CONTENTS

Preface ................................................................................................................................. ix

COMBINING EMPIRICAL AND EXPLANATION-BASED LEARNING

Unifying Themes in Empirical and Explanation-Based Learning................................. 2
Pat Langley

Induction Over the Unexplained: Integrated Learning of Concepts with Both Explainable
and Conventional Aspects................................................................................................. 5
Raymond Mooney and Dirk Ourston

Conceptual Clustering of Explanations........................................................................... 8
Jungsoon P. Yoo and Douglas H. Fisher

A Tight Integration of Deductive and Inductive Learning........................................... 11
Gerhard Widmer

Multi-Strategy Learning in Nonhomogeneous Domain Theories............................... 14
Gheorghe Tecuci and Yves Kodratoff

A Description of Preference Criterion in Constructive Learning: A Discussion of Basic Issues............. 17
Jianping Zhang and Ryszard S. Michalski

Combining Case-Based Reasoning, Explanation-Based Learning, and Learning from Instruction...... 20
Michael Redmond

Deduction in Top-Down Inductive Learning.................................................................. 23
F. Bergadano, A. Giordana and S. Ponsero

One-Sided Algorithms for Integrating Empirical and Explanation-Based Learning.......... 26
Wendy E. Sarrett and Michael J. Pazzani

Combining Empirical and Analytical Learning with Version Spaces............................ 29
Haym Hirsh

Finding New Rules for Incomplete Theories: Explicit Biases for Induction with Contextual
Information....................................................................................................................... 34
Andrea Pohoreckyj Danyluk

Learning from Plausible Explanations............................................................................ 37
Tom E. Fawcett

Augmenting Domain Theory for Explanation-Based Generalisation............................ 40
Kamal M. Ali

Explanation Based Learning as Constrained Search....................................................... 43
David Haines

Reducing Search and Learning Goal Preferences......................................................... 46
Steven Morris

Adaptation-Based Explanation: Explanations as Cases.................................................. 49
Alex Kass

A Retrieval Model Using Feature Selection.................................................................. 52
Colleen M. Seifert

Improving Decision-Making on the Basis of Experience.............................................. 55
Bruce Krulaich, Gregg Collins and Lawrence Birnbaum

Explanation-Based Acceleration of Similarity-Based Learning................................... 58
Masayuki Numao and Masamichi Shimura
Contents

Knowledge Acquisition Planning: Results and Prospects
Lawrence Hunter

"Learning by Instruction" in Connectionist Systems
Joachim Diederich

Integrating Learning in a Neural Network
Bruce F. Katz

Explanation-Based Learning with Weak Domain Theories
Michael J. Pazzani

Using Domain Knowledge to Improve Inductive Learning Algorithms for Diagnosis
Gerhard Friedrich and Wolfgang Nejdl

A Framework for Improving Efficiency and Accuracy
James Wogulis

Error Correction in Constructive Induction
George Drastal, Regine Meunier and Stan Raatz

Improving Explanation-Based Indexing with Empirical Learning
Ralph Barletta and Randy Kerber

A Schema for an Integrated Learning System
Michael Wollowski

Combining Explanation-Based Learning and Artificial Neural Networks
Jude W. Shavlik and Geoffrey G. Towell

EMPIRICAL LEARNING: THEORY AND APPLICATION

Learning Classification Rules Using Bayes
Wray Buntine

New Empirical Learning Mechanisms Perform Significantly Better in Real Life Domains
Matjaz Gams and Aram Karalic

Inductive Learning with BCT
Philip K. Chan

What Good Are Experiments?
Ritchey A. Ruff and Thomas G. Dietterich

An Experimental Comparison of Human and Machine Learning Formalisms
Stephen Muggleton, Michael Bain, Jean Hayes-Michie and Donald Michie

Two Algorithms That Learn DNF by Discovering Relevant Features
Giulia Pagallo and David Haussler

Limitations on Inductive Learning
Thomas G. Dietterich

The Induction of Probabilistic Rule Sets — The Irule Algorithm
Rodney M. Goodman and Padhraic Smyth

Empirical Substructure Discovery
Lawrence B. Holder

Learning the Behavior of Dynamical Systems from Examples
Jan Paredis

Experiments in Robot Learning
Matthew T. Mason, Alan D. Christiansen and Tom M. Mitchell

Induction of Decision Trees from Inconclusive Data
Scott Spangler, Usama M. Fayyad and Ramasamy Uthurusamy

Knowledge Intensive Induction
Michel Manago
An Ounce of Knowledge is Worth a Ton of Data: Quantitative Studies of the Trade-Off Between Expertise and Data Based On Statistically Well-Founded Empirical Induction
Brian R. Gaines

Signal Detection Theory: Valuable Tools for Evaluating Inductive Learning
Kent A. Spackman

Unknown Attribute Values in Induction
J. R. Quinlan

Processing Issues in Comparisons of Symbolic and Connectionist Learning Systems
Douglas Fisher, Kathleen McKusick, Raymond Mooney, Jude W. Shavlik and Geoffrey Towell

Bacon, Data Analysis and Artificial Intelligence
Cullen Schaffer

LEARNING PLAN KNOWLEDGE

Learning to Plan in Complex Domains
David Ruby and Dennis Kibler

An Empirical Analysis of EBL Approaches for Learning Plan Schemata
Jude W. Shavlik

Learning Decision Rules for Scheduling Problems: A Classifier Hybrid Approach
M. R. Hilliard, G. Liepins, G. Rangarajan and M. Palmer

Learning Tactical Plans for Pilot Aiding
Keith R. Levi, David Perschbacher and Valerie L. Shalitin

Issues in the Justification-Based Diagnosis of Planning Failures
Lawrence Birnbaum, Gregg Collins and Bruce Krulwich

Learning Procedural Knowledge in the EBG Context
Stan Matwin and Johanne Morin

Learning Invariants from Explanations
Jean-François Puget

Using Learning to Recover Side-Effects of Operators in Robotics
Ralph P. Sobek and Jean-Paul Laumond

Learning to Recognize Plans Involving Affect
Paul O'Rorke, Timothy Cain and Andrew Ortony

Learning to Retrieve Useful Information for Problem Solving
Randolph Jones

Discovering Problem Solving Strategies: What Humans Do and Machines Don't (Yet)
Kurt VanLehn

Approximating Learned Search Control Knowledge
Melissa P. Chase, Monte Zweben, Richard L. Piazza, John D. Burger, Paul P. Maglio and Haym Hirsh

Planning in Games Using Approximately Learned Macros
Prasad Tadepalli

Learning Approximate Plans for Use in the Real World
Scott W. Bennett

Using Concept Hierarchies to Organize Plan Knowledge
John A. Allen and Pat Langley

Conceptual Clustering of Mean-Ends Plans
Hua Yang and Douglas H. Fisher

Learning Appropriate Abstractions for Planning in Formation Problems
Nicholas S. Flann
<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discovering Admissible Search Heuristics by Abstracting and Optimizing</td>
<td>240</td>
</tr>
<tr>
<td>Jack Mostow and Armand E. Prieditis</td>
<td></td>
</tr>
<tr>
<td>Learning Hierarchies of Abstraction Spaces</td>
<td>241</td>
</tr>
<tr>
<td>Craig A. Knoblock</td>
<td></td>
</tr>
<tr>
<td>Learning from Opportunity</td>
<td>246</td>
</tr>
<tr>
<td>Tim Converse, Kristian Hammond and Mitchell Marks</td>
<td></td>
</tr>
<tr>
<td>Learning by Analyzing Fortuitous Occurrences</td>
<td>249</td>
</tr>
<tr>
<td>Steve A. Chien</td>
<td></td>
</tr>
<tr>
<td>Explanation-Based Learning of Reactive Operators</td>
<td>252</td>
</tr>
<tr>
<td>Melinda T. Gervasio and Gerald F. Dejong</td>
<td></td>
</tr>
<tr>
<td>On Becoming Reactive</td>
<td>255</td>
</tr>
<tr>
<td>Jim Blythe and Tom M. Mitchell</td>
<td></td>
</tr>
<tr>
<td>KNOWLEDGE-BASE REFINEMENT AND THEORY REVISION</td>
<td></td>
</tr>
<tr>
<td>Knowledge Base Refinement and Theory Revision</td>
<td>260</td>
</tr>
<tr>
<td>Allen Ginsberg</td>
<td></td>
</tr>
<tr>
<td>Theory Formation by Abduction: Initial Results of a Case Study Based on the Chemical Revolution</td>
<td>266</td>
</tr>
<tr>
<td>Paul O'Rorke, Steven Morris and David Schulenburg</td>
<td></td>
</tr>
<tr>
<td>Using Domain Knowledge to Aid Scientific Theory Revision</td>
<td>272</td>
</tr>
<tr>
<td>Donald Rose</td>
<td></td>
</tr>
<tr>
<td>The Role of Experimentation in Scientific Theory Revision</td>
<td>278</td>
</tr>
<tr>
<td>Deepak Kulkarni and Herbert A. Simon</td>
<td></td>
</tr>
<tr>
<td>Exemplar-Based Theory Rejection: An Approach to the Experience Consistency Problem</td>
<td>284</td>
</tr>
<tr>
<td>Shankar A. Rajamoney</td>
<td></td>
</tr>
<tr>
<td>Controlling Search for the Consequences of New Information During Knowledge Integration</td>
<td>290</td>
</tr>
<tr>
<td>Kenneth S. Murray and Bruce W. Porter</td>
<td></td>
</tr>
<tr>
<td>Identifying Knowledge Base Deficiencies by Observing User Behavior</td>
<td>296</td>
</tr>
<tr>
<td>Keith R. Levi, Valerie L. Shalin and David L. Perschbacher</td>
<td></td>
</tr>
<tr>
<td>Toward Automated Rational Reconstruction: A Case Study</td>
<td>302</td>
</tr>
<tr>
<td>Chris Tong and Phil Franklin</td>
<td></td>
</tr>
<tr>
<td>Discovering Mathematical Operator Definitions</td>
<td>308</td>
</tr>
<tr>
<td>Michael H. Sims and John L. Bresina</td>
<td></td>
</tr>
<tr>
<td>Imprecise Concept Learning Within a Growing Language</td>
<td>314</td>
</tr>
<tr>
<td>Zbigniew W. Ras and Maria Zemankova</td>
<td></td>
</tr>
<tr>
<td>Using Determinations in EBL: A Solution to the Incomplete Theory Problem</td>
<td>320</td>
</tr>
<tr>
<td>Sridhar Mahadevan</td>
<td></td>
</tr>
<tr>
<td>Some Results on the Complexity of Knowledge Base Refinement</td>
<td>326</td>
</tr>
<tr>
<td>Marco Valtoria</td>
<td></td>
</tr>
<tr>
<td>Knowledge Base Refinement as Improving an Incorrect, Inconsistent and Incomplete Domain Theory</td>
<td>332</td>
</tr>
<tr>
<td>David C. Wilkins and Kok-Wah Tan</td>
<td></td>
</tr>
<tr>
<td>INCREMENTAL LEARNING</td>
<td></td>
</tr>
<tr>
<td>Incremental Learning of Control Strategies with Genetic Algorithms</td>
<td>340</td>
</tr>
<tr>
<td>John J. Grefenstette</td>
<td></td>
</tr>
<tr>
<td>Tower of Hanoi with Connectionist Networks: Learning New Features</td>
<td>345</td>
</tr>
<tr>
<td>Charles W. Anderson</td>
<td></td>
</tr>
</tbody>
</table>
A Formal Framework for Learning in Embedded Systems ................................................................. 350  
Leslie Pack Kaelbling

A Role for Anticipation in Reactive Systems that Learn ................................................................. 354  
Steven D. Whitehead and Dana H. Ballard

Uncertainty Based Selection of Learning Experiences ................................................................. 358  
Paul D. Scott and Shaul Markovitch

Improved Training Via Incremental Learning ............................................................................... 362  
Paul E. Utgoff

Incremental Batch Learning ........................................................................................................... 366  
Scott H. Clearwater, Tze-Pin Cheng, Haym Hirsh and Bruce G. Buchanan

Incremental Concept Formation with Composite Objects .......................................................... 371  
Kevin Thompson and Pat Langley

Using Multiple Representations to Improve Inductive Bias: Gray and Binary Coding for Genetic Algorithms ......................................................................................................................... 375  
Richard A. Caruana, J. David Schaff er and Larry J. Eshelman

Focused Concept Formation ............................................................................................................ 379  
John H. Gennari

An Exploration Into Incremental Learning: The INFLUENCE System .................................................. 383  
Antoine Cornuejols

Incremental, Instance-Based Learning of Independent and Graded Concept Descriptions .............. 387  
David W. Aha

Cost-Sensitive Concept Learning of Sensor Use in Approach and Recognition ......................... 392  
Ming Tan and Jeffrey C. Schlimmer

Reducing Redundant Learning ........................................................................................................ 396  
Joel D. Martin

Incremental Clustering by Minimizing Representation Length ...................................................... 400  
Jakub Segen

Information Filters and Their Implementation in the SYLLOG System ............................................ 404  
Shaul Markovitch and Paul D. Scott

Adaptive Learning of Decision-Theoretic Search Control Knowledge ............................................. 408  
Eric H. Wefald and Stuart J. Russell

Atoms of Learning II: Adaptive Strategies A Study of Two- Person Zero-Sum Competition ............. 412  
Oliver Selfridge

An Incremental Genetic Algorithm for Real-Time Learning ........................................................... 416  
Terence C. Fogarty

Participatory Learning: A Constructivist Model ............................................................................... 420  
Ronald R. Yager and Kenneth M. Ford

REPRESENTATIONAL ISSUES IN MACHINE LEARNING

Representational Issues in Machine Learning .................................................................................. 426  
Devika Subramanian

Labor Saving New Distinctions ......................................................................................................... 430  
John Woodfill

A Theory of Justified Reformulations ............................................................................................... 434  
Devika Subramanian

Reformulation from State Space to Reduction Space ......................................................................... 439  
Patricia J. Riddle
<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge-Based Feature Generation</td>
<td>441</td>
</tr>
<tr>
<td>James P. Callan</td>
<td></td>
</tr>
<tr>
<td>Enriching Vocabularies by Generalizing Explanation Structures</td>
<td>444</td>
</tr>
<tr>
<td>Richard Maclin and Jude W. Shavlik</td>
<td></td>
</tr>
<tr>
<td>Higher-Order and Modal Logic as a Framework for Explanation-Based Generalization</td>
<td>447</td>
</tr>
<tr>
<td>Scott Dietzen and Frank Pfenning</td>
<td></td>
</tr>
<tr>
<td>Towards a Formal Analysis of EBL</td>
<td>450</td>
</tr>
<tr>
<td>Russell Greiner</td>
<td></td>
</tr>
<tr>
<td>A Mathematical Framework for Studying Representation</td>
<td>454</td>
</tr>
<tr>
<td>Robert C. Holte and Robert M. Zimmer</td>
<td></td>
</tr>
<tr>
<td>Refining Representations to Improve Problem Solving Quality</td>
<td>457</td>
</tr>
<tr>
<td>Jeffrey C. Schlimmer</td>
<td></td>
</tr>
<tr>
<td>Comparing Systems and Analyzing Functions to Improve Constructive Induction</td>
<td>461</td>
</tr>
<tr>
<td>Larry Rendell</td>
<td></td>
</tr>
<tr>
<td>Evaluating Alternative Instance Representations</td>
<td>465</td>
</tr>
<tr>
<td>Sharad Saxena</td>
<td></td>
</tr>
<tr>
<td>Evaluating Bias During Pac-Learning</td>
<td>469</td>
</tr>
<tr>
<td>Lonnie Chrisman</td>
<td></td>
</tr>
<tr>
<td>Constructing Representations Using Inverted Spaces</td>
<td>472</td>
</tr>
<tr>
<td>Pankaj Mehra</td>
<td></td>
</tr>
<tr>
<td>A Constructive Induction Framework</td>
<td>474</td>
</tr>
<tr>
<td>Christopher J. Matheus</td>
<td></td>
</tr>
<tr>
<td>Constructive Induction by Analogy</td>
<td>476</td>
</tr>
<tr>
<td>Luc De Raedt and Maurice Bruynooghe</td>
<td></td>
</tr>
<tr>
<td>Concept Discovery Through Utilization of Invariance Embedded in the Description Language</td>
<td>478</td>
</tr>
<tr>
<td>Mieczyslaw M. Kokar</td>
<td></td>
</tr>
<tr>
<td>Declarative Bias for Structural Domains</td>
<td>480</td>
</tr>
<tr>
<td>Benjamin N. Gros оф and Stuart J. Russell</td>
<td></td>
</tr>
<tr>
<td>Automatic Construction of a Hierarchical Generate-and-Test Algorithm</td>
<td>483</td>
</tr>
<tr>
<td>Sunil Mohan and Chris Tong</td>
<td></td>
</tr>
<tr>
<td>A Knowledge-Level Analysis of Informing</td>
<td>485</td>
</tr>
<tr>
<td>Jane Yung-jen Hsu</td>
<td></td>
</tr>
<tr>
<td>An Object-Oriented Representation for Search Algorithms</td>
<td>489</td>
</tr>
<tr>
<td>Jack Mostow</td>
<td></td>
</tr>
<tr>
<td>Compiling Learning Vocabulary from a Performance System Description</td>
<td>492</td>
</tr>
<tr>
<td>Richard M. Keller</td>
<td></td>
</tr>
<tr>
<td>Generalized Recursive Splitting Algorithms for Learning Hybrid Concepts</td>
<td>496</td>
</tr>
<tr>
<td>Bruce Lambert, David Tcheng and Stephen C-Y Lu</td>
<td></td>
</tr>
<tr>
<td>Screening Hypotheses with Explicit Bias</td>
<td>499</td>
</tr>
<tr>
<td>Diana Gordon</td>
<td></td>
</tr>
<tr>
<td>Building A Learning Bias from Perceived Dependencies</td>
<td>501</td>
</tr>
<tr>
<td>Ch. de Sainte Marie</td>
<td></td>
</tr>
<tr>
<td>A Bootstrapping Approach to Conceptual Clustering</td>
<td>503</td>
</tr>
<tr>
<td>Katharina Morik and Joerg-Uwe Kietz</td>
<td></td>
</tr>
<tr>
<td>Overcoming Feature Space Bias in a Reactive Environment</td>
<td>505</td>
</tr>
<tr>
<td>Hans Tallis</td>
<td></td>
</tr>
</tbody>
</table>

Author Index ........................................................................... 509
This volume contains the written versions of papers presented at the Sixth International Workshop of Machine Learning held at Cornell University, Ithaca, New York (USA) in June of 1989. Since the first of these workshops was held at Carnegie Mellon University in July of 1980, the field has witnessed a steady growth in the number of active researchers. In 1988 an open conference, held at the University of Michigan, supplanted the normally invitation-only workshop and succeeded in attracting over 320 participants!

In an effort to strike a balance between the increased breadth of larger meetings and the intimacy of earlier workshops, this edition of the workshop was organized as a set of six independent study groups. Each group consisted of about 50 participants and was devoted to a particular issue in machine learning. Plenary sessions were held for invited lectures, and some study group chairs elected to hold joint sessions where topics of interest to both groups could be discussed. Over 350 researchers were invited, with 130 papers presented during the three day workshop.

The workshop would not have been possible without the help of many talented individuals. The organizing committee provided invaluable assistance and the collective wisdom of five previous workshops. They were responsible for selecting topics, choosing study group chairs, and picking invited speakers. The study group chairs, who collectively comprise the program committee, spent countless hours reviewing submissions and leading the discussion at workshop sessions. The invited speakers (Tom Dietterich, J. Ross Quinlan, and Bob Simpson) prepared intellectually stimulating lectures to plenary sessions.

Funding for the conference was provided by both the Artificial Intelligence and Robotics Program and the Cognitive Science Program of the Office of Naval Research, by the Knowledge Models and Cognitive Systems Program of the National Science Foundation, and by the Cornell University Department of Computer Science. We are grateful for their financial backing. Morgan Kaufmann Publishers made these proceedings available to participants at cost. I would like to thank Shirley Jowell of Morgan Kaufmann for her excellent editorial and technical advice.

I am very grateful to Lesley Fredericks of Cornell University Conference Services who handled the local arrangements, and to Vicki Feinberg, Michele Drier and Debbie Smith of the Cornell University Computer Science Department for providing staff support for the workshop. I would also like to thank all of my students, especially Wendy Amai for coordinating the student volunteers and Jennifer Turney for providing the ML89 audio puzzle.

Alberto Maria Segre
Cornell University
Ithaca, New York (USA)
This page intentionally left blank
Combining Empirical and Explanation-Based Learning
Unifying Themes in Empirical and Explanation-Based Learning

Pat Langley

Department of Information & Computer Science, University of California, Irvine, CA 92717 USA

The Need for Unified Theories of Learning

A central activity of science is the search for unifying principles that account for apparently diverse phenomena within a single framework. However, recent work in machine learning has tended to emphasize the differences between learning methods. In this paper, I argue that two of the major paradigms - induction and explanation-based learning - are more similar than the literature suggests, and that we must focus on these similarities before we can build a unified theory of learning mechanisms.

Significant differences certainly exist between explanation-based and empirical methods, but the perceived chasm is far greater than the actual one. This perception has resulted partly from a literature that abounds with rhetorical statements claiming superiority of one method over another. Other causes for the perceived distinction include divergent notations and different measures of performance, which hide the underlying similarity of mechanisms and tasks. In this paper, I present examples of misleading rhetoric and conflicting metrics that the field must overcome before it can approach a unified theory of learning.

Learning from One Instance and Many Instances

One common claim is that empirical methods require many instances to learn, whereas EBL can learn from a single instance (e.g., Mitchell, Keller, & Kedar-Cabelli, 1986, pp. 47–48). This misleading statement probably results from comparisons between explanation-based methods (which are typically incremental) and nonincremental induction methods, such as Quinlan’s (1986) ID3. However, if one examines incremental inductive methods, such as Fisher’s (1987) COBWEB, the true situation becomes apparent. Any incremental approach to induction (even neural networks) can learn something from a single instance, though it may not learn as much as an EBL technique.

The above claim also suggests that EBL methods can learn everything they need to know from a single instance, but this is clearly false as well. Analytic techniques require one instance for each proof structure they compile. For example, Pazzani’s (1988) OCCAM acquires four schemata for recognizing when economic sanctions will fail and three schemata for predicting when they will succeed; thus, it requires not one training instance for this domain, but seven. Although EBL techniques may learn more rapidly than empirical methods, this is a difference in learning rate, not a difference between one and many instances.

Learning With and Without Search

A second popular belief is that empirical methods require extensive search, whereas explanation-based methods can learn without search. Again, this statement is misleading on two fronts. First, it focuses on inductive methods like Mitchell’s (1982) version-space algorithm, which use memory-intensive search techniques to consider competing hypotheses. However, many inductive methods rely on memory-limited methods such as greedy algorithms (Quinlan, 1986) and incremental hill climbing (Fisher, 1987). Although such methods operate within a space of hypotheses, they do not ‘search’ in the usual sense of this term.

On the other hand, if one views the explanation process as a component of learning (rather than as performance), then EBL itself can involve extensive search through the space of explanations. Work in this paradigm has not emphasized this search because, to date, most tests have involved relatively small domain theories. In addition, one goal of EBL is to improve efficiency, and Minton (1988) has shown that adding
compiled rules to the knowledge base sometimes produces just the opposite effect. To deal with this issue, his PRODIGY system computes statistics for learned rules, deleting those that are not worth retaining. One can view this process as search through a space of compiled rules, just as empirical methods search a space of induced rules. Whether one labels either activity as 'search' is less important than the realization that both frameworks must deal with large rule spaces.

Learning With and Without Domain Knowledge

Yet another claim is that explanation-based methods take domain knowledge into account during learning, whereas empirical methods are knowledge free (e.g., Mitchell et al., 1986, p. 48). The first part of this statement is true enough, but the second half ignores the fact that any incremental induction system inevitably changes its knowledge level over time. After such a system has seen \( n \) instances, it will process instance \( n + 1 \) differently than if it had seen it first. For example, Fisher's (1987) COBWEB constructs a concept hierarchy that organizes instances it has encountered, and the structure of this memory influences not only the predictions it makes on new instances, but the learning that occurs. Thus, COBWEB takes advantage of domain knowledge to direct the learning process. The fact that it acquires this knowledge itself (rather than receiving it from the programmer) makes it no less knowledge intensive.

As another example, consider Wolff's (1982) SNPR algorithm, which is generally viewed as lying at the extreme end of the tabula rasa spectrum. This system accepts a sequence of letters as input, and carries out a hill-climbing search through the space of phrase-structure grammars, using two basic operators. The first notes frequently occurring sequences of symbols and defines new 'chunks', which correspond to words and phrases. The second learning operator notes when sets of symbols tend to occur in the same context (i.e., next to a common symbol); this defines new disjunctive classes, which correspond to parts of speech and alternative forms of phrases.

If one looks only at the relation between SNPR's inputs and outputs, it appears to be the prototypical 'knowledge free' induction system. However, the algorithm is semi-incremental, in that it processes only part of its input at a given time, using the knowledge it gains from earlier data in processing its later experience. Specifically, SNPR constructs a partial grammar to summarize the letter sequences it has observed, and it uses this grammar to rewrite new strings at a higher level of description (i.e., using nonterminal symbols in the grammar). One can view this activity as constructing partial explanations of the input, and one can view the later stages of grammar induction as a form of knowledge-intensive learning that involves extending an incomplete domain theory (the set of grammar rules). Although phrase-structure grammars are a constrained form of domain theory, they are very similar in structure to those used by many EBL systems.

Justified and Unjustified Learning

A fourth claim is that explanation-based methods are justified, whereas empirical learning is inherently unjustified (e.g., Mitchell et al., 1986, p. 48). The latter statement is clearly true, since empirical learning involves an inductive leap from instances to general rules. However, the justified nature of EBL is not so clear. Rules generated by analytic methods are guaranteed to be as accurate as the original domain theory, since the deductive closure does not change. However, they may not be as efficient as the original rule set. The common assumption that EBL will improve efficiency is based on the belief that training and test instances will follow similar distributions. Thus, analytic methods make an inductive leap with respect to efficiency that is no more justified than the leap made by empirical methods regarding accuracy.

In addition, one can extend the basic explanation-based learning framework to domains in which the inference rules, rather than being deductively valid, are plausible or probabilistic. In such domains, the process of compiling multi-step explanations may generate 'bad' inference rules that have very low predictive ability, since transitivity does not hold for probabilistic inference chains as it does for deductive chains. In
such an extended framework, analytic learning methods are not even justified with respect to predictive accuracy. Given a reasonably accurate domain theory, such methods may still lead to more rapid learning, but they are not any more ‘correct’ than inductive methods.

Accuracy and Efficiency in Machine Learning

The term learning suggests some change in performance, and the empirical and explanation-based communities have been further divided by their concern with different performance measures. Most research on induction has focused on improving predictive accuracy, whereas most analytical work has (implicitly if not explicitly) focused on efficiency. However, both measures of performance have an important role to play in both approaches to learning.

For example, any performance system has limited memory size and processing time; thus, adding rules that reduce memory load or increase speed can let one finish complex tasks that were impossible before learning. This means that EBL can produce improvements in predictive accuracy, and suggests that researchers should measure it in future studies. Similarly, any induction system that deals with a complex domain will create many different concepts. If organized ineffectively, this acquired knowledge may drastically slow the performance system. This means that retrieval time is a central issue in empirical learning, and that induction researchers should examine this performance measure as well.

As work in both paradigms starts to bridge this gap, it may reveal previously unsuspected connections between induction and EBL. For instance, psychological studies suggest that humans recognize certain basic-level categories more rapidly than other concepts. Fisher’s (1987) COBWEB/2 – an empirical learning system – models this effect with a mechanism that creates direct indices to some nodes in its concept hierarchy and that bypasses other concepts. In spirit, this operation is remarkably similar to the caching process by which many EBL methods store operationalized definitions of concepts to improve retrieval efficiency.

Like the examples in previous sections, this connection suggests that empirical and explanation-based methods have much more in common than the literature leads one to expect. If researchers in the two paradigms can rise above the rhetoric and assumptions that have kept them apart, they can move together toward a unified science of machine learning that incorporates insights from both frameworks.

References


INDUCTION OVER THE UNEXPLAINED:  
Integrated Learning of Concepts with Both Explainable and Conventional Aspects*

Raymond Mooney  Dirk Ourston  
Department of Computer Sciences  
University of Texas  
Austin, TX 78712

ABSTRACT

This paper presents a new approach to combining explanation-based and empirical learning called Induction Over the Unexplained (IOU). Unlike other approaches to integrated learning, which use one method to focus the other or provide it with information, IOU uses each method to learn a different part of the final concept definition. It is therefore suited for learning concepts with both explainable and unexplainable aspects. An initial nonincremental feature-based implementation of IOU is presented together with an example illustrating IOU's advantage over a purely empirical or analytical system and over other integrated learning systems such as IOE.

INTRODUCTION

Current approaches to integrating empirical and explanation-based learning (EBL) methods use one of the methods to focus the other method or supply it with needed information (e.g. [Flann88, Lebowitz86, Pazzani88]). Although the first method helps to bias the overall system in these approaches, the final concept definition is completely constructed by the second method. An alternative approach is to use each method to learn different aspects of individual concepts. Many concepts have aspects which can be explained in terms of functionality or intentional-ity as well as others aspects which cannot be explained by the current theory, and may be just "conventional." We are developing a learning technique called Induction Over the Unexplained (IOU), which combines EBL and empirical techniques to efficiently learn both explanatory and nonexplanatory aspects of a concept. This paper describes the basic IOU approach and presents an initial implementation of IOU.

THE IOU APPROACH

Many important concepts have both explanatory and nonexplanatory aspects. Scripts for events such as a birthday party or a wedding have goal-directed as well as ritualistic actions. Concepts for artifacts such as a cup or a building have functionally important features as well as aesthetic or conventional ones. Animals have some attributes with clear survival value as well as more obscure features. Diseases have some symptoms which can be causally explained by current biological theory as well as others which are simply known to be correlated with the condition.

The general method we are proposing for learning such concepts is to use EBL techniques to learn as much as possible and then use similarity-based learning (SBL) methods to detect regularities in the unexplainable aspects of the examples and thereby add "conventional" features to the concept definition. Features which can be explained are learned from a single instance using standard explanation-based learning techniques. These aspects are then removed from the initial example and all subsequent examples and the reduced example descriptions are passed on to an empirical system which finds additional commonalities and adds them to the concept definition.

In IOU, SBL complements EBL's inability to learn unexplainable features of a concept. On the other hand, EBL immediately identifies certain features as important, resulting in early and efficient learning of these aspects of the concept. In previous research, the approach has been to focus the SBL component on aspects of the examples which explanations reveal may be relevant. In IOU, the approach is to allow EBL to eliminate those features which explanations reveal are definitely relevant and focus on unexplained aspects of the examples.

The general problem IOU addresses is the theory-based concept specialization problem [Flann88]. The system is assumed to have a correct theory for a generalization of the concept to be learned. The theory is incomplete in that it is incapable of justifying the restrictions necessary for the specialized concept. As an example of a

* This research was partially supported by the University Research Institute and the Department of Computer Sciences at the University of Texas at Austin.
problem suitable for IOU, consider the classic CUP example. The domain theory is the standard one except the "target" concept is renamed DRINKING-VESSEL since the theory cannot actually distinguish between the concepts CUP, GLASS, MUG, SHOT-GLASS etc.

\[
\begin{align*}
\text{STABLE}(x) \land \text{LIFTABLE}(x) \land \text{OPEN-VESSEL}(x) & \rightarrow \text{DRINKING-VESSEL}(x) \\
\text{HAS-BOTTOM}(x) \land \text{FLAT-BOTTOM}(x) & \rightarrow \text{STABLE}(x) \\
\text{GRASPABLE}(x) \land \text{LIGHT}(x) & \rightarrow \text{LIFTABLE}(x) \\
\text{HAS-HANDLE}(x) & \rightarrow \text{GRASPABLE}(x) \\
\text{WIDTH}(x, \text{SMALL}) \land \text{INSULATING}(x) & \rightarrow \text{GRASPABLE}(x) \\
\text{HAS-CONCAVITY}(x) \land \text{UPWARD-POINTING-CONCAVITY}(x) & \rightarrow \text{OPEN-VESSEL}(x)
\end{align*}
\]

Assume the set of examples includes cups, shot-glasses, mugs and cans as shown in Table 1. The problem is to use the domain theory and explanation-based techniques to learn the explainable features of a cup and to use empirical techniques to learn the nonexplanatory features which rule out shot glasses and mugs.

**AN INITIAL IOU ALGORITHM**

The current implementation of IOU is nonincremental and restricted to a purely featural language. A description of the basic algorithm used by the current system is show below:

1) Compute and generalize proofs demonstrating that each positive example is an instance of the overly-general "target" concept.
2) Disjunctively combine the resulting definitions to form the explanatory component \(C_e\) of the concept.
3) Disregard any negative examples which do not satisfy the explanatory component.
4) Remove features mentioned in the explanatory component from the descriptions of the positive examples and remaining negative examples.
5) Give the "reduced" set of examples to a standard inductive learning from examples system to compute the nonexplanatory component of the concept \(C_n\).
6) Output: \(C_e \land C_n\) as the final concept description.

Step one uses standard EBL techniques to construct and generalize explanations for each of the positive examples. A version of the EGGS system [Mooney88] is used for this task. Step two combines the resulting definitions disjunctively to form the explanatory component of the concept. For the CUP example, this produces the following definition for \(C_e\):

\[
\text{HAS-BOTTOM}(x) \land \text{FLAT-BOTTOM}(x) \land \text{HAS-CONCAVITY}(x) \land \text{UPWARD-POINTING-CONCAVITY}(x) \land \text{LIGHT}(x) \\
\land \begin{cases}
\text{HAS-HANDLE}(x) \\
\text{WIDTH}(x, \text{SMALL}) \land \text{INSULATING}(x)
\end{cases}
\]

Step three eliminates negative examples which do not satisfy the explanatory constraints on the concept. Since the explanatory component adequately explains why these examples cannot be members of the concept, there is no need to pass them along to the empirical system. In the CUP example, the negative CAN-1 instance can be discarded since it does not quite meet the functional requirements of a drinking vessel. Step four removes the explained features of the remaining examples to allow the empirical component to focus on their unexplained aspects. The resulting reduced set of data for the example is show in Table 2. In step five, the unexplained data is given to a standard empirical system for learning from examples. We currently have implementations of the version-space

<table>
<thead>
<tr>
<th>NAME(CLASS)</th>
<th>BOTTOM</th>
<th>FLAT</th>
<th>CONC</th>
<th>UP</th>
<th>LIGHT</th>
<th>HANDLE</th>
<th>WIDTH</th>
<th>INSUL</th>
<th>COLOR</th>
<th>VOLUME</th>
<th>SHAPE</th>
</tr>
</thead>
</table>
| CUP-1(+)    | YES    | YES  | YES  | YES| YES   | YES    | SMALL | NO    | WHITE | SMALL | CYLINDER
| CUP-2(+)    | YES    | YES  | YES  | YES| YES   | YES    | NO    | SMALL | RED   | SMALL | CYLINDER
| SHOT-GLASS-1(-) | YES | YES | YES | YES| YES | YES | NO | SMALL | CLEAR | TINY | CYLINDER |
| MUG-1(-)    | YES    | YES  | YES  | YES| YES   | YES    | MED   | NO    | COPPER | LARGE | CYLINDER
| CAN-1(-)    | YES    | YES  | YES  | YES| YES   | YES    | NO    | SMALL | SILVER| SMALL | CYLINDER

<table>
<thead>
<tr>
<th>NAME(CLASS)</th>
<th>COLOR</th>
<th>VOLUME</th>
<th>SHAPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUP-1(+)</td>
<td>WHITE</td>
<td>SMALL</td>
<td>CYLINDER</td>
</tr>
<tr>
<td>CUP-2(+)</td>
<td>RED</td>
<td>SMALL</td>
<td>CYLINDER</td>
</tr>
<tr>
<td>SHOT-GLASS-1(-)</td>
<td>CLEAR</td>
<td>TINY</td>
<td>CYLINDER</td>
</tr>
<tr>
<td>MUG-1(-)</td>
<td>COPPER</td>
<td>LARGE</td>
<td>CYLINDER</td>
</tr>
</tbody>
</table>
algorithm, ID3, and A⁴, any of which can be used as the empirical component of IOU. For the example, all of these systems generate the same most-general (simplest) description for \( C_n \): VOLUME(x, SMALL). The version-space algorithm also generates the most specific description: VOLUME(x, SMALL) \& SHAPE(x, CYLINDER). The final step of IOU simply conjunctively combines the explanatory and nonexplanatory descriptions into a final concept definition.

There are two important aspects to notice about IOU's use of the SBL component. First, it can use any SBL system which supports the description language used by the overall system since the IOU algorithm is independent of the details of the SBL component. Even a neural-net learning algorithm can be used to learn the nonexplanatory part of the concept. Second, the amount of data given to the SBL component can be greatly reduced by the EBL component. This decreases computational complexity and helps focus the empirical component.

**IOU VERSUS PURE SBL AND IOE**

By using an existing domain theory to learn the explanatory part of a concept and to focus empirical learning on unexplained features, IOU is capable of learning a concept from many fewer examples than a purely empirical system. When ID3 and A⁴ are run on the original examples shown in Table 1, their bias for simple hypotheses causes them to focus on the irrelevant feature, COLOR, and construct the erroneous definition: COLOR(x, RED) \& COLOR(x, WHITE). ID3 or A⁴ would require a much larger set of examples to learn the correct definition. Comprehensive experiments and formal analysis are needed to better determine the advantage IOU has over a purely empirical system.

Many approaches to combining EBL and SBL are also inappropriate for learning concepts with explanatory and conventional aspects. For example, *Induction Over Explanations* (IOE) [Flann88] is incapable of learning such concepts since it assumes that all relevant features are present in the explanations of the examples. If IOE is applied to the example presented above, the resulting concept description is identical to the explanatory component learned by IOU which fails to correctly classify all of the examples in Table 1.

**CONCLUSIONS AND FUTURE RESEARCH**

IOU is an integrated learning mechanism for acquiring concepts with both explanatory and nonexplanatory aspects. It combines existing explanation-based and similarity-based learning systems in order to take advantage of the benefits of each. Unlike other approaches to integrated learning which use one method to focus the other or provide it with information, IOU allows each method to contribute to a different part of the overall concept description. As a result, it is capable of modeling various aspects of the human learning of explanatory and nonexplanatory information [Ahn88].

In addition to experimental and analytical evaluation of the existing system, there are a number of research directions we plan to pursue. These include constructing an incremental version of the algorithm, extending it to handle structural descriptions, enhancing the classification method for IOU concepts, and developing IOU techniques for learning from observation.

**References**


CONCEPTUAL CLUSTERING OF EXPLANATIONS

Jungsoon P. Yoo and Douglas H. Fisher

Department of Computer Science, Box 1679, Station B
Vanderbilt University, Nashville, TN 37235
yoojp@vuse.vanderbilt.edu & dfisher@vuse.vanderbilt.edu

ABSTRACT:
Inductive and explanation-based learning methods traditionally differ in the extent that they exploit background knowledge. Each is search intensive and sensitive to domain imperfections, albeit in different ways. Hybrid systems that abstract over explanations promise the advantages of each approach, but they require augmentation if their promise is to be fully realized. Conceptual clustering can be profitably applied to the abstraction and organization of explanations so that they may be efficiently and appropriately reused.

INDUCTION-BASED AND EXPLANATION-BASED LEARNING

Empirical learning from examples systems induce concepts from preclassified examples. In general, many examples may be required to converge on a concept(s) that appropriately cover the observed instances. Moreover, empirical techniques depend on domain independent biases (e.g., language constraints such as conjunctive concepts only) to constrain the search for concepts. In addition, acquired concepts can not be completely justified; relationships may be coincidental or misperceived. In contrast, explanation-based learning (EBL) from examples systems exploit a domain theory to explain that an instance is a member of the specified class. For example, Mitchell, Keller, and Kedar-Cabelli's (1986) EBG constructs a proof tree that explains how a training example satisfies a target concept. The proof(AND) tree is generalized to the weakest conditions such that the proof of the target concept (i.e., root of the tree) is still valid. Leaves of the proof tree are generalized properties of the training example that are relevant to the proof. A new instance with similar properties can be efficiently matched against this generalization; thus, it is operational since it may avoid the need to prove membership from scratch.

EBG constructs one proof of class membership for each object, but many proofs may exist. Justified generalizations for alternative proofs may cover distinct instances. To cover all class members, several 'seed' training examples may be required. In principle, an EBL learner may extract all legitimate explanations for class membership from the domain theory without the aid of any training examples, but the search is considerably focused by carefully selected training examples.

Several researchers have suggested empirical/explanation-based hybrid systems. Pazzani (1988) and Hirsh (1988) empirically generalize over explanations for multiple examples. Multiple explanations for the same target concept are juxtaposed and explanatory substructures that are not shared are excised. The remaining explanation structure is then generalized in a manner similar to EBG. Flann & Dietterich (in press) term this process mEBG (i.e., multiple-example EBG). Conceptually, their Induction over Explanations (IOE) method concatenates a specialization process onto the end of mEBG and respecializes the generalized structure to fit the peculiarities of the training examples (e.g., if all instances share a constant term, the constant is reinserted into the explanation structure).

OPEN PROBLEMS

Empirical learning requires that class-wide properties be expressed over a pool of training instances. Extracting criterial properties is open to error and many training examples may be required. EBL still relies on training examples to 'seed' criterial portions of domain theory. Methods that generalize over explanations hope to reduce the number of training examples further and more concisely represent a class by extracting commonality over explanations, but there are problematic issues to be addressed before these techniques can be fully exploited.

IOE and mEBG treat unlike explanation substructure as contradictory, rather than as alternatives. Moreover, if we wish to use explanatory structure to predict class membership (in addition to justifying it), then particular care must be taken to avoid overgeneralization or undergeneralization. Finally, EBL methods are not traditionally concerned with noise, but Minton (1988) has explored the related issue of explanation utility. If training ‘seeds’ are ill-chosen, then the corresponding explanation structures may be unrepresentative and/or infrequently used. In either case, they can be detrimental to system performance. IOE and mEBG may give too much credence to unrepresentative cases. Finally, these techniques remove those portions of the explanation structure that are most operational (i.e., leaves); explanation applicability can be more costly to determine. We propose to mitigate some of these problems through conceptual clustering. Alternative explanations can be maintained in different nodes of an abstraction hierarchy. In addition, the utility of explanations can be evaluated over time and removed if they prove unhelpful. Finally, the abstraction hierarchy can be exploited for efficient classification and reuse of explanations for justification and prediction.

CONCEPTUAL CLUSTERING OF EXPLANATIONS

Conceptual clustering may be used to organize objects into classes so as to promote accurate prediction of environmental properties (e.g., Fisher’s (1987) COBWEB system). Conceptual clustering can be adapted to the task of organizing explanations. Proposals by Stepp & Michalski (1986) come close to this idea: a goal-dependency network identifies object attributes that are relevant to a particular goal and other background rules enable default inferences of unknown attribute value. Thus, background knowledge is used to identify relevant attributes and predict their value through a ‘proof’ procedure. However, once done attribute value ‘explanations’ would apparently be thrown out – they would not participate in the abstraction hierarchy. We combine Michalski & Stepp’s ideas of explanatory ability with COBWEB’s concerns of prediction ability; explanatory structures should be an explicit part of the conceptual hierarchy.

An explanation (proof) structure is typically built from n-ary predicates. In addition, predicate variables may be related by equality or subordination. We propose that a good class, $C_k$, is one that increases the expected number of predicates that can be correctly predicted of class members that could not be predicted without knowledge of class membership (Gluck & Corter, 1985):

$$P(C_k) \sum_i [E(\text{pred}_i | C_k) - E(\text{pred}_i)]$$

Initially, we assume that a class’s concept description is a conjunction of all predicates that are true of all class members; $E(\text{pred}_i | C_k)$ is interpreted as the number of predicates in the conjunctive expression of $C_k$. Assuming that $C_k$ is in an abstraction hierarchy, $E(\text{pred}_i)$ is the number of predicates true of $C_k$’s parent. $P(C_k)$ is the proportion of all generated explanations that are members of $C_k$ vice one of $C_k$’s siblings; it weights the increase in expectation by the proportion of the explanations to which the increase applies.

In order to classify an explanation, we juxtapose it with each class of the tree’s first level. Since we insist that each predicate be true of all of a node’s descendents, some matches may insist that we drop one or more predicates or generalize the predicate to accommodate the new instance (i.e., replace a constant to variable or remove an equality between two variables). In fact, this generalization procedure is identical to that used by IOE. In any case, generalization will increase the scope of the class, but also lower the expected increase in the number of predicates that can be predicted with certainty. The best matching class is chosen to incorporate the explanation.

As an example, consider the task of solving an algebra story problem. For example, an ‘opposite direction’ problem is “Two trains leave the same station at the same time. They travel in opposite directions. One train travels 64 km/h and the other 104 km/h. In how many hours will they be 1008 km apart? ” The solution can be obtained as follows: the time can be found from the formula $Time = Distance/Rate$, where the distance is given as 1008 km and the rate can be derived by adding 64 km and 104 km since two trains travel in opposite directions. The ‘proof’ or ‘explanation’ is a solution trace that illustrates the relation between problem statement (i.e., operational knowledge) and the correct solution (i.e., ‘target concept’).
Conceptual clustering allows simultaneous maintenance of explanation structures at several levels of abstraction which efficiently directs the search for reusable solution traces. Figure 1 illustrates a classification tree over explanations of algebra story problems. In general, we expect that an explanation hierarchy responds to queries framed in operational terms. For example, a problem statement requires that we predict a solution (target concept) via a generalized solution trace from previous problems. Thus, we use problem statement knowledge to label arcs of the tree and guide classification. Partial solutions are stored at nodes and are recovered during classification. Problem solving on new problems is a matter of collecting normative portions of the solution trace as one descends the hierarchy.

CONCLUDING REMARKS

We have proposed a method of conceptual clustering of explanations. The anticipated advantages of this work are the efficient and appropriate reuse of explanations for justification and prediction. In addition, our approach is sensitive to the frequency of explanation access; in the case of some explanations, it may be more expensive to store and continuously check their applicability than to generate them from scratch (Minton, 1988). The classification hierarchy helps to isolate and prune unhelpful explanations.

References


A Tight Integration of Deductive and Inductive Learning*

GERHARD WIDMER (gerhard%ai-vie.uucp@uunet.uu.net)
Department of Medical Cybernetics and Artificial Intelligence, University of Vienna, and
Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A–1010 Vienna, Austria

Abstract
The paper presents a novel approach to combining deductive and inductive learning strategies in a flexible way. Our method is best described under the general heading 'Explain and Compile': by loosening our notion of an explanation, we arrive at a uniform learning framework which provides for a flexible combination of inductive and deductive tendencies. The algorithm was developed in the context of an interactive Learning Apprentice System for two-voice counterpoint composition.

1 Introduction
In recent years, there has been a growing awareness in the machine learning community that for many realistic applications, neither purely empirical, inductive learning methods nor extremely knowledge-intensive techniques such as Explanation-Based Learning will be adequate approaches. This has led more and more researchers to explore the 'space' between these two extremes. There appear to be two paths that can be followed: we can try to find classes of knowledge that are weaker than the strong, implicative theories required for EBL, or experiment with various combinations of inductive and deductive methods to try to get the best of both worlds.

A number of proposals have been made for combining empirical and analytic learning methods. Some of these systems realize only a loose coupling, performing empirical generalizations before [Lebowitz 86] or after [Danyluk 87, Flann & Dietterich 86] a deductive analysis of the instances. Other systems use inductive generalization to learn rules that are missing from incomplete domain theories. Systems of this type include OCCAM [Pazzani 88], SIERRA [VanLehn 87], and Hall's PA and RR algorithms [Hall 88]. Bergadano & Giordana developed an integrated, but non-incremental algorithm that can handle domain theories that are both incomplete and inconsistent with the data [Bergadano & Giordana 88].

The approach presented here tightly integrates deductive and inductive learning and can learn incrementally from incomplete domain theories. The method was developed and implemented in the context of a Learning Apprentice System for a musical application, namely, two-voice counterpoint composition.

2 A new integration framework: generalized explanations
For domains whose structure permits the construction of hierarchical explanations, we propose the following new approach: learning can be viewed (as in EBL) as consisting in first explaining a training example, and then generalizing and compiling the explanation. If we are willing to widen our notion of an explanation, this gives us a uniform framework for learning with incomplete domain theories. Explanation structures in our system are trees as in EBL. However, while in EBL the tree is a proof tree, we also allow weaker links – determinations and similarity arguments – in the explanation:

- A feature F can be explained by finding a rule F :- Cond and recursively explaining the validity of Cond in the current context; this is the kind of deductive explanation used in EBL.
- A feature F can be explained if we know that F is more or less functionally determined by other features F1 ... Fn, and if we already know another situation where the same combination of values for F and F1 ... Fn holds as in the current situation; this is determination-based analogy [Russell 86].
- Finally, if we do not have any strong knowledge tying F to any known conditions, we can still 'explain' the fact that F holds by appealing to some similarity between the current situation and some other situation(s) where F also holds. If there are domain-specific heuristics relevant to determining similarity, these can be taken into account. There are two distinct possibilities:
  - If there are other examples with the same classification that all share certain features with the situation to be explained, these similarities may constitute an 'explanation' of the current situation. This kind of weak argument, when compiled into a rule, leads to inductive generalization effects.
  - If there is a learned rule which almost fits the situation to be explained, this rule may in fact be too specific; the similarities between the situation and the rule's preconditions may suffice to explain the situation. As a side-effect, this kind of argument leads to incremental generalization of learned rules.
There are a few things to note about our particular implementation: first, since the system is a Learning Apprentice, we heavily exploit the presence of a human teacher; the teacher serves as an oracle confirming or rejecting hypothesized explanations presented by the system. In particular, we use determinations not for purposes of analogy, but as patterns for plausible explanations, which must then be confirmed by the teacher (see example). Secondly, deciding which examples and similarities might be relevant to the explanation of a specific situation is a very search-intensive process (the well-known problem of empirical learning). When looking for known situations that are similar to the example to be explained, we perform a best-first search that trades off the number of examples sharing some attributes against the degree of generality of the common description; also, domain-specific heuristics are used to evaluate proposed generalizations. Situations that cannot be explained with the help of rules or determinations and that do not exhibit a reasonable degree of similarity to other known examples (there is a pre-defined threshold) are simply stored; they may be used for empirical generalization (i.e., explanations by similarity) later on.

3 An application example

Two-voice counterpoint composition is a comparatively simple problem in tonal music. The task consists in writing a second line (the counterpoint), given the melody (the 'cantus firmus') of a piece. The concepts to be learned are rules that classify candidate notes as either good, bad, or totally unacceptable as partial solutions in a particular context. Fig.1 lists some of the musical knowledge (domain theory) relevant to the example. Fig.2 shows a new example and the explanation tree that is constructed by the system. This explanation combines all three types of arguments: at the top level, a determination suggests a possible explanation, and its two preconditions are verified partly through deduction (EBL - left subtree), and partly by appeal to the similarity between the example and RULE 15, a rule that has already been learned (in both cases, there is a melodic leap of at least a perfect fifth (p5)). The rule compiled out of this explanation tree is shown at the bottom of Fig.2. Note that both the explanation by similarity and the application of the determination must be confirmed by the teacher.

4 Discussion

Our approach has several important qualities:

- Tight, fine-grained integration of inhomogeneous domain knowledge: Every single explanation step is guided by the specific knowledge applicable. Strict rules can be combined with determinations and similarity heuristics as well as with data from known examples, and all the relevant knowledge influences the appearance of the final rule.

- Integration of determinations: Determinations are a very natural form for expressing abstract, high-level domain knowledge. In our model, there is no need to be concerned about the operationality of the features a determination refers to: non-operational features will be refined in the recursive explanation process. This allows us to formulate determinations directly at the appropriate level of abstraction.

1) The abstract rule shown in Fig.1.b is not a determination in the strict sense as defined by Russell; it does not specify a symmetrical relationship between the acceptability of a note and its explanation - a note can be unacceptable for other reasons, too. Thus, the rule represents the sufficiency part of a determination argument, but not the necessity part. This is due to the particular structure of our problem domain. However, there is nothing that precludes the use of genuine determinations in our learning framework, if such are available.
A Tight Integration of Deductive and Inductive Learning

Fig. 2: Explanation of new training instance and compiled rule

- **Incrementality**: Learned rules can be used later on in the learning process: they can be further generalized, and they can be used as parts of larger arguments (see example). However, our system does not as yet have an algorithm for discrimination. At the moment, it deals with overgeneralizations (and inadequacies of the representation language) by explicitly storing exceptions to rules and trying to generalize them as more exceptions become known. Since we rely on the teacher to prevent overgeneralization, we also have not had to deal with the kinds of truth maintenance problems treated in [Pazzani 88a].

- The approach is a step toward a uniform view of learning 'between' EBL and SBL: The algorithm is a learning schema which can realize various forms of more or less knowledge-intensive learning, depending on the strength of the domain theory. Obviously, the method is somewhat weaker on the inductive side; coercing an inherently bottom-up process like data-driven empirical induction into a top-down framework does not do much for efficiency. Consequently, the method performs best with a rather strong, but possibly incomplete, background theory.

We are currently conducting further research in two directions: some strategy for discrimination must be developed, and we plan to perform experiments with different fields of application in order to find out which classes of domains are amenable to this approach. It is our hope that the learning strategy will prove to be applicable to a wide variety of domains and will thus be of practical use to the construction of intelligent Learning Apprentice Systems.

References


MULTI-STRATEGY LEARNING IN NONHOMOGENEOUS DOMAIN THEORIES

Gheorghe Tecuci
Research Institute for Computers and Informatics
71316, Bd. Miciurin 8-10 Bucharest 1, ROMANIA

Yves Kodratoff*
George Mason University, AI Center
Fairfax VA 22030-4444

ABSTRACT

This paper presents DISCIPLE, a system illustrating a theory and a methodology for learning expert knowledge in the context of a nonhomogeneous domain theory (Kodratoff & Tecuci, 1987). DISCIPLE integrates a learning system and an empty expert system, both using the same knowledge base. It is initially provided with a nonhomogeneous domain theory and learns problem solving rules from the problem solving steps received from its expert user, during interactive problem solving sessions. In the context of a complete theory about the example, DISCIPLE uses explanation-based learning to improve its performance. In the context of a weak theory about the example, it combines explanation-based learning, learning by analogy, and empirical learning.

INTRODUCTION

We have been developing an experimental multi-strategy learning system (MILS) called DISCIPLE. Precursors to MILS are Learning Apprentice Systems (LAS) which have been defined (Mitchell, Mahadevan & Steinberg, 1985) as interactive knowledge-based consultant, provided with an initial domain theory and able to assimilate new problem-solving knowledge by observing and analyzing the problem solving steps contributed by its users, through their normal use of the system. MLIS have the same general purpose as the LAS, but they use a multi-strategy approach to learning, instead of being based on deductive reasoning. Representative examples of LAS are the systems LEAP (Mitchell, Mahadevan & Steinberg, 1985) and GENESIS (DeJong & Mooney, 1986). The domain of expertise of LEAP is the VLSI design and that of GENESIS is story understanding. A common feature of LEAP and GENESIS is that they are based on a strong (complete) domain theory which allows them to learn a general rule or schemata from a single example. A more recent contribution is the system ODYSSEUS which uses explanations to improve an incomplete domain theory by observing the human expert's actions (Wilkins, 1988). When the descriptions are complete, DISCIPLE behaves as a LAS, and uses EBL to build up new efficient rules. When the descriptions are incomplete, DISCIPLE uses an interactive learning method that combines explanation-based learning, learning by analogy, and empirical learning.

DISCIPLE AS AN EXPERT SYSTEM

To build an Expert System with DISCIPLE, one has to first introduce a knowledge base into DISCIPLE. Next, DISCIPLE may be used to interactively solve problems: The user gives DISCIPLE the problem to solve and the expert subsystem starts solving this problem by showing the user each problem solving step. The user may agree or reject it. In the latter case, or when DISCIPLE is unable to propose any partial solution, the user is compelled to give his own solution.

THE LEARNING PROBLEM

It can be formulated as follows. Given: a domain theory; a problem to solve and a partial solution to the problem, determine: a general rule, with its preconditions for application.

* On leave from CNRS, Univ. Paris-Sud, LRI, Bat. 490, F-91405 Orsay France
For example, **Given:** the theory of loudspeaker manufacturing; the problem of attaching two parts of the loudspeaker (the 'ring' and the 'chassis-membrane-assembly') and a user given solution in the form of a decomposition of this problem into two simpler subproblems expressing the glueing of the two parts. (Example 1)

**Solve the problem**
**ATTACH OBJECT ring ON chassis-membrane-assembly**

**by solving the subproblems**
**APPLY OBJECT mowicoll ON ring**
**PRESS OBJECT ring ON chassis-membrane-assembly**

**Determine:** a general decomposition rule indicating the conditions under which one may reduce an 'attachment' problem to a process of glueing, for instance

\[
\text{IF (x TYPE solid) \& (y TYPE solid) \& (x PARTIALLY-FITS y) \& (z ISA adhesive) \& (z TYPE fluid) \& (z GLUES x) \& (z GLUES y)}
\]

**THEN**
**solve the problem**
**ATTACH OBJECT x ON y**

**by solving the subproblems**
**APPLY OBJECT z ON x**
**PRESS OBJECT x ON y**

**LEARNING IN A COMPLETE THEORY DOMAIN**

In the case of a complete theory about Example 1, the learning method of DISCIPLE follows the explanation-based learning paradigm developed by (DeJong & Mooney, 1986; Fikes, Hart & Nilsson, 1972; Mitchell, Keller & Kedar-Cabelli, 1986). It proves that the solution indicated by the user is indeed a solution of the problem to solve, then it generalizes the proof tree as much as possible so that the proof still holds. This is done as in (Mooney & Benet, 1986) by replacing each instance of action model or inference rule with its general pattern and by unifying these patterns. Finally, it produces a learned rule from the generalized proof by extracting the generalized problem, its generalized solution, and the generalized relevant features which are the applicability condition of the rule.

**LEARNING IN A WEAK THEORY DOMAIN**

The learning method is the following one:

**Explanation-Based Mode**

(1). Find an explanation of the user's solution (Example 1) and call it Explanation 1

**Analogy-Based Mode**

(2). Over-generalize Example 1, by simply turning all the objects into variables, and call it General Rule 1.
(3). Take Explanation 1 as a Lower Bound for the applicability condition of General Rule 1.
(4). Over-generalize Explanation 1 to the most general expression that may still be accepted by the user as an explanation of General Rule 1.
(5). Take the over-generalized explanation as an Upper Bound for the applicability condition of General Rule 1. The Upper Bound, the Lower Bound, and the General Rule 1 define a reduced version space for the rule to be learned.
(6). Look in the knowledge base for "interesting" objects satisfying the Upper Bound. If there are such objects then Call Explanation-i the properties of these objects which were used to prove that they satisfy the Upper Bound and go to step 7. If there are no such objects then show the Upper Bound, the Lower Bound, and the General Rule 1 to the user as an uncertain rule and stop.
(7). Take Instance-i as a near miss (negative example) of the rule to be learned. (8). Find an explanation of why Instance-i was rejected by the user and call it Failure-Explanation-i.
(9). Specialize the Upper Bound as little as possible, so as not to cover Failure-Explanation-i. If the new Upper Bound is identical with the Lower Bound then take it as a necessary and sufficient condition of General Rule 1, show them to the user and stop, else go to step 12.
(12). Specialize (if necessary) the Lower Bound as little as possible, so that not to cover Failure-Explanation-i.
(13). Go to step 6.
(14). Take Instance-i as a new positive example of the rule to be learned and Explanation-i as a true explanation of Instance-i. (15). Look for a maximally specific common generalization of the Lower Bound and Explanation-i. Two cases may occur: (151). if this generalization is not identical with the Upper Bound, then take it as the new Lower Bound and go to step 6; (152). if this
CONCLUSIONS

Trying to cope with the complexity of the real world applications, we have made the hypothesis that DISCIPLE’s domain theory is nonhomoeneous, describing completely some parts of the domain, but only incompletely or even poorly, the other parts.

In the context of a weak theory DISCIPLE integrates different learning paradigms: explanation based learning, learning by analogy, empirical learning, and learning by questioning the user. Among the most relevant features of this learning method one could mention: the notion of “explanation” in a weak theory and a heuristic method to find such explanations, the use of analogy to define a reduced version space for the rule to be learned, the use of both the explanations of the successes and the explanations of the failures to search the rule in its version space, the formulation of “clever” questions in order to extract useful knowledge from the expert, the possibility of hiding the learned rules to the expert since variablized rules may disturb him, a great confidence in the human expert.

In the context of an incomplete theory, DISCIPLE learns by combining the method corresponding to the complete theory with the method corresponding to the weak theory. This method borrows features from both the learning method in a complete theory (may reject incorrect examples, learns justified rules) and from the learning method in a weak theory (use of analogy, clever questions to the user, etc.).

It is interesting to notice that, although in each of the presented cases the system learned the same general rule, the effect of this rule on the future behavior of the system depends of the domain theory: in a complete theory, the learned rule improves only the performance of the system, in a weak theory it develops the competence of the system, and, in an incomplete theory, it may develop both the performance and the competence.

There are several weaknesses of DISCIPLE, on which will shall direct our future research.

The use of DISCIPLE is severely restricted to problem solving situations in which some variablization is meaningful. For instance, a set of zeroth-order rules solving a problem cannot be refined using DISCIPLE. An other limitation is relative to the disposition of the variables in the rules. DISCIPLE must follow two rules. The expressions it deals with cannot contain actual function evaluation. Secondly, all the variables that appear in an action must also appear in the solution of that action. In more formal words, this states that DISCIPLE rules must be equivalent to the so-called “groundable” clauses.

The generality of the learned rule is limited by the generality of the over-generalized explanation (the analogy criterion) which may not be in the most general form, the method of finding an explanation in a weak theory is not powerful enough. Other sources of knowledge are needed, as well as meta-rules for finding far off explanations. DISCIPLE uses control knowledge in the form of meta-rules but such knowledge is not learned, having to be provided by the user. Therefore, if two experts provide different solutions to the same problem, DISCIPLE simply generates two different rules. The learning mechanisms of DISCIPLE should be used to propose explanations of this difference and find meta-explanations that can become meta-preconditions on the use of the rules.

References

A DESCRIPTION OF PREFERENCE CRITERION IN CONSTRUCTIVE LEARNING: A Discussion of Basic Issues

Jianping Zhang & Ryszard S. Michalski
Center For Artificial Intelligence
George Mason University
4400 University Drive
Fairfax VA, 22030

1. INTRODUCTION

The criterion for preferring one concept description over the other plays an important role in both empirical and analytical learning systems. Without the preference criterion, these learning systems would lack the means for deciding which of the many alternative candidate descriptions should be chosen. For example, an empirical learning system could simply take the disjunction of all positive examples as the learned description of the concept while an analytical learning system could take the goal concept description as its learned concept description.

In inductive learning, given examples, background knowledge, and optionally, an initial concept description, the system hypothesizes a general concept description. Usually a large number of general descriptions can be generated for any set of examples and/or initial concept descriptions. For an analytical learning system, given a complete domain theory, a goal concept description, and a positive instance, a set of descriptions which explains the given instance are deductively generated from the goal concept description. To choose among the candidate descriptions in both systems one needs a criterion for preferring one description over the other.

The problem of evaluating description is not new to empirical learning systems, and a number of measures of description quality have been developed in the past. Some of them concentrate solely on the aspects of completeness and consistency. Other measures include also additional criteria, such as the simplicity and the cost of evaluating the learned descriptions. Broader aspects of the problem of what should be the preference criterion for judging competing inductive hypotheses are discussed in (Michalski, 1983; Utgoff, 1986; Bergadano et al., 1988).

In early analytical learning systems (e.g., Mitchell et al., 1986), operationality was the only criterion used for selecting the target concept description. Recently some researches (e.g., Segre, 1987) called for considering a trade-off between operationality and generality.

In order to design a learning system that integrates empirical and analytical methods, an unified preference criterion that combines the criteria used in both methods has to be defined first. This paper proposes a preference criterion that can be used in a constructive learning (CL) system (Michalski et al., 1988) which will be discussed in next section. The proposed criterion combines four basic criteria: accuracy, operationality, generality, and simplicity.

2. CONSTRUCTIVE LEARNING

Currently, there are two main approaches to concept learning: syntactic learning and analytical learning (Michalski & Kodratoff, 1989). The simplest form of syntactic learning is empirical learning which performs inductive inference from a training set of concept instances, without use of extensive amount of background knowledge. In this form of learning, the systems usually generates descriptions that use descriptors (attributes, predicates and terms) selected among those present in the descriptions of learning examples (selective induction). In more advanced syntactic methods, the systems perform constructive induction which is able to generate new descriptors that are not present in the input data. Constructive induction is usually guided by domain theory or domain independent heuristics.

The fact that empirical approach does not need much domain knowledge is both an advantage and a disadvantage. On one side, since it primarily relies on the examples given to the system, and examples are often easy to obtain, this approach is very attractive for many applications. On the other side, because it utilizes little domain knowledge, it can hardly be applied to learning in complex, knowledge intensive domains.

Analytical learning approach, in particular, its most popular form -- explanation-based learning (EBL), produces an operational concept description starting from a domain theory, a goal concept description and, a single concept instance. In order to obtain a concept description, complete and consistent domain knowledge is required. When incomplete, inconsistent or uncertain domain knowledge is present, pure analytical learning systems fail to generate any concept description. The requirement of complete and consistent domain knowledge is often too hard to satisfy in the real world.

As many researchers (e.g., Pazzani, 1988) pointed out, both pure empirical and analytical learning methods fail as general theories of learning. They should not compete against each other, instead they should complement to each other in an integrated learning paradigm. Some systems that integrate these two learning strategies have been
described in the literature (e.g., Pazzani, 1988). The results generated from these systems have confirmed that a
multistrategy methodology is worthwhile to pursue.

In the following, we will briefly describe the fundamental ideas underlying constructive learning and then concentrate
on criterion for evaluating the quality of a concept description.

Constructive learning aims at developing concept learning systems that unify and generalize syntactic and analytical
learning. In constructive learning, the system, given a learning task, explores first the relationship between its
background knowledge, the goal of learning and the task to be performed. Whenever prior knowledge is incomplete,
correct or not useful for the task, the system executes a form of inductive learning to derive required knowledge.
This form of learning may be selective induction, if the prior knowledge is very limited or it can be constructive
induction, that utilizes background knowledge to construct new descriptors and concepts in the process of learning
(Michalski, 1983). If the system has sufficient knowledge for the task, then it acts as an analytical learning system.
The knowledge the system acquires is always assimilated in the knowledge base so that it can be used in subsequent
learning. Such a feature is called closed-loop learning. Thus, a constructive learning system integrates selective and
constructive induction, analytical learning, and is also capable of closed-loop learning. For more detail, see
(Michalski et al., 1988).

3. INDIVIDUAL CRITERIA AND THEIR RELATIONSHIPS

This section first discusses four single criteria: accuracy, operationality, generality, and simplicity, that enter into the
unified preference criterion proposed in this paper.

Accuracy represents the description's ability to produce correct classifications. It is measured by the correctness of a
concept description with respect to the information and the knowledge available to the learning system. The basic
and easy-to-measure criterion that relates to accuracy of a conjunction is the ratio of the number of the positive
training instances covered by the conjunction and the number of total instance covered by the conjunction (negative
and positive). Most of learning systems generate completely accurate descriptions. It is not necessary to generate
such completely accurate descriptions for the following reasons:

1. the concept itself may be imprecise and flexible,
2. some examples may have been erroneous,
3. an approximate description is acceptable if an accurate description is hard to find (or expensive to use).

For these reasons, a less accurate description may be better than an accurate one. In fact, in order to achieve
completeness and consistency (total accuracy) in the presence of noise, one may have to generate overly complex and
detailed descriptions. Such descriptions, however, may not perform well in future cases and examples. This is the
well known phenomenon of overfitting.

Operationality The goal of explanation-based learning is to obtain an operational concept description from a non-
operational goal concept description by analyzing a particular instance of that concept. Generally speaking, an
descriptor is operational, if it can be used efficiently by the performance element. We will use a degree of
operationality instead of binary operationality used by most of EBL systems.

Generality There are two aspects of generality of a concept description in an inductive learning system. The first
is related to the coverage of positive instances of concept. The more positive instances a concept description covers,
the more general and more desirable the description. The second is related to the coverage of the unseen instances.
This aspect of generality affects the predictive power of the description. The more general a description, the more
predictive it is and there is more chance to commit errors when using it. Relative generality is easily defined. If the
system has a domain theory with logical axioms, then a concept q can be shown to be at least as general as the
concept p, if it can be shown that p \rightarrow q.

Simplicity is related to the comprehensibility. An important requirement from an AI system is that its knowledge
should be explicit and easily understandable by human experts. This is crucial for systems that need to communicate
with experts. A black box classifier will not be accepted by experts as a help in their work, even if it performs very
well. Knowledge acquired automatically should be easy to understand, contain descriptors used by experts, and not be
syntactically too complex.

Generally speaking, these four criteria are separate and distinct, one constitutes different dimensions in our unified
criterion. In many situations, in order to satisfy one criterion, one has to sacrifice another. We will now discuss
some of the relationships among these criteria.

Segre (1987) discusses the trade-off between operationality and generality in EBL systems. This trade-off plays an
important role in our constructive learning system. The goal concept description used in EBL can be generated by an
inductive learning system, if operationality of descriptors is not considered. This is because the goal concept
description is usually most general, accurate and simplest. Yet, it may be too general to be used by a performance
element. On the other hand, an EBL system may generate an operational concept description which may be too
specific to be used for the future.
In some applications, there may not exist both very general and operational description. There are three reasons that a too specific concept description may not be efficient, even it is operational. First, it may take a lot of memory to save a complete concept description. Second, it may take time to search a conjunction from the whole concept description which is matched with the current instance to be classified. Finally, a too specific description may have not enough predictive power, and be not applied to some cases in the future. Consider a concept description that is the disjunction of all instances, each conjunction, an instance, is operational. It is obvious that it is not what we want to learn. In the case discussed above, a less operational but more general and simpler description may be used much more efficiently by a performance element.

There is one more reason to prefer a less operational, but more general and simpler concept description. It is not necessary that a concept learning system only learns concept descriptions for the performance element. In a domain which is not completely known, a human user may want to know the principles behind some concepts in the domain and understand the concept completely. In this case, a less operational description preferred. The trade-off between operationality and generality is dependent on the goal of learning a concept.

The relationship between operationality and accuracy is less obvious. In fact, they are two independent criteria. But in some domains, like medical domains, one can gain operationality of a description at the expense of accuracy, or one can gain accuracy by sacrificing operationality. This is because some approximations may have to be made to obtain an operational description. For example, to have an accurate diagnosis in some situations, expensive tests have to be made.

In the context of learning an imprecise and flexible concept, it is generally true that the more general a description is, the less accurate it is. This is because that a more general description may cover more exceptions or rare cases of an imprecise and flexible concept.

A more general description usually is a simpler description. The relationship between simplicity and the other two criteria (accuracy and operationality) is similar to the relationship between generality and these two criteria.

We have discussed all four components in the quality measure and their relationships. These individual criteria need to be combined into a single evaluation procedure that can be used to compare different concept descriptions. Several mechanisms, such as linear weighted function, lexicographic evaluation function (Michalski, 1972), or the combination of these two, can serve for this purpose. The mechanism used to combine these criteria should be flexible enough to allow the user to specify his own preference. And also the system should have the capability to adjust the the description quality criterion to adapt to the learning task at hand.

Acknowledgements

This research was done in the Center for Artificial Intelligence of George Mason University. The activities of the Center are supported in part by Defence Advanced Projects Agency under grant, administered by the Office of Naval Research, No. N00014-87-K-0874, in part by the Office of Naval Research under grant No. N00014-88-K0397, and in part by the Office of Naval Research under grant No. N00014-88-K-0226. The authors thank Tak Wai Chen and Pawal Stefanski for useful criticism.

Reference


Pazzani, M. J., "Integrated Learning with Incorrect and Incomplete Theories", Proceedings of the Fifth International Conference on Machine Learning, Ann Arbor, MI: Morgan Kaufmann, 1988


COMBINING CASE-BASED REASONING, EXPLANATION-BASED LEARNING, AND LEARNING FROM INSTRUCTION

Michael Redmond
School of Information and Computer Science
Georgia Institute of Technology
Atlanta, Georgia 30332-0280

ABSTRACT

Learning from instruction is a powerful technique for improving problem solving. An active student will predict the instructor's actions and then try to explain the differences from the predictions. We expand the concept of explanation beyond the provably correct explanations of Explanation-based learning to include other methods of explanation. The explanations can use deductions from causal domain knowledge, plausible inferences from the instructor's actions, previous cases of problem solving, and induction. The explanations result in improved diagnosis and improved future explanation. This combination of learning techniques leads to more opportunities to learn.

INTRODUCTION

Some of the barriers that researchers run into when pursuing individual machine learning techniques can be surmounted by the use of a combination of techniques. Redmond and Martin [1988; Martin and Redmond 1988], VanLehn [1987], Wilkins [1988], and Hall [1988] have explored the use of instruction to go beyond Explanation-based Learning (EBL) [DeJong and Mooney 1986; Mitchell, Kellar, and Kedar-Cabelli 1986] to learn truly new knowledge. Hammond and Hurwitz [1988], and Barletta and Mark [1988] used EBL techniques to improve Case-based Reasoning (CBR) [Kolodner and Simpson 1984] by creating appropriate predictive indices for cases. Martin [1988] has used inductive techniques to help CBR retrieve appropriate cases. Our current approach uses a combination of instruction, deductive explanation, previous cases, and induction to create a more complete learning system.

The foundation of the approach is the use of instruction in the form of solved example problems to focus learning. The student makes predictions of what the instructor will do, using the same techniques s/he would use to diagnose problems on his own. When those predictions are not met, the student has an opportunity to learn. The motivated student searches for explanations of why the instructor performed a particular action or proposed a particular solution.

Explanation of the problem solving demonstrated in instruction leads to new knowledge and knowledge refinement that will enhance future problem solving and future explanation. The explanation can be achieved by several methods. The effect of the explanations can be reflected in change to the domain model, new cases to use in explaining future instruction, and inductive shift in feature salience to lead to more appropriate cases being retrieved.

The current approach is an extension of previous work by Redmond and Martin on Learning by Understanding Explanations (LBUE). LBUE was originally treated by Redmond and Martin as an integration of learning from instruction and EBL, enabling greater learning, and not requiring a complete and correct domain theory. We have generalized the LBUE process to use more than causal chaining in explanations.

A student or learning system applying this process has a background domain model that is not necessarily complete or consistent. This model also holds cases and generalized cases that involve failures in the domain. Case-Based Reasoning (CBR) [Kolodner and Simpson, 1984] is a method of using previous episodes and evaluation of their results to suggest solutions to new problems. CBR is an important problem solving technique because it can allow faster solutions when previous experiences are available. CBR can also be useful in explaining observed problem solving. At the same time, explanation-based and inductive techniques and instruction help improve the performance of CBR.

In this paper we discuss explanations involving:
- Inferring the instructor's current goal
- Inferring the place of the current goal and actions in the diagnosis episode.
- Adjusting the saliency of features for future case retrieval.
- Causally explaining actions.

1This research was supported by the Army Research Institute for the Behavioral and Social Sciences under Contract No. MDA-903-86-C-173. The author wishes to thank Janet Kolodner for her advice and guidance, and Joel Martin for helpful comments on earlier versions of the paper.
INFERRING INSTRUCTOR’S GOAL

Since the instructor’s goal is usually not explicitly stated, it must be inferred from his actions. The predicted goal is the first goal considered as a possibility. If the instructor’s actions are not consistent with that goal, the goal must be inferred bottom up, with all possible goals being possible. This means that if the student gets lost in the example, s/he can find actions that make sense and get back to following along from there, and salvage something from the instructional episode.

In a diagnostic domain some possible goals include generating a hypothesis, testing a hypothesis, interpreting a test, fixing a fault, and clarifying a complaint. The representation for goals includes a modifiable, and therefore learnable, structure specifying what types of actions and statements are reasonably expected for fulfilling that goal. Some goals require particular types of actions. Some action types are inappropriate for some goals. Some action types can occur multiple times in the pursuit of a particular goal, some can only occur once. To give one example of the type of inference involved, testing a hypothesis must include an ask type action in order for results to be obtained. If none of the diagnosis-specific goals are appropriate, a more general goal can be considered, which could result in a diagnosis-specific specialization of the goal being learned. Once the system knows what goal is being pursued, then the same explaining is done as if the goal had been correctly predicted. The student can recover and resume following the instructor.

INFERRING PLACE IN CURRENT DIAGNOSIS

Inferring the place of the current goal and actions in the diagnosis episode is important in understanding what the instructor is doing, and for saving the episode in a useful form as a case. The instructor in most cases diagnoses hierarchically. The experienced mechanic considers a system as a potential source of the problem, then narrows the hypothesis down until a replaceable unit is determined to be malfunctioning. The instructor’s actions are sequential, the observer recognizes the tree. A system cannot rely on a given pattern of actions from the instructor, but must actually explain or understand what is going on. Figure 1 demonstrates this with an example diagnosis sequence. The top part of Figure 1 shows the structure of the instructor’s actions shown in the bottom part.

Figure 1: Inferred Diagnosis Structure.

Note that a test does not necessarily follow the hypothesis it relates to. Also note that there is more than one reason that a hypothesis can directly follow another hypothesis. It could be a refinement, as with the ‘leak fuel system’ hypothesis following ‘malfunction fuel system’, or it could be another possibility at the same level, such as with the ‘clogged fuel lines’ hypothesis directly following the ‘leak fuel system’ hypothesis.

Knowledge is necessary to understand the hierarchy being used. Causal knowledge and structural relationships from the model are both useful for this process. The new action, in order to be linked to the preceeding action, must actually be related to the previous one, by being more specific or causally related. For example, in Figure 1, the hypothesis ‘clogged fuel lines’ is more specific than the hypothesis ‘malfunction fuel system’ because the predicate is more specific and the component is at least as specific. Since the hypothesized fault ‘clogged spark plug gap’ can be causally linked to the previously hypothesized ‘no spark from spark plug’, it is a refinement, and can be placed below that hypothesis. If the action cannot go after the most recent action then
the system must search for its proper place. Many of the other heuristics are limitations on this process, either avoiding potential incorrect placements, or cutting off search that will prove to be unfruitful. Once the structure of the observed diagnosis has been determined, the case can be stored in memory for future problem solving and explanation use. The case is stored in pieces so that the particular pieces can be accessed as necessary, while still being linked to preserve the structure of the case.

**ADJUSTING THE SALIENCE OF FEATURES**

It may not seem that adjusting the salience of features for future case retrieval is really explanation. However, when two or more hypotheses are both correct hypotheses, in that they can both cause the observed symptom, causal EBL-like explanations do not provide a way of distinguishing between them. If the student predicts a different hypothesis than the instructor makes, then adjusting the matching function by adjusting the importance of features will lead to better prediction in the future. The intuition is that such weighting of competitive hypotheses in diagnosis is generally inductive, the mechanic doesn’t know for a fact that \( x \) fails more often than \( y \), statistics aren’t readily available or used, nor can such preference be explained deductively.

The method of adjusting the salience of features is fairly simple. It is based on the idea of increasing the importance of features that match when the problem solver is successful, and decreasing the importance of features that match when the problem solver is unsuccessful. When the student predicts a different action than the instructor, the student has been unsuccessful. The adjustment is made by retrieving another case piece in which the instructor’s action was done. Those features of the current context that more closely match the context of the ‘correct’ piece than the context of the ‘incorrect’ piece will be made more important, and those features that more closely match the ‘incorrect’ piece than the ‘correct’ piece are made less important.

This will lead to the correct piece being retrieved in the same situation in the future. A combination of instruction, CBR, and induction has been used to improve the performance of the CBR part of diagnosis.

**CAUSAL EXPLANATION OF ACTIONS**

Causal explanations of actions enable filling gaps in the causal domain knowledge through LBUE methods described in Redmond and Martin [1988]. In addition to enabling filling gaps in the causal domain knowledge trying to causally explain actions can make causal explanations available as indices to the new case containing the action. This second use hasn’t yet been implemented in the current system.

**CONCLUSION**

Explanation of solved example problems uses EBL-like deduction, induction, and retrieval of previous cases in order to improve future diagnoses and future explanations of observed problem solving. Existing LBUE methods of explaining the instructor’s actions lead to better future explanation by improving the causal domain model. The inductive explanation of feature saliency helps improve future problem solving through better case retrieval. At the same time, the instruction episode is saved in a case that will be useful in both future problem solving and future explanation. The exploitation of instruction turns out to be a powerful way of learning, and integrates several learning techniques.

References


DEDUCTION IN TOP-DOWN INDUCTIVE LEARNING

F. Bergadano, A. Giordana and S. Ponsero

Dipartimento di Informatica, Università di Torino,
C.so Svizzera 185, 10149 - Torino, Italy

This paper proposes a new flexible strategy for combining Analytic and Empirical learning in order to acquire conceptual descriptions in real domains such as fault-diagnosis or medical diagnosis, in which imperfect and intractable theories are, in general, available. The learning strategy is seen in the framework of a learning model (presented by the authors in [BGS88]) based on a top-down specialization process which can now be guided by interleaving deductive and inductive steps [BG88]. This work introduces two main novelties in comparison to the earlier ideas presented in [BG88]. The first consists in the adoption of both logical and dependency relations for describing the domain theories, in which it is possible to distinguish axioms which are sure from axioms which are just possible and other ones which can be just partially specified. This knowledge representation formalism can be effectively exploited in order to reflect the real domain knowledge possessed by a technician and offers a good background in order to decide where and how to activate deductive or inductive steps, and where to hypothesize theory incompletenesses and inconsistencies. The second novelty, which is described in more detail in [BGP89], consists in the repertory and in the flexibility of the reasoning schemes which can be used in the learning process. Moreover, the method has now been experimented on larger case studies, and good results were obtained.

The domain theory is represented as a set of Horn clauses, but, in order to account for incompleteness and uncertainty, some predicates can be defined only partially, by using dependency rules of the following kind:

\[
\{P_1(x_1,\ldots,x_n),\ldots,P_s(y_1,\ldots,y_m)\} \rightarrow Q(z_1,\ldots,z_t)
\]  

(1)

This expression means that \(Q(z_1,\ldots,z_t)\) could be defined precisely using a set of clauses, where only \(P_1,\ldots,P_n\) occur. Such a definition is useful when the user cannot supply a precise specification of a predicate through a set of Horn clauses, but does have some idea about which predicates could be used in such a specification. This kind of knowledge is related (but not equal) to the concept of "dependency" in databases and of "determination" as defined in [DR87]. Fully defined predicates are operational (and will be written lowercase in the following examples) if they occur in the examples we learn from, that is, if they are not defined with Horn clauses, using other predicates.

For example, the following clause could be used:

\[
P(x,y) \land R(x,w) \rightarrow S(x)
\]

(2)

and \(R(x,w)\) could be partially defined as follows:

\[
\{m(a,t,u),N(v)\} \rightarrow R(t,s)
\]

(3)

\[
H(v-1) \land J(v) \rightarrow N(v), \quad L(v,1) \rightarrow H(v)
\]

(4)
meaning that $R$ could be defined through a set of Horn clauses, although we do not know how, and the predicates that could be used in these clauses are $m(a,t,u)$ and $N(v)$. Also, clauses defining $N(v)$ are supplied. It should be noted that the use of variables provides additional constraints in the partial definition of $R(t,s)$, since $t$ must be bound to the same constant $x$ is bound to in (2), in predicates $P$ and $R$. Therefore once this constant is fixed, it can be substituted for $t$ in $m(a,t,u)$ thus limiting the number of alternative definitions of $R$. Partially defined predicates can be nested, e.g. $J$ occurs in the partial definition of $R$. If a dependency rule has an empty antecedent, as again is the case for $J$, this simply means that the predicate can be dropped wherever it occurs.

This knowledge representation scheme is extremely general and yet easier to use than a language that does not localize incompleteness. Several methods for learning from examples and knowledge can be framed into this approach.

- Using background knowledge in inductive learning [Mic83]: there is one top level dependency relation, $\{p(x_1, ..., x_n) | p \text{ is operational}\} \sim \text{Concept}$. We can use constructive rules as defined in [Mic83], by augmenting the theory with the corresponding clauses and by allowing $p$ in the above dependency relation to be either an operational predicate or one of the predicates defined by the constructive rules.

- "Pure" explanation based learning: no dependency rules are present, and predicates are either operational or non operational. Nevertheless, many examples are used rather than one.

- Integration of explanation based learning and empirical learning as defined by the authors in [BG88]: every predicate is defined by a set of Horn clauses and by the most general dependency rule, saying that the predicate could also be defined by any other clause using the available operational predicates. The domain theory was then restricted to Horn clauses, but the obtained operational descriptions could then be modified in any way.

We propose a more general approach than the one presented in [BG88], since the weaknesses of the domain theory are now explicitly stated when possible. The induction of discriminant concept descriptions from a set of examples is formalized as a top-down search in a space of logical formulas. To this aim, a logic framework (similar to the one used by deductive databases) in which both induction and deduction are possible, is used. The same framework was already adopted in [BGS88,BG88], where a system able to find both discriminant and characterizing concept descriptions is presented.

If a domain theory is available this search process can be made more efficient and its results can be more reliable and understandable. In particular, the domain theory can contain some incomplete description of the concept, that is a Horn clause $A \rightarrow H$ where the conclusion $H$ is the concept to be learned and the antecedent $A$ may include partially defined predicates. In order to obtain a concept description we have to (a) operationalize the concept definition in the domain theory, that is perform deductive steps until we get a concept description that contains just operational predicates (b) specialize intermediate formulas and thus solve the incompleteness in the theory by substituting partially defined predicates with an operational definition which is consistent with the corresponding constraints. During every step of the deduction and of the solution of incomplete definitions we maintain the extensional representation of formulas, and thus we can check on the examples their completeness and consistency.
The control strategy accomplishes the selection of the node in the search tree and of the operator to be applied. As it is unrealistic to think of developing all the tree (which can even be infinite if the theory is recursive) this choice is extremely important.

The strategy that we propose trades off the following criteria:
(a) **completeness and consistency**, which is tested on the extensional representation of the formulas: formulas more selective and covering more examples are considered to be better.
(b) **complexity**: formulas that are more simple and obtained using a minor number of operations are considered better.
(c) **justifiability**: during the inductive specialization process, the formulas that can be justified with some reasoning based on the theory, if if not deduced in a strict sense, are considered to be better.

One kind of such reasoning is based on abduction, that is a formula is considered to be weakly justified if it can be obtained from the theory though an interleaved application of deductive and abductive steps. The examples that are available are then used for the inductive verification of this reasoning process. Another type of reasoning, which is more effective in order to obtain consistent descriptions quickly, is based on the consequent $H$ of a classification rule $\phi \rightarrow H$. The assertion $H'$, should contain the concept $h$ that we want to learn. Then, if it is possible to deduce from the theory an operational description $\psi \rightarrow H'$ such that $h \in H \cap H' \subset H$, we can generate the rule $\psi \land \phi \rightarrow H \land H'$ which is more discriminant than $\phi \rightarrow H$ and is justified by the domain theory.

The described methodology has been experimented on a test case taken from a real problem of troubleshooting in a mechanical domain. The problem was that of learning how to diagnose from findings four types of failures for a given type of pump. As the number of examples available in the learning set was small, 13 examples distributed non homogeneously on the classes, the pure inductive approach gave a very poor result. Using a domain theory consisting of 46 rules, the learning system ML-SMART learned a set of 26 diagnostic rules able to perfectly diagnose all the learning set, which resulted still correct for a small test set of 5 new cases and which, the most important, resulted perfectly understandable for a technician, expert of the domain. More extensive problems of the same domain are now being experimented.

References


The purpose of this paper is to describe a framework for integrating empirical learning with explanation-based learning (EBL) [DeJong & Mooney 1986; Mitchell, Keller & Kedar-Cabelli 1986] and to present an algorithm which does this with both pure conjunctive concepts and \( k \)-CNF concepts. Our framework involves using an empirical and an explanation-based method to form separate hypotheses and then combining the hypotheses from the separate sources to form a composite hypothesis. An additional important complication arises because the system is required to learn the domain theory (via an empirical method) at the same time it is using the domain theory to support the explanation-based method. The empirical methods that we use are one-sided algorithms that never generate a hypothesis that is more general than the correct hypothesis (assuming that the hypothesis can be represented in the hypothesis representation language). In addition, the empirical algorithms that we consider maintain a single hypothesis that is generalized as little as possible to accommodate positive examples.

The hypotheses produced by explanation-based learning with a domain theory acquired with such a one-sided empirical learning method will also never be more general than the correct hypothesis. Since both the empirical and explanation-based hypotheses are not more general than the correct hypothesis, they can be combined by finding the least general hypothesis consistent with both hypotheses. In this manner, the integrated hypothesis will be the least general hypothesis that is consistent with both the observed data and the domain knowledge. This hypothesis may be more general than either the empirical or explanation-based hypotheses. Some regularities may be ruled out because they are not consistent with the data. Other regularities may be ruled out because they are not consistent with the domain theory, although they may be supported by the data.

Performance and Foundational Examples

In order to understand the framework for integrated learning, it is important to note that there are two different kinds of training examples. Performance examples are training examples of the complete performance task. For example, imagine the task is for a small child to learn when other people will become angry. Performance examples here would be examples of people becoming angry when the child performs some action (e.g., breaking Daddy's watch). Foundational examples are training examples from which background domain knowledge can be learned. These differ from performance examples in that they can be viewed as a subproblem of the performance task. The performance task of predicting a way that a person will get angry can be broken into two subproblems, predicting what actions will cause an object to break, and predicting what sort of objects that become broken will anger what sort of people. More formally, assume the background knowledge is of the form: \( A \) and \( X_{A,B} \rightarrow B \) and \( X_{B,C} \rightarrow C \) and we wish to acquire a predictive relationship: \( A \) and \( X_{A,C} \rightarrow C \).
The goal is to learn the relationship “If you play with an expensive, glass object, the owner will become angry.” The background knowledge might be acquired when a mother tells a child “Don’t play with Daddy’s watch; if you drop it, it will break and Daddy will get angry.” Here, $A$ represents dropping an object, $B$ represents an object breaking, and $C$ stands for a person getting angry. The term $X_{A,B}$ represents a number of unspecified conditions that restrict the class of objects that are broken when dropped (e.g., objects composed of glass), $X_{B,C}$ refers to additional conditions which are needed to determine what class of persons will be come angry when what class of objects breaks (e.g., the owner becomes angry when an expensive object breaks). These conditions are not specified in the domain theory; they must be acquired empirically from foundational examples. The goal of learning is to acquire $X_{A,C}$. This can be learned empirically from performance examples, or analytically ($X_{A,C} = X_{A,B}$ and $X_{B,C}$) from a domain theory acquired from foundational examples. In the next section, we give an algorithm that combines empirical and explanation-based methods in solving this problem.

THE IOSC and $k$-IOSCNF ALGORITHM

The first empirical learning strategy we consider is the Wholist strategy [Bruner, Goodnow & Austin 1956]. This strategy has also been called the One-Sided Algorithm for Pure Conjunctive Concepts [Haussler 1987]. Wholist works as follows: when a positive instance of the concept is seen but the current concept definition would classify it as a negative instance, the concept definition is redefined to be the intersection of the current concept definition and the instance. This process removes from the definition any features which are not in the instance and therefore can not be in the true definition of the concept. In Bruner’s work, the initial hypothesis was a conjunction of all features in the first positive example. Here, we initialize the hypothesis to be the conjunction of all features in the example description language.

The first algorithm we will present is the integrated One-Sided Algorithm for Pure Conjunctive Concepts (IOSC). In IOSC, the empirical algorithm is used for two purposes. First, it is used as the only learning algorithm to acquire the domain theory from foundational examples. Second, it is used to form one hypothesis for the performance concept from the performance examples. Explanation-based learning produces a second hypothesis for the performance example. These hypotheses are combined to form IOSC’s hypothesis. When there is no domain theory, IOSC is equivalent to Wholist. In this manner, IOSC can be used on both performance and foundational examples. For simple conjunctive concepts, EBL merely finds the conjunction of the features in the antecedents of the rules in the domain theory. Since the regularities present in the performance examples may differ from the regularities present in the foundational example, the hypotheses produced by empirical and explanation-based means will differ. In IOSC, there are two reasons that are sufficient for dropping a feature from a hypothesis: either the feature was not present in all positive performance examples, or the feature is not needed to explain why a performance example is an instance of the performance concept.

Input:

- $H$ - the current hypothesis of the concept definition to be learned (current concept definition – initialized to the entire list of features).
- $B$ - the current background theory
- $E$ - a training example (instance of either the goal concept (performance example) or background concepts (foundational example)).
- $Member$? - Boolean value indicating whether or not $E$ is a positive example.

Output: An updated concept definition.

The algorithm:
• iosc($H, B, E, Member?$)

1. $C = \text{Classify}(H, E)$ (this classifies the instance as positive or negative based on the current concept definition).

2. If $(C \text{ negative and } Member? = \text{True})$ Then
   
   (a) If domain-theory-explains($E, B$) THEN (if you can use EBL)
   
   i. $H1 = \text{"remove features from } H \text{ which are not in } E\text{"}$;
   ii. $H2 = \text{"create hypothesis with EBL"}$;
   iii. “update $H$ by removing those features of $H1$ which are not in $H2$.”

   (b) ELSE “update $H$ by removing features from $H$ which are not in $E$.”

We have extended this algorithm to the case of $k$-CNF by reformulating the definition of a feature. In IOSC, features are simply the surface features of the training examples and hypotheses are conjunctions of these features. For $k$-CNF, composite features can be constructed that are disjunctions of the surface features of length $k$ or less. Hypotheses in $k$-IOSCNF are conjunctions of these composite features. Instances are defined as a list of Boolean variable settings (e.g., $L = 1$, $M = 0$, etc.) which satisfy the true definition of the concept (either goal-concept or background concept) being learned with this instance. The empirical component for $k$-CNF from Valiant (1984) is used instead of Wholist.

Both Wholist and Valiant’s $k$-CNF algorithm have two important properties that enable the combination of the empirical and analytic hypothesis. First, both algorithms are one-sided (i.e., the hypothesis is never more general than the true concept definition). Thus, the hypothesis formed by EBL with such a domain theory, learned by these algorithms, will also never be more general than the true concept definition. Second, the hypothesis representation language is closed under conjunction. Therefore, EBL will produce a hypothesis in the same representation language as the empirical component, permitting the two hypothesis to be combined.

CONCLUSION

In this paper we have discussed a framework for learning $k$-CNF by combining empirical learning with EBL. This algorithm can easily be extended to include truth maintenance. Finally, although IOSC is limited in applicability by its constrained representation language, $k$-CNF expressions are powerful enough to be used in describing application domains such as medical diagnoses.

References


COMBINING EMPIRICAL AND ANALYTICAL LEARNING
WITH VERSION SPACES

Haym Hirsh
Computer Science Department
Stanford University
Stanford, CA 94305
Haym.Hirsh@cs.cmu.edu

ABSTRACT

This paper describes an approach to combining empirical and analytical learning using incremental version-space merging (Hirsh, 1989). The basic idea is to use analytical learning to generalize training data before doing empirical learning. The combination operates like empirical learning given no knowledge, but can utilize knowledge when provided, and thus exhibits behavior along a spectrum from knowledge-poor to knowledge-rich learning.

INTRODUCTION

The problem of concept learning—forming general rules from specific situations—has been well-studied in machine learning and artificial intelligence. There have been two principal approaches to this problem. The first, empirical learning (also known as similarity-based learning), finds concept descriptions that best cover a set of training data. At the other extreme is analytical learning (also known as explanation-based learning), which finds the weakest preconditions on a knowledge-based analysis of a single instance, forming a generalization that covers all instances that have the same analysis.

The two techniques each have shortcomings. Empirical approaches do not provide general methods for using knowledge. On the other hand, analytical methods require very strong forms of knowledge. This paper presents a combination of the two techniques that overcomes shortcomings they each have when used in isolation. The basic idea is to use empirical learning on the results of analytical learning, rather than on the ground data. The combination operates like empirical learning given no knowledge, but can utilize knowledge when provided, and thus exhibits behavior along a spectrum from knowledge-poor to knowledge-rich learning.

Explanation-based generalization (EBG) (Mitchell, Keller, and Kedar-Cabelli, 1986) is the form of analytical learning used in this work. EBG proves that a goal concept holds for an example of the concept called the training instance, using a domain theory of rules and facts about the goal concept. EBG forms a generalization of the instance, defining the class of instances that are examples with the same proof as the training instance. It does so by finding the weakest preconditions on the proof, restricting such conditions to expressions that satisfy an operationality criterion on the merit of the generalization for the problem solver.

Incremental version-space merging (Hirsh, 1989) is the form of empirical learning used in this work. It is based on a generalization of Mitchell’s (1978) version-space approach to concept learning that removes its assumption of strict consistency with data. A version space is generalized to be any set of concept definitions in a concept description language representable by boundary sets.* The key observation is that concept learning can be viewed as the two-step process of specifying sets of relevant concept definitions and intersecting these sets. For each piece

*The boundary sets $S$ and $G$ contain the most specific and general concept definitions in the set. These bound the set of all concept definitions in the version space—the version space contains all concepts as or more general than some element in $S$ and as or more specific than some element in $G$. 
of information obtained—typically an instance and its classification—*incremental version-space merging* forms the version space containing all concept definitions that are potentially relevant given the information (determined as appropriate for the given learning task). The resulting version space is then intersected with the version space based on all past data. This intersection takes place in boundary-set form (using the *version-space merging algorithm* (Hirsh, 1989)), and yields the boundary-set representation for a new version space that reflects all the past data plus the new information.

The general algorithm proceeds as follows:

1. Form the version space for the new piece of information.
2. Intersect this version space with the version space generated from past information.
3. Return to the first step for the next piece of information.

Use of incremental version-space merging requires a specification of how the individual version spaces should be formed in the first step for each iteration. For example, using simple consistency with instances (i.e., forming the version space of all concepts that correctly classify the current instance) results in an emulation of Mitchell's candidate-elimination algorithm (Mitchell, 1978). Other examples are presented elsewhere (Hirsh, 1989), as are further details of the generalized version-space approach. The key idea in this work is to use explanation-based generalization to form instance version spaces.

**USING INCREMENTAL VERSION-SPACE MERGING ON THE RESULTS OF EBG**

This work applies incremental version-space merging to the results of EBG, rather than on ground data. The problem addressed is:

*Given:*

- Training Data: Positive and negative examples of the concept to be identified. Training data are expressed within an instance description language, whose terms are assumed to be operational.
- Concept Description Language: A language in which the final concept must be expressed. It is a superset of the instance description language, and may (and is indeed likely to) include generalization hierarchies.
- Positive-Data Domain Theory (optional): A set of rules and facts for proving that an instance is positive. Proofs terminate in elements of the instance description language.
- Negative-Data Domain Theory (optional): A set of rules and facts for proving that an instance is negative. Proofs terminate in elements of the instance description language.

*Determine:*

- A set of concept definitions in the concept description language consistent with the data.

The method processes a sequence of instances as follows, starting with the first instance:

1. (a) If possible, apply EBG to the current instance to generate a generalized instance. Do so for all possible explanations. If no explanation is found, pass along the ground data.
   (b) Form the version space of all concept definitions consistent with the (perhaps generalized) instance. If there are multiple explanations include those concept definitions consistent with any single explanation.
2. Intersect this version space with the version space generated from all past data.
3. Return to the first step for the next instance.
Combining Empirical and Analytical Learning with Version Spaces

Note that this is merely an instantiation of the general incremental version-space merging algorithm given in the preceding section. The basic technique is to form the version space of concept definitions consistent with the explanation-based generalization of each instance as it is obtained (rather than the version space of concept definitions consistent with the ground data). The version space for a single training instance reflects the explanation-based generalization of the instance, representing the set of concept definitions consistent with all instances that are explained in the same manner as the given instance. The merging algorithm has the effect of updating the version space with the many examples sharing the same explanation, rather than with the single instance. In this manner irrelevant features of the instances are removed, and learning can converge to a final concept definition using fewer instances.

The technique also applies to cases of multiple competing explanations, when only one explanation need be correct. In such cases the version space of concept definitions consistent with one or more of the potential results of EBG is formed. EBG is applied to every competing explanation of an instance, each yielding a competing generalization of the instance. The space of candidate generalizations for the single instance contains all concept definitions consistent with at least one of the competing generalizations. The final generalization after multiple instances must be consistent with one of them. The version space of all concept definitions consistent with at least one explanation-based generalization of the instance is the union of the version spaces of concept definitions consistent with each individual explanation-based generalization. For positive examples this union has as its $S$ boundary set the set of competing explanation-based generalizations, and the $G$ boundary set contains the universal concept that labels everything positive. Over multiple instances these version spaces consistent with the explanation-based generalizations are incrementally intersected to find the space of concept definitions consistent with the analytically generalized data.

The approach is also useful given theories for explaining negative data, when the system is provided with a theory capable of explaining why an instance is negative. For example, in search control an example of a state in which an operator should not be used is a negative instance, and a theory for explaining why the instance is negative would analyze why the instance is negative—that the operator does not apply, or that it leads to a non-optimal solution (Minton, 1988). This theory is then used to generalize the negative instance to obtain a generalization covering all instances that are negative for the same reason. Incremental version-space merging then uses this generalized instance by setting the $S$-set equal to the empty concept that says nothing is an example of the concept, and setting the $G$-set equal to all minimal specializations of the universal concept that do not cover the generalized negative instance. If there are multiple competing explanations these version spaces consistent with the explanation-based generalizations are incrementally intersected to find the space of concept definitions consistent with the analytically generalized data.

Note that it is not necessary to have a complete theory capable of explaining (and generalizing) all correct instances for this technique to work. The version space of all concept definitions consistent with plain non-generalized instances—whether negative or positive examples—can always be formed. Instead of using EBG, the version space consists of all concept definitions consistent with the instance, rather than its explanation-based generalization. If a theory for only explaining positive instances exists, negative instances can be processed without using EBG. If an incomplete theory exists (i.e., it only explains a subset of potential instances), when an explanation exists the version space for the explanation-based generalization of the instance can be used, otherwise the pure instance version space should be used. When there is no domain theory the learner behaves like the candidate-elimination algorithm. The net result is a learning method capable of exhibiting behavior at various points along the spectrum from knowledge-free to knowledge-rich learning.

**PERSPECTIVES**

If either empirical or analytical learning alone were sufficient for learning, there would be no reason to combine the two techniques. It is therefore useful to study and understand what each of the learning techniques offers the other beyond what they could do in isolation. From the perspective of EBG the use of incremental version-space merging
addresses the imperfect theory problem (Mitchell et al., 1986). From the perspective of incremental version-space merging EBG allows the use of knowledge as an explicit bias on learning.

**IMPERFECT DOMAIN THEORIES**

As discussed earlier, this approach can utilize examples that cannot be explained; this occurs when the domain theory can only explain a subset of the potential correctly classified instances. In such cases of incomplete domain theories the version space of concept definitions consistent with ground data, rather than their explanation-based generalizations, is formed and merged with the version space for past data. Although EBG cannot be used, the instance can still be utilized. In the extreme case when there is no domain theory the approach still applies, emulating the candidate-elimination algorithm.

The combination approach described here also applies when explanations are overspecialized—additional regularities exist in the domain, but they are not represented in the domain theory. It, too, is a subcase of the incomplete theory problem. The assumption is that there are further regularities in the domain that empirical learning finds, perhaps covering more cases than the original theory does. Empirical learning, as done by incremental version-space merging, finds these regularities, generalizing to new situations not covered by the domain theory. New rules more general than those obtainable by EBG are formed.

Finally, the combination of EBG and incremental version-space merging solves a subcase of the inconsistent theory problem. It occurs when there are multiple mutually incompatible explanations for an instance, each yielding different results. If there are multiple competing explanations only one of which is correct, the original domain theory embodies incorrect explanations that cover situations that should not be covered. Although concept definitions consistent with each explanation are considered, the method presented here will drop those generalizations consistent with incorrect explanations if they are not consistent with later data.

**BIASING SEARCH**

The alternative way to view this approach to combining empirical and analytical learning is from the perspective of empirical learning, asking what contributions analytical learning makes to empirical learning. The answer is that analytical learning hastens convergence to a final concept definition. Rather than updating the version space—doing empirical learning—with single instances, each instance has the effect of multiple instances. Updating a version space with the explanation-based generalization of an instance yields the same new version space as updating the version space with all (ungeneralized) instances that have the same explanation as the original instance. In this view the power of the analytical learning method is to allow convergence to a final concept definition using fewer instances.

The knowledge encoded in a domain theory, when used in the manner describe here, excludes concept definitions from consideration, resulting in a smaller search space. For example, concept definitions that only cover a single instance will no longer be considered—only those including all instances covered by the explanation-based generalization of instance are considered. The more specific portions of version spaces are excluded for positive instances, and the more general portions are excluded for negative instances. After each instance more of the version space of candidate concept definitions is ruled out, requiring fewer instances to converge to a final concept definition. Fewer concept definitions are considered; the knowledge biases the search space.

**RELATED WORK**

Most similar to this work is Mitchell’s (1984) proposal for combining empirical and analytical learning of search control. Mitchell proposed applying analytical learning to raw data, then doing incremental empirical learning on
the generalized data. In addition to simply implementing Mitchell's proposal, the technique presented here goes beyond Mitchell's original proposal, handling incomplete theories as well as inconsistent theories when there are multiple conflicting explanations for a single instance.

The IOE method of Flann and Dietterich (1988) bears a close resemblance to this work. In contrast to this work, however, IOE generalizes across explanations, rather than across the results of EBG on those explanations. If a constant appears in all explanations, it will remain in the generalization, even if EBG would have variablized the constant. IOE forms rules that are never more general than what EBG can create. In their view explanations are overgeneral, and IOE will find specializations of such explanations. This work instead finds concept definitions more general than the results of EBG. However, extensions to their work would permit generalizing beyond what appears in the explanation, to obtain results similar to those here.

**SUMMARY**

This paper has described one approach to combining empirical and analytical learning using incremental version-space merging. The central idea is to apply EBG to training data, then do empirical learning on the generalized data. This process permits the use of incomplete and inconsistent domain theories in analytical learning, and utilizes empirical learning to handle such domain-theory imperfections. The combination can also be viewed as allowing the use of knowledge as an explicit bias in learning. The combination approach is capable of exhibiting behavior along a spectrum from knowledge-free to knowledge-rich learning, depending on the amount of knowledge given to the learner.

**Acknowledgments**

Discussions with Bruce Buchanan, Lonnie Chrisman, Tom Dietterich, Oren Etzioni, Nick Flann, Rich Keller, Tom Mitchell, Paul Rosenbloom, Paul Utgoff, and Monte Zweben were invaluable. Computing resources were funded in part under NIH grant RR-00785 from the Division of Research Resources Biomedical Research Technology Program. The author was supported by a Shell Doctoral Fellowship. The arrangement of facilities by Tom Mitchell at CMU for the completion of this work is greatly appreciated.

**References**


FINDING NEW RULES FOR INCOMPLETE THEORIES:
EXPLICIT BIASES FOR INDUCTION WITH CONTEXTUAL INFORMATION

Andrea Pohoreckyj Danyluk
Department of Computer Science -- Columbia University
New York, NY 10027
andrea@cs.columbia.edu

INTRODUCTION

Two disparate machine learning approaches have received considerable attention: explanation-based learning (EBL) and similarity-based learning (SBL). EBL (e.g., [Mitchell et al. 86]) is a deductive approach in which a definition of a concept is learned, usually after observing only a single example of that concept. SBL (e.g., [Michalski and Stepp 83]) is an empirical technique that involves the comparison of a large number of input examples. EBL and SBL have been applied to problems in a variety of domains. Both methods have clear problems, however. EBL assumes that a system is given an explicit theory of the domain that is complete, correct, and tractable. These assumptions are clearly unrealistic for most real-world problems. SBL suffers because of its lack of an explicit domain theory.

A major focus of the research presented here is the elimination of the assumption that the domain theory of EBL is complete. In particular, it considers the problem of working with an incomplete theory by suggesting a method by which gaps in an EBL system's knowledge can be detected and filled. When EBL cannot derive a complete explanation, the partial explanation forms a context in which learning takes place. Information extracted from partial explanations, as well as from complete explanations, can be exploited by SBL to do better induction of the missing domain knowledge. The extracted information constitutes a strong bias for SBL. Our approach differs from earlier work in that it does not require an auxiliary domain theory. (e.g., [Rajamoney 88] requires a theory of experimentation.) Nor do we require substantial interaction with an expert (as, e.g., [Wilkins 88]). [Fawcett 89] applies one of our heuristics, but concentrates on the selection of a single best explanation from multiple possibilities and is, therefore, not concerned with either testing or characterizing strong inductive biases.

In the following section we describe some of the contextual information that can be extracted from partial explanations and how it can be utilized by SBL for the induction of new rules. In the subsequent section we summarize some of our results to date.

HEURISTICS EXPLOITING CONTEXTUAL INFORMATION AS A STRONG INDUCTIVE BIAS

During the course of explaining that an input example is an instance of a particular concept, EBL might find that its domain knowledge is inadequate to complete the proof. (See e.g., [Rajamoney and DeJong 87] for a characterization of the incomplete theory problem.) Standard techniques of SBL might be used to induce the missing knowledge. Rather than using such "weak" methods alone, we claim that any partial explanation derived provides a context in which the missing knowledge is to be learned. Additional contextual information may be found in previously derived partial and complete explanations. Contextual information may be exploited by SBL in order to constrain the number of hypothesized concept definitions that must be considered. We have identified a number of dimensions that might define explanatory contexts from which to learn. Among these are:

1. Past complete explanations were derived in which the currently unproved subgoal was proved. -vs- No past complete explanations were derived in which the subgoal was proved;
2. The unproved subgoal is not at all provable given the domain theory. \(-vs-\) Rules exist that would allow it to be deduced, but they do not apply to the current example;

3. The unproved subgoal has appeared in explanations in the past in which the final goals were the same as the current goal concept. \(-vs-\) The unproved subgoal has appeared in explanations in the past in which the final goals differed from that of the current example.

If information provided by explanatory contexts is to be useful to SBL, some mechanism must exist by which that information may be extracted and exploited. Such mechanisms constitute a set of strong inductive biases [Mitchell 80]. Heuristics for exploiting explanatory contexts include, among others:

1. lowering the priority of features already used in the proved parts of partial explanations;
2. lowering the priority of features used in the proved parts of the current explanation and not used multiple times in any single past explanation;
3. lowering the priority of features with high occurrence in past complete explanations.

Behind each of these heuristics is a rationale for its application. For example, the reason for using the first method is that often input features are not referred to multiple times in a single explanation. These heuristics are, however, not guaranteed to work. We believe that extensive empirical testing will allow us to characterize the sets of heuristics, or biases, that work best, i.e., that allow the most correct rules to be induced with the fewest input examples.

**EMPIRICAL SELECTION OF BIASES**

Our goal is to evaluate the use of strong inductive biases that exploit contextual information from derived explanations. Specifically, we would like to select those sets of biases that lead to the induction of correct rules with the smallest number of examples required for doing the induction. We are performing the evaluation empirically using the Gemini integrated learning system.

Preliminary results have been obtained in the domain of network fault diagnosis. In order to test the use of context-exploiting biases in this domain we began with a complete rule base and deleted a small subset of rules. Our objective was two-fold. First, we wanted to determine whether each of the missing rules could be induced. Second, we wanted to determine how well each of two sets of heuristics worked in varying contexts. In the following, induction set refers to a set of \langle goal, example, (partial) explanation \rangle triples that are input to the induction algorithm. Three different learning contexts were studied. In the first, the unproved subgoal of an induction set was the actual explanation goal. That is, for each example in the induction set, the missing rule would have been the complete explanation. In the second, the missing rule was part of a larger explanation structure. The partial explanations were identical for all examples in the induction set. In the third, the goal concepts differed; corresponding partial explanations of the examples differed significantly. In each context we considered two separate heuristics, or biases. One eliminated example features already appearing in the (partial) explanations. The other, in addition to applying the first bias, eliminated features appearing with too high a frequency across unrelated explanations. Some of our results are summarized in Figure 1. In the graph of Figure 1 the y-axis indicates the number of extraneous conjuncts in the induced rule, and thus measures its over-specificity. We do not include a measure of over-generality as there was only one instance of an over-generalization. This, as well as a more thorough analysis of our results, is discussed in [Danyluk 89].

**CONCLUSION**

We have presented preliminary results in evaluating the use of heuristics that exploit explanatory knowledge and that act as a set of strong inductive biases. The current state of testing with Gemini includes the consideration of
additional domains, new sets of biases, and the loosening of a number of assumptions made in the earlier testing.

Acknowledgments

This research was supported in part by the Defense Advanced Research Projects Agency under contract N00039-84-C-0165 and by a student co-op position that the author holds with Philips Laboratories.

REFERENCES


LEARNING FROM PLAUSIBLE EXPLANATIONS

Tom E. Fawcett (fawcett@cs.umass.edu)
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

INTRODUCTION

This work addresses the incomplete-theory problem [Mitchell et al, 1986], in which a learning system has an explicit domain theory that cannot generate an explanation for every example. The general method is to use the existing domain theory to generate a plausible explanation of the example, and to extract from it one or more rules that may then be added to the domain theory. This method is an application of abductive reasoning in that it is attempting to account for a known conclusion (the goal concept) by proposing various hypotheses which, together with the existing domain theory, may account for it. The plausibility measures are heuristics used to determine which hypothesis is best, based on the explanation that results from each.

THE LEARNING METHOD

The learning system is given positive examples, each consisting of a set of observable features and a goal concept. Every feature provided is correct, but it is not assumed that every observable feature is provided. Only the observable features of an example are available; the system has no information about the structure of the explanation (cf. the Gemini system [Danyluk, 1989]).

It is assumed that the domain theory comprises a set of rules that are descriptive and perform deductions on a single state (as opposed to rewrite rules which transform one state into another). A complete explanation for a concept is one which is either satisfied by a fact; or one for which there is a rule whose consequent is the concept, and all of the rule's antecedents have complete explanations. This is equivalent to a deductive proof. A partial explanation is simply one which is not complete: an explanation tree in which one or more rule antecedents are unproved. It is possible that a goal concept may have no explanation at all.

The learning method is summarized in Figure 1. The goal of the entire process is to acquire new rules so as to improve the explanation capability of the system. This is done by creating a plausible explanation for an example, acquiring one or more rules from the example, and then generalizing existing rules if possible. The new rules may then be used in future explanations.

GENERATING PLAUSIBLE EXPLANATIONS

If a complete explanation can be created for an example, the domain theory is adequate and need not be extended. If the example cannot be completely explained, there are usually many partial explanations that can be generated to explain it. Because the space of partial explanations is very large, two classes of constraints are used to limit the search:

1. Absolute measures or restrictions on the partial explanations:

   (a) Certain relations (e.g. is_a and part_of) are considered special, in that they cannot be left unproved in a partial explanation.

---

1Most of this work was completed under Prof. Don Smith at Rutgers University. Mike Pazzani, Andrea Danyluk and Bernard Silver provided valuable comments on earlier drafts of this paper.
Given a domain theory and an example $E$:

1. Generate the set of partial explanations of $E$. Call this set $P$.
2. Rank the elements of $P$ by their $f$ values (see text). Let $P_{max}$ be the highest-ranked element of $P$.
3. For every unproved antecedent $A$ in the partial explanation $P_{max}$:
   (a) Generate a new (maximally specific) rule for $A$ using the example features from $E$.
   (b) Generalize the rule by variablizing common constants.
   (c) Add the rule to the rule base.

Figure 1: A summary of the learning method

(b) Once a complete explanation has been found for a given antecedent, no explanations will be considered in which this antecedent does not have a complete explanation. This pruning heuristic is very effective in reducing the number of explanations that need actually be generated by the algorithm.

2. Relative measures for comparing two partial explanations. For example, given two competing explanations $X$ and $Y$, $X$ might be preferred to $Y$ because:

(a) $X$ uses fewer rules than $Y$, i.e. $X$’s explanation is shorter and more succinct. This is a conciseness measure, biasing in favor of explanations with a smaller number of inferences.

(b) $X$ uses (i.e. matches) more of the example features than $Y$. This is an accountability measure, biasing in favor of explanations that account for more of the known attributes of the example.

(c) $X$ contains fewer unproved antecedents than $Y$. This biases against explanations that introduce unproved antecedents (unverifiable assumptions or unobserved features) into an explanation.

(d) $X$ uses more specific rules than $Y$. For example, $Y$ may use an inference based on a default value, whereas $X$ references only observed features.

It is difficult to apply these relative preferences directly because they often conflict: one relative measure may prefer explanation $X$ to explanation $Y$, while another prefers $Y$ to $X$. For example, a shorter explanation will be preferred to a longer explanation by measure $2a$, but if the longer one accounts for more example features it will be preferred by measure $2b$. Therefore, these rating measures are combined in an explanation rating function which reduces an explanation to an integer value estimating plausibility. Although the values assigned by this function have little absolute meaning, they may be used to compare the plausibility of two explanations. The function is defined recursively to be:

$$ f(z) = \begin{cases} 
  \text{UnprovedValue} & \text{if } z \text{ is an unproved antecedent} \\
  \text{FeatureValue} & \text{if } z \text{ matches an example feature} \\
  \left( \sum_{\text{Ant} \in \text{antecedents}(z)} f(\text{Ant}) \right) - \text{RulePenalty} & \text{if } z \text{ is a rule}
\end{cases} $$

The values of the constants in bold can be adjusted to shift preference among the metrics. The values used\(^2\) are $\text{UnprovedValue} = -10$, $\text{FeatureValue} = +10$, and $\text{RulePenalty} = +2$. This combination biases strongly against explanations that leave rules’ antecedents unproved, but by using a small $\text{RulePenalty}$, allows for longer explanations to be generated if they "justify themselves" by accounting for more observed features.

\(^2\)These values were derived empirically. Experimentation with the implementation showed the explanation ordering to be sensitive to the relationships among the values, but relatively insensitive to the values themselves.
LEARNING FROM PLAUSIBLE EXPLANATIONS

Each of the terms in an explanation may be regarded as a concept. Although the example was given explicitly as an instance of the goal concept, it is also implicitly an instance of each of the terms in the explanation as well. Since the purpose of the learning system is to extend its domain theory, it is the terms of the unproved antecedents in the plausible explanation whose definitions should be inductively extended, rather than the goal concept provided with the example.

Therefore, once the most plausible explanation is chosen, the system creates a rule for each of the unproved antecedents using the example features. This rule is initially maximally specific, in that it matches only the particular example (and any superset of it). Some generalization is then performed on this rule: any object referenced in the goal concept is variablized, and any constant used as the argument of a unary predicate is variablized throughout the definition.

The resulting conjunction is then added as a new right-hand side for each unproved term in the explanation. This presents an opportunity for inductive generalization. For a given rule, once a number of these new right-hand sides have been acquired, the system could attempt to generalize them inductively. This generalization would consist of going through the set of acquired right-hand sides and removing terms that did not occur throughout the set. This step has not been implemented in the prototype system.

CONCLUSION

A method has been presented whereby a domain theory may be used to explain an example even when no complete explanation can be generated. The “best” explanation that can be derived is then used to extend the domain theory. A prototype system has been implemented that is able to extend its domain theory this way. It currently runs on a small experimental set of examples in the “cup” domain [Mitchell et al, 1986], with a domain theory of about thirty rules.

The goal of this method is to increase the explanatory power of the domain theory, rather than to acquire a specific way of recognizing instances of the goal concept. The latter approach has been taken by Pazzani [Pazzani, 1988], who faces a similar problem of selecting the best complete explanation of an example, in order that the explanation-based learning process derive the most accurate rule. Pazzani’s heuristics for selecting the best explanation are very similar to the relative measures given above.

A useful extension to this work would be to generalize the right-hand sides of a rule after a number of them have been acquired. This approach has been taken by Danyluk [Danyluk, 1989], whose preliminary results indicate that generalization over multiple examples is useful in eliminating irrelevant terms. However, such generalization is not always necessary for the rules to be useful. Since plausible explanation generation does not require all rule antecedents to be satisfied, newly acquired rules may be used in explanations before they are completely rid of irrelevant terms.

References


AUGMENTING DOMAIN THEORY FOR EXPLANATION-BASED GENERALISATION

Kamal M. Ali, Dept. of Information and Computer Science (ali@bonnie.uci.edu)
University of California, Irvine, CA 92717

INTRODUCTION

Explanation-based generalisation (EBG) is a form of learning which uses a strong domain-theory to constrain the search for generalisations of the concept that is being learned. EBG attempts to explain why the example is a positive instance of the target concept. As the domain theory already has general rules for proving the concept, the benefit of the method is its ability to operationalise and generalise the explanation (Mitchell, 1985; DeJong, 1986; Kedar-Cabelli, 1987). This paper addresses one of the shortcomings of EBG: namely, that EBG requires the domain theory to be complete.

For this system, examples consist of ground predicates (some of which may be irrelevant). If the system can construct a proof, an operationalised definition of the target concept is formed. Otherwise, new rules are hypothesised to complete the explanation. Failure to generate an explanation could be due to the following reasons:

1. Missing rule connecting a leaf in the incomplete domain theory to predicates used to describe examples (special case of a missing disjunct).
2. Missing disjunct for some concept (when some rules already exist for that concept).
3. Some conjunctive rule is too specific (extra conjunct).
4. Some conjunct is too specific (too low in is-a hierarchy).

The system only learns rules of the first two kinds. Missing conjuncts cannot be detected because the rule is overly general so it will always generate an explanation whenever the correct rule would have.

AUGMENTING THE DOMAIN THEORY

The first step taken by the system when an example is presented, is to try to prove the nominated root concept (which is kill in the example below). A leaf node in the proof is labelled failed if it cannot be deduced from the example. Usually, the proof is partially constructed when the first failed node is encountered, so some of the arguments in the failed predicate have been bound to other predicates. This is important because any tentative rules formed from this proof then contain these restrictions between the arguments. Consider the incomplete domain theory (variation on Mitchell, 1985) in Fig. 1 which is missing the rule:

\[ \text{possess(Person, Weapon)} \leftarrow \text{buy(Person, Weapon)} \]

In trying to prove \( \text{possess(Person, Weapon)} \), \( \text{Person} \) is bound to \( \text{john} \), and later \( \text{Weapon} \) is bound to \( \text{gun1} \), so the failed predicate does not have any unbound arguments. The predicate \( \text{possess(john, gun1)} \) is flagged as failed and the proof construction process resumes from that point.

\[ \text{kill(P,P)} \leftarrow \text{depressed(P) and possess(P,Weapon) and weapon(Weapon)} \]
\[ \text{depressed(Person)} \leftarrow \text{hate(Person,Person)} \]
\[ \text{weapon(Weapon)} \leftarrow \text{gun(Weapon)} \]
\[ \text{hate(john,john), buy(john,gun1), weapon(gun1), aunt(john,aunt1), house(house1)} \]

Figure 1: Incomplete domain theory

After failing to prove the concept, a rule is hypothesised from relevant predicates describing the example to the failed concept. A predicate is potentially relevant if it shares instantiated arguments with predicates used in the proof, or it shares arguments with other relevant predicates.

A tentative rule is then formed for the failed concept. As disjunctive concepts are allowed in this framework, it is possible that the proof failed because a disjunct is missing for some non-operational concept. So, the program not only hypothesises rules connecting predicates in the example to leaf concepts, but hypothesises extra disjuncts for concepts higher up in the tree. In general, it is possible that a disjunct is missing for any concept on a path from a failed leaf concept (in the best partial proof) to the root concept.

When a concept fails again, the tentative rule for that concept is updated by intersecting the set of predicates in the old rule with the relevant predicates from the current example. So if the pre-existing...
When a concept fails again, the tentative rule for that concept is updated by intersecting the set of predicates in the old rule with the relevant predicates from the current example. So if the pre-existing tentative rule was
\[ c1(A_1, A_2) \leftarrow a(A_1, B), b(B, A_2), c(A_2). \]
and the tentative rule for this example is
\[ c1(X, Y) \leftarrow c(Y), d(Y, X), a(X, Z), e(Z). \]
the resultant tentative rule is formed from intersecting the sets of predicates to yield:
\[ c1(X, Y) \leftarrow a(X, Z), c(Y). \]
If no predicates are in common, or there is no way to find a consistent set of bindings between the two tentative rules to be merged, the tentative rule is retracted from the database. Such a situation indicates that the concept is disjunctive.

When a tentative rule stabilises, it is asserted into the rule database and may be used to form explanations. Currently, a rule is considered stabilised if for 4 consecutive failures of the target predicate P, no alteration to the set of predicates in the antecedent was necessary. If the root concept fails but P was not part of the best explanation, that example has no bearing on the tentative rule for the target predicate. If P is part of the best explanation, other provable predicates in the proof constitute a context for the P. The hope is that the context is a constraint on the subset of examples where P fails, thus allowing the rule to stabilise. Tentative rules are only formed when the proof has been made as complete as possible and the best proof has been selected. Proofs which use the largest number of the facts presented in the example are preferred. If there is a tie, the proof with the least number of failures is chosen.

EXPERIMENTAL ANALYSIS

In order to test this method, the instance language was kept constant to allow random generation of examples and incomplete domain theories. In the test runs, the concept kill was the root target concept. The system learns for each predicate that fails and is part of the best partial explanation.

The incomplete domain theories are generated by deleting between 1 leaf node and 50% of the leaf nodes. On average, 5 predicates were needed for a positive example description and 3 irrelevant predicates from a pool of 27 were chosen to complicate the learning process. The complete concept description (domain theory)
has 18 predicates. Figure 2 shows the percentage of test cases correctly classified by 3 theories after various stages of learning. The curve with black circles corresponds to a run with the 3rd theory with twice as many irrelevant predicates in training examples. One training set and one testing set were randomly generated, and used for all runs. At certain intervals the theory was tested on the 50 test examples. The curve with filled boxes corresponds to the strongest theory. The percentages are sensitive to missing rules and when they do increase, do so in large increments. For example, note that if the root concept is a disjunction of 2 conjunctions and one of the conjuncts in the second disjunction has only one missing rule, errors of omission will occur 50% of the time. \( pdt_2 \) (open boxes) was obtained by deleting 2 disjuncts in \( pdt_1 \). In \( pdt_3 \) (open circles), approximately 75% of the leaf nodes were absent.

PROBLEMS AND FUTURE RESEARCH

The domain theories that are output by this process contain some deficiencies. For example, the following infrequently used operationalised definition of \( \text{kill}() \) was learned:

\[
\text{kill}(X, X) \leftarrow \text{hate}(X, X), \text{bridge}(Y), \text{jump}(X, Y).
\]

Another problem is that of invariant conjuncts. In the training set the predicate \( \text{drink.alcohol} \) occurs in every example that \( \text{psychotic} \) occurs because \( \text{drink.alcohol} \) and \( \text{psychotic} \) have a common conjunctive ancestor node. The rule we want to learn is:

\[
\text{mentally.ill}(X) \leftarrow \text{psychotic}(X).
\]

but the rule that is learned is:

\[
\text{mentally.ill}(X) \leftarrow \text{psychotic}(X), \text{drink.alcohol}(X).
\]

The following rule is also learned:

\[
\text{take.pills}(X) \leftarrow \text{drink.alcohol}(X).
\]

because the ground predicate \( \text{drink.alcohol} \) and the intermediate predicate \( \text{take.pills} \) are conjuncts in a disjunct for \( \text{kill} \), so whenever \( \text{take.pills} \) fails, \( \text{drink.alcohol} \) is present in the example.

CONCLUSION

The system is capable of learning disjunctive structured concepts one disjunct at a time. It can acquire new rules, missing disjuncts and cope with an expanding instance language. It also forms operationalised definitions of the concept to improve performance. It learns best when several concepts with shared sub-concepts are learned simultaneously.

Disadvantages of this approach comprise an inability to learn intermediate concepts and missing conjuncts. It also forms overly specific rules in contexts where one conjunct is invariant. The system is not discriminating enough when linking predicates to find new rules. Some form of background knowledge as in OCCAM (Pazzani, 1988) would be useful. An ability to detect and retract erroneous rules would also be useful.

More discriminating forms of generalisation would also allow learning of more complicated concepts. Some possibilities are generalising using an is-a hierarchy, and generalising to a range of integers. Negative examples would be useful for detecting over-generalisation.

References


