. Simply assigning examples not covered by any rule to a default class, as done in most rule induction systems, might be sufficient. The "Rules" column in Table 3 shows the results obtained using this policy, and confirms the importance of "nearest-rule" classification in RISE. The sum of the "No" columns in the right half of Table 3 is the percentage of test cases assigned to the default class. This is often very high, the more so because RISE tends to produce rules that are more specific than those output by general-to-specific inducers. The use of nearest-rule is thus essential. Note that the results reported in the "Rules" column are for applying the default rule during both learning and classification; applying it exclusively during classification produced only a slight improvement.

Another important component of RISE whose utility needs to be determined is the conflict resolution policy, which in RISE consists of letting the tied rule with the highest Laplace accuracy win. This was compared with simply letting the most frequent class win ("No tie-b." column in Table 3). The sum of the "Multi" columns in the right half of Table 3 is the percentage of cases where tie-breaking is necessary. This is typically small, and the increase in accuracy afforded by RISE's conflict resolution strategy is correspondingly small (0.7% on average, for all datasets). However, this increase is consistently produced, as evinced by the fact that RISE is more accurate than its lesioned version with a 0.2% significance by the Wilcoxon test.

Taken together, the lesion studies show that each of RISE's components is essential to its performance, and that it is their combination in one system that is responsible for the excellent results obtained by RISE *vis-à-vis* other approaches.

RISE as Rule Induction

Our hypothesis is that RISE's advantage relative to "divide and conquer" rule induction algorithms is at least in part due to its greater ability to identify small

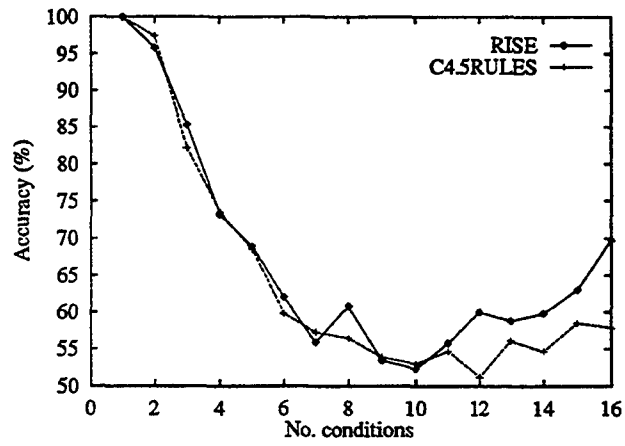


Figure 1: Accuracy as a function of concept specificity (16 features).

regions in the instance space (i.e., regions that are represented by few examples in the training set). Thus RISE should be more accurate than a "divide and conquer" algorithm when the target concepts are fairly to very specific, with the advantage increasing with specificity. Thus the independent variable of interest is the specificity of the target concept description. A good operational measure of it is the average length of the rules comprising the correct description: rules with more conditions imply a more specific concept. The dependent variables are the out-of-sample accuracies of RISE and of a "divide and conquer" algorithm; C4.5RULES (Quinlan 1993a) was used as the latter. Concepts defined as Boolean functions in disjunctive normal form were used as targets. The datasets were composed of 100 examples described by 16 attributes. The average number of literals C in each disjunct comprising the concept was varied from 1 to 16. The number of disjuncts was set to $\text{Min}\{2^{C-1}, 25\}$. This attempts to keep the fraction of the instance space covered by the concept roughly constant, up to the point where it would require more rules than could possibly be learned. Equal numbers of positive and negative examples were included in the dataset, and positive examples were divided evenly among disjuncts. In each run a different target concept was used, generating the disjuncts at random, with length given by a binomial distribution with mean C and variance $C(1-\frac{C}{16})$; this is obtained by including each feature in the disjunct with probability $\frac{C}{16}$. Twenty runs were conducted, with two-thirds of the data used for training and the remainder for testing.

The results are shown graphically in Fig. 1. The most salient aspect is the large difference in difficulty between short and long rules for both learners. Concepts with very few (approx. three or less) conditions

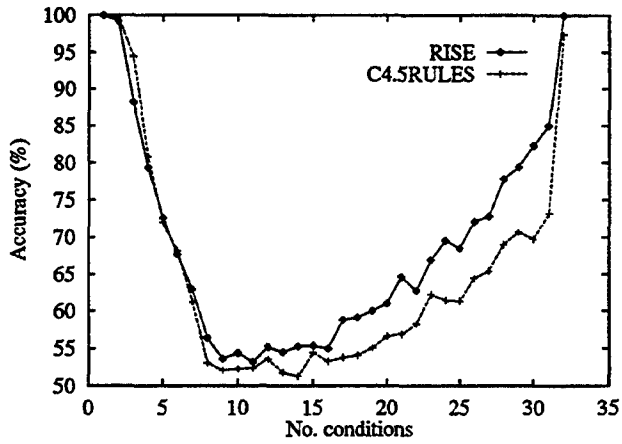


Figure 2: Accuracy as a function of concept specificity (32 features).

per rule are so simple that both RISE and C4.5RULES are able to learn them easily. In separate experiments, corrupting the data with 10% and 20% noise degraded the performance of the two algorithms equally, again giving no advantage to C4.5RULES. At the other end, however, RISE has a clear advantage for concepts with 12 or more conditions per rule; all differences here are significant at the 5% level using a one-tailed paired t test.[3]

The slight upward trend in C4.5RULES's curve for $C > 10$ was investigated by repeating the experiments with 32 attributes, 400 examples, a maximum of 50 rules and $C = 1, \ldots, 32$. The results are shown in Fig. 2. C4.5RULES's lag increases, but the upward trend is maintained; on inspection of the rules C4.5RULES produces, this is revealed to be due to the fact that, as the concept rules become more and more specific, it becomes possible to induce short rules for its negation. The hardest concepts, for which both the concept and its negation have necessarily long rules, are for intermediate values of C.

In summary, the results of this study support the hypothesis that the specificity of the regions to be learned is a factor in the difference in accuracy between RISE and "divide and conquer" rule induction systems, with greater specificity favoring RISE.

RISE as IBL

High sensitivity to irrelevant features has long been recognized as IBL's main problem. A natural solution is identifying the irrelevant features, and discarding them before storing the examples for future use. Several algorithms have been proposed for this purpose (see (Kittler 1986) for a survey), of which two of the most widely known are forward sequential search (FSS) and backward sequential search (BSS) (Devijver & Kittler 1982). Many variations of these exist (e.g., (Aha & Bankert 1994)). Their use can have a large positive impact on accuracy. However, all of these algorithms have the common characteristic that they ignore the fact that some features may be relevant only in context (i.e., given the values of other features). They may discard features that are highly relevant in a restricted sector of the instance space because this relevance is swamped by their irrelevance everywhere else. They may retain features that are relevant in most of the space, but unnecessarily confuse the classifier in some regions.

Consider, for example, an instance space defined by a set of numeric features $\mathbf{F}$, and a class composed of two hyperrectangles, one of which is defined by intervals $f_i \in [a_i, b_i]$ in a subset $\mathbf{F_1}$ of the features, and the other by intervals in a subset $\mathbf{F_2}$ disjoint from the first. Current feature selection algorithms would retain all features in $\mathbf{F_1}$ and $\mathbf{F_2}$, because each of those features is relevant to identifying examples in one of the hyperrectangles. However, the features in $\mathbf{F_2}$ act as noise when identifying examples defined by $\mathbf{F_1}$, and vice-versa. Instead of storing the same set of features for all instances, a better algorithm would discard the features in $\mathbf{F_2}$ from the stored instances of the first hyperrectangle, and the features in $\mathbf{F_1}$ from those of the second one. RISE has this capability.

Our hypothesis is that, viewed as an instance-based learner, RISE derives strength from its ability to perform context-sensitive feature selection (since different examples may be covered by different rules, and thus different features will be used in their classification). Thus, RISE's advantage relative to IBL using conventional feature selection methods should increase with the degree of context sensitivity of feature relevance. To empirically investigate this hypothesis, a concrete measure of the latter is required. If the target concept description is composed of a set of prototypes, one such possible measure is the average D for all pairs of prototypes of the number of features that appear in the definition of one, but not the other:

$$D = \frac{2}{P(P-1)} \sum_{i=1}^{P} \sum_{j=1}^{i-1} \sum_{k=1}^{F} d_{ijk} \qquad (6)$$

where P is the number of prototypes, F is the total number of features, and d_{ijk} is 1 if feature k appears in the definition of prototype i but not in that of prototype j or vice-versa, and 0 otherwise. This "feature difference" measure was taken as the independent variable in the study.

RISE's pure IBL component (see the section on lesion studies) was taken as the basic instance-based learner, and FSS and BSS were applied to it. For comparison, RISE's generalization procedure was also applied, but in order to ensure the fairness of this pro-

[3] See the previous footnote regarding this test.

cedure, all aspects of RISE that do not relate to feature selection were disabled: numeric features were not generalized to intervals, but either retained as point values or dropped altogether,[4] generalization for each rule stopped as soon as an attempted feature deletion for that rule failed (as opposed to only when attempts failed for all rules simultaneously), and duplicate rules were not deleted. The resulting simplified algorithm will hereafter be referred to as "RC". Thus the dependent variables of interest were the accuracies of RC, FSS and BSS.

Two-class problems were considered, with 100 examples in each dataset, described by 32 features. In each domain, each feature was chosen to be numeric or Boolean with equal probability (i.e., the number of numeric features is a binomial variable with expected value $F/2$ and variance $F/4$). Class 1 was defined by ten clusters, and class 0 was the complement of class 1. Each prototype or cluster was defined by a conjunction of conditions on the relevant features. The required value for a Boolean feature was chosen at random, with 0 and 1 being equally probable. Each numeric feature i was required to fall within a given range $[a_i, b_i]$, with a_i being the smaller of two values chosen from the interval [-1, 1] according to a uniform distribution, and b_i the larger one. A cluster was thus a hyperrectangle in the relevant numeric subspace, and a conjunction of literals in the Boolean one.

The choice of relevant features for each prototype was made at random, but in a way that guaranteed that the desired value of D for the set of prototypes was maintained on average. The feature difference D was varied from 0 to 8, the latter being the maximum value that can be produced given the number of features and prototypes used. Twenty domains were generated for each value of D, and two-thirds of the examples used as the training set. The average accuracy of RC, FSS and BSS on the remaining examples is shown graphically as a function of D in Figure 3.

All differences in accuracy between RC and FSS are significant at the 5% level, as are those between RC and BSS for $D = 1$, 2, 4, 5, and 8. The smallest difference occurs when $D = 0$, as our hypothesis would lead us to expect. All accuracies are negatively correlated with D, but the absolute value of the correlation is much smaller for RC (0.49) than for FSS and BSS (0.89 and 0.82, respectively). The downward slope of the regression line for RC's accuracy as a function of D (–0.35) is also much smaller than that for FSS (–1.21) and BSS (–0.61). We thus conclude that RC's higher performance is indeed at least partly due do its context sensitivity.

[4]The policy adopted was to compute the mean and standard deviation of each numeric feature from the sample in the training set, and attempt dropping the feature only when the values for the rule and the example to which its generalization is being tried differ by more than one standard deviation.

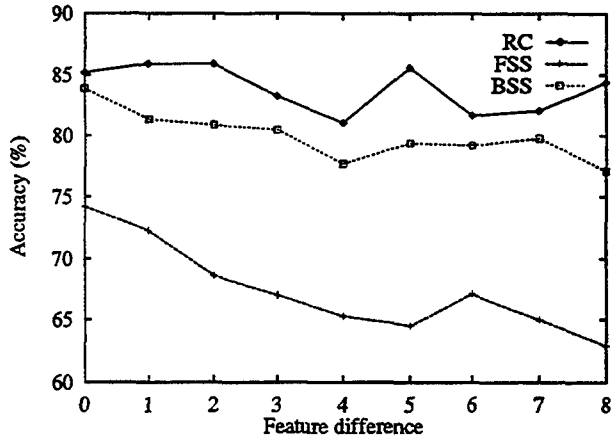


Figure 3: Accuracy as a function of context dependency.

Conclusion

In this paper we investigated the bias of RISE, an algorithm that combines rule induction and instance-based learning, and has been observed to achieve higher accuracies than state-of-the-art representatives of either approach. Lesion studies using benchmark problems showed that each of the two components is essential to RISE's high performance. Studies in carefully controlled artificial domains provided evidence for the hypothesis that, compared to rule inducers, RISE's strength lies in its ability to learn fairly to highly specific concepts, and, compared to instance-based learners, in its ability to detect context dependencies in feature relevance.

Directions for future research include: elucidating further factors in the differential performance of RISE relative to rule induction and IBL; repeating the experiments described here with a wider variety of rule and instance-based learners and artificial domains; and bringing further types of learning into RISE's framework, including in particular the use of analytical learning from expert-supplied domain knowledge.

Acknowledgments

This work was partly supported by a JNICT/PRAXIS XXI scholarship. The author is grateful to Dennis Kibler for many helpful comments and suggestions, and to all those who provided the datasets used in the lesion studies. Please see the documentation in the UCI Repository for detailed information.

References

Aha, D. W., and Bankert, R. L. 1994. Feature selection for case-based classification of cloud types: An empirical comparison. In *Proceedings of the 1994*

AAAI Workshop on Case-Based Reasoning, 106–112. Seattle, WA: AAAI Press.

Aha, D. W.; Kibler, D.; and Albert, M. K. 1991. Instance-based learning algorithms. *Machine Learning* 6:37–66.

Branting, L. K., and Porter, B. W. 1991. Rules and precedents as complementary warrants. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 3–9. Anaheim, CA: AAAI Press.

Clark, P., and Boswell, R. 1991. Rule induction with CN2: Some recent improvements. In *Proceedings of the Sixth European Working Session on Learning*, 151–163. Porto, Portugal: Springer-Verlag.

Cost, S., and Salzberg, S. 1993. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning* 10:57–78.

DeGroot, M. H. 1986. *Probability and Statistics*. Reading, MA: Addison-Wesley, 2nd edition.

Devijver, P. A., and Kittler, J. 1982. *Pattern Recognition: A Statistical Approach*. Englewood Cliffs, N.J.: Prentice/Hall.

Domingos, P. 1995a. The RISE 2.0 system: A case study in multistrategy learning. Technical Report 95-2, Department of Information and Computer Science, University of California at Irvine, Irvine, CA.

Domingos, P. 1995b. Rule induction and instance-based learning: A unified approach. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1226–1232. Montréal, Canada: Morgan Kaufmann.

Golding, A. R., and Rosenbloom, P. S. 1991. Improving rule-based systems through case-based reasoning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 22–27. Menlo Park, CA: AAAI Press.

Holte, R. C.; Acker, L. E.; and Porter, B. W. 1989. Concept learning and the problem of small disjuncts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 813–818. Detroit, MI: Morgan Kaufmann.

Kittler, J. 1986. Feature selection and extraction. In Young, T. Y., and Fu, K. S., eds., *Handbook of Pattern Recognition and Image Processing*. New York, NY: Academic Press.

Michalski, R. S.; Mozetic, I.; Hong, J.; and Lavrac, N. 1986. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, 1041–1045. Philadelphia, PA: AAAI Press.

Michalski, R. S. 1983. A theory and methodology of inductive learning. *Artificial Intelligence* 20:111–161.

Murphy, P. M., and Aha, D. W. 1995. UCI repository of machine learning databases. Machine-readable data repository, Department of Information and Computer Science, University of California at Irvine, Irvine, CA.

Niblett, T. 1987. Constructing decision trees in noisy domains. In *Proceedings of the Second European Working Session on Learning*, 67–78. Bled, Yugoslavia: Sigma.

Quinlan, J. R. 1993a. *C4.5: Programs for Machine Learning*. San Mateo, CA: Morgan Kaufmann.

Quinlan, J. R. 1993b. Combining instance-based and model-based learning. In *Proceedings of the Tenth International Conference on Machine Learning*, 236–243. Amherst, MA: Morgan Kaufmann.

Stanfill, C., and Waltz, D. 1986. Toward memory-based reasoning. *Communications of the ACM* 29:1213–1228.

Multistrategy Learning to Apply Cases for Case-Based Reasoning

David B. Leake, Andrew Kinley, and David Wilson
Computer Science Department
Lindley Hall 215, Indiana University
Bloomington, IN 47405
{leake,akinley,davwils}@cs.indiana.edu

Abstract

Investigations of learning in case-based reasoning (CBR) have traditionally focused on learning two types of knowledge: new cases and new indexing criteria for case retrieval. However, there is increasing recognition that other types of knowledge also play crucial roles in the case-based reasoning process. The effectiveness of a CBR system depends not only on having and retrieving relevant cases, but also on selecting which retrieved cases to apply and determining how to adapt them to fit new situations. Consequently, case-based reasoning can benefit from using multiple learning strategies to acquire, in addition to new cases and indices, new case adaptation strategies and similarity criteria. This paper describes ongoing research that studies how multiple types of learning can improve the case-based reasoning process and examines their interrelationship in contributing to the overall performance of a CBR system.

Introduction

Case-based reasoning (CBR) solves new problems by retrieving records of similar prior problem-solving episodes and adapting their solutions to fit new needs. Learning by acquiring new cases is a fundamental part of case-based reasoning, and the process of learning by case acquisition has been a central focus of CBR research. Recently, however, there has been increasing awareness of the importance of multiple types of knowledge to guide the CBR process. For example, Richter (1995) points out that the knowledge of a case-based reasoner is contained not only in its case base and indexing scheme, but also in the reasoner's similarity metric and in its case adaptation knowledge. Thus an important question is how these types of knowledge may be acquired.

The need to acquire multiple types of knowledge in CBR provides a natural opportunity for multistrategy learning (e.g., Michalski & Tecuci, 1994). Multistrategy learning can enable CBR systems to learn not only new cases, but also how to retrieve cases more reliably, how to judge the relevance of candidate cases more perspicaciously, and how to adapt cases to new situations more effectively. Using different learning strategies for each type of knowledge enables a CBR system to tailor its learning according to the task requirements of different parts of its reasoning process.

Because the component steps of CBR are strongly related, multistrategy learning has an added benefit as well: learning that improves one component may help overcome deficiencies in others. For example, better retrieval can reduce the need for adaptation knowledge, by providing more relevant cases; conversely, better adaptation knowledge can decrease the need for high-quality retrieval, by generating successful solutions even if less than ideal cases are retrieved.

Our research studies how multiple types of learning can improve the case-based reasoning process and examines their interrelationship in contributing to the overall performance of a CBR system. We are investigating how introspective reasoning during CBR can identify needs for information during case adaptation and satisfy them by introspective question transformation, a memory search process involving strategically redescribing needed information to guide memory search. We are also studying how memory search strategies, case adaptation strategies, and similarity criteria can be learned from experience.

This paper first describes our task domain and the basic structure of our testbed system, and summarizes our system's learning strategies and their relationships. It then describes how the system learns to improve its case adaptation process, discusses the effects of case adaptation learning for a small initial set of test examples, and describes how adaptation learning determines new similarity criteria. The paper closes by placing our approach in context of other research on learning from multiple parts of the CBR process.

Task and System Overview

Our system's task domain is disaster response planning. Disaster response planning is the initial strategic planning used to determine how to assess damage, evacuate victims, etc., in response to natural and man-made disasters such as earthquakes and chemical spills. There are no hard-and-fast rules for disaster response planning, and human disaster response planners appear to depend heavily on prior experiences when they address new problem situations (Rosenthal, Charles, & Hart 1989).

Our testbed system, DIAL,[1] processes a conceptual representation of a news story describing the initial events in a disaster, and proposes a response plan by retrieving and adapting the response plan for a similar prior disaster. DIAL includes a simple schema-based story understander, a response plan retriever and instantiator, a simple evaluator for candidate response plans, and an adaptation component to adapt plans when problems are found. Its basic processing sequence is as follows:

- A story is input to the system.
- Candidate response plan cases for similar problem situations are retrieved, using coarse-grained static similarity assessment criteria.
- A finer-grained similarity assessment process uses learned information about difficulty of adaptation to select the candidate case whose response plan is expected to be easiest to adapt.
- Problems in the response plan of the selected case are repaired by case adaptation. During adaptation, DIAL learns by storing traces of its case adaptation process and of the memory search process used to find needed information. If its adaptation attempt fails, DIAL can also learn by recording a trace of a user-guided adaptation process.
- The resulting response plan case is stored for future reuse by transformational analogy.

The system's case-based planning framework is based in a straightforward way on previous case-based planners using transformational analogy, such as CHEF (Hammond 1989). Consequently, we will not discuss DIAL's planning process per se, but instead will focus on how it learns to improve its case adaptation, memory search, and similarity assessment.

Learning Methods and Relationships

Five steps of DIAL's reasoning process involve knowledge transmutations (Michalski 1994) and learning, using a mix of learning strategies:

1. **Response plan learning:** The "baseline" learning method for DIAL is learning by case acquisition, the normal learning of case-based reasoning systems. Response plans are reapplied by **transformational analogy** (Carbonell 1983).
2. **Memory search:** When information is needed from memory (e.g., to adapt a case to a new situation), DIAL identifies the needed information and generates explicit *knowledge goals* (Hunter 1990; Ram 1987) for that information. Its memory search process transforms descriptions of needed information and of known information in memory, using self-knowledge about the system's memory organization, in order to satisfy its knowledge goals. This **introspective question transformation** process is based on the goal-driven learning principle of reasoning strategically about how to satisfy needs for information (desJardins 1992; Ram & Leake 1995).
3. **Memory search strategy learning:** When DIAL generates a memory search plan, it stores a trace of that plan as a *memory search case* for reuse by **derivational analogy** (Carbonell 1986).
4. **Adaptation learning:** DIAL adapts cases by an introspective reasoning process that determines which transformations to apply and which knowledge goals must be satisfied to apply them. If DIAL cannot generate an acceptable adaptation, it asks a user to guide the case adaptation process interactively. Traces of internally-generated adaptations and of user adaptations are stored as *adaptation cases* for reuse by **derivational analogy**.
5. **Similarity learning:** When a case-based reasoning system performs similarity assessment to determine which case is most similar to a new situation, the goal is to select the case that will be easiest to adapt (Birnbaum *et al.* 1991; Leake 1995a; Smyth & Keane 1995). DIAL's similarity assessment process uses prior experiences with case adaptation to estimate case adaptation cost for new problems, by a **transformational analogy** process. Consequently, adaptation learning and similarity learning are coupled: when adaptation cases are learned, that learning provides not only knowledge to use during future case adaptation, but to use during similarity assessment as well.

Table 1 summarizes DIAL's learning processes and the mechanisms used. DIAL's response plan cases, memory search cases, and case adaptation cases can be reused independently, allowing different lessons drawn

[1] For Disaster response with Introspective Adaptation Learning.

Process	Learning mechanism
Response plan learning	CBR/transformational analogy
Memory search process	Introspective question transformation
Memory search strategy learning	CBR/derivational analogy
Adaptation learning	CBR/derivational analogy, applied to traces of internal processing or user adaptations
Similarity learning	CBR/transformational analogy

Table 1: DIAL's processes and learning mechanisms.

from a single episode to be reapplied where most appropriate in the future. DIAL's close coupling of adaptation cases and similarity criteria enables the selection of response plan cases based on the state of its case adaptation knowledge. Each aspect of learning complements the others in supporting the CBR process as a whole. The following sections discuss specific aspects of DIAL's learning in more detail.

Learning to Adapt Cases to New Situations

Case adaptation is important to CBR because the ability of CBR systems to solve novel problems depends on adapting prior solutions to fit new circumstances. Unfortunately, hand-coding appropriate case adaptation knowledge has proven to be very difficult, to the point that experts in both CBR research (e.g., Kolodner, 1991) and applications (e.g., Barletta, 1994; Mark et al., 1996) agree that it is not currently practical to deploy CBR applications with automatic adaptation. Thus there is strong practical motivation for developing effective methods for learning to improve case adaptation.

Our approach to case adaptation learning models a transition from general (but non-operational) adaptation knowledge to more specific and operational knowledge. The initial knowledge for our approach is a small set of abstract transformation rules and memory search methods. When presented with a new adaptation problem, our system first selects a transformation rule to apply and then performs memory search to find the information needed to operationalize the transformation rule and apply it to the problem at hand (e.g., if a *substitution* transformation is selected, to find what to substitute). The system learns to improve its adaptation capabilities by case-based reasoning applied to the case adaptation process itself: a trace of the steps used in solving an adaptation problem is saved to be reused by derivational analogy when similar adaptation problems arise in the future (Leake 1995b; Leake, Kinley, & Wilson 1995). In this way, a CBR system doing adaptation can acquire specific adaptation procedures starting from domain-independent "weak methods" for adaptation when no specific knowledge is available. At the same time, traces of the memory search process are stored for future reuse, to facilitate building up new adaptations in the future.

DIAL's adaptation component takes two inputs: an instantiated disaster response plan and a description of the problems in the response plan that must be repaired. When presented with an adaptation problem, DIAL's adaptation component performs the following steps:

1. **Case-based adaptation:** DIAL first attempts to retrieve an adaptation case that applied successfully to a similar adaptation problem. If retrieval is successful, the adaptation process traced by that case is re-applied and processing continues with step 3.
2. **Rule-based adaptation:** When no relevant prior case is retrieved, DIAL selects a transformation associated with the type of problem that is being adapted. (E.g., it may decide to *substitute* a new plan step for one that does not apply.) Given the transformation, the program generates a knowledge goal for the information needed to apply the transformation. (E.g., when performing a substitution, the knowledge goal is to find an object that satisfies all the case's constraints on the component being replaced.)

   The knowledge goal is then passed to an introspective planning component that reasons about possible memory search strategies (Leake 1995c) to guide search for the needed information. This search process generates a memory search plan whose operators may include both operators from an initial set of memory search strategies and *memory search cases* stored after solving previous adaptation problems. If the needed information is found, it is used to apply the selected transformation to the retrieved response plan. If it is not found, the process continues with step 4, manual adaptation.
3. **Plan evaluation:** The adapted response plan is evaluated by a simple evaluator that checks the

compatibility of the current plan with explicit constraints from the response plan. A human user performs backup evaluation. If the new response plan is not acceptable, other adaptations are tried.

4. **Manual adaptation:** If the previous autonomous case adaptation steps fail to generate an acceptable solution, an interface allows the user to guide the adaptation process, selecting a transformation and suggesting memory search paths to consider. During the adaptation, the system records a trace of the adaptation process. The trace is represented in the same form as the traces of system-generated adaptations, so that the system can learn an adaptation case from the interactive episode.
5. **Storage:** When DIAL successfully adapts a response plan, it learns by storing (1) the new *response plan case*, (2) *memory search cases* encapsulating the memory search steps performed during case adaptation, and (3) *adaptation cases,* which encapsulate information about the adaptation problem as a whole—the transformations and memory search cases used when solving the adaptation problem—and its solution.

Thus the system learns not only new response plan cases but also new ways of adapting existing cases to new situations. In addition, new adaptation cases that are learned are used not only to perform new adaptations, but also to estimate adaptation cost during the similarity assessment process for retrieving new cases; learning adaptation cases corresponds to learning new similarity criteria as well.

The Basis of DIAL's Adaptation Learning

DIAL's learning is based on introspective reasoning about requirements for solving adaptation problems. To support reasoning about adaptation problems, a uniform framework is needed for characterizing case adaptation. Following the framework of *adaptation strategies* (Kass 1990), DIAL's rule-based case adaptation treats the case adaptation process as involving two parts: selecting *structural transformations* (e.g., additions, substitutions, and deletions) and performing *memory search* to find the information needed to apply the transformations. Accordingly, two types of case adaptation knowledge are needed: abstract transformations and memory search strategies. It is widely accepted that a small set of transformations is sufficient to characterize a wide range of adaptations (Carbonell 1983; Kolodner 1993), but a large amount of domain-specific reasoning may be required to find the information needed to apply those transformations. The following sections discuss how DIAL's rule-based adaptation process represents the information it needs in the form of knowledge goals, uses the knowledge goals to guide the formation of memory search plans, and packages a trace of its memory search process and other reasoning during adaptation for future use.

Knowledge Goals

Knowledge goals provide explicit descriptions of needed information. DIAL's knowledge goals are satisfied by a planning process for how to carry out memory search. Memory search plans are built from simple primitive memory search operators (e.g., to extract slot values or find abstractions) and traces of successful memory searches satisfying similar previous knowledge goals. Knowledge goals represent information about the type of knowledge needed, about the context of the search, about the reasoning giving rise to the knowledge goal, and about what to do with the knowledge, once found. Thus they reflect the basic principle of goal-driven learning, applied to information search in memory: decisions about the knowledge to acquire and how to acquire it should be based on goal-derived criteria and satisfied by a strategic planning process.

In DIAL, initial knowledge goals are generated to obtain information necessary for a specific adaptation, in response to a problem or inconsistency in applying a retrieved response plan to a current disaster situation. For example, a problem applying the response plan for a flood in Bainbridge, Georgia to a flood in Allakaket, Alaska, is that the Salvation Army provided shelter during the Bainbridge flood, but does not exist in Allakaket. A new relief group local to Allakaket must be found. Consequently, a knowledge goal is generated to find a substitute for the Salvation Army that can provide shelter.

Memory search cases

A memory search case consists of a trace of primitive memory search operators (or previously-stored memory search cases) that were used in a successful previous memory search. Initial memory search cases may be built up by applying "weak methods" of memory search, such as "local search," that are built into the system. (Local search is a common strategy for finding substitutions (Kolodner 1993); it attempts to find concepts that are "near-by" in the system's memory, and progressively widens the search until a suitable substitution is found). Memory search cases may also be built up interactively, by recording traces of a user-guided memory search process.

Retrieved memory search cases provide an initial strategy for finding needed information. Memory search cases are indexed both under the adaptation

cases that have successfully used them, and under the knowledge goals they satisfy.

Adaptation cases

Adaptation cases package the results of a successful adaptation. An adaptation case consists of three parts: indexing information, adaptation information, and evaluation information. The indexing information includes a representation of the type of problem to adapt and information about the response plan for which the adaptation case was generated. This information guides selection of the adaptation cases to use for new adaptation problems. The adaptation information packages both a transformation type (e.g., substitute, add, delete) and the memory search steps used to find the information needed to apply the transformation. Evaluation information records the cost of applying the adaptation.

Stored adaptation cases are organized in memory by the types of problems they address. The vocabulary of problem types is similar in spirit to the problem vocabularies used to guide adaptation in other CBR systems (e.g., Hammond, 1989; Leake, 1992). For example, a role-filler in a candidate response plan may be inappropriate, and a new role-filler needed, because of problems such as:

FILLER-PROBLEM:UNAVAILABLE-FILLER
The role filler specified in the plan is unavailable. For example, a police commissioner may be out of town and unable to be reached in an emergency situation.

FILLER-PROBLEM:ROLE-MISMATCH
The role filler specified in the plan is incompatible with the given role. For example, a mismatch occurs when a plan for dealing with an industrial disaster is applied to a school disaster (whose victims are children rather than workers), and the industrial response plan calls for notifying the victim's union (instead of parents).

FILLER-PROBLEM:FILLER-UNSPECIFIED
The role specification may be incomplete, simply because of missing information that must be filled in. For example, a plan might call for a rescue without specifying who should carry it out.

The Effects of Learning Plan Cases and Adaptation Cases

An important question is the benefit of augmenting the traditional learning strategy of case-based reasoning—case acquisition—with case adaptation learning. To obtain initial indications of the effects of adaptation learning in DIAL, we performed ablation tests of the system with initial test examples. The system's initial memory included nodes for 800 concepts; the initial case library included 3 disaster response plan cases, and the test examples involved performing a total of 26 adaptations to develop response plans for 6 stories.

Stored cases and new stories were based on the Clarinet News Service newswire and the *INvironment* newsletter for air quality consultants. Stored cases involved an earthquake in Los Angeles, an air quality disaster at a manufacturing plant, and a flood in Bainbridge, Georgia. The tests considered only the efficiency of performing an adaptation, but as described later in this paper, efficiency is only one possible dimension for measuring the value of case adaptation learning.

In the baseline condition, the system performed no learning of either cases or adaptations. In addition, it performed all memory search during case adaptation by "local search." The second condition added learning of response plan cases, but no learning of adaptations; this reflected the "standard" configuration of most CBR systems. The third condition included learning of adaptation cases, but not of response plan cases. The fourth condition included learning of both response plan cases and adaptation cases. The fifth and sixth conditions replaced "local search" with other memory search strategies, such as attempting to extract constraints on acceptable role-fillers of a schema and using them to define knowledge goals to be satisfied by a knowledge planning process. The fifth condition involved response plan learning only, and the sixth involved learning of both response plans and adaptation cases. We expected that either learning of response plan cases alone (conditions 2 and 5) or adaptation cases alone (condition 3) would improve performance over no learning (condition 1), and that the performance would be better with both types of learning (conditions 4 and 6) than with either individually.

Because most of the system's adaptation cost comes from memory search, efficiency was estimated by two different criteria reflecting memory search cost: the number of primitive memory operations performed and the number of memory nodes visited. In each case, lower values suggest less effort expended, but the two numbers can vary significantly as multiple operations can be applied to a single memory node, and, conversely, many nodes may be examined but never have operations applied directly to them. In general, variations in the order of presenting the problems may also have an effect on overall performance, but for the sample set changes in problem order did not appear to

	Nodes Visited					Memory Ops				
	Max	Min	Avg	Med	Dev	Max	Min	Avg	Med	Dev
Using "local search" to find needed information										
1. *No learning*	252	8	77	41	41	164	6	45	28	40
2. *Plan learning only*	252	8	55	36	64	164	6	32	22	38
3. *Adaptation learning only*	159	5	45	14	50	86	2	25	10	27
4. *Plan + Adaptation learning*	159	5	38	11	49	86	2	22	6	27
Using multiple strategies to find needed information										
5. *Plan learning*	1270	111	369	217	360	284	72	119	90	65
6. *Plan + Adaptation Learning*	1268	5	147	42	281	238	1	36	4	59

Table 2: Effort expended adapting the five sample cases.

have a significant effect. Table 2 shows the results for a single problem order. The table shows the maximum, minimum, average, median and standard deviations of the number of operations applied when processing the test stories in each condition.

In trials using local search, results were as predicted, with a combination of response plan learning and adaptation learning performing best overall. When multiple search strategies were used as opposed to local search, similar results were achieved, however, both the number of memory nodes visited and the number of operations performed were significantly higher than in tests using only local search. While initially surprising, this result can be explained by the fact that initial selection of the multiple search strategies is largely arbitrary. The dramatic decrease of the median in condition 6 suggests that adaptation learning is resulting in much more effective strategy selection.

The benefit of using multiple search strategies is evident in problems which are very difficult using local search methods alone. For example, adapting a Los Angeles earthquake response plan to generate a response plan for an earthquake in Liwa, Indonesia, requires adapting the means of transportation for relief supplies: The Los Angeles response plan involves the Red Cross sending supplies in by truck, but the roads to Liwa are impassable. In the real episode, the solution was a military airlift. The Red Cross and the military are distant in the system's memory and are each characterized by different constraints, making local search extremely costly, but performing a more strategic search process, characterizing the problem as "lack of access" and searching for actions to overcome that impediment and actors who could carry out those actions, suggests a more direct solution. In the tests, such problems did not arise often, however. When adaptation cases based on both local search and other strategies are saved and reused, average performance is better than for either method individually.

We note that, consistent with predictions, response plan learning did improve performance, as did adaptation learning. Interestingly, adaptation learning alone was somewhat more effective than case learning alone. As was also expected, when no adaptation cases are learned, learning additional response plan cases enables the system to solve new problems with less adaptation effort—more similar cases are available. This is the foundation for the benefits of learning found in most CBR systems. Adding adaptation learning to response plan learning produced a moderate drop in cost when memory search during adaptations was based on local search. There was much greater benefit when response plan learning was combined with adaptation learning using other memory search strategies.

These data are only suggestive; they involve a very small set of examples and the typicality of the examples is unclear. We plan to follow up on these initial data by performing a more controlled analysis of the effects of learning for a larger set of problem examples, and also to examine the potential utility problem (Francis & Ram 1993; Minton 1988) as the number of adaptation cases grows.

Learning similarity from adaptability

Adaptation learning provides the motivation for another type of learning, learning to refine similarity criteria. A central role of similarity judgments in case-based reasoning is to determine which cases to apply to a new situation and how to adapt them to fit new circumstances. As Smyth & Keane (1995) observe, CBR systems often base similarity judgments on semantic similarity, but the real goal of their "similarity assessment" is to determine *adaptability:* how easily an old case can be adapted to fit the requirements of a new situation. If new adaptation strategies are learned, static similarity criteria do not keep pace with new capabilities for performing adaptations. When adaptation learning makes it easier to apply particular cases, those cases should be judged more relevant to a new situation. Thus similarity assessment criteria should change as new adaptation knowledge is acquired.

We have begun to study methods aimed at enabling

DIAL to improve its similarity assessment process by using learned adaptation cases to provide estimates of the cost of adapting particular types of problems. After retrieving an initial set of candidate response plan cases using traditional indexing techniques, DIAL compares their adaptability. For each problem to be repaired by adaptation, DIAL retrieves the adaptation case for the most similar prior problem (using the description of the adaptation problem). If the adaptation case was generated to solve an identical adaptation problem, the solution to that previous adaptation can be reapplied directly, resulting in very low adaptation cost: the cost is simply the cost to perform the needed transformation. If the adaptation case dealt with an adaptation problem that was similar but not identical, the cost of applying it to the new problem is estimated as the cost of the transformation used, plus the cost of the primitive memory search operations used in the prior adaptation. The rationale is based on the principle of derivational analogy: If a previous adaptation for a similar problem had to extract certain features and constraints, and transform them in certain ways to generate an appropriate adaptation, the process for the current situation should follow the same basic steps, even if the specifics of the situation are different.

If no similar adaptation case is found, DIAL uses a crude estimate of the cost: it maintains a record of the average cost (measured in primitive memory search operations) of adapting problems in each problem category, starting from scratch, and estimates the cost of the current problem using that average. By basing similarity assessment directly on the current state of its changing adaptation knowledge, DIAL's similarity assessment process reflects its adaptation knowledge. The aim is to improve the overall CBR process by favoring cases that are likely to be easier to adapt.

Perspective

Motivations for multistrategy learning during case-based reasoning

Our application of multistrategy learning to CBR is motivated by a number of potential benefits. One of these is that learning new similarity criteria and storing and "replaying" adaptations, by derivational analogy, will make it possible both to select better (more easily adaptable) prior cases and to expedite the adaptations that are performed, providing speedup learning.

In domains such as disaster response planning, for which no hard-and-fast rules are available to characterize what constitutes a good plan, an equally important potential benefit is increasing the quality of the solutions generated. Part of the appeal of reasoning from prior cases is to reflect regularities of a situation that may not be explicitly represented in a reasoner's domain theory. Thus storing and replaying successful adaptations may help to generate better adaptations than would be generated by reasoning from scratch. In the examples we have considered so far, the results generated by reusing adaptation cases are reasonable, and much more reliable than the results of, for example, simply selecting candidate substitutions by "local search." However, although we see this as an important potential benefit, the comparative effects on quality remain to be tested.

A final motivation for studying this multistrategy learning during CBR comes from cognitive modeling. Although the previous discussion provides functional arguments for learning to improve case adaptation skills and for adjusting similarity criteria as adaptation knowledge is learned, some psychological studies point to related aspects of human reasoning. Gentner & Toupin (1986), for example, demonstrate a developmental shift in the similarity criteria used by children for analogical reasoning, and show that the shift is manifested in how they adapt stories to apply to new characters. Experiments by Suzuki et al. (1992), studying adults' similarity judgments for the Towers of Hanoi problem, show that novices' judgments about the similarity of problem states can be characterized by the number of shared surface features, but that experts' judgments are best characterized by the goal-relevant criterion of the number of operators required to transform each problem state to the goal state. Chi et al. (1981) note a dramatic difference between the similarity criteria of novice physics problem-solvers, who rely on surface features, and physics experts, who classify problems according to the underlying methods needed to solve them. Finally, Keane (1994) has shown that when selecting analogues for an analogical problem-solving task, subjects favor analogues that are more readily adaptable to the new problem situation.

Relationship to other computer models

A number of previous CBR systems learn by both case acquisition and refining their indexing criteria (e.g., Hammond, 1989; Veloso & Carbonell, 1994). A few include restricted mechanisms for learning limited forms of adaptation knowledge. For example, CHEF (Hammond 1989) bases its adaptations on both a static library of domain-independent plan repair strategies and a library of special-purpose *ingredient critics,* which suggest steps that must be added to any recipe using particular ingredients (e.g., that shrimp should be shelled before being added to a recipe). CHEF uses special-purpose procedures to learn new ingredient

critics when omitted preparation steps cause recipes to fail. However, the learned adaptations can only be reused in very similar situations, while the adaptation cases learned by DIAL can be reused more flexibly because they are derivational traces of the results of a general introspective reasoning mechanism.

Learning to refine similarity criteria has been investigated in Prodigy/Analogy (Veloso & Carbonell 1994). That system's "foot-print" similarity metric focuses consideration on goal-relevant portions of the initial state, in order to retrieve cases that refer to the prior problem situations with the most relevant similarities. Our adaptability-based similarity method focuses on a different issue, estimating the costs of repairing relevant differences that have been found. Our emphasis on adaptation cost is shared by Smyth and Keane (1995) , who have developed a CBR system that ties similarity judgments directly to adaptability, using heuristics coded to recognize the difficulty of performing particular types of adaptations. In their system, adaptation-guided retrieval results in significant improvements in overall problem-solving cost. In their work, however, similarity and adaptation knowledge are static. As our method learns new adaptations, it derives similarity criteria directly from its own experience with adaptation problems, changing both as it acquires adaptation experience.

DIAL's approach to memory search by question transformation is inspired by the memory search process of CYRUS (Kolodner 1984), and is similar to recent research on applying heuristic search to gathering information for argumentation (Rissland, Skalak, & Friedman 1994) and on strategic methods for information retrieval (Baudin, Pell, & Kedar 1994). Neither of these methods, however, learns from the search process. Our use of analogical techniques for internal reasoning is related to Ram and Cox's (1994) theory of *meta explanation patterns*, Kennedy's (1995) *internal analogy*, and Oehlmann's (1995) *metacognitive adaptation*.

Our use of transformational analogy for case-based planning, and derivational analogy for case-based reasoning applied to case adaptation, combines benefits of both learning strategies. Transformational CBR approaches store and adapt a *solution* to a problem, while derivational approaches store and replay a *derivational trace* of the problem-solving steps used to generate a previous solution. For CBR tasks such as disaster response planning, derivations of solutions are not generally available, and planning from scratch is not satisfactory because domain theories are inaccurate and intractable. However, examples of prior solutions are readily available in news stories and casebooks used to train disaster response planners (e.g., Rosenthal et al., 1989). This favors a transformational approach to reusing disaster response plans. On the other hand, derivational approaches can simplify the reapplication of a case to a new situation, and the rationale for the system's choice of particular steps during adaptation of prior cases *is* available. This makes it possible to use derivational analogy for learning about case adaptation and memory search.

Conclusion

We have described ongoing research on multistrategy learning within a case-based reasoning context, focusing on how case-based reasoners can learn to apply cases more effectively, both by learning how to adapt prior cases to new situations and by learning which types of adaptations are difficult to perform. Our approach to learning response plan cases uses transformational analogy; our approach to learning about case adaptation uses derivational analogy, which it applies both to memory search during adaptation and to the adaptation process as a whole. Our approach to learning similarity criteria builds on the adaptation learning process, to consider cases "usefully similar" if they are expected to be easy to adapt, given experience with prior adaptations. Preliminary trials of the effects of adaptation learning on our system are encouraging for decreasing memory search cost, but more thorough tests are needed, both to study how the process "scales up" when large numbers of adaptations are learned and to determine effects on the quality of the response plans generated. Tests are also needed to examine how well current estimates of adaptation cost predict the difficulty of future adaptations. Our model is now being refined in preparation for more extensive tests of the system as a whole and the effects of its multiple forms of learning.

Acknowledgments

This work was supported in part by the National Science Foundation under Grant No. IRI-9409348.

References

Barletta, R. 1994. A hybrid indexing and retrieval strategy for advisory CBR systems built with ReMind. In *Proceedings of the Second European Workshop on Case-Based Reasoning*, 49–58.

Baudin, C.; Pell, B.; and Kedar, S. 1994. Using induction to refine information retrieval strategies. In *Proceedings of the twelfth national conference on artificial intelligence*, 553–559.

Birnbaum, L.; Collins, G.; Brand, M.; Freed, M.; Krulwich, B.; and Pryor, L. 1991. A model-based

approach to the construction of adaptive case-based planning systems. In Bareiss, R., ed., *Proceedings of the DARPA Case-Based Reasoning Workshop*, 215–224. San Mateo: Morgan Kaufmann.

Carbonell, J. 1983. Learning by analogy: Formulating and generalizing plans from past experience. In Michalski, R.; Carbonell, J.; and Mitchell, T., eds., *Machine Learning: An Artificial Intelligence Approach.* Cambridge, MA: Tioga. 137–162.

Carbonell, J. 1986. Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In Michalski, R.; Carbonell, J.; and Mitchell, T., eds., *Machine Learning: An Artificial Intelligence Approach*, volume 2. Los Altos, CA: Morgan Kaufmann. 371–392.

Chi, M.; Feltovich, P.; and Glaser, R. 1981. Categorization and representation of physics problems by experts and novices. *Cognitive Science* 5(2):121–153.

desJardins, M. 1992. Goal-directed learning: A decision-theoretic model for deciding what to learn next. In *Proceedings of the Machine Discovery Workshop, Ninth International Machine Learning Conference.* San Mateo: Morgan Kaufmann.

Francis, A., and Ram, A. 1993. Computational models of the utility problem and their application to a utility analysis of case-based reasoning. In *In Proceedings of the Workshop on Knowledge Compilation and Speed-Up Learning.*

Gentner, D., and Toupin, C. 1986. Systematicity and surface similarity in the development of analogy. *Cognitive Science* 10(3):277–300.

Hammond, K. 1989. *Case-Based Planning: Viewing Planning as a Memory Task.* San Diego: Academic Press.

Hunter, L. 1990. Planning to learn. In *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, 261–268. Cambridge, MA: Cognitive Science Society.

Kass, A. 1990. *Developing Creative Hypotheses by Adapting Explanations.* Ph.D. Dissertation, Yale University. Northwestern University Institute for the Learning Sciences, Technical Report 6.

Keane, M. 1994. Adaptation as a selection constraint on analogical mapping. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society*, 490–495.

Kennedy, A. 1995. Using a domain-independent introspection mechanism to improve memory search. In *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*, 72–78. Stanford, CA: AAAI Press. Technical Report WS-95-05.

Kolodner, J. 1984. *Retrieval and Organizational Strategies in Conceptual Memory.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Kolodner, J. 1991. Improving human decision making through case-based decision aiding. *The AI Magazine* 12(2):52–68.

Kolodner, J. 1993. *Case-Based Reasoning.* San Mateo, CA: Morgan Kaufmann.

Leake, D.; Kinley, A.; and Wilson, D. 1995. Learning to improve case adaptation by introspective reasoning and CBR. In *Proceedings of First International Conference on Case-Based Reasoning*, 229–240. Sesimbra, Portugal: Springer Verlag.

Leake, D. 1992. *Evaluating Explanations: A Content Theory.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Leake, D. 1995a. Adaptive similarity assessment for case-based explanation. *International Journal of Expert Systems* 8(2):165–194.

Leake, D. 1995b. Combining rules and cases to learn case adaptation. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, 84–89.

Leake, D. 1995c. Representing self-knowledge for introspection about memory search. In *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms*, 84–88. Stanford, CA: AAAI Press. Technical Report WS-95-05.

Mark, W.; Simoudis, E.; and Hinkle, D. 1996. Case-based reasoning: Expectations and results. In Leake, D., ed., *Case-Based Reasoning: Experiences, Lessons, and Future Directions.* Menlo Park, CA: AAAI Press. In press.

Michalski, R., and Tecuci, G., eds. 1994. *Machine Learning: A Multistrategy Approach.* San Mateo, CA: Morgan Kaufmann.

Michalski, R. 1994. Inferential theory of learning: Developing foundations for multistrategy learning. In Michalski, R., and Tecuci, G., eds., *Machine Learning: A Multistrategy Approach.* Morgan Kaufmann. chapter 1, 3–61.

Minton, S. 1988. *Learning Search Control Knowledge: An Explanation-Based Approach.* Boston: Kluwer Academic Publishers.

Oehlmann, R. 1995. Metacognitive adaptation: Regulating the plan transformation process. In *Proceed-*

ings of the Fall Symposium on Adaptation of Knowledge for Reuse. AAAI.

Ram, A., and Cox, M. 1994. Introspective reasoning using meta-explanations for multistrategy learning. In Michalski, R., and Tecuci, G., eds., *Machine Learning: A Multistrategy Approach.* Morgan Kaufmann. 349–377.

Ram, A., and Leake, D. 1995. Learning, goals, and learning goals. In Ram, A., and Leake, D., eds., *Goal-Driven Learning.* MIT Press.

Ram, A. 1987. AQUA: Asking questions and understanding answers. In *Proceedings of the Sixth Annual National Conference on Artificial Intelligence*, 312–316. Seattle, WA: Morgan Kaufmann.

Richter, M. 1995. The knowledge contained in similarity measures. Invited talk, the First International Conference on Case-Based Reasoning, Sesimbra, Portugal.

Rissland, E.; Skalak, D.; and Friedman, M. 1994. Heuristic harvesting of information for case-based argument. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 36–43. Seattle, WA: AAAI.

Rosenthal, U.; Charles, M.; and Hart, P., eds. 1989. *Coping with crises: The management of disasters, riots, and terrorism.* Springfield, IL: C.C. Thomas.

Smyth, B., and Keane, M. 1995. Experiments on adaptation-guided retrieval in case-based design. In *Proceedings of First International Conference on Case-Based Reasoning.*

Suzuki, H.; Ohnishi, H.; and Shigermasu, K. 1992. Goal-directed processes in similarity judgment. In *Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society*, 343–348. Bloomington, IN: Lawrence Erlbaum.

Veloso, M., and Carbonell, J. 1994. Case-based reasoning in PRODIGY. In Michalski, R., and Tecuci, G., eds., *Machine Learning: A Multistrategy Approach.* Morgan Kaufmann. chapter 20, 523–548.

Inductive Logic Programming + Stochastic Bias = Polynomial Approximate Learning

Michèle Sebag [1,2] and Céline Rouveirol[2] and Jean-François Puget[3]
(1) LMS, CNRS-URA 317, Ecole Polytechnique, 91128 Palaiseau
(2) LRI, CNRS-URA 410, Université Paris Sud, 91405 Orsay
(3) ILOG, 9 rue de Verdun, 94250 Gentilly
FRANCE
Michele.Sebag@polytechnique.fr, Celine.Rouveirol@lri.fr, puget@ilog.ilog.fr

Abstract

A major difficulty in Inductive Logic Programming (ILP) lies in the size of the hypothesis space, and in the number of possible matchings between a candidate hypothesis and a training example.
This paper investigates the use of a stochastic bias in order to make induction a tractable problem. A goal-directed sampling mechanism of the matching space is implemented. The exhaustive exploration of the matching space is replaced by considering a fixed, user-supplied, number of samples. One thereby constructs a theory which is only approximately consistent, with a polynomial computational complexity in term of the size of the data.
This approach significantly differs from the ILP approaches based on genetic algorithms or genetic programming, where stochastic sampling is directly used to explore the hypothesis space; in our approach, the sampling mechanism is combined to induction instead of replacing it.
Experiments on the mutagenicity problem fully validates the approach in terms of both predictive accuracy and computational cost.

Introduction

The framework of Inductive Logic Programming (Muggleton & De Raedt 1994) allows induction to handle problems having relational descriptions. This very expressive formalism however rises two major questions: that of dealing with numerical values, and that of mastering the computational complexity pertaining to first-order logic.

Handling numbers in ILP has mainly been tackled in two ways: the first one consists in transforming the current ILP problem into an attribute-value problem, as done in the LINUS approach (Lavrac, Dzeroski, & Grobelnick 1991) or via moriological reformulations (Zucker & Ganascia 1994). Another possibility consists in providing the learner with some kind of "numerical knowledge" (e.g. the definition of predicate `less-than`); this knowledge can either be built in the learner, as in *FOIL* (Quinlan 1990), or added to the domain theory, as in *PROGOL* (Muggleton 1995).

Yet another approach is based on the use of *Constraint Logic Programming* (CLP), which subsumes Logic Programming and allows for *interpretation* of structures, notably in domains $\mathbb{R}$ and $\mathbb{N}$ (Jaffar & Lassez 1987). Learner *ICP* (for *Inductive Constraint Programming*) was described in an earlier work (Sebag & Rouveirol 1996); it basically uses constraints to prevent negative examples from matching candidate hypotheses. Since all possible (generally numerous) matchings between a hypothesis and a negative example must be examined, *ICP* therefore fully faces with the combinatorial complexity inherent to First Order based languages.

The complexity issue is addressed in the ILP literature by using either search biases or syntactic biases; e.g., *FOIL* considers one literal at a time (Quinlan 1990); *FOCL* allows a limited amount of look-ahead (Pazzani & Kibler 1992); *GOLEM* and *PROGOL* respectively set restrictions like ij-determinacy or maximal number of literals on the candidate hypotheses (Muggleton & Feng 1990; Muggleton 1995); restrictions on the matching space are also possible (Zucker & Ganascia 1994).
Adjusting these biases requires a precise *a priori* knowledge, which is far from always available.

This is the reason why we preferred a versionspace-like approach, where the only bias is that of the hypothesis language (Mitchell 1982). Precisely, our induction goal is to characterize the *whole* set of hypotheses that are consistent with the training examples and that cover at least one example, denoted *Th*. In this sense, this approach is "bias free"; the only problem is that the explicit characterization of *Th* is intractable, even in a propositional language (Haussler 1988). We therefore developed an approach based on an implicit characterization of *Th*, which is of polynomial complexity in attribute-value languages (Sebag 1996). This approach was extended to ILP (Sebag 1994a; Sebag & Rouveirol 1994) and then CLP (Sebag & Rouveirol 1996) but still suffered from the exponential complexity typical of first order logic. The present paper describes how an approximate characterization of *Th* can be obtained *with polynomial complexity* in a CLP language.

This approach is illustrated on the mutagenesis problem which comes from the field of organic chemistry (King, Srinivasan, & Sternberg 1995). The shortcoming of *ICP* on this problem comes from the number of possible matchings between a hypothesis and an example of molecule, which is exponential in the number of atoms in the molecule (up to 40). This drawback is sidestepped by sampling, rather than exhaustively exploring, the set of such matchings: only a restricted number of samples is considered during induction. These matching samples are produced by a stochastic goal-driven mechanism. The hybrid learner obtained by embedding this sampling mechanism in *ICP*, termed *STILL* for *Stochastic Inductive Logic Learner*, constructs a theory that is only ensured to be approximately discriminant, since it only explores a subset of the matching space. But it does so with a polynomial computational complexity in terms of the size of the training set.

In order for this paper to be self contained, *ICP* is first briefly described. The complexity issue is examined, and we present a goal-driven mechanism that samples the matching space explored by induction. This stochastic sampling mechanism yields tractable algorithms for approximate induction and classification. An experimental validation of *STILL* on the mutagenicity testbed is last presented; our results are compared to those of *FOIL* and *PROGOL*, reported from (Srinivasan, Muggleton, & King 1995).

ICP and Mutagenicity

This section illustrates our approach of induction on the mutagenicity problem which is one well-known testbed in ILP. A detailed description of *ICP*, and a discussion about induction in Constraint Logic Programming can be found in (Sebag & Rouveirol 1996).

Data and language of examples

The mutagenicity problem consists in discriminating organic molecules (nitroaromatic compounds) with high mutagenic activity from other organic molecules. This still open problem is of utmost practical interest, for these compounds occur in car exhaust fumes, and high mutagenicity is considered carcinogenic.

The basic description of molecules, referred to as background knowledge $\mathcal{B}_1$ in (Srinivasan & Muggleton 1995), includes the description of the molecule atoms and the bonds between these atoms. As the description of atoms involves numbers (partial electric charge), "numerical knowledge" (e.g., definition of predicate `less-than`) was added to background knowledge $\mathcal{B}_1$, to form background knowledge $\mathcal{B}_2$. Additional information is given via five attributes measuring the hydrophobicity of the molecule, the energy of the molecule lowest unoccupied molecular orbital, and so on. Background knowledge $\mathcal{B}_3$ stands for $\mathcal{B}_2$ augmented by this non-structural description. There exists a still more sophisticated description of the molecules (background knowledge $\mathcal{B}_4$), that takes into account elementary structural chemical concepts (e.g. methyl or benzene groups); but it will not be considered in this paper.

A molecule a is thus described by a clause, an excerpt of which is:

$$\begin{aligned} tc(a) :- \quad & atom(a, a_1, carbon, 22, -0.138), \ldots, \\ & atom(a, a_{26}, oxygen, 40, -0.388), \ldots \\ & bond(a, a_1, a_2, 7), \ldots, bond(a, a_{24}, a_{26}, 2), \\ & logp(a, 4.23), lumo(a, -1.246), \ldots \end{aligned}$$

where tc stands for the target concept satisfied by a, here *active* or *inactive*.

Literal $atom(a, a_1, carbon, 22, -0.138)$ states that in compound a, atom a_1 is a carbon, of type 22, with partial charge -0.138. Literal $bond(a, a_1, a_2, 7)$ expresses that there exists a (unique) bond between atoms a_1 and a_2 in a, the type of which is 7.

This problem thus typically involves numerical and relational features.

Overview of *ICP*

ICP is a divide-and-conquer algorithm in the line of the *AQ* algorithms (Michalski 1983; Michalski *et al.* 1986): it successively generalizes one example Ex, termed *seed*, against all examples not satisfying the same target concept as Ex, termed *counter-examples* to Ex, noted $Ce_1, \ldots, Ce_n$.

ICP differs from other divide-and-conquer algorithms in two respects. First, most authors (Michalski 1983; Muggleton 1995) restrict themselves to learning one target concept (e.g. the *activity*), while *ICP* learns both the target concept and its negation. This hopefully allows the effects of noisy positive and negative examples to counterbalance each other.

Second, *ICP* considers *all* training examples instead of pruning the examples covered by previous hypotheses. The problem with pruning is that it induces an additional bias (the eventual theory depends on the choice of seeds) while increasing the overall complexity of induction (this will be detailed further).

Another key difference between *ICP* and all other learners, as far as we know, is that *ICP* does involve neither search biases nor syntactic biases: it aims at characterizing the set *Th(Ex)* of all hypotheses consistent with $Ex, Ce_1, \ldots, Ce_n$, i.e. the Version Space discriminating Ex from any Ce_i (Mitchell 1982).

In opposition, *FOIL* (Quinlan 1990), *FOCL* (Pazzani & Kibler 1992) and *PROGOL*, among others, aim at finding "the" best hypothesis covering a training example, according to the more or less greedy optimization of a numerical criterion (quantity of information for *FOIL* and *FOCL*, MDL principle for *PROGOL*). To a lesser extent, *ML-Smart* (Bergadano & Giordana 1990; Botta & Giordana 1993) and *REGAL* (Giordana & Saitta 1993) also look for concise theories.

Formally, *ICP* builds a theory Th which is the disjunction of the versionspaces $Th(Ex)$ for Ex ranging over the training set. $Th(Ex)$ includes all hypotheses that cover Ex and discriminate counter-examples to Ex; it is given by the conjunction of the set of hypotheses $D(Ex, Ce)$ that cover Ex and discriminate Ce, for Ce ranging over the counter-examples to Ex:

Disjunctive Version Space Algorithm

$Th = false.$
For each Ex training example
  $Th(Ex) = True$
  For each Ce counter-example to Ex
    Build $D(Ex, Ce)$ *(see below)*
    $Th(Ex) = Th(Ex) \wedge D(Ex, Ce)$
  $Th = Th \vee Th(Ex).$

Language of hypotheses

The elementary step of *ICP* consists in characterizing the set of hypotheses $D(Ex, Ce)$ that cover the current seed Ex and discriminate a counter-example Ce to the seed.

Consider two examples of (simplified and arbitrary) molecules:

Ex : active(ex) :- *atom(ex, a, oxygen, -3.38),*
atom(ex, b, carbon, 1.24).
Ce : inactive(ce) :- *atom(ce, c, carbon, -2.75),*
atom(ce, d, oxygen, 0.33).

Example Ex is decomposed as $C\theta$, where C stores the structural information of Ex, i.e., that Ex is an *active* molecule having two atoms:

$C : active(X) : -atom(X', Y, Z, T), atom(X'', U, V, W)$

and θ carries all other information in Ex:

$$\theta = \{ \; X/ex,\ X'/ex,\ X''/ex,\ Y/a,\ Z/oxygen,\ T/-3.38,\ U/b,\ V/carbon,\ W/1.24\}$$

This decomposition allows induction to simultaneously explore two search spaces:

- The space of logical clauses generalizing C. Exploring this space is fairly simple since all variables in C are distinct: the only way to generalize C is by dropping literals.
- In what regards θ, we take advantage of the fact that θ, and more generally any substitution on C, are special cases of constraints on C. Variable grounding $\{X/v\}$ amounts to domain constraint $(X = v)$, which is analogous to a selector (Michalski 1983). And variable linking $\{X/Y\}$ amounts to binary constraint $(X = Y)$. *ICP* then explores the set of constraints on C generalizing θ. (See (Jaffar & Lassez 1987) for a comprehensive presentation of CLP).

Constructing $D(Ex, Ce)$ amounts to finding out all pairs (C, ρ), where C generalizes C (i.e. describes a molecule satisfying the same target concept as Ex and including at most the same number of atoms and bonds) and ρ is a constraint on C that generalizes θ; furthermore, C and ρ must be such that $C\rho$ discriminates Ce.

In the general case, discrimination can be based on predicates: if C involves a predicate that does not appear in Ce, C does not generalize Ce. Predicate-based discrimination amounts to simple boolean discrimination: presence/absence of a predicate (Sebag & Rouveirol 1996). It is not further considered in this paper since all molecules, whatever the target concept they satisfy, are described by the same predicates (*atom*, *bond*,..): predicate-based discrimination thus does not apply here.

Constraint-based discrimination takes place when the body of C (or of the current hypothesis) generalizes that of Ce. Then there exists at least one substitution σ matching the body of C with the body of Ce, called *negative substitution*.
For instance, such a substitution σ respectively maps the first and second atoms in C onto the first and second atoms in Ce:

$$\sigma = \{ \; X/ce,\ X'/ce,\ X''/ce,\ Y/c,\ Z/carbon,\ T/-2.75,\ U/d,\ V/oxygen,\ W/0.33\}$$

But if such a σ exists, C is inconsistent: its body generalizes the bodies of both Ex and Ce, which yet satisfy opposite target concepts by definition.

Constraint-based discrimination prevents such inconsistencies by specializing C: it adds "conditions" (that is, constraints) to the body of C such that negative substitution σ does not satisfy these "conditions". For instance, constraint

$$\rho = (V = carbon)$$

is incompatible with σ, since $V.\sigma = oxygen$. By the way, ρ must also generalize the substitution θ derived from Ex, in order for $C\rho$ to still generalize Ex. For instance, constraint

$$\rho' = (V = hydrogen)$$

is also incompatible with σ, but $C\rho'$ does not generalize Ex any more. A formal presentation of constraint entailment and generalization order will be found in (Jaffar & Lassez 1987); roughly, constraint ρ_1 generalizes ρ_2 (equivalently, ρ_2 *entails* ρ_1) iff all substitutions satisfying ρ_2 also satisfy ρ_1.

Note that building constraints that generalize θ and are incompatible with a negative substitution σ amounts to an attribute value discrimination problem. This is particularly clear if we restrict our language of constraints to domain constraints (of the form

$(X = \mathcal{V})$, where $\mathcal{V}$ is a subset of the domain of X). This is also true when binary logical and arithmetic constraints are considered (e.g. $(X \neq Y)$, $(Z < T + 10)$, $(S > U - 20)$), by introducing auxiliary variables (this point is detailed in (Sebag & Rouveirol 1996)). However, binary constraints will not be further considered here, for two reasons. First of all, introducing binary constraints does not significantly modify the complexity of induction (it only affects its polynomial part), which is our primary concern in this paper. Second, unary constraints turned out to be sufficient to reach a good level of predictive accuracy on the mutagenesis problem.

Finally, our language of constraints is restricted to unary constraints of the form $(X = \mathcal{V})$, where

- $\mathcal{V}$ is an interval if X is a real or integer-valued variable;
- $\mathcal{V}$ is a value if X is a nominal variable.

Characterizing $D(Ex, Ce)$

Let us assume first that there exists a single negative substitution σ derived from Ce.

For any variable X, let $\rho_{X,\sigma}$ denote the maximally general constraint on variable X that generalizes θ and is incompatible with σ. Of evidence, $\rho_{X,\sigma}$ exists iff $X.\theta$ and $X.\sigma$ are distinct constants. When it is the case, and when X is an integer or real-valued variable, $\rho_{X,\sigma}$ is $(X = \mathcal{V})$ where $\mathcal{V}$ is the maximum interval including $X.\theta$ and excluding $X.\sigma$. And in case X is a nominal variable, $\rho_{X,\sigma}$ is $(X = X.\theta)$.
In our toy example, $(T =]-\infty, -2.75))$ (or for the sake of simplicity $(T < -2.75)$), is the most general constraint on T that generalizes θ and is incompatible with σ; similarly, $(Z = oxygen)$ is the most general constraint on Z (in our constraint language) that generalizes θ and is incompatible with σ.

Let ρ_σ denote the disjunction of $\rho_{X,\sigma}$ for X ranging over the variables in $\mathcal{C}$ such that $X.\theta \neq X.\sigma$. One easily shows that any constraint generalizing θ discriminates σ iff it entails (is generalized by) ρ_σ. A clause $C\rho$ therefore belongs to $D(Ex, Ce)$ iff two conditions hold: $C\rho$ must generalize Ex and ρ must entail ρ_σ.
If σ were the only negative substitution on $\mathcal{C}$ derived from Ce, any clause $C\rho$ generalizing Ex such that ρ entails the disjunction

$$(Z = oxygen) \vee (T < -2.75) \vee (V = carbon) \vee (W > .33)$$

would discriminate Ce.
But consider the negative substitution σ' mapping the first atom in $\mathcal{C}$ onto the second atom in Ce, and the second atom in $\mathcal{C}$ onto the first atom in Ce:

$$\sigma' = \{ \begin{array}{l} X/ce,\ X'/ce,\ X''/ce, \\ Y/d,\ Z/oxygen,\ T/0.33, \\ U/c,\ V/carbon,\ W/-2.75, \} \end{array}$$

Of evidence, constraint $\rho = (Z = oxygen)$ is quite compatible with σ'; hence $C\rho$ is inconsistent. This shows that a discriminant constraint ρ must be incompatible with *all* negative substitutions derived from Ce, in order for $C\rho$ to be consistent with Ce.

In the general case, let $\Sigma_{Ex,\ Ce}$ be the set of negative substitutions on $\mathcal{C}$ derived from Ce. The previous result extends as (Sebag & Rouveirol 1996):

Proposition 1. $C\rho$ belongs to $D(Ex, Ce)$ iff $C\rho$ generalizes Ex and ρ entails ρ_σ for all σ in $\Sigma_{Ex,\ Ce}$.

To sum up, $D(Ex, Ce)$ is computationally described by $\mathcal{C}$ and the set of constraints $\{\rho_\sigma\ s.t.\ \sigma \in \Sigma_{Ex,\ Ce}\}$. The question of explicitly characterizing $D(Ex, Ce)$ and then the whole theory Th from this computational characterization was addressed in (Sebag & Rouveirol 1996) and will not be considered in this paper. Rather, we focus on using the computational description of $D(Ex, Ce)$ in order to classify further instances of the problem domain.

As a matter of fact, this computational description enables one to check whether an unseen instance E is covered by a hypothesis in $D(Ex, Ce)$, or, for short, *belongs to* $D(Ex, Ce)$. It is shown (Sebag & Rouveirol 1996) that:

Proposition 2. E belongs to $D(Ex, Ce)$ iff E can be expressed as $C\tau$, where C generalizes $\mathcal{C}$ and τ entails ρ_σ for all σ in $\Sigma_{Ex,\ Ce}$.

Classifying further examples

The important point in the above result, is that it allows one to classify unseen instances of the problem domain:

- From Proposition 2, one can compute whether any given instance E is covered by a hypothesis in *Th(Ex)*, or, for short, *belongs to Th(Ex)*: E belongs to *Th(Ex)* iff E belongs to $D(Ex, Ce)$, for all Ce counter-example to Ex.
- Knowing whether E belongs to *Th(Ex)* for all training examples Ex gives means to classify E, via a nearest neighbor-like process. Let E be termed *neighbor* of Ex if E belongs to *Th(Ex)* ; the class of E can thereafter be determined by a majority vote among its neighbors in the training set. (see (Sebag 1996) for a discussion about the advantages of this nearest neighbor-like classification process).

Neighbor (E, Ex) : (E *belongs to* Th(Ex))

```
For each Ce counter-example to Ex
  if NOT Belongs(E, D(Ex,Ce))
    return false
return true
```

Checking whether E belongs to $D(Ex, Ce)$ is computed as follows. Let $\Sigma_{Ex,\ E}$ denote the set of substitutions on $\mathcal{C}$ matching E. Then:

Belongs(E, $D(Ex, Ce)$)

```
For each τ in Σ_{Ex, E}
  If τ entails ρ_σ for all σ in Σ_{Ex, Ce},
    return true.
return false.
```

Simply put, *ICP* rather constructs an oracle than an explicit theory. This oracle achieves the classification of further examples; it is made of theory *Th*, stored as the list of

$$D(Ex_i, Ex_j) = (\mathcal{C}_i, \{\rho_\sigma \ s.t. \ \sigma \in \Sigma_{Ex_i, Ex_j}\})$$

for Ex_i in the training set and Ex_j counter-example to Ex_i; and this theory is interpreted according to Proposition 2. Actually, the classifier constructed by *ICP* consists of *Th* and of the standard nearest neighbor algorithm, calling the above *Neighbor* boolean function.

Complexity

Under the standard assumption that the domain of any variable is explored with a bounded cost, the complexity of building ρ_σ is linear in the number of variables in $\mathcal{C}$. Let $\mathcal{X}$ and $\mathcal{S}$ respectively denote upper-bounds on the number of variables in $\mathcal{C}$ and on the number of substitutions in Σ_{Ex_i, Ex_j}. The characterization of $D(Ex_i, Ex_j)$ is then in $\mathcal{O}(\mathcal{X} \times \mathcal{S})$.

Let N be the number of training examples. Since all $D(Ex_i, Ex_j)$ must be characterized, the complexity of learning in *ICP* is

$$\mathcal{O}(N^2 \times \mathcal{X} \times \mathcal{S})$$

And, since checking whether an instance E belongs to *Th(Ex)* requires to consider all substitutions τ on $\mathcal{C}$ (with $Ex = \mathcal{C}.\theta$) matching E, the number of which is upper-bounded by $\mathcal{S}$, the complexity of classification in *ICP* is

$$\mathcal{O}(N^2 \times \mathcal{X} \times \mathcal{S}^2)$$

In particular, since pruning a training example requires to check whether it is covered by a previous theory *Th(Ex)*, the complexity of learning increases up to $\mathcal{O}(N^3 \times \mathcal{X} \times \mathcal{S}^2)$ if *ICP* follows a standard divide-and-conquer strategy.

The crux of complexity lies in factor $\mathcal{S}$: in the simple case of molecules with two atoms, $\mathcal{S}$ is 2^2 (any atom in $\mathcal{C}$ can match any atom in Ce). This makes the *ICP* algorithm intractable in the mutagenesis problem, where molecules involve up to 40 atoms: $\mathcal{S}$ then amounts to 40^{40}...

Polynomial Approximate Learning

ICP suffers from two major drawbacks: first, it is intractable for medium-sized truly relational problems, as shown above. Second, it basically stems from the Version Space framework, and therefore ill-prepared to deal with noisy and sparse data.

The tractability limitation is first addressed via a stochastic bias: the idea consists in sampling, rather than exhaustively exploring, the set of substitutions $\Sigma_{Ex,Ce}$. We again illustrate the stochastic sampling mechanism on the mutagenesis problem.

Second, two heuristics, taken from the propositional version of *ICP* (Sebag 1996), are used to relax the standard consistency and generality requirements of Version Spaces, and cope with noise and sparseness.

Stochastic Bias

Let us examine the structure of examples in the mutagenesis problem. Note that the semantics of a molecule is not modified by changing the identifiers of the atoms (nominal constants), provided the change is consistent.

These identifiers can thus be arbitrarily set to $1, 2, \ldots, n$, if n denotes the number of atoms in Ex. A negative substitution σ on $\mathcal{C}$ is completely defined by associating each atom i in $\mathcal{C}$ to an atom in Ce, noted $\sigma(i)$ by abuse of notations. The intractability comes from the fact that, if Ce involves n' atoms, the number of such negative substitutions is in $n'^{\,n}$.

Basically, discriminating σ from θ requires to

- discriminate at least one atom i in Ex from atom $\sigma(i)$ in Ce; or
- discriminate at least one bond in Ex, linking atoms i and j, from the bond in Ce linking atoms $\sigma(i)$ and $\sigma(j)$, if such a bond exists.

The more "similar" atoms i in Ex and $\sigma(i)$ in Ce, the more difficult it is to discriminate them, and the more informative the negative substitution σ is: this notion parallels that of near-misses in attribute-value languages. Formally, a partial order can be defined on the substitutions in $\Sigma_{Ex,Ce}$, and it is shown that non-minimal substitutions can soundly be pruned by discriminant induction (Sebag 1994a; Sebag & Rouveirol 1994): this pruning is analogous to the pruning of non near-misses examples in the propositional case (Smith & Rosenbloom 1990; Sebag 1994b). Unfortunately, building the set of such minimal substitutions turns out to be intractable too.

Another possibility is to consider *only one* substitution σ, defined as minimizing some distance to θ, in the line of the structural similarity developed in (Bisson 1992). The "optimal" substitution in $\Sigma_{Ex,Ce}$ would thus minimize the sum of the distances between atom i in Ex and atom $\sigma(i)$ in Ce, plus the sum of the distances between bonds $i-j$ in Ex and bonds $\sigma(i)-\sigma(j)$ (if such a bond exists) in Ce. The distance between any two atoms makes no problem: as noted by (Srinivasan & Muggleton 1995), the description of an atom can be

handled as a single tree-structured feature since the element of an atom commands its atom type and its atom type commands its charge.

However, using such an optimization approach to determine which substitution to consider in $\Sigma_{Ex,Ce}$ rises two problems: first of all, we feel that a single substitution, even optimal, cannot be representative of the whole set $\Sigma_{Ex,Ce}$; second, the optimization routine in itself can be computationally expensive.

Finally, we decided to consider several substitutions, the number of which to be supplied by the user. These substitutions are not purely random: as stated above, substitutions nearer to θ should be preferred. When constructing a substitution σ, one thus tries to associate to any atom i in Ex the atom j in Ce which is most similar to i, provided that j is not already associated to another atom in Ex. A substitution σ constructed this way constitutes a local optimum with respect to the above minimization problem.
Currently, the sampling mechanism of the substitutions in $\Sigma_{Ex,\ Ce}$, where Ex and Ce respectively includes n and n' atoms, is implemented as follows:

Select σ in $\Sigma_{Ex,\ Ce}$

```
while possible
  Select i in {1,...,n} not yet selected
    Select j in {1,...,n'} not yet selected
      such that atom j in Ce is as close as
      possible to atom i in Ex,
      Do σ(i) = j.
```

Index j is deterministically selected depending on i: atom j in Ce has same electric charge as atom i in Ex, if possible; otherwise, it has same atom type; otherwise, it is of same element.

Index i is stochastically selected with uniform probability in $\{1,\ldots,n\}$. This way, any atom i in Ex will in average be associated to a similar atom in Ce, provided the sampling mechanism is run a sufficient number of times.

More precisely, the above stochastic sampling mechanism ensures that a set of samples captures an arbitrarily precise representation of $\Sigma_{Ex,\ Ce}$ with high probability, provided the number of samples allowed is "sufficient". Further work is concerned with formalizing this intuition, as well as improving the selection mechanism via taking into account also the bonds between atoms.

Overview of *STILL*

The *STILL* algorithm combines the general approach of *ICP* and the above sampling mechanism. This stochastic bias is used to make both induction and classification tractable.

Approximate Learning. Consider the building of the set of hypotheses $Th(Ex)$ that cover Ex and are consistent. Instead of exploring the whole sets of substitutions $\Sigma_{Ex,\ Ce}$ for Ce ranging over the counter-examples to Ex, *STILL* only processes η substitutions, where η is a positive integer supplied by the user. To give an order of idea, η was set to 300 in our experiments on the mutagenesis problem.
This way, it constructs a set of hypotheses $Th_\eta(Ex)$ that cover Ex and are only partially ensured to be consistent, since only sampled substitutions are surely discriminated.

Concretely, the set of hypotheses $Th_\eta(Ex)$ is characterized as follows. Let n be the number of counter-examples to Ex. For each counter-example Ce, $\frac{\eta}{n}$ samples of substitutions are selected in $\Sigma_{Ex,\ Ce}$. The computational description of $Th_\eta(Ex)$ consists of clause C and the set $\mathcal{R}$ of discriminant constraints ρ_σ, corresponding to the η sampled substitutions σ derived from all counter-examples. This way, the number of constraints in $\mathcal{R}$ does not depend on whether Ex belongs to the majority or the minority class. Otherwise, the number of constraints in $\mathcal{R}$ would be much higher, and hence $Th_\eta(Ex)$ much more specific for the minority class than for the majority class. This heuristics was adopted for reasons of empirical accuracy: experimentally, this makes a difference in the mutagenesis problem, where examples are typically two active to one inactive.

Characterize $Th_\eta(Ex)$:

```
R = φ.
n = Number of counter-examples to Ex
For Ce counter-example to Ex
  For j = 1...η/n,
    Select σ in Σ_{Ex, Ce},
    Build ρ_σ
    Do R = R ⋃ { ρ_σ }
return (C, R).
```

The disjunction Th_η of theories $Th_\eta(Ex)$ for Ex ranging over the training set, is termed *approximate theory*; the rate of approximation is the number η of allowed samples. Note that Th_η is more general than Th; it tends toward Th as η increases.

Approximate classification. The classification process in *ICP* is based on checking whether the instance E to classify is neighbor of Ex, i.e. belongs to $Th(Ex)$, for all training examples Ex. In order to do so, it still explores the set $\Sigma_{Ex,\ E}$ of substitutions on C (where $Ex = C\theta$), matching E. The size of this set similarly makes classification intractable.

This limitation is addressed in *STILL* via the sampling mechanism too: instead of exhaustively exploring $\Sigma_{Ex,\ E}$, *STILL* only considers a fixed number K of substitutions in this set, where K is a positive inte-

ger supplied by the user. To give an order of idea, K was set to 3 in our experiments on the mutagenesis problem.

The *Neighbor* function is therefore modified as:

Approx_Neighbor (E, Ex) :

$(\mathcal{C}, \mathcal{R})$ = Characterize $Th_\eta(Ex)$
For i = 1...K
  Select τ in $\Sigma_{Ex,E}$
    If τ entails all ρ in $\mathcal{R}$
      return *true*
return *false*

As soon as a substitution τ among K samples of substitutions in $\Sigma_{Ex,E}$ entails all discriminant constraints in the characterization of $Th_\eta(Ex)$, E is considered an approximate neighbor of Ex.

Note the above function corresponds to an "interpretation" of $Th_\eta(Ex)$ that is more specific than $Th_\eta(Ex)$ itself; this overspecificity decreases as K increases.

Parameter K controls the number of trials allowed to get an answer from theory Th_η; metaphorically speaking, K corresponds to the "patience" of the constructed expert.

Coping with noisy and sparse examples

$Th(Ex)$ (which is the theory $Th_\eta(Ex)$ tends toward as η increases) includes consistent hypotheses only, and maximally general consistent hypotheses in particular. No doubt this approach is ill-suited to real-world datasets: when erroneous examples are encountered, strictly consistent hypotheses have few predictive accuracy (Clark & Niblett 1987). And when examples are sparse, maximally general consistent hypotheses are too general: most instances come to be covered by a hypothesis in most $Th_\eta(Ex_i)$, and therefore get unclassified, or classified in the majority class.

These limitations were already encountered in the attribute-value version of *ICP*, and have been addressed by two heuristics (Sebag 1996), which simply extend to first-order logic owing to the computational characterization of the constructed theory.

The first one addresses the presence of noise in the data, by allowing one to relax the consistency requirement. The originality consists in relaxing this requirement and allowing a limited number of inconsistencies *during the classification of an instance rather than during induction.* In opposition, the number of acceptable inconsistencies in *PROGOL* for instance is set before starting induction; and if one wants to modify this bias, s/he must restart induction from scratch.

This is done as follows. By definition, E belongs to $Th(Ex)$ and is considered as neighbor of Ex iff it belongs to $D(Ex, Ce)$ for all Ce counter-example to Ex. This definition is simply relaxed as: E is from now on considered as neighbor of Ex iff it belongs to $D(Ex, Ce)$ for all Ce counter-example to Ex, but at most ε of them, where ε is a positive integer supplied by the user. Of evidence, the greater ε, the more general the interpretation of $Th(Ex)$ is.

The second heuristics addresses the sparseness of the data, by allowing one to increase the specificity of the produced theory. This is done at the level of the discriminant constraints ρ_σ. By construction, ρ_σ is the maximally general constraint that discriminates σ and generalizes θ; it is the disjunction over the variables X in $\mathcal{C}$, of domain constraints $\rho_{X,\sigma}$. A given substitution τ entails ρ_σ iff there exists at least one variable X such that $X.\tau$ satisfies $\rho_{X,\sigma}$.

The specificity of the theory is simply modified by interpreting now ρ_σ as a M-of-N concept: from now on, substitution τ is considered to entail ρ_σ iff there exists at least M variables X such that $X.\tau$ satisfies $\rho_{X,\sigma}$, where M is a positive integer supplied by the user. And the greater M, the more specific the interpretation of $Th(Ex)$ is.

Note that theory Th_η does not depend in any way on the values of parameter ε or M. In particular, *STILL* requires no *a priori* knowledge regarding the rate of noise and representativity of the data. Parameters M and ε can be adjusted from the experimental classification results — but with no need to restart induction. See (Sebag 1996) for a discussion about the advantages of such *a posteriori* biases.

Complexity

As expected, the complexity of *STILL* is much more affordable than that of *ICP*.

Let $\mathcal{X}$ still denote an upper-bound on the number of variables in $\mathcal{C}$. The complexity of building ρ_σ is still linear in $\mathcal{X}$. The cost of selecting σ is quadratic in $\mathcal{X}$ (this is a large over-estimation). Hence, the complexity of learning $Th_\eta(Ex)$ is in $\mathcal{O}(\mathcal{X}^3 \times \eta)$. Finally, the computational complexity of induction in *STILL* is linear in the rate of approximation and in the number of training examples, and cubic in the number of variables in one example:

$$\mathcal{O}(N \times \mathcal{X}^3 \times \eta)$$

In the mutagenicity problem, N is 188, $\mathcal{X}$ is less than 200. The rate of approximation η was set to 300, to be compared with the typical size of a set $\Sigma_{Ex,\,Ce}$, that is 30^{30}.

The complexity of classification is that of learning, increased by factor K (which was set to 3 in our experiments):

$$\mathcal{O}(N \times \mathcal{X}^3 \times \eta \times K)$$

Note that the heuristics designed to cope with noise and sparseness do not modify the computational complexity of classification.

Experimentation

This section presents an experimental validation of our approximate learning and classification scheme on the mutagenicity problem, which is presented in details in (Srinivasan & Muggleton 1995).

The data

The experimentations reported in this section consider the data set composed of 188 compounds, described via the background knowledge $\mathcal{B}_1$ (including the description of atoms and bonds in the molecules), $\mathcal{B}_2$ ($\mathcal{B}_1$ augmented with definitions of numerical inequalities), and $\mathcal{B}_3$ ($\mathcal{B}_2$ augmented with five non structural attributes).

For all experiments that follow, the atoms in relevant background theories are partitioned in 188 ground clauses, each clause describing all information relevant to a given compound.

The reference results obtained by *PROGOL* and *FOIL* on this problem (reported from (Srinivasan & Muggleton 1995) and personal communication from A. Srinivasan), are:

Background knowledge	Accuracy	
	FOIL	*PROGOL*
$\mathcal{B}_1$	60 ± 4	76 ± 3
$\mathcal{B}_2$	81 ± 3	81 ± 3
$\mathcal{B}_3$	83 ± 3	83 ± 3

Table 1: Results of FOIL *and* PROGOL *on the 188-compound problem: Average predictive accuracy on the test set*

Experimental Settings

The parameter η used to ensure the tractability of induction (rate of approximation) is set to 300. The parameter K used to ensure the tractability of classification is set to 3.

Parameter M used to control the specificity of the theory varies from 1 to 10. Parameter ε, used to control the consistency of the theory, varies from 0 (strictly consistent hypotheses only) to 15%. The consistency of a hypothesis is from now on defined in term of *percentage* of inconsistencies, rather than in term of *number* of inconsistencies.

All results are averaged over 15 independent runs, where each run consists in learning from 90% of the 188 compounds (randomly selected such that the ratio of active/inactive compounds in the training set is same as in the global data, i.e. about two to one) and classifying the remaining 10% of the data. This protocol of validation is similar to the ten-fold cross validation used in (Srinivasan & Muggleton 1995); the number of runs is only slightly increased (from 10 to 15), as suggested for stochastic approaches (Kinnear 1994).

STILL is written in C^{++}. The run-time on HP-735 workstations (including the construction of the theory and the classification of the test examples), varies from 60 to 120 seconds. Despite the difference in the language of implementation (Language C for *FOIL* and Prolog for *PROGOL*), this demonstrates that *STILL* is two or three orders of magnitude faster than *FOIL* and *PROGOL* (respectively 5 000 and 117 000 seconds on HP-735 when handling background knowledge $\mathcal{B}_1$).

Experiments with $\mathcal{B}_1$

STILL is first experimented in the context of background knowledge $\mathcal{B}_1$; this means that variables describing the atom type and the partial electric charge of atoms are handled as nominal variables instead of respectively integer and real-valued variables. With those settings, *STILL* can only learn discriminant *instantiations* of those variables (e.g., $(Charge_i = .84)$). Results of these experiments are therefore to be compared to results of *FOIL* and *PROGOL* in the context of background knowledge $\mathcal{B}_1$.

Table 2 shows how the average predictive accuracy on the test set varies with ε and M (label *Accur*). It also gives the standard deviation of the accuracy, as well as the average percentages of unclassified and misclassified examples (label *Unclass* and *Misclass*). Examples happen to be unclassified either when they have no neighbor in the training set, or when the nearest-neighbor process ends up in a tie.

ε	M	$\mathcal{B}_1$			
		Accur.	Unclass.	Misclass.	Time
0	1	79 ± 2	1.19	19.8	69
	2	**84.5** ± 2	3.17	12.3	73
	3	77.8 ± 3	7.94	14.3	77
	4	76.2 ± 3	11.9	11.9	78
	5	68.3 ± 2	21	10.7	79
5	1	80.2 ± 2	0	19.8	64
	2	81 ± 2	0	19	68
	3	82.5 ± 1	1.98	15.5	72
	4	79.4 ± 2	1.59	19	75
	5	80.6 ± 3	2.78	16.7	77
10	1	73 ± 2	0	27	60
	2	79.4 ± 3	0.397	20.2	64
	3	83.3 ± 3	0	16.7	69
	4	78.2 ± 2	0.794	21	73
	5	75.8 ± 2	3.57	20.6	76
15	1	69.8 ± 0.7	0	30.2	59
	2	80.6 ± 2	0.794	18.7	63
	3	81 ± 2	0.397	18.7	68
	4	83.7 ± 2	0.794	15.5	72
	5	76.6 ± 2	0.794	22.6	74

Table 2: STILL, *Average predictive accuracy on the test set, with background knowledge* $\mathcal{B}_1$, $\eta = 300$, $K = 3$

Note that, as M increases, theories $Th_\eta(Ex)$ get more specific, hence any instance has less and less neighbors in the training set. This is shown as more and more

examples shift from *Accur* and *Misclass* to *Unclass* (especially for $\varepsilon = 0$).

Note also that for increasing values of ε, theories $Th_\eta(Ex)$ get more and more general, and the best predictive accuracy moves toward larger values of M: the overspecificity due to large values of M resists the overgenerality due to large values of ε.

The predictive accuracy reached for the best adjustment of parameters ε and M ($\varepsilon = 0$ and $M = 2$) is significantly better than the results achieved by *PROGOL* and *FOIL* for background knowledge $\mathcal{B}_1$. The question of automatically tuning the learning parameters nevertheless remains open.

Of course, a fair comparison would require to see how the predictive accuracy of *PROGOL* similarly varies depending on the maximum number of inconsistencies and the maximum number of literals in the clauses, which are respectively set to 5 and 4 in (Srinivasan & Muggleton 1995).

A tentative explanation of the differences between the performances of *PROGOL* and that of *STILL* on this particular problem is the following. *PROGOL* and *STILL* operate in similar search spaces and use different heuristics to the same end, namely constructing partially complete and partially consistent clauses. This suggests that the better performances of *STILL* are due to its inherent redundancy: *PROGOL* starts from some training examples and constructs the "best" hypothesis covering these examples, while *STILL* considers all training examples and constructs all approximately consistent hypotheses covering these examples. The redundancy of the constructed theory has frequently been viewed as a factor of robustness and reliability of the classifier: see (Gams 1989; Nok & Gascuel 1995) among others.

Experiments with $\mathcal{B}_2$

We then check the added value of using a CLP formalism: variables describing the atom type and electric charge of atoms are from now on handled as integer and real-valued variables. The only difference is that *STILL* can now learn discriminant domain constraints such as ($Charge_i > .143$) or ($AtomType_i < 22$), instead of simple discriminant instantiations ($Charge_i = .84$) as before.

The results obtained here (Table 3) must thus be compared to those of *FOIL* and *PROGOL* in the context of background knowledge $\mathcal{B}_2$.

Unexpectedly, learning constrained clauses only results in a slight improvement of the overall predictive accuracy (from 84.5 to 86.5): allowing to set numerical inequalities on variables $Charge_i$ and $AtomType_i$ makes few difference. In retrospect, $\mathcal{B}_1$ mainly involves structural information and few distinct numerical values : as noted in (Kohavi 1995), the presence of numerical variables does not mean that a problem is basically numerical.

One side effect of allowing numerical inequalities, is that the best predictive accuracy is obtained for higher values of ε and M. This can be explained as follows: an inequality is more often satisfied than an equality. This implies that the theories constructed with constrained clauses containing inequalities are more general than those built with pure Horn clauses. Increasing the value of parameter M allows to resist this overgenerality, while increasing the value of ε aims at keeping a desirable level of generality.

ε	M	$\mathcal{B}_2$			
		Accur.	Unclass.	Misclass.	Time
0	4	81.5 ± 2	0	18.5	100
	5	83.3 ± 2	0	16.7	107
	6	**85.6** ± 2	0.37	14.1	113
	7	70.7 ± 2	0.741	28.5	122
	8	69.6 ± 3	2.59	27.8	128
5	4	73.7 ± 2	0	26.3	89
	5	81.1 ± 2	0.37	18.5	95
	6	81.9 ± 2	0.37	17.8	100
	7	83.7 ± 2	0	16.3	106
	8	76.3 ± 3	0.741	23	115
10	4	68.1 ± 0.6	0	31.9	88
	5	74.8 ± 2	0	25.2	93
	6	77.8 ± 2	0.741	21.5	98
	7	86.3 ± 2	0	13.7	105
	8	80.4 ± 2	0.37	19.3	109
15	5	70.8 ± 1	0.694	28.5	92
	6	77.4 ± 2	0.347	22.2	96
	7	79.9 ± 2	0.694	19.4	101
	8	**86.5** ± 2	0	13.5	108
	9	83.3 ± 2	0.741	15.9	115

Table 3: STILL, *Average predictive accuracy on the test set, with background knowledge* $\mathcal{B}_2$, $\eta = 300$, $K = 3$

Additional experiments show that the predictive accuracy only slightly increases with η: e.g. increasing η from 300 to 700 improves the best predictive accuracy by only one point; in counterpart, the computational cost is nearly twice as much as for $\eta = 300$.

Experiments with $\mathcal{B}_3$

As noted earlier, the relatively disappointing results of *STILL* with background knowledge $\mathcal{B}_2$ may be explained by the little number of distinct numerical constants involved in the description of atoms.

A third set of experiments has thus considered background knowledge $\mathcal{B}_3$, that is $\mathcal{B}_2$ enriched with (truly) numerical attributes.

The best results, obtained for $\varepsilon = 0$ and $M = 6$, are impressive (Table 4); as far as we know, they are significantly better than the results obtained by various learners using a superset of background knowledge $\mathcal{B}_3$, and in particular, by ILP learners having numerical skills (Karalic 1995).

Experiments with $\mathcal{B}_3$ also confirm the trend observed when experimenting with $\mathcal{B}_2$: the optimal accuracy moves toward higher values of M; the produced theory needs some extra specialization when it is allowed to include inequality constraints.

Again, the main question here appears that of adjusting automatically parameters ϵ and M.

ϵ	M	$\mathcal{B}_3$		
		Accur.	Unclass.	Misclass.
0	6	**93.4 ± 1**	0.347	6.25
	7	88.5 ± 2	1.04	10.4
	8	89.9 ± 1	1.74	8.33
	9	88.9 ± 2	3.47	7.64
	10	88.9 ± 2	2.78	8.33
5	6	89.6 ± 2	0	10.4
	7	91 ± 1	0	9.03
	8	83 ± 2	0.694	16.3
	9	83.7 ± 2	0.694	15.6
	10	79.5 ± 2	0.694	19.8
10	6	85.8 ± 3	0	14.2
	7	84.7 ± 1	1.04	14.2
	8	90.3± 2	0.694	9.03
	9	85.8 ± 2	0.347	13.9
	10	77.8 ± 3	1.39	20.8
15	6	85.1 ± 2	0.347	14.6
	7	86.8 ± 2	0.347	12.8
	8	89.2 ± 2	0	10.8
	9	87.5 ± 2	1.04	11.5
	10	83.7 ± 2	0.694	15.6

Table 4: STILL, *Average predictive accuracy on the test set, with background knowledge* $\mathcal{B}_3$, $\eta = 300$, $K = 3$

Conclusion

We have experimentally demonstrated the potentialities of the stochastic approximate learner *STILL* for classification purposes.

The main interest of this work in our sense, is to show how stochastic processes can be engineered to cut into the combinatorial complexity pertaining to ILP. Note that this sampling mechanism is combined with, rather than replaces, induction. This is a strong difference with the genetic side of machine learning and ILP (Kovacic 1994; Wong & Leung 1995)[1]. More precisely, what is sampled here is related to examples rather than to solutions.

Another interest lies in the non-standard use of the Version Space framework: the computational representation of the constructed theory sidesteps the intrinsic combinatorial complexity of Version Spaces. Further, it allows one to relax at no additional cost the consistency and generality requirements, whenever this is required by the defects, noise and sparseness of the data. Last, the degrees of consistency and generality can be tuned with no need to modify the constructed theory, and hence without restarting induction.

The main weakness of our learning approach is that it constructs nothing like an intelligible theory[2]. Further work is concerned with pruning and compacting the inarticulate theory underlying the oracle; the challenge lies in providing a readable version of this theory *having same predictive accuracy*. The key question is that of the long debated trade-off between intelligibility and predictive accuracy.

This approach will also be given a learnability model, be it based on PAC-learnability (Valiant 1984) or U-learnability (Muggleton. 1994). In particular, in the *Probably Approximately Correct* (PAC) framework, parameter η used to sample the substitutions naively corresponds to the probability of getting the desired theory, the approximation of which is given by ϵ.

Acknowledgments. We gratefully thank S. Muggleton, A. Srinivasan and R. King, who formalized, studied and made available the mutagenicity problem: this nice problem was determinant for the orientation of the presented work. The work of the authors has been partially supported by the ESPRIT BRA 6020 Inductive Logic Programming and by the ESPRIT LTR 20237 *ILP*^{2}.

References

Bergadano, F., and Giordana, A. 1990. Guiding induction with domain theories. In Kodratoff, Y., and Michalski, R., eds., *Machine Learning : an artificial intelligence approach*, volume 3. Morgan Kaufmann. 474–492.

Bisson, G. 1992. Learning in FOL with a similarity measure. In *Proceedings of* 10^{th} *AAAI.*

Botta, M., and Giordana, A. 1993. Smart+ : A multi-strategy learning tool. In *Proceedings of IJCAI-93*, 937–943. Morgan Kaufmann.

Clark, P., and Niblett, T. 1987. Induction in noisy domains. In Bratko, I., and Lavrac, N., eds., *Proceedings of EWSL-87*, 11–30. Sigma Press.

Gams, M. 1989. New measurements highlight the importance of redundant knowledge. In Morik, K., ed., *Proceedings of EWSL-89*, 71–80. Pitman, London.

Giordana, A., and Saitta, L. 1993. REGAL: An integrated system for learning relations using genetic algorithms. In *Proceedings of* 2^{nd} *International Workshop on Multistrategy Learning*, 234–249. Harpers Ferry.

Giordana, A.; Saitta, L.; and Zini, F. 1994. Learning disjunctive concepts by means of genetic algorithms.

[1] In most GA-based learners, solutions are sampled and evaluated from their predictive accuracy, i.e. through deduction only. An exception is the system described in (Giordana, Saitta, & Zini 1994), where the selection process is driven by inductive heuristics.

[2] Though the constructed oracle is *definitely not* a black box: for any classified instance, we can extract a set of clauses justifying this classification (Sebag 1995).

In Cohen, W., and Hirsh, H., eds., *Proceedings of ICML-94*, 96–104. Morgan Kaufmann.

Haussler, D. 1988. Quantifying inductive bias : AI learning algorithms and Valiant's learning framework. *Artificial Intelligence* 36:177–221.

Jaffar, J., and Lassez, J. L. 1987. Constraint logic programming. In *Proc. of the fourteenth ACM Symposium on the Principles of Programming Languages*, 111–119.

Kinnear, K. E. 1994. A perspective on GP. In K. E. Kinnear, ed., *Advances in Genetic Programming*. Cambridge, MA: MIT Press. 3–19.

Karalic, A. 1995. *First Order Regression*. Ph.D. Dissertation, Institut Josef Stefan, Ljubljana, Slovenia.

King, R.; Srinivasan, A.; and Sternberg, M. 1995. Relating chemical activity to structure: an examination of ILP successes. *New Gen. Comput.* 13.

Kohavi, R. 1995. The power of decision tables. In Lavrac, N., and Wrobel, S., eds., *Proceedings of ECML-95*, 174–189. Springer-Verlag.

Kovacic, M. 1994. MILP: A stochastic approach to ILP. In Wrobel, S., ed., *Proceedings of ILP-94*.

Lavrac, N.; Dzeroski, S.; and Grobelnick, M. 1991. Learning non recursive definitions of relations with LINUS. In *Proceedings of EWSL'91*.

Michalski, R.; Mozetic, I.; Hong, J.; and Lavrac, N. 1986. The AQ15 inductive learning system: an overview and experiment. In *Proceedings of IMAL*.

Michalski, R. 1983. A theory and methodology of inductive learning. In Michalski, R.; Carbonell, J.; and Mitchell, T., eds., *Machine Learning : an artificial intelligence approach*, volume 1. Morgan Kaufmann.

Mitchell, T. 1982. Generalization as search. *Artificial Intelligence* 18:203–226.

Muggleton, S., and De Raedt, L. 1994. Inductive logic programming: Theory and methods. *Journal of Logic Programming* 19:629–679.

Muggleton, S., and Feng, C. 1990. Efficient induction of logic programs. In *Proceedings of the 1st conference on algorithmic learning theory*. Ohmsha, Tokyo, Japan.

Muggleton., S. 1994. Bayesian inductive logic programming. In Warmuth, M., ed., *Proceedings of COLT-94, ACM Conference on Computational Learning*, 3–11. ACM Press.

Muggleton, S. 1995. Inverse entailment and PROGOL. *New Gen. Comput.* 13:245–286.

Nok, R., and Gascuel, O. 1995. On learning decision committees. In Prieditis, A., and Russell, S., eds., *Proceedings of ICML-95*, 413–420. Morgan Kaufmann.

Pazzani, M., and Kibler, D. 1992. The role of prior knowledge in inductive learning. *Machine Learning* 9:54–97.

Quinlan, J. 1990. Learning logical definition from relations. *Machine Learning* 5:239–266.

Sebag, M., and Rouveirol, C. 1994. Induction of maximally general clauses compatible with integrity constraints. In Wrobel, S., ed., *Proceedings of ILP-94*.

Sebag, M., and Rouveirol, C. 1996. Constraint inductive logic programming. In de Raedt, L., ed., *Advances in ILP*. IOS Press.

Sebag, M. 1994a. A constraint-based induction algorithm in FOL. In Cohen, W., and Hirsh, H., eds., *Proceedings of ICML-94*. M. Morgan Kaufmann.

Sebag, M. 1994b. Using constraints to building version spaces. In Raedt, L. D., and Bergadano, F., eds., *Proceedings of ECML-94*. Springer Verlag.

Sebag, M. 1995. 2^{nd} order understandability of disjunctive version spaces. In *Workshop on Machine Learning and Comprehensibility, IJCAI-95*. Report LRI, Université Paris-Sud.

Sebag, M. 1996. Delaying the choice of bias: A disjunctive version space approach. In Saitta, L., ed., *Proceedings of ICML-96*. Morgan Kaufmann.

Smith, B., and Rosenbloom, P. 1990. Incremental non-backtracking focusing: A polynomially bounded generalization algorithm for version space. In *Proceedings of AAAI-90*, 848–853. Morgan Kaufmann.

Srinivasan, A., and Muggleton, S. 1995. Comparing the use of background knowledge by two ILP systems. In de Raedt, L., ed., *Proceedings of ILP-95*. Katholieke Universiteit Leuven.

Srinivasan, A.; Muggleton, S.; and King, R. 1995. Comparing the use of background knowledge by two ILP systems. In *Technical Report, Oxford, UK*.

Valiant, L. 1984. A theory of the learnable. *Communication of the ACM* 27:1134–1142.

Wong, M., and Leung, K. 1995. Combining genetic programming and inductive logic programming using logic grammars. In Fogel, D. B., ed., *Second IEEE International Conference on Evolutionary Computation*, 733–736. IEEE Press.

Zucker, J.-D., and Ganascia, J.-G. 1994. Selective reformulation of examples in concept learning. In Cohen, W., and Hirsh, H., eds., *Proceedings of ICML-94*, 352–360. Morgan Kaufmann.

Theory Restructuring: Coarse-grained Integration of Strategies for Induction & Maintenance of Knowledge Bases

Edgar Sommer
Artificial Intelligence Research Division (FIT.KI)
GMD — German National Research Center for Information Technology
Schloss Birlinghoven, D–53757 Sankt Augustin, Germany
Fax +49(2241)14-2072, eddi@gmd.de

Abstract

Even after the publication of (Michalski & Tecuci 1994), the debate over what precisely constitutes *multistrategy* learning continues. While some authors have advocated a very fine-grained view of cooperation between relatively small learning operators, (Emde *et al.* 1993) argued that significant practical benefits can be obtained from a more coarse-grained interaction between different learning modules, each of which is complete by itself. I subscribe to this view, but add non-inductive functionality to establish a multistrategy context for the design and maintenance of knowledge bases.

This paper describes (the implementation of) a coarse-grained, multistrategy architecture for theory induction and maintenance, or *theory restructuring*, called RRT. The aim of theory restructuring, and RRT, is to close the gaps between knowledge engineering, knowledge acquisition and machine learning, in order to better support the design, and maintenance over time, of knowledge bases. To this end, a number of subtasks in the KB-maintenance process have been identified, namely

- **induction**: offering access to existing implementations of approaches to machine learning in the hope of automatically generating useful rules from exemplary data,
- **analysis**: offering insight into the status quo of a KB,
- **reorganization**: offering transformations that change the form and structure of a KB, whilst retaining the set of computed answers, and
- **evaluation**: offering a collection of simple, computable criteria that may help in assessing the suitability of different forms of a KB to different tasks.

Implementations of approaches to these tasks have been incorporated in RRT, but can be and are being augmented by new tools as these become available, so I prefer to view the current toolchest as an initial proof-of-concept collection — even though it works quite well already.

This paper gives an overview of the setup, describes the current tools in some detail, shows how third-party tools fit seamlessly into the coarse-grained, loosely coupled architecture, and then reports on empirical results in two non-toy problem domains.

Theory Restructuring: a Multistrategy Perspective on Maintenance of Knowledge Bases

A knowledge base (KB) is a formal model of some problem domain and consists, at the core and very generally, of facts and rules. The facts describe the problem domain, and the task of a knowledge based system (KBS) is to infer new facts on the basis of these given ones and the rules, and/or to answer queries about the validity of a new fact with yes or no.

Practical experience with designing such KBs (Sommer *et al.* 1994) has shown a need for addressing the task of *maintenance*, as well as the tasks addressed in the established fields of Knowledge Acquisition, Representation & Engineering, Machine Learning and Logic Programming. More succinctly, *reorganization and correcting knowledge bases is the most time-consuming phase during KBS development* (Carbonell 1991).

At various points in the life and design cycle of a knowledge base, the rules may be redundant, inconsistent and only partially cover the concepts [1] that represent the inferential goal, and it may be very difficult to gain insight into this status quo. Naturally, which rules are desirable and which are not depends on the application, but even on an abstract level, an analysis of a KB's status quo can be variously motivated:

- "Semantically" motivated: How good are the rules at solving their prime objective, covering the goal concept(s), that is, inferring new instances and/or answering queries? If coverage is not complete, can this failure be pinpointed and characterized? Are some rules redundant, in the basic sense that omitting them has no effect on the set of computable answers? Can this redundancy be characterized? Is the rule set inconsistent and can reasons for inconsistency be pinpointed?
- Pragmatically motivated: Is the model of the real problem domain as simple as it could be? Is it coherent and homogeneous not only in the sense of performance, but in

[1] I use the word *concept* in loose synonymy with the word *predicate* throughout this paper. Likewise, clause and rule are used more or less interchangeably.

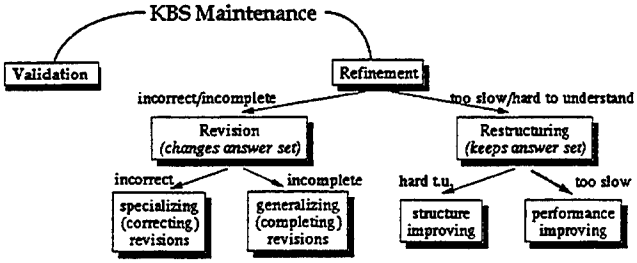


Figure 1: KBS maintenance (adapted from (Wrobel 1996))

the sense that similarities in the "real world" are reflected in similar structures in the model? More specifically, are valid and useful relationships in the real world *explicit* as concepts in the model? Are these concepts put to use consistently and throughout? Is the inferential goal reached by way of a chain of inferences with intermediate concepts (deep theory) or in a single, complex step (flat theory; see figures in the appendix)? The latter may be desirable in view of run-time optimization,[2] while the former may be easier to understand and modify.

- Externally motivated: Is the model adequate, i.e. correct with respect to the aspect of the world it is meant to represent?

From this perspective, the maintenance task can be divided into three interrelated topics: validation, revision, and restructuring (Figure 1). *Validation* is concerned with determining whether the knowledge base, as a formal model of reality, does in fact fulfill the purpose. (Meseguer 1992) gives a concise overview of this issue. *Revision* is concerned with changing the computed answer set of the KBS to fix some problem found in validation, for instance. (Wrobel 1994) gives a detailed treatment. *Restructuring* is concerned with changes to the knowledge base that do not alter the computed answers, but rather improve other criteria, such as understandability or execution time.

Theory maintenance: Insight, clean-up & reorganization

My main interest lies in the investigation and development of methods for maintaining knowledge bases of logical knowledge based systems, i.e. knowledge bases that can be interpreted as restricted first order logical theories.

Services associated with *insight* are meant to provide information on various aspects of the current state of the theory, ranging from simple statistical information and graphical presentations to such criteria as redundancy, utility and coverage of the rules in the theory, explanations of why specific inferences do or do not occur, and computation of criteria intended to help evaluate theory quality.

[2] As the "utility problem" (Minton 1988) in explanation-based learning and subsequent research on "speed-up learning" (Boström 1992; Subramanian & Hunter 1992; Clark & Holte 1992) have shown, this is only a rule of thumb, and often not the case in practice.

Clean-up involves acting on the prior analysis, such as elimination of unnecessary rules in theory and unnecessary conditions in individual rules, fixing non-generative rules, eliminating unused concepts and renaming concepts to better reflect their intended use.

Reorganization concerns the modification of a given theory's inferential structure without changing the answer set. Some examples of such modification are:

- Introduction of new concepts into the theory that allow a more concise re-expression of the rules. Motivation is twofold: finding meaningful and useful new concepts in a theory, and providing a deeper inferential structure of the theory, making it more modular, easier to understand and modify.
- Flattening of inferential structure by replacing intermediate concepts with their definitions where they occur in rules. This is the complement of the previous, and minimizes the number of inferences the KBS must perform at runtime.

Embedding Machine learning & Knowledge Acquisition

Restructuring is embedded in a context that combines Machine Learning, Knowledge Acquisition and Knowledge Engineering. Machine learning is concerned with developing algorithms capable of discovering new relationships — expressed as rules — between concepts in a given KB. In the context of KBS, these activities can also be viewed as forms of automatic analysis of existing knowledge, resulting in novel interpretations of existing data and/or the discovery of new relationships and structures in initially unstructured data (Sommer 1995c). More simply, ML offers an alternative to manual elicitation and formalization of rules from experts in cases where data rather than human expertise is more readily available.

Early work in ML was not concerned with this embedded aspect of learning algorithms, but rather with "pure" algorithms that take input and produce a result (a concept or rule) not necessarily expressed in the same language as the input, and not interpreted in any way by the algorithm itself (one-shot learning). More recent work (Morik *et al.* 1993) is taking ML in the direction of open systems, where learning results may function as input for subsequent learning passes (closed-loop learning), and where a uniform knowledge representation allows knowledge sharing between different components for differing purposes (such as knowledge browser, inference engine, explanation, revision and truth maintenance modules).

Work along these lines — using ML as a provider of rules for a KBS — has shown a need for more explicit and elaborate forms of evaluation and post-processing than either the areas of ML and Knowledge Acquisition have been concerned with. In ML, the main success criterion has been accuracy: what percentage of the given examples are covered by the induced theory, and in some cases, what percentage of probable future examples will be covered? This does not solve the task of combining the result of several algorithms. Other aspects of utility, such as understandabil-

ity, modularity, maintainability have not been in the main focus of ML research.

In Knowledge Acquisition, emphasis has been on manual construction of rules via elicitation, often in the absence of significant numbers of concrete examples. Here, the success criterion is based on experts judgment: does the expert agree with the rules that have been formulated by the knowledge engineer on the basis of the expert's utterances?

Theory restructuring aims at filling the gap between these two

- by allowing for various forms of inspection and evaluation of a KB,
- by offering ways of reorganizing a KB to make it more understandable, maintainable and more modular,
- by offering means of characterizing alternative, empirically equivalent forms of KBs,
- and by offering means of sifting a large number of rules produced by a number of competing ML algorithms (and knowledge engineers) for a most concise set of necessary rules.

RRT: an architecture for multistrategy theory induction & maintenance

This section describes the coarse-grained integration of diverse strategies aimed at offering the knowledge engineer a broad palette of services she may wish to take advantage of at different times in the prolonged process of designing and maintaining a knowledge base. RRT is implemented as one of a number of tools in the MOBAL system (Morik *et al.* 1993; Sommer *et al.* 1996). The following subsections briefly introduce the modules that currently make up the RRT architecture.[3] The selection of modules described here is intended to be representative only — a number of third-party induction algorithms can already be used from with RRT, and we are adding new ones, as well as new forms of restructuring as they become available. A final caveat: the analytical services that round off my perspective on KB maintenance (recall the introduction) are also omitted here, but described in (Sommer 1996b; 1995d; Sommer *et al.* 1996).

LINK: selective, incremental, flexible, "anytime" induction

LINK (Sommer 1994a) is a relational induction algorithm based on the idea of *describing* examples in terms of what is known about them (or, more precisely, about the objects that appear as arguments in the given examples), rather than defining a border between positive and negative examples, and making assumptions about what is not known.

LINK's attempt at being descriptive is accomplished, first, by viewing an incrementally extendable, but comparatively small, excerpt of a given knowledge base as a graph, and using heuristics to select a subgraph and treat it as a hypothesis. Second, a hypothesis is evaluated by computing the number of positive, negative, predicted and uncovered examples, and testing these values against two criteria that are parameters of the system. This allows LINK to accept hypotheses that cover some negative examples or predict some new ones (this method of testing hypotheses is due in large part to (Kietz & Wrobel 1991) and was first implemented in RDT).

Third, since the hypotheses are always created around individual examples, it is a simple matter to achieve a basic but effective type of incrementality: LINK can be told to use only uncovered examples in constructing hypotheses, so if it does find acceptable rules, these are guaranteed to provide a measurable contribution toward the goal of constructing a complete definition of the goal concept.

Outside of the context provided by RRT, LINK can be viewed as a multistrategy induction algorithm, in the sense that it combines aspects of three major approaches to induction:

- In the manner of bottom-up, least general generalization (LGG (Plotkin 1970)) algorithms, LINK generates sets of candidate premise literals by collecting facts about the terms that appear in goal examples — i.e. constructs *fact chains* to be used as starting clauses.
- Rather than using maximal fact chains as starting clauses, however, it treats each fact chain as a reservoir of potential premise literals, using a set of built-in, heuristic criteria to select amongst these candidates, in order to form hypothesis clauses (conjunctions of the selected literals, with the current example playing the role of head or conclusion literal). In other words, the decision about which literals to add to a clause is heuristically guided (although LINK does not climb hills).
- It uses additional, parameterized, structural criteria to select among possible hypotheses in the manner of declarative bias-based systems.

This may serve to illustrate the distinction between fine- and coarse-grained integration of strategies: LINK is an instance of fine-grained integration, where several strategies work symbiotically within one implementation,[4] while RRT is an instance of coarse-grained integration, where several autonomous implementations of strategies cooperate in a very loosely coupled manner — the advantage being that they need not be designed for a multistrategy context. In sum, this combination makes LINK especially suited to the restructuring context I am stipulating in this paper, by allowing it

[3] Since I wish to stress the cumulative effect of these tools in this paper, they are not described in detail. As it happens, all of them are described in (Sommer 1996b), along with the individual references given.

[4] LINK's predecessor INCY (Sommer 1993) took this tight integration one step further by treating the induced rules as rule models and feeding them into RDT as declarative bias. INCY's rules functioned as example rules, in a manner of speaking, which RDT then used for a more elaborate exploration of the hypothesis space. (Lübbe 1995; Kietz & Lübbe 1994) later implemented a much more intricate version of this approach to cooperation between inducers.

- to produce results without relying on complex declarative bias or negative examples,
- to work with sparse data,
- to produce generative rules,
- to produce more characteristic rather than discriminant definitions,
- to work incrementally, and
- to be flexible about which acceptance criteria for hypotheses it applies (e.g., allowing some negatives to be covered, or some new instances to be predicted).

XRA: pragmatic reduction of (sub-) theories

The topic of redundancy is a difficult one, mainly because no general-purpose definition exists. I have adopted the viewpoint defended in (Buntine 1988): "In the final analysis, a rule or some condition in the rule is redundant for a particular KBS if, on its removal, the system would continue to perform as required and still do so after any feasible extensions to the systems knowledge have been made." Nevertheless, most approaches to detecting and eliminating redundancy outside of ML have been of a more theoretical nature, including Buntine's own *generalized subsumption*, an extension of Plotkin's seminal subsumption algorithm (Plotkin 1970). By "theoretical" I mean that rather than analyzing (and taking advantage of) the target KBS's inferential capabilities, these are simulated by some idealized logical inference mechanism, such as SLD-resolution and subsumption. I call this approach **intensional**, because it detects redundancies on the basis of concepts' definitions, i.e. their intensions, as opposed to their *extensions*, the set of their instantiations.

In contrast, the **extensional** approach, which should be familiar to ML researchers since it is often included as a post-processing step in induction algorithms, makes decisions based on the known examples of a clause: GOLEM (Muggleton & Feng 1990), for instance, reduces an acceptable hypothesis by successively dropping literals from the premise and computing the coverage of the shorter clause. If it is still within bounds, the shorter clause replaces the original hypothesis, and this process continues until the shorter clause is deemed over-general. Conceptually similar post-processing functionality can be found in most TDIDT programs, such as the tree pruning offered by C4.5 (Quinlan 1992).

A second, orthogonal dichotomy is that of **clause** vs. **literal** redundancy: deciding whether an entire rule can be dropped without penalty, and deciding whether some condition(s) within a rule can go, turn out to be two different tasks, contrary to what one might expect. This said, XRA can be characterized as a collection of services offering extensional clause and literal reductions for restricted first order theories (no function symbols, a restriction inherited from MOBAL).[5] I believe in offering this functionality *outside* of individual algorithms for reasons which may be obvious if you have followed me this far:

- Not all induction programs perform this service on the clauses they induce.
- More importantly, an individual program is partially blindfolded in a number of ways:
  - It is not, in general, aware of the rules it may have induced in previous passes.
  - It is unaware of bigger picture, consisting of rules induced by other programs, as well as rules entered by hand.

In combination, this has bearing on the longer-term goal, which I stipulate in this paper, of *maintaining* a KB. We cannot expect arbitrary programs to take into account the status quo of the KB when proposing new rules, and certainly very few ML programs go back and re-evaluate the rules they produced last week on the basis of the current, changed state of affairs. So we require the services XRA offers in order to maintain a coherent, reduced version of the KB.

FENDER: reorganization through predicate invention

In a complex theory consisting of many long rules, we may ask if the theory can be re-expressed in a more understandable way, without losing the desired complete coverage. In the concept definition discussed in following results section, consisting of 30 rules with an average length of seven premise literal making reference to ten different concepts, there must be a lot of repetition in the conditions — such recurring conditions could be used to define new concepts, which, "folded"[6] into the originals, would reduce the size of the theory and add structure to the inferences of the target KBS. Moreover, the rules could be made simpler by "hiding" some of the variables that appear in the body, but not in the conclusion (non-head variables).

In (Sommer 1994b; 1995b) the notion of stratifying a given theory was introduced, essentially separating the inductive properties of inverse resolution (Muggleton & Buntine 1988) from its (re-) structuring properties. Following

[5] XRA offers a number of other services aimed at the knowledge engineer wishing to gain insight into this state of affairs: focussed excerpts of the KB in text and graphics windows, performance statistics, distinction of different types of concept instances (input, derived, contradictory, redundantly covered, uncovered), explanations of (un-) coveredness & redundancy, unused concepts, minimum required inputs for the inference of a specified goal (Sommer 1995d; Sommer *et al.* 1996).

[6] Folding and unfolding are basic transformations of logic programs investigated in the field of Logic Programming (not surprisingly), see e.g. (Bruynooghe *et al.* 1994). Unfolding should be familiar to most ML researchers, as it is the formal variant of what is done in explanation-based {learning|generalization}. Folding is the inverse operation, replacing a conjunction of literals in a rule premise with a single literal, that elsewhere in theory, is defined precisely as the consequence of that conjunction. From a Logic Programming perspective, constructive induction systems, as well as FENDER, perform "constructive folding", because the fold literal is being defined on-the-fly, rather than already existing in the theory.

a specific strategy, FENDER performs a number of inter- and intraconstruction steps on a given ruleset, restructuring by introducing new intermediate concepts, *retaining* the set of computed answers of the theory. In other words, FENDER performs the constructive part of constructive induction, leaving the induction part to some other algorithm. So, while the input to a (constructive) induction program consists of facts, FENDER's consists of rules already induced by another module (or entered by the knowledge engineer). The result is a new inferential structure that is deeper and more modular, and possibly easier to understand and maintain. The new intermediate concepts are intensionally defined and put to immediate use; they make implied relationships explicit by exploiting similarities and differences between original clauses of the theory. Figures 6 and 7 show the inferential structure of a flat theory, and corresponding deep theory using concepts invented by FENDER; see also the section on access policies below.

EVAL: criteria for communication about theories

(Sommer 1995a) (Sommer 1996a) motivated and defined some simple, computable criteria for the evaluation of theories. The primary purpose was to look beyond the established ML criteria concerning accuracy, and attempt to capture the more elusive quality of understandability; as such, the proposed criteria are similar in spirit to the intuitions expressed in (Bergadano *et al.* 1988), although quite different in detail. The criteria are summarized in Figure 2, and Figures 3, 4 and 5 make use of some in comparing the theories described in the next section.

Empirical results: Access Policies & Machine Shop Scheduling

Access policy in a telecom network

This subsection gives brief account of RRT being used in concert with the learning algorithms RDT and FOIL, applied to the SPEED domain (Sommer *et al.* 1994). The task is to find rules that correctly explain or derive the known instances of `may-operate(<User>, <Component>, <Operation>`, an instance of which specifies that `<User>` may apply `<Operation>` to `<Component>` in the distributed network of telecommunication routing components (examples below). The available background knowledge concerns various attributes of the employees, companies, switches and operations involved, and relationships between them, such as who owns or rents which components, who works for whom and in which department, etc. There are 26 different concepts in all and 1215 instances of the goal,

Induction We apply two different ML algorithms in the hope of "discovering" a policy in the distribution of permitted accesses, represented by the known instances of `may-operate`. The results of FOIL (Quinlan 1990) on this dataset are not entirely satisfactory: it produces a theory consisting of two rules that cover only 60% of the given instances, one of which is recursive (i.e. makes reference to `may-operate` in defining `may-operate`, which, at least, is not a very intuitive way of implementing an access policy).

Next, we apply RDT (Kietz & Wrobel 1991) to the same data. Here we get a large number of rules (30) that do however cover all instances[7]. One example of the 30 original non-redundant goal-concept rules:

```
  owner(X, Y)
& has-dept(Y, Z)
& manages(Z, X)
& works-in(U, Z)
& operator(U)
& optype(V, threshold-read)
→  may-operate(U, X, V).
```

It is to be understood as follows: "If an employee works in the department that manages a switch, and that department belongs to the company that owns the switch, and the employee has operator status, then the employee may perform operations of type `threshold-read` on that switch".

Analysis The theory induced by RDT provides two important types of insight into the underlying data. First, the rules explain how the known access rights can be derived from the other data (the basic description of the problem domain, which might be sourced from a GIS or DBMS). Second, of the 26 relations present in the data, 16 are not referred to in the body of any rule. This has obvious implications for data analysis and database design: if the target relations represent the goal of the database, i.e. the queries that are to answered by the system, then those relations found irrelevant need not be included in the data. In the present state of the theory induced by RDT, this means information about geographical locations, and about the employee relationship between users and companies (as opposed to departments) is irrelevant for determining who may do what to which component.

Understandability & Reorganization At this point we may ask the question posed in the section on FENDER above: can the current theory, consisting of 30 clauses with average length of seven premise literals, be re-expressed in a more understandable way, without losing the desired complete coverage? FENDER suggests one way of achieving this by introducing four new concepts into the theory. In the following, constructed predicates have been renamed for clarity. Note also that if they are deemed relevant and their definitions have been understood by the user, these new concepts and their user-supplied names make explicit valid relationships between objects of the problem domain; they add structure and useful terminology to the theory on the "knowledge level" as well as on the implementational level.

[7] RDT uses an entirely different approach than FOIL, based on additional declarative bias in the form of rule models, so that this should not be taken as a general indication of the relative merits of the two techniques; see (Sommer *et al.* 1994) and the original sources for details.

The first concept found by FENDER makes a relationship between components, departments and users explicit that is implicit in all of the original rules:

```
    has-dept(C, D)
& works-in(U, D)
& manages(D, S)
→ cu_manages(S, C, U)
```

Rewriting the original rules by using this new concept (folding) does not reduce the number of rules, but makes them simpler, both by shortening them and by eliminating any reference to departments, i.e. eliminating a non-head variable from the top-level rules. The second new concept makes use of the first, and is disjunctively defined:

```
  rented-by(S,C)
& works-for(U,C)
& cu_manages(S,C,U)
→ responsible(S,U)
```

```
  owner(S,C)
& works-for(U,C)
& cu_manages(S,C,U)
→ responsible(S,U)
```

Folding this into the rules reduces their number, since any occurrence of either the one or the other definition body is replaced, so that several pairs of rules may "melt" into one. Note also that the top-level rules are again shortened and any reference to companies is eliminated. The third and fourth concepts introduced, `manager_op(OP)` and `operator_op(OP)`, group the constants found in the original rules according to the context they appear in. This eliminates constants from the top-level rules and further reduces their number, so that finally, the 30 are subsumed by these two:

```
  responsible(S,U)
& manager_op(OP)
& manager(U)
→ may-operate(U,S,OP).
```

```
  responsible(S,U)
& operator_op(OP)
& operator(U)
→ may-operate(U,S,OP).
```

Figures 6 and 7 show the inferential structure of, respectively, the flat theory induced by RDT and corresponding deep theory as reorganized by FENDER.

Evaluation This definition of the goal concept is arguably easier to understand than the original one, as it contains far less rules than the original, and the individual rules are shorter and use fewer variables. A more detailed comparison and discussion of understandability issues is given in (Sommer 1996a); see also the figures in the appendix.

Next to providing insight into the data and making the induced theory more understandable, the concepts proposed during restructuring may also be useful in a database design context (Sommer 1995c).[8] Suppose the original database with 26 relations were an initial rough draft. After induction and restructuring, one might consider replacing those 26 with the 4 found by FENDER, i.e. modify the input format of the database to include only required information in a concise format. Notably, the fact that FENDER was able to fold both `rented-by` and `owner` into a single intermediate concept (`responsible`) means that this distinction need not be made in the future (although it may be required for purposes other than determining access privileges).

[8] Thanks to Luc deRaedt for pointing out this possibility.

Reducing individual rules The 30 rules induced by RDT are not clause redundant (recall the discussion in section on XRA above), in fact no instance is covered by more than one rule — even though RDT does not guarantee such results per se. They may however be literal redundant, and this may serve as brief illustration of extensional reduction's value in practice: As the rules have been induced from facts, no background theory relating concepts to one another is available, so that simple subsumption is the only applicable form of intensional analysis. No rule is subset of another, so intensional analysis offers no form of reducing the ruleset.

Extensional analysis, on the other hand, finds one or more premise literals in each rule that can be dropped without changing the rule's coverage, so that the average number of premise literals is reduced from seven to 5.5, and the size of the theory is reduced from 750 to 615 symbols. Moreover, one concept has been eliminated entirely from the list of required minimum inputs.

Allowing generalizations — accepting reduced versions of the rule that cover more examples than the original, as long as this remains within the confines of the overall set of known examples — has an even more marked effect: the average number of premise literals is reduced from seven to four, and the size of the theory is reduced from 750 to 240 symbols. The list of required minimum inputs is reduced from nine to five. Most significantly, the number of rules has been halved, from 30 to fifteen, because the deletion of literals has rendered pairs of rules equivalent. These results are summarized in Figure 3.

Reducing a set of rules As a brief illustration of the distinction between intensional and extensional reduction in practice, consider a version of SPEED containing a large number of rules from a variety of sources (RDT, GOLEM, LINK, FENDER, user input and a procedure that produces random permutations of rules). The rules are both literal-redundant and, since several algorithms have been applied independently, clause redundant.

Intensional reduction here is able to spot the rules that are random permutations of others, but not, for instance, rules that make use of synonymous concepts, such as those produced by repeated calls to FENDER. Extensional reduction is able to weed out all rules up to the two constructed by FENDER (above), thus attaining the best possible result. These results are summarized in Figure 4.

Relearning In a final set of experiments, we return to FOIL. Using the new concepts formed by FENDER, it is able to produce two rules that correctly cover all instances (they are almost identical to FENDER's reformulations above). Furthermore, in a larger dataset containing more operations and more than 7000 instances of `may-operate`, FOIL is also able to induce a correct theory using the new concepts, while it fails without them. This provides a second indication of the new concepts' utility. Note this could be a general-purpose way of proceeding: first using a excerpt of data together with an exhaustive-search algorithm like RDT, then letting FENDER invent highly informative

(in the information-gain sense) new concepts, and finally applying a faster, heuristically guided algorithm such as FOIL to the larger dataset, including the new concepts. (Wnek 1993) reports on successful applications of a similar strategy, "hypothesis-driven constructive induction", in attribute/value domains.

Machine Shop Scheduling

Third-party work This is work in progress conducted principally by Wolfgang Ewert and colleagues at the Technical University of Chemnitz, Germany (Ewert, Dürr, & Oley 1994; Dürr, Ewert, & Leidhold 1995).

Induction In work towards her Master' thesis (Zöller 1995), Gina Zoeller describes the use of RRT in restructuring a theory governing the sequence in which form elements are processed in the construction of workpieces. In prior steps, definitions were induced by RDT for three concepts stating whether elements are to be processed in direct succession, simultaneously (an important concept in scheduling, as it means the elements must mounted on the vise together) or in general succession (allowing other elements to be handled in between).

Reduction XRA was able to significantly reduce the size of these (sub)theories, e.g. from 183 to 14 and 267 to 16 rules in different versions of the domain (Zöller 1995, p. 50). Dropping redundant literals from the remaining rules reduced their average premise length from 3 literals to 2.

Reorganization FENDER was applied to these reduced theories and was able to suggest several intermediate concepts reflecting recurring conditions in the original rules. As some of these where disjunctively defined (and deemed useful by the domain expert), this led to further reductions in the number of rules needed. It also "discovered" a concept describing situations in which several pairs of elements are to be processed in one production step.

In summary, RRT was judged useful both in (Zöller 1995) and when applied to previous work in the machine shop scheduling domain (Oley 1994).

Continuing work The machine shop scheduling problem looks to be a very promising application for the coarse-grained integration of strategies in RRT. Although the work is very much in progress at the time of this writing, I'd like to list some the results attained more recently:

Incremental induction In a domain with some 2500 instances of a binary ordering relation specifying which form element should by processed before which in the construction of a complex machine tool, LINK was able to induce a partial definition, covering approx. 90% with no false classifications, but some predictions we are not sure about at this time — i.e. the theory may be slightly over-general in practice. This definition initially consisted of well over 300 clauses, and the induction time was considerable: all in all, over 24 hours processor time on a Sparc IPX was necessary. LINK "built-in incrementality" came in very handy here, as we were forced to abort several learning passes — when we restarted, LINK was able to take into account the clauses induced in previous passes, and since it can be made to use only uncovered instances as seeds for the construction of new hypotheses, the definition grew in a truly useful incremental manner.

Wolfgang Ewert had previously applied FOIL, GOLEM and RDT to the same domain. Even with hand-crafted negative examples (with hundreds of form elements, a blind closed-world assumption would produce prohibitive megabytes of negative examples) neither of the first two were able to produce good results — note that LINK does not need negative examples in general, and did not use any here. RDT did produce some promising results, but only with the help of some hand-crafted rule models (RDT's primary form of declarative bias, higher order constructs in which predicate symbols are variables and must be instantiated with predicate names from the KB at hand) produced by Wolfgang Ewert in the course of a year of trail and error, whereas LINK produced results immediately, from the "bare facts" only.

Reduction Next, the 300-plus clause definition was reduced down to approximately 100 clauses using XRA's extensional clause and literal reduction facilities. RRT offers a complementary module implementing some intensional reductions (Plotkin and Buntine in the section on XRA above), but this offers little help in practice: in the presence of representative examples, the "redundancies" discovered intensionally are always a subset of the extensional ones. There is a basic tradeoff between the two — extensional analysis may determine "too many" redundancies, which may become invalidated over time in unfavorable cases, while intensional analysis may discover too few, that, however, are guaranteed to be valid across monotonic growth of the KB. This is due to an *intensional completeness assumption* that all valid relationships between concepts in a KB be explicitly note in the form of rules — without such rules, redundancy can only be detected extensionally. This dependency is especially relevant when rules are learned from facts only: since the concepts are defined by enumeration only, intensional analysis has nothing to work with (except the simplest subsumption relation between the induced clauses themselves) (Sommer 1996b, Ch. 3).

Accordingly, only a handful of rules were found to be intensionally redundant here, but a number of rules with relatively low coverage were made unnecessary by rules LINK induced in one of its subsequent passes (obviously, this is *a posteriori* analysis, and another indication of the power of extensional reductions in the task of consolidating the results of several passes and/or several induction algorithms). Also, some of the premise literals were found to be extensionally redundant (dropping them had no adverse effect on rules' coverage), so the average length of rules was also reduced, along with the number of rules.

Reorganization Finally, the reduced theory was fed into FENDER, which came up with about ten new intermediate concepts. Folded into the reduced theory, they reduced the size by another significant factor. Wolfgang Ewert is currently trying to talk the local engineering experts in Chemnitz into judging these intermediate concepts for real-world relevance. Since real-world expertise is a well-known bottle-neck in our game, we are also rerunning the experiments with FOIL, GOLEM and LINK, to see if the new concepts provide additional help in the induction process, as they did in the SPEED domain above.

Conclusion

I hope to have illustrated the benefits of adopting my perspective on KB maintenance, that of *theory restructuring*. By offering the services of induction, analysis, evaluation and reorganization under a coherent roof, the overall functionality of the resulting multistrategy system exceeds the net profit of using the tools individually, in a number of ways:

- Induction can be used as a form of data analysis for unfamiliar domains, both in the obvious vein of proposing new or alternate definitions for goal concepts, and in the less obvious one of offering insights into relevance and irrelevance of specific concepts.
- Incremental induction can be used to complete incomplete definitions, and update the theory as new cases become known.
- Predicate invention on the basis of induced and expert-elicited theories
  - offers automatic delivery of novel perspectives on the current KB by producing alternatives that are more concise, modular, arguably easier to understand, make recurring conditions explicit, and yet are empirically equivalent to the original;
  - adds further insight into the possibilities of designing more concise representations or database schemas;
  - may even provide added help for subsequent induction (recall "Relearning" in the section on access policy above).
- Stand–alone reduction, as opposed to post-processing steps provided by individual inducers, can produce a concise, coherent theory from large collections of rules from a variety of sources, and can detect redundancies that occur over time, and between competing sub-theories, which individual induces are blind to.
- Simple, computable evaluation criteria allow comparison of alternative, empirically equivalent theories, and may allow cooperation and competition between inducers working on the same set of tasks.

The loose coupling of diverse strategies exemplified by LINK (induction), XRA (reduction), FENDER (reorganization through predicate invention) and EVAL (evaluation) put the power of introspection, incremental reduction, modularity and dynamic restructuring at the knowledge engineer's fingertips.

Current Work Although I am sceptical about the ultimate goals of so-call "hard AI", promising experiences with the present system have motivated us to look into the possibilities of adding more autonomy to the services provided by RRT. We are currently working on a process envelope to RRT that would result in a more autonomous, agent-like system able to accept high-level goal specifications and perform corresponding experiments on its own. The central idea is a simplified stock market metaphor, in which brokers mediate between an arbitrary number of autonomous agent processes that either compete on one and the same subtask or cooperate by addressing different aspects of a larger conglomerate task. The brokers would communicate in a language based on the criteria implemented in EVAL,[9] coordinating their clients' (e.g. FOIL, GOLEM, LINK, XRA, FENDER) induction and maintenance activities via a traditional blackboard scheme, the advantage being that these clients need not be designed for a distributed setting — from their point of view, they are performing precisely the services they were designed for.

While RRT represents a coarse-grained integration of strategies, this new system would be a more finely integrated multistrategy system from the user's point of view. In RRT, the functionality of the tools is put at her fingertips, but she retains control. In the new system, her intuitions about which tool might yield interesting results in the next step would have to be replaced by explicitly implemented control strategies based on the current status quo in the theory, as reflected in the numbers computed by EVAL. In the vein of empowering rather than automating (Winograd & Flores 1986), it seems to me one cannot hope to replace domain experts' intuitions with hard-coded strategies, although one may hope to compensate with computational brute force, in the manner of chess playing systems. However, while in chess success is determined by whether or not the program wins, when relying on a KBS in practice, we will be more apt to understand and trust a theory whose evolution we ourselves have guided than one issued from a black box.

Acknowledgments This work was partially funded by ESPRIT project "Inductive Logic Programming" (ILP). I thank Werner Emde, Stefan Wrobel, Dietrich Wettscherek, Jörg-Uwe Kietz, and the other members of the ML groups at GMD and Dortmund University for valuable discussions. Thanks is due to Ross Quinlan and colleagues (FOIL5) and Stephen Muggleton (GOLEM) for allowing their programs to be incorporated in MOBAL, which is available free of charge and liability for academic purposes via `http://nathan.gmd.de/projects/ml/home.html`.

[9] Thanks to Donato Malerba of Bari University, Italy, for suggesting something like this.

Criterion	Rationale
$\|R_{goal}\|$	# rules about goal concept; less is more
$Ratio = \frac{\|Pos_{goal}\|}{\|R_{goal}\|}$	relation between # instances & # rules; more is better
$Av.Coverage = \frac{\sum_{i=1}^{n} \|cov(r_i)\|}{\|R_{goal}\|}$	av. # instances covered by one rule; more is better
$RedundancyIndex = 1 - \frac{Ratio}{Av.Coverage}$	normalized ratio between the two above; a value > 0 indicates instances are covered by more than one rule
$Size(R_{goal})$	# symbols required to represent theory
$Compression = \frac{Size(R_{goal})}{Size(Pos_{goal})}$	rel. between above & # symbols required to represent the instances; a value > 1 indicates remembering the instances is "cheaper" than remembering the theory
$Inference\ Depth$	max. # of inference steps required to compute an answer;
$Min.\ Req.\ Inputs$	# concepts used in rules of the theory; restructuring tranformations such as un/folding & reduction influence this; rules may reference concepts "unnecessarily"
$Av.\ Length\ of\ clauses$	av. # of premise literals in the rules of the theory; a rough approximation of the complexity of indiviual rules
$Av.\ \#Vars\ per\ clause$	# variables in rules' premises; another rough approximation of the complexity of indiviual rules
$Av.\ \#NonHead\ Vars\ per\ clause$	# variables appearing in premise, but not in conclusion; these "clutter" rules unnecessarily & might be avoided by better representation design (intermediate concepts that "suppress" non-head variables)
$Av.\ \#Constants\ per\ clause$	constants in rules might be considered an indication of over-specific rules and/or imperfect representation design (could be avoided with intermediate concepts that "suppress" constants)

Figure 2: Summary of EVAL's theory evaluation criteria (Sommer 1996a)

	Original		Reduce		Allow Generalizations
#Rules	30	⇝	30	⇝	15
$Inst/RulesRatio$	40.5	⇝	40.5	⇝	81
$TheorySize$	750	⇝	615	⇝	240
$Compression$	0.154	⇝	0.126	⇝	0.049
$Av.Length$	7	⇝	5.5	⇝	4
$Av.Vars$	5	⇝	5	⇝	4
$Av.NonHead$	2	⇝	2	⇝	1
$Req.InputConcepts$	9	⇝	8	⇝	5

Figure 3: Reducing individual rules (Literal redundancy). See "Reducing individual rules" in the section on access policies.

	Original		Intensional		Extensional
#Rules	260	⇝	109	⇝	2
$Inst/RulesRatio$	4.67	⇝	11.14	⇝	607.5
$Av.Coverage$	60.75	⇝	86.2	⇝	607.5
$RedundancyIndex$	0.92	⇝	0.87	⇝	0.0
$TheorySize$	5605	⇝	1844	⇝	22
$Compression$	1.153	⇝	0.379	⇝	0.004
$Av.Length$	5.76	⇝	4.1	⇝	3
$Av.Vars$	4.67	⇝	4.23	⇝	3
$Av.NonHead$	1.67	⇝	1.23	⇝	0
$Av.Constants$	0.98	⇝	0.96	⇝	0
$Req.InputConcepts$	9	⇝	9	⇝	5

Figure 4: Intensional and extensional reduction (Clause redundancy). See "Reducing a set of rules" in the section on access policies.

Evaluation Criteria	**Original**		FENDER
$\|R_{goal}\|$ (# rules about goal concept)	30	⇝	2 (20)
$Ratio = \frac{\|Pos_{goal}\|}{\|R_{goal}\|}$	40.5	⇝	607.5
$Av.Coverage = \frac{\sum_{i=1}^{n} \|cov(r_i)\|}{\|R_{goal}\|}$	40.5	⇝	607.5
$RedundancyIndex = 1 - \frac{Ratio}{Av.Coverage}$	0.0	⇝	0.0
$Syntactic\ Size\ of\ R_{goal} : Size(R_{goal})$	750	⇝	22 (151)
$Compression = \frac{Size(R_{goal})}{Size(Pos_{goal})}$	0.154	⇝	0.004 (0.031)
$Inference\ Depth$	1	⇝	3
$Av.\ Length\ of\ clauses$	7.0	⇝	3 (1.50)
$Av.\ \#Vars\ per\ clause$	5.0	⇝	3 (1.55)
$Av.\ \#NonHead\ Vars\ per\ clause$	2.0	⇝	0 (0.15)
$Av.\ \#Constants\ per\ clause$	1.0	⇝	0 (0.75)

Figure 5: Comparison of original (induced by RDT) and restructured (FENDER) versions of SPEED with 1215 example instances of `may-operate`. The criteria are motivated and defined in (Sommer 1996a). Briefly, however:

- Values for the *required theory* are in parentheses: in a deep theory using intermediate concepts in inferences, it makes sense to investigate the values both for the *top-level* theory (the set of rules defining the goal concept) and the *required theory*, which is the union of the former and the set of rules defining the intermediate concepts. In a flat theory, the two are identical, since there are no intermediate concepts.
- The syntactic size of formulae is determined by counting the number of symbols (as in (Muggleton & Buntine 1988), for example.
- $cov(r_i)$ is the set of instances covered by rule (clause) $r_i \in R_{goal}$. Via $Av.Coverage$, this is used in $RedundancyIndex$, which represents as simple approximation of a given (sub-) theory's redundancy. The closer this value is to 1, the more "redundant" the theory is, in the following sense: ideally, each instance should be covered by only one rule in the theory; if this is the case, then $Ratio = AvCov$ and consequently $RedundancyIndex = 0$. The more instances are multiply covered, the larger $Av.Coverage$'s value, while $Ratio$'s remains constant, so that $Red \rightarrow 1$. Note that in the example above, the original ruleset is not redundant, so the restructured one isn't either.

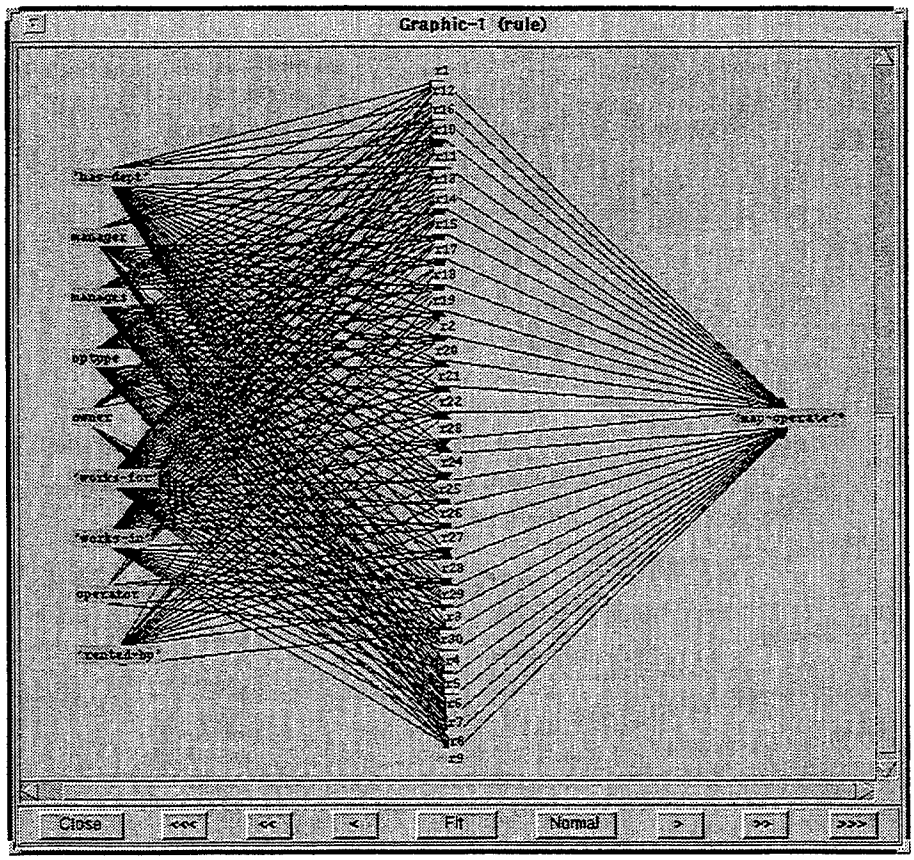


Figure 6: Inferential structure of (flat) SPEED theory induced by RDT, consisting of 30 rules (see also Figure 5). Left to right: **input concepts** used in rule premises → top-level **rule layer** → **goal concept**.

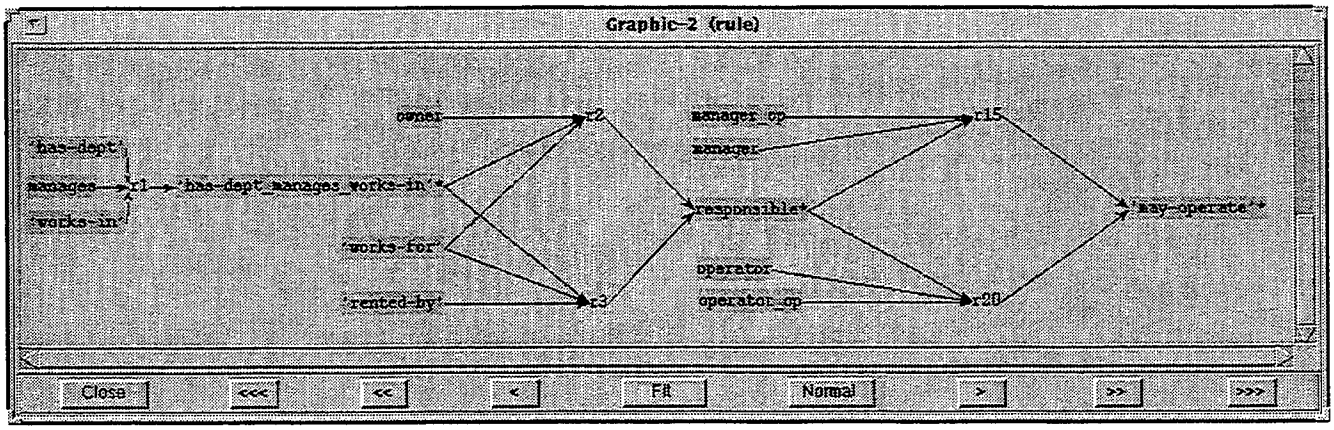


Figure 7: Inferential structure after reorganization by FENDER (inference depth 3, see also Figure 5). Left to right: input concepts → intermediate-level 1 rules → intermediate concepts 1 → intermediate-level rules 2 → intermediate concepts 2 → top-level rules → goal concept)

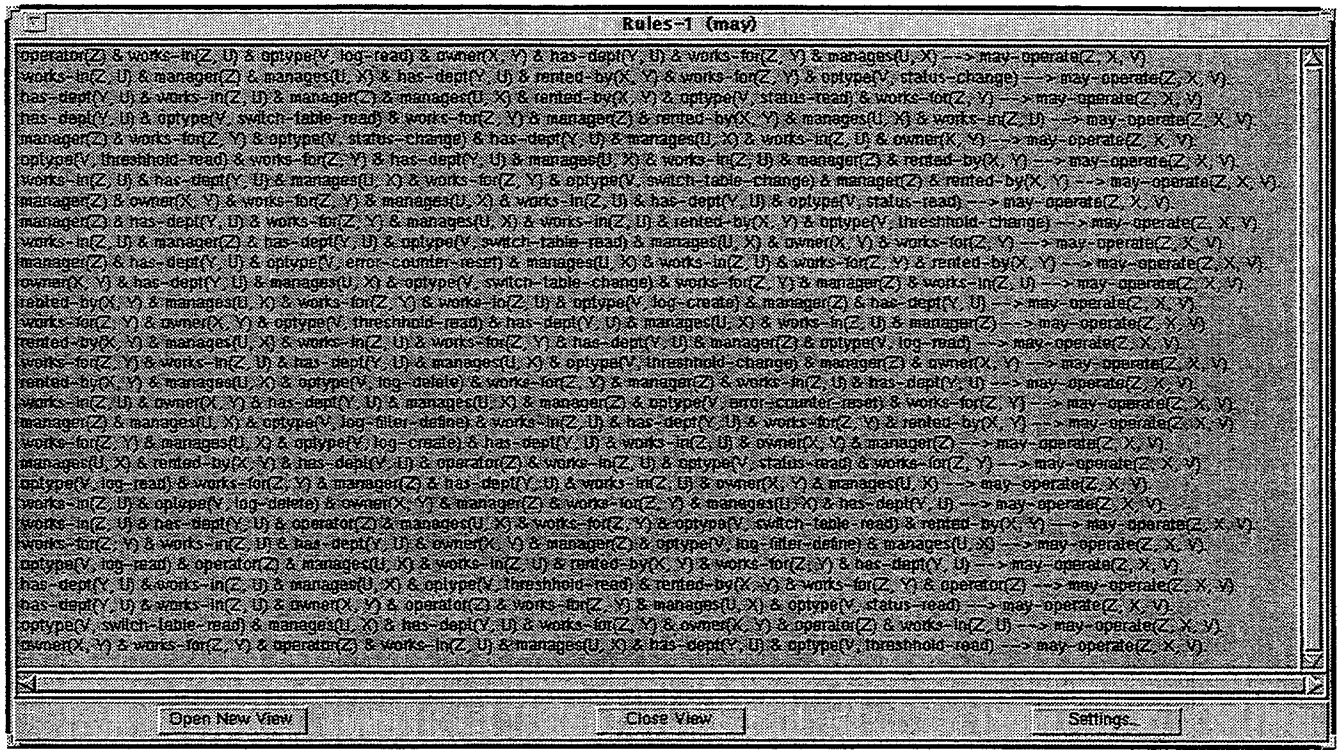


Figure 8: Flat SPEED theory induced by RDT. The corresponding reorganized version consists of only two rules plus definitions of the concepts invented by FENDER (see "Understandability & Reorganization" in the section on access policies).

References

Bergadano, F.; Matwin, S.; Michalski, R.; and Zhang, J. 1988. Measuring quality of concept descriptions. In Sleeman, D., ed., *Third European Working Session on Learning, Glasgow*. London: Pitman Publishing. Congressional Quarterly Almanac, Volume XLII, 1986.

Boström, H. 1992. Eliminating redundancy in explanation-based learning. In Sleeman, D., and Edwards, P., eds., *ML92 Proc. 9th Intl. Conf. on Machine Learning*, 37–42. Morgan Kaufman.

Bruynooghe, M.; Debray, S.; Hermeneglio, M.; and Maher, M., eds. 1994. *Journal of Logic Programming: Special Issue Ten Years of Logic Programming*, volume 19/20. New York: Elsevier Science.

Buntine, W. 1988. Generalized subsumption and its applications to induction and redundancy. *Artificial Intelligence* 36:149–176.

Carbonell, J. G. 1991. Scaling up kbs via machine learning. In Kodratoff, Y., ed., *Proc. Fifth European Working Session on Learning (EWSL-91)*. Springer.

Clark, P., and Holte, R. 1992. Lazy partial evaluation: An integration of explanation-based generalization and partial evaluation. In *Proc. Ninth Intern. Workshop on Machine Learning*. Morgan Kaufman.

Dürr, H.; Ewert, W.; and Leidhold, F. 1995. Machine learning in generative process planning. In Morik, K., and Herrmann, J., eds., *Proceedings FGML-95 (German ML Workshop)*, number 580 in FB Infortmatik Research Reports.

Emde, W.; Kietz, J. U.; Sommer, E.; and Wrobel, S. 1993. Cooperation between internal and external learning modules in mobal: different facets of multistrategy learning. In Saitta, L., ed., *Proc. of the First MLNet Workshop on Multi-Strategy Learning*. Available via WWW `http://nathan.gmd.de/projects/-ml/lit/mlpublist.html`.

Ewert, W.; Dürr, H.; and Oley, G. 1994. Learning in first order logic in a mechanical engineering planning domain. Unpublished.

Kietz, J.-U., and Lübbe, M. 1994. An efficient subsumption algorithm for inductive logic programming. In *Proc. Eleventh International Conference on Machine Learning (ML-94)*.

Kietz, J.-U., and Wrobel, S. 1991. Controlling the complexity of learning in logic through syntactic and task-oriented models. In Muggleton, S., ed., *Proc. 1st Int. Workshop on ILP*, 107 – 126. Also in S.Muggleton (ed.), Inductive Logic Programming, Academic Press, 1992.

Lübbe, M. 1995. Datengesteuertes lernen von syntaktischen einschränkungen des hypothesenraumes für modellbasiertes lernen. Master's thesis, Fachbereich Informatik der Universität Dortmund, Dortmund, Germany. (in German; available as Tech. Report LS-8/15).

Meseguer, P. 1992. Towards a conceptual framework for expert system validation. *AI Communications* 5(3):119–135.

Michalski, R., and Tecuci, G., eds. 1994. *Machine Learning: A Multistrategy Approach, Vol. IV*. San Mateo, CA: Morgan Kaufmann.

Minton, S. 1988. Quantitative results concerning the utility of explanation-based learning. In *Proc. 7th National Conference on Artificial Intelligence*, 564–569. Morgan Kaufman.

Morik, K.; Wrobel, S.; Kietz, J.-U.; and Emde, W. 1993. *Knowledge Acquisition and Machine Learning*. London: Academic Press.

Muggleton, S., and Buntine, W. 1988. Machine invention of first-order predicates by inverting resolution. In *Proc. Fifth Intern. Conf. on Machine Learning*. San Mateo, CA: Morgan Kaufman.

Muggleton, S., and Feng, C. 1990. Efficient induction of logic programs. In *Proc. First Conf. on Algorithmic Learning Theory*. Tokyo: Ohmsha Publishers.

Oley, G. 1994. Akquisistion von strategie- und planungswissen mit dem system mobal. Master's thesis, Fakultät Informatik, Technische Universität Chemnitz. (in German).

Plotkin, G. D. 1970. A note on inductive generalization. In Meltzer, B., and Michie, D., eds., *Machine Intelligence*, volume 5. American Elsevier. chapter 8, 153–163.

Quinlan, J. R. 1990. Learning logical definitions from relations. *Machine Learning* 5(3):239 – 266.

Quinlan, J. R. 1992. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Sommer, E.; Morik, K.; Andre, J.; and Uszynski, M. 1994. What on-line learning can do for knowledge acquisition. *Knowledge Acquisition* 6:435–460.

Sommer, E.; Emde, W.; Kietz, J.-U.; and Wrobel, S. 1996. Mobal 42 user guide ((always) draft). Arbeitspapiere der gmd, GMD. Available via WWW `http://nathan.gmd.de/-projects/ml/lit/mlpublist.html`.

Sommer, E. 1993. Cooperation of data-driven and model-based induction methods for relational learning. In Michalski, R., and Tecuci, G., eds., *Second International Workshop on Multistrategy Learning*, 180–187. Available via WWW `http://nathan.-gmd.de/projects/ml/lit/mlpublist.html`.

Sommer, E. 1994a. Learning relations without closing the world. In *Proc. of the European Conference on Machine Learning (ECML-94)*. Berlin: Springer-Verlag.

Sommer, E. 1994b. Rulebase stratification: an approach to theory restructuring. In *Proc. 4th Intl. Workshop on Inductive Logic Programming (ILP-94)*. Available via WWW `http://nathan.-gmd.de/projects/ml/lit/mlpublist.html`.

Sommer, E. 1995a. An approach to quantifying the quality of induced theories. In Nedellec, C., ed., *Proc. IJCAI Workshop on Machine Learning and Comprehensibility*. Available via WWW `http://nathan.gmd.de/projects/-ml/lit/mlpublist.html`.

Sommer, E. 1995b. Fender: An approach to theory restructuring. In Wrobel, S., and Lavrac, N., eds., *Proc. of the European Conference on Machine Learning (ECML-95)*, volume 912 of *Lecture Notes in Artificial Intelligence*. Berlin: Springer-Verlag.

Sommer, E. 1995c. Induction, evaluation, restructuring: Data analysis as a machine learning loop. In Lasker, G. E., ed., *Proc. of the Conference on Intelligent Data Analysis (IDA-95)*.

Sommer, E. 1995d. Mobal's theory restructuring tool rrt. Technical report, ESPRIT Project ILP (6020). ILP Deliverable GMD 2.2.

Sommer, E. 1996a. An approach to measuring theory quality. In Shadbolt, N., ed., *Proc. European Workshop on Knowledge Acquisition (EKAW-96)*, Lecture Notes in Artificial Intelligence. Berlin: Springer-Verlag.

Sommer, E. 1996b. *Theory Restructuring*. NN. (submitted).

Subramanian, D., and Hunter, S. 1992. Measuring utility and the design of provably good ebl algorithms. In *Proc. Ninth Intern. Workshop on Machine Learning*. Morgan Kaufman.

Winograd, T., and Flores, F. 1986. *Understanding computers and cognition: A new foundation of design*. Norwood, NJ: Ablex.

Wnek, J. 1993. *Hypothesis-driven Constructive Induction.* Ph.D. Dissertation, George Mason University, Fairfax VA.

Wrobel, S. 1994. *Concept Formation and Knowledge Revision.* Dordrecht, Netherlands: Kluwer Academic Publishers.

Wrobel, S. 1996. First order theory refinement. In Raedt, L. D., ed., *Advances in ILP.* Amsterdam: IOS Press. ftp://ftp.gmd.de/ml-archive/GMD/papers/ML73.ps.gz.

Zöller, G. 1995. Strukturierung and systematisierung einer wissensbasis f r das lernen von planungswissen mit dem system mobal. Master's thesis, Fakultät Informatik, Technische Universität Chemnitz. (in German).

Decision Combination based on the Characterisation of Predictive Accuracy

Kai Ming Ting
Department of Computer Science
The University of Waikato, New Zealand
E-mail: kaiming@cs.waikato.ac.nz

Abstract

In this paper, we first explore an intrinsic problem that exists in the theories induced by learning algorithms. Regardless of the selected algorithm, search methodology and hypothesis representation by which the theory is induced, one would expect the theory to make better predictions in some regions of the description space than others. We term the fact that an induced theory will have some regions of relatively poor performance *the problem of locally low predictive accuracy.*

Having characterised the problem of locally low predictive accuracy in Instance-Based and Naive Bayesian classifiers, we propose to counter this problem using a composite learner that incorporates both classifiers. The strategy is to select an estimated better performing classifier to do the final prediction *during* classification. Empirical results from fifteen real-world domains show that the strategy is capable of partially overcoming the problem of locally low predictive accuracy and at the same time improving the overall performance of its constituent classifiers in most of the domains studied. The composite learner is also found to outperform three methods of stacked generalisation which include the cross-validation method in most of the experimental domains studied. We provide explanations of why the proposed composite learner performs better than stacked generalisation, and discern a condition under which the composite learner performs better than the better of its constituent classifiers.

Introduction and Motivation

One important measure of performance used in the machine learning community for learning algorithms is estimated *accuracy* on unseen cases. This measure indicates the estimated overall performance of a learned theory over the whole population from which the training dataset used to induce the theory is generated. However, no one would assume that every prediction made by a theory would have the same probability of being correct (i.e., proportional to its overall estimated accuracy). Regardless of the selected algorithm, search methodology and hypothesis representation by which the theory is induced, one would expect the theory to make better predictions in some regions of the description space than others. Holte, Acker and Porter (1989) were the first to address this intrinsic problem in learning systems that describe the induced theory as a disjunction of conjunctions of conditions. They use the coverage of each rule (or disjunct) to characterise this intrinsic problem. They observe that rules that cover a small number of instances often entail high error rates; thus they name the phenomenon *the problem of small disjuncts.* Several other refined definitions of small disjuncts have been proposed by Ting (1994a) and have been found to work well in overcoming relatively poor performance regions in decision trees by replacing small disjuncts with instance-based methods. We would anticipate the existence of a similar problem in Instance-Based Learning (IBL) algorithms (Aha, Kibler & Albert 1991) and Naive Bayesian classifiers or Naive Bayes (Cestnik 1990). Because both of these algorithms employ different representations from decision trees and rules, measures other than disjuncts are needed to characterise this intrinsic problem in these representations. This consideration has prompted us to rename the problem so that it can be brought to a more general perspective. We term the fact that an induced theory would have some regions of relatively poor performance **the problem of locally low predictive accuracy**.

Very little research has been carried out to gain insight into this problem that is intrinsic to all learning algorithms. The difficulties in this research are (i) how to characterise the various regions of differing predictive accuracies, and (ii) how to estimate predictive accuracy in these regions. The general strategy of our research is to probe into the underlying characteristics of the learning methods to look for an answer. We use the term – *the characterisation of predictive accuracy* (hereafter shortened to *the characterisation*) to mean the use of a measure in an induced theory as an indicator for its predictive accuracy. To address the second difficulty, we estimate the predictive accuracy of the characterisation using a cross-validation method.

Once we can characterise predictive accuracy of each prediction of the induced theories, we can overcome the

problem of locally low predictive accuracy by replacing low performance regions in one theory (from one algorithm) with those of other theory (from another algorithm) which have higher predictive accuracy. The proposed decision combination approach dynamically selects a learning algorithm in each prediction *during* classification.

We will describe how predictive accuracy of each prediction in a IBL algorithm and a Naive Bayesian classifier can be characterised in the next section. Then, we show how the characterisation can be used to overcome the problem of locally low predictive accuracy by working cooperatively between the IBL algorithm and the Naive Bayes in a composite learner framework. The last two sections describe some related work and summarise the findings.

Characterising Predictive Accuracy in IBL and Naive Bayes

We investigate two intuitive methods of characterising the accuracy of each prediction in this section. For Naive Bayes, the natural method is *posterior probability* – predictions made with high *posterior probability* would be more likely to be correct than those made with low *posterior probability*. This characterisation agrees with the Bayesian approach in decision making.

We explore an intuitive method for characterising the accuracy of each prediction in a IBL algorithm, namely *typicality*. This characterisation has its root in cognitive psychology (Rosch & Mervis 1975). In our setting, the *typicality* of an instance (Y) is defined as (Zhang 1992; Ting 1994b):

$$Typicality(Y) = \frac{(\sum_{i=1}^{n} Euclid(X_i, Y))/n}{(\sum_{j=1}^{p} Euclid(X_j, Y))/p} \qquad (1)$$

where the numerator denotes the *inter-concept distance* of an instance (Y) which is defined as its average (Euclidean) distance to instances of different classes (n), and the denominator denotes the *intra-concept distance* of an instance which is defined as its average distance to other instances of the same class (p).

Thus, an atypical instance would have a low value of the *typicality* measure and vice versa. In the framework of IBL, we would expect a prediction made with a typical nearest neighbour is more likely to be correct than one made with an atypical nearest neighbour.

The two algorithms IB1-MVDM* and NB* (Ting 1994c; 1996) are used to test the above characterisations. IB1-MVDM* is a variant of IB1 (Aha, Kibler & Albert 1991) that incorporates the modified value-difference metric (Cost & Salzberg 1993) and NB* is an implementation of the Naive Bayes (Cestnik 1990) algorithm. Both algorithms include a method (Fayyad & Irani 1993) for discretising continuous-valued attributes in the preprocessing (indicated by "*"). This preprocessing improved the performance of the two algorithms (IB1-MVDM and NB) in most of the continuous-valued attribute domains studied by Ting (1994c; 1996). We use the nearest neighbour for making prediction in IB1-MVDM* and the default settings are as used in IB1[1] in all experiments. No parameter settings are required for NB*.

Specifically, we will conduct experiments to test the following hypotheses:

Hypothesis 1: *the higher the posterior probability of a prediction, the higher its predictive accuracy* in NB*.

Hypothesis 2: *the more atypical a nearest neighbour used in a prediction, the lower its predictive accuracy* in IB1-MVDM*.

The next section describes a method to estimate predictive accuracy of the characterisation which will be used to test the hypotheses in the subsequent section.

Estimating Predictive Accuracy of the Characterisation

We use a *cross-validation* approach to estimate predictive accuracy of the characterisation. The individual test results from the v-fold cross-validation are first sorted according to the values of the characterisation (i.e., *posterior probability*, or *typicality*), and then they are used to produce a binned graph that relates the average value of the characterisation to its binned predictive accuracy for each class.

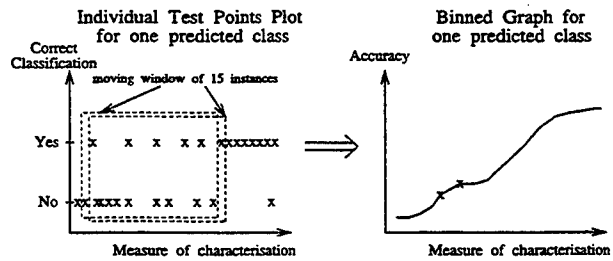


Figure 1: Transforming individual cross-validation test points to a binned graph for one predicted class

Figure 1 depicts the process of transforming the cross-validation test results of individual instances (i.e., correct or incorrect classifications) to a binned graph for one predicted class. Each bin contains a fixed number of instances (half of the total number for each class or fifteen[2], whichever is bigger). Each point in the graph is produced from a "moving window", i.e., the next bin is obtained by dropping the leftmost

[1] IB1 stores all training instances and uses maximum differences for attributes that have missing values, and computes Euclidean distance between any two instances.

[2] This number is about 10% of one of the small size datasets. It is rather ad hoc, but it should not be too big to accommodate small datasets or too small to avoid graph fluctuation due to minor changes in the bin.

Table 1: Details of experimental domains

Domain	*#Inst.*	*#Classes*	*#Attr & Type*
bcw	699	2	9C
diabetes	768	2	8C
waveform21	300	3	21C
waveform40	300	3	40C
heart	303	2	13C
glass	214	6	9C
hypothyroid	3163	2	18B+7C
hepatitis	155	2	13B+6C
automobile	205	6	4B+6N+15C
echo	131	2	1B+6C
horse	368	2	3B+12N+7C
soybean	683	19	16B+19N
led7	200	10	7B
led24	200	10	24B
splice	3177	3	60N

B: Binary, N: Nominal, C: Continuous.

instance and adding an instance adjacent to the rightmost instance of the current bin[3]. When the bin size is bigger than fifteen, the end points of the graph are extended by reducing the bin size one instance at a time until the bin size reaches fifteen. This is to ensure all graphs produced have the same bin size of fifteen at the end points. Note that we use the binned graphs for classification rather than function approximation problems.

Three-fold cross-validation is used in all experiments instead of ten-fold cross-validation (Breiman et al 1984; Schaffer 1993), because it is faster and provides comparable results when used to combine the two algorithms.

Experiments

We conduct the experiments in fifteen well-known domains obtained from the UCI repository of machine learning databases (Murphy & Aha 1994). They are the breast cancer Wisconsin (bcw), pima diabetes, waveform21, waveform40, Cleveland heart disease (heart), glass, hypothyroid, hepatitis, automobile, echocardiogram (echo), horse colic, soybean, splice junction, led7 and led24. The characteristics of the experimental domains are given in Table 1. The example binned graphs of two domains are shown in Figures 2 and 3, which are produced from one trial using 90% of the dataset in each domain. Separate graphs are produced for each class in each domain for the two methods of characterisation, i.e., typicality for IB1-MVDM* and probability for NB*.

We also use a fairly robust (sometimes termed "distribution-free") test of significance, i.e., a Wilcoxon ranksum test (see e.g., Howell (1982)) on the raw data (e.g., the left diagram depicted in Figure 1) to see whether the measure of characterisation is related to trends of correct classification. For each individual predicted class, we rank the individual test points (from

[3]When there are ties, they are resolved randomly.

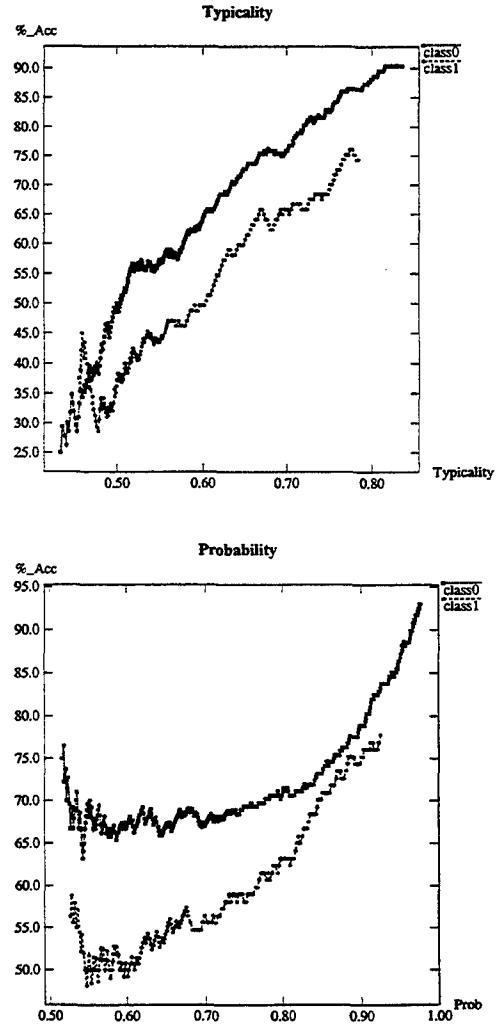


Figure 2: Binned graphs in the diabetes domain

three-fold cross-validation) by the measure of characterisation, e.g., the lowest value of typicality (or probability) is ranked 1. The statistic, W_s is the sum of ranks of all test points that have correct classifications. The distribution of W_s approximates a normal distribution. We can then calculate z from the mean and the standard error of a normal distribution, where $z = (W_s - mean)/(standard\ error)$. We consider a significance level of 90% ($|z| < 1.28$) with a one-tailed test. The results for the test are tabulated in Table 2. A positive value for z indicates that the classifications are more likely to be correct with higher values of typicality/probability. A negative value for z indicates a reverse trend. Table 2 shows the results of the test. The insignificant trends are indicated as boldface z values.

We summarise the findings as follows.

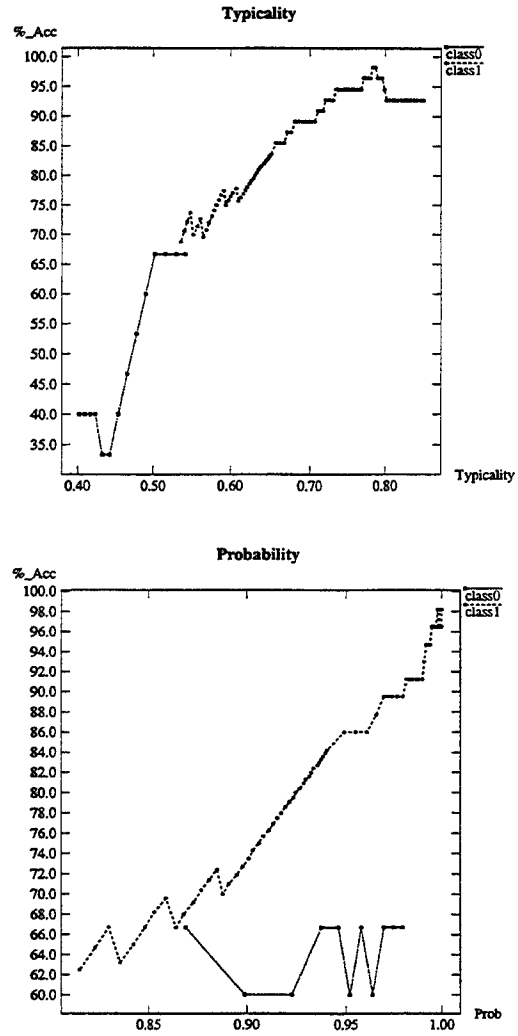


Figure 3: Binned graphs in the hepatitis domain

For **Hypothesis 1**, all classes in ten domains are found to conform to the hypothesis that the prediction made with high *posterior probability* is more likely to be correct. However, the trend seem to be the opposite in one class in the led7 domain. Insignificant trends appear in one class in each of the hepatitis and echocardiogram domains, and in three, four and five classes in the automobile, led7 and led24 domains respectively.

For **Hypothesis 2**, all classes in eight domains provide consistent evidence to the hypothesis that the prediction made with a *typical* nearest neighbour (i.e., with high value of typicality) is more likely to be correct. However, a reverse trend is observed in one class of the automobile domain. Insignificant trends appear in one class in each of the waveform21, waveform40 and echocardiogram domains, and in three classes in each of the glass, automobile, soybean and led24 domains, and in four classes in

Table 2: The results of a Wilcoxon rank-sum test for trends of correct classification with respect to the measure of characterisation depicted in Figure 1

		Typicality			*Probability*		
	PC	n_1	n_2	z	n_1	n_2	z
bcw	0	399	6	4.03	396	3	2.96
	1	214	10	1.35	217	13	2.26
diab	0	355	102	8.22	369	87	6.68
	1	142	92	5.16	157	78	3.98
wave21	0	83	38	2.95	61	8	2.59
	1	45	17	**-1.22**	60	23	4.72
	2	73	14	1.40	89	29	5.78
wave40	0	75	25	2.08	52	6	1.97
	1	53	21	**-0.91**	59	27	2.56
	2	78	18	3.05	93	33	6.22
heart	0	116	27	2.12	127	27	3.68
	1	99	30	6.48	99	19	4.13
glass	0	53	20	**0.80**	56	29	2.50
	1	50	17	2.85	47	21	3.80
	2	4	6	**1.07**	1	3	1.34
	3	5	3	1.34	1	0	–
	4	5	2	**0.00**	5	2	1.94
	5	22	5	2.87	23	4	2.73
hypo	0	111	33	4.68	123	27	4.56
	1	2673	29	7.35	2679	17	6.17
hepa	0	15	13	2.28	17	8	**0.12**
	1	98	13	1.87	103	11	3.19
auto	0	16	9	**0.40**	14	21	1.95
	1	44	10	-1.56	36	18	**-1.01**
	2	41	19	1.85	31	13	**0.22**
	3	17	3	**0.26**	11	15	**0.54**
	4	21	3	**-0.83**	20	5	1.43
echo	0	64	24	2.76	68	21	2.64
	1	13	16	1.45	16	12	**-0.05**
horse	0	180	31	2.30	177	32	1.54
	1	89	31	4.39	88	34	4.63
soyb	0	17	0	–	17	1	1.64
	1	17	0	–	17	1	1.64
	2	19	0	–	19	14	4.66
	3	78	0	–	60	1	1.70
	7	79	6	1.89	69	10	4.39
	8	19	1	1.65	19	2	1.32
	11	39	0	–	39	5	3.61
	12	18	4	**0.17**	15	4	2.10
	13	70	15	**0.74**	80	26	3.99
	14	65	12	4.65	58	4	3.09
	18	8	4	**0.51**	8	0	–
led7	0	15	5	**1.27**	14	5	1.85
	1	14	4	2.23	14	2	2.06
	2	14	9	3.78	15	8	2.97
	3	13	11	**0.32**	12	10	**-0.07**
	4	8	4	2.55	9	3	**0.83**
	5	8	3	**-0.61**	10	2	-1.93
	6	19	10	1.74	18	12	2.37
	7	12	4	1.70	13	4	**1.13**
	8	7	7	**0.06**	5	9	1.53
	9	12	1	1.60	12	3	**0.00**

PC : predicted class.
n_1/n_2: number of correct/incorrect classifications.
Boldface indicates $|z| < 1.28$ (i.e., an insignificant trend).

Table 2 (continues)

		Typicality			*Probability*		
	PC	n_1	n_2	z	n_1	n_2	z
led24	0	10	6	1.52	7	8	1.62
	1	12	3	2.17	14	5	1.39
	2	15	2	**-0.45**	14	5	1.85
	3	9	10	2.20	8	8	**0.00**
	4	7	3	**-0.11**	5	1	**-0.88**
	5	7	1	**1.09**	2	3	**1.15**
	6	19	13	1.78	20	16	1.78
	7	9	9	2.52	8	3	**1.22**
	8	8	14	2.25	8	15	**-1.16**
	9	13	10	2.48	12	18	1.86
splice	0	641	60	8.69	643	33	7.60
	1	631	88	8.57	644	46	9.17
	2	1400	39	9.02	1441	52	10.42

the led7 domain.

The overall result is that most classes in the fifteen domains provide consistent evidence to the hypotheses. The *typicality* and the *posterior probability* have been shown to be a satisfactory characterisation of predictive accuracy in IB1-MVDM* and NB*, respectively.

Discussion

There are a number of reasons for these methods of characterisation to show insignificant trends in some classes in some domains. First, when the underlying concept is hard to learn for these learning algorithms we do not expect any methods of characterisation to provide a good accuracy predictor. Put it in another way, the models or assumptions of these learning algorithms could be all wrong in these cases. An example is the parity concept, where both of these algorithms are known to have difficulty learning it. The second reason could be attributed to the nature of the characterisation itself. *Typicality* measures the global property of instances of a category. In some cases, local properties in the description space might also be important in classification; exceptions or rare cases and the parity problem are examples where *typicality* would fail to characterise. Another important reason is the estimation method used to generate the graphs. Both the bin size and the "moving window" method have an effect on the shape of the graphs. Sparse data in some classes in some domains further compounds the problem.

Note that using these methods of characterisation, one does not explicitly know the exact regions in the description space where the predictive accuracy is high or low, as in the case of small disjuncts in decision trees or rules. But this is the result of the underlying representation rather than the methods of characterisation. The decision boundaries of IBL and Naive Bayes are intrinsically implicit in their representations. One does not know the exact decision boundaries unless one takes extra effort to draw them out (e.g., into a Voronoi diagram (Watson 1981) for IBL).

The characterisation of predictive accuracy described here can be viewed as a means of calibrating an algorithm's estimate (i.e., typicality and posterior probability) with its (estimated) predicted accuracy. Cross-validation is used here as a tool in performing calibration. In statistics, the idea of calibration is presented as a way of realigning one's estimates with reality or some standard (e.g., Dawid (1982), Dunn (1989) and Efron & Tibshirani (1993)). Statisticians tend to use a linear model for calibration. In the current work, one may want to fit a model for the characterisation, which may or not be a linear model.

For Instance-Based classifiers, there are other choices with regard to the method of characterisation. For example, one may intend to use some method of kernel density estimation (Silverman 1986), which may solve the problem faced by using *typicality* mentioned above. We have attempted another method of characterisation that uses the *nearest neighbour distance*. The intuition is that one prediction made with a closer nearest neighbour would be expected to have higher predictive accuracy than another prediction made with a farther nearest neighbour. Mandler and Schürmann (1988) employ a similar approach in Instance-Based classifiers. However, this method does not performs well for a number of reasons. In domains that have multi-modal overlapping distributions, i.e., the data has maximal density in the regions of decision boundaries and sparse density elsewhere, the *nearest neighbour distance* would produce graphs which are completely opposite to the hypothesis; the test instances that are far from the boundaries are more likely to have larger nearest neighbour distances than those near to the boundaries (due to the data distribution); thus, they are more likely to predict correctly. Noise is another factor that would obscure the *nearest neighbour distance* as a good accuracy predictor in IB1-MVDM*.

While the methods of characterisation and estimation are not perfect for one reason or another, it suffices for our purpose here to show that the methods can be used to characterise and estimate predictive accuracy in most of the real-world domains studied. The real test is how effective are these methods in overcoming the problem of locally low predictive accuracy in the two learning algorithms, which is the topic of the next section.

The Method of Combining IB1-MVDM* and NB*

Having characterised and estimated predictive accuracy in IB1-MVDM* and NB*, we can now make use of their individual predictive accuracy estimation for each classification for the purpose of decision combination. The combining strategy that we use is to select an estimated better performing classifier to do the final prediction and we name the resultant classifier the composite learner (CL). Because only high performance predictions are selected, the CL's performance

is expected to be better than the individual algorithms. Employing the characterisations, combining the two algorithms using this strategy is straightforward.

Both IB1-MVDM* and NB* are first trained independently. During training, the algorithms perform three-fold cross-validation to estimate predictive accuracy of each prediction from the characterisations. This is done by using the test results from the three-fold cross-validation to produce a binned graph that relates the average value of the characterisation to its binned predictive accuracy (as described in the last section). For each classification, the predictive accuracy of each algorithm is obtained by referring to its binned graph with the corresponding value of the characterisation. The algorithm which has the higher estimated predictive accuracy is chosen to make the final prediction. Figure 4 depicts the classification process.

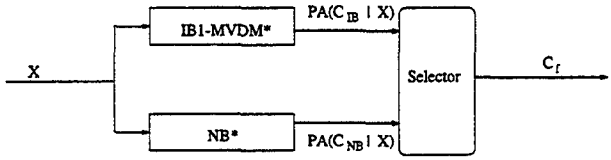


Figure 4: CL's classification process

Formally, the selection process can be defined as follows.

$$C_f = C_{IB} \text{ if } \mathrm{PA}(C_{IB}|X) > \mathrm{PA}(C_{NB}|X),$$
$$C_f = C_{NB} \text{ if } \mathrm{PA}(C_{IB}|X) < \mathrm{PA}(C_{NB}|X),$$
$$\text{else random select.}$$

where C_f : final prediction;
C_{IB} : IB1-MVDM*'s prediction;
C_{NB} : NB*'s prediction;
$\mathrm{PA}(C|X)$: predictive accuracy of prediction C given instance X.

Performance of the Composite Learner

We conduct experiments in the fifteen real-world domains to compare the performance of the composite learner and its constituent algorithms. CL combines IB1-MVDM* which uses typicality and NB* which uses posterior probability.

We randomly split the dataset into 90% and 10% disjoint subsets. Each experimental trial uses the 90% subset as a training set and the 10% subset as a testing set, and it is repeated over 50 trials. This experimental method is used throughout the rest of the paper.

Table 3 shows the average classification error rates and their standard errors of IB1-MVDM*, NB*, CL and BESTof3 – the best algorithm selected among IB1-MVDM*, NB* and CL based on the test results of three-fold cross-validation on training data. The best result in each domain is shown in boldface. The $\oplus$ (or $\ominus$) symbol in front of a figure indicates that it is significantly better (or worse) than the better of IB1-MVDM* and NB* by more than or equal to two

Table 3: Average error rates of CL and BESTof3

	*IB**	*NB**	*CL*	*BESTof3*
wave40	21.7 ±1.1	21.8 ±1.1	$\oplus$ **18.4**	$\oplus$ **18.4**
horse	20.0 ±1.0	19.9 ±0.9	$\oplus$ **16.9**	$\oplus$ 17.2
hypo	1.7 ±0.1	1.5 ±0.1	$\oplus$ **1.2**	$\oplus$ **1.2**
led24	38.4 ±1.4	38.0 ±1.8	$\oplus$ **35.5**	37.9
wave21	22.1 ±1.2	22.9 ±1.1	$\oplus$ **18.5**	$\oplus$ 19.2
heart	19.4 ±0.9	18.5 ±0.9	**17.7**	18.6
splice	5.6 ±0.1	4.5 ±0.1	$\oplus$ **4.0**	$\oplus$ **4.0**
bcw	4.6 ±0.3	2.9 ±0.3	**2.8**	2.9
soyb	5.8 ±0.4	7.7 ±0.5	$\oplus$ **5.0**	5.6
glass	27.7 ±1.2	29.9 ±1.3	**26.6**	28.4
diab	28.8 ±0.6	25.1 ±0.6	**24.1**	24.3
led7	32.7 ±1.4	**28.9** ±1.3	29.4	29.9
hepa	20.6 ±1.6	**14.9** ±1.2	15.5	**14.9**
echo	36.0 ±1.9	**28.1** ±1.5	29.3	29.3
auto	**14.8** ±1.0	31.0 ±1.1	$\ominus$ 18.9	**14.8**

IB* : IB1-MVDM*.

Table 4: Summary of Table 3

	CL	*BESTof3*
#wins vs #losses	11-4	9-4
#signif. wins vs #signif. losses	7-1	5-0
Sign test	96.5	96.9

standard errors ($\geq$ 95% confidence). The domains are ordered by the absolute error rate difference between IB1-MVDM* and NB*. This ordering divides these domains into two groups; one that CL performs better than the better of IB1-MVDM* and NB*, and one that CL performs worse. A Wilcoxon rank-sum one-tailed test indicates that the ordering is strongly related to the performance of CL with 99.9% confidence. In eleven domains where the performance of IB1-MVDM* and NB* have small differences, CL achieves better results than the better of its constituent algorithms. In the other four domains, where the performance of IB1-MVDM* and NB* differ substantially, CL performs in between the performance of its constituent algorithms but never performs better than the better of the two algorithms.

A summary of these results is given in Table 4. The first row shows the number of domains in which CL or BESTof3 achieves higher accuracy than the better of IB1-MVDM* and NB* versus the number in which the reverse happened. The second row considers only those domains in which the difference is significant with at least 95% confidence. The third row shows the results of applying a sign test to the values in the second row. This results in a confidence more than 95% that either CL or BESTof3 is a more accurate learner than the better of IB1-MVDM* and NB*.

Ablation Analysis

To gain insight into the factors that affect the performance of the composite learner, we examine (i) the classification overlaps made by the constituent algorithms and (ii) the correct classifications made by only

one of them. Table 5 shows a complete breakdown of average correct classifications in the diabetes domain and Table 6 summarises the results in the fifteen domains. Figures in the last row of Table 5 are computed using Equations (2) and (3). All figures in Table 6 are calculated using Equations (6)-(9) which correspond to the last four columns in the table. All equations are listed as follows.

$$CC[Total] = CC[0] + CC[1] + CC[2] \quad (2)$$

$$CC_{CL}[Total] = CC_{CL}[1] + CC_{CL}[2] \quad (3)$$

$$CC_{CL}[0] = 0 \quad (4)$$

$$CC_{CL}[2] = CC[2] \quad (5)$$

$$\%\ \text{Overlap} = \frac{CC[0] + CC[2]}{CC[Total]} \times 100 \quad (6)$$

$$R = \frac{CC[1]_{IB1-MVDM*}}{CC[1]} \times 100 \quad (7)$$

$$S = \frac{CC[1]_{NB*}}{CC[1]} \times 100 \quad (8)$$

$$W = \frac{CC_{CL}[1]}{CC[1]} \times 100 \quad (9)$$

where $CC[N]$ is the average number of correct classifications made by N algorithms. $CC_{CL}[N]$ is the average number of correct classifications out of $CC[N]$ made by the composite learner. $CC[1]_{AL}$ is the portion of $CC[1]$ for which only algorithm AL makes the correct classification. R (S) denotes the percentage of IB1-MVDM* (NB*) only correct classifications. W denotes the percentage of CL correct classifications from R+S.

The composite learner improves performance if it makes the correct selection most of the time when only one of its constituent algorithms makes the correct classification; thus, $CC_{CL}[1] \leq CC[1]$. The composite learner can do nothing when both algorithms make incorrect or correct classifications. $CC[0]$ is just the average number of classifications in which both algorithms are incorrect. Since the composite learner always makes incorrect classifications in this portion of predictions, whichever algorithm is selected, $CC_{CL}[0]$ must be zero. Equation (6) gives the percentage of classification overlap between IB1-MVDM* and NB* over all classifications. The proportions of either IB1-MVDM* and NB* making correct classification and CL making the right choice over $CC[1]$ are given in Equations (7)-(9).

The results from Table 6 reveal that the degree of classification overlap does not affect the performance of CL. Summing the average percentage of CL correct classification over fifteen domains in Table 6 (the last column), the mean is 68.1% (an oracle would achieve 100% and a random choice is 50%). This indicates that CL can obtain better results than either of its constituent algorithms in those domains in which either IB1-MVDM* or NB* covers less than 68% of the total only one algorithm correct classification (the only exception is the bcw domain).

Table 5: Average #correct classifications in the pima diabetes domain

N	$CC[N]$ (%)	*#CL correct classifications* $CC_{CL}[N]$ (%)
0	12.3 (16.0%)	0.0 (0.0%)
1	16.9 (21.9%)	10.6 (13.8%)
2	47.8 (62.1%)	47.8 (62.1%)
Total	77.0 (100.0%)	58.4 (75.9%)

Table 6: Average percentage of classification overlap and correct classification

	% Overlap (#test inst.)	R	S	*% CL Correct from R+S, W*
wave40	79.3 (30)	50.2	49.8	**66.2**
horse	86.1 (37)	49.6	50.4	**72.5**
hypo	98.6 (317)	45.3	54.7	**74.7**
led24	67.7 (20)	49.1	50.9	**62.4**
wave21	79.0 (30)	51.7	48.3	**68.9**
heart	84.1 (31)	47.2	52.8	**57.3**
splice	94.1 (318)	40.0	60.0	**69.1**
bcw	97.4 (70)	16.7	83.3	**86.7**
soyb	92.1 (69)	62.5	37.5	**72.8**
glass	77.8 (22)	55.7	44.3	**61.3**
diab	78.1 (77)	41.6	58.4	**63.1**
led7	85.3 (20)	26.8	**73.2**	67.1
hepa	85.0 (16)	30.8	**69.2**	65.0
echo	83.3 (14)	26.5	**73.5**	66.7
auto	72.5 (21)	**84.9**	15.1	67.3

Boldface indicates the maximum value in the last three columns for each domain.

In the following experiments, we compare the composite learner with a closely related method of decision combination but without using the characterisations.

Comparison with Stacked Generalisation

Stacked generalisation (Wolpert 1992) is proposed as a more sophisticated version of cross-validation (Linhart & Zucchini 1986) that uses a learning algorithm to determine how the classification outputs of the primitive learning algorithms should be combined. The primitive inputs and the primitive learning algorithms or generalisers are referred to as operating in the zero-level description space. The first-level inputs are the outputs of the zero-level generalisers (possibly plus the zero-level inputs). The learning algorithm used in this level is called the first-level generaliser.

Here, CL is compared to three methods of stacked generalisation:

a. simple stacking: use only the outputs of IB1-MVDM* and NB* as inputs to the first-level generaliser, SG_a,

b. use the zero-level inputs and the outputs of IB1-MVDM* and NB* as inputs to the first-level generaliser, SG_b,

c. use only the average test results of cross-validation (Schaffer 1993) on the training set to select the better of IB1-MVDM* and NB*, denoted as BETTERof2.

The first-level generaliser in SG_a and SG_b can either be IB1-MVDM* or NB* in the following experiments. Note that three-fold cross-validations are used in all experiments. Table 7 shows the average classification error rates of three types of stacked generalisation with comparison to the composite learner. SG_x(IB1) uses IB1-MVDM* as the first-level generaliser and SG_x(NB) uses NB* as the first-level generaliser. The $\ominus$ ($\oplus$) symbol in front of a figure indicates that it is significantly worse (better) than CL (i.e., the difference between CL and a stacked generalisation method is more than or equal to two standard errors ($\geq$ 95% confidence)). The best result in each domain is shown in boldface.

CL performs better than or equal to the first two types of stacked generalisation, SG_a and SG_b in most domains and a considerable number of the differences are significant. For example, CL is significantly better than SG_b(IB1) in nine domains. Only in a few domains, do some versions of stacked generalisation achieve minor improvements over CL but the differences are not significant (for example, in the hepatitis, echocardiogram and automobile domains). Note that these are the domains which the algorithms have poor characterisations of predictive accuracy (i.e., demonstrate insignificant trends in Table 2).

BETTERof2 performs close to selecting the better of IB1-MVDM* and NB*. While this is a strong point of the method, it also demonstrates that this method cannot produce better results than the better primitive algorithm. CL achieves significantly better results than BETTERof2 in five domains and significantly worse in one domain.

Table 8 shows a summary of these results. The first row shows the number of domains in which CL achieves higher accuracy than a stacked generalisation method versus the number in which the reverse happened. A similar comparison for only those domains in which the difference is significant is shown in the second row. The third row shows the results of a sign test on the values of the second row. This reveals that CL is a more accurate learner than the three types of stacked generalisation with at least 95% confidence.

Table 7: Comparison with stacked generalisations (average error rates)

	CL	*SG_a* (IB1)	*SG_a* (NB)	*SG_b* (IB1)	*SG_b* (NB)	*BET*
wave40	**18.4**	$\ominus$ 21.0	$\ominus$ 20.8	$\ominus$ 22.0	$\ominus$ 21.3	$\ominus$ 21.6
horse	**16.9**	17.8	17.4	$\ominus$ 19.8	$\ominus$ 20.1	$\ominus$ 20.4
hypo	**1.2**	$\ominus$ 1.4	1.3	$\ominus$ 1.4	$\ominus$ 1.6	$\ominus$ 1.6
led24	**35.5**	$\ominus$ 41.0	$\ominus$ 38.4	**37.5**	**35.0**	$\ominus$ 38.8
wave21	**18.5**	$\ominus$ 21.1	20.6	$\ominus$ 21.1	$\ominus$ 22.4	$\ominus$ 22.5
heart	**17.7**	18.4	18.4	$\ominus$ 20.8	**18.5**	**18.5**
splice	**4.0**	$\ominus$ 4.5	$\ominus$ 4.6	$\ominus$ 4.5	$\ominus$ 4.4	$\ominus$ 4.5
bcw	**2.8**	3.0	$\ominus$ 3.9	$\ominus$ 3.7	**2.9**	**2.9**
soyb	**5.0**	5.1	5.4	$\ominus$ 5.9	$\ominus$ 7.5	$\ominus$ 5.8
glass	**26.6**	$\ominus$ 29.6	27.4	$\ominus$ 30.7	**28.6**	**28.5**
diab	**24.1**	$\ominus$ 25.3	$\ominus$ 25.7	$\ominus$ 26.6	**25.1**	**25.2**
led7	29.4	31.9	30.5	$\ominus$ 32.2	**29.0**	29.4
hepa	**15.5**	**15.1**	17.6	$\ominus$ 19.0	**14.8**	**14.6**
echo	**29.3**	$\ominus$ 32.3	**29.9**	$\ominus$ 36.3	**28.7**	**28.9**
auto	**18.9**	**19.3**	**18.1**	**17.4**	$\ominus$ 29.4	$\oplus$ **14.8**

$\oplus$ or $\ominus$: significantly better or worse than CL; p$\leq$0.05.
BET : BETTERof2.

Table 8: Summary of Table 7

	SG_a (IB1)	*SG_a* (NB)	*SG_b* (IB1)	*SG_b* (NB)	*BET*
#wins	14-1	14-1	14-1	11-4	11-3
#signif. wins	8-0	5-0	13-0	7-0	7-1
Sign test	99.6	96.9	99.99	99.2	96.5

Discussion

Because the composite learner employs a strategy that probes into the intrinsic problem of the induced theories of its constituent algorithms and uses only those predictions that are more likely to be correct, it is able to improve the performance of the individual algorithms. Using a cross-validation algorithm such as BETTERof2 does not get better performance than the better primitive learning algorithm. Cross-validation uses global information (i.e., the overall estimated error rate) to do model selection *before* classification, whereas the composite learner uses local information (i.e., the characterisation of predictive accuracy of each prediction) to select which model to use *during* classification.

The fact that CL achieves only 68% correct on average (in the portion where only one of IB1-MVDM* and NB* has correct classification) explains why the current implementation of the composite learner can only achieve better results in domains where the constituent algorithms have comparable performance. This indicates that there is still room for further improvement. First, the methods for characterising predictive accuracy can be improved. Some of the problems with *typicality* have been discussed in the last discussion section. Use of an exponential distance function rather than Euclidean distance function in Equation (1) may improve the characterisation. Second, the predictive accuracy estimation method of producing binned graphs can be improved. Current weaknesses can be mitigated by employing cross-validation to select the best algorithm among the composite learner and its constituent algorithms. This method, BESTof3 is very effective in domains such as hepatitis and automobile where the performance between IB1-MVDM* and NB* differs substantially. Note that CL does not perform well in these two domains because the characterisations of predictive accuracy are poor (indicated as insignificant trends in Table 2). Overall, BESTof3 is shown to achieve results comparable to the best results of CL and its constituent learning algorithms in all the ex-

perimental domains. Third, in two-class domains, an improvement can possibly be made by simply reverse the prediction when the binned graph for a class is below 50% accuracy (see the bottom left portions of the binned graphs for typicality in Figures 2 and 3).

There are two situations when one algorithm performs comparably to the other. The first situation is when the decision boundaries almost overlap; in this case, the classification overlap would be very high and we say that the algorithms are highly correlated. In the second situation, as the classification overlap is relatively low, the proportion classified correctly by one algorithm while the other algorithm is incorrect is about the same for both algorithms. Nevertheless, the composite learner is capable of performing well under both situations. Classification correlation between the constituent algorithms in some domains does not affect the performance of the composite learner. This is shown in the bcw and hypothyroid domains where the percentage of classification overlap is over 97% (see Table 6).

There are a number of reasons why the composite learner performs better than stacked generalisation. Stacked generalisation merely delays the decisions to be made when one is confronted with a learning problem. The decisions regarding a learning problem such as the number of inputs to be used (*the feature selection problem*) and what type of learning algorithm is suitable (*the selective superiority problem*, Brodley 1993) in the first-level cannot be solved in stacked generalisation; they still have to be decided (manually by users) in the new transformed space at the higher level. The same problems faced in the new transformed space do not become simpler, in fact, they are as hard as those faced in the original space. We have also tried to add the characterisations (and also a binary attribute indicating whether the characterisation from one algorithm is better than that from the other) as first-level inputs in the new transformed space, but stacked generalisations still fail to achieve better results. All these are due to the fact that the first-level inputs (i.e., the zero-level outputs possibly including the characterisations) are highly correlated or they are redundant with respect to the primitive inputs (John, Kohavi & Pfleger 1994; Langley & Sage 1994). The composite learner addresses the selective superiority problem given that Instance-Based and Naive Bayesian classifiers are chosen to solve the learning problem at hand, by using the characterisation of predictive accuracy in each classifier. Because only the characterisations are used, the feature selection problem does not exist in the composite learner framework. The success of the composite learner relies on the accuracy of this characterisation and its predictive accuracy estimation.

The composite learner can be viewed as an "informed" simple generaliser that makes its decision to select a classifier based on the weight of the prediction. The weight of the prediction can be regarded as the classifier's "confidence" concerning the prediction made. This simple generaliser is demonstrated to be better than more complex generalisers (e.g., SG_x(IB1) and SG_x(NB)), mainly due to the weight truly reflecting the classifier's "confidence" about the prediction made.

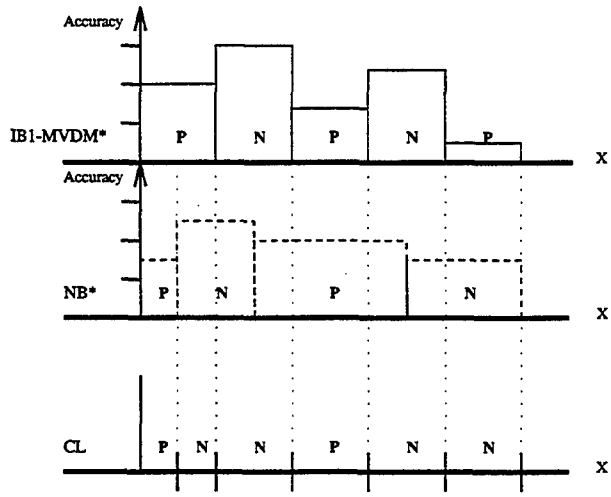


Figure 5: Theory superposition based on predictive accuracy

Though the composite learner might be viewed as a special case of stacked generalisation that uses an "informed" simple generaliser based on the prediction weights, it does not generalise in a new transformed space. What it does is a superposition of the theories learned from the primitive attributes. Figure 5 illustrates how one theory is superpositioned on another theory in a one-dimension space domain. The vertical axes in the first two diagrams (for IB1-MVDM* and NB*, respectively) indicate the estimated predictive accuracies in each region of the description space (plotted in the same scale); for simplicity, they are assumed to be uniform in each region. P shows the positive class regions and N the negative class regions. The bottom plot shows the regions cast by the composite learner by selectively choosing the more highly predictive regions.

The proposed composite learner can be easily extended to include other learning algorithms, provided appropriate characterisations of those induced theories can be found. For example, the probabilities of a neural network's outputs could be used as the characterisation. In such cases, a scheme that combines the weights/evidences of the predictions might need to be incorporated into the system (such as those in Buntine (1991) and Perrone & Cooper (1993)).

Related Work

The most closely related work that exploits local information in combining different learning algorithms

is Ting's (1994a) treatment of the problem of small disjuncts in decision trees. The key idea of this work is to use Instance-Based methods to solve the problem of small disjuncts in decision trees by replacing the small disjunct regions with Instance-Based methods, which are shown to have better performance in these regions in most domains. Though a similar idea is used here, the roles of decision trees and Instance-Based methods are fixed in that framework – decision trees work on large disjuncts regions and Instance-Based methods work in small disjuncts regions. This arrangement is based on the assumption that decision trees perform better than Instance-Based methods in the regions of large disjuncts. In domains where this assumption does not hold, the performance of this composite learner will be worse than Instance-Based methods. KBNGE (Wettschereck 1994) in many respects resemble Ting's work without using an explicit characterisation of the problem of small disjuncts. It uses BNGE (i.e., a nested generalised exemplars algorithm (Salzberg 1991)) in regions that clearly belong to one class and a k-nearest neighbour algorithm otherwise.

In an independent work, Merz (1995) describes another method of stacked generalisation where selection of algorithms can be done during classification. The DS (dynamic selection) algorithm builds two m by n matrices during training, where m is the number of training instances and n is the number of learning algorithms. The first matrix records the predictions of the induced theories and the second matrix records the test results of w v-fold cross-validations for each training instance, where w must be more than 1. All induced theories are used in classifying a new instance to construct a row-vector that can be compared in the first matrix to identify the rows "closest" to the row-vector of the new instance. The accuracies for each induced theory are computed from the corresponding rows (by averaging) in the second matrix. The induced theory having the highest accuracy is selected for final prediction.

There are several differences between DS and CL despite the facts they both use cross-validation and perform algorithm selection during classification. First, as other methods of stacked generalisation, DS uses another generaliser (i.e., some forms of nearest-neighbour classifier) in the high-level transformed space. The characterisation of various regions of differing predictive accuracies is more explicit in CL. It is much harder to identifying those regions in DS. CL is using a binned graph for each class whereas DS is using two matrices. DS requires w v-fold cross-validations and CL only requires one v-fold cross-validation. DS relies only on the test results of cross-validations and CL requires an extra measure of characterisation for each induced theory. Merz (1995) reports that '..DS frequently outperforms a cross-validation algorithm for selecting a learning algorithm and occasionally outperforms the algorithm with the best test accuracy.' [4] CL performs better than both BETTERof2 and the better of its constituent algorithms in most of the domains tested.

MCS (Brodley 1993) uses a hand-crafted rule to guide the construction of three different models at the nodes of a decision tree, and each model is trained on part of the training set. In contrast, the two algorithms in the composite learner are trained independently on the total training data and work cooperatively according to the characterisation of predictive accuracy in each learned theory. Thus, no hand-crafted knowledge is used in the composite learner. CL selects a model during classification; MCS fixes the various models at the nodes of a decision tree during training.

Methods of selecting a learning algorithm for a given domain (e.g., Schaffer 1993; Aha 1992; Breiman et al 1984) using the entire set of training data only choose the best performing algorithm, at best. Some other methods split the data into (i) mutually exclusive subsets (Tcheng et al 1989; Utgoff 1989; Chan & Stolfo 1995) or (ii) resampling (with replacement) subsets (Drucker, Schapire & Simad 1993). Different classifiers are trained using these subsets.

Breiman's (1994) bagging (for bootstrap aggregating) predictors combine multiple models produced from a single algorithm using bootstrap replicate training sets. Predictions are combined either using a majority vote or averaging the estimated class probabilities of all bootstrap models. Kwok and Carter (1990) use voting over multiple trees generated by using alternative splits.

Several methods of re-ordering ranks when combining multiple models have been proposed in the literature. Buntine (1991) introduced an algorithm design strategy based on the approximating Bayesian decision theory of learning class probability decision trees. The class ranking was re-ordered after averaging the class probabilities from the multiple trees. With the present work on the characterisation of predictive accuracy, Buntine's strategy can be readily applied to include different models. Perrone and Cooper's (1993) ensemble methods work in a similar manner by merely (weighted) averaging the corresponding outputs of the multiple neural networks. Error correlation among the combined algorithms will severely affect the weighting and the performance of the ensemble method. Smyth, Goodman & Higgins (1990) and Kononenko & Kovačič (1992) use Naive Bayesian combination of decisions of several different rules in re-ordering ranks. Ho, Hull and Srihari (1994) employ logistic regression to re-rank classes across different types of classifiers, and only the ranks of the classes in the predictions are considered.

Finally, Rachlin et al (1994) and Ting & Cameron-Jones (1994) have addressed issues (e.g., learnability in the limit, the two algorithms' relationships in one framework) regarding the two types of algorithm used

[4] This implies that a cross-validation algorithm does not perform better than the best performing algorithm. We have the same finding as indicated in Table 7.

in this paper.

Conclusion

The main contribution of this paper is two-fold. First, we have characterised an intrinsic problem, i.e., the problem of locally low predictive accuracy in IBL and Naive Bayes. Two intuitively sound methods, the *typicality* and the *posterior probability* have been found to be a satisfactory characterisation of predictive accuracy in IB1-MVDM* and NB*, respectively. Second, knowing the weaknesses in each algorithm, the simple strategy of selecting as the final prediction the one that has the higher estimated predictive accuracy has been demonstrated to partially overcome the problem of locally low predictive accuracy. When IB1-MVDM* and NB* demonstrate comparable performance, we strongly recommend that the proposed composite learner is used to improve the overall performance. The composite learner is found to outperform three methods of stacked generalisation (including a common cross-validation method for model selection, BETTERof2) in most of the experimental domains studied. BESTof3 incorporating the composite learner achieves results better than or comparable to the better performance of the composite learner's constituent learning algorithms in all the experimental domains.

Acknowledgement

The core of this research was done when this author was in Basser Department of Computer Science, The University of Sydney. It was partially supported by an Australia Research Council grant (to J.R.Quinlan) and by a research agreement with Digital Equipment Corporation. Numerous discussions with J.R. Quinlan, R.M. Cameron-Jones, Z. Zheng and P. Langley have been very helpful. Thanks to D.W. Aha for providing IB1 algorithm. This author was partially supported by the Equity and Merit Scholarship Scheme.

References

Aha, D.W. (1992), Generalizing from case studies: A Case Study, in *Proceedings of the Ninth International Conference on Machine Learning*, pp. 1-10. Morgan-Kaufmann.

Aha, D.W., D. Kibler & M.K. Albert (1991), Instance-Based Learning Algorithms, *Machine Learning, 6*, pp. 37-66.

Breiman, L. (1994), Bagging Predictors, *Technical Report 421*, Department of Statistics, University of California, Berkeley, CA.

Breiman, L., J.H. Friedman, R.A. Olshen & C.J. Stone (1984), *Classification And Regression Trees*, Belmont, CA: Wadsworth.

Brodley, C.E. (1993), Addressing the Selective Superiority Problem: Automatic Algorithm/Model Class Selection, in *Proceedings of the Tenth International Conference on Machine Learning*, pp. 17-24.

Buntine, W. (1991), Classifiers: A Theoretical and Empirical Study, in *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pp. 638-644, Morgan-Kaufmann.

Cestnik, B. (1990), Estimating Probabilities: A Crucial Task in Machine Learning, in *Proceedings of the European Conference on Artificial Intelligence*, pp. 147-149.

Chan, P.K. & S.J. Stolfo (1995), A Comparative Evaluation of Voting and Meta-learning on Partitioned Data, in *Proceedings of the Twelfth International Conference on Machine Learning*, pp. 90-98, Morgan Kaufmann.

Cost, S & S. Salzberg (1993), A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features, *Machine Learning, 10*, pp. 57-78.

Dawid, A.P. (1982), The Well-Calibrated Bayesian, *Journal of the American Statistical Association*, Vol. 77, No. 379, pp. 605-610.

Dietterich, T.G. (1990), Machine Learning, in *Annual Review of Computer Science 4*, pp. 255-306.

Drucker, H., R. Schapire & P. Simad (1993), Improving the performance in neural networks using a boosting algorithm, *Advances in Neural Information Processing Systems*, 5, pp. 42-49.

Dunn, G. (1989), *Design and Analysis of Reliability Studies*, Oxford University Press.

Efron, B. & R.J. Tibshirani (1993), Chapter 18 in *An Introduction to the Bootstrap*, Chapman & Hall.

Fayyad, U.M. & Irani, K.B. (1993), Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning, in *Proceedings of 13th International Joint Conference on Artificial Intelligence*, pp. 1022-1027.

Ho, T.K., J.J. Hull & S.N. Srihari (1994), Decision Combination in Multiple Classifier Systems, in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.16, no.1, pp. 66-75.

Holte, R.C., L.E. Acker & B.W. Porter (1989), Concept Learning and the Problem of Small Disjuncts, in *Proceedings of the 11th International Joint Conference on Artificial Intelligence*, pp. 813-818.

Howell, D.C. (1982), *Statistical Methods for Psychology*. PWS Publishers.

John, G.H., R. Kohavi & K. Pfleger (1994), Irrelevant Features and the Subset Selection Problem, in *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 121-129, Morgan Kaufmann.

Kononenko, I. & M. Kovačič (1992), Learning as Optimization: Stochastic Generation of Multiple Knowledge, in *Proceedings of the Ninth International Conference on Machine Learning*, pp. 257-262, Morgan Kaufmann.

Kwok, S. & C. Carter (1990), Multiple Decision Trees, *Uncertainty in Artificial Intelligence 4*, ed. R. Shachter, T. Levitt, L. Kanal and J. Lemmer, pp. 327-335, North-Holland.

Langley, P. & Sage, S. (1994), Induction of Selective Bayesian Classifiers, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pp. 399-406, Seattle, WA: Morgan Kaufmann.

Linhart, H. and Zucchini, W. (1986), *Model Selection*, NY: Wiley.

Mandler, E. and J. Schürmann (1988), Combining the Classification Results of Independent Classifiers based on the Dempster/Shafer Theory of Evidence, in *Pattern Recognition and Artificial Intelligence* by E.S. Gelsema and L.N. Kanal (Editors), pp. 381-393, North-Holland.

Merz, C.J. (1995), Dynamic Learning Bias Selection, in *Proceedings of the Fifth International Workshop on Artificial Intelligence and Statistics*, Ft. Lauderdale, FL: Unpublished, pp. 386-395.

Murphy, P. M. & D. W. Aha (1994), *UCI Repository of machine learning databases*, [http://www.ics.uci.edu/ mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

Perrone, M.P. & L.N. Cooper (1993), When Networks Disagree: Ensemble Methods for Hybrid Neural Networks, in *Artificial Neural Networks for Speech and Vision*, R.J. Mammone (editor), Chapman-Hall.

Rachlin, J., S. Kasif, S. Salzberg, & D.W. Aha (1994), Towards a better understanding of memory-based reasoning systems, in *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 242-250. Morgan Kaufmann.

Rosch, E. & C.B. Mervis (1975), Family Resemblances: Studies in the Internal Structures of Categories, *Cognitive Psychology*, vol. 7, pp. 573-605, Academic Press.

Salzberg, S. (1991), A Nearest Hyperrectangle Learning Method, *Machine Learning*, 6, pp. 251-276.

Schaffer, C. (1993), Selecting a Classification Method by Cross-validation. *Preliminary Papers of the Fourth International Workshop on Artificial Intelligence and Statistics*, pp. 15-25.

Silverman, B.W. (1986), *Density Estimation for Statistics and Data Analysis*, Chapman and Hall.

Smyth, P., R.M. Goodman & C. Higgins (1990), A Hybrid Rule-based/ Bayesian Classifier, in *Proceedings of the Ninth European Conference on Artificial Intelligence*, pp. 610-615.

Tcheng, D., B. Lambert, C-Y. Lu & L. Rendell (1989), Building Robust Learning Systems by Combining Induction and Optimization, in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pp. 806-812, Morgan Kaufmann.

Ting, K.M. (1994a), The Problem of Small Disjuncts: its remedy in Decision Trees, in *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, pp. 91-97.

Ting, K.M. (1994b), The Problem of Atypicality in Instance-Based Learning, in *Proceedings of the Third Pacific Rim International Conference on Artificial Intelligence*, pp. 360-366, International Academic, Beijing.

Ting, K.M. (1994c), *Discretization of Continuous-Valued Attributes and Instance-Based Learning*, Technical Report No.491, Basser Department of Computer Science, University of Sydney.

Ting, K.M. (1996), Discretisation in Lazy Learning Algorithms, in the special issue of Lazy Learning in *Artificial Intelligence Review Journal*. Forthcoming.

Ting, K.M. & Cameron-Jones, R.M. (1994), Exploring a Framework for Instance Based Learning and Naive Bayesian Classifiers, in *Proceedings of the Seventh Australian Joint Conference on Artificial Intelligence*, World Scientific, pp. 100-107.

Utgoff, P.E. (1989), Perceptron Trees: A case study in hybrid concept representations, *Connection Science*, 1, pp. 337-391.

Watson, D.F. (1981), Computing the n-dimensional Delaunay Tessellation with application to Voronoi Polytopes, *The Computer Journal*, vol.24, no.2, pp. 167-172, Heyden & Son Ltd.

Wettschereck, D. (1994), A Hybrid Nearest-Neighbor and Nearest-Hyperrectangle Algorithm, in *Proceedings of the Seventh European Conference on Machine Learning, LNAI-784*, pp. 323-335, Springer Verlag.

Wolpert, D.H. (1992), Stacked Generalization, *Neural Networks*, vol.5, pp. 241-259, Pergamon Press.

Zhang, J. (1992), Selecting Typical Instances in Instance-Based Learning, *Proceedings of the Ninth International Conference on Machine Learning*, pp. 470-479, Morgan Kaufmann.

A Multistrategy Learning System for Planning Operator Acquisition

Xuemei Wang*
Computer Science Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213, U.S.A
wxm@cs.cmu.edu

Abstract

This paper describes a multistrategy learning approach for automatic acquisition of planning operators. The two strategies are: (*i*) learning operators by observing expert solution traces, and (*ii*) refining operators through practice in a learning-by-doing paradigm.

During observation, OBSERVER uses the knowledge that is *naturally observable* when experts solve problems, without the need of explicit instruction or interrogation. During practice, OBSERVER generates its own learning opportunities by solving practice problems. The *inputs* to our learning system are: the description language for the domain, experts' problem solving traces, and practice problems to allow learning-by-doing operator refinement. Given these inputs, our system *automatically* acquires the preconditions and effects (including conditional effects and preconditions) of the operators.

Our approach has been fully implemented in a system called OBSERVER on top of a non-linear planner PRODIGY. We present empirical results to demonstrate the validity of our approach in a process planning domain. These results show that the system learns operators in this domain well enough to solve problems as effectively as human-expert coded operators. The results also show the learned operators are more effective if both strategies are used, than if only one strategy is used. Therefore, both learning strategies contribute significantly to the learning process.

Introduction

Acquiring and maintaining domain knowledge is a key bottleneck in fielding planning systems for realistic domains (Chien *et al.* 1995). For example, significant effort has been devoted to writing and debugging planning operators for successful applications of planning systems (desJardins 1994; Chien 1996; Gil 1991).

While considerable research has focused on *knowledge acquisition tools* for rule-based systems (see (Boose and Gaines 1989) for a summary), and some on such tools for specialized planning systems (Chien 1994), these systems all require a considerable amount of direct interactions with domain experts. It is possible to *automate* the process of knowledge acquisition when a simulator is available, although there has been little work in this area.

Simulators are commonly available in areas such as CAD, robotics assembly planning, manufacturing, process planning. Simulating is different from planning. Simulators simply act as simplified surrogates for the real world, bypassing problems of coping with sensors and physical effectors. In contrast, planning is the process of generating a sequence of actions to transform an initial situation into a goal situation. Planning requires a specific form of knowledge that simulators do not have.

We have developed a multistrategy learning system OBSERVER, that automatically transforms simulator knowledge into planning operators that can be used for planning. Our approach is to learn planning operators by observing expert solution traces and to further refine the operators through practice by solving problems in the simulator in a learning-by-doing (Anzai and Simon 1979) paradigm. During observation, OBSERVER uses the knowledge *naturally observable* when experts solve problems, without need of explicit instruction or interrogation. Learning from observation allows OBSERVER to bootstrap itself, otherwise since OBSERVER initially does not have any knowledge about the preconditions and the effects of the operators, it has to uses random exploration, which is much less efficient. During practice, OBSERVER generates its own learning opportunities by solving practice problems.

The *inputs* to OBSERVER are:

- The description language for the domain. This includes the types of objects and the predicates that describe states and operators.
- Observations of an expert agent consisting of: 1) the sequence of actions being executed, 2) the state in which each action is executed (*pre-state*), and 3) the state resulting from the execution of each action (*post-state*).

*This research is sponsored by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant number F33615-93-1-1330. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of Wright Laboratory or the United States Government. Thanks to Jaime Carbonell, Jill Fain, Douglas Fisher, Herb Simon, and Manuela Veloso for their suggestions and support.

- Practice problems used by the learning system for practice in order to further refine incomplete operators.

Given these inputs, our system *automatically* acquires the preconditions and the effects of the operators. This representation is exactly what is required by most operator-based planners such as STRIPS (Fikes and Nilsson 1971), TWEAK (Chapman 1987), PRODIGY (Carbonell *et al.* 1992; Veloso *et al.* 1995), SNLP (McAllester and Rosenblitt 1991), and UCPOP (Penberthy and Weld 1992). Our learning method has been fully implemented on top of the PRODIGY architecture, a complete, state-space, non-linear planner (Veloso and Stone 1995).

The learning method described in this paper is domain-independent, although the examples are from the process planning domain. The process planning task is to generate plans to produce parts given part specifications, such as the shape, the size along each dimension, and the surface quality, by using processes such as drilling and milling. Different machines, such as drills, milling-machines, and different tools such as spot drills, twist-drills are available.

The structure of this paper is as follows. We first present a high level view of the learning system, followed by discussions on the issues that arise from learning by observation and practice. We then describe the detailed algorithm for learning operators from observation. These initially learned operators maybe be incomplete and incorrect, they are further refined during practice. We describe our algorithms for planning with incomplete and incorrect operators during practice, and our algorithm for refining operators during practice. Finally we present empirical results and analysis to demonstrate the effectiveness of the learning system, and the significance of both learning strategies. We end with discussions on related work, future work, and conclusions.

Learning architecture overview

OBSERVER's learning algorithms make several assumptions. First, since OBSERVER is operating within the framework of classical planners, it assumes that the operators and the states are deterministic. Second, OBSERVER assumes noise-free sensors, i.e., there are no errors in the states. And finally, we notice that in most application domains, the majority of the operators have conjunctive preconditions only. Therefore, OBSERVER assumes that the operators have conjunctive preconditions. This assumption greatly reduces the search space for operator preconditions without sacrificing much of the generality of learning approach.

Figure 1 shows the architecture of our learning system OBSERVER. There are three main components:

Learning operators from observation: OBSERVER inductively learns an initial set of operators from the observations of expert solutions given very little initial knowledge. Details of the learning algorithm are described in this paper.

Planning, plan repair, and execution: The initial set of operators learned from observation can be incomplete and incorrect in many ways. They are further refined when OBSERVER uses them to solve practice problems. Given a practice problem, OBSERVER first generates an initial plan to solve the problem. The initial plan is executed in the environment, resulting in both successful and unsuccessful executions of operators. OBSERVER uses these executions as positive or negative training examples for further operator refinement. The planner also repairs the failed plans upon unsuccessful executions. The repaired plans are then executed in the environment. This process repeats until the problem is solved, or until a resource bound is exceeded. Details of the integration of planning, execution, and learning can be found in (Wang 1996b).

Refining operators during practice: The successful and unsuccessful executions generated during practice are effective training examples that OBSERVER uses to further refine the initial imperfect operators. Details for operator refinement during practice are described in this paper.

Issues of learning planning operators

OBSERVER learns STRIPS-like operators that include preconditions and effects (including conditional effects and preconditions). Figure 2 is an example of such an operator from the process planning domain. Note that in Figure 2, `(not (has-burrs <part>))` is a *negated precondition*, meaning that HOLD-WITH-VISE can only be applied when `(has-burrs <part>)` is absent in the state. Also note that `(add (holding-weakly <machine> <holding-device> <part> <side>))` and `(add (holding <machine> <holding-device> <part> <side>))` are *conditional effects*, with their corresponding *conditional preconditions* being `(shape-of <part> CYLINDRICAL)` and `(shape-of <part> RECTANGULAR)`. Each conditional effect occurs *only* when its corresponding conditional preconditions are satisfied.

An important point is that negated preconditions are rare in most application domains. For example, in our process planning domain, there are only 25 negated preconditions among 350 preconditions in the human-expert coded operators. In many other domains implemented in PRODIGY, such as the extended-strips domain, there are no negated preconditions. This is because people tend to prefer thinking in terms of "something should be true in the state," which corresponds to non-negated preconditions, to thinking "something can not be true", which corresponds to negated preconditions. Also, one can easily convert a negated precondition to a non-negated precondition by introducing a new predicate,

OBSERVER learns operator preconditions by generalizing from the observed *pre-state*. In our application domains, the number of facts in the *pre-state* and *post-state* are typically much larger than the number of preconditions and effects of the corresponding operators. This is because many facts in the state are not relevant for the operator. For example, in the process planning domain, the *pre-state*, or *post-state* typically includes 50 to 70 assertions, while an operator usually has 2 to 6 preconditions or effects. In the absence of background knowledge for identifying the portion of the world state relevant to the planning operator, it

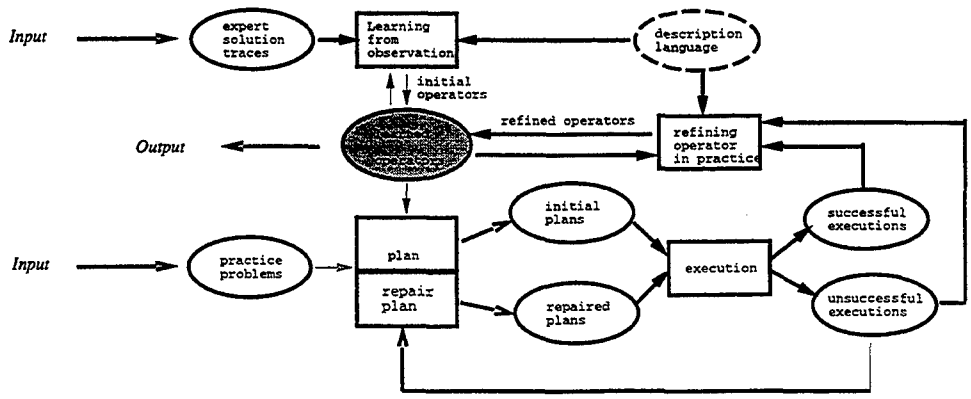


Figure 1: Overview of OBSERVER's learning architecture.

```
(Operator HOLD-WITH-VISE
 (preconds ((<hd> VISE) (<side> Side)
            (<machine> Machine) (<part> Part))
  (and (has-device <machine> <hd>)
       (not (has-burrs <part>))
       (is-clean <part>)
       (on-table <machine> <part>)
       (is-empty-holding-device <hd> <machine>)
       (is-available-part <part>)))
  (effects
   (del (on-table <machine> <part>))
   (del (is-available-part <part>))
   (del (is-empty-holding-device <hd> <machine>))
   (if (shape-of <part> CYLINDRICAL)
       (add (holding-weakly
              <machine> <hd> <part> <side>)))
   (if (shape-of <part> RECTANGULAR)
       (add (holding
              <machine> <hd> <part> <side>)))))
```

Figure 2: Operator HOLD-WITH-VISE from the process planning domain. This operator specifies the preconditions and effects of holding a part with a vise. This is operator OBSERVER learns after observation and practice.

is computationally expensive to find the maximally specific common generalization MSCG from the observed *pre-state*. In fact, Haussler (Haussler 1989) shows that finding an existential conjunctive concept consistent with a sequence of m examples over an instance space defined by n attributes is NP-complete. Furthermore, he shows that the size of MSCG can grow exponentially with the number of examples m. Although Haussler notes that heuristic methods for learning conjunctive concepts can be effective, the existing inductive algorithms (Vere 1980; Hayes-Roth and McDermott 1978; Watanabe and Rendell 1990) do not apply well to our operator learning problem. For example, Vere's counterfactual algorithm (Vere 1980) requires both positive and negative examples and is not incremental. Hayes-Roth's interference matching algorithm (Hayes-Roth and McDermott 1978) uses some complex heuristics to prune the space of all the matches between two examples, and thus prevent some operator preconditions from being learned. It also uses the one-to-one parameter binding assumption that does not hold for our operator preconditions (in general, two or more different variables can be bound to the same object). Watanabe and Rendell's X-search algorithm (Watanabe and Rendell 1990) prunes the search space by using connectivity constraint. However operator preconditions usually form only one connected component and thus preventing the heuristic from being effective. FOIL (Quinlan 1990), a greedy algorithm, does not apply well to our problem either, because FOIL requires both positive and negative training instances and it is not incremental.

OBSERVER uses an incremental algorithm to learn an operator's preconditions by building a general representation (**G-rep**) and a specific representation (**S-rep**) in a manner similar to the version spaces method (Mitchell 1978). The **S-rep** is a collection of literals that represents a specific boundary of the precondition expression being learned. The **S-rep** is updated both from from observations and during practice. The **G-rep** is a collection of literals that represents a general boundary of the precondition expression being learned. The **G-rep** is initialized to the empty set and is updated using negative examples OBSERVER generates during practice.

Learning operators from observation

During learning from observation, OBSERVER first initializes each operator using the first observation for the operator, then it refines each operator by learning the **S-rep** of operators preconditions and the operator effects.

Initializing the operators

Given the first observation of an operator, OBSERVER initializes the specific representation S-rep to the parameterized pre-state of the observation, and the effects of the operator to the parameterized delta-states. During parameterization, objects (except for *domain constants*) are generalized to typed variables. Domain constants are given to OBSERVER as part of the input, they are only generalized to a variable if OBSERVER notices a different constant in a later observation. The type for each variable in the operator is the most specific type in the type hierarchy for the corresponding object. See operator in Figure 6 learned from one observation in Figure 3 as an example. Note that

constants `width, rectangular, iron` etc are not generalized to variables. Also note that the operator learned from this one observation has some extraneous preconditions such as `(size-of <v1> height 2.75)`. They will later be generalize or removed with more observations and/or practice.

```
(operator hold-with-vise
  (preconds ((<v3> Drill) (<v2> Vise)
             (<v1> Part) (<v5> Spot-drill))
   (and (has-device <v3> <v2>)
        (is-available-table <v3> <v2>)
        (is-empty-holding-device <v2> <v3>)
        (holding-tool <v3> <v5>)
        (size-of <v1> width 2.75)
        (size-of <v1> height 4.25)
        (size-of <v1> length 5.5)
        (shape-of <v1> rectangular)
        (on-table <v3> <v1>)
        (hardness-of <v1> soft)
        (is-clean <v1>)
        (material-of <v1> copper)
        (is-available-part <v1>)
  (effects
   (add (holding <v3> <v2> <v1> side5))
   (del (on-table <v3> <v1>))
   (del (is-available-part <v1>))
   (del (is-empty-holding-device <v2> <v3>))))))
```

Figure 6: Learned operator `HOLD-WITH-VISE` when the observation in Figure 3 is given to OBSERVER. The preconditions shown in this figure is the S-rep of the preconditions of the operator `HOLD-WITH-VISE`.

Learning operators incrementally with new observations

After initializing the operators from the first observation, these initial operators are further generalized incrementally with more observations. OBSERVER refines the corresponding operator with each new observation as follows: first OBSERVER matches the effects of the operator against the delta-state of the new observation. The matching produces partial bindings for variables in the operator. Then OBSERVER uses the matching results to update the specific representation S-rep of the operator preconditions. And finally the effects of the operators are updated.

Learning variable bindings of the operator from observation

Learning the variable bindings can help reducing the ambiguity in finding the mapping between an operator and a new observation of the operator. OBSERVER learns bindings by matching the effects of the operator against the delta-state of the observation. In the presence of ambiguity during matching, this procedure chooses the mapping between the operator effects and the delta-states in the observation that leads to fewer conditional effects being learned. This implies that any effect that is unifiable with a unique element in the delta-state should be unified with it.

In addition to learning the bindings of the variables in the effects, OBSERVER generalizes a constant to a variable if this constant is unified with a different constant in the bindings. It also generalizes the type of a variable to the least common ancestor type of the type of the variable in the operator and the type of the corresponding object in the bindings.

For example, given the second observation of the operator `HOLD-WITH-VISE` shown in Figure 7, and the initial operator shown in Figure 6 that was learned from the first observation, OBSERVER matches the effects of the operator against the delta-state of the observation. `(holding <v3> <v2> <v1> side5)` is unified with `(holding milling-machine1 vise1 part1 side4)` to produce *bindings* = {part1/<v1>, vise1/<v2>, milling-machine1/<v3>, side4/side5}. The type of the variable <v3> is generalized to the least common type of milling-machine and drill, i.e. Machine, and the constant `side5` is generalized to a new variable <v4> whose type is `Side`.

Updating the S-rep from observations

Figure 10 shows the procedure **update_S_rep_from_obs** for updating the S-rep of the operator preconditions given an observation. OBSERVER updates the **S-rep** by removing from it those literals that are not present in the pre-state of the observation, and by generalizing *domain constants* to variables if different constants are used in the pre-state. Determining which preconditions in the S-rep are not present in the pre-state of the observation is non trivial when bindings for the variables are unknown. In fact, there are multiple generalizations under different mappings of variables to objects. OBSERVER's approach is to remove preconditions that are definitely not met in the pre-state (i.e. is not unifiable with any literal in the pre-state) under all the possible bindings for the variables in the operator, as long as they are consistent with the bindings returned by **learn_variable_bindings**.

How does OBSERVER know if a precondition is not unifiable with any literal in the pre-state? For each precondition *prec* in the **S-rep** of the operator, OBSERVER unifies *prec*, substituted with the bindings returned by **learn_variable_bindings**, with every literal in the pre-state of the observation. For every literal *lit* in the pre-state, let $\theta(\{l_1/p_1, l_2/p_2, ..., l_n/p_n\}) =$ **unify**(*lit*, (**substitute**(*p*, *bindings*)). We call *lit* a **potential-match** of *prec* if and only if *lit* and *prec* are unifiable with respect to bindings *bindings*. **find_potential_matches**(*prec, obs, bindings*) finds all the potential matches for a precondition *prec*. If *prec* does not have any potential matches, that is, *prec* can not be present in the pre-state no matter what variable bindings are given, then it is removed from the S-rep. This usually happens when no literal in the pre-state has the same predicate as the precondition *prec*. If *prec* has exactly one potential match, then *prec* is generalized through unification with its potential-match in the pre-state. Otherwise, *prec* has several potential matches, and OBSERVER handles this ambiguity by keeping *prec* as it is in the S-rep.

As an example, given the second observation of the operator `HOLD-WITH-VISE` shown in Figure 7, and the initial operator shown in Figure 6 that was learned from the first observation, and *bindings* = {`<v1>/part1, <v2>/vise1, <v3>/milling-machine1, side5/side4`}, learned by matching

op-name: hold-with-vise

Pre-state:

```
(has-device drill0 vise0)
(on-table drill0 part0)
(is-clean part0)
(is-empty-holding-device vise0 drill0)
(is-available-table drill0 vise0)
(holding-tool drill0 spot-drill0)
(is-available-part part0)
(hardness-of part0 hard)
(material-of part0 iron)
(size-of part0 width 2.75)
(size-of part0 height 4.25)
(size-of part0 length 5.5)
(shape-of part0 rectangular)
```

Delta-state:

adds:

```
(holding drill0 vise0 part0 side5)
```

dels:

```
(is-empty-holding-device vise0 drill0)
(on-table drill0 part0)
(is-available-part part0)
```

Figure 3: An observation of the state before and after the application of the operator HOLD-WITH-VISE. Many literals in the pre-states are irrelevant information for applying the operator HOLD-WITH-VISE.

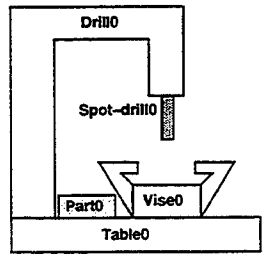


Figure 4: Pre-state of observation in Figure 3. Note that the part is on the table.

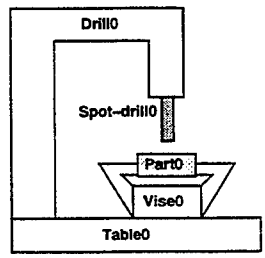


Figure 5: Post-state of observation in Figure 3. Note that the part is being held.

op-name: hold-with-vise

Pre-state:

```
(has-device milling-machine1 vise1)
(on-table milling-machine1 part1)
(is-available-table milling-machine1 vise1)
(is-empty-holding-device vise1 milling-machine1)
(is-available-tool-holder milling-machine1)
(is-available-part part1)
(is-clean part1)
(size-of part1 length 4)
(size-of part1 width 3)
(size-of part1 height 2.25)
(shape-of part1 rectangular)
(hardness-of part1 soft)
(material-of part1 bronze)
```

Delta-state:

adds:

```
(holding milling-machine1 vise1 part1 side4)
```

dels:

```
(is-empty-holding-device vise1 milling-machine1)
(on-table milling-machine1 part1)
(is-available-part part1)
```

Figure 7: The 2nd observation of the operator HOLD-WITH-VISE. The main differences with the first observation shown in Figure 3 are (1) the machine in use is a milling-machine instead of a drill, (2) there is no drill-tool being held by the machine, (3) the part is made of different material, and has different sizes.

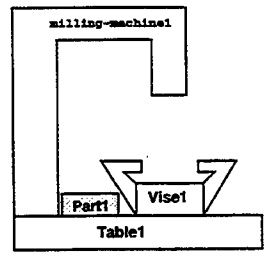


Figure 8: Pre-state of observation in Figure 7. The part is on the table.

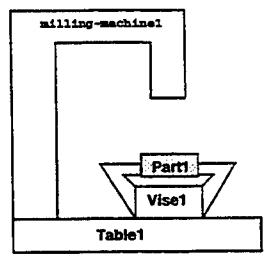


Figure 9: Post-state of observation in Figure 7. The part is being held by vise1.

procedure: update_S_rep_from_obs

Input: *op, obs, bindings*
Output: S-rep, the updated specific precondition representation

1. for each *prec* ∈ S-rep)
2. *potential-matches(prec)* ← **find_potential_matches**(*prec,pre-state(obs),bindings*)
3. if *potential-matches(prec)*=∅
4. then S-rep ← S-rep \{*prec*}
5. if *potential-matches(prec)* has 1 element
6. then S-rep ← (S-rep \{*prec*}) ∪**generalize**(*prec, bindings*)
7. if *potential-matches(prec)* has more than 1 element
8. then do not modify the S-rep

Figure 10: Updating the **S-rep**, a specific boundary of the operator preconditions, given an observation. This algorithm runs in polynomial time in terms of the number of literals in the pre-state or the preconditions.

```
(operator hold-with-vise
  (preconds ((<v3> Machine) (<v2> Vise)
             (<v1> Part) (<v4> Side))
   (and (has-device <v3> <v2>)
        (is-available-table <v3> <v2>)
        (is-empty-holding-device <v2> <v3>)
        (shape-of <v1> rectangular)
        (on-table <v3> <v1>)
        (is-clean <v1>)
        (hardness-of <v1> soft)
        (is-available-part <v1>)
  (effects
   (add (holding <v3> <v2> <v1> <v4>))
   (del (on-table <v3> <v1>))
   (del (is-available-part <v1>))
   (del (is-empty-holding-device <v2> <v3>))))
```

extraneous preconditions:

```
(holding-tool <v3> <v5>)
(size-of <v1> width <v6>)
(size-of <v1> height <v7>)
(size-of <v1> length <v8>)
(material-of <v1> <v10>)
```

Figure 11: Learned operator HOLD-WITH-VISE when the observations in Figures 3 and 7 are both given to OBSERVER. The preconditions listed here are the literals in the S-rep of the operator HOLD-WITH-VISE. Note that the type of variable <v3> has been generalized to Machine.

the effects with the delta-state, OBSERVER computes the *potential-matches* for each precondition in the specific representation of the operators giving *bindings*. For example, there are no *potential-matches* for `(holding-tool <v3> <v5>)` because none of the element in the pre-state has the predicate `holding-tool`, i.e. for every `i` in the pre-state **unify**(`i`, `(holding-tool <v3> <v5>)`) = NULL. Here are some examples of what **find_potential_matches** returns:

potential-match(`(holding-tool <v3> <v5>)`) = ∅;
potential-match(`(size-of <v1> width 2.75)`)
= {`(size-of part1 width 3)`};

Therefore, `(holding-tool <v3> <v5>)` is removed from S-rep. `(size-of <v1> width 2.75)` is generalized to `(size-of <v1> width <anything>)` by unifying it with `(size-of part1 width 3)`. Since `<anything>` is not constrained by any other preconditions in S-rep, OBSERVER learns that the width of a part can be any value, and therefore is irrelevant to the operator and is removed from the S-rep. Similarly, preconditions `(size-of <v1> height 2.75)`, `(size-of <v1> length 5.5)`, `(material-of <v1> copper)` are removed from the preconditions. The modified operator is shown in Figure 11, along with the extraneous preconditions that are removed from the initial operator.

Updating the effects from observations

Figure 12 describes the procedure **update_effects_from_obs** for updating the effects of the operator from an observation. OBSERVER first generalizes those effects that have unambiguous matches in the delta-state (steps 4-8). Steps 9–17 describes how OBSERVER learns conditional effects. For every effect of the operator, if it does not unify with any element in the delta-state of the observation, then it is learned as a conditional effect. The specific representation of the preconditions of this conditional effect is initialized to the **S-rep** of the preconditions of the operator (steps 9–12). Every element in the delta-state that does not unify with any effect is also learned as a conditional effect (steps 13–17), and its specific representation for the preconditions of this conditional effect is the parameterized pre-state of the current observation. OBSERVER does not update the effects that have ambiguous mappings with elements in the delta-state.

To illustrate how OBSERVER generalizes an effect, let's look at the learned operator HOLD-WITH-VISE as shown in Figure 6 learned from the first observation, and the second observation shown in Figure 7. Matching the effects of the operator against the delta-state of the observation generates the following bindings: {part1/<v1>, vise1/<v2>, milling-machine1/<v3>, side4/side5}. The effect `(add (holding <v3> <v2> <v1> side5))` is generalized to `(add (holding <v3> <v2> <v1> <v4>))` because the constant `side5` in the effect is substituted with a different constant `side4` in the delta-state. Other effects are kept as they are. Also since the mapping between the effects and the *delta-state* is unambiguous, no conditional effects are learned. The updated operator is shown in Figure 11.

The next example illustrates how conditional effects are learned. Given the third observation shown in Figure 13, and the operator that are learned from the first two observations as shown in Figure 11, OBSERVER matches the effects of the operator against the delta-states of the third observation and produces bindings {drill2/<v3>, part2/<v1>, vise2/<v2>}. Effects `(del (on-table <v3> <v1>))`, `(del (is-available-part <v1>))`, and `(del (is-empty-holding-device <v2> <v3>))` have unambiguous matches in the delta-states, i.e. `(del (on-table drill2 part2))`, `(del (is-available-part part2))`, and `(del (is-empty-holding-device vise2 drill2))`, respectively. Effect `(add (holding <v3> <v2> <v1> <v4>))` is not unifiable

procedure: update_effects_from_obs

Input: *bindings*, learned operator *op*, new observation *obs*.
Output: modified effects(op).

1. *effects* ← *Effects*(*op*).
2. *deltas* ← *delta-state*(*obs*).
3. for every $e \in$ **Effects**(*op*)
4. if there $\exists\ d \in$ *deltas* s.t. **is_unambiguous_pair**(*e,d*) then
5. $e' \leftarrow$ **generalize**(*e, bindings*).
6. replace *e* with e'.
6. if *e* is a conditional effect, update its conditional preconditions
7. *effects* ← *effects* \ {*e*}
8. *deltas* ← *deltas* \ {*d*}
9. for every $e \in$ *effects*
10. if $\forall\ d \in$ *deltas*, **unify**(*e, d1*) = NULL then ;;; learning a conditional effects
11. **conditional_effects**(*op*) ← **conditional_effects**(*op*) ∪ *e*
12. S-rep(e) ← S-rep(op)
13. for every $d \in$ *deltas*,
14. if $\forall$ *e1* $\in$ *effects*, **unify**(*e1, d*) = NULL then ;;; learning a conditional effects
15. *new-effect* = **parameterize**(*d*)
16. **conditional_effects**(*op*) ← **conditional_effects**(*op*) ∪ {*new-effect*}
17. *S* − *rep*(*new-effect*) ← **parameterize**(*pre-state*(*obs*))

Figure 12: Updating the effects of an operator based on the new observation. Every effect that has a unique match with the delta-state is generalized according to the match. Every effect that is not unifiable with any element in the delta-state is learned as a conditional effect. Every element in the delta-state that is not unifiable with any effect is also learned as a conditional effect.

with any literal in the delta-state, and therefore is learned as a conditional effect of the operator. Delta-state `(add (holding-weakly drill2 vise2 part2 side0))` is not unifiable with any effect of the operator, and therefore is parameterized and learned as a conditional effect. The resulting operator with conditional effect is shown in Figure 16.

```
(operator hold-with-vise
  (preconds ((<v3> Machine) (<v2> Vise)
             (<v1> Part) (<v4> Side))
   (and (has-device <v3> <v2>)
        (is-available-table <v3> <v2>)
        (is-empty-holding-device <v2> <v3>)
        (on-table <v3> <v1>)
        (is-clean <v1>)
        (hardness-of <v1> soft)
        (is-available-part <v1>)
  (effects (<v6> spot-drill))
   (if (and (size-of <v1> diameter 4.75)
            (size-of <v1> length 5.5)
            (shape-of <v1> cylindrical)
            (hardness-of <v1> soft)
            (material-of <v1> bronze)
            (holding-tool <v3> <v6>))
       (add (holding-weakly <v3> <v2> <v1> side0)))
   (if (and (shape-of <v1> rectangular)
            (has-device <v3> <v2>)
            (is-available-table <v3> <v2>)
            (is-empty-holding-device <v2> <v3>)
            (on-table <v3> <v1>)
            (is-clean <v1>)
            (is-available-part <v1>)
       (add (holding <v3> <v2> <v1> <v4>)))
   (del (on-table <v3> <v1>))
   (del (is-available-part <v1>))
   (del (is-empty-holding-device <v2> <v3>)))))
```

extraneous preconditions:

```
(holding-tool <v3> <v5>)
(size-of <v1> width <v6>)
(size-of <v1> height <v7>)
(size-of <v1> length <v8>)
(material-of <v1> <v10>)
(shape-of <v1> <v11>)
```

Figure 16: Operator HOLD-WITH-VISE after learning from observations shown in Figures 3, 7, 13. Note the conditional effects and conditional preconditions.

Planning while learning

The operators learned from observation can be incomplete and incorrect in several ways. The planning system must be able to plan using these operators during practice. This challenges planning in the following ways:

1. Classical planners presume a correct domain model. In our learning system however, the newly acquired operators are possibly incomplete and incorrect. How can the planner generate plans to solve practice problems?
2. Because of incomplete and incorrect operators used during practice, the plans generated by the planner may be incorrect, which in turn may lead to execution failures. Thus plan repair upon execution failure is necessary. How can the planner effectively repair the incorrect plan using incomplete and incorrect operators?
3. How should planning and execution be interleaved so that the system can solve practice problems effectively

Pre-state:

```
(has-device drill2 vise2)
(on-table drill2 part2)
(is-available-table drill2 vise2)
(is-clean part2)
(is-empty-holding-device vise2 drill2)
(holding-tool drill2 spot-drill2)
(is-available-part part2)
(size-of part2 diameter 4.75)
(size-of part2 length 5.5)
(material-of part2 bronze)
(hardness-of part2 soft)
(shape-of part2 cylindrical)
```

Delta-state:

adds:

```
(holding-weakly drill2 vise2 part2 side0)
```

dels:

```
(is-empty-holding-device vise2 drill2)
(on-table drill2 part2)
(is-available-part part2)
```

Figure 13: The 3rd observation of the operator HOLD-WITH-VISE. Note that the effect has "holding-weakly" instead of "holding" as in previous observations. OBSERVER thus learns conditional_effects.

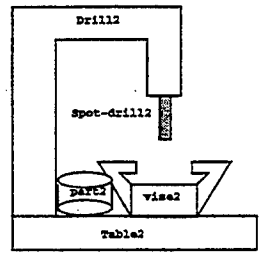


Figure 14: Pre-state of observation in Figure 13

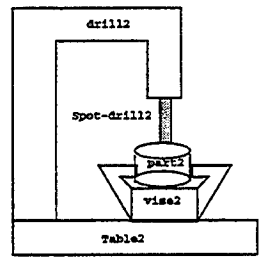


Figure 15: Post-state of observation in Figure 13

and concurrently generate learning opportunities for refining operators using incomplete and incorrect domain knowledge?

Our approach to address the above issues is to use the **G-rep** of the operator preconditions for planning. The individual plans generated for achieving some top-level goals achieve the preconditions in the **G-rep** of each operator, but do not require achieving preconditions in the **S-rep** of the operators. This has the advantage of generating an initial plan quickly and of generating opportunities for operator refinement. While executing an operator in the environment, there are the following two possible outcomes, generating negative or positive training instances, respectively, for refining operators.

- The first outcome is that the state does not change after executing *op*. In this case, we say that *op* is executed *unsuccessfully*.

  An operator may execute unsuccessfully because OBSERVER achieves the preconditions in the **G-rep** of each operator during planning without necessarily achieving all the preconditions in the **S-rep**. This introduces the possibility of incomplete or incorrect plans in the sense that a real precondition may be unsatisfied. Unsuccessful executions form the negative examples that OBSERVER uses for refining operators as discussed in (Wang 1995). Upon each unsuccessful execution, OBSERVER updates the **G-rep** by learning *critical preconditions* using *near misses* of negative examples. OBSERVER also attempts to generate a *repaired-plan*. If such a plan is found, OBSERVER continues execution using the repaired plan; otherwise, OBSERVER removes the failed operator and continues execution with the remaining plan.
- The other possible outcome is that the state changes after executing *op*. In this case, we say that *op* is executed *successfully*.

  Successful executions form the positive examples that OBSERVER uses for refining operator as described in (Wang 1995). Note that a successful execution may still have incorrect predictions of how the state changes due to the possibility of incomplete operator effects. Upon each successful execution, OBSERVER updates the **S-rep** by removing from it the preconditions that are not satisfied in *current-state*, and updates the effects of *op* if missing effects are observed in the *delta-state* of the execution. OBSERVER then updates the current state and continues execution with the remaining plan.

Detailed descriptions of the algorithms for planning with incomplete and incorrect operators and plan repair are described in (Wang 1996b).

Refining operators during practice

This section describes how OBSERVER refines operators using the successful and unsuccessful executions generated during practice. This involves updating the **S-rep** of operator preconditions, learning the **G-rep** of the operator preconditions, and updating the effects, including conditional and conditional preconditions.

Updating the S-rep of operator preconditions

During observations, OBSERVER cannot observe the bindings for the variables in the operators. However, during practice, the bindings are determined during planning. Therefore, when refining operators during practice, OBSERVER no longer needs to learn the bindings through matching the effects with the delta-state. This eliminates the ambiguity in the matching process that is necessary for learning from observations. Therefore, removing extrane-

ous preconditions from the S-rep of the operator preconditions during practice is relatively straightforward

OBSERVER may also learn negated preconditions upon unsuccessful executions: if all the preconditions in the **S-rep** are satisfied in the pre-state, and the operator fails to apply, then the failure is due to the existence of negated preconditions. The potentially negated preconditions are those literals that are true in the pre-state of the execution but are not specified in the **S-rep** of the operator preconditions, nor in the extraneous preconditions previously removed from the operator. OBSERVER conjectures the negations of these literals as negated preconditions, and adds them to the **S-rep**. Some of the negated preconditions thus added are extraneous preconditions, and are removed from the **S-rep** accordingly using the algorithm for updating the **S-rep**.

Learning the G-rep of operator preconditions

The **G-rep** of an operator is learned incrementally as OBSERVER confirms elements of the **S-rep** to be true preconditions of the operator. The **G-rep** is only updated when the negative example is a *near miss*. A near miss is a negative example where only one literal in the **S-rep** is not satisfied in the state. *Critical-preconditions* are learned from near miss negative example. A *critical-precondition* p of an operator *op* is a precondition such that there exists a pair of states s_1 and s_2 that OBSERVER encountered during practice such that: (1) p is satisfied in s_1 but not in s_2; (2) everything else in s_1 and s_2 are identical; (3) *op* is applicable in s_1 but not in s_2. The **G-rep** is specialized by adding to it the *critical-preconditions*. More details for learning the **G-rep** can be found in (Wang 1996a).

Refining operator effects

The procedure for refining operator effects during practice is very similar to the procedure **update_effects_from_obs** described in Figure 12 for updating the effects based on observations. The main difference is that when learning from executions, there is no ambiguity in variable bindings, and therefore learning is more efficient.

Empirical results

The learning algorithm described in this paper has been fully implemented on top of the PRODIGY4.0 architecture. In this section, we present empirical results to demonstrate that OBSERVER learns operators in a process planning well enough to solve problems as effectively as using human-expert coded operators in terms of the total number of solvable test problems. The human expert coded operators are encoded in a period of 6 months by an AI expert through interactions with domain experts. We also present empirical results to show that both learning from observation and learning from practice contribute significantly to the learning process.

Experiment set up

Our experiments include the following three phases:

1. *Learning operators from observation phase,* where OBSERVER learns operators from expert solutions traces.
2. *Refining operators during practice phase,* where OBSERVER refines operators during practice.
3. *Testing phase,* where the learned operators are compared against human expert-coded operators in terms of the number of solvable problems in a set of randomly generated test problems.

OBSERVER is implemented in the context of PRODIGY4.0(Carbonell *et al.* 1992). PRODIGY4.0 is a non-linear operator-based planner. There are many choice points for plannings: choice points for choosing a goal, for choosing an operator to achieve the goal, etc. In the absence of control knowledge, PRODIGY4.0's default strategy is to always choose the first alternative. This pre-determined order is arbitrary but introduces a systematic bias in search time that we want to factor out. In order for the planner not to be biased by a pre-determined ordering of operators, we introduced a random selection mechanism at each choice point in our experiments, and ran the planner multiple times to measure average performance.

The effectiveness of the overall learning system

OBSERVER first learns 33 new operators from observed solutions of 100 problems. Then it is given a set of 180 problems to refine operators from practice. OBSERVER's performance on a test set of 34 problems that are *different* from the learning problems is measured by the total number of solvable test problems using the learned operators. We measure the performance 5 times and computes the averages at the end of observationk, and after every 15 practice problems.

Figure 17 compares the total number of problems solved using learned operators and expert human-codes operators. The spikes of the curve are due to the fact that the planner makes random choices at each decision point. The curve indicates that after observation and practice, the total number of solvable problems are the same using OBSERVER-learned operators or expert human-coded operators.

Previous results given in (Wang 1995) showed that the learned operators are as effective as the expert human-coded operators in problem solving according to two other criteria: (*i*) the average amount of planning time to solve each problem, and (*ii*) the average length of the plans generated to solve each problem.

The role of observation

To evaluate the role of observation, OBSERVER is run multiple times. During each run, OBSERVER is given the same problems during practice, but *different number* of initial observation problems that are randomly drawn from a fixed set of observation problems. We measure the total number of solvable test problems after observation and practice, given *different number* of initial observation problems.

Figure 18 shows how the total number of problems that solvable after the same set of practice problems is given, but

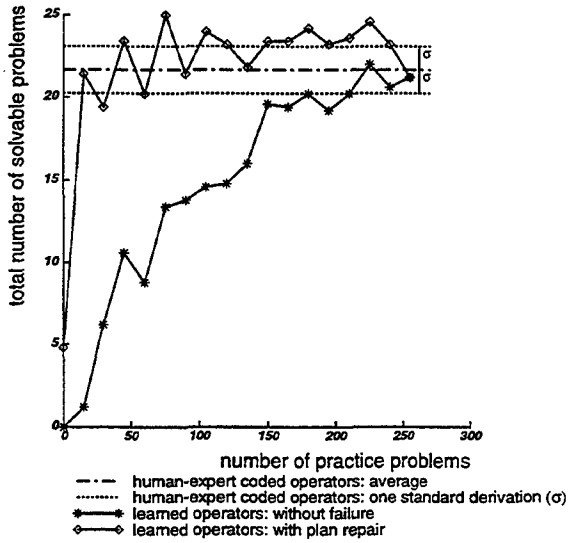


Figure 17: Total number of solvable test problems as a function of the number of practice problems in training.

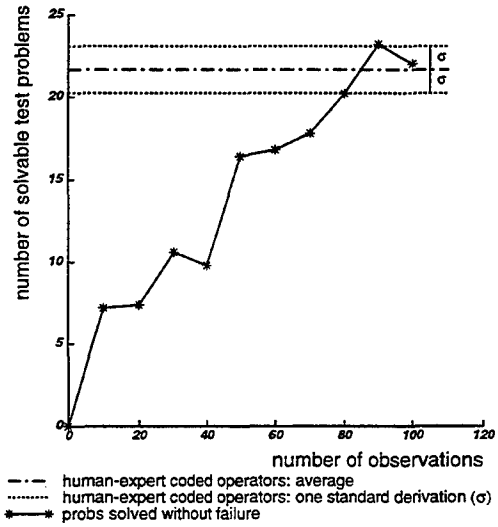


Figure 18: Total number of problems solvable with the same set of practice problems, but with different amount of observation before practice, as a function of the number of initial observation problems in training. We see that after 80 observation training problems in the process planning domain, more observation problems do not significantly increase the solvability. This indicates the right point to stop learning from observation, and to start practicing.

expert solutions traces of different number of observation problems are given to OBSERVER initially.

The role of practice

In order to demonstrate the role of practice, we show that OBSERVER is able to learn operators better and faster if it practices using the planning and plan repair algorithms described in this paper, after initial operator acquisition by observation, than if it is given only the observations of expert solutions of the total sequence of problems (both observation sequence and practice problems presented as additional observations).

During evaluation, OBSERVER is first given expert solutions traces for a set of observation problems. OBSERVER thus learns a set of operators from observation. These operators may be incomplete and incorrect in a number of ways. We then compare how OBSERVER refines these learned operators in the following two scenarios in terms of total number of solvable test problems.

scenario 1: OBSERVER is given a set of problems to practice. During practice, OBSERVER uses the planning and plan repair algorithms described in this paper to generate plans to solve the practice problems, and refines the initial incorrect and incomplete operators using both successful and unsuccessful executions.

scenario 2: OBSERVER is given the same set of problems. However, OBSERVER is only allowed to refine the operators based on the expert solutions for these problems, and is not allowed to practice searching for its own solutions.

The evaluation of solvable test problems includes the following two cases:

1. The total number of solvable test problems, including problems solved where executions failures occur but are subsequently repaired using our plan repair algorithm.
2. The total number of solvable test problems without execution failure, and therefore without requiring plan repair.

Figure 19 illustrates the comparison of the learned operators that are refined through practice (scenario 1), with the operators that are refined based on observation only (scenario 2), in terms of the total number of solvable problems in the test set *without execution failure*. This figure shows that:

- With practice where OBSERVER uses the planning and plan repair algorithms in this paper, the total number of solvable test problems without execution failure increases steadily as the number of training problems increases.
- If OBSERVER is only given the expert solution traces of the same problems, OBSERVER can not solve any test problem without failure.

Figure 20 illustrates the comparison of the operators that are further refined through practice (scenario 1) with the operators that are learned and refined based on observations (scenario 2) only, in terms of the total number of solvable test problems *without execution failure, and therefore without requiring plan repair.* This figure shows that:

- In both scenarios, the total number of solvable test problems increases as the number of training problems increases, whether OBSERVER uses the training problems for practice or as observations;
- The total number of solvable problems increases much faster in scenario 1, where OBSERVER practices using our planning and plan repair algorithms, than in scenario 2, where OBSERVER is only given the expert solution traces of the same problems.

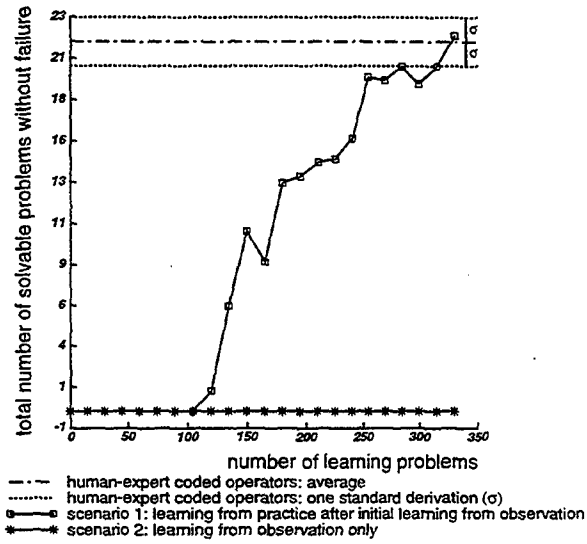


Figure 19: *Total number of solvable test problems without execution failure during testing, as a function of the number of practice problems in training.* This figure compares the operators learned from scenario 1 (OBSERVER practices) with operators learned in scenario 2 (OBSERVER does not practice, and learns from observation only). In the comparison, OBSERVER first learns a set of operators from observation in both scenarios. Then OBSERVER refines operators from practice in scenario 1, and OBSERVER learns from observations only in scenario 2.

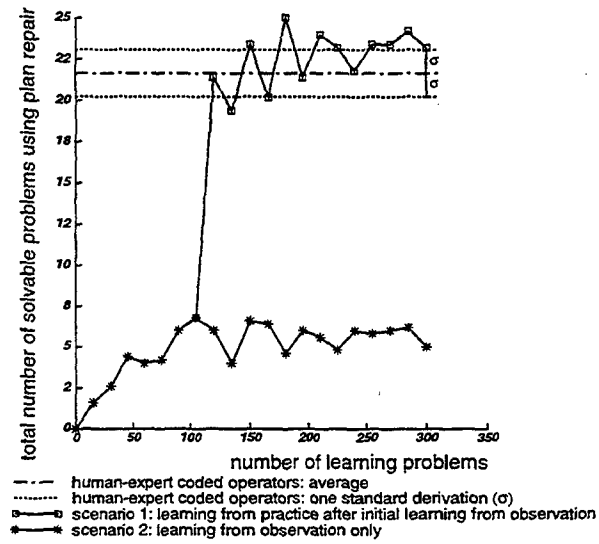


Figure 20: *Total number of solvable test problems allowing execution failure and plan repair during testing, as a function of the number of practice problems in training.* This figure compares the operators learned from scenario 1 (OBSERVER practices) with operators learned in scenario 2 (OBSERVER does not practice, and learns from observation only). In the comparison, OBSERVER first learns a set of operators from observation in both scenarios. Then OBSERVER refines operators from practice in scenario 1, and OBSERVER learns from observations only in scenario 2.

The above results can be explained by the fact that OBSERVER uses the **G-rep** for planning, and that the correctness of the plans generated depends on how well OBSERVER has learned the **G-rep**. After practice, OBSERVER is able to update the **G-rep** of the operator preconditions based on its own executions (scenario 1). However, without practice, OBSERVER can never update the **G-rep** (scenario 2) — observations of expert solutions do not have negative examples, and the **G-rep** can only be made more specific using negative training examples.

Comparing Figure 20 and Figure 19, we see that in scenario 1 where OBSERVER practice, although the number of solvable test problems given plan repair during testing increases faster than without plan repair, the total number of solvable test problems in both cases for the total number of solvable test problems are comparable at the end of learning. However, in scenario 2 where OBSERVER only learns from observation, more test problems can be solved where plan repair is used when solving test problems than if plan repair is not used when solving test problems even at the end of learning. This indicates that plan repair plays an important role when solving problems with incomplete and incorrect operators.

Related work

LEX(Mitchell *et al.* 1983) is a system that learns heuristic problem-solving strategies through experience in the domain of symbolic integration. It starts with exact preconditions for the operators, but lacks preconditions (control rules) for when should the operators be applied. The control knowledge is learned from experience as additional preconditions. OBSERVER is learning the preconditions of the basic domain operators, not search heuristics, and OBSERVER also learns from observations of expert solution traces.

LIVE (Shen 1994) is a system that learns and discovers from the environment. It integrates action, exploration, experimentation, learning, and problem solving. However it does not learn from observing others. LIVE has only been demonstrated in simplistic domains (e.g. the Little Prince World with only about 15 possible states), and does not scale up to large complex domains OBSERVER has been applied to. LIVE also avoids the complexity of planning with incomplete and incorrect operators by using a set of *domain-dependent search heuristics* during planning. These heuristics are part of the input to LIVE. OBSERVER differs in that it deals explicitly with imperfect operators without relying on any human-coded, domain-dependent heuristics.

EXPO (Gil 1992) is a learning-by-experimentation module for refining incomplete domain knowledge. Learning is triggered when plan execution monitoring detects a divergence between internal expectations and external observations. The initial knowledge given to EXPO and OBSERVER is different in that EXPO is given operators with over-general preconditions and incomplete effects, whereas OBSERVER does not have any knowledge about the preconditions or the effects of the operators. OBSERVER learns

from observations of expert solution traces that EXPO does not need to address, since EXPO already starts with workable, if incomplete operators. EXPO designs experiments to determine which preconditions and effects are missing from the operators. During execution failure, EXPO resets the problem to its original initial state. This differs from OBSERVER in that planning, execution, and plan repair form a closed loop in OBSERVER.

Benson's system (Benson 1995) learns actions models from its own experience and from its observation of a domain expert. Benson's work uses an action model formalism that is suited for reactive agents, which is different from STRIPS operators. It does not have a complicated planning and plan repair mechanism, and relies on an external teacher when an impasse is reached.

In DesJardins' work (desJardins 1994), users enter partially specified operators that reflect a rough description of how a subgoal may be solved. These operators usually have predicates with underspecified arguments. The learning system inductively fills these in by generalizing from feedback from evaluation models and from the user's planning. OBSERVER differs in that it starts with minimal initial knowledge, and must learn the preconditions themselves as well as the arguments in each precondition.

Future work and conclusions

We have presented a novel multistrategy learning method to automatically learn planning operators by observation and practice. Unlike previous approaches, our approach does not require a considerable amount of direct interactions with domain experts, or initial approximate planning operators, or strong background knowledge. We have shown that our system learns operators in a large process planning domain well enough to solve problems as effectively as human-expert coded operators, and that both strategies, i.e., learning operators from observation, and refining operators during practice, contribute significantly to the learning process.

Future work involves extending the learning algorithms to handle uncertainty in the domains. There are several sources of uncertainties that are possible in a domain. The first is perception noise in which observation of states may be incorrect — some properties may be missing in the observation, while some maybe erroneously observed, and others incorrectly observed. The second form of uncertainties is in the operators — operators may have effects that occur only with certain probability in a manner not fully predictable by the planner. The third type of uncertainty comes from external exogenous events that change the state.

In order to develop a learning system to handle uncertainty in the domain, one must first study what types of noise is present. Assuming no noise in the domain enables OBSERVER to converge fast in complex domains (e.g. two to three hundreds problems suffices for the learning to converge in the complex process planning domain). However, our framework for learning operators from observation and practice is still valid for acquiring operators in the presence of noise. The learning algorithm can be extended to handle noise by, for example, maintaining an occurrence count of each literal in the pre-state of the observations or practice, and only removing preconditions from the specific representation of the corresponding operator if the occurrence count is lower than a pre-specified frequency (the frequency is a function of the maximal level of noise the system will tolerate), and only adding a precondition to the general representation if the occurrence count is higher than a pre-specified frequency. But the learning rate will be much slower if there is noise in the domain. Some recent work has started research along this direction (Oates and Cohen 1996; Tae and Cook 1996). However, these approaches have only been applied to simple domains.

Our work makes two major contributions. First, it demonstrates that inductive learning techniques can be applied effectively to the problem of learning complex action descriptions - an open question for the machine learning community. Second, our work on automatic learning of planning operators provides a good starting point for fielding planning systems as it addresses a key bottleneck for this endeavor, namely, the knowledge acquisition problem.

References

Y. Anzai and H. A. Simon. The theory of learning by doing. *Psychological Review*, 86:124–140, 1979.

S. Benson. Inductive learning of reactive action models. In *Proceedings of 12th International Conference on Machine Learning*, Tahoe City, CA, July 1995.

J. H. Boose and B. R. Gaines. Knowledge acquisition for knowledge-based systems: Notes on the state-of-the-art. *Machine Learning*, 4:377–394, 1989.

Jaime G. Carbonell, and the PRODIGY Research Group: Jim Blythe, Oren Etzioni, Yolanda Gil, Robert Joseph, Dan Kahn, Craig Knoblock, Steven Minton, Alicia Pérez, (editor), Scott Reilly, Manuela Veloso, and Xuemei Wang. PRODIGY4.0: The manual and tutorial. Technical Report CMU-CS-92-150, School of Computer Science, Carnegie Mellon University, June 1992.

D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32:333–378, 1987.

S. Chien, R. Jr Hill., X. Wang, T. Estlin, K. Fayyad, and H. Mortensen. Why real-world planning is difficult: A tale of two applications. In M. Ghallab, editor, *Advances in Planning*. IOS Press, 1995.

S. Chien. Towards an intelligent planning knowledge base development environment. In *AAAI-94 Fall Symposium Series: Planning and Learning: On to Real Applications*, New Orleans, LA, 1994.

S. Chien. Static and completion analysis for planning knowledge base development and verification. In *Proceedings of the Third International Conference on AI Planning Systems*, Edinborgh, Scotland, 1996.

M. desJardins. Knowledge development methods for planning systems. In *AAAI-94 Fall Symposium Series: Planning and Learning: On to Real Applications*, New Orleans, LA, 1994.

R. E. Fikes and N. J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3,4):189–208, 1971.

Yolanda Gil. A specification of process planning for PRODIGY. Technical Report CMU-CS-91-179, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, August 1991.

Y. Gil. *Acquiring Domain Knowledge for Planning by Experimentation*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1992.

D. Haussler. Learning conjunctive concepts in structural domains. *Machine Learning*, 4:7–40, 1989.

F. Hayes-Roth and J. McDermott. An interference matching technique for inducing abstractions. In *CACM*, volume 26, pages 401–410, 1978.

D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 1991.

T. Mitchell, P. Utgoff, and R. Banerji. Learning by experimentation: Acquiring and refining problem-solving heuristic. In *Machine Learning, An Artificial Intelligence Approach, Volume I.* Tioga Press, Palo Alto, CA, 1983.

T. Mitchell. *Version Spaces: An Approach to Concept Learning*. PhD thesis, Stanford University,, 1978.

T. Oates and P. Cohen. Searching for planning operators with context-dependent and probabilistic effects. In *Proceedings of the 13th National Conference on Artificial Intelligence*, Portland, Oregon, August 1996. AAAI Press/The MIT Press.

J.S. Penberthy and D. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of KR-92*, 1992.

R. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

W. Shen. *Autonomous Learning from the Environment*. Computer Science Press, W.H. Freeman and Company, 1994.

K. Tae and D. Cook. Experimental knowledge-based acquisition for planning. In *Proceedings of 13th International Conference on Machine Learning*, Bari, Italy, July 1996.

Manuela Veloso and Peter Stone. Flecs: Planning with a flexible commitment strategy. *Journal of Artificial Intelligence Research*, 3, 1995.

M. Veloso, J.G. Carbonell, M. A. Pérez, D. Borrajo, E. Fink, and J. Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1), January 1995.

S. A. Vere. Multilevel counterfactuals for generalizations of relational concepts and productions. *Artificial Intelligence*, 14:139–164, 1980.

X. Wang. Learning by observation and practice: An incremental approach for planning operator acquisition. In *Proceedings of 12th International Conference on Machine Learning*, Tahoe City, CA, July 1995.

X. Wang. *Learning Planning Operators by Observation and Practice*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1996.

X. Wang. Planning while learning operators. In *Proceedings of the Third International Conference on AI Planning Systems*, Edinborgh, Scotland, 1996.

L. Watanabe and L. Rendell. Effective generalization of relational descriptions. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA, July 1990. AAAI Press/The MIT Press.

On-line Metalearning in Changing Contexts: METAL(B) and METAL(IB)

Gerhard Widmer
Department of Medical Cybernetics and Artificial Intelligence,
University of Vienna, and
Austrian Research Institute for Artificial Intelligence,
Schottengasse 3, A-1010 Vienna, Austria
gerhard@ai.univie.ac.at

Abstract

Many real-world concepts are heavily context-dependent. Changes in context can produce more or less radical changes in the associated concepts. On-line concept learning in such domains requires the ability to recognize and adapt to such changes.

This paper concentrates on a class of learning tasks where the domain provides explicit *clues* as to the current context (e.g., attributes with characteristic values). A general two-level learning model is presented that effectively adjusts to changing contexts by trying to detect (via 'meta-learning') contextual clues and using this information to focus the learning process. Context learning and detection occur *during* regular on-line learning, without separate training phases for context recognition.

Two operational systems based on this model are presented that differ in the underlying learning algorithm and in the way they use contextual information: METAL(B) combines meta-learning with a Bayesian classifier, while METAL(IB) is based on an instance-based learning algorithm. Experiments with synthetic domains as well as a 'real-world' problem show that the algorithms are robust in a variety of dimensions, and that meta-learning produces substantial improvement over simple object-level learning in situations with changing contexts.

Motivation

The fact that concepts in the real world are not eternally fixed entities or structures, but can have a different appearance or definition or meaning in different contexts has only gradually been recognized as a relevant problem in concept learning. Michalski (1987) was one of the first to formulate it; he suggested a specialized *two-tiered representation* formalism to represent different aspects of context-dependent concepts (see also Bergadano et al., 1992). Recently, context dependence has been recognized as a problem in a number of practical machine learning projects (e.g., Katz et al., 1990; Turney, 1993; Turney & Halasz, 1993; Watrous, 1993; Watrous & Towell, 1995). There, various techniques for context normalization etc. were developed. All of these methods either assume that contextual attributes are explicitly identified by the user, or require separate pre-training phases on special data sets that are cleanly separated according to context.

We are studying the effects of context dependence and changing contexts in the framework of *incremental* (or *on-line*) learning, and we are interested in learners that can adapt to different contexts without explicit help from a teacher. The scenario is as follows: assume that a learner is learning on-line from a stream of incoming (labeled) examples. Assume further that the concepts of interest depend on some (maybe hidden) *context*, and that changes in this context can induce corresponding changes in the target concepts. As a simple example, consider weather prediction rules, which may vary drastically with the change of seasons. The visible effects of such changes are increased prediction error rates.

The development of on-line learners that can cope with concept drift and changing contexts has been the subject of recent work on the FLORA family of algorithms (Widmer & Kubat, 1993, 1996; Widmer, 1994). The basic strategy in the FLORA algorithms is to continually monitor the success of on-line prediction and to make educated guesses at the occurrence of context changes and corresponding concept changes. There is no explicit representation of contexts.

But maybe one can do better. In some domains, the data may in fact contain explicit *clues* that would allow one to identify the current context, if one knew what these clues are. Technically, such clues would be attributes or combinations of attributes whose values are characteristic of the current context; more or less systematic changes in their values might then indicate a context change.

As a simple example, consider the license plates attached to vehicles in a particular country. An agent crossing the border between, say, Austria and Germany might notice that all of a sudden the license plates look different, in a systematic way, and that might lead it to suspect that it is now in a different environment where some of the rules it had learned

before may not be valid any more. Many other examples of such *contextual clues* come to mind (climate or season in weather prediction, environmental conditions like exterior temperature in technical process control tasks, lighting conditions or background color in automatic vision, characteristics of particular rooms in in-door robotic tasks, speaker nationality and sex in speech processing, etc.). In the following, we will refer to such context-defining attributes as *contextual clues*.

In this paper, we describe a general two-level learning model, and its realization in two specific systems named METAL(B) and METAL(IB), that can *learn* to *detect* such contextual clues, and can react accordingly when a change in context is suspected. The model consists of a *base level* learner that performs the regular on-line learning and classification task, and a *meta-learner* that tries to identify attributes and features that might provide contextual clues. Context learning and detection occur *during* regular on-line learning, without separate training phases for context recognition. Perceived context changes are used to focus the on-line learner specifically on information relevant to the current context. The result is faster adaptation to changed concept definitions, and generally an increase in predictive accuracy in dynamically changing domains. At the moment, both object-level and meta-level learning are restricted to nominal (discrete) attributes, but extensions of the model to numeric domains seem rather straightforward.

Preliminaries: Bayesian Classifiers

For the moment, let us assume that our basic incremental induction algorithm is a *simple* (or *naive*) *Bayesian classifier* (as is indeed the case in the first of our algorithms to be presented below, METAL(B)). That will make it easier to explain the meta-level learning strategy, which also has a distinct Bayesian flavor.

A *simple Bayesian classifier* is a probabilistic classification scheme that uses Bayes' theorem to determine the probability that an instance belongs to a particular class, given the instance's description. In the following, we assume that examples are described in terms of (discrete) *attributes* a_i; we will use the term *feature* for a specific attribute-value combination, notated as $a_i : v_{ij}$. Examples are assumed to belong to mutually exclusive classes c_i. Bayes' theorem defines the posterior probability that some new instance I belongs to class c_i as

$$p(c_i|I) = \frac{p(c_i)p(I|c_i)}{p(I)}$$

where $p(c_i)$ is the prior probability of class c_i and $p(I|c_i)$ is the probability of an instance like I, given class c_i. Assuming that I is a conjunction of attribute values v_j, the above formula can be rewritten as

$$p(c_i|\bigwedge v_j) = \frac{p(c_i)p(\bigwedge v_j|c_i)}{\sum_k p(\bigwedge v_j|c_k)p(c_k)}$$

To make this formula operational, one usually assumes that the attributes are independent, so that $p(\bigwedge v_j|c_k)$ can be computed as the product of the $p(v_j|c_k)$.

Incremental induction of a Bayesian classifier is straightforward. One maintains a number of counters, from which the prior and conditional probabilities can be estimated: a count N of the total number of instances encountered so far, a table C_i that keeps track of the relative frequency of class c_i observed so far; a table AV_{ij} that records the number of examples with attribute value $a_i = v_{ij}$, and a table AVC_{ijk} that records the number of examples with $a_i = v_{ij}$ belonging to class c_k. Learning then simply consists in updating these counters after processing each instance. The algorithm is simple, naturally incremental, and robust in the face of noise. Its major weakness is that we must assume independence of the attributes, which severely limits the class of learnable concepts.

In the following, we take this to be our basic incremental learning algorithm, with one important modification: our learner maintains a *window* of fixed length. As new examples are added to the window, the oldest ones are deleted from it if the window size is exceeded. This is to ameliorate the problem that very old instances pertaining to an outdated context may prevent the learner from effectively adjusting to new hypotheses. The window size is a user-settable parameter, but it remains fixed during the entire learning process. The tables C_i, AV_{ij}, and AVC_{ijk} are always updated with respect to the examples in the current window.

Meta-Learning: Learning to Recognize Contextual Clues

When the underlying target concept drifts or changes due to a changed context, the Bayesian classifier (indeed, any induction algorithm that bases its hypotheses on the contents of the window) will eventually adjust to the new concept, if the new context is stable for a sufficiently long period. The smaller the window, the faster the adjustment, as old, contradictory examples will be forgotten more quickly. However, in domains that provide explicit context clues, one would expect more: the learner should learn to recognize such clues and react in some appropriate way when they signal a potential context change. To operationalize this goal, we first need to define the central notions.

Definitions

What are contextual clues? Turney (1993) was one of the first to explicitly acknowledge the problem of context in learning and to try to give a formal definition of contextual and context-sensitive features. Eventually, however, he relied on the user to explicitly identify contextual features beforehand. His particular approach was motivated by batch learning problems where the *testing examples* (i.e., those to which the learned concepts would eventually be applied) might be governed

Table 1: Tables maintained for identifying predictive and contextual attributes.

Table	Counts occurrences/rel.frequency of	Computed over	Used at
C_i	# examples in class c_i	current window	base-level
AV_{ij}	# examples with $a_i = v_{ij}$	current window	base-level
AVC_{ijk}	# examples with $a_i = v_{ij}$ in class c_k	current window	base-level
$\hat{C}_{ij}$	# examples in meta-class $\hat{c}_{ij}$	entire history	meta-level
$\hat{AV}_{ij}$	# examples with $a_i = v_{ij}$	entire history	meta-level
$\hat{AVC}_{ijkl}$	# examples with $a_i = v_{ij}$ in meta-class $\hat{c}_{kl}$	entire history	meta-level

by a different context than the training examples from which the concepts were learned. The contextual features were then used for different kinds of normalization at prediction time. In contrast, we present a method to automatically *detect* contextual attributes in an on-line learning setting and to utilize this information *during learning*.

Our operational definition of contextual attributes, i.e., attributes that provide contextual clues, is based on the notion of *predictive features*. Intuitively speaking, an attribute is predictive if there is a certain correlation between the distribution of its values and the observed class distribution. This is formalized in the following two definitions:

Definition 1 *(Predictive features).* A feature (attribute-value combination) $a_i : v_{ij}$ is *predictive* if $p(c_k \mid a_i = v_{ij})$ is significantly different from $p(c_k)$ for some class c_k.

Definition 2 *(Predictive attributes).* An attribute a_i is *predictive* if one of its values v_{ij} (i.e., some feature $a_i : v_{ij}$) is predictive.

The most obvious kind of contextual clue one could imagine is that one or more attributes would have constant values during a certain context (regardless of an instance's class). Think, for instance, of the color of the walls in a particular room. This cannot be expected in general. We will rely on a more abstract and powerful notion: a feature is considered a contextual clue if there is a strong correlation between its temporal distribution of values and the times when certain other attributes are predictive. Intuitively, a contextual attribute is one that could be used to predict which attributes are predictive at any point in time.[1] This notion is formalized in definitions 3 and 4:

Definition 3 *(Contextual features).* A feature $a_i : v_{ij}$ is *contextual* if it is predictive of predictive features, i.e., if $p(a_k : v_{kl}$ *is predictive* $\mid a_i = v_{ij})$ is significantly different from $p(a_k : v_{kl}$ *is predictive*$)$ for some feature $a_k : v_{kl}$.

[1]Indeed, contextual attributes will be used for this purpose in the algorithm METAL(IB).

Definition 4 *(Contextual attributes).* An attribute a_i is *contextual* if one of its values v_{ij} is contextual.

We thus have a two-level definition of contextual attributes, with both levels of definition being of the same type. Definition 2 (*predictive attributes*) is identical to Turney's (1993) notion of *primary feature*. Definition 4 (*contextual attributes*) is more specific and operational than Turney's definition of *contextual features* (which essentially defines an attribute as contextual if it is not in itself predictive, but would lead to less predictive accuracy if omitted). We now specify procedures to identify potentially predictive and contextual features and attributes during the incremental learning process.

Identifying contextual features through meta-learning

Assume our base-level Bayesian classifier is learning on-line from a stream of incoming examples. After each learning step, we use a *statistical χ^2 test of independence* to determine which features are currently *predictive*:

Criterion 1 *(Predictive features)*: A feature $a_i : v_{ij}$ is recognized as *predictive* if the distribution of classes in examples with $a_i = v_{ij}$ is significantly different (as determined by a χ^2 test with a given significance level) from the unconditioned distribution of classes within the current window.

Predictive features are computed relative to the *current window* because predictivity is a temporary quality that may change with time and context. The information needed for the χ^2 test is readily available in the tables C_i and AVC_{ijk} that are maintained by the base-level learner.

Contextual features are also determined by a χ^2 test, on a higher level. To this end, we define 'meta-classes' $\hat{c}_{ij}$: an instance I is in class $\hat{c}_{ij}$ if feature $a_i : v_{ij}$ is recognized as predictive at the time of classification of I. Analogously to above, tables are maintained for these meta-classes: the table $\hat{C}_{ij}$ counts the number of examples in meta-class $\hat{c}_{ij}$, $\hat{AV}_{ij}$ counts the number of examples with attribute value $a_i = v_{ij}$, seen since the

Table 2: The two-level algorithm of METAL(B)

Parameters: W (fixed window size), α (significance level for χ^2 test).

for each new incoming instance I do
begin
  CAs := current_context_attributes($\hat{C}_{ij}$, $\hat{AV}_{ij}$, $\hat{AVC}_{ijkl}$, α);
  Vs := values of attributes CAs in current instance I;
  RIs := examples in current window with values Vs for attributes CAs;
  $Class$:= class predicted for I by naive Bayesian classifier from selected examples RIs;
  add I to current window and drop oldest instance from window;
  update tables C_i, AV_{ij}, AVC_{ijk} for base-level Bayesian learning;
  PFs := currently_predictive_features(C_i, AV_{ij}, AVC_{ijk}, α);
  update tables $\hat{C}_{ij}$, $\hat{AV}_{ij}$, $\hat{AVC}_{ijkl}$ for meta-level learning, given PFs
end;

very beginning, and $\hat{AVC}_{ijkl}$ counts the number of examples with $a_i = v_{ij}$ in meta-class $\hat{c}_{kl}$. In other words, $\hat{AVC}_{ijkl}$ keeps track of the number of co-occurrences of $a_i : v_{ij}$ combinations in examples and the predictiveness of certain features $a_k : v_{kl}$. These three tables are maintained with respect to the entire learning history (not the current window), as changes of context and the emergence of skewed predictivity distributions are long-term processes. Table 1 summarizes the various tables that need to be maintained. There are then two conceivable operational criteria by which one could detect contextual features and attributes:

Criterion 2a *(Contextual features)*: A feature $a_i : v_{ij}$ is recognized as *contextual* if the distribution of meta-classes $\hat{c}_{kl}$ in examples with $a_i = v_{ij}$ is significantly different (as determined by a χ^2 test with a given significance level) from the unconditioned distribution of the $\hat{c}_{kl}$, observed over the entire learning history.

Criterion 2b *(Contextual features)*: A feature $a_i : v_{ij}$ is recognized as *contextual* if, for some feature $a_k : v_{kl}$, the distribution of meta-class $\hat{c}_{kl}$ versus $\overline{\hat{c}_{kl}}$ in examples with $a_i = v_{ij}$ is significantly different (as determined by a χ^2 test with a given significance level) from the unconditioned distribution of $\hat{c}_{kl}$ versus $\overline{\hat{c}_{kl}}$, observed over the entire learning history.

Criterion 2a pays attention to global distribution changes between the predictivity of different features, while criterion 2b is basically a direct translation of definition 3 above: $a_i : v_{ij}$ is contextual if its values correlate with the predictivity of *some* other feature $a_k : v_{kl}$. After some experimentation with both approaches, we have settled for criterion 2b (though criterion 2a yields very similar results in most cases).

Recognizing contextual features and attributes via this two-stage process constitutes an act of *meta-learning*: the base-level learning process is monitored, and the temporal coincidence of predictivity of certain features and the presence of other features in training instances leads to the identification of attributes that could provide contextual clues. The contextual features are taken as a description or *identifier* of the current context. In the following, we present two learning systems with different underlying learning algorithms that take advantage of the information provided by meta-learning.

A Bayesian Classifier with Meta-Learning: METAL(B)

Our first algorithm is called METAL(B) (META-Learning with underlying Bayes classifier) and uses a simple Bayesian classifier as its underlying ('object-level') incremental learning algorithm. It was first presented and is described in more detail in (Widmer, 1996).

In METAL(B), the contextual attributes identified by meta-learning are used to *focus* the base-level Bayesian classifier on relevant examples when making predictions: whenever a new instance comes in, the set of attributes that are currently contextual (if any) is established, and the Bayesian classifier is then made to use for prediction only those examples from the window that have the same values for the contextual attributes as the new instance to be classified. In other words, the base-level classifier uses only those instances as a basis for prediction that seem to belong to the *same context* as the incoming example. If no attributes are currently recognized as contextual, the entire window is used for Bayesian classification. After classification, the true class of the new instance is read, and the learning tables for both base and meta-level are updated. Table 2 summarizes the complete two-level algorithm of METAL(B).

A consequence of this selective strategy is that base-level prediction becomes more expensive: the Bayesian classifier can no longer use the available tables C_i, AV_{ij}, and AVC_{ijk}, as these summarize all examples

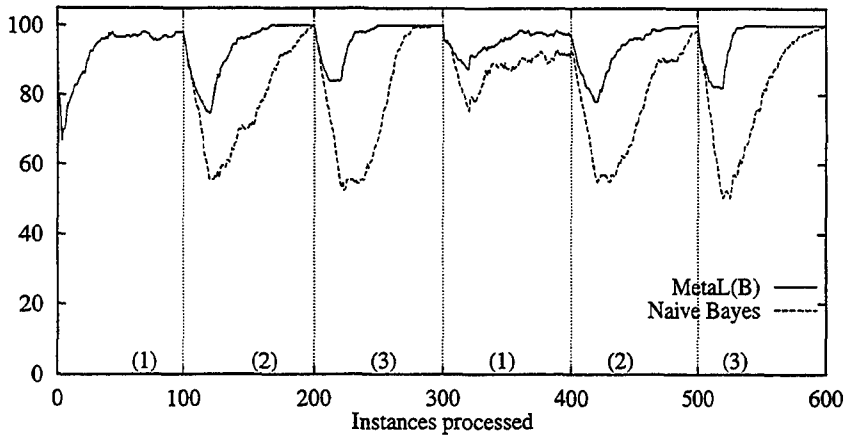


Figure 1: Effect of context identification in STAGGER concepts.

from the window. The relative frequencies required for the application of Bayes' rule must now be computed dynamically from the set of currently relevant instances (which are a subset of the window). On the other hand, this is no more expensive than the lookup in an instance-based learning algorithm (Aha et al., 1991) would be, and the fixed window size at least puts an upper bound on the number of examples that must be examined at each point in the learning process.

Note that METAL(B) depends on two *parameters*. All experiments reported below were performed with α=0.01 (99% confidence that the observed difference between conditioned and unconditioned distributions is not due to chance) and window size $W = 100$.

Experiments with METAL(B)

METAL(B)'s behavior was studied in a series of experiments in synthetic domains. Some of these are briefly described below. More detail can be found in (Widmer, 1996). In addition, METAL(B) was also evaluated on a hard 'real-world' problem where contextual effects might play a role, but the exact characteristics of the problem are not known.

Basic context identification

The first experiment demonstrates the basic effects of the context detection mechanism. The artificial test domain used here will serve as a running theme throughout most of our subsequent experiments.

METAL(B) was applied to a sequence of simple concepts that were first introduced by Schlimmer and Granger (1986) to test their concept drift tracking algorithm STAGGER. The concepts were later on also studied extensively in experiments with the FLORA algorithms (Widmer and Kubat, 1996). In a simple blocks world defined by three nominal attributes *size* $\in \{small, medium, large\}$, *color* $\in \{red, green, blue\}$, and *shape* $\in \{square, circular, triangular\}$, we define a sequence of three target concepts (1) $size = small \wedge color = red$, (2) $color = green \vee shape = circular$ and (3) $size = (medium \vee large)$. The (hidden) target concept will switch from one of these definitions to the next at certain points, creating situations of extreme concept drift. In addition, we introduce a fourth attribute *ctxt* $\in \{c1, c2, c3\}$, which is used to create perfect contextual information: whenever concept (1) is in effect, all examples (positive and negative) are made to have value $ctxt = c1$, etc.

Figure 1 plots the on-line prediction accuracy of METAL(B) vs. the simple Bayesian classifier on the concept sequence 1-2-3-1-2-3. Sequences of 600 examples each were generated randomly and labeled according to the currently ruling concept; after every 100 instances the context plus underlying concept was made to change. On-line accuracy was computed as a running average over the last 20 predictions. The figure plots the averages over 10 (paired) runs. Parameter settings were $\alpha = 0.01$ and *window size* $= 100$.

The curves show convincingly that METAL(B) does make effective use of the contextual information contained in the data. We witness both a less dramatic drop in accuracy at points of context change, and significantly faster re-convergence to high accuracy levels. Obviously, METAL(B) quickly identifies the contextual attribute *ctxt* (soon after the context has first switched from 1 to 2) and from then on concentrates only on examples pertaining to the new context, whereas the naive Bayes algorithm gives equal consideration to all examples in the current window, including those the still pertain to the old context. The fact that both algorithms fail to reach an accuracy of 100% in context (1) is due to the sparsity of the concept (only 11% of the examples are positive). The improvement produced by meta-learning, however, is evident.

(Widmer, 1996) analyzes in more detail the process of identifying predictive and contextual attributes. In this particular experiment, the process works basically

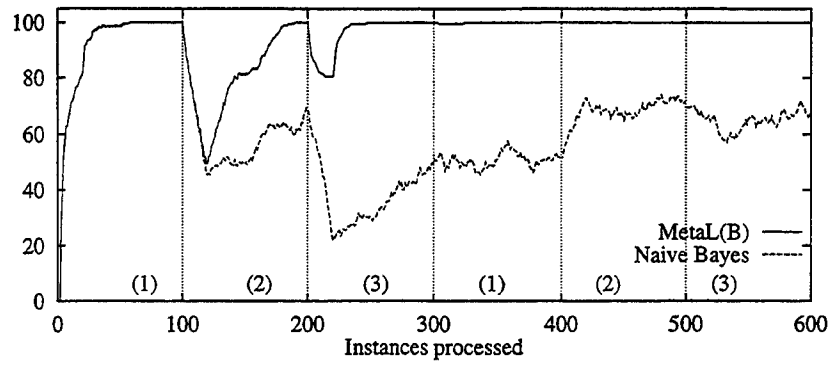


Figure 2: METAL(B) on STAGGER concepts — window size 300.

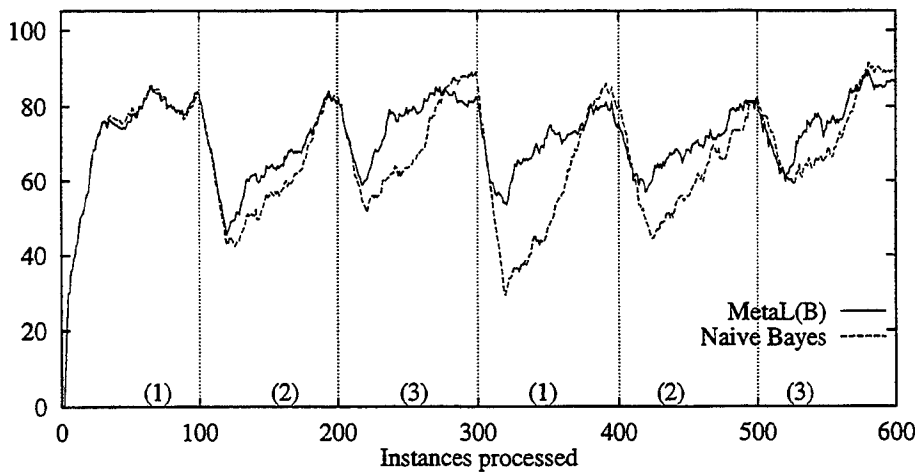


Figure 3: STAGGER concepts — 5 irrelevant attributes and 20% attribute noise.

as expected: the attributes *color* and *size* are recognized as predictive about midway during context (1), *color* and *shape* during context (2), and *size* during context (3). The attribute *ctxt* is recognized as a contextual clue towards the beginning of the (first occurrence of) context (2), due to that fact that *size* ceases to be predictive, and *shape* becomes predictive.

The effect of the system parameters

METAL(B) depends on two parameters: the significance level α used in the χ^2 tests at two levels, and the fixed *window size*. A level of $\alpha = 0.01$ has produced good results in all our experiments. Less strict significance levels make the meta-learner less discriminating: it becomes more easily possible for features to be accidentally 'recognized' as predictive for short periods, which causes frequent changes in the distribution of predictive features. The effect is instability. On the other hand, tighter significance levels (e.g., $\alpha = 0.001$) have left our results virtually unchanged.

As for the *window size*, the smaller the window, the faster the base-level learner will adapt to changes, and thus the smaller the advantage afforded by meta-learning. Too narrow a window is detrimental, as the Bayesian classifier bases its predictions on too few data points. Too large a window, on the other hand, reduces the base-level learner's ability to adjust to changes and, if it permanently contains conflicting instances from different contexts, may prevent the learner from ever reaching high predictive accuracy. Figure 2 plots the results on the STAGGER task when the window size is set to 300. METAL(B) detects the contextual attribute somewhere during context (2) and uses its values to always select the correct examples for prediction. After the first 300 examples, the window always contains instances from all three contexts, and METAL(B) can perfectly predict from then on. For the same reason, the simple Bayesian classifier fails completely.

Irrelevant attributes and noise

Bayesian learners are known to be quite robust in the face of irrelevant attributes and noise. Experiments

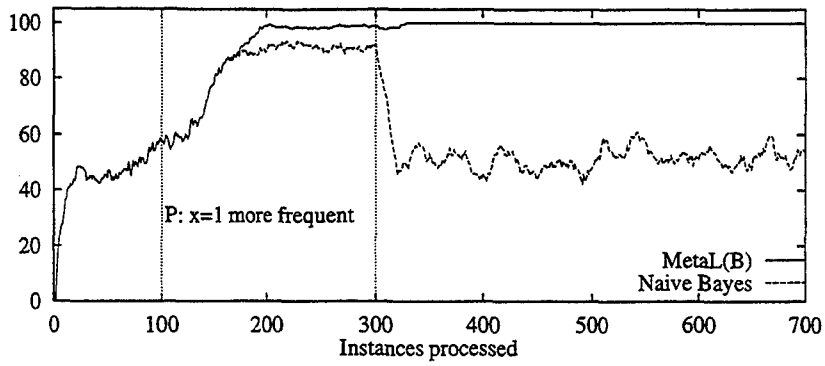


Figure 4: "Quasi-contextual" learning: XOR with 5 irrelevant attributes.

with METAL(B) have confirmed this, both for object-level and meta-level learning. As a rather extreme example, Figure 3 plots the performance on the STAGGER task when the examples were subjected to 20% attribute noise,[2] with an additional 5 irrelevant attributes with random values. It is evident that meta-learning still leads to improved adjustment to changes, though such high noise levels may eventually produce effects of instability, as we can see in context (3). Table 5 below gives more information.

Other experimental dimensions

Gradual vs. abrupt context changes:
Experiments also confirm that METAL(B) is rather insensitive to the *speed* of a context change, i.e., whether a new context takes over abruptly or only gradually. Unlike systems like FLORA4 (Widmer, 1994), METAL(B) has no problems with slow drift — it is the overall distributions of the predictivity of features that determine METAL(B)'s behavior, not necessarily dramatic short-term changes. This is important, because many realistic applications are presumably characterized by more or less gradual (and noisy) context changes. More details can be found in (Widmer, 1996).

Complex and imperfect contextual clues:
In real life, contexts may be defined by more complex combinations of features, and contextual attributes may be changing. Preliminary experiments suggest that METAL(B) has no problems with *conjunctively* defined contexts. Contexts defined by *disjunctions* of features create problems, especially when the contextual attributes are not the same in different contexts. The main problem is that contextual information is used conjunctively in METAL(B)'s focusing strategy, which makes the base-level learner rely on very few examples in some cases. Generally, of course, the meta-learner suffers from the same limitation as the base-level Bayesian classifier: it must assume independence between *contextual attributes*.

Another potential complication is that contexts may not always be characterized by perfectly constant values. Experiments with noise in the context attributes suggest that METAL(B) is robust to that kind of imperfection, but more refined experiments will be needed to study other types of imperfect clues (e.g., changing value distributions).

"Quasi-contextual learning":
An interesting side-effect of meta-learning is that it may in certain cases help the Bayesian learner to overcome its inherent representational limitations, incurred by the attribute independence assumption. As an example, consider the XOR function: in the concept $x = 1 \oplus y = 1$, neither of the two attributes x and y in isolation contributes directly to the classification. A Bayesian classifier will always linger around the base-level accuracy of 50%, given a set of uniformly distributed examples. The same holds for METAL(B), as long as the examples are presented in random order. However, if for some time examples appear in a skewed distribution, meta-learning may exploit this by learning to regard one of the attributes as contextual and the other as predictive. This two-level view of the XOR function would then allow the system to perfectly classify from that point on: if context is $x = 1$, then $y = 0$ implies XOR, and vice versa.

Figure 4 shows the learners on sequences of XOR examples with five additional irrelevant attributes, where during a certain period P (between the 100th and the 300th instance), examples — both positive and negative — with $x = 1$ were more frequent than those with $x = 0$ (90% vs. 10%). Before example 100 and after example 300, instances were presented in a uniform distribution. The results are again averages over 10 runs.

The effect is very clear: METAL(B) does indeed single out x as the contextual attribute at some point dur-

[2] A noise level of η% means that for each attribute in a training example, its true value is replaced by a random value from its domain with probability $\eta/100$. Both the 'base-level' attributes *color*, *size*, and *shape* and the context attribute *ctxt* were equally subjected to noise.

Table 3: Results of Schubert experiment.

Learning algorithm	Mean acc. (%)	Std. dev.	# runs better	max.better
Naive Bayes:	68.75	1.07	0	—
METAL(B):	74.62	1.44	50	9.22

ing period P, which allows it to reach an accuracy level of (almost) 100% quickly, and also to hold on to this level during the following random period. The simple Bayesian classifier improves its performance between examples 100 and 300 (it can never reach 100%), but quickly drops to 50% as the instances are again uniformly distributed.[3] We may conclude from this that meta-learning can be useful also in settings that are not normally thought of as characterized by changing contexts.

Learning from real data

METAL(B) was also tested on a difficult 'real-world' problem with unknown characteristics. The problem comes from the domain of tonal music and consists in learning to predict (on-line) what chord should accompany the next note in a given melody. Specifically, the task is to correctly predict one of three classes: *tonic harmony* (e.g., the note is to be played with a C major chord, if the piece is in the key of C major), *dominant* (i.e., a G major chord in the key of C), or *other*. In terms of a real scenario, imagine a guitar player who is trying to accompany a singer in real time on pieces she does not know and tries to get at least the two most important chord types (tonic and dominant) right.

The data used for the experiment were the melodies of Franz Schubert's *German Mass*, a collection of 8 songs of varying length (between 42 and 113 notes). There are 553 melody notes in total. The distribution of classes is 290 (52%) *tonic*, 179 (32%) *dominant*, and 84 (15%) *other*.

The individual notes were described by 11 discrete attributes: the *mode* of the current piece (major or minor), the *meter* (e.g., 4/4, 3/4, or 6/8), the current *tactus* (i.e., whether the major metrical level — the level at which one would tap one's foot in rhythm — is the level of quarter of eighth notes), the current *local key* (to describe modulations within a piece), and various attributes that describe the current note itself and its predecessor: *scale degree* (a tonality-independent abstraction of the note name), *duration*, and *metrical strength* of the current note, *duration* of the note's predecessor, the *interval* and its *direction* between the previous and the current note, and the *harmony* that accompanied the previous note.

[3]That METAL(B) fails to achieve a perfect 100% during period P, but then performs perfectly afterwards may seem surprising. The explanation lies in the highly unbalanced instance distribution during P. See (Widmer, 1996) for more detail.

We conjectured that more global properties like mode, meter, tactus, and local key might have a context-defining effect in certain cases, i.e., that the rules determining the harmonic role of a note might be slightly different in some of these contexts. However, we don't know this in detail, and the contextual effects, if any, might be weak and difficult to discover.

What makes the problem even harder is that the given attributes are highly *inadequate*: there are numerous cases of notes with the same description but different classification. Harmonic choices are by no means unique, and the specific decision also depends on aspects of larger context (musical form, harmonic rhythm, etc.) that are not captured by our local representation. It is thus clearly impossible to achieve a predictive accuracy of anything close to 100%.

To reduce the effect of the specific ordering of the songs, the algorithms were run on 50 random permutations of the 8 songs. Table 3 shows the results in terms of total classification accuracy. METAL(B)'s advantage of 5.87 percentage points is significant at the 0.001 level, according to a two-sided t-test. It scored better than the simple Bayesian classifier in *all* of the 50 runs, with a maximum advantage of 9.2 percentage points.

The attributes most often singled out as contextual were meter and tactus, less frequently mode, and very rarely local key (which was surprising to us, but probably indicates that the periods of modulation are too short and unstable to be contextually significant). Interestingly, also note duration was sometimes considered contextual: it does not help in directly predicting the harmony, but it is useful as a 'secondary' decision criterion. In other words, there is some dependency between this attribute and some more predictive ones, and METAL(B) resolves the dependency by treating note duration as a contextual attribute.

As a kind of afterthought, we propose an alternative view of meta-learning: instead of a focusing or selection strategy, it could also be interpreted as a process of *transfer* of (learned) knowledge *across contexts*. That leads one to ask the following question: could it be that the 8 pieces are so different that there cannot be much useful transfer from one piece to the next, in other words, that one would achieve better results by learning from each piece *separately*, simply starting from scratch with every new piece? And indeed, it turns out that the base-level learner, when run on each piece separately, reaches a total accuracy over the 8 songs of 69.54%, which is slightly *higher* (though not at a sta-

tistically significant level) than the 68.75% achieved by simple Bayesian learning with a fixed window over the whole sequence! METAL(B), on the other hand, achieves markedly higher accuracy. The conclusion is that indiscriminate transfer can indeed be harmful, but METAL(B) performs what might be called *selective cross-contextual transfer* — only those pieces of information are carried over that appear relevant to the current context.

An Alternative Realization of the Model: METAL(IB)

METAL(B) is but one possible incarnation of a very general learning framework. It seemed natural and elegant to use a Bayesian classifier for object-level learning, because metalearning as defined here also has a distinct Bayesian flavor. However, it should be possible in principle to use any other incremental induction algorithm for base-level learning. Given the serious inherent limitations of a naive Bayesian classifier, we have recently developed an alternative realization of the general model: METAL(IB) (METALearning with underlying Instance-Based classifier) realizes the same meta-learning strategy as METAL(B), but uses a simple *instance-based* learning algorithm (Aha et al., 1991) as base-level learner. New incoming examples are classified by a straightforward 1-NN (nearest-neighbor) method, using Euclidean distance as the (dis)similarity measure. Basis for prediction are the examples in the current window.

An instance-based learner allows us to use context information in a more flexible way, e.g., for feature weighting, which is not directly possible in METAL(B)'s Bayesian classifier. Also, instance-based learners can in principle approximate any target function, while Bayesian classifiers are severely limited. In the following, we present four variants of METAL(IB) that differ in the way they use the information about suspected contextual clues; their advantages and shortcomings, especially in relation to METAL(B), are then briefly investigated in the following section:

1. **Exemplar Selection** — METAL(IB)–ES:
   Here, contextual information is used in the same way as in METAL(B), namely, to *select relevant exemplars* during classification: only those instances from the window are used for prediction that have the same values for the current context attributes (i.e., presumably belong to the same context) as the current instance (the one to be classified).
2. **Exemplar Weighting** — METAL(IB)–EW:
   In this variant, contextual clues are used for exemplar *weighting* rather than strict selection. When classifying by nearest neighbor, each instance (*exemplar*) in the current window is assigned a weight W, and the measured similarity between new instance and exemplar is multiplied by W. The idea is that exemplars that are more likely to belong to the same context as the new instance should have more influence in classification. The weight is computed as

$$W(E) = 1/(1 + D_{|CAs|}(E, CI))$$

   where $D_{|CAs|}$ is the distance between the exemplar E and the current instance CI, measured only with respect to the context attributes CAs.
3. **Feature Weighting** — METAL(IB)–FW:
   Here, instead of weighting the examples in the window, METAL(IB) uses a weighted similarity measure that assigns different importance to individual attributes during nearest neighbor classification. Features or attributes believed to be *predictive relative to the current context* should receive correspondingly higher weights. To this end, we augment the meta-level algorithm with a component that *predicts* which attributes are or should be *predictive* for each incoming example, *given the instance's values for the currently recognized context attributes*. This can be easily done by using Bayes' rule on the tables updated in the meta-learning process ($\hat{C}_{ij}$, $\hat{AV}_{ij}$, and $\hat{AVC}_{ijkl}$). METAL(IB) computes a weight for each attribute as follows:

$$W(A) = 1 + P_{pred}(A)$$

   where $P_{pred}(A)$ is the probability which the system assigns to the belief that A is a predictive attribute relative to the current instance (computed via Bayes' rule at the meta-level).
4. **Combined Exemplar/Feature Weighting** — METAL(IB)–COM:
   This variant performs both exemplar and feature weighting.

Our expectations were that weighting approaches would generally be less brittle than strict exemplar selection, and that *feature weighting* in particular would produce a new, interesting effect: as the feature weights are derived, via Bayes' rule, from METAL(IB)'s meta-level tables ($\hat{C}_{ij}$, $\hat{AV}_{ij}$, $\hat{AVC}_{ijkl}$), which in turn are a kind of *memory of past contexts*, this feature weighting strategy should enable the system to readjust more quickly to contexts that have already occurred before.

METAL(IB)–COM, finally, should combine the advantages of exemplar weighting (fast reaction to perceived context change by disregarding exemplars in the window that pertain to an outdated context) and feature weighting (fast adjustment to contexts that have been encountered before).

Preliminary Experimental Results

Here are some preliminary results. Figure 5 compares the simple instance-based learner (with fixed window of size 100) with its various meta-learning

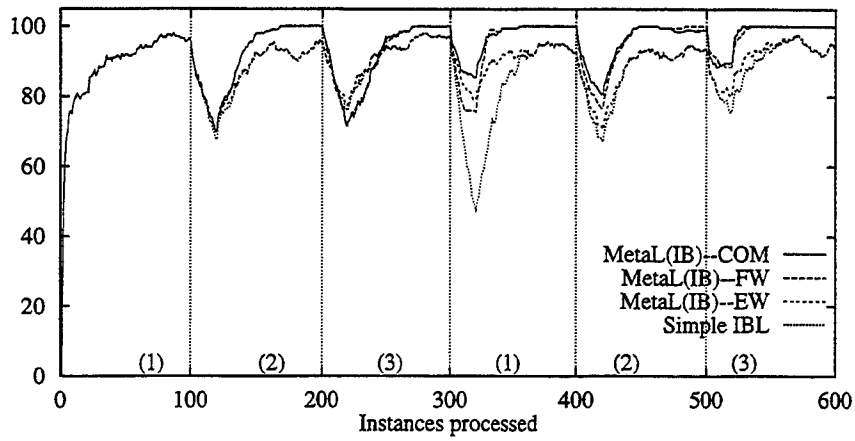


Figure 5: Context identification in METAL(IB).

Table 4: METAL(B) vs. METAL(IB) on simple STAGGER problem.

	No irrelevant attrs Acc.% (σ)	10 irrelevant attrs Acc.% (σ)
Naive Bayes:	82.75 (2.02)	77.51 (2.12)
METAL(B):	95.75 (0.49)	85.90 (3.07)
Simple IBL:	88.10 (0.69)	68.68 (2.34)
METAL(IB)–ES:	90.05 (0.91)	74.35 (3.48)
METAL(IB)–EW:	90.05 (0.91)	74.38 (3.42)
METAL(IB)–FW:	93.75 (0.73)	73.70 (2.57)
METAL(IB)–COM:	94.23 (0.69)	78.03 (4.09)

variants[4] on the basic STAGGER task (compare this to figure 1). Again, the improvement brought about by meta-learning is evident. Among the meta-learning approaches, feature weighting performs markedly better than exemplar weighting (and equally exemplar selection), and the two feature weighting variants (METAL(IB)–FW and METAL(IB)–COM) do indeed seem to adjust to contexts faster when they reappear (see the second appearance of contexts (1), (2), and (3)). However, this claim has yet to be tested in more extensive experiments.

Table 4 summarizes the results of this experiment, as well as another experiment with 10 additional irrelevant attributes, in terms of the *total predictive accuracy* achieved by each learner over the entire sequence of 600 training instance (averaged over the 10 runs, σ = standard deviation), and also gives the corresponding figures for METAL(B). The figures establish METAL(IB)–COM as the best of the METAL(IB) strategies. We note that simple instance-based learning is markedly better than simple Bayesian classification in the task with no irrelevant attributes. We may attribute this to the inherent representational limitations of Bayesian classifiers; METAL(B)'s result shows that this handicap is compensated by meta-learning. In contrast, the instance-based algorithms are significantly inferior in the situation with 10 irrelevant attributes, which confirms one of the fundamental (and well-known) shortcomings of instance-based approaches.

A similar picture emerges when the data are subjected to *attribute noise*. Table 5 lists overall accuracies in the STAGGER domain (with 5 additional irrelevant attributes), for different noise levels. Again, meta-learning brings considerable improvement, but the amount of improvement decreases with higher noise levels. The combined strategy METAL(IB)–COM again turns out to be the best METAL(IB) variant, but the Bayesian system METAL(B) clearly outperforms the instance-based learners, indicating that both noise and irrelevant attributes are detrimental to instance-based approaches.

A domain where METAL(IB) turns out to be superior to METAL(B) is our musical chord prediction problem, where simple instance-based learning achieved an average total accuracy of 76.14% and METAL(IB)–COM reached 79.58% (compare this to 68.75% and 74.62%, respectively, for the Bayesian

[4]The curve of METAL(IB)–ES is not included in this plot; exemplar selection and exemplar weighting performed nearly identically in all our experiments.

Table 5: Overall results on noisy STAGGER problem (with 5 irrelevant attributes).

	0% noise Acc.% (σ)	10% noise Acc.% (σ)	20% noise Acc.% (σ)	40% noise Acc.% (σ)
Naive Bayes:	79.70 (1.20)	75.73 (1.96)	73.78 (1.41)	66.80 (1.93)
METAL(B):	88.40 (2.41)	80.95 (1.79)	75.60 (2.48)	66.00 (2.22)
Simple IBL:	72.40 (1.81)	68.03 (2.17)	63.78 (2.02)	60.05 (1.30)
METAL(IB)–ES:	78.03 (2.53)	72.10 (2.96)	66.00 (2.45)	61.11 (1.46)
METAL(IB)–EW:	77.95 (2.45)	72.12 (2.88)	65.90 (2.41)	61.11 (1.50)
METAL(IB)–FW:	80.47 (1.76)	73.23 (2.57)	67.80 (2.00)	61.37 (2.34)
METAL(IB)–COM:	83.22 (2.12)	75.57 (2.87)	68.58 (2.41)	62.62 (2.22)

learners — see table 3). Here, the representational flexibility of an instance-based approach seems to pay off.

There are a number of other known methods in instance-based learning for exemplar selection, exemplar weighting, and feature weighting (see, e.g., Aha, 1989; Aha et al., 1991; Cost & Salzberg, 1993) that should be experimentally compared to METAL(IB). Preliminary results achieved with David Aha's algorithms *IB3* (which performs a form of dynamic exemplar selection based on monitored predictive accuracy) and *IB4* (dynamic feature weighting) in the musical chord prediction problem suggest that METAL(IB) is clearly superior on this problem (79.6% for METAL(IB)–COM vs. 61.1% (IB3) and 68.3% (IB4)). More extensive and detailed experimental comparisons are under way.

Conclusion

The main contribution of this paper is to have shown that it is indeed possible for an incremental learner to autonomously detect, during on-line object-level learning, contextual clues in the data if such exist. The key is an operational definition of predictive and, based on those, contextual features. Identification and subsequent use of contextual information is an act of *meta-learning*. There are various ways in which such context information can be used. This paper has presented two different realizations of the meta-learning model: METAL(B) relies on a Bayesian classifier as the underlying incremental learner and uses context information to select those known instances as a basis for prediction that seem to belong to the same context as the new instance. METAL(IB) is based on an instance-based algorithm and uses contextual information for exemplar and feature weighting. The feasibility of context recognition and its benefits have been shown in a number of experiments.

Both METAL(B) and METAL(IB) are currently limited to domains described by symbolic, discrete attributes. A generalization to numeric features should not prove too difficult. Instead of maintaining counts of attribute-value and attribute-value-class combinations, the Bayesian classifier could simply assume that numeric features follow a normal distribution and keep track of the mean values and standard deviations. At the meta-level, the χ^2 tests would have to be replaced by an appropriate test of independence for continuous distributions.

Generally, we regard the work presented here as a small first step into what might become a rich field of research. The identification of contextual features is a first step towards *naming*, and thus being able to *reason about*, contexts. That is the level where we expect the full power of meta-learning to become apparent. Reasoning and learning about contexts could be used in a variety of ways and for a number of purposes, e.g., constructive induction, the recognition of (and faster readjustment to) previously encountered contexts, the emergence of expectations and predictions of the next context, etc.

There is a very interesting connection between our learning model and the notions of *transfer* and *life-long learning*, as recently proposed by Thrun and Mitchell (1995). As noted above in the context of the Schubert experiment, our learners can be interpreted as performing *cross-contextual transfer*, and they certainly are 'lifelong' learners. At first sight, the two models might appear to be orthogonal (one performs transfer across learning tasks, the other across contexts within a single task), but there are interesting parallels, and further research might lead to the formulation of a more general and powerful model that integrates both aspects of transfer.

Acknowledgments

The Austrian Research Institute for Artificial Intelligence gratefully acknowledges financial support from the Austrian Federal Ministry for Science, Research, and the Arts.

References

Aha, D.; Kibler, D.; and Albert, M. 1991. Instance-based learning algorithms. *Machine Learning* 6(1):37–66.

Aha, D. 1989. Incremental, instance-based learning of independent and graded concept descriptions.

In *Proceedings of the 6th International Workshop on Machine Learning (ML-89)*. San Mateo, CA: Morgan Kaufmann.

Bergadano, F.; Matwin, S.; Michalski, R.; and Zhang, J. 1992. Learning two-tiered descriptions of flexible contexts: The POSEIDON system. *Machine Learning* 8(1):5–43.

Cost, S., and Salzberg, S. 1993. A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning* 10(1):57–78.

Katz, A.; Gately, M.; and Collins, D. 1990. Robust classifiers without robust features. *Neural Computation* 2(4):472–479.

Michalski, R. 1987. How to learn imprecise concepts: A method employing a two-tiered knowledge representation for learning. In *Proceedings of the 4th International Workshop on Machine Learning*. Los Altos, CA: Morgan Kaufmann.

Schlimmer, J., and Granger, R. 1986a. Beyond incremental processing: Tracking concept drift. In *Proceedings of the 5th National Conference on Artificial Intelligence (AAAI-86)*. Los Altos, CA: Morgan Kaufmann.

Schlimmer, J., and Granger, R. 1986b. Incremental learning from noisy data. *Machine Learning* 1(3):317–354.

Thrun, S., and Mitchell, T. 1995. Learning one more thing. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*. San Mateo, CA: Morgan Kaufmann.

Turney, P., and Halasz, M. 1993. Contextual normalization applied to aircraft gas turbine engine diagnosis. *Journal of Applied Intelligence* 3:109–129.

Turney, P. 1993. Robust classification with context-sensitive features. In *Proceedings of the 6th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-93)*. Edinburgh, Scotland.

Watrous, R., and Towell, G. 1995. A patient-adaptive neural network ecg patient monitoring algorithm. In *Proceedings Computers in Cardiology 1995*. Vienna, Austria.

Watrous, R. 1993. Speaker normalization and adaptation using second-order connectionist networks. *IEEE Transactions on Neural Networks* 4(1):21–30.

Widmer, G., and Kubat, M. 1993. Effective learning in dynamic environments by explicit context tracking. In *Proceedings of the 6th European Conference on Machine Learning (ECML-93)*. Berlin: Springer Verlag.

Widmer, G., and Kubat, M. 1996. Learning in the presence of concept drift and hidden contexts. *Machine Learning* (in press).

Widmer, G. 1994. Combining robustness and flexibility in learning drifting concepts. In *Proceedings of the 11th European Conference on Artificial Intelligence (ECAI-94)*. Chichester, UK: Wiley & Sons.

Widmer, G. 1996. Recognition and exploitation of contextual clues via incremental meta-learning. In *Proceedings of the 13th International Conference on Machine Learning (ICML-96)*. San Francisco: CA: Morgan Kaufmann. Extended version available as Report TR-96-01, Austrian Research Institute for Artificial Intelligence, Vienna.

Learning Weighted Prototypes using Genetic Algorithms

Jianping Zhang and Qiu Fan
Department of Computer Science
Utah State University
Logan UT 84322-4205
jianping@zhang.cs.usu.edu

Abstract

Smith and Osherson proposed the prototype view for concept representation and category classification. In the prototype view, concepts are represented as prototypes. A prototype is a collection of salient properties of a concept. Under the prototype view, an instance is classified as a member of a concept if it is sufficiently similar to the prototype of this concept. Although the prototype view has been extensively researched in cognitive science, it has not been widely adopted in machine learning. In this paper, we discuss some preliminary work on a genetic algorithms approach to learning weighted prototypes. In this approach, a concept is represented as one or more weighted prototypes, each of which is a conjunction of weighted attribute values. In this approach, every prototype maintains its own attribute weights. A genetic algorithm is applied to generate prototypes and their attribute weights. This approach has been implemented in GABWPL (Genetic Algorithm Based Weighted Prototype Learning) and empirically evaluated on several artificial datasets.

1. Introduction

Smith and Osherson (1984) proposed the prototype view for concept representation and category classification. In the prototype view, concepts are represented as prototypes. A prototype is a collection of salient properties of a concept. Under the prototype view, an instance is classified as a member of a concept if it is sufficiently similar to the prototype of this concept. The prototype representation strongly supports human concept formation. People tend to remember those most often encountered instances and forget those rarely encountered instances. Concepts involved in real world applications usually possess graded structures (Barsalou 1985). Instead of being equivalent, instances of a concept may be characterized by a degree of typicality in representing the concept. Prototypes represent the central tendencies of such graded structures, so concepts described by prototypes are more human understandable than those described by regular instances and also easier for human to capture the basic principles underlying these concepts.

Although the prototype view has been extensively researched in cognitive science, it has not been widely adopted in machine learning. In recent years, instance-based learning (IBL) becomes popular for several reasons. First, it is strongly motivated by similar psychologically plausible algorithms that perform concept formation (Smith & Medin 1981; Aha & Goldstone 1992; Zhang 1992). Second, polymorphy and imprecision of natural concepts prevent many inductive learning approaches from inducing general concept descriptions, while instance-based (exemplar-based) approaches perform well in domains involving polymorphous and imprecise concepts (Bareiss, Porter, & Wier 1990). Third, instance-based learning algorithms are simple and easy to understand and implement. A number of instance-based learning methods, e.g. Protos (Bareiss, Porter, & Wier 1990), IBn (Aha, Kibler, & Albert 1991), and Each (Salzberg 1991), were developed and applied to many practical problems. These problems include clinical audiology (Bareiss, Porter, & Wier 1990), diagnosis of heart diseases, classification of congressional voting records (Aha & Kibler 1989), prediction of protein secondary structures, word pronunciation (Stanfill & Waltz 1986), and prediction of DNA promoter sequences (Cost & Salzberg 1991). The results obtained by these IBL methods were comparable to those obtained from other learning methods.

Unfortunately, IBL has its disadvantages. Because concept descriptions are represented by many individual instances, they are not human understandable. As we mentioned above, a prototype represents a central tendency of a concept, thus prototypes are easy for human to understand the concepts represented. Each classification of a novel instance involves measuring the distance between the novel instance and each of the stored instances, so it becomes very inefficient to make classifications when the number of stored instances becomes large. Therefore, it is important to develop effective storage reduction algorithms for IBL. Skalak (1993) showed that significant reduction in storage of instances can be achieved in some datasets using a few prototypes without decreasing IBL's classification accuracy.

Genetic algorithms are search techniques. A genetic algorithm represents a specialized search technique, which, although it is not guaranteed to arrive at a "best" solution, generally arrives at a good or near optimal solution. Furthermore, in terms of time, it is considerably more efficient than random or exhaustive searches when the search space is large. Learning a concept description is a search problem. There have been many applications of genetic algorithms to machine learning (e.g., Wilson 1987; De Jong 1988; De Jong, Spears, & Gordon 1993). Learning weighted prototypes is to find a set of prototypes in a given instance space and their attribute weights. A prototype is an instance, but may not be a training instance. The number of subsets of an instance space is huge. The search space becomes much larger after attribute weights are used. Therefore, we choose a genetic algorithms approach to the task of learning weighted prototypes.

In this paper, we discuss some preliminary work on a genetic algorithms approach to learning weighted prototypes. In this approach, a concept is represented as one or more weighted prototypes, each of which is a conjunction of weighted attribute values. In this approach, every prototype maintains its own attribute weights. A genetic algorithm is applied to generate prototypes and their attribute weights. This approach has been implemented in GABWPL (Genetic Algorithm Based Weighted Prototype Learning) and empirically evaluated on several artificial datasets.

Michalski (1994) defined a multistrategy learning system as a learning system that integrates two or more inferential and/or computational strategies. According this definition, GABWPL is a multistrategy learning system because it combines a genetic algorithm approach with a prototype learning approach.

2. Previous work

Recently, there have been some attempts at creating systems for learning prototypes in machine learning (de la Maza 1991; Zhang 1992; Datta & Kibler 1995). This section discusses the work in applying genetic algorithms to learning prototypes.

The work done by Sen & Knight (1995) is the most closely related to the work reported in this paper. Sen and Knight implemented a genetic algorithm based prototype learning system PLEASE. In PLEASE, a concept is represented by one or more prototypes and a prototype is a set of attribute values. Each population structure represents descriptions of all possible concepts and consists of one or more prototypes belonging to each of the possible concepts. PLEASE does not use the bit-string representation, instead it uses real attribute values. Three operators (*mutation*, *creep*, and *two point crossover*) are used to evolve population structures. PLEASE was run on a set of artificial datasets and achieved higher classification accuracy than C4.5 on these datasets.

The work reported in this paper improves PLEASE with several extensions. First, the attribute values of a prototype are weighted in our work. Each prototype maintains its own set of attribute weights. The attribute weights of a prototype are learned with the prototype. Second, we use a different set of operators: *crossover*, *mutation*, *addition*, and *deletion*. The first two operators are similar to those used in PLEASE and addition and deletion are new operators. Addition adds a new prototype to a population structure, while deletion deletes a prototype from a population structure. Third, we use a different fitness function which takes not only the classification accuracy on training set but also the number of prototypes into consideration. The population structure with fewer prototypes is preferred over the one with more prototypes. In PLEASE, the number of prototype is limited by a user input parameter.

Skalak (1993) describes an application of the random mutation hill climbing algorithm to prototype learning. In Skalak's method, the number of prototypes is fixed and determined by a parameter m. All prototypes are selected from training instances. In his method, Skalak applies random mutation hill climbing to select the best combination of m prototypes, which maximizes the classification accuracy on the training set. Skalak also applies random mutation hill climbing to select attributes. Experiments conducted by Skalak show that significant reduction in storage of instances can be achieved by his simple search method without decreasing classification accuracy on the selected datasets.

Kelly and Davis (1991) apply genetic algorithm technique to learn real-valued attribute weights for a simple instance-based learning algorithm. In their method, attributes weights are the same in the entire instance space. They propose to maintain a different set of weights for each concept.

3. Learning and Classification with Weighted Prototypes

Given a attributes, a weighted prototype P is a vector of a pairs (v_i w_i) plus its class membership, where v_i is P's value of the ith attribute and w_i is P's weight of the ith attribute taking a value between 0 and 1. A different weighted prototype has a set of different attribute weights. Given a set of classified training instances, the task of learning weighted prototypes is to generate a minimum number of weighted prototypes, which maximizes the classification accuracy on the training instances. At least one weighted prototype is generated for each class.

A novel instance is classified to the class of a weighted prototype which is the closest to the novel instance among all weighted prototypes. The distance metric used in our work is the weighted Euclidean distance metric, in which the distance between an instance $I = (x_1, ..., x_a)$ and a weighted prototype $P = ((v_1\ w_1), ..., (v_a\ w_a))$ is

$$\sqrt{\sum_{i=1}^{a} w_i\ (x_i - v_i)^2},$$

Zhang and Yang (1996) show that the decision boundary between two weighted prototypes is a quadratic hypersurface and that the decision boundary between two unweighted prototypes is a hyperplane. When all prototypes share the same set of weights, the decision boundary is still a hyperplane. Depending on the weights of the two weighted prototypes, the decision boundary between them could be an ellipse, a hyperbola, a parabola, or two line in a two dimensional instance space.

4. Genetic Algorithms for Learning Weighted Prototypes

In this section, we describe the genetic algorithms method which we use to learn weighted prototypes. We introduce population structures, fitness functions, and operators used in our method.

4.1 Population Structures

Given *n* classes, a population structure consists of n class descriptions, each consisting of one or more weighted prototypes. In many genetic algorithms based learning systems (e.g. De Jong, Spears, & Gordon 1993), a rule is encoded as a string of binary bits. This binary bit representation is good for rule induction learning, but not for prototype or instance-based learning. In our method, a weighted prototype is represented as a string of *a* pairs of real values, where *a* is the number of attributes. The first value of a pair is the value of the corresponding attribute and the second value is the attribute weight and takes a value between 0 and 1.

Weighted prototypes of the same class are placed consecutively, and classes are separated by a comma. The number of classes is fixed, but the number of weighted prototypes in a class is variable. Therefore, the length of a population structure in our method is not fixed and may grow or shrink during evolution. The following is an example of a population structure in a domain with two attributes and two classes.

$v_{11}w_{11}v_{12}w_{12}v_{21}w_{21}v_{22}w_{22}$, $v_{31}w_{31}v_{32}w_{32}v_{41}w_{41}v_{42}w_{42}$

where v_{ij} is the value of the jth attribute of the ith prototype and w_{ij} is the weight of the jth attribute of the ith prototype. In this example, the population structure consists of four prototypes two for each class.

4.2 Fitness Function

A fitness function measures the goodness of a population structure during evolution. The population structure with a greater fitness value is stronger so as to get a greater survival chance. Fitness functions used in many genetic algorithms based learning systems (e.g., De Jong, Spears, & Gordon 1993; Kelly & Davis 1991) use only classification accuracy. In our fitness function, we take both classification accuracy and the number of prototypes into consideration. We prefer the population structure with high classification accuracy and a small number of prototypes. Too many prototypes may cause overfiting. Namely, they may perform well on training data but not on test data.

The fitness function used in our method is:

$$f(x,y) = \frac{x}{C+y},$$

where x is the percentage of correctly classified instances, y is the number of prototypes in population structure, and C is a nonnegative real value which controls the tradeoff between the classification accuracy and the number of weighted prototypes. When C is 0, every time the number of prototypes increases by 1, the classification accuracy must increase by more than the average percentage of classification accuracy per prototype in order to get a larger fitness value. A larger C means that when a new prototype is added to a population structure, a less increase in classification accuracy is required in order to get a larger fitness value. C is dynamically adjusted during evolution.

4.3 Operators

Four operators are used in our method to evolve population structures. They are *crossover*, *mutation*, *addition*, and *deletion*. The first two operators are often used in almost all genetic algorithms methods. *Addition* and *deletion* are specially designed operators to change the length of a population structure.

A two point crossover operator is applied in our method. The crossover operator is executed as follows. The two crossover points are first randomly selected on the parent with a shorter length. The two crossover points on the second parent are the same as those on the first parent. The two offsprings have the same lengths as the two parents respectively. The probability of crossover is determined by the parameter P_{cross}. Table 1 shows an example that demonstrates an application of the crossover operator in a two class and two attribute domain.

Table 1. Example of an application of the crossover operator

Parent #1:	$v_{11}^1 w_{11}^1 v_{12}^1 \mid w_{12}^1 v_{21}^1 w_{21}^1 v_{22}^1 w_{22}^1, v_{31}^1 w_{31}^1 v_{32}^1 w_{32}^1 v_{41}^1 w_{41}^1 \mid v_{42}^1 w_{42}^1$
Parent #2:	$v_{11}^2 w_{11}^2 v_{12}^2 \mid w_{12}^2 v_{21}^2 w_{21}^2 v_{22}^2 w_{22}^2 v_{31}^2 w_{31}^2 v_{32}^2 w_{32}^2, v_{41}^2 w_{41}^2 \mid v_{42}^2 w_{42}^2 v_{51}^2 w_{51}^2 v_{52}^2 w_{52}^2$
Offspring #1:	$v_{11}^1 w_{11}^1 v_{12}^1 \mid w_{12}^2 v_{21}^2 w_{21}^2 v_{22}^1 w_{22}^2 v_{31}^2 w_{31}^2 v_{32}^2 w_{32}^2, v_{41}^2 w_{41}^2 \mid v_{42}^1 w_{42}^1$
Offspring #2:	$v_{11}^2 w_{11}^2 v_{12}^2 \mid w_{12}^1 v_{21}^1 w_{21}^1 v_{22}^1 w_{22}^1 v_{31}^1 w_{31}^1 v_{32}^1 w_{32}^1, v_{41}^1 w_{41}^1 \mid v_{42}^2 w_{42}^2 v_{51}^2 w_{51}^2 v_{52}^2 w_{52}^2$

The first parent has four prototypes two for each class, and the second parent has five prototypes three for the first class and two for the second class. The first offspring has four prototypes two for each class, and the second offspring has five prototypes three for the first class and two for the second class.

Mutation changes the value or weight of an attribute of a prototype, which is randomly selected. The amount changed is dependent on the ratio of the number of correctly classified instances to the number of incorrectly classified instances. The higher the ratio is, the smaller the amount changed is. The change may be increase or decrease determined randomly. The probability of mutation is determined by the parameter P_{mut}.

Addition adds a training instance to a class of a population structure as a new prototype. In a population structure, the new prototype is added to the class with the most error omission (the number of training instances belonging to the class, but misclassified to other classes). The training instance with the largest distance to the class is added as the new prototype of the class and all weights are initialized to 0.5. The probability of addition is determined by the parameter P_{add}.

Deletion deletes a prototype from a class of a population structure. In a population structure, the new prototype from the class with the most error commission (the number of training instances belonging to other classes, but misclassified to this class). The prototype to be deleted is randomly selected. The probability of addition is determined by the parameter P_{del}.

5. Experiments

The method discussed in Section 4 was implemented in a system called in GABWPL (Genetic Algorithm Based Weighted Prototype Learning). In this section, we report the experimental results on eight artificial datasets.

5.1 Test Problems And Experiments

All eight problems are defined in a two dimensional space. The two attributes are numeric attributes with values ranged from 0 to 1. All problems contain two classes: class 0 and class 1. The first four problems are the same as those used in (Sen & Knight 1995). In these four problems, decision regions are separated by one or more lines. They are called 1/1 line, 1/2 line, 1/3 line, and 1/4 line respectively. Figure 1 shows these four problems.

Next four problems are similar to the first four problems, but their regions are separated by arcs not lines. These four problems are more complex than the first four problems. The are called 1/1 arc, 1/2 arc, 1/3 arc, and 1/4 arc respectively. Figure 2 shows the four arc problems.

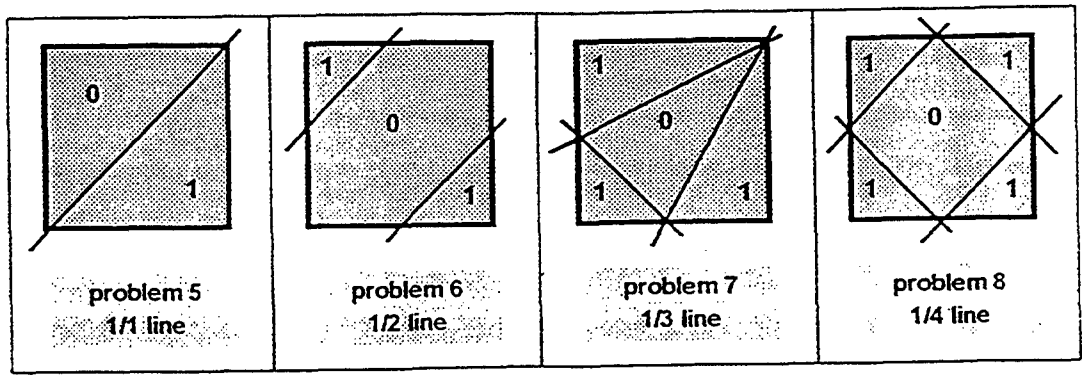


Figure 1. Four line problems

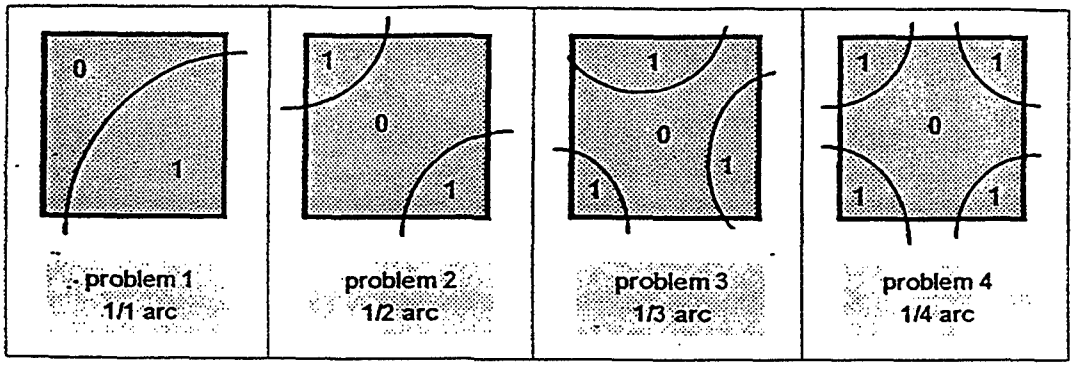


Figure 2. Four arc problems

In all experiments, 800 training instances and 200 test instances were used. All instances were randomly generated and the training and test datasets are disjoint. All experiments were repeated ten times on different set of training and test datasets. We ran two versions of GABWPL, one with no attribute weights and one with attribute weights. Our hypotheses are that GABWPL with weights should attain higher classification accuracy than GABWPL with no weight on the four arc problems while GABWPL with no weight may attain higher classification than GABWPL with weights on the four line problems. The decision boundary between two weighted prototypes is an arc while the decision boundary between two unweighted prototypes is a line (Zhang & Yang 1996). With some weight values, the decision boundary between two weighted prototypes becomes a line.

In all experiments, P_{corss} was set to 0.6, P_{mut} was set to 0.05, P_{add} was set to 0.2, and P_{del} was set to 0.15. The population size was 40 and the number of generations was 1200. Actually, experiments on all problems except for 1/3 arc and 1/4 arc converged before generation 200 and experiments on 1/3 arc and 1/4 arc converged around generation 400.

5.2 Experimental Results

All results reported in this section are the average of ten runs. Table 2 reports the classification accuracy on both training and test datasets. As we expect, GABWPL with weights attains about 2% higher classification accuracy than GABWPL with no weight on the four arc problems. GABWPL with weights achieves about the same classification accuracy as GABWPL with no weight on 1/1 line and 1/2 line and about 2% lower classification accuracy than GABWPL with no weight on 1/3 line and 1/4 line.

Table 3 shows the average number of prototypes generated. On all arc problems, fewer weighted prototypes were generated than unweighted prototypes, because an arc needs to be approximated by more than one line segments. Figure 3 and 4 show where generated prototypes were located on the line and arc problems, respectively. The two numbers around a weighted prototype are the two weights of the weighted prototype.

Table 2. Average Classification accuracy

Test Problems	Training Set		Test Set	
	No weight	With Weight	No weight	With Weight
1/1 line	100%	100%	99.5%	99.5%
1/2 line	99.5%	99.5%	99.7%	99.1%
1/3 line	99.4%	97.1%	98.5%	96.3%
1/4 line	98.3%	97.1%	97.8%	95.1%
1/1 arc	97.9%	100%	96.9%	99.8%
1/2 arc	97.3%	98.9%	96.9%	98.5%
1/3 arc	93.8%	96.0%	90.8%	92.9%
1/4 arc	97.5%	98.2%	95.7%	97.3%

Table 3. Average number of prototypes generated

Test Problems	No weight	With weight
1/1 line	2	2
1/2 line	3	3
1/3 line	4	3.6
1/4 line	5	4.2
1/1 arc	3	2
1/2 arc	3.3	3.1
1/3 arc	5.4	4.5
1/4 arc	7.6	5.5

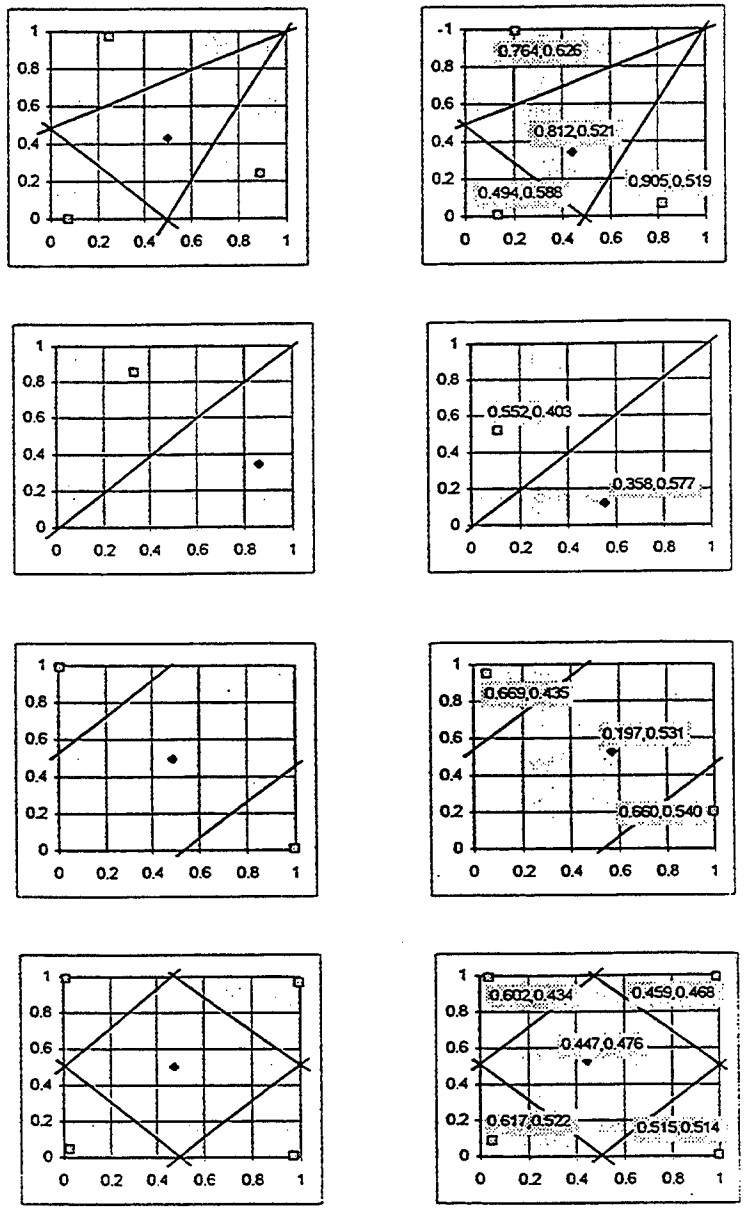


Figure 3. Prototypes generated for the line problems

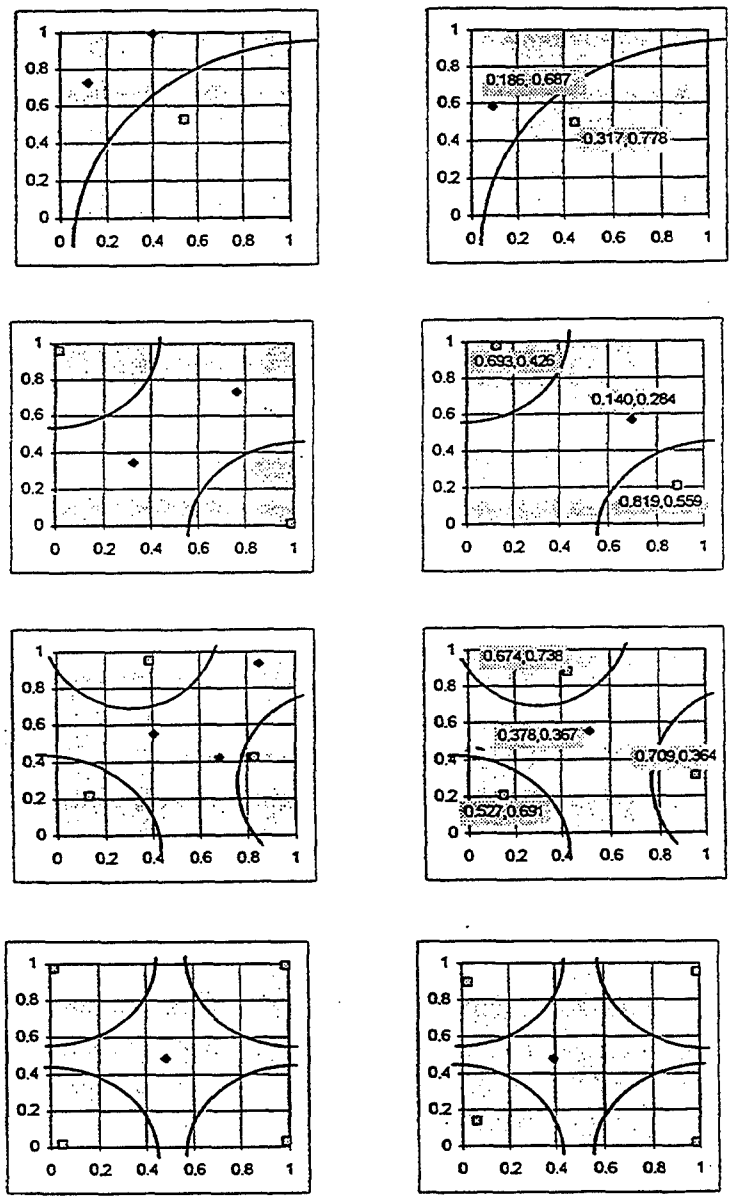


Figure 4. Prototypes generated for the arc problems

6. Conclusion and Future Work

In this paper, we have described a multistrategy learning system that combines a genetic algorithm approach with a prototype learning approach. The major novelty of this approach is to apply genetic algorithm to learn attribute weights and prototypes together. Our experimental results show that this approach works well on some simple artificial problems. Other contributions include the use of operators *addition* and *deletion* and a fitness function which takes the number of prototypes into consideration.

The work reported in this paper is preliminary. In the future, a number of problems need to be addressed. First, more experiments need to be conducted. We need to run GABWPL on more difficult problems: problems with more attributes and problems with irrelevant attributes. We need to apply GABWPL to some real world applications. And also we need to compare GABWPL with other learning methods, particularly instance-based learning methods.

Second, we will explore how domain knowledge (knowledge extracted from training instances) can be used in our method to speed up the search for the best descriptions. Use of domain knowledge may be very important for complex problems.

Third, the current version of GABWPL works only for numeric attributes and it should be extended to symbolic attributes. Fourth, the fitness function needs to adjusted. Finally, more operators should be investigated.

References

Aha, D. and Kibler, D. 1989. "Noise-tolerant instance-based learning algorithms," Proceedings of the 11th International Joint Conference on Artificial Intelligence, Detroit, MI..

Aha, D.W. & Goldstone, R.L. 1992. "Concept learning and flexible weighting," Proceedings of Fourteenth Annual Conference of the Cognitive Science Society.

Aha, D., Kibler, D, and Albert, M. 1991. "Instance-Based Learning Algorithm," *Machine Learning* 6.

Bareiss, E. R., Porter, B. W., and Wier, C. C. 1990. "Protos: An Exemplar-Based Learning Apprentice," in Kodratoff & Michalski (Eds), *Machine Learning: An Artificial Intelligence Approach V III*, San Mateo, CA: Morgan Kaufmann

Barsalou, L. 1985. "Ideals, central tendency, and frequency of instantiation as determinants of graded structure in categories," in Journal of Experimental Psychology: Learning, Memory and Cognition, 11.

Cost, S., and Salzberg, S. 1991. "Q weighted nearest neighbor algorithm for learning with symbolic features," Technique report, Department of Computer Science, The Johns Hopkins University

Datta, P., & Kibler, D. 1995. "Learning prototypical concept descriptions," Proceedings of the 12th International Conference on Machine Learning

Dejong, K.A. 1988. "Learning with genetic algorithms: an overview," *Machine Learning* 3.

Dejong, K.A., Spears, W.M., & Gordon, D.F. 1993. "Using genetic algorithms for concept learning," *Machine Learning* 13.

de la Maza, J. 1991. "A prototype based symbolic concept learning system," Proceedings of the 8th International Workshop on Machine Learning.

Kelly, J.D., & Davis, L. 1991. "A hybrid genetic algorithm for classification," Proceedings of the 12th International Joint Conference on Artificial Intelligence.

Michalski, R.S. 1994. "Inferential theory of learning," in Michalski & Tecuci (Eds), *Machine Learning: A Multistrategy Approach*, Vol. IV, San Mateo, CA: Morgan Kaufmann.

Salzberg, S. 1991. "A nearest hyperrectangle learning method," Machine Learning, 6:3.

Sen, S., & Knight, L. 1995 "A genetic prototype learner," Proceedings of the 14th International Joint Conference on Artificial Intelligence.

Skalak, D.B. 1994. "Prototype and feature selection by sampling and random mutation hill climbing algorithms," Proceedings of the 11th International Conference on Machine Learning.

Smith, E. E., & Medin, D. L. 1981. *Categories and Concepts*. Harvard University Press.

Smith, E.E., & Osherson, D.N. 1984. "Conceptual combination with prototype concepts. *Cognitive Science* 9.

Stanfill, C. and Waltz, D. 1986. "Toward memory-based reasoning," Communications of the ACM, 29:12.

Wilson, S.W. 1987. "Classifier systems and the animate problem," *Machine Learning* 2.

Zhang, J. 1992. "Selecting Typical Instances in Instance-Based Learning," Proceedings of the Ninth International Conference on Machine Learning.

Zhang, J., & Yang, J. 1996. "The role of attribute weights in instance-based learning," Unpublished manuscript.

IV. Special Topics and Applications

Revising User Profiles: The Search for Interesting Web Sites

Daniel Billsus and Michael Pazzani

Department of Information and Computer Science
University of California, Irvine
Irvine, California 92717
{dbillsus, pazzani}@ics.uci.edu

Abstract

We describe Syskill & Webert, a software agent that learns to rate pages on the World Wide Web (WWW), deciding what pages might interest a user. The user rates explored pages on a three point scale, and Syskill & Webert learns a user profile by analyzing the information on each page. We focus on an extension to Syskill & Webert that lets a user provide the system with an initial profile of his interests in order to increase the classification accuracy without seeing many rated pages. We represent this user profile in a probabilistic way, which allows us to revise the profile as more training data is becoming available using "conjugate priors", a common technique from Bayesian statistics for probability revision. Unseen pages are classified using a simple Bayesian classifier that uses the revised probabilities. We compare our approach to learning algorithms that do not make use of such background knowledge, and find that a user defined profile can significantly increase the classification accuracy.

Introduction

There is a vast amount of information on the World Wide Web (WWW) and more is becoming available daily. How can a user locate information that might be useful to that user? In previous work, we discussed Syskill & Webert, a software agent that learns a profile of a user's interest from an individual's ratings of the interestingness of Web pages, and uses this profile to identify other interesting web pages. Initial experiments with our system (Pazzani, Muramatsu, Billsus, 1996) have shown that although it is much more accurate than chance at predicting whether a user would be interested in a page, the accuracy does not increase substantially as the user rates more and more pages. In this paper, we address an approach to increasing predictive accuracy by obtaining an initial profile of the user indicating the user's interest. As more rated pages become available this profile is gradually revised to make it fit the actual observed rating. We represent this user profile in a probabilistic way, which allows us to revise the profile as more training data is becoming available using "conjugate priors", a technique from Bayesian statistics for probability revision.

Syskill & Webert uses a user profile to identify interesting web pages in two ways. First, it can annotate any HTML page with information on whether the user would be interested in visiting each page linked from that page. Second, Syskill & Webert can construct a LYCOS (Mauldin & Leavitt, 1994) query and retrieve pages that might match a user's interest, and then annotate this result of the LYCOS search. Figure 1 shows a Web page on independent rock bands that has been annotated by Syskill and Webert. In this case, the user has e.g. indicated strong interest in the band "The Breetles" (indicated by two thumbs up), a mild interest in "Dead Flower Bloom" (indicated by one thumb up and one thumb down) and no interest in "Middle Passage" (indicated by two thumbs down). The other annotations are the predictions made by Syskill & Webert about whether the user would be interested in each unexplored page. A smiley face indicates that the user hasn't visited the page and Syskill & Webert recommends the page to the user. The international symbol for "no" is used to indicate a page hasn't been visited and the learned user profile suggests the page should be avoided. Following any prediction is a number between 0 and 1 indicating the probability the user would like the page.

In this paper, we first describe how the Syskill & Webert interface is used and the functionality that it provides. Next, we describe the underlying technology for learning a user profile and how we addressed the issues involved in applying machine learning algorithms to classify HTML texts rather than classified attribute-value vectors. Based on results from the application of learning algorithms that do not make use of background knowledge, we suggest to provide the system with an initial profile of a user's interests. We show how this profile can be represented probabilistically and explain the underlying theory of conjugate priors that we use to gradually revise the model to reflect

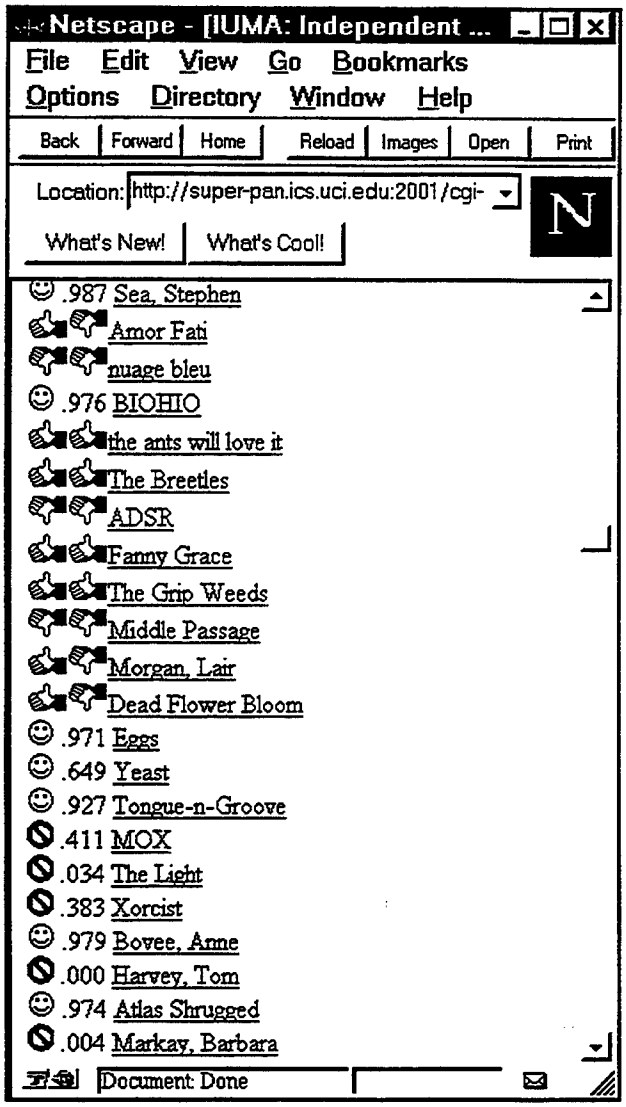


Figure 1: An example of an annotated page

available data. Finally, we describe experiments that compare the accuracy of our new approach with learning algorithms that do not make use of such background knowledge.

Syskill & Webert

Syskill & Webert learns a separate profile for each topic of each user. We decided to learn a profile for user topics rather than users for two reasons. First, we believe that many users have multiple interests and it will be possible to learn a more accurate profile for each topic separately since the factors that make one topic interesting are unlikely to make another interesting. Second, associated with each topic is a URL that we call an index page. The index page is a manually constructed page that typically contains a hundred or more links to other information providers (see e.g. Figure 1). Syskill & Webert allows a user to explore the Web using the index page as a starting point. In one mode of using Syskill & Webert, it learns a profile from the user's ratings of pages and uses this profile to suggest other pages accessible from the index page. To collect ratings, the HTML source of web pages is intercepted, and an additional functionality is added to each page (see Figure 2). This functionality allows the user to rate a page as either hot (two thumbs up), lukewarm (one thumb up and one thumb down), or cold (two thumbs down). The user can return to the index page or switch topics. Furthermore, the user can instruct Syskill & Webert to learn a user-profile for the current topic, make suggestions or consult LYCOS to search the Web.

When a user rates a page, the HTML source of the page is copied to a local file and a summary of the rating is made. The summary contains the classification (hot, cold, or lukewarm), the URL and local file, the date the file was copied (to allow for the bookkeeping that would occur when a file changes), and the page's title (to allow for the production of a summary of the ratings).

Syskill & Webert adds functionality to the page (see Figure 2) for learning a user profile, using this user profile to suggest which links to explore from the index page, and forming LYCOS queries. The user profile is learned by analyzing all of the previous classifications of pages by the user on this topic. If a profile exists, a new profile is created by reanalyzing all previous pages together with any newly classified pages.

Once the user profile has been learned, it can be used to determine whether the user would be interested in another page. However, this decision is made by analyzing the HTML source of a page, and it requires the page to be retrieved first. To get around network delays, we allow the user to prefetch all pages accessible from the index page and store them locally. Once this has been done, Syskill & Webert can learn a new profile and make suggestions about pages to visit quickly. Once the HTML has been analyzed, Syskill & Webert annotates each link on the page with an icon indicating the user's rating or its prediction of the user's rating together with the estimated probability that a user would like the page. The default version of Syskill & Webert uses a simple Bayesian classifier (Duda & Hart, 1973) to determine this probability. Note that these ratings and predictions are specific to one user and do not reflect on how other users might rate the pages.

The extension to Syskill & Webert that we describe in this paper lets a user provide the system with words that are good indicators for a page to be interesting. In addition, the user can specify probability estimates that indicate how likely it is that a certain word would appear in an interesting page. In this paper we describe the way such a predefined user profile is revised as more rated pages are

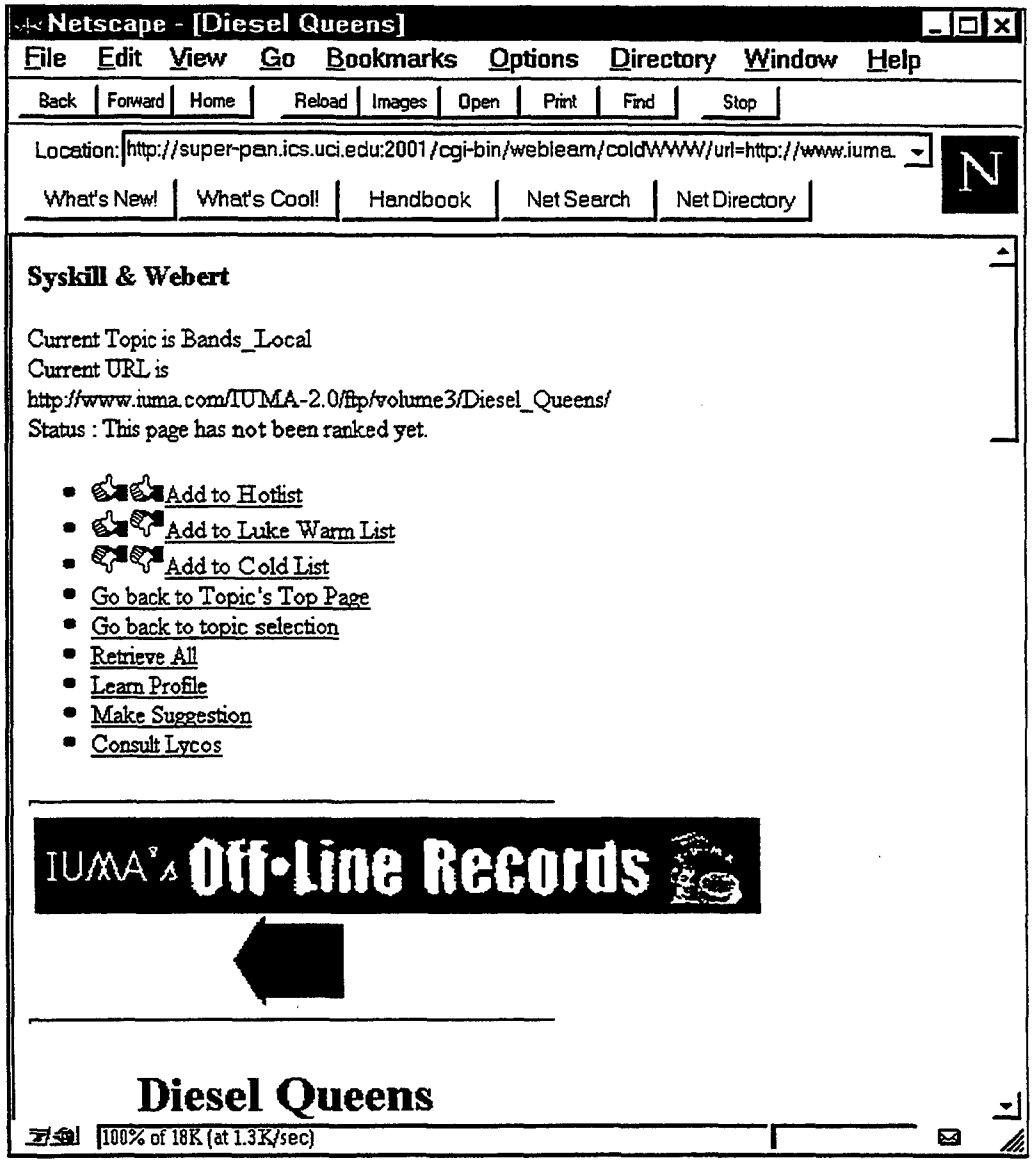


Figure 2: Syskill & Webert interface for rating pages

becoming available, and how we use the profile to classify unseen pages.

Learning a User Profile

Learning algorithms require a set of positive examples of some concepts (such as web pages one is interested in) and negative examples (such as web pages one is not interested in). In this paper, we learn a concept that distinguishes pages rated as *hot* by the user from other pages (combining the two classes *lukewarm* and *cold*, since few pages are rated lukewarm, and we are primarily interested in finding pages a user would consider *hot*). Most learning programs require that the examples be represented as a set of feature vectors. Therefore, we have constructed a method of converting the HTML source of a web page into a Boolean feature vector. Each feature has a Boolean value that indicates whether a particular "word" is present (at least once) or absent in a particular web page. For the purposes of this paper, a word is a sequence of letters, delimited by nonletters. For example, the URL *<A HREF= http://golgi.harvard.edu/biopages/all.html>* contains nine "words" *a, href, http, golgi, harvard, edu, biopages, all,* and *html*. All words are converted to upper case.

Learning a user profile consists of two main steps. First, features are automatically extracted from rated pages, and all rated pages are converted to Boolean vectors containing only those extracted features. Second, learning algorithms are applied to the feature vectors.

Feature Selection

Not all words that appear in an HTML document are used as features. We use an information-based approach, similar to that used by an early version of the NewsWeeder program (Lang, 1995) to determine which words to use as features. Intuitively, one would like words that occur frequently in pages on the hotlist, but infrequently on pages on the coldlist (or vice versa). This is accomplished by finding the expected information gain *(E(W,S))* (e.g., Quinlan, 1986) that the presence or absence of a word (W) gives toward the classification of elements of a set of pages (S):

$$E(W,S) = I(S) - [p(W = present)I(S_{W=present}) + p(W = absent)I(S_{W=absent})]$$

where

$$I(S) = \sum_{c \in \{hot,cold\}} - p(S_c)\log_2(p(S_c))$$

and $P(W = present)$ is the probability that W is present on a page, and ($S_{W=present}$) is the set of pages that contain at least one occurrence of W and S_c are the pages that belong to class c.

Using this approach, we find the set of k most informative words. In the experiments discussed in this paper, we use the 96 most informative words, because previous web page classification experiments with a simple Bayesian classifier resulted in the highest average accuracy over all tested domains for this value of k (Pazzani, Muramatsu, Billsus, 1996). Table 1 shows some of the most informative words obtained from a collection of 140 HTML documents on independent rock bands as an example for features selected by expected information gain.

nirvana	suite	lo
pop	records	rockin
july	jams	songwriting
following	today	vocals
island	tribute	previous
favorite	airplay	noise

Table 1: Some of the words used as features

Some of these selected words seem to be "music related" and might indeed discriminate well between interesting and uninteresting pages (e.g. *nirvana, pop, songwriting, vocals* etc.). In contrast, some words were selected because they were good discriminators on the training data, although it seems unlikely that they will be useful as part of a generally applicable profile to distinguish between interesting and uninteresting pages (e.g. *today, following, previous*). In addition, the user who rated these pages felt that most of the words he would have selected intuitively as good discriminators were not present in the set of automatically selected features. This problem can be addressed with an initial user defined profile.

Learning Algorithms

Once the HTML source for a given topic has been converted to positive and negative examples represented as feature vectors, it is possible to run many learning algorithms on the data. In previous experiments (Pazzani, Muramatsu, Billsus, 1996) we have investigated the accuracy of 5 machine learning algorithms (simple Bayesian classifier, nearest neighbor, PEBLS, decision trees (ID3), and neural nets).

In summary, it appeared that decision tree learners as ID3 are not particularly suited to this problem, as one might imagine since they learn simple necessary and sufficient descriptions about category membership. Although one must be careful not to read too much into averaging accuracies across domains, the naive Bayesian classifier had the highest average accuracy (77.1%). In contrast, PEBLS was 75.2%, a neural net with backpropagation was 75.0%, nearest neighbor was 75.0% and ID3 was 70.6% accurate on average.

We have decided to use the naive Bayesian classifier as the default algorithm in Syskill & Webert for a variety of reasons. It is very fast for both learning and predicting. Its learning time is linear in the number of examples and its prediction time is independent of the number of examples. It is trivial to create an incremental version of the naive Bayesian classifier. It provides relatively fine-grained probability estimates that may be used to rank pages in addition to rating them. For example, on the biomedical domain (see experimental evaluation), we used a leave-one-out testing methodology to predict the probability that a user would be interested in a page. The ten pages with the highest probability were all correctly classified as interesting and the ten pages with the lowest probability were all correctly classified as uninteresting. There were 21 pages whose probability of being interesting was above 0.9 and 19 of these were rated as interesting by the user. There were 64 pages whose probability of being interesting was below 0.1 and only 1 was rated as interesting by the user.

Using Predefined User Profiles

There are two drawbacks of our system that can be addressed by providing a learning algorithm with initial knowledge about a user's interests. First, looking at the features selected by expected information gain revealed that some of the features are irrelevant for discriminating between interesting and uninteresting pages. This occurs most frequently when there is only few training data available. Using lots of irrelevant features makes the learning

task much harder and leads most likely to a lower classification accuracy. Second, the classification accuracy tends to increase very slowly while more training data is becoming available. Since some users might be unwilling to rate lots of pages before the system can give reliable predictions, initial knowledge about the user's interests can be exploited to give accurate predictions even when there are only a few rated pages.

The only way to assess an initial user profile, especially when the training data is sparse, is to elicit it from the user. In Syskill & Webert we are asking the user to provide words that are good indicators for an interesting page and allow him to state as many or few words as he can come up with. In addition, we are asking for words that are good indicators for a page being uninteresting. However, the concept of an "uninteresting webpage" is hard to define, because an "uninteresting webpage" can be anything diverging from the user's interests. Therefore, it might be hard or somewhat unnatural to think of words that are indicators for uninteresting pages, and therefore the user does not have to state any words if he cannot think of any.

Since there might be different levels of how relevant a single word is to the classification of pages, we are also asking the user to provide an estimate of a probability for each word that indicates how likely it is that a page rated as hot (or cold respectively) would contain the word. If the user cannot assess this probability we are using a default value instead. In our current implementation we set this default probability to 0.7, because it is more likely that a feature given by the user correctly discriminates between interesting and uninteresting pages, and therefore a value greater than 0.5 should be used.

The initial user profile is only assessed once for each user and each topic and is then reused in all following sessions. However, a user can redefine his profile if his interests have changed or if he would like to add additional words to the profile.

The initial user profile used in Syskill & Webert consists of a set of probability tables. We associate one probability table with each word stated by the user, where each probability table contains the probabilities for the word appearing (or not appearing) in a page, given that the page was rated hot (or cold respectively). Such a probability table $p(word_i \mid class_j)$ contains 4 probabilities, namely $p(word_i\ present \mid hot)$, $p(word_i\ absent \mid hot)$, $p(word_i\ present \mid cold)$, and $p(word_i\ absent \mid cold)$. If the user provided probability estimates, they become the initial values for the corresponding probability $p(word_i\ present \mid class_j)$ The probability $p(word_i\ absent \mid class_j)$ is set to $1 - p(word_i\ present \mid class_j)$. Since a user would usually not state the same word as being an indicator for hot pages and cold pages, only two probabilities out of the four probabilities are defined by the user. The remaining probabilities can be estimated from the training data.

After an initial user profile has been assessed, it has to be determined to which degree the system should "believe" in the user's estimates. This decision should clearly be correlated to the amount of available training data. While there are initially only a few rated pages available, the system should rely more on the profile given by the user than on hypotheses formed by looking at only a few available rated pages. This is due to the fact, that a few rated pages do not provide enough information to form concept descriptions that are general enough to accurately predict unseen pages. As more training data is becoming available, the system should gradually increase the belief in its own hypotheses and gradually decrease the belief in the initial user profile. The user profile should be gradually revised to form a user profile that fits the training data.

Since an initial user profile is represented as a set of probability tables, the process of revising the user profile consists of gradually updating the given probabilities in a way to make them reflect the seen training examples. A theoretically sound way of doing this is to express the uncertainty in a probability estimate with a conjugate probability distribution. While observing more training data this probability distribution can be updated to reflect both, the changed expected value of the probability, and the increased confidence in the accuracy of the probability estimate.

The probability tables that represent the current profile of the user's interests can be directly used in Syskill & Webert's default classification algorithm, the simple Bayesian classifier.

Conjugate Priors

Conjugate priors are a traditional technique from Bayesian statistics to update probabilities from data (Heckerman, 1995). Using this approach, a probability density function $p(\theta)$ is associated with a random variable Θ whose value θ has to be learned. The probability density function $p(\theta)$ is gradually updated to reflect the observed data.

In our system Θ corresponds to the probability $p(word_i\ present \mid class_j)$, and the observed data corresponds to the events $(word_i\ present\ \&\ class_j)$ and $(word_i\ absent\ \&\ class_j)$. We are uncertain about the value of Θ, and as we observe $word_i$ appearing or not appearing in a document rated as being of $class_j$ we update our probability distribution for Θ. Suppose we observe $(word_i\ present)$ in a document rated as $class_j$. Bayes theorem can now be applied to compute the posterior probability distribution $p(\theta \mid word_i\ present)$:

$$p(\theta \mid word_i\ present) = c\, p(word_i\ present \mid \theta)\, p(\theta)$$
$$= c\theta\, p(\theta)$$

where c is a normalization constant.

Since $p(word_i\ absent\ |\theta)$ is simply $1 - p(word_i\ present\ |\ \theta)$ we can update $p(\theta\ |\ word_i\ absent)$ equally easily:

$$p(\theta|word_i absent) = c\ p(word_i absent|\theta)\ p(\theta)$$
$$= c(1-\theta)\ p(\theta)$$

These two equations can now be combined to compute the posterior probability distribution $p(\theta\ |\ \alpha, \beta)$ where α is the number of times $(word_i\ present\ \&\ class_j)$ is observed and β is the number of times $(word_i\ absent\ \&\ class_j)$ is observed:

$$p(\theta|\alpha,\beta) = c\,\theta^{\alpha}\,(1-\theta)^{\beta}\,p(\theta)$$

A probability distribution given by the equation above is known as a *beta distribution* (since two parameters α, and β are a sufficient statistic to update the distribution). A beta distribution has the property to be *conjugate*, which means that applying Bayes law to a prior beta distribution results in a posterior distribution, which is also a beta distribution.

The effect of additional observed samples on a prior beta distribution is shown in Figure 3 and Figure 4. Figure 3 shows a beta distribution for the parameters $\alpha = 2$, and $\beta = 2$. Since both events were observed 2 times, both events are assumed to be equally likely and so the graph is centered at 0.5 (the expected value of θ, written as $E(\theta)$, is 0.5). The distribution is only based on a small sample size $(\alpha + \beta = 4)$ and has therefore a high variance.

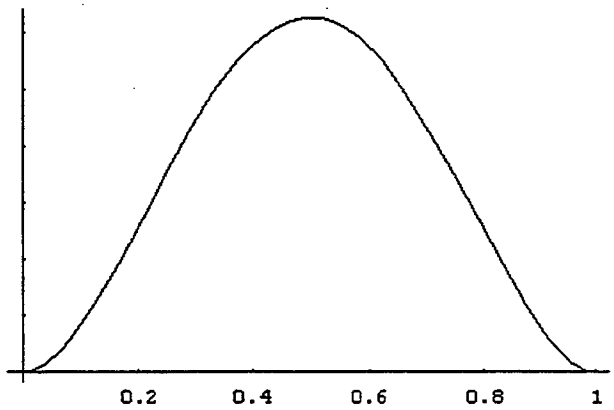


Figure 3: Beta distribution (α = 2, β =2)

Now, suppose that we observe the event corresponding to α 8 times, while we do not observe any events corresponding to β. Figure 4 shows the resulting posterior beta distribution. The posterior distribution is significantly shifted to the right and has a smaller variance (the graph is narrower; it has a lower spread of possible values). The graph gets narrower due to the increased sample size $(\alpha + \beta = 12)$. The sample size therefore directly reflects our confidence in the probability to be estimated.

Using beta distributions our current expectation of θ can be computed easily:

$$E(\theta) = \frac{\alpha}{\alpha+\beta}$$

These properties make the beta distribution a useful distribution for learning.

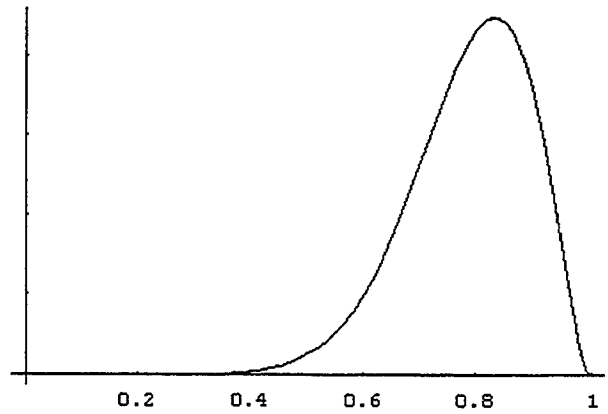


Figure 4: Beta distribution (α = 10, β = 2)

In Syskill & Webert we are using the initial profile given by the user to compute prior beta distributions for the probabilities $p(word_i\ present\ |\ class_j)$. One way to assess a beta distribution is the equivalent-sample-size method (Winkler, 1967). The method is based on the idea to define the equivalent sample size as the number of observations we would have had to have seen starting from complete ignorance in order to have the same confidence in the values of θ. In our current implementation we do not ask the user to provide levels of confidence for each probability estimate he provided. Instead, we are using a default sample size for all initial probability estimates. In the experiments described in this paper we set the equivalent sample size to 50, i.e. we define the "weight" of the user's estimate to be equivalent to observing 50 samples.

The parameters α and β can be determined using the equations:

$$\alpha + \beta = \text{ equivalent sample size}$$
$$\frac{\alpha}{\alpha+\beta} = \text{user's probability estimate}$$

The initial user profile can therefore be used to determine the values of α and β for the prior probability distributions. As more examples are observed, α and β are gradually increased, thus changing the expected value $E(\theta)$ and the confidence in the estimate.

Using the Simple Bayesian Classifier

The probability tables that represent the current profile of the user's interests can be directly used in Syskill & Webert's default classification algorithm, the simple Bayesian classifier (SBC). The SBC (Duda & Hart, 1973) is a probabilistic method for classification. It can be used to deter-

mine the probability that an example i belongs to class C_j given feature values of the example. Applied to Syskill & Webert, this means that we are interested in the probability of a page being interesting or uninteresting, given that it contains or does not contain specific words:

$$P(class_j | word_1 \& word_2 \& \ldots \& word_n)$$

where $word_1$ to $word_n$ are Boolean variables that indicate whether a certain word appears or does not appear in the page. Applying Bayes rule reveals that this probability is proportional to:

$$P(class_j) P(word_1 \& word_2 \& \ldots \& word_n | class_j)$$

Under the assumption that words appearing or not appearing in a page are independent events given the class of the page, $P(word_1 \& word_2 \& \ldots \& word_n | class_j)$ can be expressed as:

$$\prod_i^n P(word_i | class_j)$$

Therefore, the probability of an example belonging to $class_j$ is proportional to:

$$P(class_j) \prod_i^n P(word_i | class_j)$$

To determine the most likely class of an example, the probability of each class is computed, and the example is assigned to the class with the highest probability. The probabilities used in this computation may be estimated from training data. In Syskill & Webert the class priors $P(class_j)$ are estimated from training data only. In contrast, if the initial user profile contains information about the user's estimate of a probability, the corresponding conditional probability $P(word_i | class_j)$ is determined using conjugate priors as described in the previous section and reflects both the initial profile provided by the user and the observed training data.

The assumption of attribute independence is clearly an unrealistic one in the context of text classification. However, the SBC performed well in our experiments, and it also performs well in many domains that contain clear attribute dependence. Domingos and Pazzani (1996) explore the conditions for the optimality of the SBC and conclude that the SBC can even be optimal if the estimated probabilities contain large errors.

Algorithms to Evaluate Feature Selection

In the previous sections we showed how initial probabilities given by the user can be gradually revised to fit the observed training data. However, updating probabilities is only one part of the user profile revision process. Since it is very likely that the user will not mention all the words that are good discriminators between interesting and uninteresting pages, the question arises whether it is possible to increase the classification accuracy by using automatically selected features in addition to the features (words) provided by the user.

In order to assess the effect the feature selection has on the classification accuracy, we implemented three variations of the SBC.

SBC-IFeatures. This is the standard simple Bayesian classifier. It does not make use of features and probabilities provided by the user. All the features are automatically extracted using expected information gain. In the experiments described in this paper we are using 96 features.

SBC-UFeatures. This is the simple Bayesian classifier, operating only on the features that the user provided. The probabilities given by the user are not used. All the probabilities are estimated from data only.

SBC-CFeatures. In this approach, the simple Bayesian classifier is operating on both, the features provided by the user and the features extracted by expected information gain. The same number of features as in the SBC-IFeatures approach is used, where the k features provided by the user replace the k features that have the lowest information gain. Again, no probabilities provided by the user are used.

In addition, we have experimented with SBC variants that add different numbers of automatically extracted features to the user features (see experimental evaluation).

Algorithms to Evaluate Conjugate Priors

In order to assess the effect of revising probabilities on the classification accuracy, we have implemented a set of SBC variants that are only using the features provided by the user.

SBC-User. This approach updates the probabilities given by the user using conjugate priors. Note, that a user does not have to define a complete profile: usually only the probability for a feature being an indicator for a hot page *or* for being an indicator for a cold page is defined, i.e. only half of the associated probability tables are given by the user, and the other half is estimated from data.

SBC-Complete. In order to evaluate the effect of estimating the missing probabilities in the SBC-User approach, we asked users to define a complete user profile. A complete user profile contains two probabilities for every feature: the probability of the feature being an indicator for a hot page, and the probability of a feature being an indicator for a cold page. All provided probabilities are revised using conjugate priors.

User	Topic	URL of topic's index page	Pages
A	Bands (listening)	http://www.iuma.com/IUMA-2.0/olas/location/USA.html	57
B	Bands (reading)	http://www.iuma.com/IUMA-2.0/olas/location/USA.html	154
A	Biomedical	http://golgi.harvard.edu/biopages/medicine.html	127
A	LYCOS (Biomedical)	*not applicable* / results of a *LYCOS* search on biomedicine	54
A	Goats	*not applicable* / results of an *Inktomi* search on goats	70

Table 2: Topics used in our experiments

SBC-Fixed. This approach is similar to SBC-Complete: a completely defined user profile is used, but probabilities are never changed. In fact, apart from estimating the class probabilities *p(hot)* and *p(cold)* there is no learning involved in this variant. However, it provides a basis to assess whether the SBC-User and SBC-Complete approaches do significantly better than using the user's estimates alone.

Furthermore, we have experimented with SBC variants that add different numbers of automatically extracted features to the user features. The user probabilities are revised using conjugate priors, all the remaining probabilities are estimated from data.

Experimental Evaluation

To determine the effect of initial user defined profiles on the classification accuracy, we have had 2 users use the Syskill & Webert interface to rate pages and provide the system with initial user profiles. A total of five different profiles were collected (since one user rated pages on four different topics). The topics are summarized in Table 2 together with the total number of pages that have been rated by the user. Two users rated pages on independent rock bands. One (A) listened to an excerpt of songs, and indicated whether the song was liked. Of course, the machine learning algorithms only analyze the HTML source describing the bands and do not analyze associated sounds or pictures. Another user (B) read about the bands (due to the lack of sound output on the computer) and indicated whether he'd be interested in the band.

Syskill & Webert is intended to be used to find unseen pages the user would like. In order to evaluate the effectiveness of the learning algorithms and initial profiles, it is necessary to run experiments to see if Syskill & Webert's prediction agrees with the users' preferences. Therefore we use a subset of the rated pages for training the algorithm and evaluate the effectiveness on the remaining rated pages. For an individual trial of an experiment, we randomly selected k pages to use as a training set, and reserved the remainder of the data as a test set. From the training set, we found the 96 most informative features, and then recoded the training set as feature vectors to be used by the learning algorithm. Next, the test data was converted to feature vectors using the same features used to convert the training set. Finally, the learned user preferences were used to determine whether pages in the test set would interest the user. For each trial, we recorded the accuracy of the learned preferences (i.e., the percent of test examples for which the learned preferences agreed with the user's interest). We ran 50 paired trials of each algorithm. The learning curves in this section show the average accuracy of each algorithm as a function of the number of training examples.

In our first set of experiments we compared the SBC-IFeatures, SBC-UFeatures, and SBC-CFeatures approaches in order to evaluate the effect of the features given by the user. Note, that no approach in this set of experiments makes use of the probabilities provided by the user. We also compare these results to the baseline accuracy (always guessing *cold* which is the most frequent class in all domains). Figures 5 to 9 show the learning curves for the five domains.

Tables 3 to 6 show the user profiles that were used for the five domains (only four profiles were defined, because the LYCOS and biomedical domains are using the same initial user profile). The two numbers next to each word are the probabilities *p(word present | hot)* and *p(word present | cold)*.

The results show the same pattern of performance for the two bands domains, the biomedicine and the goats domain. While SBC-IFeatures has the lowest classification accuracy, SBC-CFeatures performs consistently better than SBC-IFeatures, and SBC-UFeatures achieves the highest classification accuracy in all four domains. The features selected by the users seem to discriminate very well between interesting and uninteresting pages and outperform the approaches that are based on features selected by expected information gain. In addition, we ran experiments that gradually augmented the features given by the user with features extracted by expected information gain. Even when we added only two additional features (the two highest ranked) the classification accuracy was decreased. The

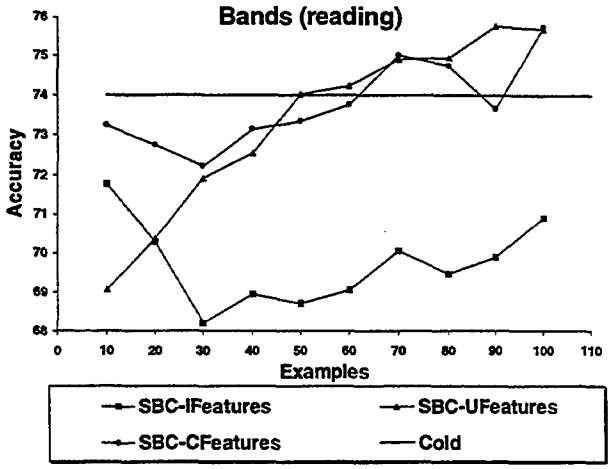


Figure 5: Test accuracy for the bands (reading) domain

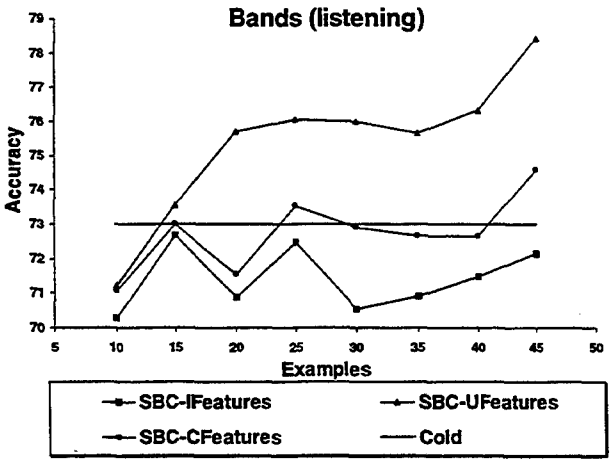


Figure 6: Test accuracy for the bands (listening) domain

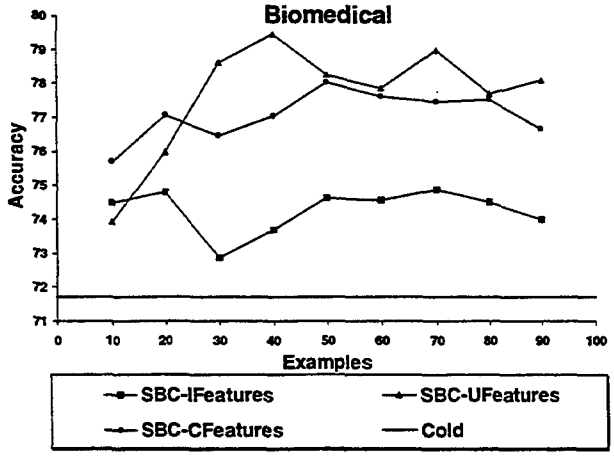


Figure 7: Test accuracy for the biomedical domain

classification accuracy decreased in an almost linear relation to the number of added features, i.e. the more features

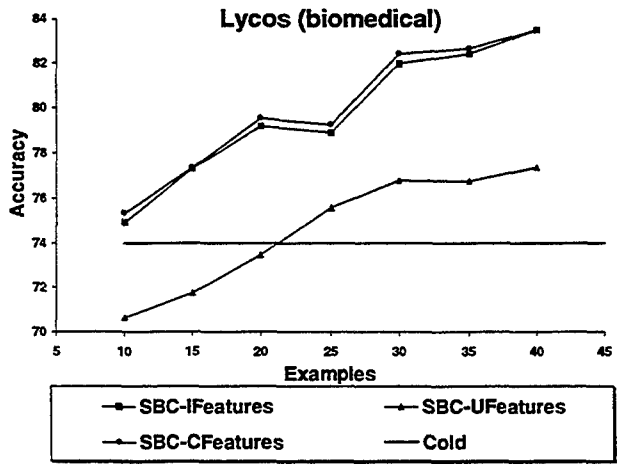


Figure 8: Test accuracy for the LYCOS domain

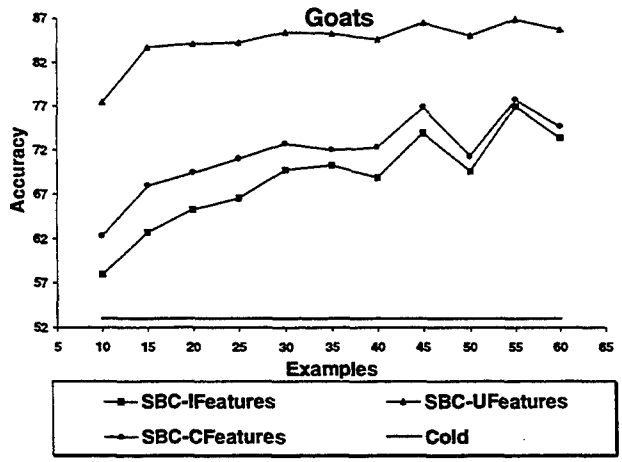


Figure 9: Test accuracy for the goats domain

we added, the lower the classification accuracy.

When we tested the algorithms on the LYCOS domain the results turned out to be very different. The LYCOS domain was generated by a LYCOS query that Syskill & Webert constructed using the profile learned from the biomedical domain. The idea is to let Syskill & Webert rate links returned by LYCOS using the same profile that the query was generated from. Therefore we used the same initial user profile as in the biomedical domain. The user ratings of only the pages retrieved by LYCOS were used to update the initial profile. In this experiment *SBC-UFeatures* performed worse than the other two approaches. The features selected by the user did not seem to be good discriminators for the pages returned by LYCOS. However, the *SBC-CFeatures* approach was not affected very much by the irrelevant user profile, and performed as good as *SBC-IFeatures*.

In summary, it appears that the *SBC-UFeatures* algorithm outperforms the other approaches in cases where the

initial user profile provides features that discriminate well between interesting and uninteresting pages. However, if the user profile is wrong or contains features that do not appear frequently in the pages to be rated, the *SBC-CFeatures* algorithm seems not to be affected negatively to a significant extent.

guitar .9 .5	*guitars .9 .5*	*acoustic .9 .5*
independent .9 .1	*nirvana .9 .1*	*pumpkins .9 .2*
alternative .9 .1	*college .9 .3*	*folk .9 .5*
synthesizer .1 .6	*keyboard .1 .6*	*dance .1 .6*

Table 3: User profile for the bands (reading) domain

acoustic .7 .1	*flute .7 .1*	*she .8 .3*
her .8 .3	*punk .6 .2*	*jazz .7 .2*
bass .7 .5	*south .1 .1*	*cassette .3 .2*
drugs .3 .1		

Table 4: User profile for the bands (listening) domain

grants .7 .2	*database .8 .1*	*genome .6 .3*
molecular .6 .1	*protein .5 .2*	*prediction .9 .1*
classification. 9 1	*structure .6 .2*	*function .6 .1*
webmaster .05 .1	*com .1 .4*	

Table 5: User profile for biomedical and LYCOS domains

dairy .7 .3	*pygmy .7 .3*	*angora .7 .3*
cashmere .7 .3	*milk .7 .3*	*doe .7 .3*
farm .7 .3	*buck .7 .3*	*wether .7 .3*
sheep .7 .3	*animals .7 .3*	*hay .7 .3*
wine .3 .7	*hill .3 .7*	*blows .3 .7*

Table 6: User profile for the goats domain

We also noticed that the classification accuracy on the training set of all the algorithms that make use of the features provided by the user was significantly lower than the training accuracy of algorithms that only use features selected by expected information gain. We interpret this as an indicator that the features given by the user might be impossible to extract automatically from the training data.

In the following set of experiments we compared the SBC-User, SBC-Complete and SBC-Fixed algorithms to the SBC-UFeatures approach in order to evaluate the effect of the probabilities provided by the user. In addition, these experiments show the effect of probability revision using conjugate priors on the classification accuracy. Figures 10 to 14 show the learning curves for the five domains. Again, the results show the same pattern of performance for the two bands domains, the biomedicine and the goats domain. SBC-UFeatures (the best performing approach in the previous set of experiments) is the only algorithm in this set of experiments that does not make use of the probabilities provided by the user. Thus, on average its classification accuracy is significantly below the other approaches. As expected, the accuracy of the SBC-Fixed approach stays approximately constant, i.e. it is independent of the number of training examples. The SBC-User and SBC-Complete algorithms perform about equally well. Interestingly, both approaches exhibit the shape of an ascending learning curve that drifts away from the SBC-Fixed accuracy, which means that the provided probabilities can in fact be revised in a way that increases the classification accuracy. Although the SBC-User approach only uses the probabilities *p(word present | hot)* and estimates the probabilities *p(word present | cold)* from data, its performance is on average not worse than that of SBC-Complete. In some cases SBC-User even outperforms SBC-Complete (e. g. in the goats domain). This result suggests that it is sufficient to specify one probability for each feature. Note that all the approaches that make use of the user's probabilities perform significantly better than the SBC-UFeatures approach. SBC-UFeatures in turn performs better than all approaches that do not make use of the features provided by the user, such as SBC-IFeatures and all the standard machine learning algorithms that we tested in previous experiments (Pazzani, Muramatsu, Billsus, 1996). The difference between the accuracy of the algorithms that make use of the probabilities provided by the user and SBC-UFeatures is most significant when there are only few training examples. This result suggests, that initially defined probabilities are in fact a good way to increase the classification accuracy when the training data is sparse.

Similar to our experiments on the effect of feature extraction, we ran experiments that gradually augmented the features given by the user with features extracted by expected information gain. We used conjugate priors to re-

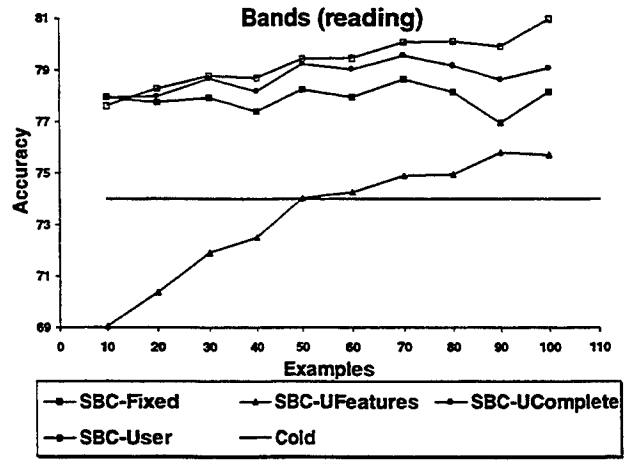


Figure 10: Test accuracy for the bands (reading) domain

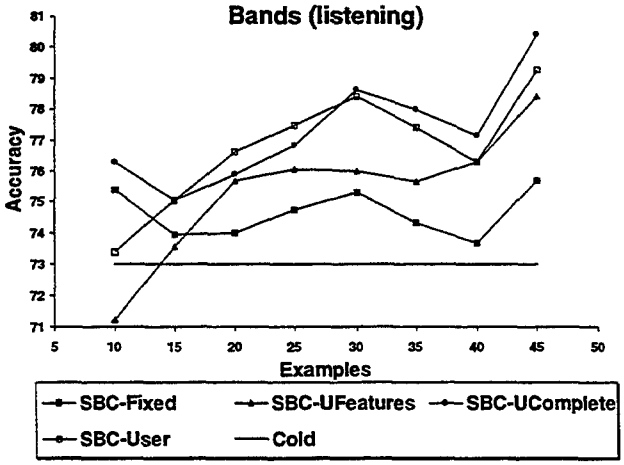


Figure 11: Test accuracy for the bands (listening) domain

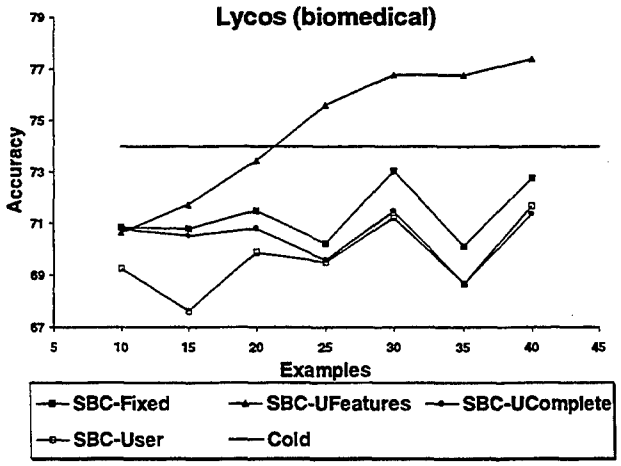


Figure 13: Test accuracy for the LYCOS domain

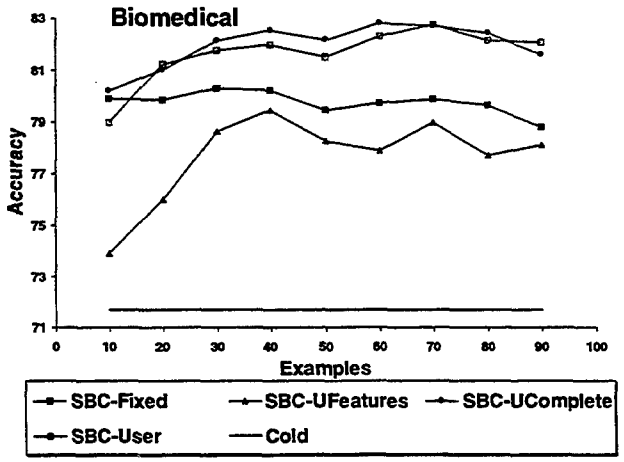


Figure 12: Test accuracy for the biomedical domain

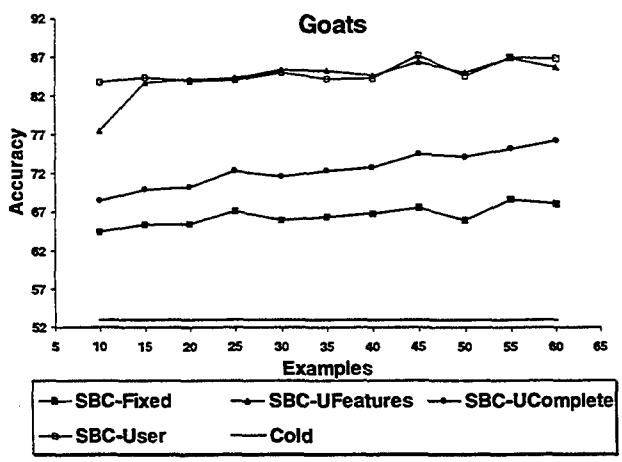


Figure 14: Test accuracy for the goats domain

vise the probabilities given by the user and estimated the remaining probabilities from data. Again, in no case did automatically extracted features added to the features provided by the user increase the classification accuracy. As in our feature extraction experiments, the results for the LYCOS domain differed significantly from the other four domains. Since the features selected by the user do not seem to discriminate well between pages returned by LYCOS, the classification accuracy cannot be improved by providing user defined probabilities.

All in all, it seems as if initially provided probabilities can significantly increase the classification accuracy in cases where the given features discriminate well between interesting and uninteresting pages. The given probabilities can be revised in order to reflect the available data, and the classification accuracy increases.

Future Work

The experiments presented in this paper have shown that the selection of features is paramount to the achievable classification accuracy. We are planning to use linguistic and hierarchical knowledge in addition to expected information gain in order to extract features that, when added to the features given by the user, increase the classification accuracy.

Currently, words in the pages are used as features without any knowledge of the relationship between words such as "protein" and "proteins." Linguistic routines such as stemming (i.e., finding the root forms of words) may in addition be helpful, since a smaller number of more informative features would be extracted. Semantic knowledge, e.g. the relationship between "pigs" and "swine", may also prove useful. Similarly, knowing that "gif," "jpg" and "jpeg" are all extensions of graphic files would facilitate Syskill & Webert learning that a user has a preference

for (or against) pages with in-line graphics.

The results of the approaches described in this paper are highly dependent on the profiles provided by the users. Clearly, more data from different users has to be collected, to confirm that our results have general validity.

Conclusions

We have introduced an agent that collects user evaluations of the interestingness of pages on the World Wide Web. We have shown that a user profile may be learned from this information and that this user profile can be used to determine what other pages might interest the user.

A predefined user profile can significantly increase the classification accuracy on previously unseen pages. The best results were achieved when only the features that were provided by the user were used. Using this approach, the classification accuracy can still be increased by revising the user defined probabilities using conjugate priors.

Since the classification accuracy on the training set significantly decreased for all tested algorithms that made use of a user defined profile, we think that the features given by the user cannot be extracted automatically from the training set by statistical means alone.

Acknowledgments

The research reported here was supported in part by NSF grant IRI-9310413 and ARPA grant F49620-92-J-0430 monitored by AFOSR.

References

Domingos, P., and Pazzani, M. 1996. Beyond Independence: Conditions for the Optimality of the Simple Bayesian Classifier. To appear in the *Proceedings of the International Conference on Machine Learning*.

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

Heckerman, D. 1995. A Tutorial on Learning with Bayesian Networks, Technical Report, MSR-TR-95-06, Microsoft Corporation.

Lang, K. 1995. NewsWeeder: Learning to filter Netnews. In Proceedings of the Twelfth International Conference on Machine Learning, 331-339. Lake Tahoe, Calif.: Morgan Kaufmann.

Mauldin, M. L., and Leavitt, J. R. 1994. Web Agent Related Research at the Center for Machine Translation. Proceedings of the ACM Special Interest Group on Networked Information Discovery and Retrieval (SIGNIDR-94).

Pazzani, M., Muramatsu J., and Billsus, D. 1996. Syskill & Webert: Identifying interesting web sites. To appear in the *Proceedings of the National Conference on Artificial Intelligence*, Portland, OR.

Quinlan, J. R. 1986. Induction of decision trees. *Machine Learning*, 1, 81-106.

Winkler, R. 1967. The assessment of prior distributions in Bayesian analysis. *American Statistical Association Journal*, 62:776-800.

"CAM-BRAIN" ATR's BILLION NEURON ARTIFICIAL BRAIN PROJECT

Hugo de Garis

Brain Builder Group, Evolutionary Systems Department,
ATR Human Information Processing Research Laboratories,
2-2 Hikaridai, Seika-cho, Soraku-gun, Kansai Science City, Kyoto, 619-02, Japan.
tel. + 81 7749 5 1079, fax. + 81 7749 5 1008, degaris@hip.atr.co.jp

Abstract

This work reports on progress made in the first 3 years of ATR's "CAM-Brain" Project, which aims to use "evolutionary engineering" techniques to build/grow/evolve a RAM-and-cellular-automata based artificial brain consisting of thousands of interconnected neural network modules inside special hardware such as MIT's Cellular Automata Machine "CAM-8", or NTT's Content Addressable Memory System "CAM-CAM". The states of a billion (later a trillion) 3D cellular automata cells, and millions of cellular automata rules which govern their state changes, can be stored relatively cheaply in giga(tera)bytes of RAM. After 3 years work, the CA rules are almost ready. MIT's "CAM-8" (essentially a serial device) can update 200 million CA cells a second. It is likely that NTT's "CAM-CAM" (Cellular Automata on Content Addressable Memory) is essentially a massively parallel device, and will be able to update a ***hundred billion*** CA cells a second. Hence all the ingredients will soon be ready to create a revolutionary new technology which will allow thousands of evolved neural network modules to be assembled into artificial brains. This in turn will probably create not only a new research field, but hopefully a whole new industry, namely "brain building". Building artificial brains with a billion neurons is the aim of ATR's 8 year "CAM-Brain" research project, ending in 2001.

1. Introduction

ATR's CAM-Brain project resulted from the experience of the author's thesis work, in which he evolved neural net modules (using concatenated bit-string weights) to control the behavior of a simulated quadruped called "LIZZY", which could walk straight, turn left, turn right, peck at food and mate (de Garis 1994). Each of these behaviors was controlled by the time varying outputs of a single evolved neural network module, and applied to the angles of the leg components of LIZZY. (As far as he is aware, the author was the first person to evolve neural net dynamics (de Garis 1991), (in the form of walking stick-legs "Walker")). Switching between behaviors involved taking the outputs from one neural net module and feeding them into the inputs of the next module. The next step was to evolve neural net detectors, e.g. for frequency, signal strength, signal strength difference, etc. Finally, neural net "production rule" modules were evolved which could map conditional inputs from detectors to output behaviors. Thus an "intelligent" artificial creature was built, which could detect prey, mates and predators, and then approach and eat or mate, or turn away and flee.

Virtually every neural net that the author tried to evolve, evolved successfully. *The evolution of these fully connected neural network modules proved to be a very powerful technique.* This success made a deep impression on the author, reinforcing his dream of being able to build much more complex artificial nervous systems, even artificial brains. However, every time the author added a neural net module to the Lizzy simulation, its speed on the screen was slowed (on a Mac II computer). Gradually, the necessity dawned on the author that some kind of evolvable hardware solution (de Garis 1993) would be needed to evolve large numbers of neural net modules and at great speed (i.e. electronic speed) in special machines the author calls "Darwin Machines" [de Garis 1993]. Evolving artificial brains directly in hardware remains the ultimate future goal of the author, but in the meantime (since the field of evolvable hardware (EHW, E-Hard) is today only in its infancy), the author compromises by using cellular automata to grow/evolve neural nets in large numbers in RAM, which is cheap and plentiful. (It is now possible to have a gigabyte (a billion byes) of RAM in one's work-station). By using cellular automata based neural nets which grow and evolve in gigabytes of RAM, it should be possible to evolve large numbers (tens of thousands) of neural net modules, and then assemble them (or even evolve their interconnections) to build an artificial brain. The bottleneck is the speed of the processor which updates the CA cells. State of the art in such processors is MIT's "CAM-8" machine, which can update 200,000,000 CA cells a second.

Recently, it has been suggested by the author's ATR colleague Hemmi, that NTT's Content Addressable Memory System "CAM-CAM" (which should be ready by the end of 1997) might be able to update CA cells at a rate *thousands* of times faster than the MIT machine, i.e. at a *hundred billion* CA cells per second. NTT's machine is massively parallel. Hemmi and his programmer assistant Yoshikawa are now (December 1995) busily engaged in writing software to convert the author's CA rules (in 2D form) into Boolean expressions suitable for the NTT machine. If they succeed in applying this machine to CAM-Brain, then a new era of brain building can begin, because the ability to evolve thousands of neural net modules would become realistic and very practical (for example, to evolve a neural net module inside a cubic space of a million CA cells, i.e. 100 cells on a side, at a hundred billion cells a second, would take at most about 500 clock cycles, i.e. about five milliseconds. *So the evolution of a population of 100 chromosomes over 100 generations could all be done in about one minute.)* All the essential ingredients for brain building would be available (lots of RAM, the CA rules, and fast CA processors). Even if Hemmi does not succeed, then a new machine can be designed to be thousands of times faster than the CAM-8 machine. The author believes the CAM-Brain breakthrough is either less than a year away, or at most only a few years away (the time necessary to design and build a "Super-CAM" machine, probably with the help of NTT).

The above gives an overview of the CAM-Brain research project. What now follows is a more detailed description of CAM-Brain, showing how one grows and evolves CA based neural net modules in 2D and 3D. We begin with the essential idea. Imagine a 2D CA trail which is 3 cells wide (e.g. Fig. 2). Down the middle of the trail, send growth signals. When a growth signal hits the end of the trail, it makes the trail extend, or turn left, or right, or split etc., depending upon the nature of the signal (e.g. see Figs. 3-6). It was the author who hand coded the CA rules which make these extensions, turns, splits etc. happen. The CA rules themselves are *not* evolved. It is the *sequence* of these signals (fed continuously over time into an initialized short trail) that is evolved. This sequence of growth signals is the "chromosome" of a genetic algorithm, and it is this sequence that maps to a cellular automata network. When trails collide, they can form "synapses" (e.g. see Fig. 7). Once the CA network has been formed in the initial "growth phase", it is later used in a second "neural signaling phase". Neural signals move along CA-based axons and dendrites, and across synapses etc. The CA network is made to behave like a conventional artificial neural network (see Fig. 11). The outputs of some of the neurons of the complex recurrent networks which result can be used to control complex time dependent behaviors whose fitnesses can be measured. These fitness values can be used to drive the evolution. By growing/evolving thousands of neural net modules and their interconnections in an incremental evolutionary way, it will be possible to build artificial brains. According to the CAM developers at MIT, it is likely that the next generation of CAMs will achieve an increase in performance of the order of thousands, within 5 years. However, to be able to evolve a billion neuron artificial brain by 2001 (ATR's goal), it is likely that a "nano-CAM" machine (i.e. one which uses nano-scale electronic speeds and densities) will need to be developed. To this end, we are collaborating with an NTT researcher who has developed a nanoscale electronics device, who wants to combine huge numbers of them to behave as nano-scale cellular automata machines.

In the summer of 1994, a two dimensional CAM-Brain simulation was completed which required 11,000 hand crafted CA state transition rules. It was successfully applied to the evolution of maximizing the number of synapses, outputting an arbitrary constant neural signal value, outputting a sine wave of a desired arbitrary period and amplitude and to the evolution of a simple artificial retina which could output the vector velocity of a "white line" which "moved" across an array of "detector" neurons. Work on the 3D simulation should be completed by early 1996, and is expected to take about 150,000 hand crafted CA rules. The Brain Builder Group of ATR took possession of one of MIT's CAM8 machines in the fall of 1994. At the time of writing (December 1995) the porting of the 2D rules from a Sparc20 workstation to the CAM8 is nearing completion. If the porting of the rules of the 3D simulation to this machine is not possible, then a "SuperCAM" machine will be designed specifically for CAM-Brain, with the collaboration of the Evolutionary Technologies (ET) group of NTT, with whom our Brain Builder group of ATR's Evolutionary Systems (ES) group, collaborates closely. The complexity of CAM-Brain will make it largely undesignable, so a (directed) evolutionary approach called "evolutionary engineering" is being used. Neural networks based on cellular automata (Codd 1968), can be grown and evolved at electronic speeds inside state of the art cellular automata machines, e.g. MIT's "CAM8" machine, which can update 200 million cells per second (Toffoli & Margolus 1990). Since RAM is cheap, gigabytes of RAM can be used to store the states of the CA cells used to grow the neural networks. CA based neural net modules are evolved in a two phase process. Three cell wide CA trails are grown by sending a sequence of growth signals (extend, turn left, turn right, fork left, fork right, T fork) down the middle of the trail. When an instruction hits the end of the trail it executes its function. This sequence of growth instructions is treated as a chromosome in a Genetic Algorithm (Goldberg 1989) and is evolved. Once gigabytes of RAM and electronic evolutionary speeds can be used, genuine brain building, involving millions and later billions of artificial neurons, becomes realistic, and should become concrete within a year or two. The CAM-Brain Project should revolutionize

the fields of neural networks and artificial life, and in time help create a new specialty called "Brain Building", with its own conferences and journals.

This work consists of the following sections. Section 2 describes briefly the idea of "Evolutionary Engineering", of which the CAM-Brain Project is an example. Section 3 describes how neural networks can be based on cellular automata (Codd 1968), and evolved at electronic speeds. Section 4 presents some of the details of CAM-Brain's implementation. Section 5 shows how using cellular automata machines will enable millions of artificial neural circuits to be evolved to form an artificial brain. Section 6 discusses changes needed for the 3D version of CAM-Brain. Section 7 deals with recent work. Section 8 deals with future work and section 9 summarizes.

2. Evolutionary Engineering

Evolutionary Engineering is defined to be *"the art of using evolutionary algorithms (such as genetic algorithms (Goldberg 1989)) to build complex systems."* This work reports on the idea of evolving cellular automata based neural networks at electronic speeds inside cellular automata machines. This idea is a clear example of evolutionary engineering. Evolutionary engineering will be increasingly needed in the future as the number of components in systems grows to gargantuan levels. Today's nano-electronics for example, is researching single electron transistors (SETs) and quantum dots. Probably within a decade or so, humanity will have full blown nanotechnology (molecular scale engineering), which will produce systems with a trillion trillion components [Drexler 1992]. The potential complexities of such systems will be so huge, that designing them will become increasingly impossible. However, what is too complex to be humanly designable, might still be buildable, as this work will show. By using evolutionary techniques (i.e. evolutionary engineering), it is often still possible to *build* a complex system, even though one does not understand how it functions. This arises from the notion of the "complexity independence" of evolutionary algorithms, i.e. so long as the (scalar) fitness values which drive the evolution keep increasing, the internal complexity of the evolving system is irrelevant. This means that it is possible to successfully evolve systems which function as desired, but which are to complex to be designable. The author believes that this simple idea (i.e. the complexity independence of evolutionary algorithms) will form the basis of most 21st century technologies (dominated by nanotechnology (Drexler 1992)). Thus, evolutionary engineering can "extend the barrier of the buildable", but may not be good science, because its products tend to be black boxes. However, confronted with the complexity of trillion trillion component systems, evolutionary engineering may be the only viable method to build them.

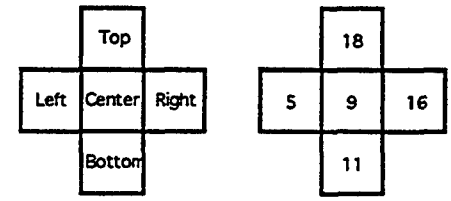
3. Cellular Automata Based Neural Networks

Building an artificial brain containing billions of artificial neurons is probably too complex a task to be humanly designable. The author felt that brain building would be a suitable task for the application of evolutionary engineering techniques. The key ideas are the following. Use evolutionary techniques to evolve neural circuits in some electronic medium, so as to take advantage of electronic speeds. The medium chosen by the author was that of cellular automata (CA) (Codd 1968), using special machines, called "Cellular Automata Machines (CAMs)", which can update hundreds of millions of CA cells a second (Toffoli & Margolus 1990).

CAMs can be used to evolve the CA based neural networks at electronic speeds. The states of the cellular automata cells can be stored in RAM, which is cheap, so one can have gigabytes of RAM to store the states of billions of CA cells. This space is large enough to contain an artificial brain. MIT's Information Mechanics Group (Toffoli and Margolus) believe that within a few years it will be technically possible to update a trillion CA cells in about 0.1 nanoseconds (p221, Toffoli & Margolus 1990). Thus, if CA state transition rules can be found to make CA behave like neural networks, and if such CA based networks prove to be readily evolvable, then a potentially revolutionary new technology becomes possible. The CAM-Brain Project is based on the above ideas and fully intends to build artificial brains before the completion of the project in 2001. The potential is felt to be so great that it is likely that a new specialty will be formed, called "Brain Building".

For the first 18 months of the CAM-Brain Project, the author simulated a two dimensional version of CAM-Brain on a Sparc 10 workstation. This work was completed in the summer of 1994. The 2D version was used briefly (before work on the 3D version was started) to undertake some evolutionary tests, whose results will be presented in the next section. The 2D version served only as a feasibility and educational device. Since trails are obliged to collide in 2D, the 2D version was not taken very seriously. Work was begun rather quickly on the more interesting 3D version almost immediately after the 2D version was ready. Proper evolutionary tests will be undertaken once the 3D version is ready, which should be by early 1996. To begin to understand how cellular automata (Codd 1968) can be used as the basis for the growth and evolution of neural networks, consider Fig. 1 which shows an example of a 2D CA state transition rule, and Fig. 2 which shows a 2D CA trail, 3 cells wide. All cells in a CA system update the state of their cells

synchronously. The new state of a given cell depends upon its present state and the states of its nearest neighbors. Down the middle of the 3 cell wide CA trail, move "signal or growth cells" as shown in Fig. 2 As an example of a state transition rule which makes a signal cell move to the right one square, consider the right hand most signal cell in Fig. 2, which has a state of 5. The cell immediately to its right has a state of 1, which we want to become a 5. Therefore the 2D state transition rule to turn the 1 into a 5 is 1.2.2.2.5-->5. These signal or growth cells are used to generate the CA trails, by causing them to extend, turn left or right, split left or right, and Tsplit. When trails collide, they can form synapses. It is the sequence of these signal cells which determines the configuration of the CA trails, thus forming a CA network. It is these CA trails which later are used as neural network trails of axons and dendrites. Neural signals are sent down the middle of these CA trails. Thus there are two major phases in this process. Firstly, the CA trails are grown, using the sequence of signal cells. Secondly, the resulting CA trail network is used as a neural network, whose fitness at controlling some system can be measured and used to evolve the original growth sequence. To make this more explicit, it is the sequence of growth cells which is evolved. By modifying the sequence, one alters the CA network configuration, and hence the fitness of the configuration when it functions as a neural net in the second phase. From a genetic algorithm (GA) point of view, the format of the GA "chromosome" is the sequence of integers which code for the signaling or growth instructions. By mutating and crossing over these integers, one obtains new CA networks, and hence new neural networks. By performing this growth at electronic speeds in CAMs, and in parallel, with one CAM per GA chromosome, and attaching a conventional programmable microprocessor to each CAM to measure the user defined fitness of the CA based neural circuit, one has a means to evolve large numbers of neural modules very quickly. Using CAMs to evolve neural circuits, is an example of a type of machine that the author labels a "Darwin Machine", i.e. one which evolves its own structure or architecture. A related idea of the author concerns the concept of "Evolvable Hardware (EHW)" (de Garis 1993) where the software instructions used to configure programmable logic devices (PLDs) are treated as chromosomes in a Genetic Algorithm (Goldberg 1989). One then rewrites the circuit for each chromosome.



CTRBL -> Cnext 9.18.16.11.5 -> 4

Fig. 1 A 2D CA State Transition Rule

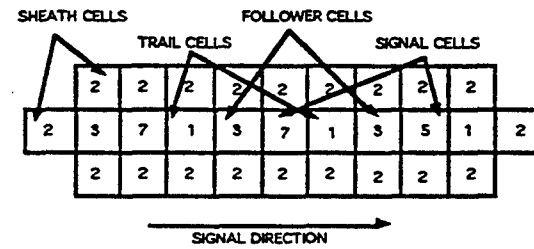


Fig. 2 Signal Cells Move Along a Cellular Automata Trail

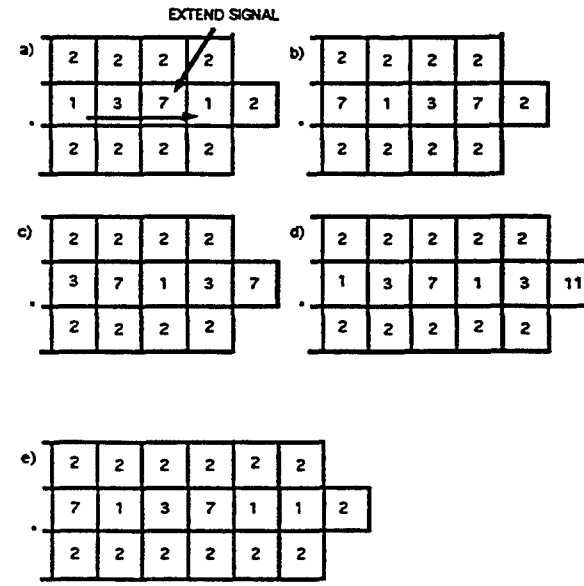


Fig. 3 Extend the Trail

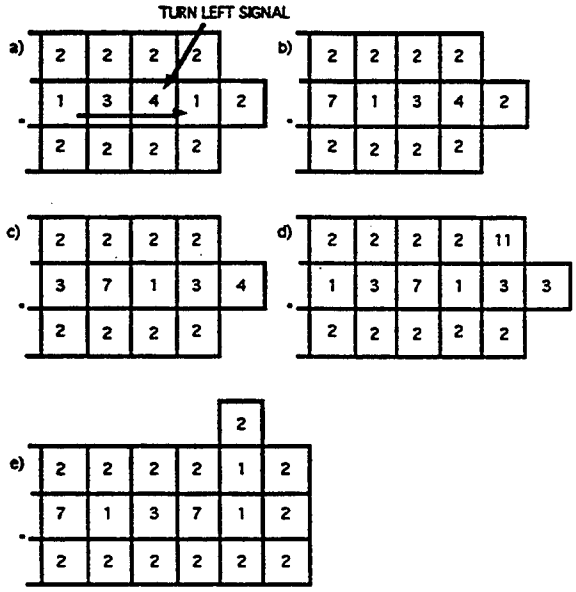


Fig. 4 Turn Trail Left

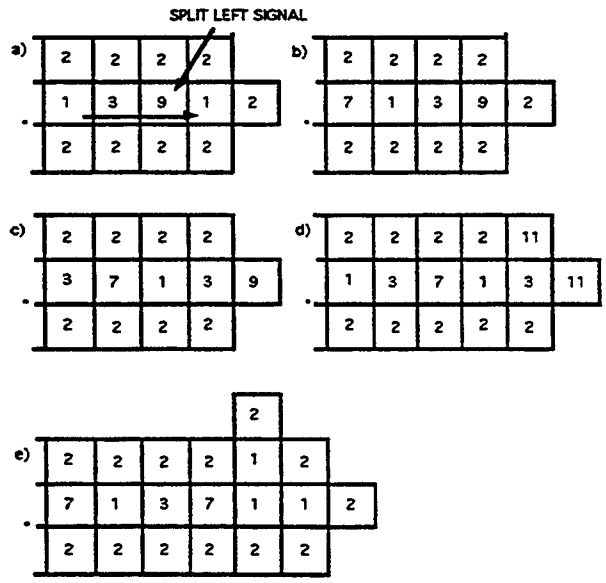


Fig. 5 Split Trail Left

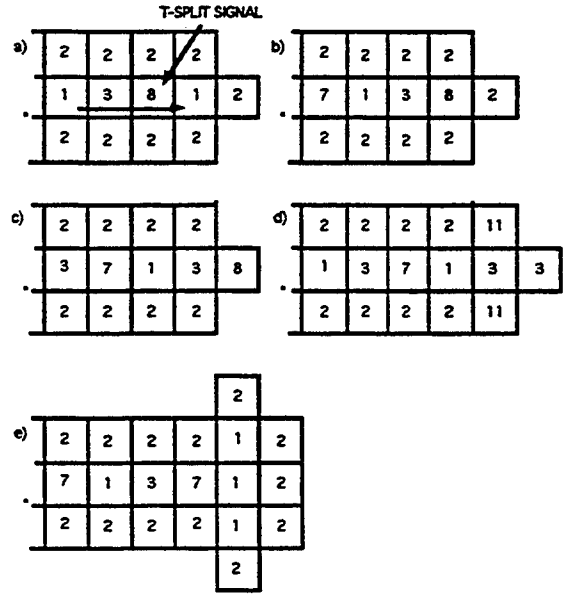


Fig. 6 T-Split Trail

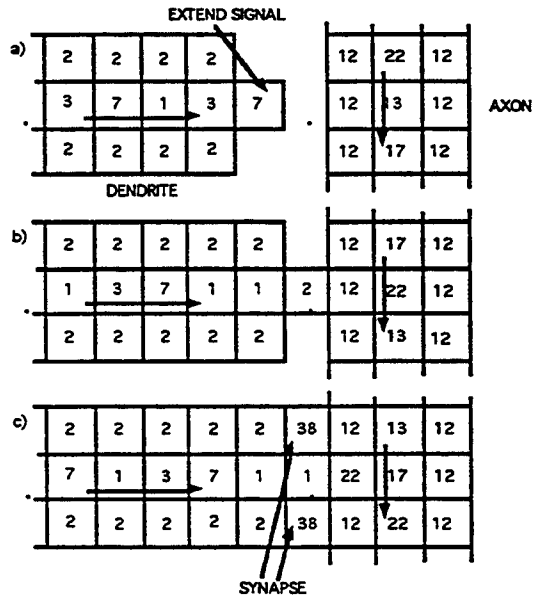


Fig. 7 Dendrite to Axon Synapsing

4. Further Details

This section provides further details on the implementation of the CA based neural networks. There are three kinds of CA trails in CAM-Brain, labeled dendrites, excitatory axons and inhibitory axons, each with their own states. Whenever an axon collides with a dendrite or vice versa, a "synapse" is formed. When a dendrite hits an excitatory/inhibitory axon or vice versa, an excitatory/inhibitory synapse is formed. An inhibitory synapse reverses the sign of the neural signal value passing through it. An excitatory synapse leaves the sign unchanged. Neural signal values range between -240 and +240 (or their equivalent CA states, ranging from 100 to 580). The value of a neural signal remains unchanged as it moves along an axon, but as soon as it crosses a synapse into a dendrite, the signal value (i.e. signal strength) begins to drop off linearly with the distance it has to travel to its receiving neuron. Hence the signal strength is proportional to the distance between the synapse and the receiving neuron. Thus the reduction in signal strength acts like a weighting of the signal by the time it reaches the neuron. But, this distance is evolvable, hence indirectly, the weighting is evolvable. CAM-Brain is therefore equivalent to a conventional artificial neural network, with its weighted sums of neural signal strengths. However, in CAM-Brain there are time delays, as signals flow through the network. When two or three dendrite signals collide, they sum their signal strengths (within saturated upper or lower bounds).

When implementing the 2D version of CAM-Brain, it soon became noticeable that there were many many ways in which collisions between CA trails could occur. So many, that the author became increasingly discouraged. It looked as though it would take years of handcoding the CA state transition rules to get CAM-Brain to work. The intention was to have rules which would cover every collision possibility. Eventually a decision was made to impose constraints on the ways in which CA trails could grow. The first such constraint was to make the trails grow on a grid 6 cells or squares (cubes) on a side. This process (called "gridding") sharply reduces the number of collision types. It also has a number of positive side effects. One is that in the neural signaling phase, neural signals arrive synchronously at junction points. One no longer needs to have to handcode rules for phase delays in neural signaling summation. By further imposing that different growth cells advance the length of the trails by the same number of squares, one can further reduce the number of collision types. With synchrony of growth and synchrony of signaling and gridding, it is possible to cover all possible types of collisions. Nevertheless, it still took over 11000 rules to achieve this goal, and this was only for the 2D version. The 3D version is expected to take about 150,000 rules, but due to the experience gained in working on the 2D version, and to the creation of certain software productivity tools, the 3D version should be completed by early 1996

Considering the fact that the 2D version takes 11,000 rules, it is impossible in this short work to discuss all the many tricks and strategies that are used to get CAM-Brain to work. That would require a book (something the author is thinking seriously about writing, if he ever makes time to do it). However, some of the tricks will be mentioned here. One is the frequent use of "gating cells", i.e. cells which indicate the direction that dendrite signals should turn at junctions to head towards the receiving neuron. To give these gating cells a directionality, e.g. a "leftness" or a "rightness", special marker cells are circulated at the last minute, after the circuit growth is stabilized. Since some trails are longer than others, a sequence of delay cells are sent through the network after the growth cells and before the marker cells. Without the delay cells, it is possible that the marker cells pass before synapses are formed.

Once the 2D simulation was completed (before the CAM8 was delivered) several brief evolutionary experiments using the 2D version were undertaken. The first, was to see if it would be possible to evolve the number of synapses. Figs. 9, 10, 11 show the results of an elite chromosome evolved to give a large number of synapses. Fig. 9 shows early growth. Fig. 10 shows completed growth, and Fig. 11 shows the neural signaling phase. In this experiment, the number of synapses increased steadily. It evolved successfully. The next experiment was to use the neural signaling to see if an output signal (tapped from the output of one of the neurons) could evolve to give a desired constant value. This evolved perfectly. Next, was to evolve an oscillator of a given arbitrary frequency and amplitude, which did evolve, but slowly (it took a full day on a Sparc10 workstation). Finally, a simple retina was evolved which output the two component directional velocity of a moving "line" which passed (in various directions) over a grid of 16 "retinal neurons". This also evolved but even more slowly. The need for greater speed is obvious.

The above experiments are only the beginning. The author has already evolved (not using CAs) the weights of recurrent neural networks as controllers of an artificial nervous system for a simulated quadrupedal artificial creature. Neural modules called "GenNets" (de Garis 1990, 1991) were evolved to make the creature walk straight, turn left or right, peck at food, and mate. GenNets were also evolved to detect signal frequencies, to generate signal frequencies, to detect signal strengths, and signal strength differences. By using the output of the detector GenNets, it was possible to switch motion behaviors. Each behavior had its own separately evolved GenNet. By switching between a library of GenNets (i.e. their corresponding evolved weights) it was possible to get the artificial creature to behave in interesting ways.

It could detect the presence and location of prey, predators and mates and take appropriate action, e.g. orientate, approach, and eat or mate, or turn away and flee. However, every time the author added another GenNet, the motion of the simulated creature slowed on the screen. The author's dream of being able to give a robot kitten some thousand different behaviors using GenNets, could not be realized on a standard monoprocessor workstation. Something more radical would be needed. Hence the motivation behind the CAM-Brain Project.

5. A Billion Neurons in a Trillion Cell CAM by 2001

Fig. 8a shows some estimated evolution times for 10 chromosomes over 100 generations for a Sparc 10 workstation, a CAM8, and a CAM2001 (i.e. a CAM using the anticipated electronics of the year 2001) for a given application. In the 2D version of CAM-Brain, implemented on a Sun Sparc 10 workstation, it takes approximately 3.4 minutes to grow a stable cellular automata network consisting of only four neurons. It takes an additional 3.2 minutes to perform the signaling on the grown network, i.e. a total growth-signaling time to measure the fitness of a chromosome of 6.6 minutes. This time scales linearly with the number of artificial neurons in the network. If one uses a population of 10 chromosomes, for 100 generations, the total evolution time (on a Sparc 10) is 100*10*6.6 minutes, i.e., 110 hours, or 4.6 days. This is obviously tediously slow, hence the need to use a CAM. MIT's CAM8 (Toffoli & Margolus 1990) can update 25 million cellular automata cells per second, per hardware module. A CAM8 "box" (of personal computer size) contains eight such modules, and costs about $40,000. Such boxes can be connected blockwise indefinitely, with a linear increase in processing capacity. Assuming an eight module box, how quickly can the above evolution (i.e. 100 generations, with a population size of 10) be performed? With eight modules, 200 million cell updates per second is possible. If one assumes that the 2D CA space in which the evolution takes place is a square of 100 cells on a side, i.e., 10,000 cells, then all of these cells can be (sequentially) updated by the CAM8 box in 50 microseconds. Assuming 1000 CA clock cycles for the growth and signaling, it will take 50 milliseconds to grow and measure the fitness of one chromosome. With a population of 10, and 100 generations, total CAM8 evolution time for a four neuron network will be 50 seconds, i.e. about one minute, which is roughly 8000 times faster. Using the same CAM8 box, and a 3D space of a million cells, i.e. a cube of 100 cells on a side, one could place roughly 40 neurons. The evolution time will be 100 times as long with a single CAM8 box. With 10 boxes, each with a separate microprocessor attached, to measure the fitness of the evolved network, the evolution time would be about eight minutes. Thus for 1000 neurons, the evolution would take about 3.5 hours, quite an acceptable figure. For a million neurons, the evolution time would be nearly five months. This is still a workable figure. Note, of course, that these estimates are lower bounds. They do not include the necessary human thinking time, and the time needed for sequential, incremental evolution, etc. However, since the CAM-Brain research project will continue until the year 2001, we can anticipate an improvement in the speed and density of electronics over that period. Assuming a continuation of the historical doubling of electronic component density and speed every two years, then over the next eight years, there will be a 16-fold increase in speed and density. Thus the "CAM-2001" box will be able to update at a rate of 200*16*16 million cells per second. To evolve the million neurons above will take roughly 13.6 hours. Thus to evolve a billion neurons, will take about 19 months, again a workable figure. But, if a million neurons can be successfully evolved, it is likely that considerable interest will be focused upon the CAM-Brain approach, so that more and better machines will be devoted to the task, thus reducing the above 19-month figure. For example, with 100 machines, the figure would be about two months. The above estimates are summarized in Figure 8a. These estimates raise some tantalizing questions. For example, if it is possible to evolve the connections between a billion artificial neurons in a CAM2001, then what would one want to do with such an artificial nervous system (or artificial brain)? Even evolving a thousand neurons raises the same question.

Sparc10	CAM8	CAM8	CAM8	CAM8	CAM2001	CAM2001
10000 CA cells	10000 CA cells	1 million CA cells	25 million CA cells	25 billion CA cells	25 billion CA cells	25 trillion CA cells
4 neurons	4 neurons	40 neurons	1000 neurons	1 million neurons	1 million neurons	1 billion neurons
1 Sparc10	1 CAM8	10 CAM8s	10 CAM8s	10 CAM8s	10 CAM2001s	100 CAM2001s
4.6 days	50 seconds	8 minutes	3.5 hours	5 months	13.6 hours	2 months

Fig. 8a Evolution Times for Different Machines & CA Cell, Neuron & Machine Numbers

48*48*24	10gens 51	40gens 63	60gens 71	100gens 93	
96*48*24	10gens 81	20gens 89	45gens 122		
96*96*24	5gens 116	10gens 116	40gens 205	45gens 205	70gens 234
96*96*48	5gens 235	10gens 235			

Fig. 8b Synapses per Neuron Doubles as 3D Space Doubles

One of the aims of the CAM-Brain research project is to build an artificial brain which can control 1000 behaviors of a "robot kitten" (i.e. a robot of size and capacities comparable to a kitten) or to control a household "cleaner robot". Presumably it will not be practical to evolve all these behaviors at once. Most likely they will have to be evolved incrementally, i.e., starting off with a very basic behavioral repertoire and then adding (stepwise) new behaviors. In brain circuitry terms, this means that the new neural modules will have to connect up to already established neural circuits. In practice, one can imagine placing neural bodies (somas) external to the established nervous system and then evolving new axonal and dendral connections to it.

The CAM-Brain Project hopes to create a new tool to enable serious investigation of the new field of "incremental evolution." This field is still rather virgin territory at the time of writing. This incremental evolution could benefit from using embryological ideas. For example, single seeder cells can be positioned in the 3D CA space under evolutionary control. Using handcrafted CA "developmental or embryological" rules, these seeder cells can grow into neurons ready to emit dendrites and axons (de Garis 1992). The CAM-Brain Project, if successful, should also have a major impact on both the field of neural networks and the electronics industry. The traditional preoccupation of most research papers on neural networks is on analysis, but the complexities of CAM-Brain neural circuits, will make such analysis impractical. However, using Evolutionary Engineering, one can at least build/evolve functional systems. The electronics industry will be given a new paradigm, i.e. evolving/growing circuits, rather than designing them. The long term impact of this idea should be significant, both conceptually and financially.

6. The 3D Version

The 3D version is a conceptually (but not practically) simple extension of the 2D version. Instead of 4 neighbors, there are 6 (i.e. North, East, West, South, Top, Bottom). Instead of 6 growth instructions as in the 2D version (i.e. extend, turn left, turn right, split extend left, split extend right, split left right), there are 15 in the 3D version. A 3D CA trail cross section consists of a center cell and 4 neighbor cells, each of different state or color (e.g. red, green, blue, brown). Instead of a turn left instruction being used as in the 2D case, a "turn green" instruction is used in the 3D case. The 15 3D growth instructions are (extend, turn red, turn green, turn blue, turn brown, split extend red, split extend green, split extend blue, split extend brown, split red brown, split red blue, split red green, split brown blue, split brown green, split blue green). A 3D CA rule thus consists of 8 integers of the form CTSENWB-->Cnew. The 3D version enables dendrites and axons to grow past each other, and hence reach greater distances. The weakness with the 2D version is that collisions in a plane are inevitable, which causes a crowding effect, whereby an axon or dendrite cannot escape from its local environment. This is not the case with the 3D version, which is topologically quite different. A 3D version is essential if one wants to build artificial brains with many interconnected neural modules. The interconnectivity requires long axons/dendrites. Fig. 12 shows an early result in 3D simulation. A space of 3D CA cells (48*48*48 cubes) was used. A single short 3D CA trail was allowed to grow to saturate the space. One can already sense the potential complexity of the neural circuits that CAM-Brain will be able to build. In 3D, it is likely that each neuron will have hundreds, maybe thousands of synapses, thus making the circuits highly evolvable due to their smooth fitness landscapes (i.e. if you cut one synapse, the effect is minimal when there are hundreds of them per neuron).

7. Recent Work

Just prior to writing this work, the author was able to test the idea that in 3D a single neuron could have an arbitrarily large number of synapses, provided that there is enough space for them to grow in. This was a crucial test, whose results are shown in Fig. 8b. Fitness was defined as the number of synapses formed for two neurons in CA spaces of 48*48*24, 96*48*24, 96*96*24, and 96*96*48 cells respectively. One can see that by doubling the space, one doubles (roughly) the number of synapses (for the elite chromosome). If this had not been the case, for example, if some kind of fractal effect had caused a crowding of the 3D circuits (similar to the crowding effect in 2D), then the whole CAM-Brain project would have been made doubtful. However, with this result, it looks as though evolvability in the 3D signaling phase will be excellent, although the author needs several months more work before completing the 3D signaling phase to confirm his confidence.

At the time of writing (December 1995), the author is completing the simulation of the 3D version, working on the many thousands of rules necessary to specify the creation of synapses. So far, more than 140,000 3D rules have been implemented, and it is quite probable that the figure may go higher than 150,000. Since each rule is rotated 24 ways (6 ways to place a cube on a surface, then 4 ways to rotate that cube) to cater to all possible orientations of a 3D trail, the actual number of rules placed in the (hashed) rule base will be over 3 million. Specifying these rules takes time, and constitutes so far, the bulk of the effort spent building the CAM-Brain system. Software has been written to help automate this rule generation process, but it remains a very time consuming business. Hence the immediate future work will be to complete the simulation of the 3D version. Probably, this will be done by early 1996.

Early in 1995, the author put his first application on the CAM8 machine (which rests on his desk). MIT's CAM8 is basically a hardware version of a look up table, where the output is a 16 bit word which becomes an address in the look up table at the next clock cycle. This one clock cycle lookup loop is the reason for CAM8's speed. It is possible to give each CA cell in the CAM8 more than 16 bits, but tricks are necessary. The first CAM8 experiment the author undertook involved only 16 bits per CA cell. This work is too short to go into details as to how the CAM8 functions, so only a broad overview will be given here. The 16 bits can be divided into slices, one slice per neighbor cell. These slices can then be "shifted" (by adding a displacement pointer) by arbitrarily large amounts (thus CAM8 CA cells are not restricted to having local neighbors). With only 16 bits, and 4 neighbors in the 2D case (Top, Right, Bottom, Left) and the Center cell, that's only 3 bits per cell (i.e. 8 states, i.e. 8 colors on the display screen). It is not possible to implement CAM-Brain with only 3 bits per CA cell. It was the intention of the author to use the CAM8 to show its potential to evolve neural circuits with a huge number of artificial neurons. The author chose an initial state in the form of a square CA trail with 4 extended edges. As the signals loop around the square, they duplicate at the corners. Thus the infinite looping of 3 kinds of growth signals supply an infinite number of growth signals to a growing CA network. There are 3 growth signals (extend, extend and split left, extend and split right). The structure needs exactly 8 states. The 8 state network grows into the 32 megacells of 16 bits each, which are available in the CAM8. At one pixel per cell, this 2D space takes over 4 square meters of paper poster (hanging on the author's wall). A single artificial neuron can be put into the space of one's little finger nail, thus allowing 25,000 neurons to fit into the space. If 16 Mbit memory chips are used instead of 4 Mbit chips, then the area and the number of neurons quadruples to 100,000.

Placing the poster on the author's wall suddenly gave visitors a sense of what is to come. They could see that soon a methodology will be ready which will allow the growth and evolution of artificial brains, because soon it will be possible to evolve many thousands of neural modules and their inter-connections. The visitors sense the excitement of CAM-Brain's potential.

Filling a space of 32 Mcells, with artificial neurons can be undertaken in at least two ways. One is to use a very large initialization table with position vectors and states. Another, is to allow the neurons to "grow" within the space. The author chose to use this "neuro-embryonic" approach. A single "seeder" CA cell is placed in the space. This seeder cell launches a cell to its right and beneath it. These two launched cells then move in their respective directions, cycling through a few dozen states When the cycle is complete, they deposit a cell which grows into the original artificial neuron shape that the author uses in the 2D version of CAM-Brain. Meanwhile other cells are launched to continue the growth. Thus the 32Mcell space can be filled with artificial neurons ready to accept growth cell "chromosomes" to grow the neural circuitry. This neuro-embryogenetic program (called "CAM-Bryo") was implemented on a workstation by the author, and ported to the CAM8 by his research colleague Felix Gers. In order to achieve the porting, use was made of "subcells" in the CAM8, a trick which allows more than 16 bits per CA cell, but for N subcells of 16 bits, the total CAM8 memory space available for CA states is reduced by a factor of N. Gers used two subcells for CAM-Bryo, hence 16M cells of 32 bits each. A second poster of roughly two square meters was made, which contained about 25,000 artificial neurons (see Fig. 13). Again, with 16Mbit memory chips, this figure would be 100,000. Gers expects to be able to port the 2D version of CAM-Brain to the CAM8 with a few weeks work, in which case, a third poster will be made which will depict about 15,000 neurons (with a lower density, to provide enough space for the neural circuitry to grow) and a mass of complex neural circuits. Once this is accomplished, we expect that the world will sit up and take notice - more on this in the next section.

The author's boss at ATR's Evolutionary Systems department, has recently set up a similar group at his company NTT, called Evolutionary Technologies (ET) department. The idea is that once the ATR Brain Builder group's research principles are fairly solid, the author and the author's boss (whose careers are now closely linked) will be able to tap into the great research and development resources of one of the world's biggest companies, when the time comes to build large scale artificial brains. NTT has literally thousands of researchers.

The author would like to see Japan invest in a major national research project within the next 10 years to build "Japan's Artificial Brain", the so-called "J-Brain Project". This is the goal of the author, and then to see such a project develop into a major industry within 20 years. Every household would like to have a cleaner robot controlled by an artificial brain. The potential market is huge.

8. Future Work

A lot of work remains to be done. The author has a list of "to dos" attached to his computer screen. The first item on the list is of course, to finish the rules for the 3D version of CAM-Brain. This should be done by early 1996, and will probably need over 150,000 CA rules. Second, the experience gained in porting the 700 rules for "CAM-Bryo" from a workstation to the CAM8 will shortly enable Gers to complete the much tougher task of porting the 2D version of CAM-Brain to the CAM8. In theory, since

there are 11,000 CA rules for the 2D version, and that each rule has 4 symmetry rotations, that makes about 45,000 rules in total to be ported. This fits into the 64K words addressable by 16 bits. The 3D version however, with its (estimated) 150,000 rules, and its 24 symmetry rotations, will require over 3 million rules in total. The 3D version may require a "Super CAM" to be designed and built (by NTT's "Evolutionary Technologies" Dept., with whom the author collaborates closely), which can handle a much larger number of bits than 16. The group at MIT who built CAM8 is thinking of building a CAM9 with 32 bits. This would be very interesting to the author. Whether NTT or MIT get there first, such a machine may be needed to put the 3D version into a CAM. However, with a state-of-the-art workstation (e.g. a DEC Alpha, which the user has on his desk) and a lot of memory (e.g. 256 Mbyte RAM), it will still be possible to perform some interesting evolutionary experiments in 3D CAM-Brain, but not with the speed of a CAM.

Another possibility for porting the 3D version to the CAM8, is to re implement it using CA rules which are more similar to those used in von Neumann's universal constructor/calculator, rather than Codd's. Von Neummann's 2D trails are only 1 cell wide, whereas Codd's 2D version are 3 cells wide, with the central message trail being surrounded by two sheath cells. The trick to using von Neumann's approach is incorporating the direction of motion of the cell as part of the state. The author's colleague Jacqueline Signorinni advises that CAM-Brain could be implemented at a higher density (i.e. more filled CA cells in the CA space) and without the use of a lookup table. The control of the new states would be implemented far more simply she feels, by simple IF-THEN-ELSE type programming. "von Neumann-izing" the 3D version of CAM-Brain might be a good task for the author's next grad student.

With the benefit of hindsight, if the 3D version is reimplemeted (and it is quite likely that my boss will have other members of our group do just that), then the author would advise the following. If possible (if you are implementing a Codd version) give the four sheath cells in a 3D CA trail cross section the same state. This would obviously simplify the combinatorial explosion of the number of collision cases during synapse formation. But, how then would the 3D growth instructions be interpreted when they hit the end of a trail, and how would you define the symmetry rotations? If possible, it would also be advisable to use the minimum number of gating cell states at growth junctions for all growth instructions. Whether this is possible or not, remains to be seen. However, if these simplifications can be implemented (and of course the author thought of them originally, but was unable to find solutions easily), then it is possible that the number of 3D CA rules might be small enough to be portable to the CAM8, which would allow 3D neural circuits to be evolved at 200 million CA cells per second (actually less because of the subcell phenomenon).

Once the 3D rules are ready, two immediate things need to be done. One is to ask ATR's graphics people to display these 3D neural circuits in an interesting, colorful way, perhaps with VR (virtual reality) 3D goggles with interactivity and zoom, so that viewers can explore regions of the dynamic circuits in all their 500 colors (states). This could be both fun and impressive. The second thing is of course to perform some experiments on the 3D version. As mentioned earlier, this will have to be done on a workstation, until a SuperCAM is built. Another possibility, as mentioned earlier is to redesign the 3D CA rules, to simplify them and reduce their number so that they can fit within the 64K 16 bit confines of the CAM8 machine.

As soon as the 2D rules have been fully ported to the CAM8, experiments can begin at speed. Admittedly the 2D version is topologically different from the 3D version (in the sense that collisions in 2D are easier than in 3D), it will be interesting to try to build up a rather large neural system with a large number of evolved modules (e.g. of the order of a hundred, to start with). At this stage, a host of new questions arise. Look at Fig. 14, which is van Essen's famous diagram of the modular architecture of the monkey's visual and motor cortex, showing how the various geographical regions of the brain (which correspond to the rectangles in the figure, and to distinct signal processing functions) connect with each other. Physiological techniques now exist which enable neuro-anatomists to know which distinct cortical regions connect to others. Thus the geography (or statics) of the biological brain is increasingly known. What remains mysterious of course, is the dynamics. How does the brain function.

Van Essen's diagram is inspirational to the author. The author would like to produce something similar with CAM-Brain, i.e. by evolving neural modules (corresponding to the rectangles, or parts of the rectangles) and their interconnections. This raises other questions about sequencing and control. For example, does one evolve one module and freeze its circuits and then evolve another module, freeze its circuits and finally evolve the connections between them, or does one evolve the two modules together, or what? Will it be necessary to place walls around each module, except for hand crafted I/O trails? The author has no clear answers or experience yet in these matters. The author's philosophy is "first build the tool, and then play with it. The answers will come with using the tool".

Another possibility for future work is to try to simplify the whole process of rule making. Perhaps higher level rules can be made which are far fewer in number and allow the author's low level rules to be generated from

them. If such a thing can be done, it would be nice, but the author believes there are still so many special cases in the specification of 3D CAM-Brain, that the number of high level rules may still be substantial. If these high level rules can be found, it might be possible to use them and put them on the CAM8, so that 3D evolutionary experiments can be undertaken at CAM8 speeds. Another idea is to use FPGAs (field programmable gate arrays) which code these high level rules and then to use them to grow 3D neural circuits. Each 3D CA cell could contain pointers to its 3D neighbors. In this way, it would be possible to map 3D neural circuits onto 2D FPGAs. This is longer term work. FPGAs are not cheap if many are needed. The author's RAM based solution has the advantage of being cheap, allowing a billion (one byte) CA cell states to be stored reasonably cheaply.

A recent suggestion coming from NTT concerns the use of an existing "content addressable memory" machine, which may be able to update CA cells effectively. There is a "CAMemory" research group at NTT that ATR is now collaborating with. If a small enough number of CAMemory Boolean function rules corresponding to CAM-Brain can be found (a big if), it is possible that a NTT's CAMemory could be thousands of times faster than the CAM8. Obviously, such a possibility is worth investigating, and if successful, could be extremely exciting, since it would mean *hundreds of billions* of CA cell updates a second.

The author feels that the nature of his research in 1996 will change from one of doing mostly software simulation (i.e. generating masses of CA rules), to learning about the biological brain (i.e. reading about brain science to get ideas to put into CAM-Brain), hardware design, and evolvable hardware. These activities will proceed in parallel. Of course, evolutionary experiments, on CAM8 for the 2D version of CAM-Brain, and on a 256 Mbyte RAM (DEC Alpha) workstation for the 3D version, will also be undertaken in parallel.

Further down the road, will be the attempt to design a "nanoCAM" or "CAM2001" based on nanoelectronics. The Brain Builder Group at ATR is collaborating with an NTT researcher who wants to build nano-scale cellular automata machines. With the experience of designing and building a "SuperCAM", a nanoscale CAM should be buildable with several orders of magnitude greater performance. Further research aims are to use CAs to make Hebbian synapses capable of learning. One can also imagine the generation of artificial "embryos" inside a CA machine, by having CA rules which allow an embryological "unfolding" of cell groups, with differentiation, transportation, death, etc. resulting in a form of neuro-morphogenesis similar to the way in which biological brains are built. The author's "CAM-Bryo" program is an early example of this kind of neuro-morphogenetic research.

9. Summary

The CAM-Brain Project at ATR, Kyoto, Japan, intends to build/grow/evolve a cellular automata based artificial brain of a billion artificial neurons at (nano-)electronic speeds inside Cellular Automata Machines (CAMs) by the year 2001. Quoting from a paper by Margolus and Toffoli of MIT's Information Mechanics group, "We estimate that, with integrated circuit technology, a machine consisting of a trillion cells and having an update cycle of 100 pico-second for the entire space will be technologically feasible within 10 years" (i.e. by 2000) (Margolus and Toffoli 1990). In a trillion 3D CA cells (cubes), one can place billions of artificial neurons. Such an artificial nervous system will be too complex to be humanly designable, but it may be possible to evolve it, and incrementally, by adding neural modules to an already functional artificial nervous system. In the summer of 1994, a 2D simulation of CAM-Brain using over 11000 hand crafted CA state transition rules was completed, and initial tests showed the new system to be evolvable. By early 1996, a 3D simulation will be completed.

If the CAM-Brain Project is successful, it will revolutionize the field of neural networks and artificial life, because it will provide a powerful new tool to evolve artificial brains with billions of neurons, and at electronic speeds. The CAM-Brain Project will thus produce the first Darwin Machine, i.e. a machine which evolves its own architecture. The author is confident that in time a new specialty will be established, based partly on the ideas behind CAM-Brain. This specialty is called simply "Brain Building".

The author and his colleague Felix Gers are about to port the 2D version of CAM-Brain to the CAM8. Hence in early 1996, it will be possible to evolve neural circuits with 25,000 neurons (or 100,000 neurons, with 16 Mbit memory chips) at 200 million CA cell updates a second. As mentioned earlier, the author expects that when this happens, the world will sit up and take notice. Twenty years from now, the author envisages the brain builder industry (i.e. intelligent robots etc.) as being one of the world's top industries, comparable with oil, automobile, and construction. He sees an analogy between the efforts of the very early rocket pioneers (e.g. the American Goddard, and the German (V2) von Braun) and the US NASA mission to the moon which followed. Today's 100,000-neuron artificial brain is just the beginning of what is to come. With adiabatic (heat generationless) reversible quantum computation, it will be possible to build 3D hardware circuits that do not melt. Hence size becomes no obstacle, which means that one could use planetoid size asteroids to build huge 3D brain like computers containing ten to power 40 components with one bit per atom. Hence late into the 21st century, the author predicts that human beings will be confronted with

the "artilect" (artificial intellect) with a brain vastly superior to the human brain with its pitiful trillion neurons. The issue of "species dominance" will dominate global politics late next century. The middle term prospects of brain building are exciting, but long term they are terrifying. The author has written an essay on this question (de Garis 1995). If you would like to be sent a copy, just email him at degaris@hip.atr.co.jp (The author will set up his home page on the web in 1996, after making the effort to learn html).

Finally, by way of a postscript - as the author was preparing the final draft, there were 6 people at ATR working on CAM-Brain (the author (3D CA rules), and his colleague Felix Gers (porting 2D to CAM-8), the author's Japanese colleague Hemmi and his programmer assistant Yoshikawa (translating CA rules to Boolean expressions), and two M. Sc. students from Nara Institute of Science and Technology (NAIST). At NTT, there were 3-4 people from the Content Addressable Memory machine group who were finding ways to apply their machine to CAM-Brain. So, things are certainly hotting up.

(Note added, April 1996) - Fig. 15 shows about 800 artificial neurons with their axons and dendrites grown using the CAM-8 machine with 128 Mega words of 16 bits. This figure is taken from an 8 square meter poster containing 100,000 neurons. In a year, this number will probably be a million. Felix Gers thinks he can port the 3D version to the CAM-8. The 3D rules are almost complete and number over 160,000, i.e. nearly 4 million with (24) rotations.

References

E.F. Codd, *Cellular Automata,* Academic Press, NY, 1968.

Hugo de Garis, "Genetic Programming: Modular Evolution for Darwin Machines,"*ICNN-90WASH-DC,* (Int. Joint Conf. on Neural Networks), January 1990, Washington DC, USA.

Hugo de Garis, "Genetic Programming", Ch.8 in book *Neural and Intelligent Systems Integration*, ed. Branko Soucek, Wiley, NY, 1991.

Hugo de Garis, "Artificial Embryology : The Genetic Programming of an Artificial Embryo", Ch.14 in book *Dynamic, Genetic, and Chaotic Programming*, ed. Branko Soucek and the IRIS Group, Wiley, NY, 1992.

Hugo de Garis, "Evolvable Hardware : Genetic Programming of a Darwin Machine", in *Artificial Neural Nets and Genetic Algorithms,* R.F. Albrecht, C.R. Reeves, N.C. Steele (eds.), Springer Verlag, NY, 1993.

Hugo de Garis, "Genetic Programming : Evolutionary Approaches to Multistrategy Learning", Ch.21 in book "Machine Learning : A Multistrategy Approach, Vol.4", R.S. Michalski & G. Tecuci (eds), Morgan Kauffman, 1994.

Hugo de Garis, "Cosmism : Nano Electronics and 21st Century Global Ideological Warfare", (to appear in a future nanotech book).

K.E. Drexler, *Nanosystems : Molecular Machinery, Manufacturing and Computation,* Wiley, NY, 1992.

D..E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison-Wesley, Reading, MA, 1989.

T. Toffoli & N. Margolus, *Cellular Automata Machines*, MIT Press, Cambridge, MA, 1987; and *Cellular Automata Machines*, in Lattice Gas Methods for Partial Differential Equations, SFISISOC, eds. Doolen et al, Addison-Wesley, 1990.

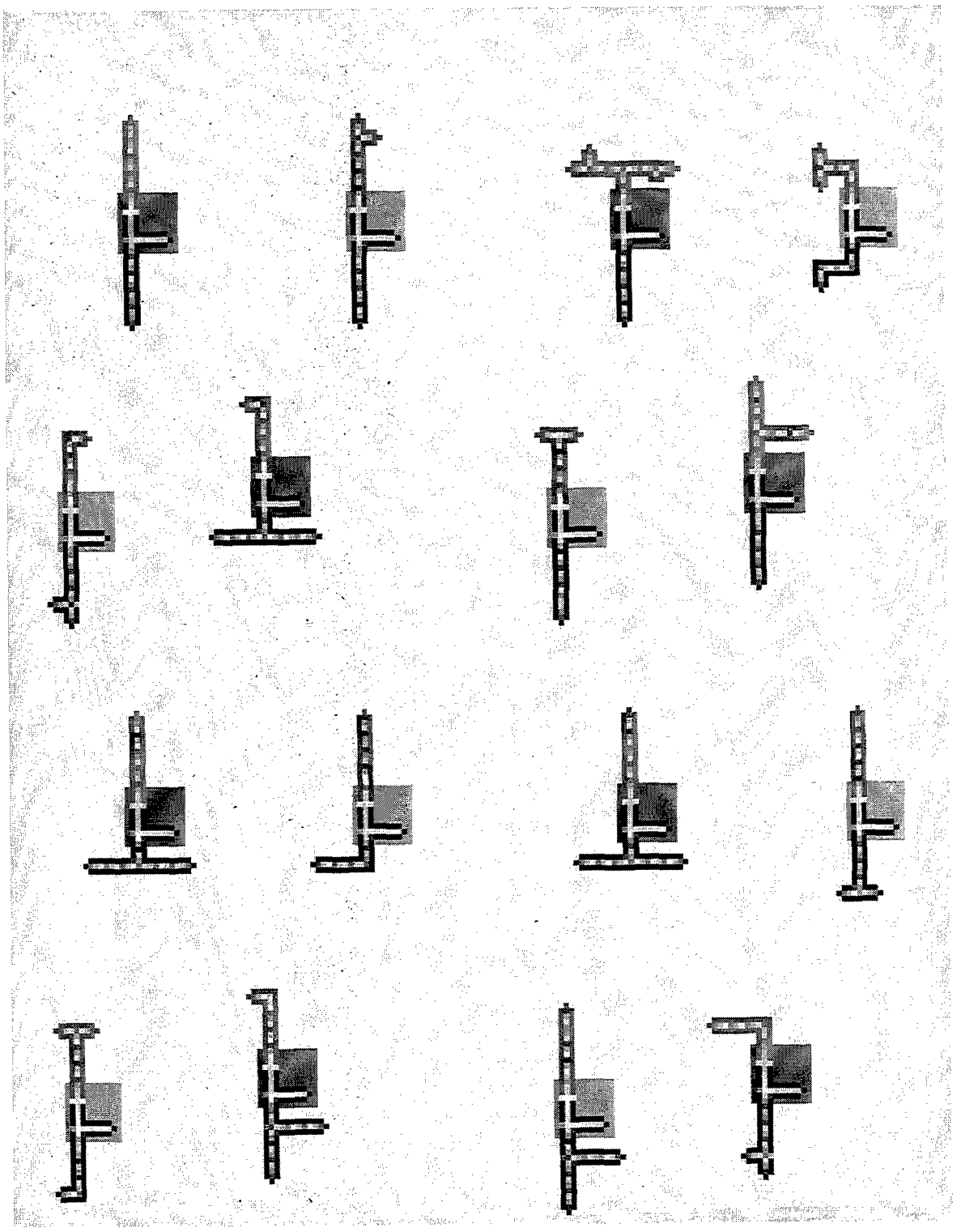


Fig. 9 2D CAM-Brain Early Growth

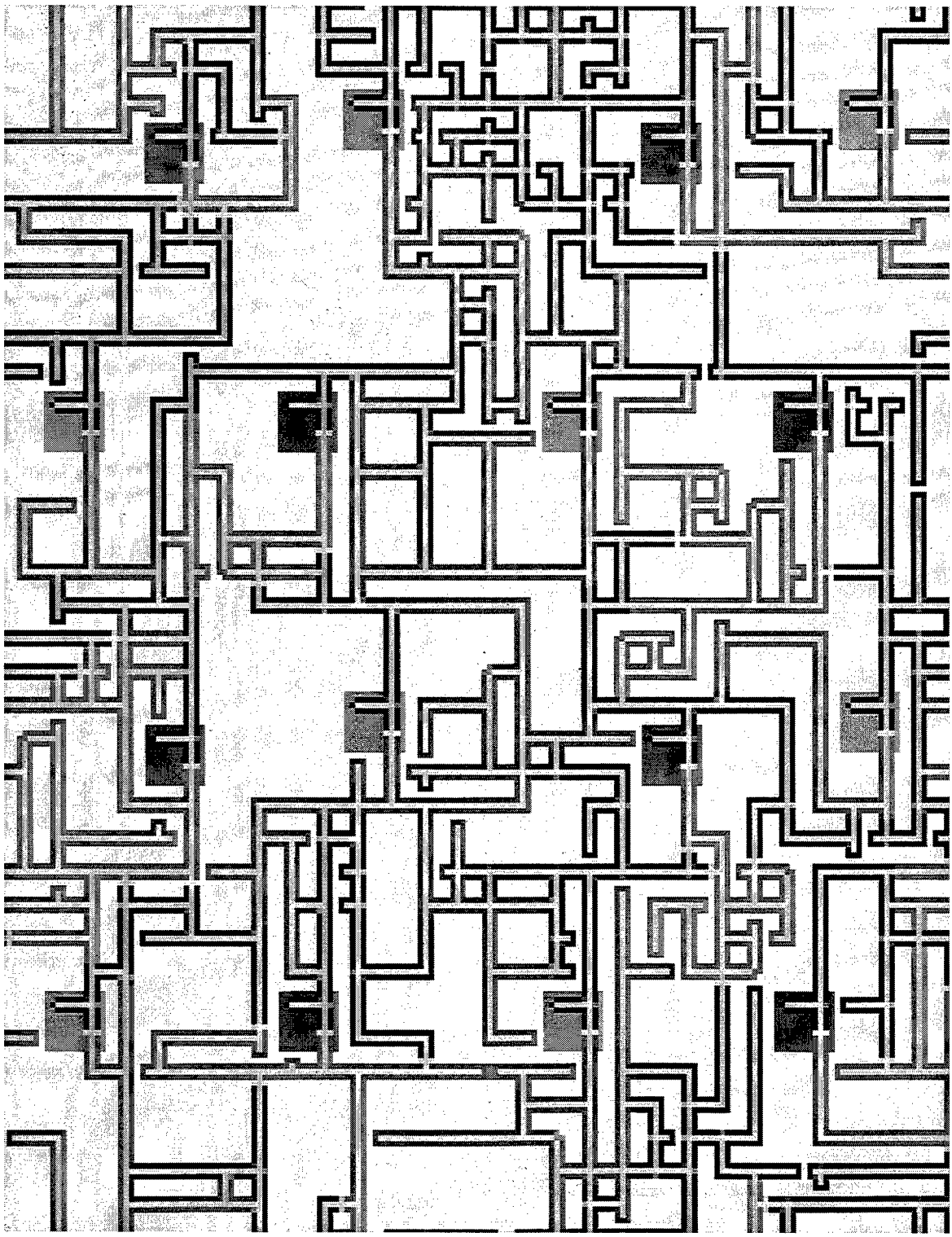


Fig. 10 2D CAM-Brain Completed Growth

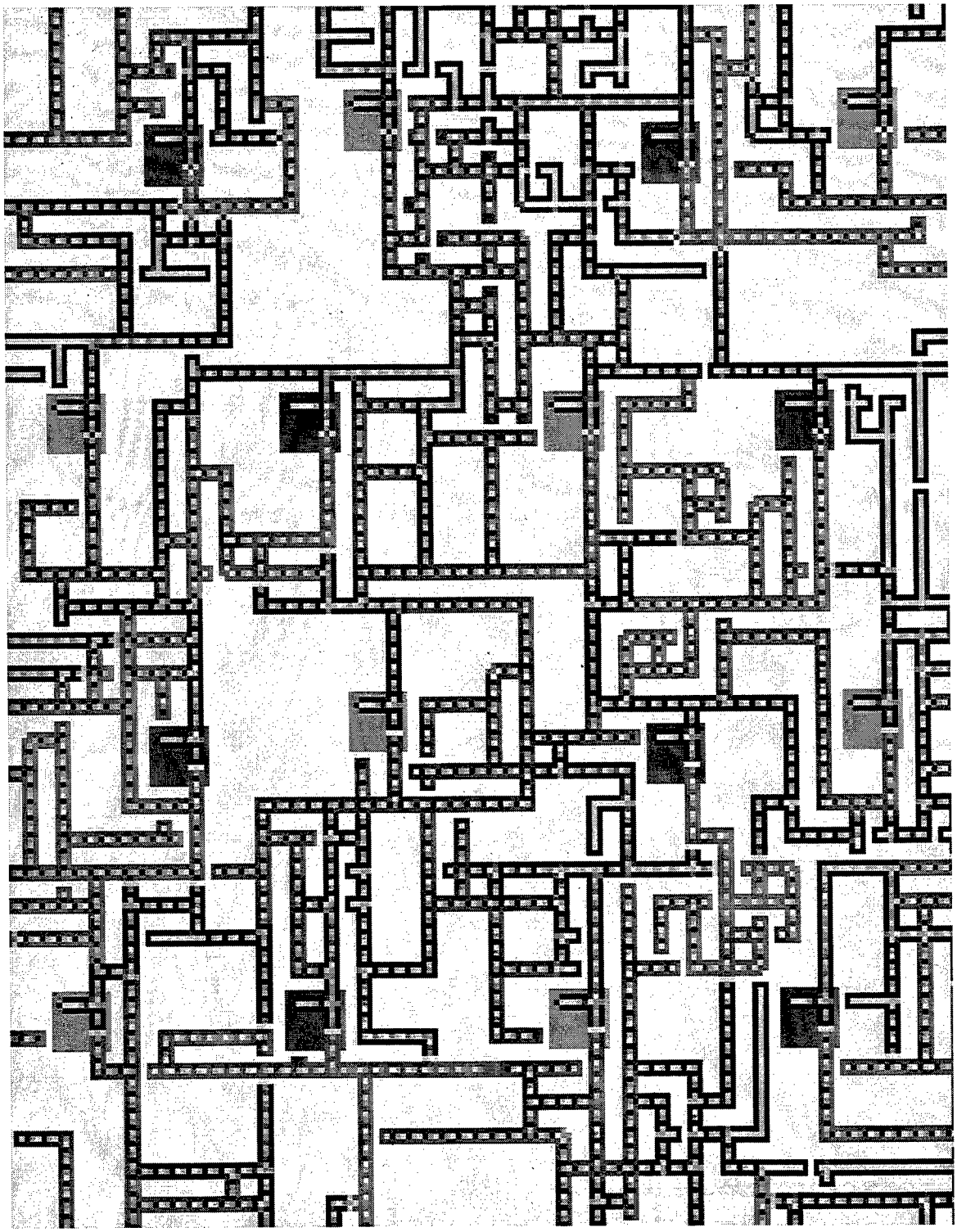


Fig. 11 2D CAM-Brain Neural Signaling

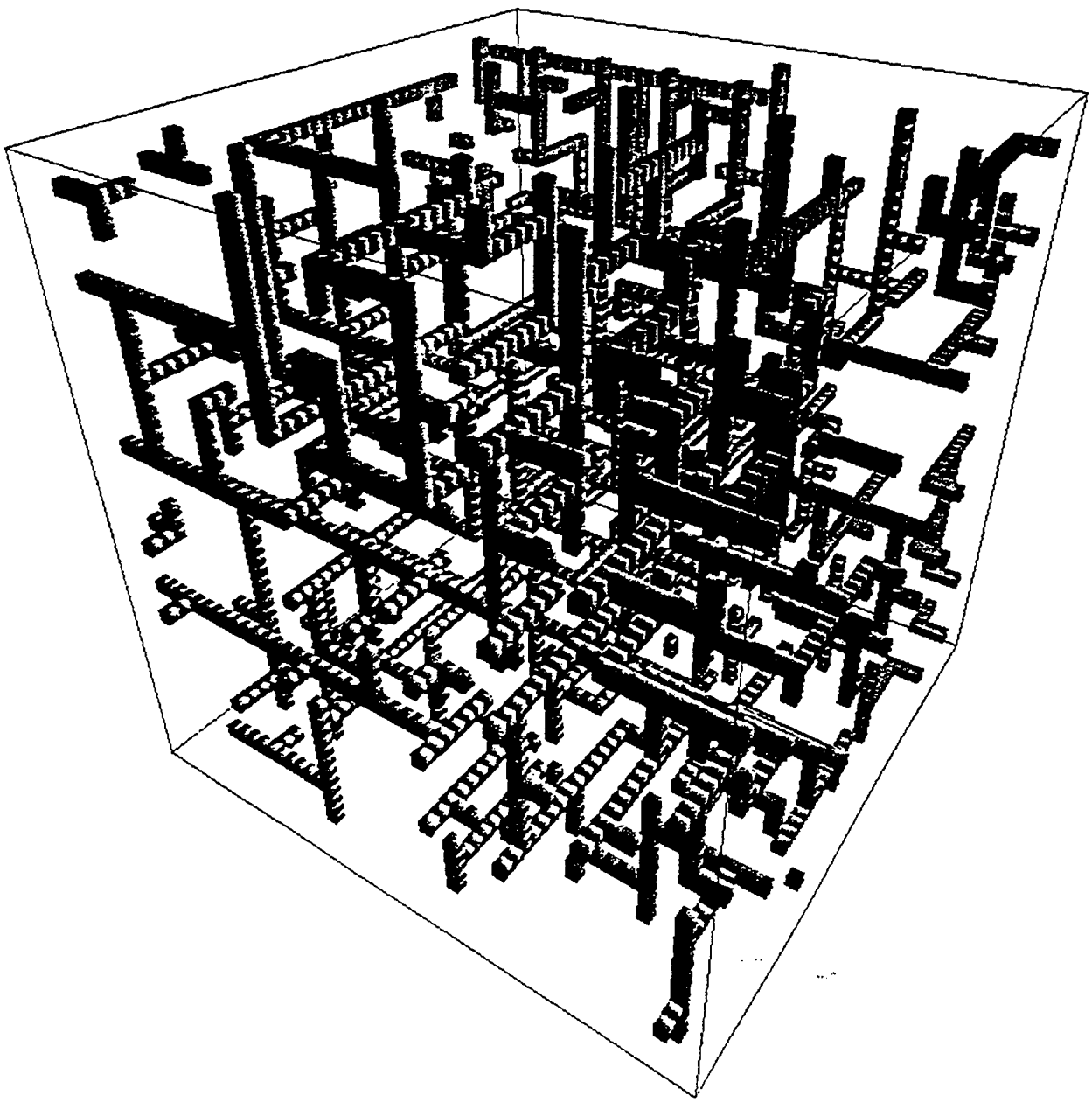


Fig. 12 3D CAM-Brain Non-Synaptic Growth

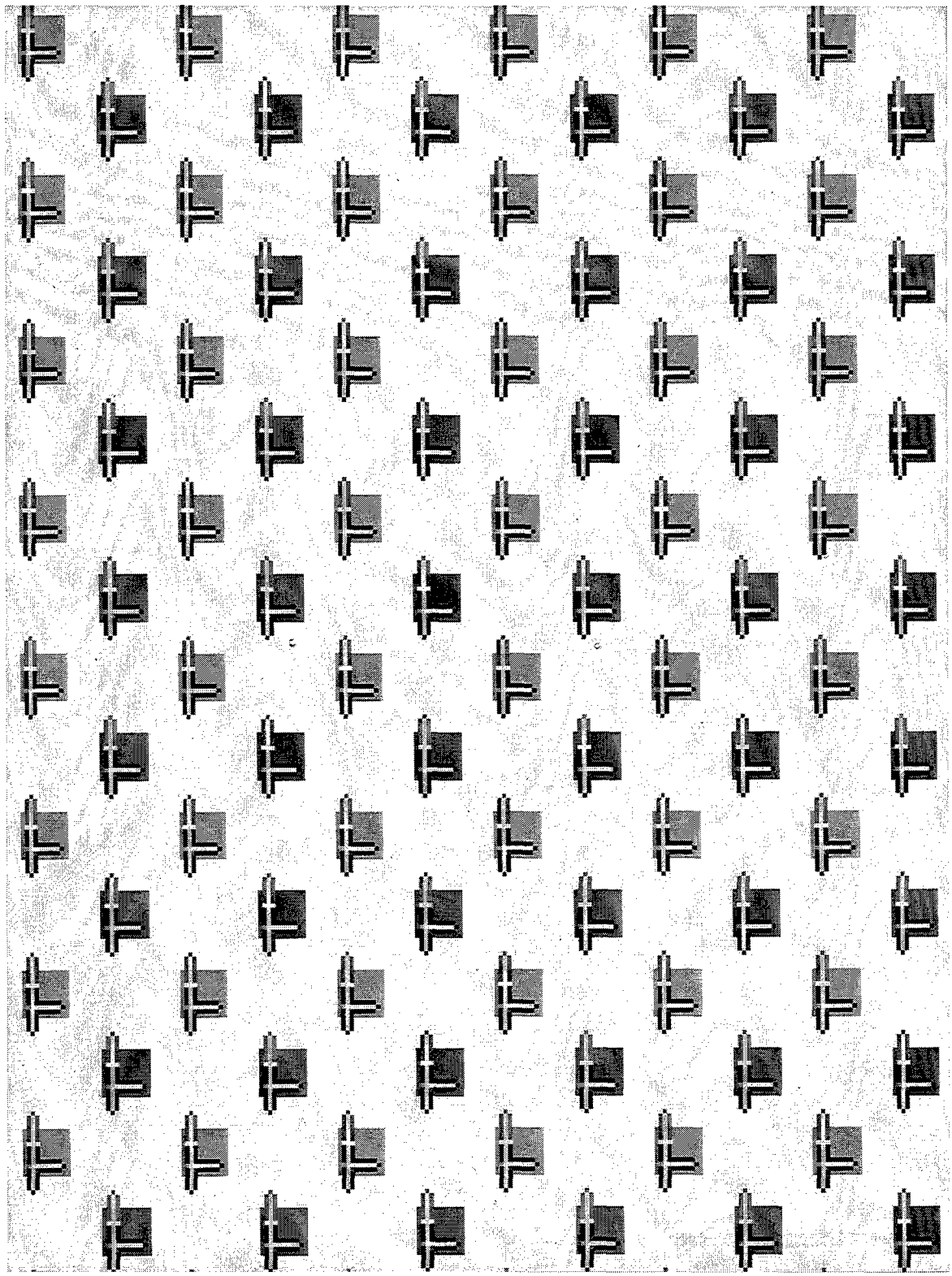


Fig. 13 2D CAM-Bryo

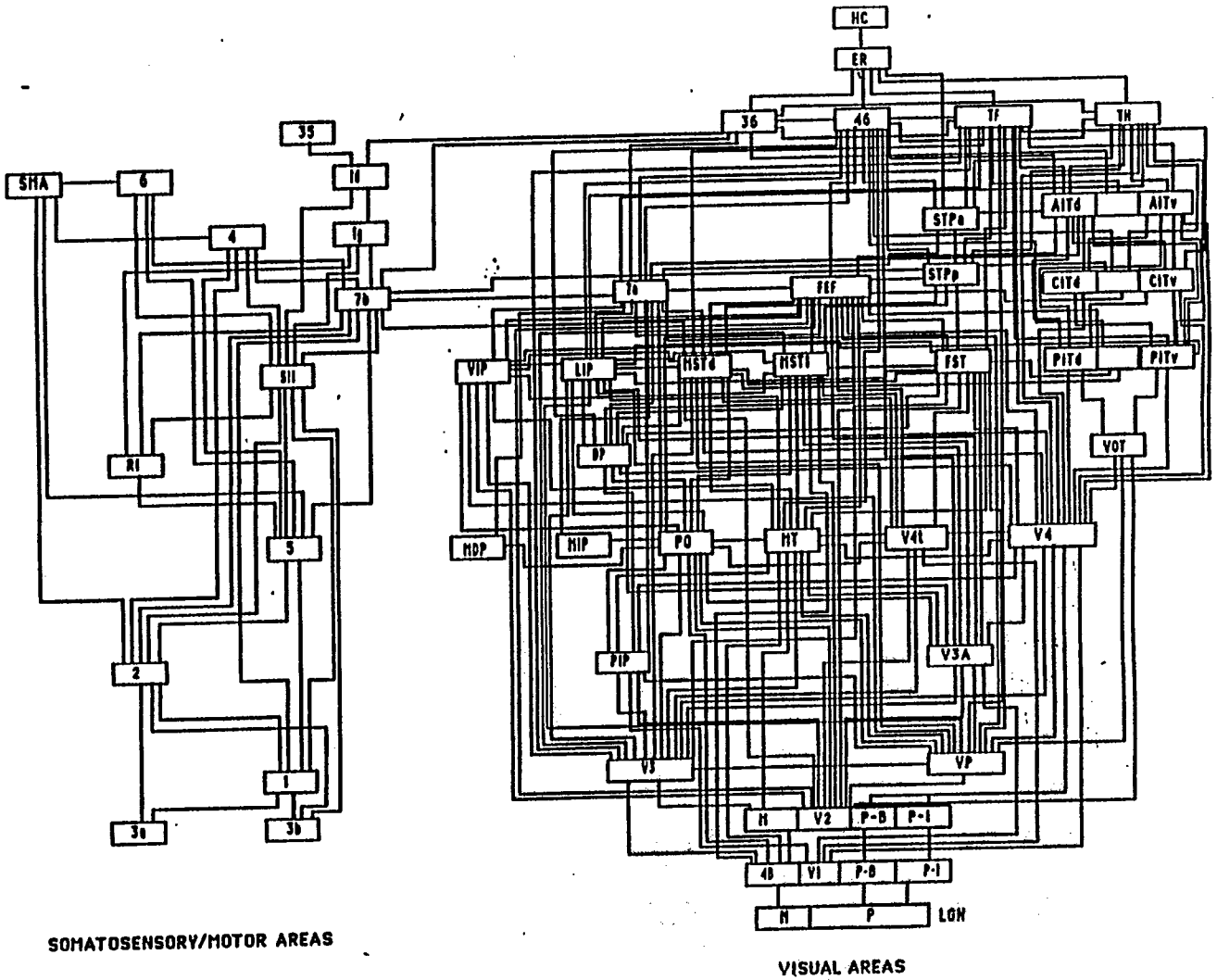


Fig. 14 van Essen's Monkey Brain Architecture

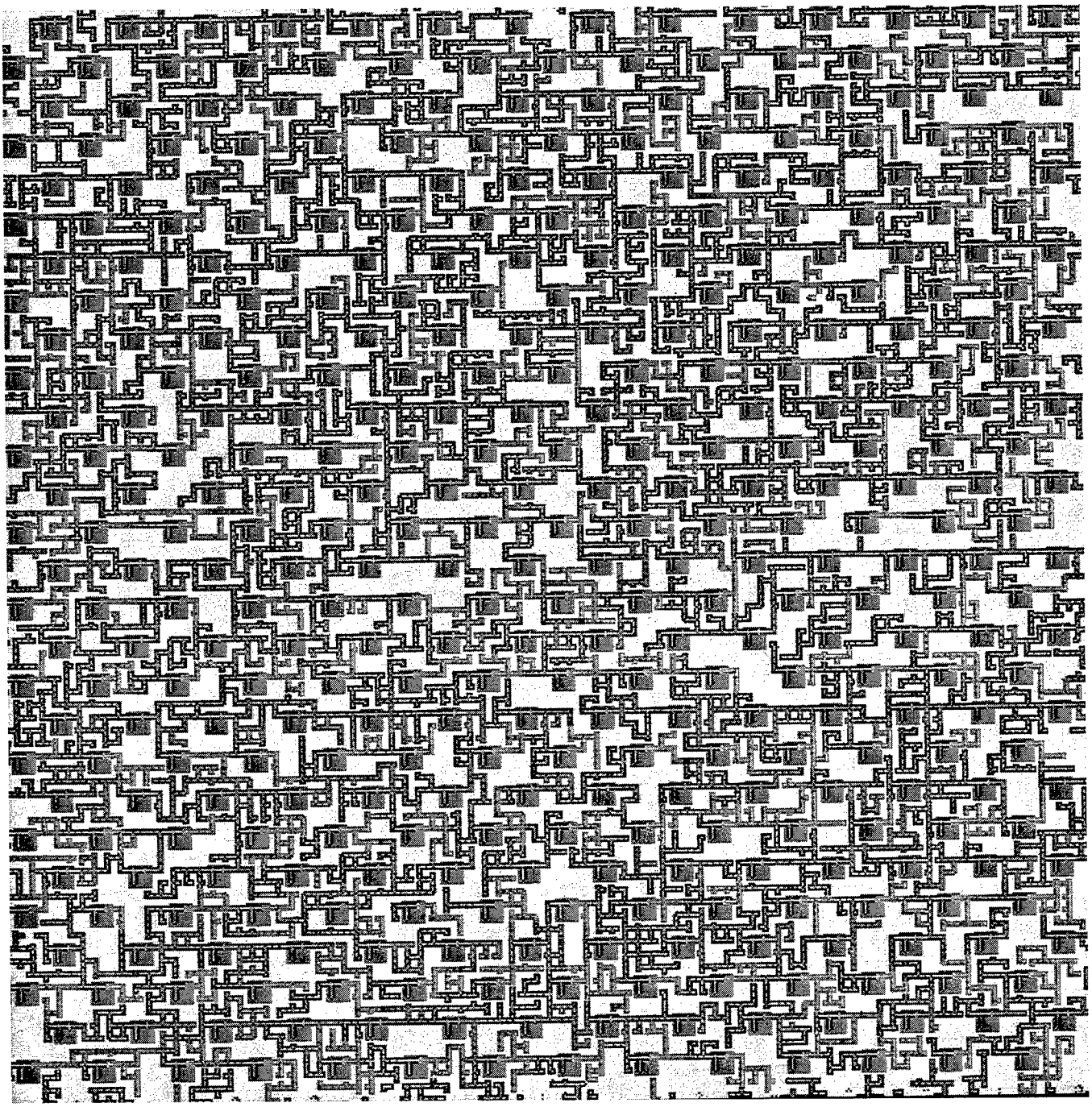


Fig. 15 2D CAM-Brain on MIT's "CAM-8"

Integrating EBL and ILP to Acquire Control Rules for Planning*

Tara A. Estlin and Raymond J. Mooney
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712
{estlin,mooney}@cs.utexas.edu

Abstract

Most approaches to learning control information in planning systems use *explanation-based learning* to generate control rules. Unfortunately, EBL alone often produces overly complex rules that actually decrease planning efficiency. This paper presents a novel learning approach for control knowledge acquisition that integrates explanation-based learning with techniques from *inductive logic programming*. EBL is used to constrain an inductive search for selection heuristics that help a planner choose between competing plan refinements. SCOPE is one of the few systems to address learning control information in the newer partial-order planners. Specifically, SCOPE learns domain-specific control rules for a version of the UCPOP planning algorithm. The resulting system is shown to produce significant speedup in two different planning domains.

Introduction

Efficient planning often requires domain-specific search heuristics; however, constructing appropriate heuristics for a new domain is a difficult, laborious task. Research in learning and planning attempts to address this important problem by developing methods that automatically acquire search-control knowledge from experience. Past systems have often employed *explanation-based learning* (EBL) to learn search-control knowledge. Unfortunately, standard EBL can frequently produce complex, overly-specific control rules that decrease rather than improve overall planning performance (Minton 1988). By incorporating induction to learn simpler, approximate control rules, we can greatly improve the utility of acquired knowledge (Cohen 1990; Leckie & Zuckerman 1993). In this paper, we describe SCOPE, a system that uses a unique combination of machine learning techniques to acquire effective search-control rules for a partial-order planner.

This research was supported by the NASA Graduate Student Researchers Program, grant number NGT-51332.

Specifically, SCOPE (Search Control Optimization of Planning through Experience) integrates *explanation-based generalization* (EBG) (Mitchell, Keller, & Kedar-Cabelli 1986; DeJong & Mooney 1986) with techniques from *inductive logic programming* (ILP) (Quinlan 1990; Muggleton 1992; Lavrač & Džeroski 1994) to learn high-utility rules that can generalize well to new planning situations. ILP techniques have often been used in the past for inducing logic programs from a set of examples. SCOPE illustrates that these techniques can also be successfully applied for improving program efficiency.

SCOPE extends previous planning and learning research by acquiring control knowledge for the newer, more efficient *partial-order* planners. Most work in this area has been in the context of linear, state-based planners (Minton 1989; Gratch & DeJong 1992; Leckie & Zuckerman 1993). These planners are usually classified as *total-order* planners, since plans steps are maintained in a strictly ordered list. Recent experimental results, however, support that partial-order planners are more efficient than total-order planners in most domains (Barrett & Weld 1994; Minton *et al.* 1992; Kambhampati & Chen 1993). In a partially-ordered plan, some steps can remain unordered with respect to each other, thereby allowing a planner to avoid premature commitments to an incorrect ordering. Though partial-order planners are considered a more proficient planning strategy, they are not a panacea for efficient planning. Added control knowledge can still dramatically effect the their performance. However, there has been little work on learning control for current partial-order planning systems (Katukam & Kambhampati 1994).

SCOPE learns control rules for a partial-order planner in the form of selection heuristics. These heuristics greatly reduce backtracking by directing a planner to immediately select appropriate plan refinements. SCOPE is implemented in Prolog, which provides a good framework for learning control knowledge. A ver-

sion of the UCPOP planning algorithm (Penberthy & Weld 1992) was implemented as a Prolog program to provide a testbed for SCOPE. Experimental results are presented on two domains that demonstrate SCOPE can significantly increase partial-order planning efficiency.

The remainder of this paper is organized as follows. In the next section, we describe the UCPOP planner and explain how control rules are implemented. Next, SCOPE's learning algorithm is described, after which experimental results are presented. Finally, we discuss related work and present ideas for future research.

Learning Control For Partial-Order Planning

Search-control information can improve the performance of a planner by guiding it to solutions quickly with minimal search. Past learning systems have usually employed either EBL (Minton 1989; Bhatnagar & Mostow 1994; Kambhampati, Katukam, & Qu 1996) or induction (Leckie & Zuckerman 1993; Langley & Allen 1991) to acquire control rules based on past planning behavior. Our approach differs from these methods by using a novel combination of analytic and inductive methods to acquire control information. In our framework, control-rule learning is viewed as a form of concept learning. SCOPE uses an inductive algorithm to learn definitions of when plan refinements should be applied. EBL focuses the inductive search so good rules are learned quickly without requiring an overly large search space.

The UCPOP Planner

The base planner we chose for experimentation is UCPOP, a partial-order planner described in (Penberthy & Weld 1992). In UCPOP, a partial plan is described as a four-tuple: $\langle \mathcal{S},\mathcal{B},\mathcal{O},\mathcal{L} \rangle$ where $\mathcal{S}$ is a set of actions, $\mathcal{O}$ is a set of ordering constraints, $\mathcal{L}$ is a set of causal links, and $\mathcal{B}$ is a set of codesignation constraints over variables appearing in $\mathcal{S}$. Actions are described by a STRIPS schema containing precondition, add and delete lists. The set of ordering constraints, $\mathcal{O}$, specifies a partial ordering of the actions contained in $\mathcal{S}$. Causal links record dependencies between the effects of one action and the preconditions of another. These links are used to detect *threats*, which occur when a new action interferes with a past decision.

UCPOP begins with a null plan and an agenda containing the top-level goals. The initial and goal states are represented by adding two extra actions to S, A_0 and A_∞. The effects of A_0 correspond to the initial state, and the preconditions of A_∞ correspond to the desired goal state. In each planning cycle, a goal is removed from the agenda and an existing or new action is chosen to assert the goal. After an action is selected, the necessary ordering, casual link and codesignation constraints are added to $\mathcal{O}$, $\mathcal{L}$, and $\mathcal{B}$. If a *new* action was selected, the action's preconditions are added to the agenda. UCPOP then checks for threats and resolves any found by adding an additional ordering constraint. UCPOP is called recursively until the agenda is empty. On termination, UCPOP uses the constraints found in $\mathcal{O}$ to determine a total ordering of the actions in $\mathcal{S}$, and returns this as the final solution.

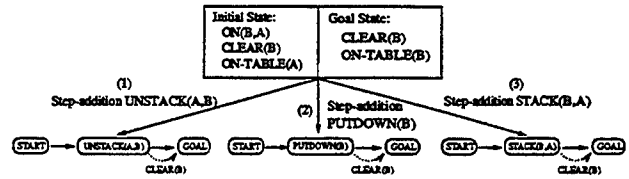


Figure 1: Three competing refinement candidates for achieving the goal *Clear(B)*.

Control Rule Format

SCOPE learns search-control rules for planning decisions that might lead to a failing search path (i.e. might be backtracked upon). Figure 1 illustrates an example from the blocksworld domain where control knowledge could be useful. Here, there are three possible refinement candidates for adding a new action to achieve the goal *Clear(B)*. For each set of refinement candidates, SCOPE learns control rules in the form of selection rules that define when each refinement should be applied. A single selection rule consists of a conjunction of conditions that must all evaluate to true for the refinement to be used. If at least one condition fails, that refinement candidate will be rejected, and the next candidate evaluated. For example, shown next is a selection rule for the first candidate (from Figure 1) which contains several control conditions.

```
Select operator Unstack(?X,?Y) to establish
  goal(Clear(?Y),s1)[1]
    If exists-operator(s2) ∧
       establishes(s2,On(?X,?Y)) ∧
       possibly-before(s2,s1).
```

This rule states that the operator `Unstack(?X,?Y)` should be selected to add *Clear(?Y)* only when there is an existing action s_2 that adds *On(?X,?Y)* and s_2 can be ordered before the action s_1, which requires *Clear(?Y)*.

Learned control information is incorporated into the

[1]Goals are represented in our planner by the *(Goal,Action)* structure where *Action* is the plan step that requires *Goal*.

planner so that attempts to select an inappropriate refinement will immediately fail. Control rules can consist of a conjunction of conditions (as shown above) or a disjunction of several conjunctions. SCOPE can also make selection rules deterministic or nondeterministic depending on the accuracy of the learned rule.

The Prolog programming language provides an excellent framework for learning control rules. Search algorithms can be implemented in Prolog in such a way that allows control information to be easily incorporated in the form of clause-selection rules (Cohen 1990). These rules help avoid inappropriate clause applications, thereby reducing backtracking. A version of the UCPOP partial-order planning algorithm has been implemented as a Prolog program.[2] Planning decision points are represented in this program as clause-selection problems (i.e. each refinement candidate is formulated as a separate clause). SCOPE is then used to learn refinement-selection rules which are incorporated into the original planning program in the form of clause-selection heuristics.

The SCOPE Learning System

SCOPE is based on the DOLPHIN speedup learning system (Zelle & Mooney 1993), which optimizes logic programs by learning clause-selection rules. DOLPHIN has been shown successful at improving program performance in several different domains, including planning domains which employed a simple state-based planner. DOLPHIN, however, has little success improving the performance of a partial-order planner due to the higher complexity of the planning search space. In particular, DOLPHIN's simple control rule format lacks the expressibility necessary to describe complicated planning situations. DOLPHIN also has difficulty successfully generalizing explanations produced by a partial-order planner. SCOPE has greatly expanded upon DOLPHIN's algorithm to be effective on more complex planning systems.

The input to SCOPE is a planning program and a set of training examples. SCOPE uses the examples to induce a set of control heuristics which are then incorporated into the original planner. Figure 2 shows the three main phases of SCOPE's algorithm, which are explained in the next few sections. A more detailed description can be found in (Estlin 1996).

Example Analysis

In the example analysis phase, two main outputs are produced: a set of selection-decision examples and a set of *generalized* proof trees. Selection-decision examples are used to record successful and unsuccessful applications of a plan refinement. Generalized proof trees provide a background context that explains the success of all correct planning decisions. These two pieces of information are used in the next phase to build control rules.

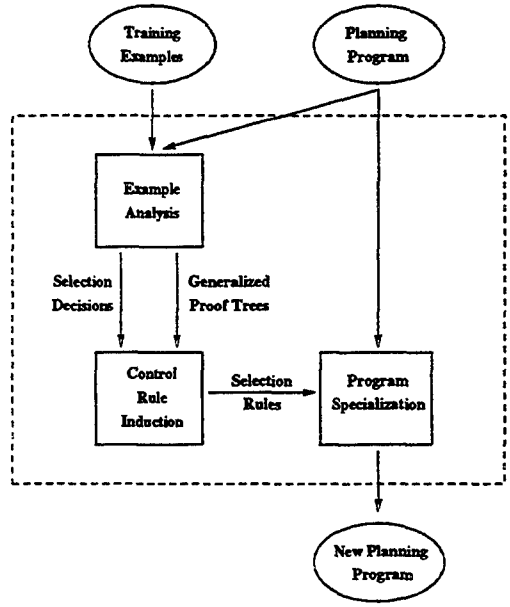


Figure 2: SCOPE's High-Level Architecture

Selection-decision examples are produced using the following procedure. First, training examples are solved using the existing planner. A trace of the planning decision process used to solve each example is stored in a proof tree, where the root represents the top-level planning goal and nodes correspond to different planning procedure calls. These proofs are then used to extract examples of correct and incorrect refinement-selection decisions. Specifically, a "selection decision" is a planning subgoal that was solved correctly by applying some plan refinement, such as adding a new action. As an example, consider the planning problem that was introduced in Figure 1. The planning subgoal represented by this figure is shown below:[3]

For $\mathcal{S}$ = (0:Start,G:Goal),
$\mathcal{O}$ = (0 $\prec$ G),
$\mathcal{L}$ = $\emptyset$,
agenda = (*Clear(B)*,G),(*On-Table(B)*,G),
Select operator ?OP to establish goal(*Clear(B)*,G)

Selection decisions such as this one are collected for all

[2] Our Prolog planner performs comparably to the standard LISP implementation of UCPOP on the sample problem sets used to test the learning algorithm (discussed in the Experimental Evaluation section).

[3] Binding constraints in our system are maintained through Prolog, therefore, the set of binding constraints, $\mathcal{B}$, is not explicitly represented in planning subgoals.

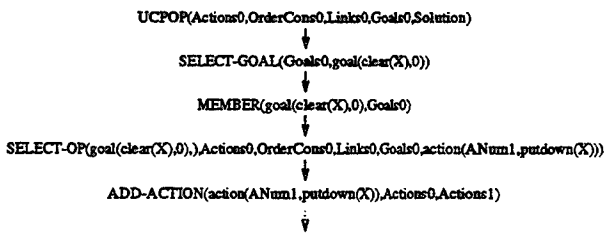


Figure 3: Top Portion of a Generalized Proof Tree

competing plan refinements. Refinements are considered "competing" if they can be applied in identical planning decisions, such as the three refinement candidates shown in Figure 1. A correct decision for a particular refinement candidate is an application of that refinement found on a solution path. An incorrect decision is a refinement application that was tried and subsequently backtracked over. The subgoal shown above would be identified as a positive selection decision for candidate 2 (adding `Putdown(A)`), and would also be classified as a negative selection decision for candidates 1 and 3 (adding `Unstack(A,B)` or `Stack(B,A)`). Any given training problem may produce numerous positive and negative examples of refinement selection decisions. Selection decision examples are used later in induction to represent positive and negative examples of when to apply particular planning refinements.

The second output of the example analysis phase is a set of generalized proof trees. Standard EBG techniques (Mitchell, Keller, & Kedar-Cabelli 1986; DeJong & Mooney 1986) are used to generalize each training example proof tree. The goal of this generalization is to remove proof elements that are dependent on the specific example facts while maintaining the overall proof structure. Generalized proof information is used later to explain new planning situations. The top portion of a generalized proof tree is shown in Figure 3. This proof was extracted from the solution trace of the problem introduced in Figure 1. The top-level goal in this proof is a call to the planner that included the initial list of plan actions, operators, causal-links, and agenda as input arguments. The last argument corresponds to the output plan solution. All subsequent planning procedure calls are included in the proof structure. The generalized proof of an example provides a context which "explains" the success of correct decisions.

Control Rule Induction

The goal of the induction phase is to produce an operational definition of when it is useful to apply a refinement candidate. Given a candidate, C, we desire a definition of the concept "subgoals for which C is useful". In the blocksworld domain, such a definition is learned for each of the candidates shown in Figure 1. In this context, control rule learning can be viewed as relational concept learning. A number of systems (Quinlan 1990; Muggleton 1992; Zelle & Mooney 1994) have been designed to tackle this type of learning problem. SCOPE employs a version of Quinlan's FOIL algorithm to learn control rules through induction.

The choice of a FOIL-like framework is motivated by a number of factors. First, the basic FOIL algorithm is relatively easy to implement and has proven efficient in a number of domains. Second, FOIL has a "most general" bias which tends to produce simple definitions. Such a bias is important for learning rules with a low match cost, which helps avoid the utility problem. Third, it is relatively easy to bias FOIL with prior knowledge (Pazzani & Kibler 1992). In our case, we can utilize the information contained in the generalized proof trees of planning solution traces.

FOIL Algorithm FOIL attempts to learn a concept definition in terms of a given set of background predicates. This definition is composed of a set of Horn clauses that cover all of the positive examples of a concept, and none of the negative examples. The selection-decision examples collected in the example analysis phase provide the sets of positive and negative examples for each refinement candidate.

```
Initialization
    Definition := null
    Remaining := all positive examples
While Remaining is not empty
        Find a clause, C, that covers some examples in
            Remaining, but no negative examples.
        Remove examples covered by C from Remaining.
        Add C to Definition.
```

Figure 4: Basic FOIL Covering Algorithm

FOIL may be viewed as a simple covering algorithm which has the basic form shown in Figure 4. The "find a clause" step is implemented by a general-to-specific hill-climbing search. FOIL adds antecedents to the developing clause one at a time. At each step FOIL evaluates all possible literals that might be added and selects the one which maximizes an information-based gain heuristic.

The generation of candidate literals to add to a developing clause normally consists of trying each background predicate with all possible combinations of variables currently in the clause and any new predicate variables. SCOPE uses an intensional version of FOIL where background predicates can be defined as Prolog predicates instead of requiring an extensional represen-

tation (Mooney & Califf 1995). Any predicates that can be used as rule antecedents must be introduced as background knowledge.

One major drawback to FOIL (and other similar inductive algorithms) is that the hill-climbing search for a good antecedent can easily explode, especially when there are numerous background predicates with large numbers of arguments. When selecting each new clause antecedent, FOIL tries *all* possible variable combinations for *all* predicates before making its choice. This search grows *exponentially* as the number of predicate arguments increases. SCOPE circumvents this search problem by using the generalized proofs of training examples. By examining the proof trees, SCOPE identifies a small set of potential literals that could be added as antecedents to the current clause definition. Literals are added in a way that utilizes variable connections already established in the proof tree. This approach nicely focuses the FOIL search by only considering literals (and variable combinations) that were found useful in solving the training examples.

Building Control Rules from Proof Trees The generalized proofs of training examples can be seen as giving the context for the appropriate applications of refinements within a proof. Some nodes of a generalized proof tree contain calls to "operational" predicates. These are usually low-level predicates that have been classified as "easy to evaluate" within the problem domain, and thus can be used to build efficient concept definitions. The operational nodes of a proof represent all of the primitive conditions that had to be satisfied for the proof to succeed. SCOPE employs induction in an attempt to identify a small set of these simple tests that will provide necessary guidance in determining whether the application of a refinement is likely to lead to a solution. Since test conditions that verify a planning decision are sometimes not executed until much later, it is important to consider an entire example proof instead of just the surrounding context of a particular decision. For instance, the planner might not verify that choosing the action `Putdown(a)` to establish the goal *Clear(a)* is correct until much later in the planning process when it checks to see if some other action has asserted *Holding(a)*.

SCOPE employs the same general covering algorithm as FOIL but modifies the clause construction step. Clauses are successively specialized by considering how their target refinements were used in solving training examples. Suppose we are learning a definition for when each of the refinement candidates in Figure 1 should be applied. The actual program predicate representing this type of refinement is select-op. This predicate is defined with several arguments including the unachieved goal and an output argument for the selected operator. (Plan state information is also automatically included as arguments to any refinement predicate.) The full predicate head of this refinement is shown below.

```
select-op(Goal,Steps,OrderCons,Links,Agenda,ReturnOp)
```

For each refinement candidate, SCOPE begins with the most general definition possible. For instance, the most general definition covering candidate 1's selection examples is the following; call this clause C.

```
select-op(Goal,Steps,OrderCons,Links,Agenda,unstack(A,B)) :-
    TRUE
```

This overly general definition covers *all* positive examples and *all* negative examples of when to apply candidate 1, since it will always evaluate to true. C can be specialized by adding antecedents to its body. This is done by unifying C's head with a (generalized) proof subgoal that was solved by applying candidate 1 and then adding an operational literal from the same proof tree which shares some variables with the subgoal. For example, one possible specialization of the above clause is shown below.

```
select-op((clear(B),S1),Steps0,OrderCons,Links,Agenda,unstack(A,B)) :-
    establishes(on(B,A),Steps1,S2).
```

Here, a proof tree literal has been added which checks if there an existing plan step that establishes the goal *On(B,A)*. Variables in a newly added antecedent can be connected with the existing rule head in several ways. First, by unifying a rule head with a generalized subgoal, variables in the rule head become unified with variables existing in a proof tree. All operational literals in that proof that share variables with the generalized subgoal are tested as possible antecedents. This method initially establishes many relevant variable connections between a rule head and its antecedents.

A second way variable connections can be established is through the standard FOIL technique of unifying variables of the same type. When SCOPE tests a literal for use in a control rule, the literal may contain input parameters that are not bound by the rule head or other existing literals in the rule. If such parameters exist, SCOPE attempts to unify these parameters with terms of the same type that are already present in the rule. For example, the rule shown above has an antecedent with an unbound input, `Steps1`, which does not match any other variables in the clause. SCOPE will modify the rule, as shown below, so that the `Steps1` is unified with a term of the same type from the rule head, `Steps0`.

```
select-op((clear(B),S1),Steps0,OrderCons,Links,Agenda,unstack(A,B)) :-
    establishes(on(B,A),Steps0,S2).
```

SCOPE considers all such specializations of a rule and selects the one which maximizes FOIL's information-gain heuristic.

SCOPE also considers several other types of control rule antecedents during induction. Besides pulling literals directly from generalized proof trees, SCOPE can use negated proof literals, determinate literals (Muggleton 1992), variable codesignation constraints, and relational clichés (Silverstein & Pazzani 1991). Incorporating different antecedent types helps SCOPE learn expressive control rules that can describe partial-order planning situations.

Program Specialization Phase

Once refinement selection rules have been learned, they are passed to the program specialization phase which adds this control information into the original planner. The basic approach is to guard each refinement candidate with the selection information. This forces a refinement application to fail quickly on subgoals to which the refinement should not be applied.

A decision is also made as to whether the control information has made the planner deterministic. If a refinement rule (or rule disjunct) covers no incorrect selection decisions in the induction phase, then it is assumed that the rule is fully accurate and no other refinement candidates will need to be considered. If a refinement rule could not exclude all incorrect decisions in the previous phase, then the planner is still allowed to backtrack over the selection of that refinement. This type of rule can still substantially improve planning efficiency by preventing many inappropriate applications of that refinement.

Figure 5 shows several learned selection rules for the first two refinement candidates (from Figure 1). The first rule allows `Unstack(A,B)` to be applied only when `A` is found to be on `B` initially, and `Stack(B,C)` should not be selected instead. The second and third rule allow `Putdown(A)` to be applied only when `A` should be placed on the table and not stacked on another block.

```
select-op((clear(B),S1),Steps,OrderCons,Links,Agenda,unstack(A,B)) :-
    find-init-state(Steps,Init),
    member(on(A,B),Init),
    not(member((on(B,C),S1),Agenda),member(on-table(B),Init)).

select-op((clear(A),G),Steps,OrderCons,Links,Agenda,putdown(A)) :-
    not(member((on(A,B),G),Agenda)).

select-op((clear(A),S1),Steps,OrderCons,Links,Agenda,putdown(A)) :-
    member((on-table(A),S2),Agenda),
    not(establishes(on-table(A),S3)).
```

Figure 5: Learned control rules for two refinement candidates

Experimental Evaluation

The blocksworld and logistics transportation domains were used to test the SCOPE learning system. In the logistics domain (Veloso 1992), packages must be delivered to different locations in several cities. Packages are transported between cities by airplane and within a city by truck. In both domains, a test set of 100 independently generated problems was used to evaluate performance. SCOPE was trained on separate example sets of increasing size. Ten trials were run for each training set size, after which results were averaged. Training and test problems were produced for both domains by generating random initial and final states. In blocksworld, problems contained two to six blocks and one to four goals. Logistics problems contained up to two packages and three cities, and one or two goals. No time limit was imposed on planning in either domain, but a uniform depth bound on the plan length was used during testing that allowed for all problems to be solved. All tests were performed on a Sun SPARCstation 5.

For each trial, SCOPE learned control rules from the given training set and produced a modified planner. Since SCOPE only specializes decisions in the original planner, the new planning program is guaranteed to be sound with respect to the original one. Unfortunately, the new planner is not guaranteed to be complete. Some control rules could be too specialized and thus the new planner may not solve all problems solvable by the original planner. In order to guarantee the completeness of the final planner, a strategy used by Cohen (1990) is adopted. If the final planner fails to find a solution to a test problem, the initial planning program is used to solve the problem. When this situation occurs in testing, both the failure time for the new planner and the solution time for the original planner are included in the total solution time for that problem. In the results presented in the next section, the new planner generated by SCOPE was typically able to solve 95% of the test examples.

Figures 6 and 7 present the experimental results. The times shown represent the number of seconds required to solve the problems in the test sets after SCOPE was trained on a given number of examples. In both domains, the SCOPE consistently produced a more efficient planner and significantly decreased solution times on the test sets. In the blocksworld, SCOPE produced modified planning programs that were an average of 11.3 times faster than the original planner. For the logistics domain, SCOPE produced programs that were an average of 5.3 times faster. These results

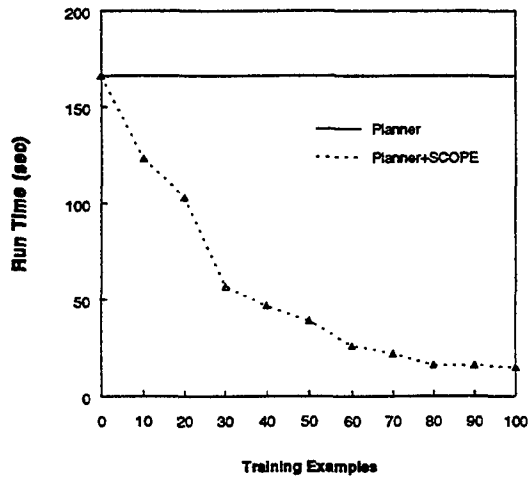


Figure 6: Performance in Blocksworld

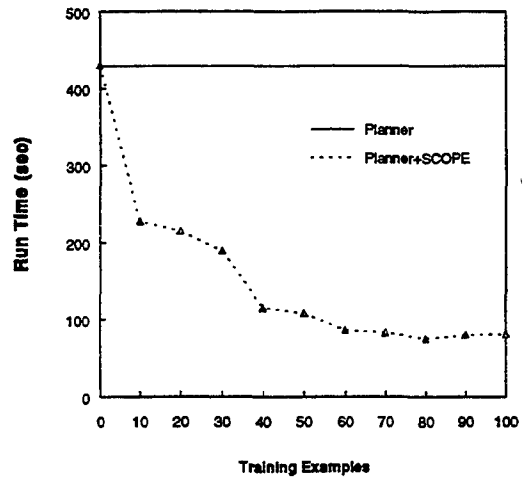


Figure 7: Performance in Logistics

indicate that SCOPE can significantly improve the performance of a partial-order planner.

Related Work

A closely related system to SCOPE is UCPOP+EBL (Kambhampati, Katukam, & Qu 1996), which also learns search control rules for UCPOP, but uses a purely explanation-based approach. Specifically, UCPOP+EBL employs the standard EBL techniques of regression, explanation propagation and rule generation to acquire search-control rules. Rules are learned only in response to past planning failures. This system has been shown to improve planning performance in several domains, including the blocksworld. To compare the two systems, we replicated an experiment used by Kambhampati, Katukam, & Qu (1996). Problems were randomly generated from a version of the blocksworld domain that contained between three to six blocks and three to four goals.[4] SCOPE was trained on a set of 100 problems. The test set also contained 100 problems and a CPU time limit of 120 seconds was imposed during testing. Data was collected on planner speedup and on the number of test problems that could be solved under the time limit. The results are shown in the following table.

System	*Orig Time*	*Final Time*	*Speedup*	*Orig %Sol*	*Final %Sol*
UCPOP+EBL	7872	5350	1.47X	51%	69%
SCOPE	5312	1857	2.86X	59%	94%

[4] In order to replicate the experiments of Kambhampati, Katukam, & Qu (1996), the blocksworld domain theory used for these tests slightly differed from the one used for the experiments presented previously. Both domains employed similar predicates however the previous domain definition consists of four operators while the domain used here has only two.

Both systems were able to increase the number of test problems solved, however, SCOPE had a much higher success rate. Overall, SCOPE achieved a better speedup ratio, producing a more efficient planner. By combining EBL with induction, SCOPE was able to learn better planning control heuristics than EBL did alone. These results are particularly significant since UCPOP+EBL uses additional domain axioms which were not provided to SCOPE.

Most other related learning systems have been evaluated on different planning algorithms, thus direct system comparisons are difficult. The HAMLET (Borrajo & Veloso 1994a) system learns control knowledge for the nonlinear planner underlying PRODIGY4.0. HAMLET acquires rules by explaining past planning decisions and then incrementally refining them. It is difficult to directly compare HAMLET and SCOPE for several reasons. First, HAMLET is directly integrated with the PRODIGY4.0 system, which does not use a partial-order planner. PRODIGY4.0 also employs some initial built-in control knowledge not given to our planner. HAMLET has successfully improved planner performance in the blocksworld and logistics planning domains. When making a rough comparison to the results reported in Borrajo & Veloso (1994b) , SCOPE achieves a greater speedup factor in blocksworld (11.3 vs 1.8) and in the logistics domain (5.3 vs 1.8).

Future Work

There are several issues we hope to address in future research. First, SCOPE should be tested on more complex domains which contain conditional effects, universal quantification, and other more-expressive planning constructs. We also plan to examine ways of using SCOPE to improve plan quality as well as planner efficiency. Borrajo (1994b) and Perez (1994) have used

learned control information to guide a planner towards better solutions. SCOPE could be modified to collect positive control examples only from high-quality solutions so that control rules are focused on quality issues as well as speedup. Lastly, we want to incorporate a method that directly evaluates control-rule utility. Replacing FOIL's information-gain metric for picking literals with a metric that more directly measures rule utility could improve ultimate planning performance.

Conclusion

SCOPE provides a new mechanism for learning search control information in planning systems. This system employs a novel learning technique that contains both explanation-based and inductive learning components. Simple, high-utility rules are learned by inducing concept definitions of when to apply plan refinements. Explanation-based generalization aids the inductive search by focusing it towards the best pieces of background information. Unlike most approaches which are limited to total-order planners, SCOPE can learn control rules for the newer, more effective partial-order planners. In both the blocksworld and logistics domains, SCOPE significantly improved planner performance; SCOPE also outperformed a competing method based only on EBL.

References

Barrett, A., and Weld, D. 1994. Partial order planning: Evaluating possible efficiency gains. *Artificial Intelligence* 67:71–112.

Bhatnagar, N., and Mostow, J. 1994. On-line learning from search failure. *Machine Learning* 15:69–117.

Borrajo, D., and Veloso, M. 1994a. Incremental learning of control knowledge for nonlinear problem solving. In *Proceedings of the European Conference on Machine Learning, ECML-94*, 64–82.

Borrajo, D., and Veloso, M. 1994b. Incremental learning of control knowledge for improvement of planning efficieny and plan quality. In *AAAI-94 Fall Symposium on Planning and Learning*, 5–9.

Cohen, W. W. 1990. Learning approximate control rules of high utility. In *Proceedings of the Seventh International Conference on Machine Learning*, 268–276.

DeJong, G. F., and Mooney, R. J. 1986. Explanation-based learning: An alternative view. *Machine Learning* 1(2):145–176. Reprinted in *Readings in Machine Learning*, J. W. Shavlik and T. G. Dietterich (eds.), Morgan Kaufman, San Mateo, CA, 1990.

Estlin, T. A. 1996. Integrating explanation-based and inductive learning techniques to acquire search-control for planning. Technical report, Department of Computer Sciences, University of Texas, Austin, TX. Forthcoming. URL: http://net.cs.utexas.edu/ml/

Gratch, J., and DeJong, G. 1992. COMPOSER: A probabilistic solution to the utility problem in speed-up learning. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 235–240.

Kambhampati, S., and Chen, J. 1993. Relative utility of EBG based plan reuse in partial ordering vs. total ordering. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 514–519.

Kambhampati, S.; Katukam, S.; and Qu, Y. 1996. Failure driven search control for partial order planners: An explanation based approach. *Artificial Intelligence*. Forthcoming.

Katukam, S., and Kambhampati, S. 1994. Learning explanation-based search control for partial order planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 582–587.

Langley, P., and Allen, J. 1991. The acquisition of human planning expertise. In *Proceedings of the Eighth International Workshop on Machine Learning*, 80–84.

Lavrač, N., and Džeroski, S., eds. 1994. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood.

Leckie, C., and Zuckerman, I. 1993. An inductive approach to learning search control rules for planning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1100–1105.

Minton, S.; Drummond, M.; Bresina, J. L.; and Phillips, A. B. 1992. Total order vs. partial order planning: Factors influencing performance. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, 83–92.

Minton, S. 1988. Quantitative results concerning the utility of explanation-based learning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 564–569.

Minton, S. 1989. Explanation-based learning: A problem solving perspective. *Artificial Intelligence* 40:63–118.

Mitchell, T. M.; Keller, R. M.; and Kedar-Cabelli, S. T. 1986. Explanation-based generalization: A unifying view. *Machine Learning* 1(1):47–80.

Mooney, R. J., and Califf, M. E. 1995. Induction of first-order decision lists: Results on learning the past

tense of English verbs. *Journal of Artificial Intelligence Research* 3:1–24.

Muggleton, S. H., ed. 1992. *Inductive Logic Programming.* New York, NY: Academic Press.

Pazzani, M., and Kibler, D. 1992. The utility of background knowledge in inductive learning. *Machine Learning* 9:57–94.

Penberthy, J., and Weld, D. S. 1992. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, 113–114.

Pérez, M. A., and Carbonell, J. 1994. Control knowledge to improve the plan quality. In *Proceedings of the Second International Conference of AI Planning Systems.*

Quinlan, J. 1990. Learning logical definitions from relations. *Machine Learning* 5(3):239–266.

Silverstein, G., and Pazzani, M. J. 1991. Relational clichés: Constraining constructive induction during relational learning. In *Proceedings of the Eighth International Workshop on Machine Learning*, 203–207.

Veloso, M. M. 1992. *Learning by Analogical Reasoning in General Problem Solving.* Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University.

Zelle, J. M., and Mooney, R. J. 1993. Combining FOIL and EBG to speed-up logic programs. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1106–1111.

Zelle, J. M., and Mooney, R. J. 1994. Combining top-down and bottom-up methods in inductive logic programming. In *Proceedings of the Eleventh International Conference on Machine Learning*, 343–351.

Forecasting of Options in the Car Industry Using a Multistrategy Approach

Stefan Ohl

Daimler-Benz AG, Research and Technology, F3S/E
P.O. Box 2360
D-89013 Ulm, Germany
ohl@dbag.ulm.DaimlerBenz.com

Abstract

In recent years, car options like air-conditioning, automatic gears, car stereo, power windows, sunroof etc. were getting more and more important for car manufacturers. Especially at those car manufacturers which offer cars individually according to customer requirements, the options affect 30% - 40% of the parts by now. Therefore, very detailed planning and forecasting of options becomes more and more important. Because customer behavior concerning the options varies from model to model and from country to country, it seems necessary and reasonable to forecast each option for each model and each country separately. The resulting huge number of data sets requires an automatic forecasting tool that adapts itself to the actual data sets and that requires almost no user interaction. Because depending on the characteristics of a time series the quality of the forecasting results varies a lot, and because "N heads are better than one," the basic idea is to select in a first step the most appropriate forecasting procedures. This selection is done by a decision tree which is generated by using a symbolic machine learning algorithm. Those selected forecasting methods produce different results that are in a second step combined to get a common forecast. In this approach there are integrated univariate time series used in the first step for running the prediction as well as symbolic machine learning algorithms for generating the decision trees as well as multivariate statistical methods and neural networks used in the combination step.

Introduction

Relevance of Options and Extras

In recent years, options of a car like air-conditioning, automatic gears, car stereo, power windows, sunroof etc. have become more and more important for car manufacturers (Dichtl et al., 1983). On the one hand, big business around the options and extras arises, but on the other hand, the huge number of extras offered optionally causes high costs as well. Mostly, these enormous costs result of a diversity of different parts and production processes that are influenced by the different options (Brändli et al., 1994; Brändli and Dumschat, 1995). In particular, at car manufacturers that offer cars individually according to customer requirements, the options affect 30% - 40% of the parts by now. Therefore, very detailed planning and forecasting of options becomes more and more important.

Situation at Mercedes-Benz

In the case investigated in this paper, the car manufacturer Mercedes-Benz offers about 400 different options for more than 100 different models structured into 4 different classes (C-, E-, S-, and SL-class) sold in more than 250 countries. In average, each customer orders about 10 - 20 options per car. Based on almost 600,000 produced cars per year, in average, only 1.4 cars are identical (Hirzel, 1995). That means there exist only a few identical cars, and in most cases these cars belong to bulk buyers like rental car companies.

Recently, even an increase in variety of models and options is expected, since new classes never offered before by Mercedes-Benz will be presented (Stegemann, 1995). Moreover, already existing classes will be extended by additional models, and completely new options like automatic windshield wipers or different kinds of navigation systems will be sold.

Analysis of Customer Behavior: Cumulated vs. Single Data

Ad Hoc Analysis

Since some options are not available for each class and because identical options differ quite often from class to class (e.g. air conditioning of C-class is different to air conditioning of E-class), it is necessary to differentiate extras per each single class. With regard to monthly

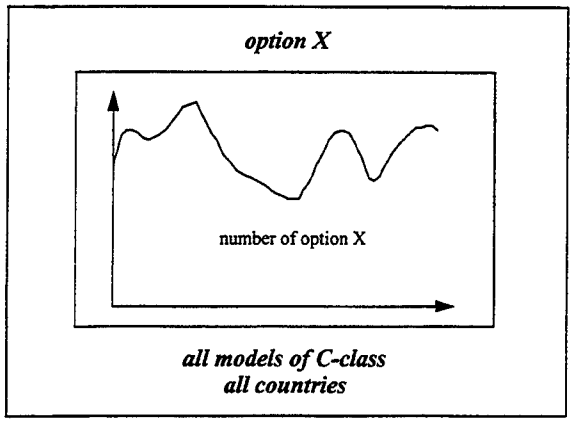


Fig. 1: Time series of single options.

cumulated data for about two years (e.g. all automatic gears sold per month in the C-class), cumulated time series of single options that represent absolute selling figures do not contain any systematic pattern as shown in figure 1. Therefore, it seems to be very difficult to forecast such kinds of time series as monthly data to a prediction horizon of 12 or even more months.

Utilizing Descriptive Statistics

However, looking at the data in a more detailed way using descriptive statistics some systematic pattern may be discovered. By analyzing the distribution of single options over different models and different countries where these options are sold, a lot of differences concerning the installation rate of an option to the total number of cars sold occur. For the options air conditioning (AC), automatic gears (AG), power windows (PW), heated seats (HS), sunroof (SR), and traction system (TS) this analysis for Mercedes C-class (Germany), and for the model C 220 (world wide), respectively, is presented in figure 2.

As it is shown in the left part of figure 2, for example, if more models C 280 and less models C 180 are produced in comparison to the previous month, the cumulative absolute number of air conditioning will increase, whereas the installation rate of air conditioning for each model will not fluctuate unsystematically a lot. Similarly, if in the current month more cars of the type C 220 are produced for Switzerland (CH) and less for Sweden (S) in comparison to the previous month, the cumulative absolute number of traction systems will increase and the cumulative installation rate of heated seats will decrease, whereas the single installation rate per model and country for these options will not vary randomly a lot (see right part of figure 2).

Based on this analysis, we detected that customer behavior concerning options varies a lot from model to model and from country to country (Ohl, 1995). Therefore, it seems reasonable to forecast and plan each single option as an installation rate for each model and each country where the option is available separately.

The less systematic kind of time series mentioned in the previous subsection of ad hoc analysis and presented in figure 1 is mainly due to the monthly shifting cars from one country to another country or from one model to another model as described before. Whereas the total number of cars produced per month is more or less stable, the number of one single model produced for one single country may vary much more (Hirzel and Ohl, 1995).

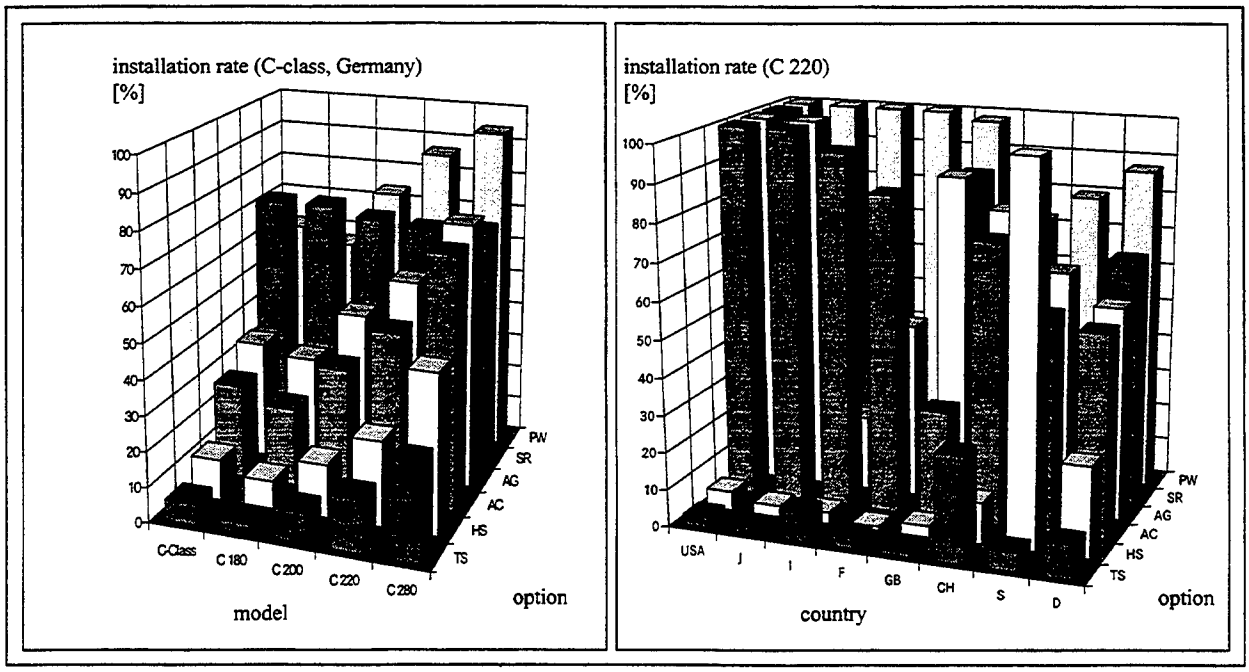


Fig. 2: Installation rates of different options at different models (Mercedes C-Class, Germany) and different countries (Mercedes C 220, world wide).

But this variation from one month to another month concerning one country or one model is in most cases planned and therefore well known, because there are seasonal effects, for instance, varying from one market to another market or from one model to another model. During summer time for instance, the number of convertibles sold is much larger than during winter time, and in markets like North America the selling figures increase when the first cars of a new model year are presented and sold in fall.

In contrast to the cumulative time series regarded in the previous subsection, analyzing single time series that represent not the absolute number but the installation rate of an option per model and country shows much more regular and systematic behavior as illustrated in figure 3.

Obviously, forecasting the single options as installation rates per model and country and multiplying these installation rates with the more or less well known planned absolute numbers of cars to be sold per model and country will yield better results than forecasting the absolute selling figures of options. This idea is illustrated in figure 3 where the time series of figure 1 is shown as the result of forecasting the single installation rates per model and country multiplied by the planned absolute number of cars as described before.

A Multistrategy Forecasting Approach

The resulting huge number of data sets requires an automatic forecasting and planning tool that adapts itself to the current data sets and that requires almost no user interaction. Another need concerning such a tool is performance, i.e. algorithms should not be too complex to avoid long computation times for doing the forecasts. The reason is that the forecasted selling figures of options are needed in a fixed company wide planning time table for further steps of the total planning process such as planning capacities of the different plants or planning the disposition of materials and parts for the next months (Ohl, 1996). This is due to the fact that the time needed for replacement of some parts and materials is longer than delivery time customers would accept. Therefore, frequently in the automotive industry it is necessary to do the disposition of parts and materials that will be delivered by internal and external suppliers before customers place their real orders.

Selection of Existing Methods

Classical Statistical Approaches. In general, four different ways of quantitative forecasting are yet well established (Makridakis et al., 1984):

- Purely judgmental or intuitive approaches.
- Causal or explanatory methods such as regression or econometric models (multivariate approaches).
- Time series methods (extrapolative univariate).
- Combinations of above mentioned techniques.

Purely judgmental or intuitive approaches are not appropriate because the user has to forecast each time series manually. The multivariate approach does not work at the application presented in this paper as well, since it is almost impossible to identify impact factors for different options for different countries. For example, what are the impact factors influencing the sunroof of the model C 220 in Italy? Therefore, only extrapolative univariate time series approaches are taken into consideration in the following.

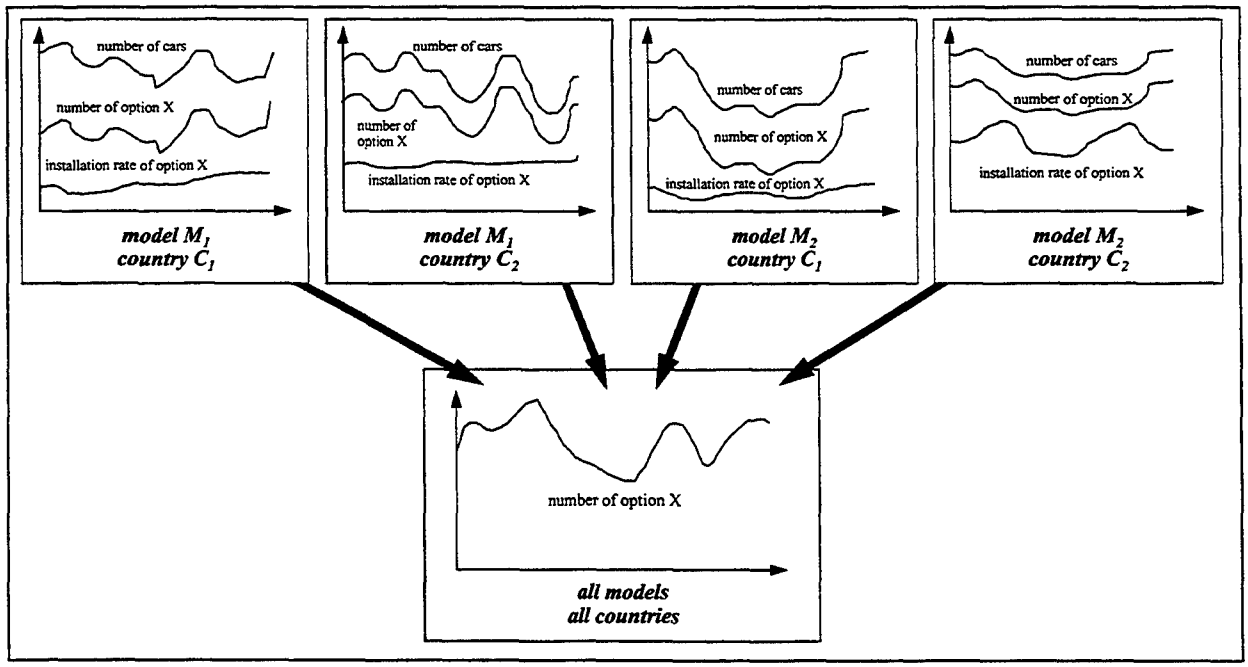


Fig. 3: Installation rates and absolute selling figures of options.

Reviewing extrapolative univariate time series analysis results in quite a lot of different methods. The most important and most popular are listed below (Jarrett, 1991):

- Naive forecast
- Moving averages
- Exponential smoothing
- Time series decomposition
- Adaptive filtering
- The Box-Jenkins methodology (ARIMA modeling)

From the scientific point of view, the most interesting approach of these methods listed above is the ARIMA modeling (Box and Jenkins, 1976). However, there are two very crucial disadvantages: The most important point in ARIMA modeling is model identification. As shown in different forecasting competitions (Makridakis et al., 1984; Makridakis and Hibon, 1979; Schwarze and Weckerle, 1982) even experts differ in choosing the most appropriate ARIMA model. And beyond, there exist forecasting methods producing in average better forecasts than ARIMA modeling does. Besides, as described before, an automatic forecasting system is needed, such that it is not possible to identify manually the appropriate model for each time series. Therefore, ARIMA modeling seems not to be reasonable for the prediction of all time series in our application.

But nevertheless, there exist definitely special types of time series, where ARIMA modeling will be the appropriate forecasting method, i.e. time series with a high autocorrelation. Besides, there exist approaches for identifying automatically the most appropriate model such as Akaike's information criterion (Akaike, 1976; Akaike, 1979). Therefore, the Box-Jenkins methodology will not be the appropriate approach for all time series, but possibly for a selected number of time series.

In the case of adaptive filtering (Kalman, 1960; Kalman and Bucy, 1961), the situation is very similar. It seems not reasonable to use this approach for all time series, but in many cases adaptive filtering will produce good prediction results without any user interaction. Therefore, adaptive filtering will not be used as forecasting technique for all time series, but definitely for the prediction of special types of time series. This result corresponding to the situation of ARIMA modeling is not surprising at all, because the adaptive filtering technique is very similar to ARIMA modeling.

On the one hand, forecasting time series using naive forecasts or moving averages does in a lot of cases not produce good results in comparison to other forecasting approaches. But on the other hand, there exist a lot of time series at Mercedes-Benz, where using naive forecasts or moving averages yields excellent results.

Discussing exponential smoothing (Brown and Meyer, 1961) means to discuss a lot of different forecasting techniques (Gardner, 1985), because there exist constant exponential smoothing approaches without any trend or seasonal parameters as well as there exist different exponential smoothing techniques considering linear and nonlinear trends as well as seasonal effects of the time series. Consequently, the different kinds of exponential smoothing could be depending on the structure of the investigated time series the appropriate approaches for our forecasting system. Generally, the most important and popular types of exponential smoothing approaches are:

- Single exponential smoothing (Brown, 1963)
- Linear exponential smoothing: Brown's one parameter method (Brown, 1963)
- Linear exponential smoothing: Holt's two parameter approach (Holt et al., 1960)
- Winters' three parameter method (Winters, 1960)

Symbolic Machine Learning and Neural Networks. Recently, the attention concerning the task of time series prediction is also focused on the application of neural networks (Weigend and Gershenfeld, 1994; Graf and Nakhaeizadeh, 1993). Although the development of neural networks at early stage was stimulated by modeling of learning process in human brain, the further development of this technology shows a very strong similarity with statistical approaches (Ripley, 1992).

There are some studies which compare neural networks with some statistical procedures like nonlinear regression from a theoretical point of view (Arminger, 1994; Ripley, 1992). However, it should be mentioned that the ability of adaptive learning which characterizes the most of neural networks is not implemented in statistical procedures like regression analysis or ARIMA modeling.

Generally, there are two alternatives to use neural networks for time series prediction:

- On the one hand, it is possible to use neural networks as multivariate forecasting techniques very similar to nonlinear regression. In this case the problem for our application is the same as with multivariate statistical prediction approaches: it is almost impossible to identify the impact factors for the huge number of time series concerning the different options in different countries.
- On the other hand, it is possible to use neural networks as univariate forecasting techniques very similar to statistical extrapolative univariate forecasting approaches. Here the problem is to identify the most appropriate model (Steurer, 1994).

Neural nets based learning algorithms are like the majority of statistical approaches polythetic procedures which consider simultaneously attributes for decision. Because of this reason, they need at least as much training examples as the number of initial attributes. It means, they are not appropriate at all for automatic attribute selection for the cases in which the number of attributes, in comparison to the number of training examples, is too high. This is often the case in dealing with time series data.

The main problem in using neural networks for prediction consists in finding the optimal network architecture. To realize this task, one has to divide the

available time series data into training set, test set, and validation set. Regarding the problem of limited number of observations in time series data which is discussed above, dividing the whole series into training set, test set, and validation sets leads to a still smaller training data set, in many circumstances.

Besides neural networks, some of symbolic machine learning algorithms based on ID3-concept (Breiman et al., 1984) can be used to predict the development of time series as well (Merkel and Nakhaeizadeh, 1992).

There are a lot of machine learning algorithms which can handle the classification task. Almost all of these algorithms are, however, not appropriate to handle the prediction task directly. The reason is that in contrast to classification, in prediction the class values are continuous rather than discrete. Exceptions are the ID3-type algorithms CART (Breiman et al., 1984) and NewId (Boswell, 1990) which can handle continuous-valued classes as well. Of course, by discretization of continuous values and building intervals, it is possible to transfer every prediction task to a classification one, but this procedure is connected, normally, with a loss of information.

The algorithms like CART and NewId can handle the continuous valued classes directly, and without discretization. They generate a predictor in the form of a regression tree that can be transformed to production rules. Furthermore, these learning algorithms apply a single attribute at each level of the tree (monothetic) and this is in contrast to the most statistical and neural learning algorithms which consider simultaneously all attributes to make a decision (polythetic).

The main advantage of symbolic machine learning approaches like regression trees is that it is possible very easily to involve other available information in prediction process, for example, by including the background knowledge of experts. Concerning rule generating from the data, it should be mentioned that this property of decision and regression trees is one of the most important advantages of these approaches. Other classification and prediction methods like statistical and neural procedures have not these property that allows considering other sources of information in a very flexible manner. In the case of statistical or neural classification and prediction algorithms, it is very difficult, and in some circumstances impossible at all, to consider such additional information.

However, like other approaches, prediction algorithms based on symbolic machine learning have also some shortcomings. Generally, they can not predict the values beyond the range of training data. Regarding the fact, especially, a lot of time series have an increasing (decreasing) trend component, it can be seen that by using just the raw class values, one can never achieve a predicted value which is outside the range of the class values used for training. But this disadvantage can be avoided by taking differences of the class values as it is the case with the Box-Jenkins approach.

To summarize, neural networks and symbolic machine learning algorithms are from the theoretical point of view very interesting, but they do not seem to be appropriate for the prediction of the huge number of time series in our application. Therefore, the forecast will be done by extrapolative univariate forecasting approaches. The only method of this class not taken into consideration any more is the method of time series decomposition because of the needed user interaction. Therefore, time series decomposition seems definitely not to be an appropriate forecasting method for performing automatic forecasts as it is needed in our application.

The Idea of Combination

After the decision to use different extrapolative univariate time series techniques except for time series decomposition for prediction of options at Mercedes-Benz, the next step will be the detailed selection of an adequate extrapolative univariate prediction model. This selection of the most appropriate forecasting model is a very hard task. In the case of the application presented in this paper, where totally different types of time series have to be forecasted, it is almost impossible to give an answer concerning the question, what generally the most appropriate forecasting model is.

There have been many comparative studies in forecasting. Unfortunately, they do not always agree on which are the best forecasters or on the reasons why one method does well and another does badly. One of the largest studies is that conducted by Makridakis and Hibon (1979) who concluded that simple methods often outperformed sophisticated methods like ARIMA modeling. The largest competition realized so far is that conducted by Makridakis (Makridakis et al., 1984) who concluded with very similar results. While the study of Makridakis and Hibon (1979) concerned 111 time series, the competition of Makridakis (Makridakis et al., 1984) concerned 1001 time series and represents a major research undertaking.

Previous studies by Newbold and Granger (1974) and Reid (1975) concluded that ARIMA modeling will be usually superior if it is appropriate. Generally, ARIMA modeling is regarded as giving extremely accurate forecasts provided the user is expert enough and sufficient computational resources are available, although simple methods are often more than adequate if the stochastic structure of the time series is sufficiently simple (Montgomery and Johnson, 1976; Jarrett, 1991).

In the meantime, many researchers have applied the relatively new techniques of neural networks and machine learning in the context of prediction. There are several recent studies comparing the performance of neural networks, machine learning and classical statistical algorithms using time series data: for example, Rehkugler and Poddig (1990), Schumann and Lohrbach (1992) as well as Graf and Nakhaeizadeh (1993).

All these studies concentrate on finding which method is best, or try to explain why one method is better than another. However, as Bates and Granger (1969) suggest, the point is not to pick the best method, but rather to see if it is possible to combine the available methods so as to get an improved predictor:

> OUR INTEREST is in cases in which two (or more) forecasts have been made of the same event. Typically, the reaction of most statisticians and businessmen when this occurs is to attempt to discover which is the better (or best) forecast; the better forecast is then accepted and used, the other being discarded. Whilst this may have some merit where analysis is the principal objective of the exercise, this is not a wise procedure if the objective is to make as good a forecast as possible, since the discarded forecast nearly always contains some useful independent information.

The general idea of combining different methods is even much older: "In combining the results of these two methods, one can obtain a result whose probability law of error will be more rapidly decreasing" (Laplace, 1818, in Clemen, 1989).

The basic principle of combining forecasts is explained in detail in figure 4. Based on a given database shown in the upper middle of figure 4, forecasting method A discovers the linear trend correctly, but does not detect the seasonality of the given data. In contrast to method A, forecasting procedure B discovers the seasonality but does not detect the linear trend. In conclusion, both methods contain useful information of the given time series, but neither approach A nor method B contains all relevant information. Thus, selecting one single method for prediction and discard the other one means to loose useful and important information. This is avoided by choosing the

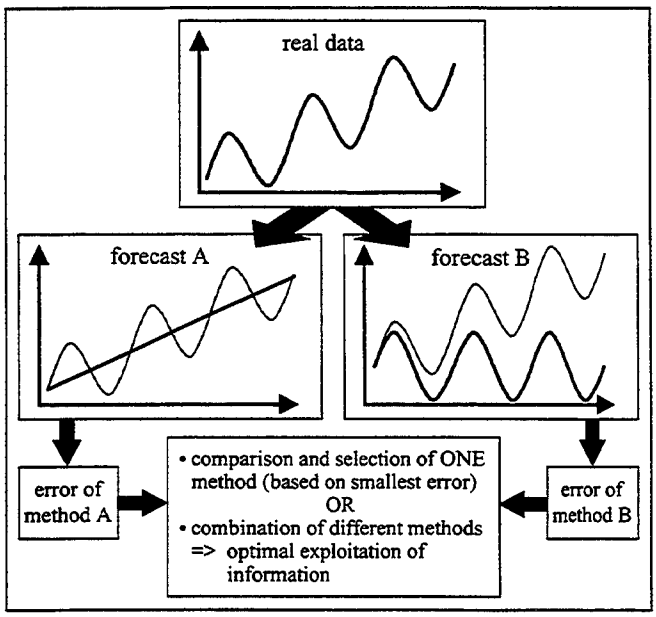


Fig. 4: Combination of forecasts.

multistrategy approach of combining both methods to get one common forecast.

This multistrategy approach of combining different forecasting methods is adapted in the considered application at Mercedes-Benz. Instead of selecting one single forecasting procedure, different forecasting techniques will be used in the forecasting system. Each of these different forecasting techniques will produce simultaneously its own forecast for the next 12 (or more) months, that will be combined subsequently in the combination step to one common forecast following Wolpert's general scheme of "Stacked Generalization" (Wolpert, 1992).

Following this approach of combined forecasts, there are three very important questions to answer:

- How many single univariate forecasting methods will be combined ?
- Which single univariate forecasting methods will be combined ?
- What is the appropriate algorithm for the combination ?

The first question to answer is, how many different forecasting approaches should be combined. Theoretically, it can be shown that the larger the number of single forecasting techniques used as input for combination, the better the produced results are (Thiele, 1993). In practice, on the one hand it can be shown, that accuracy will increase if more single forecasts are used as input (Makridakis and Winkler, 1983), but on the other hand, it can be shown as well that the difference of accuracy is not remarkable at all, if there are six, or seven, or eight different single prediction techniques used as input factors for the combination step (Hüttner, 1994). Besides, by using too many single forecasting methods as input for the combination step, there is the danger that the same or almost the same information is taken different times as input for the combination. That means to give a too strong weight to this information and not enough weight to the other independent information. Therefore, it seems reasonable not to use more than six single forecasting procedures as input for the combination step.

Classification of Time Series

The second question is, which forecasting methods should be selected for combination in what situation. As shown in different studies and forecasting competitions (Makridakis et al., 1984; Makridakis and Hibon, 1979; Schwarze and Weckerle, 1982), depending on the characteristics of the time series the different forecasting approaches yield different results. I.e., if there are autoregressive characteristics, ARIMA modeling will perform best, while for other types of time series other approaches will produce the best forecasting results.

Therefore the idea is, to select the six most appropriate single time series approaches depending on the characteristics of the different time series. The goal is to select for every time series the six most appropriate models

time series no.	attribute							forecasting approach					
	stationarity	trend	seasonality	autocorrelation	absolute level of figures	unsystematic fluctuation	...	best	2nd best	3rd best	4th best	5th best	6th best
1	yes	yes	no	no	high	no		E	L	W	A	G	H
2	no	yes	no	no	low	no		R	D	L	G	B	I
3	no	yes	no	no	middle	yes		W	C	L	B	D	G
4	no	no	yes	yes	middle	no		G	W	B	K	M	D
5	yes	no	yes	yes	high	no		F	B	V	E	B	D
6	yes	yes	no	no	low	yes		E	D	F	G	O	K
7	yes	no	yes	yes	low	yes		C	A	U	S	K	L
8	no	no	no	yes	high	no		L	K	S	B	F	J
9	yes	yes	no	no	middle	yes		H	I	D	L	R	C
10	no	no	yes	no	high	yes		A	D	C	G	I	F
...													

Fig. 5: Summary of ex post forecasting results.

and to combine these different forecasts subsequently in the combination step.

For the selection of most appropriate models a randomly selected sample of time series is analyzed and described in detail by using different attributes such as stationarity, trend, seasonality, unsystematic random fluctuation, autocorrelation, absolute level of figures, etc. The values of these attributes are determined using objective tests such as the Dickey-Fuller Test (Dickey and Fuller, 1979) for testing the stationarity and the Durbin-Watson test (Durbin and Watson, 1950, 1951, 1971) for testing autocorrelation.

After this description of all time series in the sample, more than 20 different single univariate prediction procedures as presented before make an ex post forecast for each of those time series. The different forecasting results of these ex post forecasts are compared in detail concerning the prediction quality. This is for analyzing, which prediction procedure performs best, second best, etc. for what type of time series.

As an outcome a table is generated, where the forecasting results are summarized as it is shown in figure 5. In this table the different time series of the selected sample and their characteristic attributes are shown. The single univariate forecasting techniques performing for each time series best, second best, third best etc. are summarized as well.

In best case, it would be sufficient to utilize the table shown in figure 5 to build an automatic decision system, to decide for which type of time series which forecasting techniques performs best, second best, third best, etc. The six best forecasting methods would be selected and used as input for the combination step.

But reality looks differently. For very similar time series with almost the same characteristics the different single prediction techniques perform differently. Therefore, the results of table 5 are not suitable to generate an automatic decision system directly out of it, for what type of time series which forecasting techniques perform best.

Under the assumption that the decision, which forecasting techniques perform best, second best, third best, etc. is a classification task, an appropriate tool to solve such kinds of classification problems is a machine learning algorithm. A classification tool accepts a pattern of data as input, and the output are decision rules or decision trees, which can be used for classifying new and unknown objects (see for more detail Breiman et al., 1984).

Therefore, a symbolic machine learning algorithm is integrated as a generator for an automatic decision system. The ID3-type (Quinlan, 1986, 1987) machine learning algorithm NewId (Boswell, 1990), for example, takes as input a set of examples and generates a decision tree for classification. The product of this machine learning algorithm is a decision tree that can assign an unknown object to one of a specified number of disjoint classes. In our application, such a new unknown object would be a new time series described by its characteristic attributes, and the disjoint classes are the different single forecasting approaches, which are used as input in the combination step.

Using NewId classifications are learned from a set of the attributes, which characterize the different types of time series. For every input of the combination step an extra decision tree is generated. That means that there exist six decision trees: the first decision tree for the first input of the combination step, generated by using all the attributes mentioned in figure 5 and as classification goal the row of best forecasting approaches. The second decision tree is generated by using the same attributes again, but the classification goal is not the row of best but of second best forecasting techniques. The third decision tree is generated by using the row of third best forecasting methods, etc.

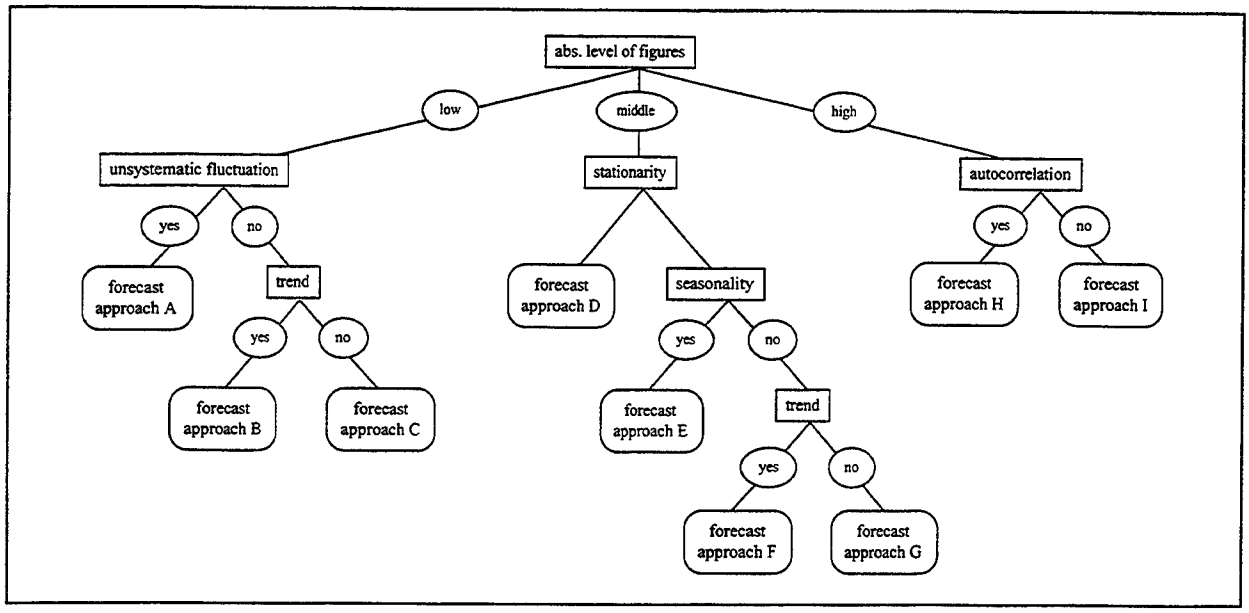


Fig. 6: Decision tree generated by NewId.

Figure 6 shows an example of a decision tree generated by using NewId. Since such a decision tree is generated for each input of the combination step, in sum six decision trees are generated. These six generated decision trees were pruned (Quinlan, 1987) and afterwards used as an automatic decision system, to decide what single forecasting approaches should be used to forecast the different types of time series.

Combining most Promising Approaches

As mentioned above, the third question to answer is, how the combination should be done. Generally, there are two paradigms for combining different forecasting approaches (Henery, 1995):

- On the one hand, it is possible to take the output of a prediction technique and use it as input for another forecast approach. This combination of different models means that only one algorithm is active at the same time (sequential multistrategy forecast).
- On the other hand, the combination of different algorithms at the same time is possible as well. In these approach several algorithms are active at the same time (parallel multistrategy forecast).

In a way, in our application we combine these two alternatives: the first step is kind of parallel multistrategy forecasting, because several forecasting algorithms are active at the same time, when the single prediction methods generate their own forecasts. The second step is kind of sequential multistrategy forecasting, because the output of some prediction methods is used as input for another forecasting procedure.

So far, we consider two different paradigms for the combination algorithm. The first way is to use statistical approaches (Thiele, 1993), the other way is to use neural network approaches (Donaldson and Kamstra, 1996).

From the scientific point of view, the most interesting alternative for the combination would be to use neural networks, i.e. using as learning algorithm the backpropagation algorithm (Werbos, 1974; Rumelhart et al., 1986; Rumelhart and McClelland, 1986). The outputs of the different single forecasting methods selected above are used in this approach as input values for the neural network, where the combination is performed.

But taking into consideration the huge number of time series to forecast, it seems not reasonable to use generally neural networks as combination tool. But nevertheless in some special situations it might be appropriate to use a neural network as forecasting tool in the combination step as it is shown in figure 7.

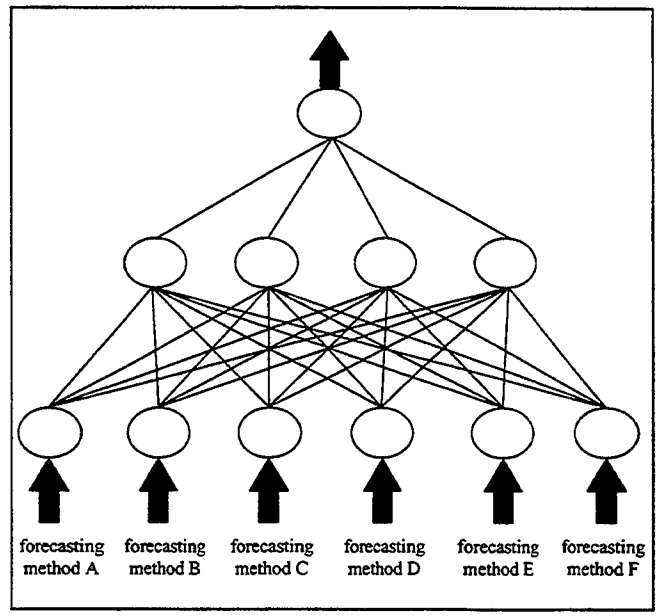


Fig. 7: Neural network as combination tool.

There is one big problem realizing the combination step by using neural networks. As already mentioned above, neural networks normally need a lot of data for training purposes. Because the time series in our application contain in many cases only 30 or 40 values, neural networks in these situations seem not to be the appropriate way for the combination step. But this problem can be solved by using techniques such as the cross-validation method (Lachenbruch and Mickey, 1968; Stone, 1974) and the bootstrap method (Efron, 1983).

The alternative way for running the combination is to use statistical approaches (Thiele, 1993). Generally, there are four classes of statistical combination approaches:

- Unweighted combination: the common forecast is the mean of the different single forecasts.
- Regression methods: the impact of each single forecast for the combination is computed based on a regression analysis (Granger and Ramanathan, 1984).
- Variance-covariance approaches: the impact of each single forecast for the combination is computed based on the error matrix (variance-covariance matrix) of the different methods used in the single prediction step (Bates and Granger, 1969; Clemen, 1989).
- probability-based combination approaches: the impact of each single forecast for the combination is computed based on probabilities, how well every single forecasting method performs (see for more detail Clemen, 1989).

Similar to the single forecasting approaches the best alternative of doing the combination depends on the characteristics of the time series. Therefore the same way is chosen as it is done for the selection of the best single forecasting methods. Again the machine learning algorithm NewId is used for generating a decision tree. This decision tree is used for an automatic selection, in which situation which way for combination seems most appropriate.

In many cases, already the simple mean of the different single forecasting procedures yields much better results than the best single forecast technique does. This confirm Kang (1986) ("A simple average is shown to be the best technique to use in practice.") and Clemen (1989) ("In many studies, the simple average of the individual forecasts has performed best or almost best".). But there are other situations as well, where it is appropriate to combine the forecasts of the single forecasting approaches by using more sophisticated techniques.

Theoretically, the regression and the variance-covariance approach yield identical results (Thiele, 1993), but caused by estimation errors in estimating the different parameters, in practice the regression method and the variance-covariance method produce different forecasts.

Sometimes, best forecasting results are produced by using probability-based approaches for running the combination step. The most interesting task using this alternative is the estimation of a priory probabilities, which are needed for running the combination of the different single forecasts (Clemen, 1989).

Evaluation

For evaluation purposes, there are two different parts of evaluation: The first part of evaluation is done by taking the DGOR (Deutsche Gesellschaft für Operations Research = German Society of Operations Research) forecasting competition of 1982 (Schwarze and Weckerle, 1982) as a general benchmark. The second and concerning the real application much more interesting part is to test the new multistrategy approach in comparison to the actual forecasting procedure, which is basically a weighted moving average method.

DGOR Forecasting Competition

The first part of evaluation is done by taking the DGOR forecasting competition of 1982 (Schwarze and Weckerle, 1982) as a general benchmark.

This competition was organized in 1982 by the forecasting group of the German Society of Operations Research and the task is to forecast 15 different time series of monthly data. All time series are real data concerning selling figures, turnover figures, etc. The longest time series contains 204 values, the shortest one contains 48 values.

Generally, there are two parts of the competition, where each time the task is to forecast the above mentioned 15 time series:

- In the first part of the competition the task is to forecast once 12 periods forecasting horizon.
- In the second part of the competition the task is to forecast 12 times 1 period forecasting horizon.

As an evaluation criteria for performance of the different forecasts Theil's U (Theil, 1971) is used.

For forecasting once 12 periods forecasting horizon, Theil's U is calculated as shown in (1) and for forecasting 12 times 1 period forecasting horizon, U is calculated as shown in (2):

$$U_1 = \sqrt{\frac{\sum_{t=t_0+1}^{t_0+T}\left(x_t - \tilde{x}_t\right)^2}{\sum_{t=t_0+1}^{t_0+T}\left(x_t - x_{t_0}\right)^2}} \quad (1) \qquad U_2 = \sqrt{\frac{\sum_{t=t_0+1}^{t_0+T}\left(x_t - \tilde{x}_t\right)^2}{\sum_{t=t_0+1}^{t_0+T}\left(x_t - x_{t-1}\right)^2}} \quad (2)$$

- x_t : real value of month t.
- $\tilde{x}_t$: forecasted value for month t.
- t_0 : actual month, where the forecast is done.
- T : total forecasting horizon (in our application: 12 months).

As shown in (1) and (2) U is calculated as square root of the quotient of the squared forecasting error divided by the squared forecasting error of the naive forecast.

Therefore, the most important advantage of using Theil's U is the fact that U automatically gives an answer concerning the question, whether it is worth at all using a more or less complex forecasting approach or whether it is sufficient to perform a prediction using the naive forecast:

- $U = 0$: perfect forecast, it is worthwhile to forecast.
- $0 < U < 1$: forecast better than naive forecast, it is worthwhile to do the forecast.
- $U = 1$: forecasting error equal to naive forecast, it is not worthwhile to do the forecast.
- $U > 1$: forecast worse than naive forecast, it is not worthwhile to do the forecast.

Using Theil's U as an evaluation criteria for performance of our forecasting system and taking the mentioned DGOR forecasting competition as a general benchmark, the approach presented yields very good results:

- In the first part of the competition (1 time 12 periods forecasting horizon) our method performs U_1=0.69, which is the second best result (best approach: U_1^{best}=0.67).
- In the second part of the competition (12 times 1 period forecasting horizon) our method performs U_2=0.58, which is also the second best result (best approach: U_2^{best}=0.55).

Note that approaches reaching better prediction results (i.e. lower U value) need manual fine tuning and produced better results after being tuned based on each single time series, whereas our combined approach presented in this paper works without any manual interaction and standard parameter values, but completely automatically.

Mercedes-Benz Data

The second and concerning the real application much more interesting part of evaluation is to test the new multistrategy approach in comparison to the actual forecasting procedure, which is basically a weighted moving average method.

Based on 200 randomly selected time series that are different to those time series used for performing the classification task in the previous section the new approach is evaluated by making ex post forecasts of the last 12 months. For this purpose similar to the evaluation using DGOR data, there are again two steps of evaluation, where each time the task is to forecast the above mentioned 200 time series of Mercedes-Benz data.

- In the first step the task is to forecast once 12 periods forecasting horizon.
- In the second step the task is to forecast 12 times 1 period forecasting horizon.

In earlier tests (Friedrich et al., 1995) it was already shown, that the actual weighted moving average forecasting method produces better results than the naive forecast does ($U < 1$). Therefore, it seems not reasonable at all to compare the new combined forecasting approach with the naive forecast as it is done by using Theil's U such as in the previous subsection.

Instead of using Theil's genuine U as an evaluation criteria a modified version of Theil's U as shown in (3) and (4) is used as evaluation criteria. In comparison to Theil's genuine U the modified U^* is calculated as square root of the quotient of the squared forecasting error not divided by the squared forecasting error of the naive forecast but divided by the squared forecasting error of the actual weighted moving average forecasting method. For forecasting once 12 periods forecasting horizon, the modified U^* is calculated as shown in (3) and for forecasting 12 times 1 period forecasting horizon, the modified U^* is calculated as shown in (4):

$$U_1^* = \sqrt{\frac{\sum_{t=t_0+1}^{t_0+T}(x_t - \tilde{x}_t)^2}{\sum_{t=t_0+1}^{t_0+T}(x_t - \hat{x}_t)^2}} \quad (3) \qquad U_2^* = \sqrt{\frac{\sum_{t=t_0+1}^{t_0+T}(x_t - \tilde{x}_t)^2}{\sum_{t=t_0+1}^{t_0+T}(x_t - \hat{x}_t)^2}} \quad (4)$$

- x_t : real value of month t.
- $\tilde{x}_t$: forecasted value for month t using the new combined forecasting approach.
- $\hat{x}_t$: forecasted value for month t using the actual weighted moving average forecasting approach.
- t_0 : actual month, where the forecast is done.
- T : total forecasting horizon (in our application: 12 months).

Similar to Theil's genuine U using Theil's modified U^* automatically gives an answer concerning the question, whether it is worth at all using the new approach or whether it is sufficient to perform a prediction using the actual weighted moving average forecasting approach:

- $U^* = 0$: perfect forecast, it is worthwhile to use the new approach.
- $0 < U^* < 1$: combined forecast better than actual forecast, it is worthwhile to use the new method.
- $U^* = 1$: forecasting error of combined forecast equal to that of actual method, it is not worthwhile to use the new approach.
- $U^* > 1$: combined forecast worse than actual method, it is not worthwhile to use the new approach.

Using the modified U^* as an evaluation criteria for performance of our forecasting system and taking the actual forecasting method as a benchmark, the combined approach presented yields very good results:

- In the first step (1 time 12 periods forecasting horizon) our new approach performs U^*_1=0.67.
- In the second step (12 times 1 period forecasting horizon) our new method performs U^*_2=0.68.

Obviously, the combined forecasting approach yields much better results than the actual method does. But due to the fact that neither Theil's genuine U nor the modified U^* are linear measures, it is very difficult to say exactly (i.e. in percent) how much better the new combined forecasting approach is in comparison to the actual method.

In summary, in comparison to the actual forecasting technique, our new combined approach yields much better results. Due to the fact that the results are confidential, they cannot be reported in more detail.

Conclusion

Summary

In conclusion, the task was to forecast cumulative time series of single options in the car industry, that do - at first glance - not contain any systematic pattern like trend or seasonality at all. By utilizing descriptive statistics, we discovered that the installation rate of single options for different models in different countries is more or less systematic and structured. Therefore, the forecast is done on installation rates, that are multiplied by the planned number of cars produced per month of each model for each country. Due to the huge number of time series to predict, automatic forecasting approaches without any manual user interaction are needed.

Because it is very difficult to identify external impact factors, no causal approaches like regression analysis were taken into consideration. Machine learning approaches and neural nets for running the prediction seem not to be appropriate as well. Therefore, univariate extrapolative prediction methods were chosen for running the forecast. But instead of selecting one single forecasting technique such as ARIMA modeling more than 20 different univariate forecasting approaches are tested for a randomly selected sample of time series. Depending on the characteristics of the time series, an automatic decision model was generated by using the machine learning algorithm NewId.

Using this decision model, for every time series the six most appropriate forecasting approaches are selected, and subsequently these six single forecasts are combined to get one common forecast. Generally, for performing this combination neural networks seem reasonable as well as statistical approaches. Which alternative is used for the combination step is again decided by a decision tree, that was generated again by using the NewId algorithm.

Selecting the most appropriate forecasting approaches depending on the characteristics of the time series and selecting again the most appropriate combination method for the combination seems to guarantee an automatic adaptation of the forecasting system to the time series.

In summary, in this paper a two step multistrategy forecasting approach is presented, that integrates machine learning techniques like decision trees for generating the automatic decision systems as well as several well known statistical univariate time series approaches such as ARIMA modeling, adaptive filtering, and exponential smoothing. Besides, multivariate statistical approaches such as linear regression are used in this approach as well as neural networks for performing the combination step. An overview of the total system is given in figure 8.

Future Work

Currently, improvement of single forecasting methods by automatic adaptation of parameters and improvement of the decision trees by increasing the sample of time series is under consideration. Besides, testing alternative combination approaches such as using other types of neural networks and other types of machine learning algorithms like Quinlan's M5 (Quinlan, 1992) for running the combination step are under consideration as well.

In addition, recently a completely new single forecasting approach called SAIPRO was developed (Friedrich and Kirchner, 1995; Friedrich et al., 1996), that will be integrated in the combined forecasting system next.

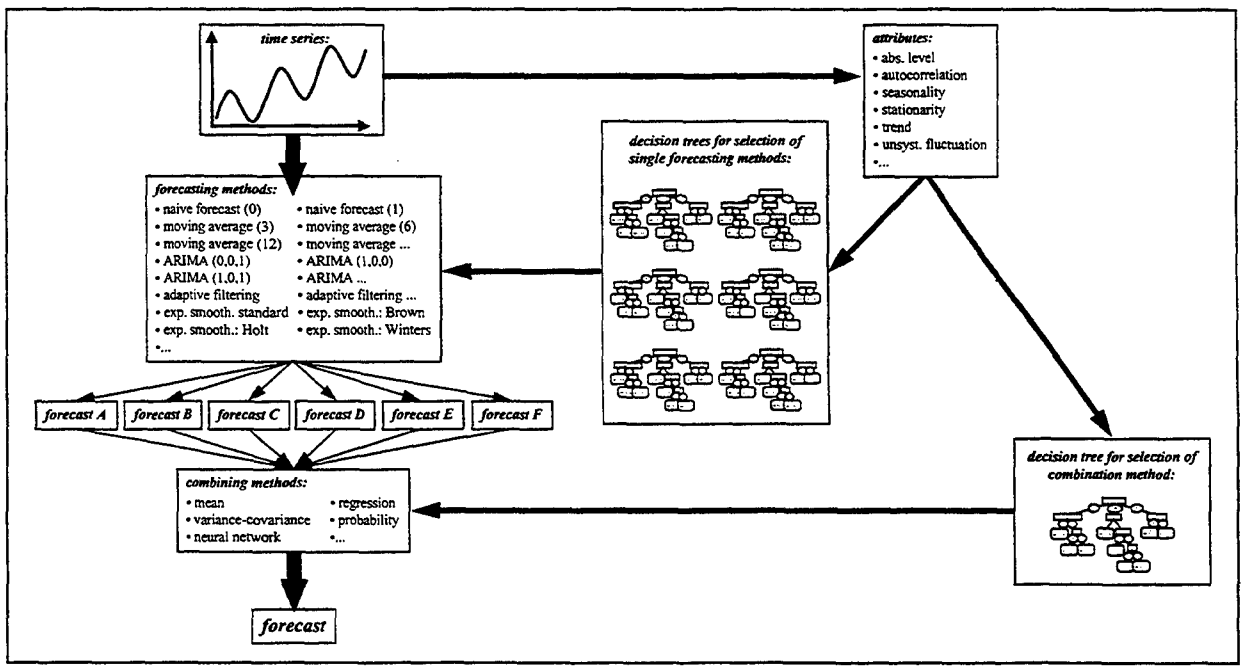


Fig. 8: Overview of the multistrategy forecasting system.

References

Akaike, H. 1976. Canonical correlation analysis of time series and the use of an information criterion. In: R.K. Mehra, and D.G. Lainiotis (eds.): System Identification, Academic Press, New York.

Akaike, H. 1979. A Bayesian extension of the minimum AIC procedure of autoregressive model fitting: *Biometrika 66: 237-242.*

Arminger, G. 1994. Ökonometrische Schätzmethoden für neuronale Netze. University of Wuppertal.

Bates, J.M., and Granger, C.W.J. 1969. The combination of forecasts: *Operations Research Quarterly 20: 451-468.*

Box, G.E.P., and Jenkins, G.M. 1976. Time series analysis, forecasting and control. Holden-Day, San Francisco.

Boswell, R. 1990. Manual for NewId. The Turing Institute.

Brändli, N., and Dumschat, R. 1995. Regelbasierte Stücklisten im Anwendungsdatenmodell ISO 10303-214: *Produktdaten Journal 1: 10-12.*

Brändli, N., Käfer, W., and Malle, B. 1994. Enhancing Product Documentation with new Information Technology Systems Based On Step: Proceedings of the 27th ISATA: Mechatronics and Supercomputing Applications in the Transportation Industries, 429-436.

Breiman, L., FRiedmann, J.H., Olshen, R.A., and Stone, C.J. 1984. Classification and regression trees. Wadsworth and Brooks, Monterey.

Brown, R.G. 1963. Smoothing, forecasting and prediction of discrete time series. Prentice Hall, Englewood Cliffs.

Brown, R.G., and Meyer, R.F. 1961. The fundamental theorem of exponential smoothing: *Operations Research 9: 673-685.*

Clemen, R.T. 1989. Combining forecasts: a review and annoted bibliography: *International Journal of Forecasting 5: 559-583.*

Dichtl, E., Raffée, H., Beeskow, W., and Kögelmayr, H.G. 1983. Faktisches Bestellverhalten als Grundlage einer optimalen Ausstattungspolitik bei Pkw-Modellen: *ZFBF 35 (2): 173-196.*

Dickey, D.A., and Fuller, W.A. 1979. Distribution of the estimators for autoregressive time series with a unit root: *Journal of the American Statistical Association 74: 427-431.*

Donaldson, R.G., and Kamstra, M. 1996. Forecast combining with neural networks: *Journal of Forecasting 15: 49-61.*

Durbin, J., and Watson, C.S. 1950. Testing for the serial correlation in least squares regression (I): *Biometrika 37: 409-428.*

Durbin, J., and Watson, C.S. 1951. Testing for the serial correlation in least squares regression (II): *Biometrika 38: 159-178.*

Durbin, J., and Watson, C.S. 1971. Testing for the serial correlation in least squares regression (III): *Biometrika 58: 1-19.*

Efron, B. 1983. Estimating the error rate of a prediction rule: improvements on cross-validation: *Journal of the American Statistical Association 78: 316-331.*

Friedrich, L., Hirzel, J., Kirchner, S., and Ohl, S. 1996. Ein neues Prognoseverfahren in der Automobilindustrie. Forthcoming.

Friedrich, L., and Kirchner, S. 1995. Das Prognose-Modell SAIPRO, Forschungsberichte 1-5: Technical Reports, Daimler-Benz AG, Berlin, Ulm.

Friedrich, L., Fooken, J., Hoffmann, S., Ludwig, R., Müller, E., Nehls, S., and Schütz, F. 1995. Forschungsbericht: Analyse und Prognose von Kundenauftragsdaten, Dept. of Mathematics and Physics, Technische Fachhochschule Berlin.

Gardner, E.S. Jr. 1985. Exponential smoothing: state of the art: *Journal of Forecasting 4: 1-28.*

Graf, J., and Nakhaeizadeh, G 1993. Application of neural networks and symbolic machine learning algorithms to predicting stock prices. In: V.L. Plantamura, B. Soucek, and G. Vissagio (eds.): Logistic and learning for quality software management and manufacturing. John Wiley & Sons, New York.

Granger, C.W.J., and Ramanathan, R. 1984. Improved methods of combining forecasts: *Journal of Forecasting 3: 197-204.*

Henery, R.J. 1995. combining forecasting procedures. In: Y. Kodratoff, G. Nakhaeizadeh, and C. Taylor (eds.): Workshop Notes of the Mlnet Familiarization Workshop *Statistics, Machine Learning and Knowledge Discovery in Databases.*

Hirzel, J. 1995. Programmplanung für variantenreiche Automobile im Wandel vom Verkäufer- zum Käufermarkt. In: G. Bol, T. Christ, and B. Suchanek (eds.): Das Potential der Lean-Technik in der Kraftfahrzeugwirtschaft: Aktuelle Entwicklungen der Material-Logistik in Theorie und Praxis.

Hirzel, J., and Ohl, S. 1995. Forecasting of Extras and Options in the Car Industry. In: P. Kleinschmidt (ed.): Operations Research Proceedings 1995, 409-410. Springer, Heidelberg.

Holt, C., Modigliani, F., Muth, J.F., and Simon, H.A. 1960. Planning production, inventories and work force. Prentice Hall, Englewood Cliffs.

Hüttner, M. 1994. Vergleich und Auswahl von Prognoseverfahren für betriebswirtschaftliche Zwecke. In P. Mertens (ed.): Prognoserechnung. 5., neu bearbeitete und erweiterte Auflage, Physika, Heidelberg.

Jarrett, J. 1991. Business forecasting methods. Second Edition, Basil Blackswell, Oxford.

Kalman, R.E. 1960. A new approach to linear filtering and prediction problems: *Journal of Basic Engineering 82: 35-44.*

Kalman, R.E., and Bucy, R.S. 1961. New results in linear filtering and prediction theory: *Journal of Basic Engineering 83: 95-107.*

Kang, H. 1986. Unstable weights in the combination of forecasts: *Management Science 32: 683-695.*

Lachenbruch, P., and Mickey, R. 1968. Estimation of error rates in discriminant analysis: *Technometrics 10: 1-11.*

Makridakis, S., Anderson, A., Carbone, R., Fildes, R., Hibon, M., Lewandowski, R., Newton, J., Parzen E., and Winkler, R. 1984. The forecasting accuracy of major time series. John Wiley & Sons, New York.

Makridakis, S. and Hibon, M. 1979. Accuracy of forecasting: an empirical investigation (with discussion): *Journal of the Royal Statistical Society A 142: 97-145.*

Makridakis, S. and Winkler, R.L. 1983. Averages of forecasts: some empirical results: *Management Science 29: 987-995.*

Merkel, A., and Nakhaeizadeh, G 1992. Application of artificial intelligence methods to prediction of financial time series: *Operations Research 91: 557-559.*

Montgomery, D.C. and Johnson, L.A.. 1976. Forecasting and time series analysis. McGraw-Hill, New York.

Newbold, P., and Granger, C.W.J. 1974. Experience with forecasting univariate time series and the combination of forecasts: *Journal of the Royal Statistical Society A 137: 131-165.*

Ohl, S. 1995. Analysis of the behavior of new car buyers. In: H.H. Bock and W. Polasek (eds.): Data analysis and information systems, statistical and conceptual approaches, studies in classification, data analysis, and knowledge organization, 197-207. Springer, Heidelberg.

Ohl, S. 1996. Forecasting virtual customer orders vs. forecasting combinations of options. Forthcoming.

Quinlan, R. 1986. Induction of decision trees: *Machine Learning 1: 81-106.*

Quinlan, R. 1987. Simplifying decision trees: *International Journal of Man-machine Studies 27: 221-234.*

Quinlan, R. 1992. Learning with continuous classes. In: Proceedings of the 5th Australian Joint Conference on Artificial Intelligence. World Scientific, Singapore.

Rehkugler, H., and Poddig, T 1990. Statistische Methoden vs. künstliche Neuronale Netzwerke zur Aktienkurs-prognose - Eine vergleichende Studie. Discussion paper, University of Bamberg.

Reid, D.J. 1975. A review of short-term projection techniques. In: Gordon, H. (ed.): Practical aspects of forecasting, Operational Research Society, London.

Ripley, B.D. 1992. Statistical aspects of neural networks. University of Oxford.

Rumelhart, D.E., Hinton, G.E., and Williams, R.J. 1986. Learning representations by back-propagating errors: *Nature 323: 533-536.*

Rumelhart, D.E., and McClelland, J.L. 1986. Parallel distributed processing. Vol. I & II. MIT Press, Cambridge.

Schwarze, J., and Weckerle, J. 1982. Prognose-verfahren im Vergleich. Technical report. University of Braunschweig.

Schumann, M., and Lohrbach, T 1992. Artificial neural networks and ARIMA models within the field of stock market prediction - a comparison. In: Preliminary Proceedings, BANKAI Workshop on Adaptive Intelligent Systems, Belgium, S.W.I.F.T.

Stegemann, B. 1995. Dampf in allen Klassen: *Auto Motor Sport 3: 8-13.*

Steurer, E. 1994. Prognose von 15 Zeitreihen der DGOR mit Neuronalen Netzen und Vergleich mit den Methoden des Verfahrensvergleichs der DGOR aus dem Jahre 1982: Technical Report, Daimler-Benz AG, Ulm.

Stone, M. 1974. Cross-validatory choice and assessment of statistical predictions: *Journal of the Royal Statistical Society 36: 111-133.*

Theil, H. 1971. Applied Economic Forecasting. North-Holland, Amsterdam.

Thiele, J. 1993. Kombination von Prognosen. Physika, Heidelberg.

Weigend, A.S., and Gershenfeld, N.A. 1994. Time series prediction: Forecasting the future and understanding the past. Addison-Wesley, Reading.

Werbos, P. 1974. Beyond regression. Ph.D. Thesis, Harvard University.

Winters, P.R. 1960. Forecasting sales by exponentially weighted moving averages: *Management Science 6: 324-342.*

Wolpert, D.H. 1992. Stacked generalization. *Neural Networks, 5: 241-259.*

How to predict it: Inductive Prediction by Analogy Using Taxonomic Information

Takashi Ishikawa
Kisarazu National College of Technology
2-11-1 Kiyomidai-higashi, Kisarazu, Chiba 292, JPN
takashi@j.kisarazu.ac.jp

Takao Terano
The University of Tsukuba
3-29-1 Otsuka, Bunkyo-ku, Tokyo 112, JPN
terano@gssm.otsuka.tsukuba.ac.jp

Abstract

This paper presents a novel machine learning technique in a logic programming environment: *Inductive Prediction by Analogy* (IPA). IPA learns the description of a target predicate similar to a source predicate from examples of the target predicate. A key feature of IPA is that it uses analogies to constrain the space of hypotheses using taxonomic information represented by first-order predicate logic. Typical problems addressed by IPA are to decide whether a given ground atom is valid or not, when no concept descriptions for the goal are available in a knowledge base. This is attained by the steps: 1) recognition of a candidate analogous source, 2) elaboration of an analogical mapping between source and target domains, 3) evaluation of mapping and inferences to given examples of the target predicate, and 4) consolidation of the outcome of the analogy. IPA can be applied to a wide variety of problems including classification problems in inductive learning. An experimental system of IPA is implemented in Prolog in order to use it as a knowledge acquisition tool for knowledge-based systems. The effectiveness of the technique is validated by a real world problem in molecular biology: the function prediction of proteins from their amino acid sequences.

Introduction

Analogical reasoning is an important research area in AI as a technique to reason from incomplete knowledge. In problem solving and learning, analogical reasoning promises to overcome the explosive search complexity of finding solutions to novel problems or inducing generalized knowledge from experiences (Hall 1989). The approach of the paper utilizes analogical reasoning in concept-learning. The key issue in this approach is how to recognize automatically an analogy between a source and a target and apply it to generating hypotheses for the target domain. Previous techniques in this approach usually use oracles from user to select candidates (De Raedt & Bruynooghe 1992) (Kedar-Cabelli 1985). Whereas the technique in this paper uses taxonomic information in the knowledge base for this purpose.

This paper presents a novel machine learning technique in a logic programming environment: *Inductive Prediction by Analogy* (IPA). IPA learns the description of a target predicate similar to a source predicate from examples of the target predicate. A key feature of IPA is that it uses analogies to constrain the space of hypotheses using taxonomic information represented by first-order predicate logic. Taxonomic information describes classification of symbols of a knowledge representation language. In a logical framework the symbols are constants to represent predicates, functions, and constant terms in sentences. IPA technique is based on an analogy as a mapping between constant symbols of a source predicate and a target predicate. Requirements for the technique are that it should be easy to represent by Horn clauses and should be easy to implement in Prolog as syntactic operations.

One of main objectives to develop IPA technique is to provide molecular biologists with an easy way to predict functions of a lot of proteins from their amino acid sequences in various kinds of database. Although there are so many amino acid sequence data available, conventional methods in molecular biology require tremendous and expensive efforts to predict the functions of proteins. They need novel but easy methods. Using AI-based symbolic techniques we will solve such real world problems.

This paper is organized as follows. The second section presents the framework for the

Inductive Prediction by Analogy and defines analogy using taxonomic information. The third section describes IPA technique in detail and gives an algorithm of IPA. In the fourth section, we report an experiment on IPA technique in molecular biology applied to the function prediction of proteins from amino acid sequences. In the fifth section we discuss the strengths and limitations of IPA technique and related work. Concluding remarks will follow in the final section.

The framework for the Inductive Prediction by Analogy

Inductive prediction consists in finding an inductive generalization of a set of examples of a concept and in applying it in order to predict whether a new instance is (or is not) a positive example of the concept (Tecuci 1993). In general, the process of inductive prediction is to generate concept-descriptions from examples. We use a logical framework for concept-learning (Genesereth & Nilsson 1987) (De Raedt & Bruynooghe 1992). In IPA technique, we assume that the knowledge base is represented by Horn clauses.

Definition 1. (Concept)
(1) A *concept* is a predicate.
(2) A *concept-description* is a set of (definite) Horn clauses defining a predicate.
(3) *Examples* are ground instances of a predicate. *Positive examples* are true and *negative examples* are false.
(4) A concept-description *covers* an example, if the example logically follows from the concept-description and the knowledge base.

A goal of the inductive prediction process is to generate hypotheses from which a target goal logically follows from the clauses in the knowledge base. The inductive prediction process can be used when a query in deductive inference fails because of the definition of the predicate is not defined in the knowledge base (Michalski & Tecuci 1994). We use a target goal as bias to restrict the form of hypotheses (Utgoff & Mitchell 1982). This type of bias focuses the concept-learner on generating hypotheses to cover a given target goal.

Definition 2. (Hypothesis)
(1) A *hypothesis* is a concept-description of a predicate not defined in the knowledge base.
(2) A *justified hypothesis* covers all positive examples and no negative examples.

Definition 3. (Inductive prediction)
(1) *Inductive prediction* is to generate justified hypotheses covering a target goal.
(2) A *target goal* is a ground instance of a predicate not defined in the knowledge base.

Inductive Prediction by Analogy

Analogy using taxonomic information

Analogical reasoning is a type of plausible reasoning based on the following assumption:

Assumption. If an analogy between a source and a target exists, then properties of the source can be projected to the target.

The notion *analogy* is defined informally as a representational mapping from the source to the target. We formalize *analogy* using taxonomic information in the following way. We use the terminology *source*, *target*, *mapping*, *analogy*, and *support* same as (Hall 1989). First, we formalize taxonomic information as a notion of *sort* (Frisch & Page 1990), then we define analogy.

Definition 4. (Sort)
(1) A *sort* τ is a subset of constant symbols of the domain. If constant c belongs to a sort τ, then we describe it as a ground clause $\tau(c) \leftarrow$.
(2) τ' is a *subsort* of a sort τ if, and only if, $\tau(X)$ can be deduced from $\tau'(X)$ and the knowledge base.

An *analogy* is considered as a mapping between elements of a source domain and a target domain. The *analogical mapping* associates or maps elements and descriptions from the source domain into the target domain. These mapped elements are analogical inferences and receive varying levels of supports from other mapped elements. This predicate mapping restriction constraints the space of possible clause mappings. IPA technique employs the observed similarity by mapping constants in concept-descriptions. Taxonomic information has the role of defining similarities among concept-descriptions. This mapping specifies the analogy among predicates. An analogy between literals is defined as follows:

Definition 5. (Analogy)
(1) A literal L_1 and a literal L_2 *have an analogy* when all corresponding constants in the literals belong to each common sort. The correspondence of symbols is decided according to the syntactical positions of symbols.
(2) A *common sort* of two constants c_1 and c_2, is a sort s, if ground clauses $s(c_1) \leftarrow$ and $s(c_2)$

$\leftarrow$ are defined in the knowledge base, or clauses $s(c_1) \leftarrow$ and $s(c_2) \leftarrow$ are deduced from the knowledge base.
(3) An *analogy* between literals L_1 and L_2 is the correspondence of their constants $\{ (c_{11}, c_{21}), (c_{12}, c_{22}), \ldots \}$ where $c_{11}, c_{12}, \ldots$ are constants in the literal L_1 and $c_{21}, c_{22}, \ldots$ are constants in the literal L_2.

Example 1 illustrates an example of an analogy between the solar system and an atomic model described in (Gentner 1983).

Example 1. prediction of an atomic model in which an electron revolves around a nucleus.

Target descriptions:
$\leftarrow$ revolves_around(electron, nucleus)
attracts(nucleus, electron) $\leftarrow$
more_massive_than(nucleus, electron) $\leftarrow$

Source descriptions:
revolves_around(P, S) $\leftarrow$
celestial_body(P), celestial_body(S),
attracts(S, P), more_massive_than(S, P)
attracts(sun, planet) $\leftarrow$
more_massive_than(sun, planet) $\leftarrow$

Taxonomic information:
physical_object(X) $\leftarrow$ celestial_body(X)
physical_object(X) $\leftarrow$ elementary_particle(X)
celestial_body(sun) $\leftarrow$
celestial_body(planet) $\leftarrow$
elementary_particle(electron) $\leftarrow$
elementary_particle(nucleus) $\leftarrow$

Analogy:
{ (nucleus, sun), (electron, planet),
(elementary_particle, celestial_body) }

Hypothesis:
revolves_around(P, S) $\leftarrow$
elementary_particle(P),
elementary_particle(S),
attracts(S, P), more_massive_than(S, P)

In this example, the mapping *{ (nucleus, sun), (electron, planet), (elementary_particle, celestial_body) }* is a recognized analogy. Constant terms *sun* and *planet* belong to a sort *celestial_body* as well as *nucleus* and *electron* belong to a sort *elementary_particle*. The sorts *celestial_body* and *elementary_particle* are subsorts of a sort *physical_object*. The target goal *$\leftarrow$ revolves_around(electron, nucleus)* is deduced from the hypothesis generated with this mapping and by substituting *celestial_body* with *elementary_particle* in the source concept-description.

The algorithm of IPA

The problem addressed by the Inductive Prediction by Analogy is formalized as follows:

Given:
- a target goal, which is a ground instance of a target concept
- examples of the target concept including the target goal
- background knowledge

Find: a hypothetical concept-description of the target concept such that the hypothesis covers all positive examples and no negative examples.

To solve the problem, IPA technique utilizes the process of analogical reasoning consisting of four main steps, *recognition*, *elaboration*, *evaluation*, and *consolidation* (Hall 1989):

(1) *recognition* of a candidate analogous source from a given target goal,
(2) *elaboration* of an analogical mapping between source and target domains, possibly including a set of analogical inferences,
(3) *evaluation* of the mapping and inferences in some context of use, including justification, repair, or extension of the mapping,
(4) and *consolidation* of the outcome of the analogy so that its results can be usefully reinstated in other context.

The steps of the algorithm of IPA technique is as follows:

Step 1: Recognition. Find a source ground clause which is similar to the target goal and search a source concept-description which covers the source ground clause. The recognized similarity gives a part of an analogy between the target and the source. If there exist multiple source ground clauses similar to the target goal, then IPA selects a candidate in the order of clause descriptions.

Step 2: Elaboration. First, transform the source concept-description using the analogy obtained in Step 1. By replacing each constant in the source concept-description with the corresponding constant in the analogy. Second, variablize all constants except for the replaced terms in the body of the transformed concept-description, and instantiate the variablized clause with the target goal and the knowledge base to get detailed analogy between the target and the source. Finally, transform the source concept-description using the detailed analogy to get a target concept-description.

Step 3: Evaluation. Add the target concept-description to the knowledge base and prove the

target goal. If the target goal can be proved, then let the target concept-description be a candidate. If not, retract the candidate from the knowledge base, then backtrack to Step 2. If the candidate covers all the positive examples and no negative examples, then let the candidate be a hypothesis. If such a candidate cannot be found, then backtrack to Step 1.

Step 4: Consolidation. Let the generated hypothesis be a concept-description of the target concept.

If the Herbrand base of a target domain is finite then the numbers of substitution for constant symbols is also finite and an analogical mapping is clearly decidable. This implies the algorithm of IPA terminates.

An illustration of the process of the algorithm of IPA is shown in Example 2.

Example 2. Decide whether a given ground atom *family(mother, mary, kate)* is valid or not, when no concept-descriptions for the goal are available in a knowledge base.

Given:
- a target goal ← *family(mother, mary, kate)*, which is a ground instance of a target concept *family(mother, X, Y)*
- an example *family(mother, lucy, sara)* ← of the target concept
- background knowledge including taxonomic information for constant terms

Find: a hypothetical concept-description of the target concept such that the hypothesis covers all positive examples and no negative examples.

In this example we assume that the following source concept-descriptions and background knowledge are defined in the knowledge base.

Target descriptions:
← family(mother, mary, kate)
sex(female, mary) ←
parent(mary, kate) ←
sex(female, lucy) ←
parent(lucy, sara) ←

Example (positive):
mother(lucy, sara) ←

Source descriptions:
family(father, X, Y) ←
sex(male, X),parent(X, Y)
family(father, john, tom) ←
sex(male, john) ←
parent(john, tom) ←
family(sister, X, Y) ←
sex(female, X),sibling(X, Y)
family(sister, ann, sara) ←
sex(female, ann) ←
sibling(ann, sara) ←

Taxonomic information:
family(father) ←
family(mother) ←
family(sister) ←
sex(male) ←
sex(female) ←

From the knowledge base above the algorithm of IPA generates a hypothesis *family(mother, X, Y)* ← *sex(female, X),parent(X, Y)* in the following way:

Step 1: Recognition. Find a source ground clause which is similar to the target goal ← *family(mother, mary, kate)*. There are two ground clauses, *family(father, john, tom)* ← and *family(sister, ann, sara)* ←, similar to the target goal, because analogies *{(mother, father)}* and *{(mother, sister)}* can be recognized using taxonomic information. The predicate *family* of arity one defines that the constants *father* and *mother* belong to the common sort *family*, and so on. The algorithm of IPA selects *family(father, john, tom)* ← as a candidate for the source ground clause according to the order of clause descriptions. Then search a source concept-description *family(father, X, Y)* ← *sex(male, X),parent(X, Y)*, which covers the source ground clause.

Step 2: Elaboration. First, transform the source concept-description using the analogy obtained in Step 1. By replacing the constant *father* in the source concept-description with the corresponding constant *mother* in the analogy. Second, variablize the constant *male* in the body of the transformed concept-description except for the replaced terms, and instantiate the variablized clause with the target goal and the knowledge base to get detailed analogy between the target and the source. Then the following target concept-description is generated: *family(mother, X, Y)* ← *sex(female, X),parent(X, Y)*.

Step 3: Evaluation. Add the target concept-description to the knowledge base and prove the target goal. As the target goal can be proved, then let the target concept-description be a candidate. As the candidate covers all the positive examples and no negative examples, then let the candidate be a hypothesis. Negative examples of a concept *family(mother, X, Y)* are all ground instances that satisfy *family(father, X, Y)* and *family(sister, X, Y)*.

Step 4: Consolidation. Let the generated

hypothesis be a concept-description of the target concept.

An experiment in molecular biology

In order to demonstrate the usefulness of IPA technique we apply it to the function prediction of proteins from amino acid sequences. An extremely important task in molecular biology is to predict the functions of a protein given its amino acid sequence (Shavlik et al. 1995) (Hunter 1993) (Schulz & Schirmer 1979). The problem of this experiment is to predict the functions of a protein having an amino acid sequence similar to that of bacteriorhodopsin (Henderson et al. 1990) in the SWISS-PROT protein sequence database (Bairoch & Boechmann 1994). Our experiment shows that the Inductive Prediction by Analogy technique is useful at least for domains with many structurally related predicates.

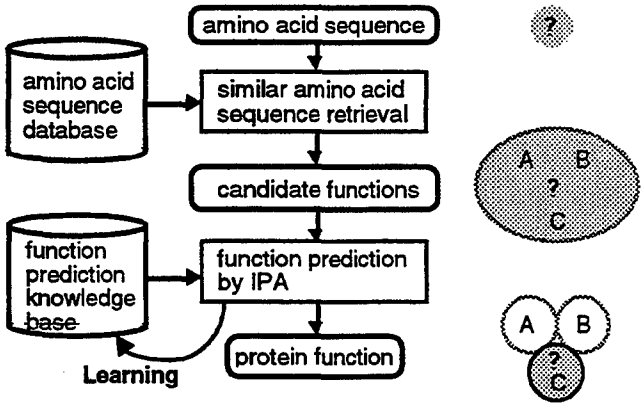


Figure 1. An overview of the system for function prediction of proteins

System. The system for the function prediction of proteins as shown in Figure 1 inputs an amino acid sequence of a protein with unknown functions and outputs a function of the protein. The system comprises two processes: *similar amino acid sequence retrieval* and *function prediction by IPA*. The similar amino acid sequence retrieval finds proteins having similar amino acid sequences with the inputted amino acid sequence from the amino acid sequence database. The system outputs a candidate function of retrieved proteins when their class falls into one function class. If the function classes are more than one class, then the system executes the next function prediction by IPA to refine function prediction of the protein. Since it is not sure that the target protein has the function same as the function of the protein having the retrieved amino acid sequences, the function prediction by IPA refines candidate protein functions to a specific protein function. The process generates concept-descriptions for the candidate functions, and apply them as classification rules to determine a specific protein function with the knowledge base. In the process, the system learns new knowledge and extends the knowledge base.

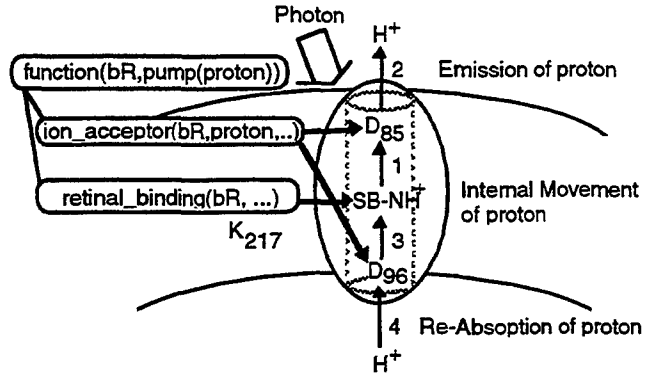


Figure 2. An abstract model of bacteriorhodopsin

Implementation. We have implemented a Prolog program based on the proposed algorithm of IPA. We have applied the program to the function prediction of proteins having amino acid sequences similar to bacteriorhodopsin (abbreviated as bR). bR is one of a few proteins whose structures and functions are well studied compared to other proteins. That is, we know its amino acid sequence, tertiary structure, and biological functions (Henderson et al. 1990). bR is a trans-membrane protein and has a function of proton pump that transports proton (H^+) across the membrane of cells. It is known that the structure of bR has seven alpha-helices and retinal as working material in it as shown in Figure 2. There are several proteins with amino acid sequences similar to that of bR in the SWISS-PROT database. Conventional method for the function prediction of proteins such as PROSITE (Bairoch & Boechmann 1994), however cannot discriminate these functions. Also bR is only one protein whose concept-description for its protein function can be given. In this experiment, the target protein

with an amino acid sequence similar to bR is halorhodopsin (abbreviated as hR) chosen from these proteins. hR transports chloride ion (Cl^-) instead of proton (H^+) in bR as an ion pump. Here we assume that we only know that the function of hR is one of functions of retrieved proteins including chloride pump. In the following, we will explain the process of IPA technique for the function prediction of hR. In the experiment, a hypothesis for the concept-description of chloride pump is generated by analogical reasoning from the concept-description of proton pump, and the function of hR is predicted by justifying the hypothesis.

Problem. The problem of the function prediction of hR is defined using Prolog descriptions as follows:

Given:
- a target goal
  `function(hR,pump(chloride))`
- examples of the target concept including the target goal
- background knowledge

Find: a hypothetical concept-description of the target concept such that the hypothesis covers all positive examples and no negative examples.

Knowledge base. First we describe the concept-description of proton pump in the knowledge base. The description has the head literal `function(X,pump(proton))` and the body literals including predicates `ion_acceptor` and `retinal_binding`. The predicate `ion_acceptor` represents that an amino acid with opposite charge of transporting `Ion` exists in `helix(I)` of the amino acid sequence of a protein `X`. The predicate `retinal_binding` represents that an amino acid `` `K` `` (retinal binding) exists in `helix(I)` of the amino acid sequence of a protein `X`. The terms `helix(1)`, ..., `helix(7)` represent indices of alpha-helices as a secondary structure of proteins. A part of Prolog descriptions of the knowledge base for this experiment is shown below.

```
%% Prolog descriptions of proton pump
function(X,pump(proton)) :-
  ion_acceptor(X,proton,helix(3),Pos1),
  ion_acceptor(X,proton,helix(3),Pos2),
  Pos1<Pos2,
  retinal_binding(X,helix(7)).
ion_acceptor(X,Ion,helix(I),Pos) :-
  trans_mem_sequences(X,SQ),
  member(NSQ,SQ,I),
  charge(Ion,C),opposite(C,AC),
  charge(Res,AC),
  string_member(Res, NSQ, P),
  in_membrane(P),length(NSQ,L),
  (member(I,[1,3,5,7]) ->
    Pos = P ; Pos is L-P+1).
retinal_binding(X,helix(I)) :-
  trans_mem_sequences(X,SQ),
  member(NSQ,SQ,I),
  string_member(`K`,NSQ,P),
  in_membrane(P).
in_membrane(Pos) :- Pos>3,Pos<20.
%% positive examples
function(bR,pump(proton)).
function(aR,pump(proton)).
function(sR,sensor(light)).
%% amino acid sequences of seven alpha-
helices
trans_mem_sequences(bR,
  [`WIWLALGTALMGLGTLYFLV`,
  `AITTLVPAIAFTMYLSMLLG`,
  `RYADWLFTTPLLLLDLALLV`,
  `ILALVGADGIMIGTGLVGAL`,
  `FVWWAISTAAMLYILYVLFF`,
  `TVVLWSAYPVVWLIGSEGAG`,
  `ETLLFMVLDVSAKVGFGLIL`]).
trans_mem_sequences(hR,
  [`LLSSSLWVNVALAGIAILVFVYMG`,
  `WGATLMIPLVSISSYLGLLSGLTV`,
  `SQWGRYLTWALSTPMILLA`,
  `SLFTVIAADIGMCVTGLAAAMTTS`,
  `FRWAFYAISCAFFVVVLSALVTDWA`,
  `AEIFDTLRVLTVVLWLGYPIVWAV`,
  `VTSWAYSVLDVFAKYVFAFILLRW`]).
trans_mem_sequences(aR,
  [`LWLGIGTLLMLIGTFYFIVKGW`,
  `SITILVPGIASAAYLSMFFGIGLTEV`,
  `ADWLFTTPLLLLDLALLA`,
  `IGTLVGVDALMIVTGLVGAL`,
  `WLFSTICMIVVLYFLATSLRA`,
  `LTALVLVLWTAYPILWIIGT`,
  `LGIETLLFMVLDVTAKVGFGFILL`]).
trans_mem_sequences(sR,
  [`TAYLGGAVALIVGVAFVWLLYRS`,
  `SPHQSALAPLAIIPVFAGLSYVGMAY`,
  `GLRYIDWLVTTPILVGYVGYAA`,
  `IIGVMVADALMIAVGAGAVV`,
  `ALFGVSSIFHLSLFAYLYVIF`,
  `QIGLFNLLKNHIGLLWLAYPLVWLFGP`,
  `GVALTYVFLDVLAKVPYVYFFYARRR`]).
%% taxonomic information
protein(bR).
protein(hR).
protein(aR).
protein(sR).
ion(proton).
ion(chloride).
protein_function(pump(proton)).
```

```
protein_function(pump(chloride)).
protein_function(sensor(light)).
helix(1).
helix(2).
helix(3).
helix(4).
helix(5).
helix(6).
helix(7).
```

Reasoning process. The algorithm of IPA starts from the following goal literal representing that the function of hR is chloride pump:

```
Goal = function(hR,pump(chloride))
```

Step 1: Recognition. Find a ground source clause (analogue) `function(bR,pump(proton))` similar to the target goal `function(hR, pump(chloride))` by making an analogy between the analogue and the target goal using taxonomic information defined in the knowledge base, and search a source clause which covers the ground source clause.

```
Analogue = function(bR,pump(proton))
Analogy = [[hR,bR],[chloride,proton],
    [pump(chloride),pump(proton)]]

Source_clause =
function(X,pump(proton)) :-
  ion_acceptor(X,proton,helix(3),Pos1),
  ion_acceptor(X,proton,helix(3),Pos2),
  Pos1<Pos2,
  retinal_binding(X,helix(7)).
```

Step 2: Elaboration. Apply the analogy to generate a candidate hypothesis for the target concept-description. To obtain a valid concept-description of `function(X,pump(chloride))`, `helix(3)` in the source clause should be transformed by variablizing two constants '3' into different variables and instantiating the variables with the knowledge base. The IPA program generates the following hypothesis for the concept-description of chloride pump.

```
Target_clause =
function(X,pump(chloride)) :-
  ion_acceptor(X,chloride,helix(3),
    Pos1),
  ion_acceptor(X,chloride,helix(6),
    Pos2),
  Pos1<Pos2,
  retinal_binding(X,helix(7)).
```

Step 3: Evaluation. Next, the generated hypothesis for the concept-description of chloride pump is refined so that the functions of proteins having similar amino acid sequences can be classified correctly. The refinement is executed by backtracking Step 2 until the condition is satisfied if needed.

Step 4: Consolidation. By adding the obtained concept-description to the knowledge base, we succeed in deciding the function of hR as a chloride pump from its amino acid sequence.

Result. By applying same procedure above, IPA technique successfully classifies protein functions of all proteins similar to bR, which are aR (archrhodopsin), sR (sensory-rhodopsin), and hR (halorhodopsin) (Ishikawa et al. 1995). Table 1 summarizes the classification features of the bacteriorhodopsin-like proteins.

Table 1. Classification features of bacteriorhodopsin-like proteins

protein	function	features
bR, aR	proton pump	two amino acids with negative charge in helix(3)
hR	chloride pump	one amino acid with positive charge in helix(3), one amino acid with positive charge in helix(6)
sR	not ion pump	no above features

Instead of using expensive biological efforts, the system using IPA technique learns these classification features from amino acid sequences and back-ground knowledge including taxonomic information. Although the discovered features for hR and sR have not yet completely certified in molecular biology, a recent research strongly suggests that these results are probable to be valid (Futai 1991). This means that the proposed technique is of use to support novel scientific discovery from biological database.

Discussion and related work

The proposed technique IPA is effective for generating classification rules from a very few number of training examples. Instead of applying the proposed technique using analogical reasoning to generate hypotheses of classification rules, it is difficult to generate the same classification rules by the direct applications of inductive inference techniques

to amino acid sequences. Because the proposed technique generates functional models for protein functions in top down manner, so obtained rules have abstract structures easy to understand for domain experts.

The use of analogical reasoning prunes meaningless generations of hypotheses in generating hypotheses. Therefore, the proposed technique improves the efficiency of learning. IPA technique generates a valid hypothesis for a target literal simply by variablizing constant terms in the explanation of the target literal and by instantiating the explanation without searching for generalizations and specializations of the terms.

IPA technique is applicable to domains with structurally related predicates, specifically to predicates of same syntactic structures. This limitation of IPA technique may be overridden by introducing the notion of abstraction-based analogy (Greiner 1988) (Ishikawa & Terano 1993). It is one direction of future research and we are conducting experiments for the purpose.

The technique Constructive Induction by Analogy (De Raedt & Bruynooghe 1992) uses second order schema, which is first introduced by (Yokomori 1986), whereas IPA technique uses taxonomic information to find analogies. However, the algorithm in (De Raedt & Bruynooghe 1992) requires asking the clause question to the user, whereas IPA technique automatically constrains candidates using taxonomic information.

The concept-learning by analogy in (Tecuci 1993) uses analogy based on *determination rules*, which are introduced in (Davies & Russell 1987). Determination rules are higher order rules and give too strong information to transform source knowledge to target knowledge. In IPA technique, simple taxonomic information plays a role of mapping the source symbols to the target symbols.

A paper concerned with logic program synthesis from examples (Sadohara & Haraguchi 1995) uses abstraction-based analogy (Greiner 1988) for explanation structures of logic programs. However, the algorithm is impractical since the enumeration of analogical mappings is computationaly explosive. Another paper concerned with analogical reasoning for logic programming (Tausend & Bell 1992) uses mechanisms of *Inductive Logic Programming* (Muggleton & Buntine 1988). However the input of the algorithm at least requires two analogical examples, whereas IPA technique automatically can find analogous examples from even only one example.

Conclusion

This paper has presented a novel machine learning technique *Inductive Prediction by Analogy* (IPA), which learns the descriptions of a target predicate similar to a source predicate from a few examples of the target predicate. IPA technique allows the learner to use analogical reasoning in generating hypotheses using taxonomic information represented by first-order predicate logic. The usefulness of the technique has been validated by a real world problem in molecular biology: the function prediction of proteins from their amino acid sequences. From the experiment, we have observed that the proposed technique generates *interesting* biological hypotheses from protein database. In this paper, although we have focused on molecular biology, however, IPA technique is applicable in various domains. We have succeeded in solving such problems as discovery of the Pythagorean theorem and prediction of the Rutherford model of atom (Ishikawa & Terano 1994). Therefore, we believe IPA technique will provide a simple but strong way in knowledge system development.

References

Bairoch, A and Boechmann, B. 1994. *Nucleic Acids Res.* 22:3578-3580.

Davies, T. R. and Russell, S. J. 1987. A logical approach to reasoning by analogy. In Proceedings of the International Joint Conference on Artificial Intelligence 1987, 264-270.

De Raedt, L. and Bruynooghe, M. 1992. Interactive concept-learning and constructive induction by analogy. *Machine Learning* 8:107-150.

Frisch, A. M. and Page Jr., C. D. 1990. Generalization with taxonomic information. In Proceedings of AAAI-90, 755-761.

Futai M. eds. 1991. *Bio-membrane Engineering* (in Japanese).:Maruzen.

Genesereth, M. and Nilsson, N. J. 1987. *Logical Foundations of Artificial Intelligence.*:Morgan Kaufmann.

Gentner, D. 1983. Structure mapping: A theoretical framework for analogy. *Cognitive*

Science 7:155-170.

Greiner, R. 1988. Learning by understanding analogies. *Artificial Intelligence* 35:81-125.

Hall, R. P. 1989. Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence* 39:39-120.

Henderson, R. et al. 1990. J. Mol. Biol. 213:899-929.

Hunter, L. ed. 1993. *Artificial Intelligence and Molecular Biology*.:AAAI Press.

Ishikawa, T. and Terano, T. 1993. Analogy by Abstraction: Theory of Case Retrieval and Adaptation in Inventive Design Problems. In Proceedings of AAAI-93 CBR workshop. also to appear in *Expert Systems with Applications* 10.

Ishikawa, T. and Terano, T. 1994. A Formulation of Analogy by Abstraction and Its Implementation with Logic Programming (in Japanese). In Proceedings of Knowledge Reformation Symposium 1994, 156-167.

Ishikawa, T., Mitaku, S., Terano, T., Hirokawa, T., Suwa, M., and Seah, B-C. 1995. Building a knowledge-base for protein function prediction using multistrategy learning. In Proceedings of Genome Informatics Workshop 1995, 39-48.

Kedar-Cabelli, S. T. 1985. Purpose Directed Analogy, In Proceedings of the Cognitive Science Society, Irvine, CA, August 1985.

Michalski, R.S. and Tecuci, G. eds. 1994. In *Machine Learning: A Multistrategy Approach Vol. IV* .:Morgan Kaufmann. 3-62.

Muggleton, S. and Buntine, W. 1988. Machine invention of first-order predicates by inverting resolution. In Fifth International Conference on Machine Learning:Morgan Kaufmann.

Sadohara, K. and Haraguchi, M. 1995. Analogical logic program synthesis from examples. In ECML-95, 232-244.

Schulz, G. E. and Schirmer, R. H. 1979.*Principles of Protein Structure* .:Springer-Verlag.

Shavlik, J., Hunter, L., and Searls, D. eds. 1995. Special Issues on Applications in Molecular Biology, *Machine Learning* 21.

Tausend, B. and Bell, S. 1992. Analogical reasoning for logic Programming. In *Inductive Logic Programming*.:Academic Press.

Tecuci, G. 1993. Plausible justification trees: A framework for deep and dynamic integration of learning strategy. *Machine Learning* 11:237-261.

Utgoff, P. E. and Mitchell, T. M. 1982. Acquisition of appropriate bias for concept learning. In Proceedings of Second National Conference on Artificial Intelligence, 414- 418.

Yokomori, T. 1986. Logic Program Forms. *New Generation Computing* 4:305-320.

Addressing Knowledge Discovery Problems in a Multistrategy Framework

Kenneth A. Kaufman

Machine Learning and Inference Laboratory,
George Mason University,
Fairfax, Virginia, 22030, USA
kaufman@aic.gmu.edu

Abstract

This paper discusses a methodology for multistrategy data analysis based on the application of diverse learning and discovery programs and tools and how it approaches some of the difficulties posed by the knowledge discovery task. Research in the area of integrated learning systems has led to the development of INLEN, an intelligent assistant for discovering knowledge in large databases. The architecture of INLEN is based on the interaction of a number of *knowledge generation operators* – manifestations of diverse learning tools within a uniform environment. Examples of the system's application to databases consisting of world economic and demographic facts demonstrate its operation. During its development, INLEN has encountered problems inherent in the application of symbolic learning programs to database analysis that do not appear in the laboratory environment; such problems are described, and the responses to these problems that have been built into INLEN are discussed.

Introduction

As the amount of electronically available information has grown, it has become both critically important and increasingly difficult to analyze the data to derive desired knowledge from them. Traditionally, tools for data analysis have employed mostly statistical concepts and methods. These methods can be particularly useful for such tasks as detecting statistical trends, correlations between attributes, data distributions, etc. They are, however, limited in the types of knowledge and regularities they can derive from data.

For example, a statistical analysis can detect a correlation between given factors, but cannot produce a conceptual explanation why such a correlation exists, nor can it formulate any specific quantitative and/or qualitative law(s) responsible for this correlation. A statistical technique can determine a central tendency and variability of some properties, or fit a curve to a set of datapoints, but it cannot explain them in terms of causal dependencies or qualitative relationships. Attributes that define a similarity and the measures of similarity involved must be given in advance. In short, these techniques require that an interpretation of the findings – a "conceptual" analysis of data – be performed by a human analyst. As the quantity of available data increases, the complexity of such an analysis can easily outstrip human capabilities.

Because of this, the machine learning community has taken an interest in the problem of knowledge extraction from databases. Machine learning approaches can overcome some of the limitations inherent in traditional data analysis methods. For instance, constructive induction and deduction methods can improve the data description space based on the nature of the data itself, the knowledge learned by the discovery system, and/or the background knowledge provided by the domain expert. Symbolic learning methods have the advantage of representing their knowledge in such a way that it is very easy for users to understand and explain the meaning of the discovered knowledge.

Machine learning researchers have developed an assortment of domain-independent programs that can each perform a narrow set of symbolic learning tasks. The weakness of these programs is the fact that they are so task-specific. No conceptual clustering program, for example, can generate equations governing quantitative data, create rules distinguishing between classes of objects, select representative examples from a larger database, or improve a ruleset based on new data.

In general, depending on the situation and the data itself, an analyst may be seeking:

- The most important factors influencing the observed behavior of a system or a process and their relationship to this behavior.
- The functional or logical dependencies that exist among concepts and attributes in the database.
- A means of determining whether a certain condition is present.
- The most illustrative examples of a given behavior.
- A listing of the elements of a system that are not behaving according to the assumed model.
- The best actions to be taken given the observed behavior.
- An understanding of how a system is changing over time, and what types of future behavior can be anticipated.

- A concise description of the data that will highlight the important trends or exceptions.
- An organization of the data into a useful hierarchy of categories.
- A meaningful consolidation of facts from different sources that can be of use in pattern discovery.
- A collection of elements extracted from a large pool of data that the analyst is likely to find interesting or useful.

Recent research in multistrategy learning (e.g., Michalski & Tecuci 1991; 1993) has attacked the problem of integrating diverse learning tools into composite systems whose wholes may exceed the sums of their parts, in order that single systems may provide many of these answers on request.

One such approach to this problem, called INLEN, applies an integrated system to the specific problem of knowledge discovery in databases (Kaufman, Michalski & Kerschberg 1991). It was designed to overcome some of the limitations of statistical data analysis by applying advanced methods of machine learning. Its architecture integrates database, knowledge base and machine learning technologies into a single package for data analysis and knowledge discovery. In doing so, it offers a data analyst a powerful tool for discovering patterns of non-statistical nature, determining logic-style data descriptions, and producing justifications or explanations of the discovered patterns (Michalski et al. 1992).

Symbolic learning programs, such as those that comprise many of INLEN's available tools, can determine rules for distinguishing between many classes of items, find conceptually useful ways to group objects, apply knowledge in order to predict missing values in a data set, select representative subsets of a large data set best suited for a particular learning task, etc. The various learning and discovery programs in INLEN are accessed in the form of *knowledge generation operators* (KGOs) that can be applied in sequence, with each KGO capable of taking advantage of its predecessor's findings. Because of the symbolic nature of these operators and their adaptability to different problems, this methodology is well-suited for many domains in which databases contain a large number of records and attributes, and in which the results of analysis must be understood by both experts and non-experts.

The conceptual architecture – employing a large set of specialized operators as tools that may be called upon when needed – has been employed in domains other than knowledge discovery in databases. For instance, CONDOR (Strat 1992) follows a similar philosophy in the domain of computer vision. Here, different operators (e.g., edge detection) are invoked by a heuristic-based control engine in an attempt to recognize different viewed objects.

Among other efforts to apply multistrategy methods to data analysis are several systems incorporating symbolic learning. For example, Alexander, Bonissone and Rau (1993) have developed a system for discovering knowledge that can be used to improve marketing strategies. It combines the C4.5 decision tree learning program (Quinlan 1990) with a statistical analysis system in order to extract information from a sales database. RECON (Simoudis et al. 1994) combines inductive and deductive reasoning into a general-purpose knowledge discovery environment. In comparison with these systems, INLEN focuses more on symbolic learning operators, and is designed for the incorporation of a large number of tools, rather than just one or two for inductive reasoning. By necessity, this leads to an increase in the complexity of the system's infrastructure and its knowledge base.

This paper describes the INLEN architecture, presents examples of INLEN's discovery process through its application to the analysis of several databases consisting of world economic and demographic facts, and then presents some of the problems inherent in applying multistrategy symbolic learning to knowledge discovery and the way in which they have affected the development of INLEN. We conclude by summarizing the development status of INLEN, the advantages and current limitations of this methodology, and outline the plans for future research.

The INLEN Methodology

The INLEN system integrates a database, knowledge base and machine learning technologies into a unified knowledge discovery environment (Figure 1). The system's components consist of a relational database, a knowledge base, and three sets of operators (Kaufman, Michalski & Kerschberg 1991). The database is used for storing data, and can be modified by the user through *data management operators*. The knowledge base stores and maintains rules, equations, decision procedures, representative examples, and concept hierarchies that are employed in the process of data analysis and knowledge discovery. The knowledge base is made up of *knowledge segments*, which store declarative, procedural, and exemplary knowledge about the application. This knowledge may include the domains of the attributes, background knowledge relevant to the problem, discovered knowledge, and any knowledge entered or modified by the user. The contents of the knowledge base can be modified by an expert through *knowledge management operators*.

The central discovery engine of the system consists of a set of operators, called *knowledge generation operators*, that invoke learning, discovery and inference programs to perform data analysis tasks. These operators take input from the database and/or knowledge base, and produce outputs that enhance the data and/or knowledge bases. These operators are adept at diverse tasks, such as learning symbolic rules to differentiate among several classes of data items, conceptually dividing a data set into two or more groups (conceptual clustering), modifying the representation space into one more suitable for a particular learning task through feature selection or constructive induction, testing a set of knowledge for consistency with respect to new data, and predicting values for missing data.

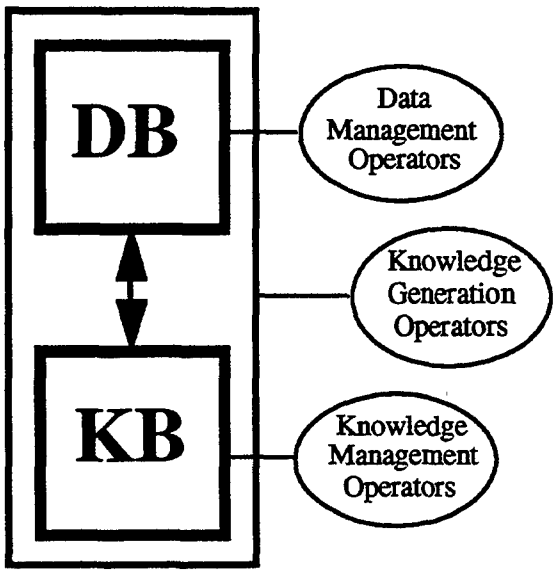


Figure 1. High-Level Architecture of INLEN

The application of these operators is controlled by the user, while the system extracts the necessary information from the data and knowledge bases to complete the input to the learning operator. In the knowledge base more information is maintained than the operators might return in stand-alone mode; facts and links that might be useful to another operator in the future are stored in the output knowledge segments. For instance, if a rule learning operator is used to find conditions distinguishing between two sets of data records, INLEN will retain not only the discovered rules, but also the learning mode and parameters used, an estimate of the informational value of the rules and their component conditions, and links to the records that satisfy the rule. This information may be later accessed by a user or another operator, such as a prediction engine or a program for selecting representative examples from the dataset.

The idea of such a multi-operator approach to knowledge discovery was formulated by Michalski in the 1980s. The first such effort, from which INLEN derived much of its conceptual architecture, was the QUIN system (Query and Inference), a combined database management and data analysis environment (Michalski, Baskin & Spackman 1982; Michalski & Baskin 1983; Spackman 1983).

Among the knowledge generation operators employed by INLEN are ones for rule generation (from sets or sequences of examples), decision tree generation (from examples or rules), equation generation (from quantitative and qualitative data), conceptual cluster and taxonomy formation, knowledge transformation (e.g., through abstraction, generalization, or incremental learning), representation space modification (through feature selection, constructive induction, and attribute quantization), event set generation (through example selection, simulation or prediction), relational analysis (through statistical and non-statistical metrics), knowledge testing for consistency and completeness, and concept visualization. The programs that form the foundations for these operators are cited and described individually in (Michalski et al. 1992).

These operators also represent various knowledge transmutations as are catalogued in Michalski's Inferential Theory of Learning (Michalski 1993). For instance, learning classification rules from examples involves an inductive generalization transmutation; conceptual clustering is an act of inductive agglomeration; representation space modification can consist of generation, selection or abstraction transmutations; and knowledge-based prediction of missing values incorporates a similization transmutation.

An important feature of the knowledge generation operators is that many of them allow the user to define parameters and tailor their performance to a specific learning task. In this way, the user (data analyst) can specify from among the many possible outputs consistent with the data and pinpoint which type of knowledge is likely to be most useful. Two examples below illustrate the choices of parameters when using a knowledge generation operator.

One of the backbone abilities of INLEN is rule learning from examples, performed using the AQ15c learning engine (Wnek et al. 1995). The user can accept the default parameters or instruct the program to bias its rule selection toward either highly specific characterizations, short rules for discrimination, a set of maximally simple rules, or rules that tend to incorporate or avoid specific attributes if at all possible. The user can also create and specify a different set of preference criteria.

Another knowledge generation operator is based on an extension of the PROMISE program (Baim 1982). It examines the set of attributes in the database and ranks them according to their applicability to a given classification problem. The user can then learn from the subset of the data consisting of only the attributes likely to be more relevant to the learning task and, in doing so, improve the efficiency of the learning process and decrease the likelihood of spurious knowledge being discovered. This operator can be applied in two modes. In the first, each attribute's overall discrimination ability is calculated using an information gain metric similar to that used by the C4.5 (Quinlan 1990) family of algorithms. As such, it is useful in the building of decision trees or other procedural structures. The second mode focuses on the attributes most likely to produce concise rules; it selects attributes that contain some values that discriminate very well between classes, even if most of the other values provide little utility in classifying the examples.

Exemplary Application: Discovery in Economic and Demographic Domains

INLEN and the programs that have been adapted for use as its operators have been applied to learning and discovery problems in such domains as engineering design, disease diagnosis, intelligence gathering and economic analysis (e.g., Arciszewski et al. 1992; Michalski et al. 1992). The following example illustrates the role an intelligent agent can play in the discovery of knowledge from data:

> *The United States government maintains records of the import and export of goods from various countries of the world. The different products and raw materials are divided and subdivided into different categories. In the early 1980s the data showed a sharp decline in the import of trucks from Japan, while there was a corresponding increase in imports from Japan in the auto parts category. It took several years before analysts noticed that fact and concluded that Japan was shipping the chassis and truck beds separately to the US, where they would be subsequently assembled, thereby avoiding a high US tariff on imported trucks that was directed primarily at Europe and had been on the books since World War II. When United States analysts inferred this, the US and Japan commenced trade negotiations pertaining to the import of trucks.*

How much sooner would that trend have been noticed had a discovery program been applied to the data and pointed out to an analyst the opposite changes in two related categories? How much revenue did the undiscovered truth cost the US before they could finally work out a new agreement with Japan? Noticing economic trends and patterns is a difficult task, as humans can easily get overwhelmed by the amount of raw data.

Based on such motivations, the analysis of economic and demographic data has become one of the focus domains for INLEN development and testing. These experiments have involved two similar data sets – one provided by the World Bank consisting of information on 95 attributes in 171 countries for the period of 1965 to 1990, and one extracted from the 1993 World Factbook (CIA 1993) containing several databases of information on 190 countries. Two examples of such experiments and results follow. The first illustrates the passage of knowledge from operator to operator as a concise description of the goal concept is formulated. The second illustrates how the background information in the knowledge base may expose anomalous data.

Distinguishing Between Two Regions' Development Patterns

An experiment that serves as an example of the linkage of different learning and discovery methods focused on distinguishing between development patterns in Eastern Europe and East Asia (Kaufman 1994). A sequence of operators consisting of attribute selection, conceptual clustering, rule generation based on set characterization, and rule optimization combined to generate the output conclusions. After the feature set was pared down to those economic indicators deemed more likely to differentiate the two regions, a conceptual clustering operator based on CLUSTER/2 (Michalski & Stepp 1983) determined that one way of distinguishing between the typical Eastern European country and the typical Far Eastern country was through examining the country's change in the percentage of its population in the labor force between 1980 and 1990. Most of the European countries had a labor force change below a threshold determined for the region by the learning program, while most of the Asian countries had changes above their region's threshold.

Based on this, the rule learning operator, based on the AQ15c inductive learning program (Wnek et al. 1995), was then called upon twice – first in characteristic mode to categorize the commonalities among the Asian-like countries (those above their regional thresholds) and among the European-like countries (those below their regional thresholds), and then in discriminant rule-optimizing mode to condense the lengthy characterizations from the previous set (4-8 conditions per rule) into the following simple decision rules:

Country is Asian-Like if:
A.1 Change in Labor Force Participation ≥ slight_gain, *(9 countries)*
or
B.1 Working Age Population ≤ 64%,
2 Life Expectancy is in 60s. *(2 countries)*

Country is European-Like if:
A.1 Life Expectancy is not in 60s,
2 Change in Labor Force Participation is near 0 or decreasing, *(7 countries)*
or
B.1 Percentage of Labor Force in Industry ≥ 40. *(2 countries)*

The rules show that the features aside from change in labor force participation instrumental in distinguishing between the European-style and Asian-style development patterns include life expectancy, working age population and degree of industrialization. In both the Asian- and European-Like cases, the first rule accounted for most of the countries fitting the class, while the second one described the remainder.

Identification of an Unusual Example

Because INLEN's knowledge base maintains records of the training data that supports the discovered rules, this information can be used to group the records within a particular class. For example, when the AQ15c rule generation operator was called upon to characterize the 13 countries of South America, it came up with two distinct

characterizations – one describing the majority of the countries, and the second describing the other four: Ecuador, French Guyana, Peru and Venezuela. By suggesting that these two subgroups of countries may have significant commononalities among themselves, the program has proposed a classification scheme for further investigation.

Another experiment clearly indicates how INLEN can detect interesting facts within the subgroups it creates. While the subgroups in a demographic domain may indicate that member countries or regions have something in common, notable exceptions may be exposed when one of the members of these constructed subsets shows a marked dissimilarity to the rest of the group. These exceptions in turn may prove be a springboard for additional discovery.

INLEN discovered several rules from the World Factbook PEOPLE database characterizing the countries with low (less than 1% per year) population growth (Kaufman & Michalski 1996a). One of the rules had three conditions that together were sufficient to distinguish 19 low growth countries from all of the countries with higher population growth rates. The rule is shown here with three weights attached to each condition: *Pos* represents the number of positive examples (countries with population growth rates below 1%) satisfying the condition, *Neg* represents the number of negative examples (countries with population growth rates above 1%) satisfying the condition, and *Supp*, defined as Pos / (Pos + Neg) in percent, represents an approximate measure of the degree of support that the condition alone provides for the conclusion that a country might have a population growth rate below 1%.

Conditions characterizing Countries with Population Growth Rates below 1%:	Pos	Neg	Supp
1 Birth Rate = 10 to 20 or over 50	46	20	69
2 Predominant Religion is not defined as Muslim or Mixed or Buddhist or Christian or Tibetan	40	68	37
3 Net Migration Rate ≤ +20	32	104	23

The first and strongest condition states that the country must have a low (under 20 per 1000 population) or very high (over 50) birth rate. The presence of a very high birth rate is extremely counterintuitive; using the links in the knowledge base, one may examine the 19 countries involved. Such an inspection points out that 18 have birth rates below 20, while only one, Malawi, has the high birth rate. INLEN had thereby identified an exception to normal patterns. When further learning was focused on Malawi, a massive outward net migration rate was discovered, by far the most extreme migration rate in the world. Further application of the knowledge discovery operators can then explore the conditions unique to Malawi and hypothesize where else they might take place in the future.

Database Analysis Based on Multistrategy Learning

The previous sections discussed and provided examples of the architecture and major components of INLEN. Through the sequential application of KGOs, a user can link different learning and discovery programs into a stream of tasks. Many of the operators are based on machine learning programs that were not designed for the analysis of large databases, but were instead written based on the assumption that they would be operating in a supervised learning environment. As a result, they often have the following characteristics:

(1) Their inference methods are tailored to simple nominal or linear feature domains without rich domain knowledge.

(2) It is assumed that most of an example's attribute values will be provided.

(3) All of the information relevant to a problem will be available in one location.

Unfortunately, these conditions do not always hold when exploring a real-world database. The attributes in the data may be based on complex hierarchies, lattices and gradations of concepts. Many of the fields in the data may be missing due to incomplete information. And the goal knowledge may be only attainable through the extraction and combination of information from multiple sources.

These problems have been addressed by the knowledge generation operators in INLEN. INLEN supports learning with complex structured data types. Not only can a user define a data attribute to be hierarchically structured, one can also designate nodes within the hierarchy as *anchor nodes* – especially significant foci for learning, generalization and specialization (Kaufman & Michalski 1996b). The justification behind the selection of such nodes is that we tend to weight the significance of nodes in a classification hierarchy unevenly. For instance, a red delicious is an apple, which is a kind of fruit, which is a type of food. In everyday usage, we will not think of a given red delicious at each of those different levels of abstraction with equal frequency.

Cognitive scientists speak of *basic* level nodes within a generalization hierarchy whose children share many sensorially recognizable commonalities (Rosch et al. 1976). Other factors that help to characterize a node's utility compared to those at higher or lower levels of abstraction are concept typicality (how common are the features of this concept among its sibling concepts), and the context in which the concept is being used (Klimesch 1988; Kubat, Bratko & Michalski 1996). Each of these factors affects the selection of a particular level of abstraction in making descriptions.

By encoding the relative utility of the nodes into the knowledge representation of a discovery system, the system can present discovered knowledge that focuses on the more useful levels of abstraction when possible. We will typically prefer to see the classification rules "An

object belongs to Class 1 if it is an apple" or "An object belongs to Class 1 if it is a fruit" instead of "An object belongs to Class 1 if it is a red delicious" or "An object belongs to Class 1 if it is food."

Another important feature in INLEN's structured data representation is its ability to work with multiple views of the data (Kaufman & Michalski, 1996b). Consider an application in which a marketing specialist is trying to target the customers who are most likely to be interested in a new product. A customer database may have extensive information including the model of the automobile driven by the customer. Automobiles may in turn be organized according to manufacturer, type (e.g., sedan, sports car, station wagon), price, year, etc. One may not know prior to a learning task which classification view will provide the most concise and useful knowledge. Also, more than one view may generate the best rules for determining whether a customer is likely to buy the product. A decision rule, for example, may include the condition that likely buyer will often drive European station wagons, while the specification of the type of vehicle or manufacturer alone may not give an accurate representation of the customer's propensities. INLEN's generalization engine automatically selects from the possible ways of expressing a set of attribute values (in this case the automobile models that are European station wagons) a concise representation of the knowledge.

INLEN's knowledge generation operators have also been enhanced to cope with the problem of incomplete data sets. Experiments with sparse data exposed some of the limitations of these operators, leading to modifications in which the logical implications of unknowns are more rigorously encoded into the learning algorithm. For example, in the AQ family of programs, attributes with unknown values are represented as having all of their values under consideration. A stipulation in earlier versions of the program required that generalizations of examples with unknown values for some attributes maintain consistency by permitting those attributes to take any value. While this will guarantee rigorous consistency with the data, trouble arises when many examples have only a few known attribute values. Generalizing just a few of them together creates a situation in which nothing can be assumed about any feature.

As an example of a domain in which this may be a real problem, intelligent agents are being developed to scan text and summarize it based on key words in several categories of interest to the user. Articles often will only contain key words in a few of these categories, leading to a very empty database. The relationships among entries in the various categories will often be tenuous ones given that for much of the data, one or more of these fields will be empty. In such a domain a discovery program must be able to sift through the information that is present without getting lost in that which is missing. At the expense of some additional computational complexity, the knowledge generation operators in INLEN have been modified in such a way that they can now generate knowledge consistent with what facts have been made available, without adhering to the assumption that unknowns must be generalized to take on all values. With the relaxation of this condition, a learning engine can detect more substantial relationships.

Another aspect of this research approaches the problem of knowledge extraction from distributed sources. The INSIGHT program (Ribeiro, Kaufman & Kerschberg. 1995) is being developed as an operator to perform a knowledge-driven search through multiple databases. The combinatorial cost of combining separate data sets is avoided through INSIGHT's mechanism of finding relationships between a database and the knowledge generated from another database.

In order to facilitate the interface with INSIGHT and other operators, INLEN maintains information on the database records relevant to each rule in its knowledge base. As was shown above in the population growth example, a use for these links is to enable the identification of significant clusters or exceptions. Another use is to allow the measurement of the degree of match between a rule and a set of records in a second database matching a given set of conditions. A high degree of match may suggest a linkage between the two concepts. For example, a rule describing the climate of a country may be cross-referenced with a database of natural disasters. If a class of natural disaster occurred in a set of countries similar to the set of countries covered by the climate rule, it may suggest a relationship between the climate and that kind of disaster.

Conclusion

The INLEN methodology is based on the application of a wide variety of machine learning and inference programs for the purpose of discovering knowledge from databases and providing concise conceptual explanations of their findings. These diverse machine learning and inference tools can work in conjunction with traditional statistical tools. Among the major advantages of this methodology is its emphasis on providing conceptually understandable results of data analysis due to the logic-style descriptions it generates. Another advantage is its modularity that makes it easy to add new knowledge generation operators.

The examples shown above involved just a few of the knowledge generation operators in INLEN. The system is growing steadily as operators are enhanced and added to its environment; while still in prototype form, it already has many functional operators.

This report describes a work in progress in which new capabilities have been added over the course of its development. Among the limitations of the current implementation are the facts that it still awaits integration with statistical data analysis methods and that there is still too much reliance on the data analyst for guidance in the selection of operators and the setting of parameters. In particular, the methodology is strongly human-driven, and an area of ongoing research is seeking to develop a reliable means to automate the discovery process. Another limitation is portability; while future versions of INLEN

will run on other platforms, it is currently limited to PC-based systems.

As described above, topics of current research include the development of methods for automated data abstraction and the analysis of distributed databases. Future projects include the addition of tools for creating new attributes for improved performance (constructive induction) based on the AQ17-MCI methodology (Bloedorn et al. 1993). This methodology combines data-driven construction of attributes based on the detection of relationships between attributes, hypothesis-driven construction of attributes based on patterns detected in preliminary rulesets, and statistically-based operators for quantization of continuous numerical attributes and summarization of notable groups of examples (e.g., a region's average per capita income over a 10 year period).

Another enhancement will be the development and incorporation of a high-level language for knowledge discovery. Such a tool will attack one of the limitations of the system mentioned above, namely that the knowledge discovery process must be closely supervised at present. With the addition of a knowledge discovery language, a user will be able to program in sequences of operators, along with instructions or heuristics detailing what conditions should cause them to be invoked. The system will be able to follow such instructions as "If a new month's data shows less than a 95% consistency with the knowledge base, update the knowledge to incorporate the new data and then use characterization operators to seek out possible explanations why the behavior has changed in the new month" or "If a high-urgency network fault is detected, access the knowledge base to predict the location of the problem and the nature of a likely solution and report it immediately."

In the presented examples, INLEN's knowledge generation operators were applied to databases in various economic and demographic domains. The system's searches have unearthed some surprising facts. The experiments described here illustrate some of the potential capabilities of the application of integrated learning strategies to large databases and indicate that such an application has the potential for determining important but heretofore unknown findings within the data.

Whether this or any other multistrategy architecture is used for data analysis, the task of database exploration presents certain problems not often faced by the learning components. When learning tools are turned toward such knowledge discovery problems, they should be equipped to represent rich domain knowledge, handle very sparse data sets, and be prepared to integrate data from different sources. This paper has described approaches to each of these problems.

The goal of all of these features is to facilitate INLEN's compatibility with real-world databases. The tools that serve as knowledge generation operators can then perform at a high level on real-world problems as well as on carefully supervised data sets, while their integration results in a multistrategy system not limited to a narrow class of discovery tasks.

Acknowledgments

The author thanks Ibrahim Imam and Eric Bloedorn for their constructive suggestions regarding earlier versions of this paper.

This research was conducted in the Machine Learning and Inference Laboratory at George Mason University. The Laboratory's research is supported in part by the Defense Advanced Research Projects Agency under Grant No. N00014-91-J-1854 administered by the Office of Naval Research, in part by the Defense Advanced Research Projects Agency under Grants No. F49620-92-J-0549 and F49620-95-1-0462 administered by the Air Force Office of Scientific Research, in part by the Office of Naval Research under Grant No. N00014-91-J-1351, and in part by the National Science Foundation under Grants No. DMI-9496192 and IRI-9020266.

References

Alexander, W.P., Bonissone, P.P. and Rau, L.F. 1993. Preliminary Investigations into Knowledge Discovery for Quick Market Intelligence. In Proceedings of AAAI-93 Workshop on Knowledge Discovery in Databases, 52-60. Washington, DC.

Arciszewski, T., Bloedorn, E., Michalski, R.S., Mustafa, M. and Wnek, J. 1992. Constructive Induction in Structural Design. *Reports of the Machine Learning and Inference Laboratory*, MLI-92-7, George Mason University, Fairfax, VA.

Baim, P.W. 1982. The PROMISE Method for Selecting Most Relevant Attributes for Inductive Learning Systems. Report No. UIUCDCS-F-82-898, Department of Computer Science, University of Illinois, Urbana, IL.

Bloedorn, E., Wnek, J., and Michalski, R.S. 1993. Multistrategy Constructive Induction: AQ17-MCI. In Proceedings of the Second International Workshop on Multistrategy Learning, 188-203, Harpers Ferry, WV..

Central Intelligence Agency 1993. *1993 World Factbook.*

Kaufman, K. 1994. Comparing International Development Patterns Using Multi-Operator Learning and Discovery Tools. In Proceedings of AAAI-94 Workshop on Knowledge Discovery in Databases, 431-440 Seattle, WA.

Kaufman, K. and Michalski, R.S. 1996a. A Multistrategy Conceptual Analysis of Economic Data. In Proceedings of the Fourth International Workshop on Artificial Intelligence in Economics and Management, Tel Aviv.

Kaufman, K. and Michalski, R.S. 1996b. A Method for Reasoning With Structured and Continuous Attributes in the INLEN-2 Knowledge Discovery System. *Second International Conference on Knowledge Discovery and Data Mining*, Portland, OR (to appear).

Kaufman, K. Michalski, R.S., and Kerschberg, L. Mining for Knowledge in Databases: Goals and General Description of the INLEN System. Chapter in Piatetsky-Shapiro, G. and Frawley, W.J. (eds.) *Knowledge Discovery in Databases,* Cambridge, MA: AAAI Press, 449-462

Klimesch, W. 1988. *Struktur und Aktivierung des Gedaechtnisses. Das Vernetzungsmodell: Grundlagen und Elemente einer uebergreifenden Theorie.* Bern: Verlag Hans Huber.

Kubat, M., Bratko, I. and Michalski, R.S. 1996. A Review of Machine Learning Techniques. Chapter in *Methods and Applications of Machine Learning and Discovery.* Forthcoming

Michalski, R.S. 1993. Inferential Theory of Learning as a Conceptual Basis for Machine Learning. *Machine Learning* 11: 111-151.

Michalski, R.S. and Baskin, A.B. 1983. Integrating Multiple Knowledge Representations and Learning Capabilities in an Expert System: The ADVISE System. In Proceedings of the 8th International Joint Conference on Artificial Intelligence, 256-258, Karlsruhe, West Germany.

Michalski, R.S., Baskin, A.B. and Spackman, K.A. 1982. A Logic-based Approach to Conceptual Database Analysis. In Proceedings of the Sixth Annual Symposium on Computer Applications in Medical Care (SCAMC-6), 792-796, George Washington University Medical Center, Washington, DC.

Michalski, R.S., Kerschberg, L., Kaufman, K. and Ribeiro, J. 1992. Mining for Knowledge in Databases: The INLEN Architecture, Initial Implementation and First Results. *Journal of Intelligent Information Systems: Integrating AI and Database Technologies* 1:85-113.

Michalski, R.S., and Stepp, R.E. 1983. Automated Construction of Classifications: Conceptual Clustering Versus Numerical Taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 5:396-410.

Michalski, R.S. and Tecuci, G. eds. 1991. *Proceedings of the First International Workshop on Multistrategy Learning.* Harpers Ferry, WV.

Michalski, R.S. and Tecuci, G. eds. 1993. *Proceedings of the Second International Workshop on Multistrategy Learning.* Harpers Ferry, WV.

Quinlan, J.R. 1990. Probabilistic Decision Trees. In *Machine Learning: An Artificial Intelligence Approach, Volume III,* Kodratoff, Y. and Michalski, R.S. eds., Morgan Kaufmann, San Mateo, CA.

Ribeiro, J.S., Kaufman, K.A. and Kerschberg, L. 1995. Knowledge Discovery From Multiple Databases. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining, 240-245, Montreal PQ.

Rosch, E., Mervis, C., Gray, W., Johnson, D. and Boyes-Braem, P. 1976. Basic Objects in Natural Categories. *Cognitive Psychology,* 8: 382-439.

Simoudis, E., Livezey, B., and Kerber, R. 1994. Integrating Inductive and Deductive Reasoning for Database Mining. In Proceedings of AAAI-94 Workshop on Knowledge Discovery in Databases, 37-48, Seattle, WA.

Spackman, K.A. 1983. QUIN: Integration of Inferential Operators within a Relational Database. ISG 83-13, UIUCDCS-F-83-917, M.S. Thesis, Department of Computer Science, University of Illinois, Urbana, IL.

Strat, T.M. 1992. *Natural Object Recognition.* New York: Springer Verlag.

Wnek, J., Kaufman, K., Bloedorn, E. and Michalski, R.S. 1995. Selective Induction Learning System AQ15c: The Method and User's Guide. *Reports of the Machine Learning and Inference Laboratory,* MLI 95-4, George Mason University, Fairfax, VA.

Automated Extraction of Expert System Rules from Databases based on Rough Set Theory

Shusaku Tsumoto and Hiroshi Tanaka
Department of Information Medicine
Medical Research Institute, Tokyo Medical and Dental University
1-5-45 Yushima, Bunkyo-ku Tokyo 113 Japan
TEL: +81-3-5803-5840 FAX: +81-3-5803-0247
E-mail: tsumoto.com@tmd.ac.jp, tanaka@cim.tmd.ac.jp

Abstract

Automated knowledge acquisition is an important research area to solve the bottleneck problem in developing expert systems. For this purpose, several methods of inductive learning, such as induction of decision trees, AQ method, and neural networks, have been introduced. However, most of the approaches focus on inducing rules which classify cases correctly. On the contrary, medical experts also learn other information which is important for medical diagnostic procedures from databases. In this paper, a rule-induction system, called PRIMEROSE3(Probabilistic Rule Induction Method based on Rough Sets version 3.0), is introduced. This program first analyzes the statistical characteristics of attribute-value pairs from training samples, then determines what kind of diagnosing model can be applied to the training samples. Then, it extracts not only classification rules for differential diagnosis, but also other medical knowledge needed for other diagnostic procedures in a selected diagnosing model. PRIMEROSE3 is evaluated on three kinds of clinical databases and the induced results are compared with domain knowledge acquired from medical experts, including classification rules. The experimental results show that our proposed method correctly not only selects a diagnosing model, but also extracts domain knowledge.

Keywords: learning and data mining, rule induction, rough sets

1. Introduction

One of the most important problems in developing expert systems is knowledge acquisition from experts (Buchanan and Shortliffe, 1984). While there have been developed many knowledge acquisition tools to simplify this process, it is still difficult to automate this process. In order to solve this problem, many inductive learning methods, such as induction of decision trees (Breiman, 1984; Cestnik, et al., 1987; Quinlan, 1986), rule induction methods (Clark and Niblett, 1989; Indurkhya and Weiss, 1991; Michalski, 1983; Michalski, et al. 1986) and rough set theory (Pawlak, 1991; Ziarko, 1991), are introduced in order to discover knowledge from large databases.

However, most of the approaches focus on inducing classification rules, which classifies cases correctly. On the contrary, medical experts also learn other kinds of knowledge, which are important for medical diagnostic procedures, from clinical cases.

In this paper, a rule induction system, called PRIMEROSE3(Probabilistic Rule Induction Method based on Rough Sets version 3.0), is introduced. This program first analyzes the statistical characteristics of attribute-value pairs from training samples, then determines what kind of diagnosing model can be applied to the training samples. Then, it extracts not only classification rules for differential diagnosis, but also other medical knowledge needed for other diagnostic procedures in a selected diagnosing model. PRIMEROSE3 is evaluated on three kinds of clinical databases and the induced results are compared with domain knowledge acquired from medical experts, including classification rules. The experimental results show that our proposed method correctly not only selects a diagnosing model, but also extracts domain knowledge.

The paper is organized as follows: Section 2 discusses rough set theory. Section 3 illustrates three diagnosing models. Section 4 presents our new method, PRIMEROSE3 and Section 5 gives experimental results. Section 6 and Section 7 discuss the problems of PRIMEROSE3 and related work, respectively. Finally, Section 8 concludes this paper.

2. Rough Sets and Rules

2.1 Rough Set Theory

Rough set theory(Pawlak, 1991) clarifies set-theoretic characteristics of the classes over combinatorial patterns of the attributes in order to acquire some sets of attributes for classification and to evaluate how precisely attributes in a database can classify data.

Let us illustrate the main concepts of rough sets which are needed for our formulation. Table 2 is

Table 1: A Small Database

	age	loc	nat	prod	nau	M1	class
1	50-59	occ	per	0	0	1	m.c.h.
2	40-49	who	per	0	0	1	m.c.h.
3	40-49	lat	thr	1	1	0	migra
4	40-40	who	thr	1	1	0	migra
5	40-49	who	rad	0	0	1	m.c.h.
6	50-59	who	per	0	1	1	m.c.h.

DEFINITIONS: loc: location, nat: nature, prod: prodrome, nau: nausea, M1: tenderness of M1, who: whole, occ: occular, lat: lateral, per: persistent, thr: throbbing, rad: radiating, m.c.h.: muscle contraction headache, migra: migraine, 1: Yes, 0: No.

a small database whose patients chiefly complain of headache. First, let us consider how an attribute "loc" classify the headache patients' set of the table. The set whose value of the attribute "loc" is equal to "who"is {2,4,5,6}(In the following, the numbers represent each record number). This set means that we cannot classify {2,4,5,6} further solely by using the constraint $R = [loc = who]$. This set is defined as the indiscernible set over the relation R, denoted by $[x]_R = \{2, 4, 5, 6\}$. In this set, {2,5,6} suffer from muscle contraction headache("m.c.h."), and {4} suffers from migraine("migra"). Hence we need other additional attributes to discriminate between "migra" and "m.c.h." Using this concept, we can evaluate the classification power of each attribute. For example, "prod=1" is specific to the case of migraine ("migra"). We can also extend this indiscernible relation to multivariate cases, such as $[x]_{[loc=who] \wedge [M1=1]} = \{2, 5, 6\}$ and $[x]_{[loc=who] \vee [M1=1]} = \{1, 2, 5, 6\}$, where $\wedge$ and $\vee$ denote "and" and "or" respectively. In the framework of rough set theory, the set {2,5,6} is called *strictly definable* by the former conjunction, and also called *roughly definable* by the latter disjunctive formula. Therefore, the classification of training samples D can be viewed as a search procedure for the best set $[x]_R$ supported by the relation R. In this way, we can define the characteristics of classification in the set-theoretic framework. For example, accuracy (SI) and coverage, or true positive rate (CI) can be defined as:

$$\alpha_R(D) = \frac{|[x]_R \cap D|}{|[x]_R|}, \text{ and } \kappa_R(D) = \frac{|[x]_R \cap D|}{|D|},$$

where $|D|$ denotes the cardinality of D and where $\alpha_R(D)$ denotes an accuracy of R with respect to classification of D, $SI(R, D)$ and $\kappa_R(D)$ denotes a true positive rate of R to D, $CI(R, D)$, respectively.

For example, let R be equal to $[loc = who]$ and D be equal to a set of "m.c.h." Then, $[x]_R$ and D is equal to $\{2, 4, 5, 6\}$ and $\{1, 2, 5, 6\}$, respectively, and the intersection of both sets, $[x]_R \cap D$, is $\{2, 4, 5, 6\} \cap \{1, 2, 5, 6\} = \{2, 5, 6\}$. Thus, $\alpha_R(D)$ is equal to $|\{2, 5, 6\}|/|\{2, 4, 5, 6\}| = 3/4 = 0.75$ and $\kappa_R(D)$ is obtained as: $|\{2, 5, 6\}|/|\{1, 2, 5, 6\}| = 3/4 = 0.75$, respectively.

It is notable that $\alpha_R(D)$ measures the degree of the sufficiency of a proposition, $R \rightarrow D$, and that $\kappa_R(D)$ measures the degree of its necessity. For example, if $\alpha_R(D)$ is equal to 1.0, then $R \rightarrow D$ is true. On the other hand, if $\kappa_R(D)$ is equal to 1.0, then $D \rightarrow R$ is true. Thus, if both measures are 1.0, then $R \leftrightarrow D$.

For further information on rough set theory, readers could refer to (Pawlak, 1991; Ziarko, 1991; Ziarko, 1993).

2.2 Probabilistic Rules

In order to describe diagnosing rules, we first define probabilistic rules, using notations of rough set theory(Pawlak, 1991). To illustrate the main ideas, we use a small database shown in Table 1.

First, a combination of attribute-value pairs, corresponding to a complex in AQ terminology(Michalski, 1983), is denoted by an equivalence relation R_f, which is defined as follows.

Definition 1 (Equivalence Relation) *Let U be a universe, and V be a set of values. A total function f from U to V is called an assignment function of an attribute. Then, we introduce an equivalence relation R_f such that for any $u, v \in U$, $u \equiv R_f v$ iff $f(u) = f(v)$.*

For example, $[age = 50 - 59] \& [loc = occular]$ will be one equivalence relation, denoted by $R_f = [age = 50 - 59] \& [loc = occular]$. Secondly, a set of samples which satisfy R_f is denoted by $[x]_{R_f}$, corresponding to a star in AQ terminology. For example, when $\{2, 3, 4, 5\}$ is a set of samples which satisfy $[age = 40 - 49]$, $[x]_{[age=40-49]}$ is equal to $\{2, 3, 4, 5\}$. [1] Finally, thirdly, U, which stands for "Universe", denotes the whole training samples.

According to this notation, probabilistic rules are defined as follows:

Definition 2 (Probabilistic Rules) *Let R_f be an equivalence relation specified by some assignment function f, D denote a set whose elements belong to a class d, or positive examples in the whole training samples (the universe), U. Finally, let $|D|$ denote the cardinality of D. A probabilistic rule of D is defined as a quadruple, $< R_f \stackrel{\alpha,\kappa}{\rightarrow} d, \alpha_{R_f}(D), \kappa_{R_f}(D) >$, where*

[1] In this notation, "n" denotes the nth sample in a dataset (Table 1).

$R_f \stackrel{\alpha,\kappa}{\rightarrow} d$ *satisfies the following conditions:*[2]

$$(1) \qquad [x]_{R_f} \bigcap D \neq \phi,$$

$$(2) \quad \alpha_{R_f}(D) = \frac{|[x]_{R_f} \bigcap D)|}{|[x]_{R_f}|},$$

$$(3) \quad \kappa_{R_f}(D) = \frac{|[x]_{R_f} \bigcap D)|}{|D|}.$$

In the above definition, α corresponds to the accuracy measure: if α of a rule is equal to 0.9, then the accuracy is also equal to 0.9. On the other hand, κ is a statistical measure of how proportion of D is covered by this rule, that is, a coverage or a true positive rate: when κ is equal to 0.5, half of the members of a class belongs to the set whose members satisfy that equivalence relation.

For example, let us consider a rule $[age = 40-49] \rightarrow$ *m.c.h.* Since $[x]_{[age=40-49]} = \{2,3,4,5\}$ and $D = \{1,2,5,6\}$, $\alpha_{[age=40-49]}(D) = |\{2,5\}|/|\{2,3,4,5\}| = 0.5$ and $\kappa_{[age=40-49]}(D) = |\{2,5\}|/|\{1,2,5,6\}| = 0.5$. Thus, if a patient, who complains a headache, is 40 to 49 years old, m.c.h. is suspected with accuracy 0.5, and this rule covers 50 % of the cases.

3. Diagnosing Model

3.1 Simplest Diagnosing Model

The simplest diagnosing model is that which only uses classification rules which have high accuracy and high coverage.[3] This model is applicable when rules of high accuracy can be derived. Such rules can be defined as:

$$R \stackrel{\alpha,\kappa}{\rightarrow} d \quad s.t. \qquad R = \vee_i R_i = \vee \wedge_j [a_j = v_k],$$
$$\alpha_{R_i}(D) > \delta_\alpha, \quad and \quad \kappa_{R_i}(D) > \delta_\kappa,$$

where δ_α and δ_κ denote given thresholds for accuracy and coverage, respectively. It is notable that this rule is a kind of probabilistic proposition with two statistical measures, which is one extension of Ziarko's variable precision model(VPRS) (Ziarko, 1993). [4] It is also notable that this model only uses inclusive rules of a RHINOS diagnosing model which is also defined below.

[2]It is notable that this rule is a kind of probabilistic proposition with two statistical measures, which is one kind of an extension of Ziarko's variable precision model(VPRS) (Ziarko, 1993).

[3]In this model, we assume that accuracy is dominant over coverage

[4]In VPRS model, the two precisions of accuracy are given, and the probabilistic proposition with accuracy and two precision conserves the characteristics of the ordinary proposition. Thus, our model is to introduce the probabilistic proposition not only with accuracy, but also with coverage.

3.2 Weak Diagnosing Model

In some probabilistic domains, it is difficult to derive classification rules which have high accuracy. In these cases, the way to overcome this problem is to induce rules which have high coverage. Such rules can be defined as:

$$d \stackrel{\kappa,\alpha}{\rightarrow} R \quad s.t. \qquad R = \vee_i R_i = \vee \wedge_j [a_j = v_k],$$
$$\kappa_{R_i}(D) > \delta_\kappa, \quad and \quad \alpha_{R_i}(D) > \delta_\alpha.$$

Thus, in this model, it is assumed that coverage is dominant over accuracy and this model only induces the necessity condition of diagnosis of class *d*.

In the subsequent subsection, we introduce the combination of the above two diagnosing models.

3.3 RHINOS Diagnosing Model

RHINOS is an expert system which diagnoses the causes of headache or facial pain from manifestations (Kimura, et al., 1985; Matsumura, et al., 1986). In this system, a diagnosing model proposed by Matsumura is applied, which is composed of the following three kinds of reasoning processes: exclusive reasoning, inclusive reasoning, and reasoning about complications.

Firstly, exclusive reasoning is the one that when a patient does not have a symptom which always appears in any case on a disease, such a disease can be excluded. Secondly, inclusive reasoning is the one that when a patient has symptoms specific to a disease, the disease can be suspected. Finally, thirdly, reasoning about complications is that when some symptoms which cannot be explained by that disease, complications of other diseases can be suspected.

Using the above diagnosing model, we consider three kinds of rules corresponding to each process, which can be described in terms of rough set theory as follows.

(1) Exclusive Rule

$$R \stackrel{\alpha,\kappa}{\rightarrow} d \quad s.t. \qquad R = \wedge_i R_i = \wedge \vee_j [a_j = v_k],$$
$$and \ \kappa_{[a_j=v_k]}(D) = 1.0.$$

Strictly Speaking, this proposition should be written as: $d \rightarrow R$. However, for comparison with other two rules, we choose this notation. In the above example, the relation R of the exclusive rule for "classic" is described as:
$[age = 40-49] \wedge ([loc = lat] \vee [loc = who]) \wedge [nat = thr] \wedge [jolt = 1] \wedge [prod = 1] \wedge [nau = 1] \wedge [M1 = 0]$.

(2) Inclusive Rule

$$R \overset{\alpha,\kappa}{\rightarrow} d \quad s.t. \qquad R = \vee_i R_i = \vee_i \wedge_j [a_j = v_k],$$
$$\alpha_{R_i}(D) > \delta_\alpha, \ and \ \kappa_{R_i}(D) > \delta_\kappa.$$

In the above example, the simplest relation R of the inclusive rule for "classic", is described as: $[nat = thr] \vee [jolt = 1] \vee [M1 = 1]$. However, induction of inclusive rules gives us two problems. First, accuracy and coverage are overfitted to the training samples. Secondly, the above rule is only one of many rules which are induced from the above training samples. Therefore some of them should be selected from primary induced rules under some preference criterion. These problems will be discussed in the next section.

(3) Disease Image:

$$R \overset{\alpha,\kappa}{\rightarrow} d \quad s.t. \qquad R = \vee R_i = \vee [a_i = v_j],$$
$$\alpha_{R_i}(D) > 0 \ (\kappa_{R_i}(D) > 0).$$

In the above example, the relation R of the disease image for "classic" is described as:
$[age = 40 - 49] \vee [loc = lat] \vee [loc = who] \vee [nat = thr] \vee [nau = 1] \vee [jolt = 1] \vee [M1 = 0]$.

It is notable that coverage κ play an important role in the definition of these rules, compared with simplest diagnosing model.

4. PRIMEROSE3

In this section, we introduce a rule-induction system, called PRIMEROSE3(Probabilistic Rule Induction Method based on Rough Sets version 3.0). This program first analyzes the statistical characteristics of attribute-value pairs from training samples, then determines what kind of diagnosing model can be applied to these training samples. Then, it extracts not only classification rules for differential diagnosis, but also other medical knowledge needed for other diagnostic procedures, based on a selected diagnosing model.

4.1 Selection of Diagnosing Model

As discussed in Section 3, coverage plays an important role in selection of a diagnosing model. Thus, PRIMEROSE3 first measures the statistical characteristics of coverage of elementary attribute-value pairs, which corresponds to selectors. Then, it measures the statistical characteristics of accuracy of the whole pattern of attribute-value pairs observed in a dataset.

In this algorithm, we use the following characteristic of coverage.

Table 2: Frequency Table of Coverage

κ	0.00	0.25	0.50	0.75	1.00
m.c.h.	5	6	5	4	2
classic	10	10	6	0	6

Proposition 1 (Monotonicity of Coverage) *Let R_{i+1} denote an attribute-value pair, $R_i \wedge [a_{i+1} = v_j]$. Then,*

$$\kappa_{R_{i+1}}(D) \leq \kappa_{R_i}(D).$$

Proof. Since $[x]_{R_{i+1}} \subseteq [x]_{R_i}$ holds, $\kappa_{R_{i+1}}(D) = \frac{|[x]_{R_{i+1}} \cap D|}{|D|} \leq \frac{|[x]_{R_i} \cap D|}{|D|} = \kappa_{R_i}(D)$. □

Furthermore, in rule induction methods, R_{i+1} is selected to satisfy $\alpha_{R_{i+1}}(D) > \alpha_{R_i}(D)$. Therefore, it is sufficient to check the behavior of coverage of elementary attribute-value pairs in order to estimate which diagnosing model should be selected, while it is necessary to check the behavior of accuracy both of elementary attribute-value pairs and of patterns observed in the databases in order to estimate the characteristics of induced rules. From these considerations, the selection algorithm is defined as follows.

(1) Calculate a coverage and an accuracy of each attribute value pair $[a_i = v_j]$.

(2) Calculate an accuracy of each pattern $\wedge_i [a_i = v_j]$.

(3) Construct a frequency table with respect to coverage and accuracy for each class.

(4) If each class has at least one attribute value pair whose coverage is equal to 1.0 and if more than half of coverage values is larger than δ_κ, goto next. Else, select simplest diagnosing model.

(5) If the median of accuracy of each is larger than δ_α, then select RHINOS diagnosing model and quit. Else select weak diagnosing model and quit.

For the above example shown in Table 1, frequency tables of coverage and accuracy is obtained as Table 2 and Table 3. Thus, let us consider a case when δ_κ is set to 0.5, and δ_α is set to 0.75. Candidates of diagnosing model will be weak diagnosing model and RHINOS diagnosing model from Table 2. Next, from Table 3, RHINOS diagnosing model will be selected, since the median of accuracy of elementary pairs is exactly 1.0.

Table 3: Frequency Table of Accuracy

Elementary Attribute Value Pairs							
α	0.00	0.25	0.33	0.50	0.67	0.75	1.00
m.c.h.	5	0	2	1	4	1	9
classic	9	1	4	1	2	0	5
Patterns Observed in a Dataset							
α	0.00	0.25	0.33	0.50	0.67	0.75	1.00
m.c.h.	0	0	0	0	0	0	4
classic	0	0	0	0	0	0	2

4.2 Algorithm for Rule Induction

For the limitation of space, we only discuss an algorithm for induction of RHINOS rules. However, since the other two models can be viewed as specific forms of RHINOS model, it is easy to derive each algorithm from the induction algorithm shown below.

An induction algorithm for RHINOS rules consists of two procedures. One is an exhaustive search through all the attribute-value pairs (*selectors* in the AQ terminology (Michalski, 1983)), and the other is a heuristic search for inclusive rules through the combinations of all the attribute-value pairs (*complexes* in the AQ terminology).

Exhaustive Search Let D and δ denote training samples of the target class d (*positive examples*) and a threshold to select attributes for inclusive rules. Then, this search procedure is defined as follows.

```
procedure Exhaustive Search;
  var
    L : List; /* A list of elementary relations */
  begin
    L := P0; /* P0: A list of elementary relations */
    while (L ≠ {}) do
      begin
        Select one pair [a_i = v_j] from L;
        if ([x]_[a_i=v_j] ∩ D ≠ φ) then do
          /* D: a set if positive examples */
          begin
            R_di := R_di ∨ [a_i = v_j];
              /* Disease Image */
            if (κ_[a_i=v_j](D) > δ_κ)
              then L_ir := L_ir + {[a_i = v_j]};
              /* Candidates for Inclusive Rules */
            if (κ_[a_i=v_j](D) = 1.0)
              then R_er := R_er ∧ [a_i = v_j];
              /* Exclusive Rule */
          end
        L := L − {[a_i = v_j]};
      end
  end {Exhaustive Search};
```

The above procedure is repeated for all the attribute-value pairs, and computes exclusive rules, disease images, and candidates of inclusive rules. These candidates are input into the heuristic search procedure, discussed in the next subsection.

In the above example in Table 1, let d be "classic", and $[age = 40 - 49]$ be selected as $[a_i = v_j]$. Since $[x]_{[age=40-49]} \cap D(= \{3,4\}) \neq \phi$, this pair is included in the disease image. However, when δ is set to 0.5, this pair is not included in the inclusive rule, because $\alpha_{[age=40-49]}(D) = 0.5$. Finally, since $D \subset [x]_{[age=40-49]}(= \{2,3,4,5\})$, this pair is also included in the exclusive rule.

Next, $[age = 50 - 59]$ is selected. However, this pair will be abandoned since the intersection of $[x]_{[age=50-59]}$ and D is empty, or $[x]_{[age=50-59]} \cap D = \phi$.

Heuristic Search Since the definition of inclusive rules is a little weak, many inclusive rules can be obtained. For the above example, a relation $[nau = 1]$ satisfies $D \cap [x]_{[nau=1]} \neq \phi$, so it is also one of the inclusive rules of "m.c.h.", although accuracy of that rule is equal to 1/3. In order to suppress induction of such rules, which have low classificatory power, only equivalence relations whose accuracy is larger than δ_α is selected. For example, when δ is set to 1/2(=0.5), the above relation $[age = 40 - 49]$ is eliminated from the candidates of inclusive rules, because accuracy of this relation is less than the precision, Furthermore, PRIMEROSE3 minimizes the number of attributes not to include the attributes which do not gain the classificatory power, called *dependent* variables. This procedure can be described as follows:

```
procedure Heuristic Search;
  var
    i : integer;   M, L_i : List;
  begin
    L_1 := L_ir; /* Candidates for Inclusive Rules */
    i := 1;   M := {};
    for i := 1 to n do
        /* n: Total number of attributes */
      begin
        while ( L_i ≠ {} ) do
          begin
            Select one pair R = ∧[a_i = v_j] from L_i;
            L_i := L_i − {R};
            if   (α_R(D) > δ_α)   and   (κ_R(D) > δ_κ)
              then do S_ir := S_ir + {R};
              /* Include R as Inclusive Rule */
            else M := M + {R};
          end
        L_i+1 := (A list of the whole combination of
                  the conjunction formulae in M);
      end
  end {Heuristic Search};
```

In the above example in Table 1, the coverage of $[M1 = 1]$ for "m.c.h" is maximum. Furthermore, since $\alpha_{[M1=1]}(D) = 1.0$, it is included in inclusive rules of "m.c.h". The next maximum one is $[nau = 0]$, whose coverage is equal to 3/4. Since this accuracy is also equal to 1.0, it is also included in inclusive rules. At this point, we have two inclusive rules as follows:

$[M1 = 1] \overset{\alpha=1.0,\kappa=1.0}{\rightarrow}$ "*m.c.h.*" and $[nau = 0] \overset{\alpha=1.0,\kappa=0.75}{\rightarrow}$ "*m.c.h.*" Repeating these procedures, all the inclusive rules are acquired.

4.3 Estimation of Accuracy and Coverage

The above definition of accuracy and coverage shows that small training samples cause the overestimates of accuracy and coverage. In the above example shown in Table 1, both of accuracy and coverage of the simplest rule for "classic" are equal to 1.0. This means that this rule correctly diagnoses and covers all the cases of the disease "classic". However, in general, these meanings hold only in the world of the small training samples. In this sense, accuracy and coverage are biased. Thus, these biases should be corrected by introducing other estimating methods, since the biases cannot be detected by the induced method.

Note that this problem is similar to that of error rates of discriminant function in multivariate analysis (Mclachlan, 1992), the field in which resampling methods are reported to be useful for the estimation.

Hence the resampling methods are applied to estimation of accuracy and coverage, as shown in the following subsection.

4.4 Cross-Validation and the Bootstrap method

Cross-validation method for error estimation is performed as following: first, the who training samples $\mathcal{L}$ are split into V blocks: $\{\mathcal{L}_1, \mathcal{L}_2, \cdots, \mathcal{L}_V\}$. Secondly, repeat for V times the procedure in which rules are induced from the training samples $\mathcal{L} - \mathcal{L}_i (i = 1, \cdots, V)$ and examine the error rate err_i of the rules using $\mathcal{L}_i$ as test samples. Finally, the whole error rate err is derived by averaging err_i over i, that is, $err = \sum_{i=1}^{V} err_i/V$ (this method is called V-fold cross-validation). Therefore this method for estimation of accuracy or coverage can be used by replacing the calculation of err by that of accuracy or coverage, respectively, and by regarding test samples as unobserved cases.

On the other hand, the Bootstrap method is executed as follows: first, empirical probabilistic distribution(F_n) is generated from the original training samples (Efron, 1982). Secondly, the Monte-Carlo method is applied and training samples are randomly taken by using F_n. Thirdly, rules are induced by using new training samples. Finally, these results are tested by the original training samples and then statistical measures, such as error rate are calculated. These four steps are iterated for finite times. Empirically, it is shown that about 200 times repetition is sufficient for estimation (Efron, 1982).

Table 4: Information about Databases

Domain	Samples	Classes	Attributes
headache	1477	10	20
meningitis	198	3	25
CVD	261	6	27

Interestingly, Efron shows that estimators by 2-fold cross-validation are asymptotically equal to predictive estimators for completely new pattern of data, and that Bootstrap estimators are asymptotically equal to maximum likelihood estimators and are a little overfitted to training samples (Efron, 1983). Hence, the former estimators can be used as the lower bound of accuracy and coverage, and the latter as the upper bound of accuracy and coverage.

Furthermore, in order to reduce the high variance of estimators by cross-validation, we introduce repeated cross-validation method, which is first introduced by Walker (Walker and Olshen, 1992). In this method, cross-validation methods are executed repeatedly(safely, 100 times), and estimates are averaged over all the trials. In summary, since our strategy is to avoid the overestimation and the high variabilities, combination of repeated 2-fold cross-validation and the Bootstrap method is adopted in this paper.

5. Experimental Results

PRIMEROSE3 is applied to headache(RHINOS's domain), meningitis, and cerebrovascular diseases, whose precise information is given in Table 4. The experiments are performed by the following three procedures. First, these samples are randomly split into pseudo-training samples and pseudo-test samples. Secondly, by using the pseudo-training samples, PRIMEROSE3 selects a model, induces rules and the statistical measures.[5] Thirdly, the induced results are tested by the pseudo-test samples. These procedures are repeated for 100 times and average each accuracy and the estimators for accuracy of diagnosis over 100 trials.

Table 5 shows the results about selection of a diagnosing model. In the domain of headache and meningitis, the dominant model is RHINOS diagnosing model and the simplest model, respectively, which matches with decision of medical experts. On the other hand, both models can be applied in the domain of CVD, although experts select the RHINOS diagnosing model. In the subsequent subsections, we only discuss the re-

[5]The thresholds δ_α and δ_κ is set to 0.75 and 0.5, respectively in these experiments.

Table 5: Selection of Diagnosing Model

Domain	Simplest	Weak	RHINOS	Experts' Model
headache	11	7	82	RHINOS
meningitis	65	13	22	Simplest
CVD	43	18	39	RHINOS

Table 6: Experimental Results on Headache (Averaged)

Method	ER	IR	DI	CPU time
PR3	95.0%	88.3%	93.2%	10.9 min
RHINOS	98.0%	95.0%	97.4%	–
C4.5	–	85.8%	–	15.6 min
CN2	–	87.0%	–	17.3 min
AQ15	–	86.2%	–	17.2 min

DEFINITIONS: PR3: PRIMEROSE3
ER: Exclusive Rule Accuracy
IR: Inclusive Rule Accuracy
DI: Disease Image Accuracy

sults of two domains in which PRIMEROSE3 correctly selects a diagnosing model.

5.1 Headache

Experimental results on headache are summarized in Table 6 and 7. In Table 6, the first column, exclusive rule accuracy denotes how many training samples that do not belong to a class are excluded correctly from the candidates. The second column is equivalent to the averaged classification accuracy and the third column shows how many symptoms, which cannot be explained by diagnostic conclusions, are detected by the disease image. And the fourth column shows CPU time needed to induced rules. The first row of Table 6 is the result of PRIMROSE3, and the second one is that of medical experts. And, for comparison, we compare the classification accuracy of inclusive rules with that of C4.5 (Quinlan, 1993), CN2 (Clark and Niblett, 1989) and AQ-15 (Michalski, et al. 1986), which is shown in the third to fifth row of Table 6. Table 7 shows the results of estimation derived by using repeated cross-validation method (R-CV) and the bootstrap method (BS).

These results is summarized to the following four points. First, the induced inclusive rules perform worse than those of medical experts, exclusive rules and disease images gain the same performance, compared with experts' rules. Secondly, our method performs a little better than four classical empirical learning methods, although the differences are not statistically significant.

Table 7: Experimental Results of Estimation

Method	Accuracy	R-CV	BS
PR3	88.3%	78.7%	91.6%
C4.5	85.8%	77.2%	90.7%
CN2	87.0%	72.9%	93.2%
AQ15	86.2%	74.6%	92.3%

Table 8: Experimental Results of Meningitis (Averaged)

Method	Accuracy	CPU time
PRIMEROSE3	83.9%	69 sec
Expert Rules	93.0%	–
C4.5	74.0%	57 sec
CN2	75.0%	60 sec
AQ15	84.7%	62 sec

Thirdly, PRIMEROSE3 is faster than the other methods. Finally, fourthly, R-CV estimator and BS estimator can be regarded as the lower boundary and the upper boundary of each rule accuracy. Hence the interval of these two estimators can be used as the estimator of performance of each rule.

5.2 Meningitis

Table 8 and 9 show the experimental results derived from databases on meningitis. The first row of Table 8 is the result of PRIMROSE3, and the second one is that of medical experts. And, for comparison, we compare the classification accuracy of inclusive rules with that of C4.5 (Quinlan, 1993), CN2 (Clark and Niblett, 1989) and AQ-15 (Michalski, et al. 1986), which is shown in the third to fifth row.

In Table 9, the results of estimation are derived by using repeated cross-validation method (R-CV) and the bootstrap method (BS).

As shown in Table 8, PRIMEROSE 3 performs as well as the other methods.

5.3 CVD

Experimental results on CVD are summarized in Table 10 and 11. The notations of both tables are the same as those of Table 6 and Table 7.

These results is summarized to the following four points. First, the induced inclusive rules perform worse than those of medical experts, exclusive rules and disease images gain the same performance, compared with experts' rules. Secondly, our method performs a little better than four classical empirical learning methods,

Table 9: Experimental Results of Estimation

Method	Accuracy	R-CV	BS
PR3	88.3%	78.7%	91.6%
C4.5	74.0%	70.9%	85.7%
CN2	75.0%	71.2%	85.7%
AQ15	84.7%	75.6%	87.3%

Table 10: Experimental Results on CVD (Averaged)

Method	ER	IR	DI	CPU time
PR3	91.0%	84.3%	94.3%	59 sec
RHINOS	97.5%	92.9%	93.6%	–
C4.5	–	79.7%	–	91 sec
CN2	–	78.7%	–	72 sec
AQ15	–	78.9%	–	81 sec

DEFINITIONS: PR3: PRIMEROSE3
ER: Exclusive Rule Accuracy
IR: Inclusive Rule Accuracy
DI: Disease Image Accuracy

although the differences are not statistically significant. Thirdly, PRIMEROSE3 is faster than the other methods. Finally, fourthly, R-CV estimator and BS estimator can be regarded as the lower boundary and the upper boundary of each rule accuracy. Hence the interval of these two estimators can be used as the estimator of performance of each rule.

6. Discussion

6.1 Exclusive Rule

As discussed in Section 3, we intend to formulate induction of exclusive rules by using the whole given attributes, although the original exclusive rules are described by the six basic questions, as shown in Appendix. Therefore induced exclusive rules have the maximum number of attributes whose conjunction R also satisfies $\kappa_R(D) = 1.0$. If this maximum combination includes the six basic attributes as a subset, then this selection of basic attributes is one of good choices of attributes, although redundant. Otherwise, the given six attributes may be redundant or the induced results may be insufficient. For the above example shown in Table 1, the maximum combination of attributes is {age, loc, nat, jolt, prod, nau, M1 } is included in both exclusive rules.

On the contrary, in the database for the above experiments, the maximum combination is 13 attributes, derived as follows: Age, Pain location, Nature of the pain, Severity of the pain, History since onset, Existence of jolt headache, Tendency of depression, and Tenderness of M1 to M6, which is a superset of the six basic attributes. Thus, this selection can be a good choice.

Table 11: Experimental Results of Estimation

Method	Accuracy	R-CV	BS
PR3	84.3%	68.7%	88.6%
C4.5	79.7%	67.2%	87.7%
CN2	78.7%	62.3%	89.2%
AQ15	78.9%	64.9%	92.3%

In this way, the induction of maximum combination can be also used as a "rough" check of induced results or our diagnosing model on exclusive rules, which can be formulated as below. [6]

Let A and E denote a set of the induced attributes for exclusive rules and a set of attributes acquired from domain experts. Thus, the following four relations can be considered. First, if $A \subset E$, then A is insufficient or E is redundant. Secondly, if $A = E$, then both sets are sufficient to represent diagnosing model in an applied domain. Thirdly, if $A \supset E$, then A is redundant or E is insufficient. Finally, fourthly, if intersection of A and E is not empty ($A \cap E \neq \phi$), then either or both sets are insufficient.

Reader may say that the above relations are weak and indeterminate. However, the above indefinite parts should be constrained by information on domain knowledge. For example, let us consider the case when $A \subset E$. When E is validated by experts, A is insufficient in the first relation. However, in general, E can be viewed as A obtained by large samples, and $A \supset E$ should hold, which shows that a given database is problematic. Moreover, the constraint on exclusive rules, $\kappa_R(D) = 1.0$, suggests that there exist a class which does not appear in the database, because the already given classes cannot support $\kappa_R(D) = 1.0$, that is, $[x]_R \cap D \neq D$ will hold in the future.

On the other hand, when E is not well given by experts and A is induced from sufficiently large samples, E will be redundant, which means that the proposed model for E does not fit to this database or this domain.

This kind of knowledge is important, because we sometimes need to know whether samples are enough to induce knowledge and whether an applied inducing model is useful to analyze databases.

[6] This discussion assumes that the whole attributes are sufficient to classify the present and the future cases into given classes.

Thus, the above four relations give simple examinations to check the characteristics of samples and the applicability of a given diagnosing model. It is our future work to develop more precise checking methodology for automated knowledge acquisition.

6.2 Precision for Inclusive Rules

In the above experiments, a threshold δ for selection of inclusive rules is set to 0.5 because of the following reason. Since the database on headache supports 10 classes, the naive "*a priori*" probability for each class is equal to $1/10 = 0.1$. Thus, when the probability of one disease should be set to 0.5, the other disease is suspected with probability $0.5/9 \approx 0.055$, which means that the assigned accuracy will become half.

Although this precision contributes to the reduction of computational complexity, this methodology, which gives a threshold in a static way, cause a serious problem. For example, there exists a case when the accuracy for the first, the second, and the third candidate is 0.5, 0.49, and 0.01, whereas accuracy for other classes is almost equal to 0. Formally, provided an attribute-value pair, R, the following equations hold: $\alpha_R(D_1) = 0.5, \alpha_R(D_2) = 0.49, \alpha_R(D_3) = 0.01$, and $\alpha_R(D_i) \approx 0 (i = 4, \cdots, 10)$. Then, both of the first and the second candidate should be suspected because those accuracies are very close, compared with the accuracy for the third and other classes. However, if a threshold is statically set to 0.5, then this pair is not included in positive rules for D_2. In this way, a threshold should be determined dynamically for each attribute-value pair. In the above example, an attribute-value pair should be included in positive rules of D_1 and D_2.

From discussion with domain experts, it is found that this type of reasoning is very natural, which may contribute to the differences between induced rules and ones acquired from medical experts. Thus, even in a learning algorithm, comparison between the whole given classes should be included in order to realize more plausible reasoning strategy.

Unfortunately, since the proposed algorithm runs for each disease independently, the above type of reasoning cannot be incorporated in a natural manner, which causes computational complexity to be higher. It is also our future work to develop such interacting process in the learning algorithm.

7. Related Work

7.1 AQ Algorithm

AQ is an inductive learning system based on incremental STAR algorithm(Michalski, 1983). This algorithm selects one "seed" from positive examples and starts from one "selector"(attribute-value pair) contained in this "seed" example. It adds selectors incrementally until the "complexes" (conjunction of attributes) explain only positive examples, called a **bounded star**. Since many complexes can satisfy these positive examples, AQ finds the most preferred ones, according to a flexible extra-logical criterion.

It would be worth noting that the positive examples which supports the complexes corresponds to the lower approximation, or the positive region in rough set theory. That is, the rules induced by AQ are equivalent to consistent rules defined by Pawlak when neither constructive generalization(Michalski, 1983; Wnek and Michalski) nor truncation(Michalski, et al. 1986) are used, and when the length of STAR is not restricted. As a matter of fact, AQ's star algorithm without constructive generalization can be reformulated by the concepts of rough sets. For example, a bounded star denoted by $G(e|U - D, m_0)$ in Michalski's notation is equal to $G = \{R_i | [x]_{R_i} = D_j\}$, such that $|G| = m_0$ where $|G|$ denotes the cardinality of G. This star is composed of many complexes, which is ordered by LEF_i, lexicographic evaluation functional, which is defined as the following pair:$< (-negcov, \tau_1), (poscov, \tau_2) >$ where *negcov* and *poscov* are numbers of negative and positive examples, respectively, covered by an expression in the star, and where τ_1 and τ_2 are tolerance threshold for criterion *poscov*, *negcov* ($\tau \in [0..100\%]$). This algorithm shows that AQ method is a kind of greedy algorithm which finds independent variables using selectors which are equivalent to equivalence relations in terms of rough sets.

Thus, our heuristic search method is very similar to AQ method, while our method uses statistical measures, rather than LEF criterion, which implicitly includes the notions of accuracy and coverage. The difference between our heuristic search procedure and AQ method is that PRIMEROSE3 explicitly uses accuracy and coverage and that it only uses elementary attribute-value pairs selected by the exhaustive search procedure, according to the characteristics of coverage, although AQ implicitly uses the criteria for both measures. The main reason why PRIMEROSE3 uses statistical measures is that discussion about the statistical characteristics of both measures is easier and that the definition of probabilistic rules is much clearer. As shown in Section 4, three kinds of rules are easily classified into three category with respect to accuracy and coverage. Especially, since coverage plays an important role in the classification of rules, it is very easy to implement an induction algorithm of exclusive rules and disease image. Thus, PRIMEROSE3 can be viewed as a combination of AQ algorithm and the exhaustive

search method.

7.2 Discovery of Association Rules

Mannila et al.(Mannila, 1994) report a new algorithm for discovery of association rules, which is one class of regularities, introduced by Agrawal et al.(Agrawal, et. al, 1993). Their method is very similar to ours with respect to the following two points.

(1) Association Rules The concept of association rules is similar to our induced rules. Actually, association rules can be described in the rough set framework.

That is, we say that an association rule over r (training samples) satisfies $W \Rightarrow B$ with respect to γ and σ, if

$$|[x]_W \cap [x]_B| \geq \sigma n, \tag{1}$$

and

$$\frac{|[x]_W \cap [x]_B|}{|[x]_W|} \geq \gamma, \tag{2}$$

where n, γ, and σ denotes the size of training samples, confidence threshold, and support threshold, respectively. Also, W and B denotes an equivalence relation and a class, respectively. Furthermore, we also say that W is *covering*, if

$$|[x]_W| \geq \sigma n. \tag{3}$$

It is notable that the left side of the above formulae (6) and (8) correspond to the formula (3) as to κ, coverage, and the left side of the formula (7) corresponds to (2) as to α, accuracy. The only difference is that we classify rules, corresponding to association rules, into three categories: exclusive rules, inclusive rules, and disease image.

The reason why we classify these rules is that this classification reflects the diagnosing model of medical experts, by which the computational speed of diagnostic reasoning is higher.

(2) Mannila's Algorithm Mannila et al. introduce an algorithm to find association rules based on Agrawal's algorithm. The main points of their algorithms are database pass and candidate generation. Database pass produces a set of attributes L_s as the collection of all covering sets of size s in C_s. Then, candidate generation calculates C_{s+1}, which denotes the collection of all the sets of attributes of size s, from L_s. Then, again, database pass is repeated to produce L_{s+1}. The effectiveness of this algorithm is guaranteed by the fact that all subsets of a covering set are covering.

The main difference between Mannila's algorithm and PRIMEROSE3 is that Mannila uses the check algorithm for covering to obtain association rules, whereas we use both accuracy and coverage to compute and classify rules.

In the discovery of association rules, all of the combination of attribute-value pairs in C_s have the property of covering. On the other hand, our algorithm does not focus on the above property of covering. It removes an attribute-value pair which has both high accuracy and high coverage. That is, PRIMEROSE3 does not search for regularities which satisfy covering, but search for regularities important for classification.

Thus, interestingly enough, when many attribute-value pairs have the covering property, or covers many training samples, Mannila's algorithm will be slow, although PRIMEROSE3 algorithm will be fast in this case. When few pairs cover many training samples, Mannila's algorithm will be fast, and our system will not be faster.

8. Conclusion

In this paper, we introduce a system, called PRIMEROSE3 which selects a diagnosing model, extracts not only classification rules for differential diagnosis, but also other medical knowledge which is needed for diagnosis, based on the selected diagnosing model. We evaluate this system by using three clinical databases, and compared the induced results with rules acquired from medical experts. The results show that our proposed method correctly induce RHINOS rules and estimate the statistical measures of rules.

Acknowledgements

The authors would like to thank the reviewers for their insightful comments. This research is supported by Grants-in-Aid for Scientific Research No.06680343 from the Ministry of Education, Science and Culture in Japan.

References

Agrawal, R., Imielinski, T., and Swami, A. Mining association rules between sets of items in large databases, *Proceedings of the 1993 International Conference on Management of Data (SIGMOD 93)*, pp. 207-216, 1993.

Breiman, L., Freidman, J., Olshen, R., and Stone, C. (1984). *Classification And Regression Trees.* Belmont, CA: Wadsworth International Group.

Buchanan, B. G. and Shortliffe, E. H.(eds.) (1984). *Rule-Based Expert Systems*, Addison-Wesley.

Cestnik, B., Kononenko, I., Bratko, I. (1987). Assistant 86: A knowledge elicitation tool for sophisticated

users. *Proceedings of the Second European Working Session on Learning*, pp.31-45, Sigma Press.

Clark, P., Niblett, T. (1989). The CN2 Induction Algorithm. *Machine Learning*, **3**,261-283.

Efron, B. (1982). *The Jackknife, the Bootstrap and Other Resampling Plans.* Society for Industrial and Applied Mathematics, Pennsylvania.

Efron, B. (1983). Estimating the error rate of a prediction rule: improvement on cross validation. *Journal of American Statistics Association*, **78**, 316-331.

Efron, B. (1986). How biased is the apparent error rate of a prediction rule ? *Journal of American Statistics Association*, **82**, 171-200.

Kimura, M., et al. (1985). RHINOS: A Consultation System for Diagnoses of Headache and Facial Pain: RHINOS. *Proceedings of the ninth International Joint Conference on Artificial Intelligence*, 393-396, Morgan Kaufmann, CA.

Indurkhya, N. and Weiss, S. (1991). Iterative Rule Induction Methods, *Applied Intelligence*, **1**, 43-54.

Mannila, H., Toivonen, H., Verkamo, A.I. Efficient Algorithms for Discovering Association Rules, *Proceedings of Knowledge Discovery in Databases (KDD-94)*, pp.181-192, AAAI press, CA, 1994.

Matsumura, Y., et al. (1986). Consultation system for diagnoses of headache and facial pain: RHINOS. *Medical Informatics*, **11**, 145-157.

Mclachlan, G. J. (1992). *Discriminant Analysis and Statistical Pattern Recognition.* John Wiley and Sons, New York.

Michalski, R. S. (1983). A Theory and Methodology of Machine Learning. Michalski, R.S., Carbonell, J.G. and Mitchell, T.M., *Machine Learning - An Artificial Intelligence Approach.* Morgan Kaufmann, Palo Alto, CA.

Michalski, R. S., Mozetic, I., Hong, J., and Lavrac, N. (1986). The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains. *Proceedings of the fifth National Conference on Artificial Intelligence*, 1041-1045, AAAI Press, Palo Alto, CA.

Pawlak, Z. (1991). *Rough Sets.* Kluwer Academic Publishers, Dordrecht.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, **1**, 81-106.

Quinlan, J.R. (1993). *C4.5 - Programs for Machine Learning*, Morgan Kaufmann, CA.

Walker, M. G. and Olshen, R. A. (1992). Probability Estimation for Biomedical Classification Problems. *Proceedings of the sixteenth Symposium on Computer Applications on Medical Care*, McGrawHill, New York.

Wnek, J. and Michalski, R.Z. Hypothesis-Driven Constructive Induction in AQ17-HCI: A Method and Experiments, *Machine Learning*, **14**, 139-168, 1994.

Ziarko, W. (1991). The Discovery, Analysis, and Representation of Data Dependencies in Databases, in: Shapiro, G. P. and Frawley, W. J.(eds), *Knowledge Discovery in Databases*, AAAI press, Palo Alto, CA, pp.195-209.

Ziarko, W. (1993). Variable Precision Rough Set Model. *Journal of Computer and System Sciences*, **46**, 39-59.

Appendix: RHINOS Rules

Using the diagnosing model introduced by (Matsumura, et al., 1986), we consider the following three kinds of rules corresponding to each process and develop the following algorithms for acquiring these rules from medical experts.

(1)Exclusive Rule This rule corresponds to exclusive reasoning. In other words, the premise of this rule is equivalent to the necessity condition of the diagnostic conclusion. From the discussion with medical experts, we selected the following six basic attributes which are minimally indispensable to defining the necessity condition: *1. Age, 2. Pain location, 3. Nature of the pain, 4. Severity of the pain, 5. History since onset, 6. Existence of jolt headache.*

For example, the exclusive rule of common migraine is the following:

```
In order to suspect common migraine,
the following symptoms are required:
pain location: not eyes and
nature: throbbing or persistent or radiating
and history: paroxysmal or sudden and
jolt headache: positive.
```

One of the reason why we selected the six attributes is to solve the interface problem of expert systems: if the whole attributes are considered, we also have to input all the symptoms which are not needed for diagnosis. To make exclusive reasoning compact, the only

minimal requirements are chosen. It is notable that this kind of selection can be viewed as the ordering of given attributes, and it is expected that such ordering can be induced from databases. Therefore, in PRIMEROSE3, an algorithm for induction of exclusive rules scans the whole given attributes. It is because the minimal requirements for describing exclusive rules can be computed after all the exclusive rules are induced. Furthermore, this ordering can be viewed as a "rough" check of induced results and applicability of our diagnosing model. This issue is discussed in Section 6.

(2)Inclusive Rule This rule consists of a set of positive rules, the premises of which are composed of a set of manifestations specific to a disease to be included. If a patient satisfy one set, we suspect this disease with some probability. This rule is derived from medical experts by using the following algorithm for each disease: *1. Take a set of manifestations by which we strongly suspect a disease. 2. Set the probability that a patient has the disease with this set of manifestations:SI(Satisfactory Index) 3. Set the ratio of the patients who satisfy the set to all the patients of this disease:CI(Covering Index) 4. If sum of the derived CI(tCI) is equal to 1.0 then end. If not, goto 5. 5. For the patients of this disease who do not satisfy all the collected set of manifestations, goto 1.* Therefore a positive rule is described by a set of manifestations, its satisfactory index (SI), which corresponds to *accuracy measure*, and its covering index (CI), which corresponds to *total positive rate* or *coverage*. Note that SI and CI are given empirically by medical experts.

Formally, each positive rule is represented as a quadruple:
$\langle d, R_i, SI_i, CI_i \rangle$, where d denotes its conclusion, and R_i denotes its premise. Hence each inclusive rule is described as: $\langle \{\langle d, R_1, SI_1(, CI_1)\rangle, \cdots, \langle d, R_k, SI_k(, CI_k)\rangle\}, tCI\rangle$, where total CI(tCI) is defined as the CI of a total rule, composed of a disjunctive formula of all rules, $R_1 \vee R_2 \vee \cdots \vee R_k$.

Let us show the inclusive rule of common migraine (tCI=0.9) as an example, which is composed of the following three rules:

```
If history: paroxysmal, jolt headache:
positive, nature: throbbing or persistent,
prodrome: no, intermittent symptom: no,
persistent time: more than 6 hours, and
pain location: not eye,
then common migraine is suspected
with accuracy 0.9 (SI=0.9) and this rule
covers 60 percent of the total cases
(CI=0.6).

If history: paroxysmal, jolt headache:
positive, nature: throbbing or persistent,
prodrome: no, intermittent symptom: no,
and pain location: not eye,
then common migraine is suspected
with accuracy 0.8 (SI=0.8) and this rule
covers 80 percent of the total cases
(CI=0.8).

If history: sudden, jolt headache:
positive, nature: throbbing or persisted,
and prodrome: no,
then common migraine is suspected
with accuracy 0.5 (SI=0.5) and this rule
covers 30 percent of the total case
(CI=0.3).
```

In the above rules, tCI shows that the disjunctive form of above three rules covers 90 percent of total cases of common migraine.[7]

It also means that 10 percent of common migraine cannot be diagnosed by the above rules.

(3)Disease Image This rule is used to detect complications of multiple diseases, acquired by all the possible manifestations of a disease. Using this rule, we search for the manifestations which cannot be explained by the diagnosed disease. Those symptoms suggests complications of other diseases. For example, the disease image of common migraine is shown as follows:

```
The following symptoms can be explained
by common migraine:
pain location: any or
tendency of depression : negative or
jolt headache: positive or ........
```

Therefore, when a patient who suffers from common migraine is depressing, it is suspected that he or she may also have other disease, because this symptom cannot be explained by common migraine.

As shown above, these algorithms are straightforward, and is based on set-theoretic framework.

[7]Since tCI is based on total coverage by the disjunctive form of rules, it is not equal to total sum of CI values of all rules.

Comparative Analysis of Amino-acid sequences based on Rough Set Theory and Change of Representation

Shusaku Tsumoto and Hiroshi Tanaka
Department of Information Medicine, Medical Research Institute,
Tokyo Medical and Dental University,
1-5-45 Yushima, Bunkyo-ku Tokyo 113 Japan
E-mail: tsumoto.com@tmd.ac.jp, tanaka@cim.tmd.ac.jp

Abstract

Protein structure analysis from DNA sequences is an important and fast growing area in both computer science and biochemistry. One of the most important problems is that two proteins, both of which have the similar three-dimensional structure, have different functions, such as lysozyme and lactalbumin. In such cases, comparative analysis of both amino acid sequences is effective to detect the functional and structural differences. In this paper, we introduce a system, called MW1.5 (Molecular biologists' Workbench version 1.5), which extracts differential knowledge from amino-acid sequences by using rough-set based classification, statistical analysis and change of representation. This method is applied to the following two domain: comparative analysis of lysozyme and α-lactalbumin, and analysis of immunoglobulin structure. The results show that several interesting results from amino-acid sequences, are obtained which have not been reported before.

1. Introduction

Protein structure analysis from DNA sequences is an important and fast growing area in both computer science and biochemistry.

One of the most important problems is that two proteins both of which has the similar three-dimensional structure have different functions, such as lysozyme and lactalbumin. In such cases, comparative analysis of both amino acid sequences is effective to detect the functional and structural differences, since local structure should be of primary importance to contribute to the characteristics of theses proteins.

However, in general, only knowledge from sequences is insufficient for analysis, because protein function is thought to be realized by chemical interaction between the components in amino-acid sequences. That is, it is necessary to incorporate domain knowledge, such as chemical knowledge to make comparative analysis be sufficient. Therefore we need to introduce a mechanism which controls the application of domain knowledge in order to analyze the characteristics of induced results and to extract as much information as possible from databases(Zytkow, 1992).

In order to incorporate the above control strategy into machine learning methods, we introduce a system, called MW1.5 (Molecular biologists' Workbench version 1.5), which extracts knowledge from amino-acid sequences by controlling application of domain knowledge automatically.

MW1.5 consists of the following five procedures. First, it exhaustively induces all the classification rules from databases of amino-acid sequences. Secondly, MW1.5 changes representation of amino-acid sequences with respect to the main chemical features. Then, thirdly, all the rules are induced from each transformed databases. Next, fourthly, the program estimates the secondary structure of amino-acid sequences via *Chou-Fasman* method (Chou and Fasman, 1974). Finally, fifthly, MW1.5 induces all the rules from the databases of secondary structure.

This method is applied to comparative analysis of lysozyme and α-lactalbumin, and analysis of structure of immunoglobulin. The results show that several interesting results are obtained from amino-acid sequences, which has not been reported before. Based on these new discovered knowledge, several experiments are being planned in order to validate discovered results. Interestingly enough, some of them are recently confirmed by biochemical experiments (Tsumoto, 1994; Tsumoto, 1995). The evaluation of other results will be reported when the whole experiments will have been completed.

The paper is organized as follows: Section 2 discusses the problems of empirical learning methods when the method is applied to amino acid sequences. Section 3 presents the discovery strategy of MW1.5 and how it works. Section 4 shows the results of application of this system to comparative analysis of lysozyme IIc and α-lactalbumin, and to analysis of structure of immunoglobulin. Section 5 discussed related work, and finally, Section 6 concludes this paper.

2. Problems of Empirical Learning Methods

It is easy to see that simple application of machine learning methods to DNA or amino-acid sequences

without using domain-specific knowledge cannot induce enough knowledge.

For example, simple application of induction of decision trees (Breiman, et al. 1984; Quinlna, 1993) generates only one rule from many possible rules. However, many attributes (exactly, 52 attributes) have the maximum value of information gain. Thus, we have to choose one of such attributes. If simplicity is preferred, that is, if the number of leaves should be minimized, then location 44 will be selected as shown below.

$$\begin{cases} 44 = N & \cdots lysozyme & \cdots (45 cases) \\ 44 = V & \cdots \alpha - lactalbumin & \cdots (23 cases) \end{cases}$$

In this case, we get a simple tree, which consists of one node and two leaves. Unfortunately, this result is not enough, since our objective is not to find a simple rule for classification, but to find as much information as possible.

However, exhaustive induction of possible rules also causes another problem: it is very difficult to interpret all the possible rules without using domain knowledge.

Hence it is very crucial to control application of domain knowledge, according to what problem we want to solve. If we need only some evidential knowledge, we should strictly apply domain knowledge, and focus only on several attributes of training samples. These cognitive aspects of machine discovery system are discussed by researchers on machine discovery (Zytkow, 1992).

3. Discovery Strategy

In order to implement discovery strategy of molecular biologists, we develop a system, called MW1.5 (Molecular biologists' Workbench version 1.5), which extracts knowledge from amino-acid sequences by controlling application of domain knowledge automatically.

MW1.5 consists of the following five procedures. First, it applies PRIMEROSE-EX2, which will be discussed in the next subsection, and exhaustively induces all the classification rules from databases of amino-acid sequences. Secondly, MW1.5 changes representation of amino-acid sequences with respect to the main chemical features of amino acids, such as the characteristics of electronic charge (i.e., basic, neutral, or acidic) (**Primary Structure Rearrangement**). That is, MW1.5 generates new databases focused on a certain chemical property from original databases. Then, thirdly, PRIMEROSE-EX2 will be applied again, all the rules are induced from each database generated by the second procedure. Furthermore, the statistics of each chemical characteristic are calculated. Next, fourthly, the program estimates the secondary structure of amino-acid sequences using *Chou-Fasman* method (Chou and Fasman, 1974) (**Secondary Structure Rearrangement**). Finally, fifthly, MW1.5 induces all the rules from the databases of secondary structure, applying PRIMEROSE-EX2.

3.1 PRIMEROSE-EX2

In order to induce rule exhaustively, we introduce a program, called PRIMEROSE-EX2 (Probabilistic Rule Induction Method based on Rough Sets for Exhaustive induction ver 2.0). This method is based on rough set theory, which gives a mathematical approach to the reduction of decision tables, corresponding to the exhaustive search for possible rules. For the limitation of the space, we only discuss the definition of probabilistic rules of PRIMEROSE-EX2 and an induction algorithm of this system. Readers, who would like to know further information on rough sets, could refer to (Pawlak, 1991; Ziarko, 1991).

Rules of PRIMEROSE-EX2 In the framework of rough set theory, we have several specific notations as follows. First, a combination of attribute-value pairs, corresponding to a complex in AQ terminology, is denoted by an equivalence relation R_f, which is defined as follows.

Definition 1 (Equivalence Relation) *Let U be a universe, and V be a set of values. A total function f from U to V is called an assignment function of an attribute. Then, we introduce an equivalence relation R_f such that for any $u, v \in U$, $u \equiv R_f v$ iff $f(u) = f(v)$.*

For example, $[a = 1]\&[b = 1]$ will be one equivalence relation, denoted by $R_f = [a = 1]\&[b = 1]$. Secondly, a set of samples which satisfy R_f is denoted by $[x]_{R_f}$, corresponding to a star in AQ terminology. For example, when $\{1, 2, 3\}$ is a set of samples which satisfy R, $[x]_{R_f}$ is equal to $\{1, 2, 3\}$ [1]. Finally, thirdly, U, which stands for "Universe", denotes the whole training samples.

According to this notation, probabilistic rules are defined as follows:

Definition 2 (Probabilistic Rules) *Let R_f be an equivalence relation specified by some assignment function f, D denote a set whose elements belong to a class d, or positive examples in the whole training samples (the universe), U, and $[x]_{R_f}$ denote the set of training samples which satisfy an equivalence relation R_f. Finally, let $|D|$ denote the cardinality of D, that is, the total number of samples in D.*

A probabilistic rule of D is defined as a quadruple, $< R_f \stackrel{\alpha,\kappa,p}{\rightarrow} d, \alpha, \kappa, p >$, where $R_f \stackrel{\alpha,\kappa,p}{\rightarrow} d$ satisfies the following conditions:

$$(1) \quad [x]_{R_f} \bigcap D \neq \phi, \tag{1}$$

$$(2) \quad \alpha = \frac{|[x]_{R_f} \bigcap D)|}{|[x]_{R_f}|}, \tag{2}$$

$$(3) \quad \kappa = \frac{|[x]_{R_f} \bigcap D)|}{|D|}, \tag{3}$$

$$(4) \quad p : p\text{-}value\ of\ \chi^2\text{-}statistics, \tag{4}$$

[1] In this notation, "1" denotes the first(1st) sample in a dataset.

where p is a p-value of χ^2-statistics when the relation between $[x]_{R_f}$, D, and U is tested as a contingency table. □

The intuitive meaning of the above three variables, α, κ, and p-value is given as follows. First, α corresponds to the accuracy measure. For example, if α of a rule is equal to 0.9, then the accuracy is also equal to 0.9. Secondly, κ is a statistical measure of how proportion of D is covered by this rule, that is, coverage or a true positive rate. For example, when κ is equal to 0.5, half of the members of a class belongs to the set whose members satisfy that equivalence relation. Finally, thirdly, p-value denotes the statistical reliability of a rule $R \stackrel{\alpha,\kappa,p}{\rightarrow} d$. For example, when p is equal to 0.95, the reliability of the rule is 95% [2]

As to the calculation of p-value, we view the relation between $[x]_{R_f}$, D, and U as a contingency table as shown in the following table.

	d	$\neg d$	Total
R_f	s	t	$s+t$
$\neg R_f$	u	v	$u+v$
Total	$s+u$	$t+v$	$s+t+u+v(=n)$

In the above table, $\neg R_f$ and $\neg d$ denotes the negation of R and d, respectively. Note that each items in the table can be described in the framework of rough set theory, that is, s, t, u, v can be described as $|[x]_{R_f} \cap D|(=s)$, $|[x]_{R_f} \cap (U-D)|(=t)$, $|D-[x]_{R_f} \cap D|(=u)$, and $|(U-D)-[x]_{R_f} \cap (U-D)|(=v)$, respectively. It is also notable that $s+t=|[x]_{R_f}|$, $s+u=|D|$, and $s+t+u+v=|U|$.

From the above table, χ^2-statistic can be calculated as:

$$\chi^2 = \frac{n(sv-tu)^2}{(s+u)(t+v)(s+t)(u+v)}, \qquad (5)$$

where n, s, t, u, v is given in the above table. This measure is a test statistic to check whether R is independent of d. In other words, it indicates whether R is not useful for classification of d or not. From the value of this statistics, p-value of null hypothesis[3] is calculated from where this value is located in the χ^2-distribution. For example, when the p-value of χ^2-statistics χ_0 is equal to 0.01, the region whose χ^2-statistics is below χ_0 occupies 1% of the whole distribution. Thus, the probability with which this event will occur is 99%.

According to those values, we classify the induced probabilistic rules into the following four categories:

1. Definite Rules: $\alpha = 1.0$ and $\kappa = 1.0$,
2. Significant Rules: $0.5 < \alpha < 1.0$ and $0.9 \leq p < 1.0$
3. Strong Rules: $0.5 < \alpha < 1.0$ and $0.5 < p < 0.9$,
4. Weak Rules: $0 < \alpha \leq 0.5$ or $0 < p \leq 0.5$.

An algorithm for PRIMEROSE-EX2 Let D denote training samples of the target class d, or *positive examples*. In the following algorithm, we provide two kinds of specific sets. The one is L_i, which denotes a set of equivalence relations whose size of attribute-value pairs is equal to $i-1$. For example, L_3 includes $[a=1]\&[b=1]$, whereas L_2 includes $[a=1]$ and $[b=1]$. The other is M_i, which denotes a set of equivalence relations for weak rules. For example, when M_2 includes a $[a=1]\&[b=1]$, the accuracy of $[a=1]\&[b=1]$ as to the target concept is lower than 0.5 or the p-value of χ^2-statistics as to the target concept is lower than 0.5. Thus, an equivalence relation in M_i is weak for classification or do not cover enough training samples.

Based on these notations, the search procedure can be described as a kind of the greedy algorithm shown in Fig. 1. The above procedure is repeated for all the attribute-value pairs. In the above algorithm, equivalence relations for significant rules and strong rules in L_i are removed from candidates for the generation of L_{i+1}, because they are not included in M_i. Thus, if significant members of L_i are not included in M_i, then computational complexity of generation of L_{i+1} is small.

How to Deal with Continuous Data Almost all the chemical charcteristics of amino acids are provided as continous data. For example, a coefficient of hydrophobicity of K(Lysine) is equal to 2.27, which means that it takes 2.27 $kcal/mol$ energy to remove water around Lysine. Thus, it is necessary to deal with continous data in order to extract knowledge from the chemical characteristics.

For solution, PRIMEROSE-EX2 transforms continous data into categorical data, some of which is similar to C4.5 (Quinlna, 1993), in the following ways. First, provided the attribute a_i, the system sort database with respect to the value of a_i, such as $\{v_1, v_2, v_3, \cdots, v_j\}$, where $v_1 < v_2 < \cdots < v_j$. Secondly, for each member in the above list, the attribute-value pair is translated into the following binary form: $[a_i \leq v_k]$ and $[a_i > v_k]$ $(1 \leq k \leq j)$. Thus, new j binary attributes will be generated. And thirdly, α, κ, and χ^2-statistics will be calculated for each generated binary attribute. Finally, fourthly, the pair which induces the best rule is selected as a candidate of transformation.

However, this tranlation is only effective to each level[4]. That is, for each level, the above transformation algorithm should be peformed.

[2]This definition is different from that in statistical test. In statistical test setting, p-value denotes the probability that **null hypothesis**, or a negation of a hypothesis to be proved, is true. Thus, p-value calculated from statistical distribution, $\hat{p}$ is equal to the probability that null hypothesis is true. On the other hand, in our setting, p-value is equal to $1-\hat{p}$, which denotes the probability that null hypothesis is false.

[3]As discussed above, null hypothesis is negation of the hypothesis to be proved.

[4]In the subsequent sections, a level l is defined as the number of attribute-value pairs in the premise of a probabilistic rule. For example, a level 2 means that the number

Thus, continuous attributes should be always included in a list M in Fig. 1, which is a list of candidates to generate the whole combination of the conjunctive formulae. For example, at the level 2, let a_1, a_2 and a_3 be attributes whose values are continous. Then, when a_1 is included only in a list of weak rules at the level 1 ($[a_1 \leq v_k]$ and $[a_1 > v_k]$)[5], the tranformation of a_1 at this level will be used to generate the combination. That is, an attribute a_1 is fixed to a binary atrribute. Next, the following combination is considered: $a_1 \& a_2$, $a_1 \& a_3$. Then, the translation procedure is performed for each combination under the condition which $[a_1 \leq v_k]$ or which $[a_1 > v_k]$.

3.2 Change of Representation

We introduce two kinds of change of representation. One is to generate new databases which focus on a certain chemical characteristic from original databases, called *primary structure rearrangement.* The other one is to transform original databases, according to the estimation of the secondary structure, called *secondary structure rearrangement.*

Primary Structure Rearrangement The most important chemical characteristics of amino acids which are thought to contribute to determine a protein structure are the following: hydrophobicity, polarity or electronic charge of a side chain, the size of an amino acid, and the tendency of an amino acid to locate the interior of proteins.

For example, in the case of hydrophobicity, which denotes how much an amino acid is intimate with water molecule, a coefficient is assigned to each amino acids: a value 3.95 is assigned to R(Arginine), which is the least hydrophobic amino acid, and a value -2.27 is assigned to F(Phenylalanine). Therefore, in the latter case, F will be translated into: $[hydrophobicity = -2.27]$. Using these notations, we can change representation of amino acid sequences. For example, let us consider a case when an attribute-value pair of an original database is $[33 = F]$, which denotes that the 33th amino acid of a protein is F (phenylalanine). Because phenylalanine (F) is hydrophobic, this attribute-value pair is transformed into: $[33 = [hydrophobicity = -2.27]]$. This procedure is repeated for all the amino-acids in an original sequence.

Then, for rule induction, the translation procedure introduced in 3.1.3 is used.

Secondary Structure Rearrangement Next, MW1.5 estimates secondary structure from amino-acid sequences using the *Chou-Fasman* method (Chou and Fasman, 1974), which is the most popular estimation method [6]. This *Chou-Fasman* method outputs the place where specific secondary structures: $\alpha - helix$, $\beta - sheet$, and *turn*. According to this estimation, MW1.5 changes representation of original databases. For example, the 4th to 10th amino acids are estimated to form α-helix structure. Based on the above results, the value of each attribute, which is the address of a primary sequence, are replaced by the above knowledge on secondary structure. In the above example, the values of the 4 th to 10th attributes are substituted for α-helix, α-helix, α-helix, α-helix, α-helix, and α-helix. That is,

Primary Structure	E	R	C	E	L	A
↓	↓	↓	↓	↓	↓	↓
Secondary Structure	α	α	α	α	α	α.

It is notable that some attributes may have no specific secondary structure. In these cases, the value of these attributes are replaced by one of the four characteristics: {hydrophobic, polar, acidic, basic}, since they play an important role in making secondary structure, as discussed in the section on primary structure rearrangement. For example, let us consider a case when an attribute-value pair of an original database is $[86 = D]$, which denotes that the 86th amino acid of a protein is D (asparatic acid). Because asparatic acid (D) is acidic, this attribute-value pair is transformed into: $[86 = acidic]$ [7].

4. Experimental Results and Discussion

4.1 Lysozyme and α-Lactalbumin

Lysozyme IIc is a enzyme which dissolves the bacterial walls and suppress the growth of bacteria. All living things have this kind of enzyme, and especially, in the category of vertebrate animals, such as fishes, birds, and monkeys, the sequences are almost preserved.

On the other hand, α-lactalbumin functions as a coenzyme of one reaction which dissolves the chemicals in milk into those easy for babies to take nutrition. So this enzyme only exists in the mammals, such as monkeys, and the marsupials, such as kangaroos.

The comparative analysis of these two proteins is one of the most interesting subjects in molecular biology because of the following two reasons (McKenzie and White, 1991). First, α-lactalbumin are thought to be originated from lysozyme IIc, since both of the sequences are very similar. According to the results of homological search, about 60 % of the sequences of α-lactalbumin matches with those of lysozyme, which suggests that they are of the same origin. In addition

of attribute-value pairs is equal to 2, such as $[a = 1] \& [b = 1]$.

[5] If no attribute is in list of weak rules, then the attribute which gives the worst rules will be selected.

[6] It is notable that our method is independent of this estimation method. Thus, we can replace the *Chou-Fasman* method with the new methods which may gain more predictive accuracy, when such methods are obtained.

[7] It is notable that this information can be retrieved from the database generated in the process of primary structure rearrangement.

Table 1: Results of Primary Structure Rearrangement

Protein	Amino Acid and its Location		
lysozyme c	N 27	(A,L 31)	K 33
α-lact	E 27	T 31	F 33
lysozyme c	E 35	N 44	(Y,D 53)
α-lact	(I,S,T 35)	V 44	E 53
lysozyme c	(A,G 76)	(A,R 107)	
α-lact	I 76	D 107	
lysozyme c	(G,D,Q 117)	L 129	
α-lact	S 117	E 129	

Table 2: Results of Secondary Structure Rearrangement

Protein	Location		
	70-77	83-94	98-104
lysozyme c	hydrophobic	hydrophobic	loop
α-lact	polar	acidic	α-helix
	107-110	113-117	
lysozyme c	α-helix	basic	
α-lact	hydrophobic	hydrophobic	

to this similarity, the global three-dimensional structure of these two proteins is almost the same. Secondly, it is not well known what kinds of sequences mainly contribute to the functions of both enzymes, although many experiments suggest that interactions of several components play an important role in those functions.

We apply MW1.5 to 23 sequences of α-lactalbumin and 45 sequences of lysozyme from PIR databases, both of which are used as original training samples. Then, as inputs of MW1.5, we use the sequences processed by multiple alignment procedures.

The induced results are shown in Table 1 and 2, where the following three interesting results are obtained [8]. First, Table 1 shows the induced definite rules before change of representation. From the second to sixth columns, alphabets denote amino-acids, and the numbers denote the location in the sequence of a protein. For example, N 27 means that the 27th amino acid of lysozyme IIc is N, or aspargine. These results mean that these amino acids are specific to each protein. In other words, the most characteristic regions are expected to be included. Actually, it is known that E 35, and Y or D 53 are the active site of lysozyme, and also K 33, N 44 and A or R 107 are said to play an important role in its function (McKenzie and White, 1991). However, N 27 and L 129 are new

[8]The shown results are mainly induced definite rules and significant rules, because including strong and weak rules takes much more space. Thus, due to the limitation of space, we only discuss the results of definite rules and significant rules.

Table 3: Results of IgG sequences

Protein	Location				
	(51)	52A,B,C	(59)	60	61
Glycosamide	(Ile)	Pro	(Tyr)	Ala	Pro
Protein	(Ile)	Lys	(Tyr)	Asn	Glu

discovery results, and no observations or experimental results are reported. Thus, these acids may contribute to the function of lysozyme. Secondly, Table 2 shows the results of the definite rules after secondary structure rearrangement. The second row shows the location in sequences, for example, 70-77 means 70th to 77th amino acid in sequences of lysozyme c. Interestingly, although specific amino acids are mainly located at the lower address part (called it N-terminal), specific local structure are mainly located at the higher address part (called it C-terminal). The most significant regions are 98-104 and 113-117, because each secondary structure is very different. Other regions also show that hydrophobic regions of lysozyme correspond to non-hydrophobic regions of α-lactalbumin, and vice versa. Thus, these regions may play an important role in realizing each function [9].

4.2 Structure of Immunoglobulin

The main function of Immunoglobulin G(IgG) is as an antibody to specific chemical agent, such as bacterial wall(Lewin, 1994). There are many kinds of IgG, some of which bind small chemicals, other of which bind large proteins. It is thought that such specificities can be determined by characteristics of "variable" region, called CDR-1, CDR-2, and CDR-3(Kabat, 1991). Those IgGs are classified into two categories: those which bind a chain of glycosamides, which is hydrophobic, and those which bind a protein, which is hydrophilic. Thus, it is expected that these characteristics are coded in the "variable" region.

We apply MW1.5 to 1438 sequences of IgG, which consists of 349 IgG specific to hydrophobic chemical agents, 1089 IgG specific to hydrophilic ones.

In this domain, no definite rules are derived, and the most important results are induced as significant rules, shown in Table 3, where Glycosamide and Protein denotes IgG which bind hydrophobic chemicals and IgG which bind hydrophilic chemicals, respectively and where the second row shows the location in sequences. For example, 52 means 52th amino acid in the sequences of IgG. (51) and (59) denotes the common amino acids in both types of immunoglobulin sequences.

Interestingly, in the neighbors of (51) and (59), there

[9]Tsumoto, K. and Kumagai, I. obtain interesting results, which suggest that 98-104th amino acids play important roles in lysozyme function (Tsumoto, 1994).

exists sequences specific to each type of IgG. As to Glycosamide type, Proline(Pro) seems to play an important role, because Pro is a typical hydrophobic protein. On the other hand, as to Protein type, Lysine(Lys), Asparatic acid(Asp), and Glutamine(Glu) seems to play an important role in its function. These chemical characteristics are also detected by significant rules induced after secondary structure rearrangement: Glycosamide type has a hydrophobic region from 51 to 65 amino acids, but Protein tye has α-helix region in this area [10].

5. Related Work

5.1 Discovery of Association Rules

Mannila et al.(Mannila, 1994) report a new algorithm for discovery of association rules, which is one class of regularities, introduced by Agrawal et al.(Agrawal, et al. 1993). Their method is very similar to ours with respect to the following two points.

(1) Association Rules The concept of association rules is similar to our induced rules. Actually, association rules can be described in the rough set framework.

That is, we say that an association rule over r (training samples) satisfies $W \Rightarrow B$ with respect to γ and σ, if

$$|[x]_W \cap [x]_B| \geq \sigma n, \tag{6}$$

and

$$\frac{|[x]_W \cap [x]_B|}{|[x]_W|} \geq \gamma, \tag{7}$$

where n, γ, and σ denotes the size of training samples, confidence threshold, and support threshold, respectively. Also, W and B denotes an equivalence relation and a class, respectively. Furthermore, we also say that W is *covering*, if

$$|[x]_W| \geq \sigma n. \tag{8}$$

It is notable that the left side of the above formulae (6) and (8) correspond to the formula (3) as to κ, coverage, and the left side of the formula (7) corresponds to (2) as to α, accuracy. The only difference is that we classify rules, corresponding to association rules, into three categories: definite rules, significant rules, and strong rules.

The reason why we classify these rules is that this type of classification can be viewed as the ordering of rules or hypothesis. That is, definite rules correspond to the strongest hypotheses. However, these strongest rules may not be interesting for discovery. Then, significant rules will be considered for the candidates of discovery. If they are not so important, then strong rules will be considered. Finally, all the three kinds of rules are found to be not important, then we should search for weak rules. In this way, we simulate the discovery strategy of biochemists by using the classification of classification rules.

(2) Mannila's Algorithm Mannila et al. introduce an algorithm to find association rules based on Agrawal's algorithm. The main points of their algorithms are database pass and candidate generation. Database pass produces a set of attributes L_s as the collection of all covering sets of size s in C_s. Then, the candidate generation calculates C_{s+1}, which denotes the collection of all the sets of attributes of size s, from L_s. Then, again, database pass is repeated to produce L_{s+1}. The effectiveness of this algorithm is guaranteed by the fact that all subsets of a covering set are covering.

The main difference between Mannila's algorithm and our MW1.5 algorithm is that Mannila uses the check algorithm for covering to obtain association rules, whereas we use statistical analysis to compute and classify rules.

In the discovery of association rules, all of the combination of attribute-value pairs in C_s have the property of covering. On the other hand, our algorithm does not focus on the above property of covering. It removes an attribute-value pair which has both high accuracy and high coverage from L_s and does not include in M_s. That is, PRIMEROSE-EX does not search for regularities which satisfy covering, but search for regularities important for classification.

Thus, interestingly enough, when many attribute-value pairs have the covering property, or covers many training samples, Mannila's algorithm will be slow, although PRIMEROSE-EX algorithm will be fast in this case. When few pairs covers many training samples, Mannila's algorithm will be fast, and our system will not be faster.

5.2 Ziarko's KDD-R

Ziarko and Shan develop a comprehensive system for knowledge discovery in databases using rough sets, called KDD-R (Ziarko, 1995b). Their system consists of the four functional units: data processing unit, a unit for analysis of dependencies, a unit for computation of rules from data, and decision unit.

The most important unit is one for computation of rules from data. This unit computes all, or some, approximate rules with decision probabilities, where the probabilities are restricted by lower and upper limit parameters specifying the area of user interest. The rules can be computed for a selected reduct using the method of decision matrix (Ziarko,1995a), which is an extention of discerniblity matrix (Skoworn and Rauzer, 1992).

The main difference between KDD-R and our system is that PRIMEROSE-EX adopts statistical measures to prune attribute-value pairs. In PRIMEROSE-EX, attribute-value pairs which have high accuracy and high coverage will be used for rule generation and re-

[10] Recently, our co-authors have got the results which suggest that Tyr(59) and its neighbors play an important role in function of IgG (Tsumoto, 1995).

moved from the candidates of complexed rules. On the other hand, KDD-R first removes dependent superfluous attributes using the extension of rough set model, called Variable Precision Rough Set model and then calculates rules using the technique of decision matrix, which is very useful to generate all approximate rules.

Thus, KDD-R focuses mainly on dependencies of attributes with respect to selection of attribute-value pairs, whereas PRIMEROSE-EX focuses on mainly on the statistical significance of attribute-value pairs, which is used for selection of attribute-value pairs. Therefore the performance of each system may depend on the characteristics of an applied domain. That is, KDD-R may outperform our method when a dataset has many dependent attributes.

6. Conclusion

In this paper, a system based on combination of a probabilistic rule induction method with domain knowledge is introduced, called MW1.5 (Molecular biologists' Workbench version 1.5) in order to detect the structural differences by using compartive analysis. This method is applied to comparative analysis of lysozyme and α-lactalbumin and to analysis of structure of immunoglobulin. The results show that we get some interesting results from amino-acid sequences, which have not been reported before.

References

Agrawal, R., Imielinski, T., and Swami, A. "Mining association rules between sets of items in large databases," *Proceedings of the 1993 International Conference on Management of Data (SIGMOD 93)*, pp. 207-216, 1993.

Breiman, L., Freidman, J., Olshen, R., and Stone, C. *Classification And Regression Trees*. Belmont, CA: Wadsworth International Group, 1984.

Chou, P.Y. and Fasman, G.D. "Prediction of protein conformation," *Biochemistry*, **13**, pp.222-244, 1974.

Hunter, L.(ed) *Artificial Intelligence and Molecular Biology*, AAAI press, CA, 1993.

Kabat, E.A. et al.(eds.) *Sequences of Proteins of Immunological Interest*, 5th edition, NIH publication, 1991.

Lewin, B. *Genes V.*, Oxford University Press, London, 1994.

Mannila, H., Toivonen, H., Verkamo, A.I. "Efficient Algorithms for Discovering Association Rules," *Proceedings of Knowledge Discovery in Databases (KDD-94)*, pp.181-192, AAAI press, CA, 1994.

McKenzie, H.A. and White, Jr., F.H. "Lysozyme and α-lactalbumin: Structure, Function, and Interrelationships," in: *Advances in Protein Engineering*, pp.173- 315, Academic Press, 1991.

Pawlak, Z. *Rough Sets*, Kluwer Academic Publishers, Dordrecht, 1991.

Quinlan, J.R. *C4.5 - Programs for Machine Learning*, Morgan Kaufmann, CA, 1993.

Skowron, A. and Rauszer, C. "The Discerniblity Matrices and Functions in Information Systems," In Slowinski, R. (ed.) Intelligent Decision Support: Handbook of Applications and Advances of Rough Sets Theory, Kluwer, Dordrecht, 1992.

Tsumoto, K. et al. "Contribution to antibody-antigen interaction of structurally perturbed antigenic residues upon antibody binding", *Journal of Biological Chemistry*, **269**, 28777-28782, 1994.

Tsumoto, K. et al. "Role of Tyr residues in the contact region of anti-lysozyme monoclonal antibody Hwhel10 for antigen-binding", *Journal of Biological Chemistry*, **270**, 18551-18557, 1995.

Ziarko, W. "The Discovery,Analysis, and Representation of Data Dependencies in Databases," in: Shapiro,G.P. and Frawley, W.J. (eds.) *Knowledge Discovery in Database*, AAAI press, 1991.

Ziarko, W. and Shan, N., "A Rough Set-Based Method for Computing All Minimal Deterministic Rules in Attribute-Value Systems," Computational Intelligence **11**, 1995(in press).

Ziarko, W. and Shan, N. 1995b. "KDD-R: A Comprehensive System for Knowledge Discovery in Databases Using Rough Sets," *Proceedings of RSSC-94*, 1995(in press).

Zytkow, J.M. (Ed.) *Proceedings of the ML-92 Workshop on Machine Discovery (MD-92)*. Wichita, KS: National Institute for Aviation Research, 1992.

```
procedure PRIMEROSE − EX2
  var
    i : integer; /* Counter */
    M, L_i : List;
begin
  L_1 := {[a_i = v_j]|[x]_[a_i=v_j] ∩ D ≠ φ};
          /* a set of all the attribute-value pairs */
          /* [a_i = v_j](selectors in terms of AQ method) */
          /* such that α > 0. */
  i := 1;
  M := {};

  while ( i := 1 or M ≠ {} ) do
    begin
      while ( L_i ≠ {} ) do
        begin
          Select one pair R(= ∧[a_i = v_j]) from L_i;
          L_i := L_i − {R};
          if ( α_R = 1.0 and κ_R = 1.0 )
            then Save the quadruple as a Definite rule of d;
          if ( α_R > 0.5) then
            begin
              Check the p-value;
              if (p > 0.9), then Register the quadruple as a Significant rule of d;
              if (p > 0.5), then Register the quadruple as a Strong rule of d;
              else /* (p ≤ 0.5) */
                begin
                  I=nclude the quadruple in a list of Weak rules of d;
                  Append R to M (M := M + {R})
                end
            end
          else /* ( α ≤ 0.5) */
            begin
              Include the quadruple in a list of Weak rules of d;
              Append R to M (M := M + {R})
            end
        end
      i := i + 1;
      L_{i+1} := (a List of the whole combination of the conjunctive formulae in M)
    end
end {PRIMEROSE − EX2}
```

Figure 1: An Algorithm for PRIMEROSE-EX2

Application of Multistrategy Learning in Finance

M. Westphal and G. Nakhaeizadeh

SGZ-Bank, Karlsruhe, Karl-Friedrich Str. 23, D-76133 Karlsruhe, Germany
martin@pnn.sgz-bank.com
and
Daimler-Benz, Research and Technology, Postfach 2369, D-89013 Ulm, Germany
nakhaeizadeh@dbag.ulm.daimlerbenz.com

Abstract

Although one can find in literature some contributions reporting on application of Multistrategy Learning (MSL) in different domains, there are only few studies dealing with application of MSL in financial fields. This paper gives an overview about the possibilities of the application of MSL in finance. Presenting some recent empirical results achieved by the authors, we discuss some advantages of application of MSL to financial domains and suggest further research topics.

Introduction

The theoretical aspects of MSL are discussed in many publications, among them in Michalski and Tecuci (1991, 1993). There are also some works in the statistic community dealing with MSL (see for example Dasarathy, Sheela, 1979). The works of Wolpert (1992 a, 1992 b) and Henery (1996) are related to the context of MSL as well, although they use a horizontal combination of different approaches.

Furthermore, one can also find in literature some empirical evidence showing that the application of MSL can improve the results obtained by single approaches. In this connection, Esposito et al. (1993) discuss the application of MSL to document understanding, Sheppard (1993) applies MSL to classifying public health data and Hunter (1991) uses MSL to predicting protein structure.

Although in the literature there are several works applying the single-strategy learning in finance, only a few of them try to use the advantages of MSL. In this paper we present some of the works dealing with application of MSL in finance - which are certainly unknown to the MSL community - and based on interpretation of their results, we will make for some suggestions further research.

MSL approaches in finance

Prediction of exchange rates

The number of the works dealing with the prediction of exchange rates using a single forecasting approach is too high and we can not discuss all of them in this study. Interested readers can however find some of the recent works in Pesaran and Potter (1993), Refenes (1995) and Bol et al. (1996).

On the contrary, the number of the contributions that use MSL to forecasting of exchange rates is too small. In following we will discuss some of the recent works.

The central question addressed by Steurer (1996) is whether nonlinear methodologies can outperform econometric methods and the naive prediction, respectively. Therefore he conducts a performance comparison concerning the prediction of the monthly DM/US-Dollar exchange rate. Furthermore he examined the question whether a combination of the single prediction approaches can improve the performance. To develop his combined approaches he applies in a first step a cointegration study to find an adequate model for exchange rate prediction. This leads to a linear regression model that determines a long-run relationship between a set of nonstationary variables. Then the residues of this model, the error-correction term, together with other stationary and statistical important variables are used in a second step as explanatory variables for a linear regression forecasting models. Instead of the regression model of the second step, he uses also some machine learning approaches as alternatives, among them Neural Nets and M5-algorithm (Quinlan, 1992).

Steurer compares the results achieved by his MSL approach with the results of the random walk model and concludes that in several used versions the performance of the combined approach is superior to those of the random walk. The above MSL approach is used also in the work of Hann and Steurer (1996). In this study they emphasis on the short term prediction using weekly data. The authors conclude that their MSL approach should go a long way towards a more accurate prediction.

Prediction of stock prices

There are many single-strategy approaches to predict stock prices. Graf and Nakhaeizadeh (1994) and Cao and Tsay (1993) are examples of such studies. The number of used MSL approaches in this domain is, however, too low as

well. A comprehensive study using MSL approach for forecasting stock prices can be found in Westphal and Nakhaeizadeh (WN) (1996). This work is motivated by Quinlan (1993) who suggested the combination of model-based and instance-based approaches. Furthermore they use various single approaches.

Single approaches

Concerning the k-NN method WN discuss the different theoretical aspects, among them the various distance measures and the selection of the best value for *k*. Furthermore they discuss the runtime behavior of k-NN that is of interest in practical applications. In their empirical study WN implement a variation of Yuncks k-NN algorithm (Yunck, 1976). This new version avoids some disadvantages of the Yunck approach, especially concerning the selection of the start value and the choice of an appropriate step-size for the iteration.

In dealing with single approaches, the performance of classical k-NN methods is compared with those of IBL's. Further WN discuss some theoretical shortcomings of IBL approaches. Using an empirical example they show that the weighting strategy used by Aha, et al. (1991) might lead to wrong results. They discuss also that there is no significant difference between IB1 and a k-NN for k=1, because of this reason one can not claim that IB1 is a new algorithm.

Furthermore, IB2-algorithm belongs to the class of edited k-NN methods, discussed in detail in Dasarathy (1991). WN argue also, that in contrast to IB2 the algorithm of Chang (1974) leads to a lower number of prototypes and as a result to a better performance. It is also independent of the order of the used training data. It means that IB2 can lead to a case-base that is totally different although the same population is trained.

WN criticizes that in contrast to Wilson (1972) the power of IB3 in noise-reduction is unknown. Furthermore IB3 does not offer the possibility to classify correctly the cases located near the concepts-boundaries. This because IB3 eliminates the corresponding prototypes due to amount of points classified with these prototypes.

According to WN the different versions of IBL-algorithms only IB4 and IB5 can be regarded as new algorithms.

Another single approach used by WN is M5 algorithm. The most advantages of M5 is the usage of linear regression models in the nodes of the tree rather than the average values as for example in CART and NEWID.

The applied Neural Nets used by WN is a multi-layer perceptron net with a topology of 43 inputs, 7 hidden and 1 output neuron. The used learning algorithm is standard Backpropagation with a learning rate of 0.05 updated after each learning epoch. To these 43 explanatory variables belong some fundamental indicators, like the Dow Jones Index, the Nikkei Index or US$-DM exchange rate, which come from an econometric model and some technical indicators, like relative strength index, momentum or Williams R%. For each of the input series a set of 4 lags was used.

Combined methods

The work of Quinlan (1993) is extended by WN in different directions. On one hand they use a classical k-Nearest Neighbor (k-NN), which has a long tradition in Statistics, as additional instance-based method. On the other hand they use the average of the predictions of two or more different methods and apply the prediction made by one technique as additional input to an alternative prediction. For example, the prediction of the k-Nearest Neighbors can be considered as an additional attribute to train a Neural net. Another extension made by WN is the application of other versions of IBL-algorithms due to Aha, et al. (1991) and Aha (1992).
The IBL-Algorithms used in the empirical study are made available by David Aha and M5 by Ross Quinlan.

Empirical results

WN apply learning algorithms to 4 different forecasting tasks, the prediction of the change of the DAX (German Stock Index) one day ahead, the change in 3 and 5 days and the change during 5 days. The results were compared to a benchmark, the naive prediction that predicts that the value will be the same as last period.
To examine the performance of the single and combined approaches the future development of DAX is predicted. Besides the technical indicators the following explanatory variables are used as input:

- the Dow Jones index
- the Nikkei index
- the exchange rate between US $ and DM,
- the German floating rate

The target variable to be predicted is the daily changes of the DAX. The period from 01.01.1990 till 01.01.1993 was taken as the training set. The remaining data of the year 1993 formed the test set. All the time series are converted to logarithmic differences of two following days.
The correct forecast of the direction of a change in price is more important. Technical indicators were used as additional input. This has two advantages: First, not so many real time series are necessary for the test, because the technical indicators are estimated from the original series. Second, the forecasting method gains access to the tools common in the praxis of financial markets, even if these tools are not scientifically established. The "technical analysis", in contrast to the "fundamental analysis", does not attempt to explain the level or the development of a series. It tries instead to identify early symptoms for changes in the market. Table 1 gives an overview about the economic variables used in the study

of Westphal and Nakhaeizadeh (1996). The variable "Umlaufrendite" means long term average interest rate, RSI 5 and RSI 14 mean Relative Strength Index estimated for 5 and 14 days, respectively. OBOS stands for Over Bought Over Sold or "Williams R%".

Table 1: Time series used for prediction

fundamental	technical indicators
DOW	RSI 5
US $ / DM	RSI 14
Nikkei	Moving Average 12-26 days
Umlaufrendite	Momentum 1
DAX	Momentum 5
	Momentum 10
	OBOS 5
	OBOS 10
	OBOS 15
	OBOS 20
	Deviation for 2 days
	Deviation for 3 days
	Deviation for 4 days

Besides Mean Square Error (MSE), WN use accuracy rate as evaluation measure where the prediction was reduced to the statements the DAX will "rise" or "fall".

Some of the prediction results of single methods

WN observe that the k-NN algorithm yield a nearly constant and the achieved accuracy-rate remains independent from the forecasting horizon. Using the single approaches the best performance was obtained by the more simple versions of the IBL-algorithms, IB1 and IB2. These results were unexpected, because at least from the theoretical point of view IB4 and IB5 should lead to better results. An explanation for these unexpected results could be that the reduction of cases to prototypes by IB3-IB5 was connected with lost of information existing in the eliminated cases. Table 2 to 4 show the results for the DAX-prediction of the next day, 3rd day and 5th day, respectively. The best performance of the k-Nearest Neighbor method was achieved with k=5. The technical indicators had mostly a high significance. Especially including of the deviations was advantageous.

Table 2: Results for the DAX-prediction of the next day:

	Neural Net	M5	IBL1	k-NN	Naive Prediction
MSE	0.0106	0.0074	*	0.0090	0.0111
Accuracy-Rate	66.80 %	69.53 %	54.50 %	53.13 %	51.95 %

Table 3: The results achieved by the different methods for the DAX-prediction for the 3rd day:

	Neural Net	M5	IBL1	k-NN	Naive Prediction
MSE	0.0101	0.0087	*	0.0092	0.0111
Accuracy-Rate	57.03 %	47.27 %	45.80 %	54.30 %	51.56 %

Table 4: The results for the 5th day prediction:

	Neural Net	M5	IBL1	k-NN	Naive Prediction
MSE	0.0103	0.0087	*	0.0090	0.0113
Accuracy-Rate	55.08 %	46.09 %	46.00 %	52.73 %	51.56 %

* The IBL-Software does not allow to estimate this value.

The results of single methods are partially very good. The employed methods seem to be able to identify hidden structures in the data. An important difference between the methods is, that in contrast to the k-NN and IBL-algorithms the Neural Net and M5 are able to estimate values also outside the range of the target variable during the training process.

Using M5, the predictions were, however, not so volatile as the ones of the Neural Net. Furthermore, using the default version of M5 showed a tendency to predict a constant value for whole time. Dealing with the used Neural Net, it becomes obvious that the choice of the net input is of greater importance than the optimal architecture and complexity of the net. The pre-choice of the input time series highly determines the performance of the prediction.

Concerning k-NN, the parameters of the applied algorithms had to be adjusted individually for each task to achieve the best performance. This was not the case for the IBL-algorithms. The versions IB1 and IB2 mostly outperformed their further developed versions namely IB3 and IB4. In theory, IB3 should be better because of the filtering of noisy data and IB4 should have been the best because it learns not only the best prototypes but also the most important attributes for the task, confirmed by Aha, (1991) and (1992). The results of the IBL-algorithms were generally inferior compared to the results of the other methods.

Selected prediction results of MSL-approaches

Some of the results achieved by WN using the combined approaches are very interesting. Specially using the prediction of one method as additional input attribute for the Neural Net or M5. The results of the combination with the classical k-NN rule are presented in table 5.

Table 5: Neural Net and M5 get the prediction of the k-NN rule as additional input:

Sort of Prediction	NN with input from k-NN Accuracy	MSE	T*	M5 with input from k-NN Accuracy	MSE	T
1 Day	65.23%	0.0088	0.7965	64.06%	0.0076	0.6829
3. Day	55.47%	0.0107	0.9686	46.09%	0.0089	0.7879
5. Day	59.38%	0.0090	0.7987	45.70%	0.0089	0.7918
in 5 Days	66.80%	0.0205	1.7012	82.03%	0.0121	0.9902

*With T is Theil's coefficient: $T = \sqrt{\frac{\sum_t (x_t - \bar{x}_t)^2}{\sum_t (x_t - x_{t-1})^2}}$ with the prediction $\bar{x}_t$

The accuracy-rates of the values at the 5th day and in 5 days are the best of all. The results of the MSE are also very good. The values of T are also very well. The prediction of the level of the DAX in 5 days by the combination with the Neural Net is the only surpassed by the "naive prediction". This case is also the only one where the combination with M5 performs better than the one with the Neural Net. Examining of the significance of the inputs of the Neural Nets shows that the prediction of the k-NN method is an important input attribute for the Neural Net model. Furthermore the past values of the DAX and the Nikkei were important as well.

Another alternative method of combination used by WN is using the average of two predictions from different methods. This average is then the new prediction. Although this is a very simple combination method its results are very interesting because nearly all the combinations have an MSE lower than that of the single methods (See Westphal and Nakhaeizadeh 1996 for more details).

Conclusions and final remarks

Although application of the MSL in finance is a rather new applied research field, the results achieved up to now are very promising. Generally, prediction of the development of financial time series is not an easy task, because the structure of many of financial time series does not differ significantly from a random walk process. An additional problem is changing the structure of data over the time. The motivation behind the application of MSL to finance has been to use the advantages of the alternative approaches to improve the forecasting results. It means, the alternative approaches are considered not as competitors but they are used in a cooperatively manner.

The empirical results reported in these works encourage further research in this area. Specially the application of the methodology used in Westphal and Nakhaeizadeh (1996) could be interesting for forecasting of the other financial time series, for example, exchange and interest rates. The predictability of the exchange rates is still an open and controversy problem. Most of the applied approaches to solve this problem are based on the statistical methods. It would be quite interesting to examine weather combination of such statistical approaches with AI-based methods within a MSL concept can contribute significantly to this controversy debate.

Another open problem is the adaptation of forecasting approaches as the structure of data changes over the time. Although there are many single-methods which examine the so called „structural change" in time series, there is no comprehensive work examining the practicability of MSL-approach to this domain. Further research in this connection would be quite interesting as well.

Acknowledgments: The authors would like to thank David Aha and Ross Quinlan for providing IBL- and M5-Software.

References

Aha, D. W., 1992. Tolerating noisy, irrelevant and novel attributes in instance-based learning algorithms. International Journal of Man-Machine Studies, 267-287, Vol 36.

Aha, D. W., Kibler, D., Albert, M. K. 1991. Instance-Based Learning Algorithms. Machine Learning, 37-666,. Kluwer Academic Publishers, Boston.

Bol, G. Nakhaeizadeh, G. and Vollmer, K. H. 1996. (Eds.). Finanzmarktanalyse und -Prognose mit innovativen quantitativen Verfahren. Physica-Verlag, Berlin.

Cao, C. Q. and Tsay, R. S. 1993. Nonlinear Time-Series Analysis of Stock Volatilities. In: Nonlinear Dynamics Chaos and Econometric, 157-178, edited by Pesaran, M. H. and Potter, S. M. Wiley.

Chang, C. L., 1974. Finding Prototypes for Nearest Neighbor Classifiers. IEEE Transactions on Computers, Volume C-23, No. 11, 1179-1184. Reprinted in Dasarathy, 1991.

Dasarathy, B. V., (Ed.), 1991. NN Pattern Classification Techniques. IEEE Computer Society Press, Los Alamitos.

Dasarathy, B. and Sheela, B. (1979). A. Composite Classifier System Design: Concept and Methodology. Proceedings of the IEEE, 708-713, Volume 67, Number 5.

Esposito, F. Malerba, d. Semeraro, G and Pazzani, M. 1993. A machine Learning Approach to Document Understanding. Proceedings of the second International Workshop on Multistrategy Learning, 276-292. Center of Artificial Intelligence, George Mason University

Graf, J., and Nakhaeizadeh, G. 1994. Application of Learning Algorithms to Predicting Stock Prices. In: Frontier Decision Support Concepts, 241-260, edited by Plantamura, V. Soucek, B. and Visaggio, Wiley.

Hann, T. and Steurer, E. 1996. Much ado about nothing ? Exchange rate forecasting: Neural Networks vs. linear models using monthly and weekly data, 1-17. Neurocomputing.

Henery, R. 1996. Combination Forecasting Procedures. Forthcoming.

Hunter, L. 1993, Classifying for Prediction: A multistrategy Approach to Predicting Protein Structure, Proceedings of the second International Workshop on Multistrategy Learning, 394-402. Center of Artificial Intelligence, George Mason University.

Michalski, R. and Tecuci, G. Eds., 1993. Proceedings of the second International Workshop on Multistrategy Learning, Center of Artificial Intelligence, George Mason University.

Michalski, R. and Tecuci, G. Eds. 1991. Proceedings of the first International Workshop on Multistrategy Learning, Center of Artificial Intelligence, George Mason University.

Pesaran, M. H. and Potter, S. M. Eds. 1993. Nonlinear Dynamics Chaos and Econometrics, Wiley

Quinlan, J. R., 1993. Combining Instance-Based and Model-Based Learning. Proceedings of the International Conference of Machine Learning, 236-243, Morgan Kaufmann.

Quinlan, J. R., 1992. Learning with Continuos Classes. Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, 343-348. Singapore: World Scientific.

Refenes, A. (Ed.), 1995. Neural Networks in the Capital Markets, Wiley, New York.

Steurer, E. 1996. Ökonometrische Methoden und maschinelle Lernverfahren zur Wechselkursprognose: Theoretische Analyse und empirischer Vergleich. Ph. D. diss. University of Karlsruhe, Germany.

Sheppard J. 1993. Applying Multiple Learning Strategies for Classifying Public Health Data. In: Proceedings of the second International Workshop on Multistrategy Learning, 309-323. Center of Artificial Intelligence, George Mason University

Westphal, M. and Nakhaeizadeh, G. 1996. Combination of Statistical and other learning methods to predict financial time series. Discussion Paper, Daimler-Benz Research Center Ulm, Germany

Wilson, D. L., 1972. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Man and Cybernetics, 408-421, Volume SMC-2, No 3. Reprinted in Dasarathy, 1991.

Wolpert, D. 1992a. Stacked Generalization. Neural Network, 241-259, Vol. 5.

Wolpert, D. 1992b. Horizontal Generalization. Working Paper, Santa Fe Institute, 92-07-033.

Yunck, T. P. 1976. A Technique to Identify Nearest Neighbors. IEEE Transactions on Systems, Man and Cybernetics, 678-683, Volume SMC-6, No 10. Reprinted in Dasarathy, 1991.

Author Index

Nabil W. Alkharouf 115
Scott Atran 71
Daniel Billsus 239
Marco Botta 125
Peter Brockhausen 17
John D. Coley 71
Michael T. Cox 135
Luc De Raedt 29
Hugo De Garis 251
Pedro Domingos 147
John F. Elder IV 53
Tara A. Estlin 271
Qiu Fan 229
Attilio Giordana 125
Diana Gordon 95
Lawrence Hunter 85
Takashi Ishikawa 295
David B. Leake 155
Elizabeth B. Lynch 71
Kenneth Kaufman 305
Andrew Kinley 155
Douglas L. Medin 71
Ryszard S. Michalski 115
Raymond J. Mooney 271
Katharina Morik 17
Gholamreza Nakhaeizadeh 333
Stefan Ohl 281
Michael Pazzani 239
Lech Polkowski 57
Jean-Francois Puget 165
Mukesh Rohatgi 103
Paul S. Rosenbloom 39
Celine Rouveirol 165
Lorenza Saitta 3
Claude Sammut 11
Michele Sebag 165
Andrzej Skowron 57
Benjamin D. Smith 39
Edgar Sommer 177
Devika Subramanian 95
Hiroshi Tanaka 313,325
Takao Terano 295
Kai Ming Ting 191
Shusaku Tsumoto 313,325
Xuemei Wang 203
Martin Westphal 333
Gerhard Widmer 217
David Wilson 155
Jianping Zhang 229