Learning Efficient Recognizers for Analytically Derived Concepts

Paul E. Utgoff
Andrew G. Barto

COINS Technical Report 86-45

September 9, 1986

University of Massachusetts
Amherst, Massachusetts 01003

Contents

1 Project Summary

A recent development in the field of Machine Learning is the advent of Explanation-Based Learning, in which a theory of the domain is employed to build an explanation of why an instance is or is not an example of a concept to be learned. An explanation is a generalized expression that shows how an instance is determined to be an example of the concept. An explanation can be used for classification by testing whether the instance is covered by the explanation. This is inefficient for the purpose of classification, however, because a classification task only requires an answer, not a justification.

The proposed research will explore how to develop an efficient recognizer for a concept defined by an explanation. A recognizer is a predicate that is true if its argument is an instance of the concept, false if its argument is not an instance of the concept, and unknown otherwise. This project addresses the bottleneck problem in Explanation-Based Learning: How can an inefficient yet correct explanation be converted to an efficient recognizer?

The investigators will jointly pursue two avenues. One direction is the study of symbolic methods from the mainstream of Artificial Intelligence. The second direction is the study of numerical pattern classification methods, including recently developed connectionist learning methods. Can a hybrid system be built in which symbolic methods and connectionist methods complement one another?

This report is the technical portion of a research proposal submitted to the National Science Foundation in August, 1986. The authors welcome comments and criticisms.

2 Project Description

Machine Learning is the field of study in which methods that enable automatic improvement are developed and evaluated. Mechanisms that permit performance improvement provide a learning capability. Learning methods are useful because they shift the burden of improving system performance from the system developer to the system itself. Furthermore, a system that improves its own performance has the potential to improve in ways not foreseen by the system developer. Learning theorists have delineated two major types of learning: inductive and deductive. The proposed research will examine how these two types of learning can be profitably combined.

Inductive methods are used to infer, from observed examples and counterexamples, a concept that includes all examples of the concept and excludes all counterexamples of the concept. Such an induced concept is a generalization of the data. Through the 1970s and early 1980s, symbolic inductive learning methods were popular [40,26,38,21]. Interest in symbolic inductive methods has waned, in part due to the bias problem [36]. Among those hypotheses that are consistent with the data, which should be induced? If the correct hypothesis, called the target concept, is to be chosen on a non-random basis, then some kind of guidance must be followed. Such guidance, however it is represented, constitutes *bias*. The inductive techniques do well when and only when the bias is well chosen. Learning mechanisms that are entirely inductive remain strongly dependent on an outside source, the program author or user, to provide an appropriate bias.

Dependence on bias is also a major issue for the inductive learning methods that have been developed in the field of pattern classification and automatic control. Within AI, numerical pattern classification research went out of fashion in the late 1960s but is currently undergoing a revival as these and related methods find application in connectionist systems for parallel computation and cognitive modelling [9,11,32,20]. The most significant recent advances in this area directly address the bias problem in the framework of numerical inductive learning procedures [33,12,3].

Deductive methods are used to infer, from a theory of the domain and a statement of the goal, a concept that is consistent with exactly those instances that satisfy the goal. The theory is the set of axioms and rules that define the problem domain and the legal operations within the domain. The result of the deductive step is an explanation, which is a generalized expression that shows how an instance is determined to be an example of the concept. A deduced explanation can also be seen as a specialization of the theory, showing how to prove whether an instance is satisfies the goal concept. The distinction between inductive methods and deductive methods lies in the inference mechanisms and their requirements, not in the target concept to be inferred. In the early 1970s, deductive learning methods were briefly explored [10,39]. Deductive methods are enjoying a renaissance, at least in part because it has been learned that in many cases it is easier to specify a correct theory than it is to specify a correct bias. The methods have been extended and now form the basis for Explanation-Based Learning.

The major bottleneck in using deductive methods for learning is that the inferred explanations are inefficient when used to classify instances. Decision strategies that use the results of learning are typically based on determining the class(es) of an instance. If an efficient recognizer can be derived from the correct but inefficient explanation produced by the deductive process), then the usefulness of deductive learning methods will be significantly advanced. A recognizer only needs to classify the instance, not explain or prove why the instance is or is not in the concept. The research

will address exactly this problem, how is an efficient recognizer generated.

2.1 Objectives

The principal advantage of using deductive methods for learning is that the learning task at hand is solved directly by deducing the solution from the theory and the goal. Although the deduction process produces an explanation, showing how to prove whether an instance satisfies the specified goal, an explanation is inefficient as a classifier. For classification purposes, a proof (and its overhead) is not needed—one simply wants the answer. The explanation should be converted to an efficient recognizer that can be used inexpensively for each classification task. Hence, a second step is needed to make deductive learning useful: construct an efficient recognizer for exactly those instances covered by an explanation. This second step can be cast as a subproblem: Learn to answer *quickly* whether an instance is covered by the explanation. This leads to the first objective of the proposed research:

Objective 1 *Identify how to form learning subproblems transparently from deductively inferred explanations.*

A *transparent* operation is one that is not seen by the user. What mechanics are necessary to generate such subproblems and incorporate the results or partially determined results?

Given that the subproblems can be automatically defined from deductive methods, the theory, and the goal, it is necessary to generate an efficient recognizer so that the learning subproblem is solved. This leads to the second objective:

Objective 2 *Identify methods that convert an explanation into an efficient classifier.*

That is, given that the subproblem is generated and the result or partial result can be used, how is the subproblem actually solved?

One avenue is to explore methods for symbolic transformations that convert the explanation into an efficient recognizer. One such method is to construct a mechanism that intersects the goal from a previous step with the range of an operator and then computes the preimage of that intersection through the operator [36]. This is potentially expensive but results in an efficient recognizer.

A second avenue is to explore inductive techniques, symbolic and numerical, including recently developed connectionist techniques. For each subproblem, the explanation can serve as the trainer. Whenever it is necessary to classify an instance according to a concept, the explanation can be used to decide the correct answer for the classification task. Given that an instance and its correct classification are then available, a reliable training instance is available for training the inductive method.

These two avenues, symbolic transformation and inductive learning, and the two varieties of induction, symbolic or numerical, provide a rich set of methods for approaching the problem of learning efficient recognizers. This leads to the third objective:

Objective 3 *Identify the relative strengths of reduction methods, symbolic induction methods, and numerical induction methods, to determine how to combine their individual strengths for the purpose of constructing efficient recognizers.*

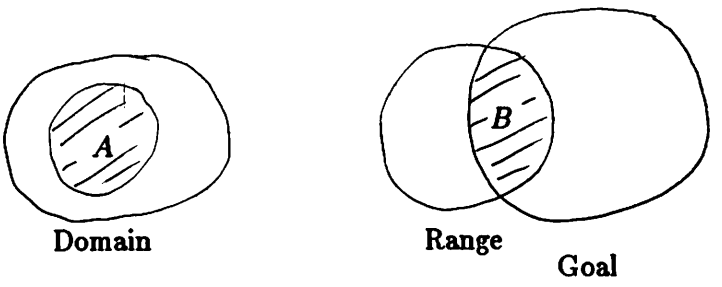


Figure 1: Preimage through an operator

To what extent can complementary capabilities of these various methods be combined? How can the the strength of one compensate for the weakness of another? The strategy of investigating a collection of methods, particularly symbolic and numerical, within the context of a single system forces a study of the relative strengths and weaknesses. This should be fruitful, particularly since the researchers in symbolic methods and the researchers in numeric methods have tended to discount each other, with the consequence of poor communication.

2.2 Issues

This section describes the major issues that motivate the proposed research.

2.2.1 Effectiveness of Deductive Learning

An example from [36,37] illustrates the effectiveness of deductive learning. The example shows how exploiting a domain theory can lead directly to a correctly generalized concept. The example also shows generation of an efficient recognizer through symbolic transformations.

The deductive mechanism used here is a simplified form of Waldinger's "goal regression" [39] that is also equivalent to Dijkstra's method of computing a "weakest precondition" [6]. The method is expressed by the recurrence relation:

$$C_n = \{x | Goal(x)\}$$

$$C_{n-1} = preimage_{op_n}(C_n \cap range(op_n))$$

That is, from a sequence of operators $op_1, \ldots, op_n$ that leads to a goal state, the preimage of any tail of that sequence can be deduced. As illustrated in Figure 1, the preimage of a set B through a given operator is the set A such that application of the operator to any element of A will produce a state in B [30]. Formally,

$$preimage_{op_n}(B) = \{a | op_n(a) \in B\}$$

Two kinds of learning can take place. First, the preimage of an operator sequence is learned by deducing and saving it. Second, new terms are synthesized, e.g. *even*, as part of generating an efficient recognizer, so that a preimage can be efficiently described.

The method is applied to the solution of $\int cos^7(x) \cdot dx$. It is solved with the following steps:

$$\int cos^7(x) \cdot dx$$

$$\int cos^6(x) \cdot cos(x) \cdot dx$$

$$\int (cos^2(x))^3 \cdot cos(x) \cdot dx$$

$$\int (1 - sin^2(x))^3 \cdot cos(x) \cdot dx$$

$$\int (1 - u^2)^3 \cdot du, u = sin(x)$$

The problem is completed by multiplying out the polynomial, rewriting the integral of sums as a sum of integrals, and solving each integral trivially. A key step is that multiplying out the polynomial can be done only when the exponent (3 in this case) is an integer. Computing the preimage from the last step shown above, where k is the set of integers, the regression proceeds:

$$\int poly^k(u) \cdot du; u = f(x) \tag{1}$$

The operator for the next step is $\int f^c(g(x)) \cdot g'(x) \cdot dx \rightarrow \int f^c(u) \cdot du; u = g(x)$. The intersection of the range of the operator and (1) is $\int poly^k(u) \cdot du; u = f(x)$. Computing the preimage gives:

$$\int poly^k(f(x)) \cdot f'(x) \cdot dx \tag{2}$$

For the next step, the operator is $cos^2 \rightarrow 1 - sin^2$. The intersection of the range and (2) is $\int (1 - sin^2(x))^k \cdot cos(x) \cdot dx$. The preimage is:

$$\int (cos^2(x))^k \cdot cos(x) \cdot dx \tag{3}$$

The next operator is $f^c \rightarrow (f^2)^{c/2}$. Intersecting the range and (3) gives $\int (cos^2(x))^{even/2} \cdot cos(x) \cdot dx$. This is an important step; the $c/2$ in the range must be constrained to be an integer. The program constructs a new term, here called *even*, defined as a predicate that is true if and only if the result of dividing its argument by 2 is an integer. The preimage is:

$$\int cos^{even}(x) \cdot cos(x) \cdot dx \tag{4}$$

Finally, the last operator is $f^c \rightarrow f^{c-1} \cdot f$. The intersection of the range and (4) results in $\int cos^{odd-1}(x) \cdot cos(x) \cdot dx$. The preimage is:

$$\int cos^{odd}(x) \cdot dx \tag{5}$$

It is deduced that the operator sequence leads to a goal when the initial state is of the form $\int cos^{odd}(x) \cdot dx$. The definition is efficient because the *preimage*() and *intersection*() functions have been reduced symbolically to the corresponding structural descriptions. Along the way, the two new concepts *even* and *odd* were identified and defined as well.

The learning was deductive. From the theory (the operators and the concept description language), the goal (produce an expression without an integral), and the solution sequence, the exact preimage was deduced. An inductive learning algorithm would have needed multiple examples and counterexamples for the sequence, together with a fortuitously chosen bias to have achieved the same conclusion, that the operator sequence solves problems of the form $\int cos^{odd}(x) \cdot dx$.

2.2.2 Need for an Efficient Recognizer

The greatest stumbling block in exploiting deductive learning techniques is that a derived explanation is inefficient as a recognizer. This is because an explanation is a derived proof procedure that can be executed to prove whether an instance satisfies the goal. Evaluating the proof procedure is an expensive method of classifying an instance. It is preferable to construct a recognizer that compiles the proof procedure. That is, an instance should be classifiable based on structural features that are predictive of whether it satisfies the goal.

The problem comes directly from the definition of preimage. Something is in the preimage A of an operator if, upon applying the operator, the produced state is in the target range B. The definition requires applying the operator, but this must be avoided if the concept is to be used in a predictive way to decide whether to apply the operator. A concept description is search-dependent if one or more successor states must be generated for matching purposes. A description is search-free if it is not search-dependent.

Consider in more detail why the definition of preimage requires application of the operator for matching purposes. Suppose that a concept A has been deduced through goal regression and is defined as:

$$A = preimage_{op_n}(C_n \cap range(op_n))$$

To determine whether an instance x matches the concept A, it is necessary to expand and evaluate A with respect to x. That is:

$$x \stackrel{?}{\in} A$$

$$x \stackrel{?}{\in} preimage_{op_n}(C_n \cap range(op_n))$$

$$x \stackrel{?}{\in} \{a | op_n(a) \in (C_n \cap range(op_n))\}$$

$$x \in A \iff op_n(x) \in C_n$$

The definition is correct but not useful because it is necessary to apply op_n and to test whether the result is in a given set.

The problem of efficient recognition and knowledge compilation has deep roots in Artificial Intelligence. It has been observed many times that novices tend to search while experts tend to recognize, e.g. [18]. The compilation of knowledge, so that patterns can be quickly recognized and associated outcomes quickly indexed, is a basic building block of high performance systems.

The proposed research will explore methods that automate the task, not only of learning how to recognize quickly, but also in identifying what should be recognizable.

2.2.3 Efficient Recognizer via Symbolic Methods

Symbolic methods for generating concept descriptions have been studied extensively since the late 1960s [22,23]. For the task of generating an efficient recognizer, the methods fall into two broad categories: reduction methods and inductive learning methods.

Reduction methods apply transformations that preserve the correctness of an expression while making it simpler. The symbolic integration example above reduces the *preimage* and *intersection* expressions of the explanation by evaluation. Because the goal concept was an algebraic expression without an integral, it was possible to reduce the explanation to $\int cos^{odd}(x) \cdot dx$. This is one way of generating an efficient recognizer or description. How does that method apply in general? What other kinds of reduction systems can be employed?

There are two classes of inductive learning methods: symbolic and numerical. To use an inductive learning method, one could train a recognizer. That is, the explanation can be used as a correct classifier to generate training instances for training the recognizer. This idea is developed below in section 2.5.1. Among symbolic methods for inductive learning, Mitchell's Candidate Elimination Algorithm [26] and Michalski's AQ algorithm [21] are known best. Are such algorithms suitable for the task of learning a recognizer? How are the problems of bias and appropriate representation to be handled? Numerical inductive learning methods are discussed in the next section.

2.2.4 Efficient Recognizer via Numerical Pattern Classification

The field of pattern classification provides a great variety of methods for constructing recognizers on the basis of training sets consisting of examples and counterexamples of concepts [7]. These inductive learning techniques have not found widespread use in AI for several reasons. First, these methods are largely restricted to instance description languages that are property or feature lists having restrictive expressive power compared to more structured symbolic descriptive languages. Second, the class of learnable concepts is restricted, depending, for example, on parameterized discriminant functions that introduce inflexible *a priori* bias. Third, the hypothesis descriptions formed by these methods are not readily interpretable by humans, being parameterized mathematical expressions. A final reason that numerical pattern classification methods are not used in AI is that these two research traditions have remained out of contact, each involving different groups of researchers.

One of the aims of the research proposed here is to foster interaction between these theoretical traditions by applying numerical pattern classification methods to the problem of converting deduced explanations into efficient recognizers. One major reason for attempting this is that although the hypothesis descriptions produced by numerical methods are not readily interpretable by humans, they are always search-free and so can be evaluated quickly. One might think of the evaluation of a numerically expressed concept as an automatic reflexive skill.

The other reason we think this approach is appropriate is that the limitations of numerical pattern classification listed above are not serious in the proposed application—being compensated

for by the strengths of the symbolic components of the system. The limited expressive power of property or feature lists need not be a serious problem because numerical techniques will be applied to learning *subproblems* generated by the deductive part of the system. Any structure not easily representable by property lists can be handled by the deductive component. The most well-known limitation that numerical methods can learn only restricted classes of concepts (e.g., perceptron learning is restricted to predicates that are linearly separable in terms of some given set of features) is no longer so serious due the availability of learning methods recently developed by connectionist researchers (discussed below). Finally, that numerically expressed concept descriptions do not lead to humanly understandable explanations is of no consequence here since the symbolic explanations remain available.

Connectionist researchers have recently developed several learning algorithms, that are examples of numerical pattern classification methods. Connectionist research represents the revival of neural-network approaches to computation. Such networks consist of large numbers of relatively simple processing units interconnected by weighted links. The recent intensification of interest in this approach is attributable in part to the potential of increased computation speed through massive parallelism, the utility of associative processing through the use of distributed representations, and the utility of learning methods that take the form of rules for changing connection weights through experience [9,32,20,11].

One way to broaden the class of hypotheses learnable by connectionist systems is to develop learning algorithms capable of adjusting the weights within multi-layer networks. A multi-layer network of elements that incorporate appropriate kinds of nonlinearity can be specified to implement any desired mapping from the set of possible input vectors to the set of output vectors. The key difficulty, however, in forming such a network by a learning process is to correctly assign credit for overall network performance to the network's component units. Despite considerable effort, only recently have effective learning methods been developed. Although several algorithms have been studied that apply to very restricted network architectures, three methods stand out in terms of theoretical justification and generality: the Boltzmann learning procedure [1,12]; the Associative Reward-Penalty method [2,3]; and the Backpropagation method [33].

Boltzmann learning is a stochastic method that has been demonstrated on a number of problems but requires excessive time compared to other methods, and we do not propose to experiment with this method. We will restrict attention to the Associative Reward-Penalty and Backpropagation methods. These are capable of learning to recognize complex properties on the basis of exposure to examples and counterexamples. Whereas the old network learning algorithms (e.g., the perceptron [31] and adaline [41]) adjust only the weights of the network's final layer, these new methods can correctly adjust weights of units within a network ("hidden units") to specify new features that are useful for reducing overall network error over the training set and for generalizing to novel inputs. As a consequence, the linear separability restriction is effectively removed. A shortcoming of these methods, however, is that they can take thousands, or even hundreds of thousands, of training examples to learn complex properties. Although this shortcoming remains significant, it is of less concern in the application being proposed here because while learning occurs, the correct but inefficient deductively derived explanations can be used for reliable classification.

2.3 Significance

Techniques that permit a machine to improve are essential in the long term. A lesson from developing expert systems is that extracting and codifying knowledge in a useful way is an arduous process. Not only are there difficulties in codifying rules, but it is often unnatural for a person to try to express his knowledge as a consistent set of rules. Within the field of Machine Learning, deductive techniques have shown great promise for learning because they have the ability to make good use of available information. Such techniques ought to be applicable whenever a theory or partial theory of a domain is available. The proposed work will investigate promising methods that would advance Explanation-Based Learning. The rest of this section discusses significant aspects of the work on which progress is expected.

2.3.1 Deriving Explanations

Deductive methods permit correct learning by identifying concepts that achieve a goal. The ability to use the available theory and goal is consistent with the tenet in AI that one wants to use available information as completely and effectively as possible. More investigation of how explanations can be derived from a theory and a goal is needed.

2.3.2 Converting Explanations to Recognizers

The single most important unsolved problem in using deductive learning methods is that of converting inefficient explanations into efficient recognizers. This problem depends on the ability to identify structural features that are predictive of whether an instance will satisfy the goal. Two approaches are to be studied jointly by the investigators, one symbolic, the other numerical.

2.3.3 Transparent Formulation of Subproblems

The proposed system's ability to formulate learning subproblems and use the results in a transparent way is new. Usually, learning methods are applied to specific problems formulated by a scientist. In the proposed work, such formulation and utilization of the results is conducted transparently in the background. Studying such an approach will shed light on the general problem of how to integrate learning methods into performance systems.

2.3.4 Symbolic Reduction Methods

Symbolic reduction techniques were used successfully in the STABB subprogram of the LEX system [27,37]. The proposed research will build on this earlier work, broadening the techniques as necessary. Methods that reduce expressions to canonical form may prove useful [13].

2.3.5 Combining Symbolic and Connectionist Approaches

The overall approach of having two investigators combine two classes of learning methods (symbolic and numerical) to solve a single kind of problem within a single system has major benefits. First, a concerted effort to combine the strengths of two approaches will result. Second, the inves-

tigators will necessarily form a bridge between the two schools of thought.

2.3.6 Connectionist Learning

The proposed use of connectionist learning methods will provide additional tests for these methods which will address the issues of how initial network architectures should be chosen (e.g., How many layers? How many hidden units?), and how these methods scale to larger problems. It will also be possible to examine possible relationships between the degree of inefficiency of an explanation and its learning difficulty for a network. Will the explanations that are least efficient for classification correspond to those for which it is most difficult to learn a fast recognizer; or will the opposite be true?

2.4 Related Work

Deductive learning mechanisms were employed in STRIPS [10]. Preconditions for sequences of operators are deduced as conjunctive predicates. Conjunctions can become very long for describing certain specializations. Waldinger pioneered goal regression not for learning but for problem solving. He used it to deduce a sequence of operators that would solve the problem. Dijkstra's "weakest precondition" method is a form of goal regression [6]. Dijkstra specifies the desired postcondition (goal) of a program segment (theory) and then regresses the postcondition through the program segment to determine the weakest precondition for which the segment is guaranteed to produce the desired postcondition.

More recently, work in the area of Explanation-Based Learning has made use of the basic principal that; given a theory and a goal, it is possible to deductively derive an explanation that describes those states for which an operator sequence leads to a goal. A survey of current work in this area is given in [28], including forms of Explanation-Based Learning that do not use operator sequences and instead use the theory to guide the generalization process.

Most of the work in Explanation-Based Learning focuses on the deduction engine and the issues surrounding generation of an explanation. Many of the problems studied have remained small enough that efficiency has been a secondary issue. Continued progress in deductive forms of learning will increase the need for techniques that address the problem of producing efficient recognizers.

Dejong [4] has examined how to infer the causes and effects for kidnapping based on an instance of kidnapping and a theory of human motivations based on love or greed.

Utgoff [36,37] showed how a theory of problem solving (the operators), an instance of a solution trace, and a statement of the goal can be used to deduce the domain of an operator sequence that leads to a goal. That work also showed one way to generate efficient recognizers for deduced concepts. This included creation of new descriptive terms.

Kelly's [16] CRITTER system reasons with behaviors or specifications of digital circuits. By propagating specifications backward through the circuit, the system deduces the domain of acceptable input values. The system does not learn however.

Mitchell [27] showed how the goal criterion can be represented declaratively and plugged in to the

explanation generator. He illustrated two different criteria and the resulting different explanations for a single example from symbolic integration.

Keller [14] discussed the problem of efficient recognition and listed 10 symbolic reduction rules for converting an explanation to an efficient recognizer. More recently, work for his Ph.D. thesis explores methods for reducing an explanation by noticing subexpressions in the explanation that tend to evaluate to the same single value whenever the explanation is used for the purpose of classification [15].

Minton [24] experimented with Explanation-Based Learning in several simple games, notably Tictactoe. He observed that although correct concepts were deduced, the system bogged down hopelessly due to the complexity of the descriptions. He advocated reducing explanations to more efficient recognizers. His Ph.D. thesis work examines methods for reducing explanations by symbolic simplification[25].

Gallant [8] has proposed an expert system architecture which combines symbolic and connectionist approaches. He shows how an explanation for a network decision can be generated by examining the network's connection weights. He does not use deduced explanations as a trainer for the network.

Other connectionist research that combines aspects of symbolic and connectionist approaches is that of Touretzky [35] and Derthick [5]. Touretzky investigates how trees and stacks can be implemented in connectionist systems. Derthick proposes to study a knowledge representation system implemented by a connectionist network. In neither of these projects are symbolic learning and connectionist learning combined.

2.5 Initial System Design and Methods

This section shows the proposed system architecture, discusses problem domains, and illustrates with an example.

2.5.1 Architecture

The proposed initial architecture is shown in Figure 2. It has three distinct modules; a problem solver, an explanation generator, and a multiple concept classifier. The problem solver solves a problem, and passes the solution trace to the explanation generator. The explanation generator uses the solution trace, theory, and goal to produce an explanation of why the solution path leads to the goal. The explanation is passed to the classifier, which sets up a learning subproblem—to generate a recognizer for efficient classification of instances according to the given concept. Recall that an explanation is inefficient for classification purposes because it must do the work of proving whether an instance is on the path to a goal.

The problem solver can ask the classifier to classify a given instance according to a named concept. The response returned by the classifier will be used by the problem solver in its decision making. Whenever the classifier is called upon to classify an instance, it must take into account the total time it has been provided before it must answer, as well as the reliability that is needed. That is, the classifier must decide whether to use the current version of its efficient recognizer (which

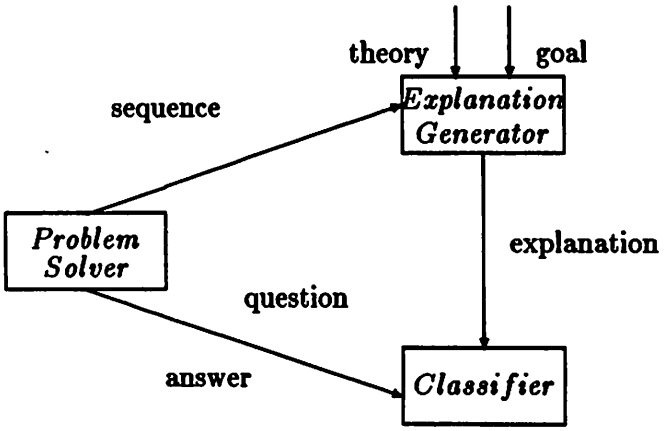


Figure 2: System Architecture

may not yet be 100% reliable), use the inefficient but correct explanation, or answer with "I don't know". The remainder of this section sketches how the subproblem of learning a recognizer can be approached.

Each concept is a complex object managed by the classifier module. The concept has the outward appearance of a predicate, returning *true* if its argument is an example of the concept, *false* if it is a counterexample, and *unknown* otherwise. The internal representation of the concept is that of a self-contained learning problem. There are two internal definitions of the concept, and a control structure for selecting which definition to use. One definition, called the slow definition, is the explanation that was deductively derived. It is inefficient but correct. The second definition, called the fast definition, is the pattern-based structural recognizer.

If inductive methods are being followed, then the slow definition serves as the oracle/trainer for the fast version. That is, if the concept (predicate) is to determine the classification of an instance so that it can return a value, it is possible (barring time constraints) to determine the correct classification by using the slow definition. Given the instance (predicate argument) and the correct classification, it is possible to then train the fast definition. The evaluator of the predicate does not see this internal activity within the concept object. The only externally observable variation in evaluating the predicate is the cost of evaluation. The predicate is probably more expensive (slower) whenever the slow definition is evaluated.

A possible control structure for a concept is the following: If some number of instances N have been observed, and the most recent M instances were correctly classified by the fast definition, then only use the fast version to determine the classification. Choice of N and M depend on the efficiency of the learning algorithm. Otherwise evaluate the slow definition, train the fast definition with the correct answer, and return the slow version's answer as the predicate value.

One weakness with this simple sketch is that as soon as a concept switches over to the fast definition, it never again checks itself or does further training with the slow version. An improvement would be to resume training whenever an unexpected misclassification is detected.

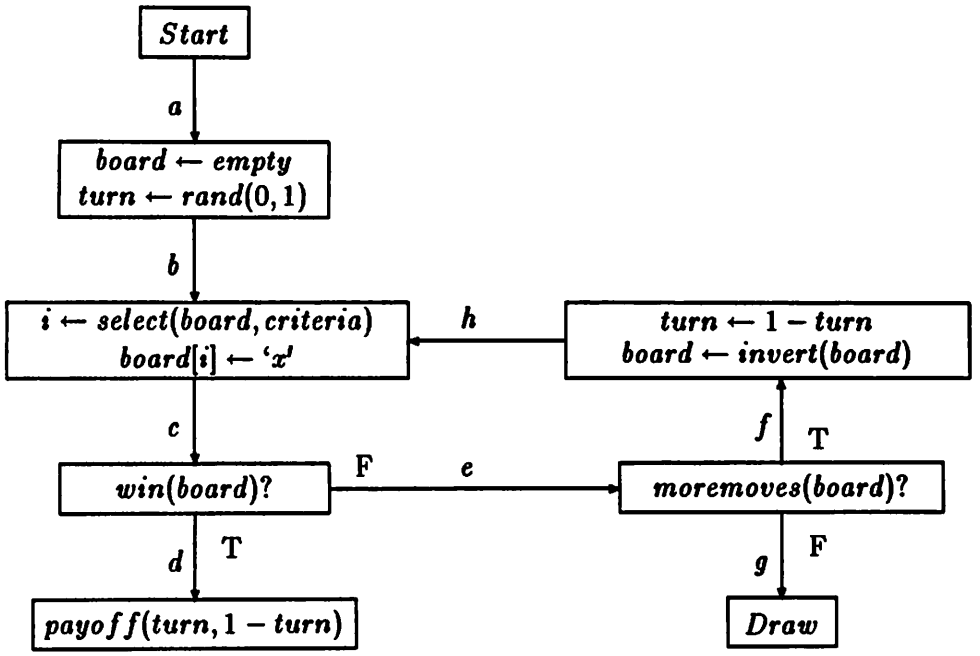


Figure 3: Partial Theory for TicTacToe

2.5.2 Problem Domains

To test and develop the ideas, a simple domain will be briefly investigated. This will permit rapid exposure of issues and verification of the methods. Following this, a larger problem-solving domain will become the vehicle for study. Any domain in which a theory or partial theory exists, even a naive theory, is a candidate for the techniques to be developed in the proposed research.

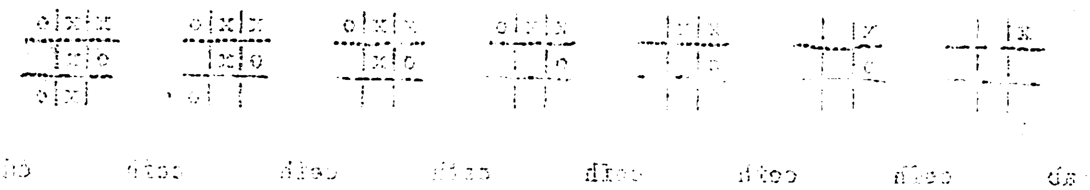
2.5.3 Example

This section illustrates the issues and mechanics by sketching how the system might work for Tictactoe. The rules of the game define the theory (legal operations). Actions that obey the rules are explained and admissable in the theory. The rules of the game also include the definition of a won game, providing a definition of the goal. Figure 3 shows a partial theory of TicTacToe. The partial theory is complete when the additional ground level definitions for win, own, the rows, and the goal are included.

Given the definition of the goal, and the statement of the theory, it is possible to deduce the preconditions at each step so that a desired sequence of transitions (operators) in the state diagram (theory) can be forced. Note that particular board positions are important only to the extent that they determine transitions. Goal regression deduces preconditions that are sufficient for following the sequence of transitions in the forward direction to the goal.

Goal regression can proceed by regressing the goal backward through the theory. When more than one transition could have preceeded a given step, it is necessary to consider all such predeces-

…ors, as indicated by the definition of preimage. In many cases the preconditions of such a given transition become empty (unsatisfiable). For example, the sequence abcd can never occur, so b will never precede c when the cell is exactly cd. Such conclusions can be difficult to prove mechanically. A simpler approach is to regress the goal through the sequence of transitions that occurred in an actual game. This has the effect that a preimage may be more specific (considering a single predecessor instead of all possible predecessors) than is necessary. At the same time however, the regression will uncover the causes and effects of the actual moves. Consider how goal regression would proceed based on the following game:



The sequence of transitions $ab(c/h)ed$ (see Fig. 3) achieves the goal for whichever player moves first. Goal regression proceeds backward over the transitions as follows:

1. The goal is to achieve the maximum payoff. That is, the first priority is to win but, barring that, the second priority is to draw. The example game shows how to achieve the first priority of the goal, a win. This occurs when transition d is taken. The precondition for transition d is that win(board) be true. To achieve the goal of winning, a board position satisfying win must be produced.

2. Transition e precedes d. It is known that the board position will satisfy win. Furthermore, the action preceding e marks the board and therefore changes its state. It is inferred that the criterion used in selecting the given move e is to produce a board state that makes win true. A selection criterion is defined as a global concept criterion^0(board) $\Longleftrightarrow$ win(board) and appended to the list of criteria (initially nil) used in selecting among candidate moves.

3. Transition h or c could precede e but h preceded e in the game, so h is considered. During h it is known $(\exists i(board[i] = 'x') \Rightarrow (criterion^0(board)))$

4. Preceding h was transition f. It is deduced that morethanone() is true.

5. Preceding f was e. During transition e it is known that $\neg$ criterion^0(board) is false.

6. Transition e precedes d. As in step 2 above it is known that the preceding action changed the board state and that a selection criterion can be deduced: criterion^1(board) $\Longleftrightarrow$ $\neg$(criterion^0(board) $\land$ $(\exists i(board[i] = 'x') \Rightarrow (criterion^0(board)))$

Another cycle of goal regression yields a criterion^2 that captures the fact that the opponent cannot make a move that makes criterion^1 false. This is the concept of a fork.

Each criterion that is produced during the explanation process becomes a subproblem for which an efficient recognizer must be learned. Each and every criterion is defined and used transparently. The system learns new concepts by explanation and learns to use the new concepts more quickly through learning of efficient recognizers.

sors, as indicated by the definition of preimage. In many cases the preconditions through a given transition become empty (unsatisfiable). For example, the sequence *abcd* can never occur, so *b* will never precede *c* when the tail is exactly *cd*. Such conclusions can be difficult to prove mechanically. A simpler approach is to regress the goal through a sequence of transitions that occured in an actual game. This has the effect that a preimage may be more specific (considering a single predecessor instead of all possible predecessors) than is necessary. At the same time however, the regression will uncover the causes and effects of the actual moves. Consider how goal regression would proceed, based on the following game:

ab **cefh** **cefh** **cefh** **cefh** **cefh** **cefh** **cd**

The sequence of transitions $ab(cefh)^6cd$ (see Fig. 3) achieves the goal for whichever player moves first. Goal regression procedes backward over the transitions as follows:

1. The goal is to achieve the maximum payoff. That is, the first priority is to win but, barring that, the second priority is to draw. The example game shows how to achieve the first priority of the goal, a win. This occurs when transition *d* is taken. The precondition for transition *d* is that *win*(*board*) be true. To achieve the goal of winning, a board position satisfying *win* must be produced.

2. Transition *c* precedes *d*. It is known at *c* that the board position will satisfy *win*. Furthermore, the action preceding *c* marks the board and thereby changes its state. It is inferred that the criterion used in selecting the given move *i* is to produce a board state that makes *win* true. A selection criterion is defined as a global concept: $criterion^0(board) \iff win(board)$ and appended to the list of criteria (initially nil) used in selecting among candidate moves.

3. Transition *b* or *h* could precede *c* but *h* preceded *c* in the game, so *h* is considered. During *h* it is known: $(\exists i|(board[i] \leftarrow \text{`}x\text{'}) \Rightarrow (criterion^0(board)))$

4. Preceding *h* was transition *f*. It is deduced that *moremoves*() is true.

5. Preceding *f* was *e*. During transition *e* it is known that *win*(*invert*(*board*)) is false.

6. Transition *c* precedes *e*. As in step 2 above it is known that the preceding action changed the board state and that a selection criterion can be deduced: $criterion^1(board) \iff$ $(\sim criterion^0(board) \wedge (\exists i|(board[i] \leftarrow \text{`}x\text{'}) \Rightarrow (criterion^0(board)))$

Another cycle of goal regression yields a $criterion^2$ that captures the fact that the opponent cannot make a move that makes $criterion^1$ false. This is the concept of a fork.

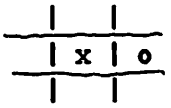
Each criterion that is produced during the explanation process becomes a subproblem for which an efficient recognizer must be learned. Each and every criterion is defined and used transparently. The system learns new concepts by explanation and learns to use the new concepts more quickly through learning of efficient recognizers.

The theory defines what is legal in the game and can serve as the performance system by running it in the forward direction. Whenever a move is to be selected (after transition *b* or *h* in Figure 3), the criteria are evaluated to determine a best move. First a move is sought for which $criterion^0$ would be true. If one is found, then that move is selected. If not, then a move that would make $criterion^1$ true is saught. Recall that the definition of $criterion^1$ makes use of $criterion^0$. By simply using the performance system, the predicates (criteria) are evaluated many times. For example, $criterion^0$ would be evaluated several thousand times in a single game. Normal use of the system will cause evaluation of the concepts. Each time a concept is evaluated, it can be used internally as a training instance for its recognizer (as discussed above). At some point the recognizer will become reliable and can be used to inexpensively evaluate the concept, resulting in improved performance. For a higher level concept, e.g. $criterion^1$, that uses lower level concepts, e.g. $criterion^0$, a speedup in the lower level concept may make evaluation of the higher level concept cheap enough to make a difference in whether it returns a value of *unknown* or a useful value (*true* or *false*).

A connectionist network can be trained to act as a recognizer for a given criterion in the following way. One represents each TicTacToe board pattern as a binary vector; for example, one might encode the contents of each block as a triple of bits indicating, respectively, whether that block contains an x, an o, or is empty. A vector of nine such triples encodes a board pattern. These vectors act as input to a layered network having a single output unit, where the intermediate layers permit the realization of predicates that are not linearly separable. The desired operation of the network is that the output unit should produce high activity if and only if the criterion in question is true for the board pattern encoded by the network's input vector.

Using the Backpropagation learning method, for example, weights on the links interconnecting the units can be learned so that the network's input/output behavior is correct. Each time the criterion is evaluated (by the slow definition) for a board pattern, the coded form of the board pattern is given as input to the network; the network uses its current weight settings to determine its output; and the discrepancy between this output and the desired output (which is known from the slow definition's evaluation) is backpropagated through the network to effect weight changes. If there is a sufficient number of intermediate units in the network, this process will produce a network that is an efficient recognizer for the criterion.

It is interesting that a criterion that is difficult to evaluate using an explanation can be very simple for a network to learn. This is apparent in the case of $criterion^4$. Consider the pattern



which is a win for x if it is x's turn. Note that the explanation of why this is a win for x is expensive, involving as much work as a game tree search. Nevertheless, it very easy to learn to recognize this pattern (and the other three patterns that are equivalent through symmetry). A network can learn this quite rapidly.

It is possible to train a single network to act as a recognizer for all of the criteria by giving it a separate output unit for each criterion. That these units share the same intermediate layers implies that features represented by "hidden units" developed during the course of learning one criterion

would automatically be available to use in learning another criterion. This should facilitate the inductive learning of the criteria.

Acknowledgements

The authors thank Paul R. Cohen for incisive comments on the ideas and presentation. Sharad Saxena provided helpful comments and observations.

3 Bibliography

[1] Ackley, D. H., Hinton, G. E., and Sejnowski, T. J. "A Learning Algorithm for Boltzmann Machines", *Cognitive Science*, vol. 9, 1985, pp. 147–169.

[2] Barto, A. G. and Anandan, P. "Pattern Recognizing Stochastic Learning Automata", *IEEE Transcations on Systems, Man, and Cybernetics*, vol. 15, 1985, pp. 360–375.

[3] Barto, A. G. "Learning by Statistical Cooperation of Self-interested Neuron-like Computing Elements", *Human Neurobiology*, vol. 4, 1985.

[4] DeJong, G. "An Approach to Learning from Observation", *Machine Learning II*, Morgan-Kaufman, 1986, pp. 571-590.

[5] Derthick, M. "A Connectionist Knowledge Representation System", Thesis proposal, Deptartment of Computer Science, Carnegie-Mellon University, 1986.

[6] Dijkstra, E. W. *A Discipline of Programming*, Prentice-Hall, 1976.

[7] Duda, R. O. and Hart, P. E. *Pattern Classification and Scene Analysis*, Wiley, 1973.

[8] Gallant, S. I. "Automatic Generation of Expert Systems from Examples", *Proceedings of the Second International Conference on Artificial Intelligence Applications*, IEEE Computer Society, Miami Beach, Florida, 1985.

[9] Feldman, J. A. (Ed.) Special issue on connectionist models and their applications. *Cognitive Science*, vol. 9, 1985.

[10] Fikes, R. E., Hart, P. E. and Nilsson, N. J. "Learning and Executing Generalized Robot Plans", *Artificial Intelligence*, vol. 3, 1972, pp. 251-288.

[11] Hinton, G. E., and Anderson, J. *Parallel Models of Associative Memory*, Erlbaum, 1981.

[12] Hinton, G. E., and Sejnowski, T. J. "Analyzing Cooperative Computation", *Proceedings of the Fifth Annual Conference of the Cognitive Science Society*, 1983.

[13] Hsiang, J. and Dershowitz, N. "Rewrite Methods for Clausal and Non-Clausal Theorem Proving", *Proceedings of the ICALP-10*, 1983.

[14] Keller, R. M. "Learning by Re-Expressing Concepts for Efficient Recognition", *Proceedings of the National Conference on Artificial Intelligence*, 1983, pp. 182-186.

[15] Keller, R. M. personal communication, August 1986.

[16] Kelly, V. E. and Steinberg, L. I. "The Critter System: Analyzing Digital Circuits by Propagating Behaviors and Specifications", *Proceedings of the National Conference on Artificial Intelligence*, 1982, pp. 284-289.

[17] Koffman, E. B. "Learning Through Pattern Recognition Applied to a Class of Games", *IEEE Transactions on Systems Science and Cybernetics*, vol. ssc-4, no. 1, March 1968, pp. 12-16.

[18] Larkin, J., McDermott, J., Simon, D.P., and Simon, H. "Expert and Novice Performance in Solving Physics Problems", Science, Vol. 208, June 1980.

[19] Mahadevan, S. "Verification-Based Learning", *Proceedings of the Ninth International Conference on Artificial Intelligence*, 1985.

[20] McClelland, J. L. and Rumelhart, D. E. (eds) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 2, Bradford Books, 1986.

[21] Michalski, R. S. and Chilausky, R. L. "Learning by being told and learning from Examples: an experimental Comparison of the two Methods of Knowledge Acquisition in the Context of developing an Expert System for Soybean Disease Diagnosis", *Policy Analysis and Information Systems*, Vol. 4, No. 2, June 1980.

[22] Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (eds) *Machine Learning*, Tioga, 1983.

[23] Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (eds) *Machine Learning*, Vol. 2, Morgan-Kaufman, 1986.

[24] Minton, S. "Constraint-based Generalization: Learning Game-Playing Plans from Single Examples", *Proceedings of the National Conference on Artificial Intelligence*, 1984, pp. 251-254.

[25] Minton, S. personal communication, August 1986.

[26] Mitchell, T. M. "Version Spaces: A Candidate Elimination Approach to Rule Learning", *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 1977, pp. 305-310.

[27] Mitchell, T. M. "Towards Combining Empirical and Analytic Methods for Learning Heuristics", *Human and Artificial Intelligence*, Elithorn and Banerji (eds), Erlbaum, 1983.

[28] Mitchell, T. M., Keller, R. M. and Kedar-Cabelli, S. T. "Explanation-Based Generalization: A Unifying View", Tech. Report ML-TR-2, Department of Computer Science, Rutgers University, August 1985.

[29] Mostow, D. J. "Machine Transformation of Advice into a Heuristic Search Procedure", *Machine Learning*, Michalski, Carbonell, Mitchell (eds), Tioga, Palo Alto, 1983, pp. 367-403.

[30] Munkres, J. R. *Topolgy A First Course*, Prentice-Hall, 1975.

[31] Rosenblatt, F., *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*, Spartan Books, 1961.

[32] Rumelhart, D. E. and McClelland, J. L. (eds) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Vol. 1, Bradford Books, 1986.

[33] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. "Learning Internal Representations by Error Propagation", in Rumelhart, D. E. and McClelland, J. L. (eds) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1, Bradford, 1986.

[34] Stefik, M. J. "Planning with Constraints", Ph.D. dissertation, Stanford University, 1980.

[35] Touretzky, D. S. "BoltzCONS: Reconciling Connectionism with the Recursive Nature of Stacks and Trees", *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, Amherst, MA, 1986.

[36] Utgoff, P. E. and Mitchell, T. M. "Acquisition of Appropriate Bias for Inductive Concept Learning", *Proceedings of the National Conference on Artificial Intelligence*, 1982, pp. 414-417.

[37] Utgoff, P. E. *Machine Learning of Inductive Bias*, Kluwer, June 1986.

[38] Vere, S. A. "Multilevel Counterfactuals for Generalizations of Relational Concepts and Productions", *Artificial Intelligence*, Vol. 14, No. 2, 1980, pp. 138-164.

[39] Waldinger, R. "Achieving Several Goals Simultaneously", *Machine Intelligence*, Elcock and Michie (eds), Wiley & Sons, New York, 1976, pp. 94-136.

[40] Waterman, D. A., "Generalization Learning Techniques for Automating the Learning of Heuristics", *Artificial Intelligence*, Vol. 1, No. 1/2, 1970, pp. 121-170.

[41] Widrow, B. and Hoff, M. E., "Adaptive Switching Circuits", 1960 WESCON Convention Record Part IV, 1960.