Appears in *Connection Science,* 5, pp.339-364, 1993
(Special issue on Architectures for Integrating Neural and Symbolic Processing)

Combining Connectionist and Symbolic Learning to Refine Certainty-Factor Rule Bases

J. Jeffrey Mahoney and **Raymond J. Mooney**
Dept. of Computer Sciences
University of Texas
Austin, TX 78712
mahoney@cs.utexas.edu, (512) 471-9589
mooney@cs.utexas.edu, (512) 471-9558

June 1, 1993

Abstract

This paper describes RAPTURE — a system for revising probabilistic knowledge bases that combines connectionist and symbolic learning methods. RAPTURE uses a modified version of backpropagation to refine the certainty factors of a probabilistic rule base and it uses ID3's information-gain heuristic to add new rules. Results on refining three actual expert knowledge bases demonstrate that this combined approach generally performs better than previous methods.

1 Introduction

In complex domains, learning needs to be biased with prior knowledge in order to produce satisfactory results from limited training data. Recently, both connectionist and symbolic methods have been developed for biasing learning with prior knowledge (Shavlik and Towell, 1989; Fu, 1989; Ourston and Mooney, 1990; Pazzani and Kibler, 1992; Cohen, 1992). Most of these methods revise an imperfect knowledge base (usually obtained from a domain expert) to fit a set of empirical data. Some of these methods have been successfully applied to real-world tasks, such as recognizing promoter sequences in DNA (Towell et al., 1990; Ourston and Mooney, 1990; Thompson et al., 1991). The results demonstrate that revising an expert-given knowledge base produces more accurate results than learning from training data alone.

To date, research in this area has focused on revising knowledge bases initially specified in the form of logical, Horn-clause (if-then) rules. However, many real-world domains require some form of probabilistic reasoning. Consequently, many knowledge bases obtained from experts are represented using a probabilistic formalism, such as certainty factors, Dempster-Shafer theory, or Bayesian networks (Shafer and J. Pearl, 1990). There has been recent work on revising such probabilistic knowledge bases (Ginsberg et al., 1988; Fu, 1989). In particular, by representing a probabilistic rule base as a connectionist network, backpropagation methods (Rumelhart et al., 1986) can be used to modify the weights representing rule strengths (Fu, 1989; Lacher, 1992). However, these methods are restricted to revising numerical parameters and are unable to make structural changes to the knowledge base, such as adding new rules.

In this paper, we describe the RAPTURE system (**R**evising **A**pproximate **P**robabilistic **T**heories **U**sing **R**epositories of **E**xamples), which combines connectionist and symbolic methods to revise both the parameters and structure of a certainty-factor rule base (Shortliffe and Buchanan, 1975; Buchanan and E.H. Shortliffe, 1984). First, the initial rules are mapped into an equivalent network in which certainty factors become the weights on connections between nodes of the network. Unlike standard neural networks, in which the total input to a node is determined by a linear sum of all incoming activations, for RAPTURE networks, the total input is the *probabilistic sum* $(x + y - xy)$ of the incoming activations. No thresholding output function is needed since the probabilistic sum already provides the required non-linearity.

Next, the network is modified to correctly classify a set of training examples. Network training is performed in two phases. First, a modified version of backpropagation is used to adjust the certainty factors on the existing rules. The normal backpropagation equations are changed in order to perform gradient descent for certainty-factor combination functions such as probabilistic sum, MIN, and MAX. This is similar to methods employed in Fu (1989); Lacher (1992). If all of the examples can be classified correctly through backpropagation alone, then the network is considered trained. Otherwise, *symbolic* methods are used to alter the network architecture. Specifically, features are added that help discriminate examples according to ID3's information gain criterion (Quinlan, 1986b) and low-weighted links are deleted. Backpropagation and node addition/deletion continue in a cycle until all of the training examples are correctly classified.

Once it has been trained, the revised rules can be read directly off of the network. In the KBANN system (Towell and Shavlik, 1992)(see next section), revised networks are mapped back into rules to improve the comprehensibility of the final result. In RAPTURE, the direct correspondence between weighted links and probabilistic rules removes any distinction between the symbolic and connectionist representations. They are equivalent ways of looking at the same knowledge. The approach therefore combines the effectiveness of connectionist learning methods with the interpretability of rules. Comprehensibility is important since it has been found that users will generally not accept a system's conclusions unless it can present meaningful explanations for them (Swartout, 1981).

RAPTURE has been tested on revising several real-world knowledge bases with encouraging results. In particular, we present results for revising rule bases for promoter recognition in DNA sequences (Towell et al., 1990), soybean disease diagnosis (Michalski and Chilausky, 1980), and diagnosis of bacterial infections (Buchanan and E.H. Shortliffe, 1984). In the last domain, RAPTURE successfully revised a version of the MYCIN knowledge-base. We compare our results to those obtained for purely inductive methods (ID3, standard backpropagation, and RAPTURE given no initial knowledge), a purely connectionist method for knowledge-base refinement (KBANN), and a purely symbolic method for knowledge-base refinement (EITHER (Ourston and Mooney, 1990)). RAPTURE generally produces more accurate results from fewer training examples than these competing approaches. In the promoter domain, it also produces a simpler revised rule base than KBANN or EITHER.

The rest of this paper is organized as follows. Section 2 presents relevant background information, and Section 3 discusses the details of the RAPTURE algorithm. Section 4 presents our results on revising several real-world knowledge bases. Section 5 discusses related work, Section 6 discusses future work, and Section 7 presents our conclusions.

2 Background

This section presents background information that is required for understanding the rest of this paper. This includes overviews of two recent theory revision systems (KBANN and EITHER), as well as the certainty-factor formalism upon which RAPTURE is based.

2.1 EITHER

EITHER (Ourston and Mooney, 1990) is a recent theory revision system that uses propositional Horn-clause logic to represent its theories. It begins with an expert-given rule base, and a set of correctly labelled (training) examples.

EITHER revises its rule base whenever a training example is classified incorrectly. One of EITHER's primary objectives is to ensure that the rule base correctly classifies all training examples. A negative example that the rules are able to prove (as positive) indicates that the rule base needs to be specialized. Similarly, a positive example that the rules are *unable* to prove indicates the need to generalize the rule base. An overriding assumption is made that the original rules are not very far from being correct, hence an attempt is made to modify the rule base as little as possible, while achieving consistency with the training examples.

EITHER supports two means for specializing its rule bases—removing rules from the rule base, and adding conjuncts to existing rules. Both of these have the net effect of making rules less likely to fire in a given situation, which will strictly specialize the rule base. Specialization is done in a manner that guarantees that no negative examples are provable, while simultaneously insuring that no positive examples become unprovable.

Generalizing the rule base proceeds similarly whenever an unprovable positive example is encountered. EITHER contains analogous mechanisms for generalization. These are adding new rules to the rule base, deleting conjuncts from existing rules, and adding new disjuncts. Each of these mechanisms facilitates rule firing, allowing more examples to be classified positively, which generalizes the rule base.

It is hoped that via minimal modification to the original rule base, much of the expert-given domain knowledge will remain intact, enabling the rules to perform at satisfactory levels of accuracy on unseen examples. Ourston and Mooney (in press) illustrate the success of this heuristic with results in several domains.

2.2 KBANN

KBANN (Towell, 1991) is a revision system that combines a rule base with neural network learning. An expert-supplied rule base is converted into a neural network, which is then trained using connectionist techniques. After training, the network is translated into symbolic rules.

The conversion process into a neural network proceeds in a straightforward manner. Each unique literal from the original rule base is translated into a unique node of the network. All antecedents for a particular consequent literal become nodes that are linked as inputs to the consequent node, using weights that approximate the conjunctions and disjunctions of the original rule. Also, *all* features from the domain theory that are not mentioned in the rules are added into the network with low-weighted links. All neighboring layers are fully connected.

Once the network is built, it is trained using backpropagation, after which symbolic rules can be extracted from the network. This translation proceeds exactly as before, except that now nodes from the network revert back to literals of the rule base. Links with sufficiently low weights (less than some threshold) are deemed inconsequential, and are not incorporated into the symbolic rules. This has the beneficial effect of simplifying the resulting rule base, though there is now no guarantee that the two representations are equivalent.

2.3 Certainty Factors

2.3.1 The Certainty-Factor Formalism

RAPTURE is a rule-base revision system that combines elements from both EITHER and KBANN. RAPTURE uses propositional certainty-factor rules to represent knowledge. These rules have the following form: $A \stackrel{0.8}{\to} D$. This expresses the idea that belief in proposition A gives a 0.8 measure of belief in proposition D.

Certainty factors can range in value from -1 to $+1$, and indicate a degree of confidence in a particular proposition. These numbers are *not* meant to be an indication of one's belief in the rule itself, but rather indicate the degree to which the presence of one (or several) proposition(s) gives evidence for belief in another. For example, knowing that someone has the sniffles will give some evidence that the person has a cold. The more evidence that accumulates, the stronger our belief becomes.

A certainty factor of $+1$ represents absolute certainty (true), whereas one of -1 represents total disbelief (false). A certainty factor of 0 suggests no evidence is being offered. Knowing that someone has blonde hair has no bearing on our belief that s/he has a cold. Certainty-factor rules allow updating of these beliefs based upon observed evidence. Given the rule $A \stackrel{0.8}{\to} D$, we can update our belief in D based upon observation of A. If A is observed to be *true*, we then have a measure of belief in D of 0.8 ($= 1 \times 0.8$). Observing A *false* turns our belief in D to one of disbelief (belief < 0), with a measure of disbelief -0.8.

As mentioned previously, rules combine evidence via probabilistic sum, which is defined (for positive evidence) as $a \oplus b \equiv a + b - ab$. Adding a second rule to our rule base, $B \stackrel{.5}{\to} D$, demonstrates this. Given A and B (to be true), our measure of belief in D becomes 0.9 ($= 0.8 + 0.5 - 0.8 \times 0.5$).

Negative certainty factors combine using $a \oplus b \equiv a + b + ab$. Given A and B with certainty factors -0.5 and -0.2 respectively, D's certainty factor becomes -0.46 ($= -0.4 - 0.1 + 0.04$). In general, all positive evidence is combined to determine the *measure of belief* (MB) for a given proposition. Similarly, all negative evidence is combined to obtain a *measure of disbelief* (MD). The certainty factor is then calculated using $CF = MB + MD$.

Certainty-factor rules may also contain multiple antecedents, as in $A \wedge B \wedge C \stackrel{.7}{\to} D$. Conjunction is handled by using MIN. The minimum certainty factor from among A, B, and C combines with the 0.7 to determine D's certainty factor. Similarly, the MAX function is used with antecedent disjunction.

A common misconception about certainty factors is that they represent probabilities. They do not. A rule of the form $\texttt{sniffles} \stackrel{0.45}{\to} \texttt{cold}$, states that the observance of sniffles gives one reason to believe that the person has a cold. This does *not* mean that there is a 45% probability of a cold, but only that there is some evidence (0.45 on scale from -1 to

1) for believing that the person has one. The probabilistic interpretation would imply that observing sniffles indicates a 55% probability of not having a cold, which is not intended. The rule simply states that sniffles suggests increased evidence in our belief that a person has a cold, but does not increase our belief that the person does not have a cold.

2.3.2 A Case For Certainty Factors

RAPTURE grew out of a desire to build a rule-base revision system that was capable of working in a probabilistic framework. This was due to the existence of a number of domains where it appeared that such reasoning was necessary in order to successfully represent goal concepts. What was needed in particular, was a formalism that was capable of handling the evidence-summing feature of uncertain reasoning, which enables small pieces of evidence to accumulate into significant evidence in favor of a conclusion.

Certainty factors were chosen for a variety of reasons. First, it is perhaps the simplest method that retains the desired evidence-summing aspect of uncertain reasoning. As each rule fires, additional evidence is contributed towards belief in the rule's consequent. All evidence can then be combined giving an overall degree of confidence in the consequent. The use of probabilistic sum enables many small pieces of evidence to add up to significant evidence. This is lacking in formalisms that use only MIN or MAX for combining evidence (Ling and Valtorta, 1991). Second, probabilistic sum is a simple, differentiable, non-linear function. This is crucial for implementing gradient descent using backpropagation. Further, other formalisms of uncertain reasoning (e.g. Bayesian networks) have been shown to be NP-Hard to evaluate in the general case (Cooper, 1987). Even more significantly, however, is the widespread use of certainty factors. Despite recent criticism of certainty factors (Shafer and J. Pearl, 1990), there have been numerous knowledge-bases implemented using the certainty-factor model, which immediately gives our approach a large base of applicability. Finally, and perhaps most importantly, are the empirical results. Results to date seem to indicate that this approach is one that is worth pursuing.

3 The Rapture Algorithm

The RAPTURE algorithm breaks down into three main phases. First, an initial rule base (created by a human expert) is converted into a RAPTURE network. The result is then trained using certainty-factor backpropagation (CFBP). The theory is further revised through network architecture modification. Once the network is fully trained, the solution is at hand—there is no need for retranslation. Each of these steps is outlined in full below.

3.1 The Initial Rule Base

It has been demonstrated (Shavlik and Towell, 1989; Ourston and Mooney, 1990) that dramatic increases in learning performance can be achieved by initializing a learning algorithm with human expertise. This initial theory can be represented in many forms, and as previously stated, RAPTURE has adopted the certainty-factor formalism for its rule representation. An expert is needed to provide a set of rules expressed using certainty factors that approximate his expertise in defining a given concept. Since only an approximation of the expertise is needed, much of the knowledge acquisition bottleneck can be alleviated, as the fine-tuning is left to the automated system.

3.2 Converting the Theory into a Network

Once the initial rule base is obtained, it is converted into a RAPTURE -network. This process is straightforward. Building the network begins by mapping all identical propositions in the rule base to the same node in the network. Input features (those only appearing as rule-antecedents) become input nodes, and are placed at the bottom of the network. Output symbols (those only appearing as rule-consequents) become output nodes, and are placed at the top of the network. The certainty factors of the rules become the weights on the links that connect nodes. Networks for classification problems contain one output for each category. When an example is presented, the certainty factor for each of the categories is computed and the example is assigned to the category with the highest value.

Consider a simple example of three rules:

$$A \xrightarrow{.7} D \qquad B \xrightarrow{.2} D \qquad C \xrightarrow{.5} D$$

The network of Figure 1(a) is the RAPTURE -network for this rule set.

These rules state that A,B, and C all contribute evidence towards D in varying degrees. Note the difference between these rules, and the single rule $A \wedge B \wedge C \wedge \xrightarrow{x} D$, which states that all of A,B, and C must be true (to some degree) in order for there to be any evidence for D.

Figure 1(b) illustrates the following more complete set of rules.

$$\begin{array}{ccc} ABC \xrightarrow{.5} D & E \xrightarrow{.7} D & C \xrightarrow{.1} G \\ EF \xrightarrow{.8} G & HI \xrightarrow{.3} C & I \xrightarrow{.2} E \end{array}$$

As shown in the network, conjuncts must first pass through a MIN node before any activation reaches the consequent node. Note that each of the conjuncts is connected to the corresponding MIN mode with a solid line. This represents the fact that the link is non-adjustable, and simply passes its full activation value onto the MIN node. The standard (certainty-factor) links are drawn as dotted lines indicating that their values *are* adjustable.

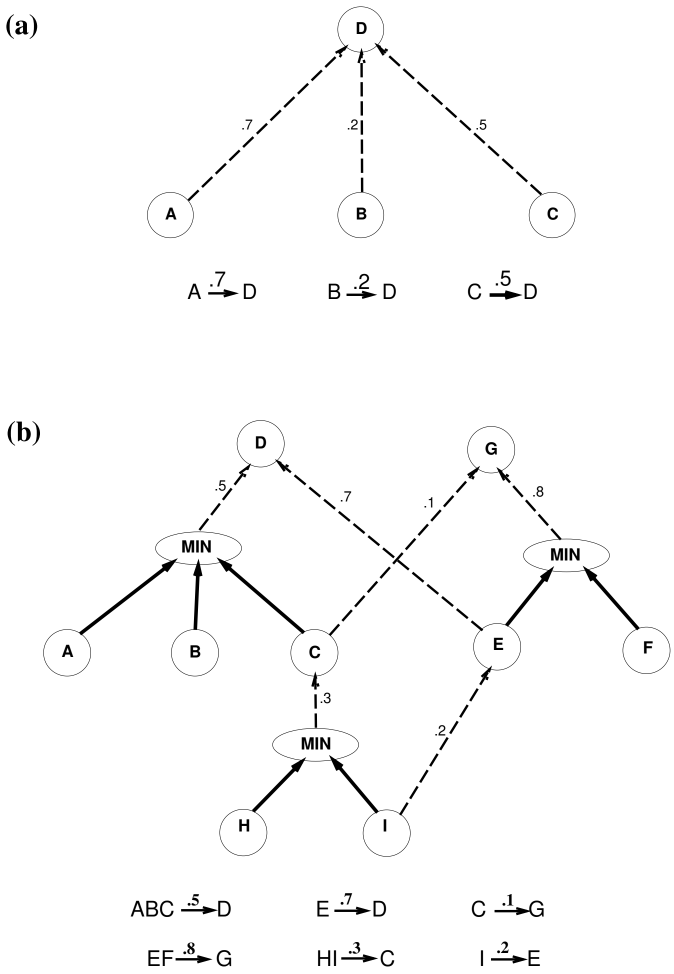

Figure 1: Converting Rules into Rapture-Networks

This construction shows how easily a RAPTURE network can model a certainty-factor rule base. Each representation can be converted into the other, without loss or corruption of information. They are two equivalent representations of the same set of rules.

3.3 Certainty-Factor Backpropagation

Using the constructed RAPTURE network, we desire to maximize its predictive accuracy over a set of training examples. This is achieved by minimizing the overall error at each of the output nodes with respect to all of the examples. Cycling through the examples one at a time, and slightly adjusting all relevant network weights in a direction that will minimize the output error, we can hill-climb until the overall output error reaches a local minimum. This is the idea behind gradient descent, which is most commonly implemented using backpropagation.

In order to train a neural network using backpropagation, the standard formula for adjusting the weight linking node i to node j (w_{ji}) after seeing pattern p is

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi} \tag{1}$$

where η is the user-defined learning rate, o_{pi} is the output of unit i for input pattern p, and δ_{pj} is the output error of unit j for pattern p. The output of a node j is $f(net_{pj})$ where the net input $net_{pj} = \sum w_{ji} o_{pi}$ for all input connections i, and where f is a nondecreasing, differentiable function (generally a logistic function). The value of δ_{pj} is determined by the type of unit. If j is an output unit, then

$$\delta_{pj} = (t_{pj} - o_{pj}) f'_j(net_{pj}). \tag{2}$$

where t_{pj} is the correct output value for unit j. If j is not an output unit, then

$$\delta_{pj} = f'_j(net_{pj}) \sum_k \delta_{pk} w_{kj}. \tag{3}$$

RAPTURE does *not*, however, use these standard formulas, since it is trying to model certainty-factor summation of evidence, as opposed to a standard neural network. In order to achieve gradient descent in a RAPTURE network, it is necessary to first derive the corresponding formulas for Certainty-Factor Backpropagation (CFBP).

The main distinction is the manner in which inputs to a node combine to give its total net input (net_{pj}). A RAPTURE network uses probabilistic sum rather than the standard linear summation of neural networks in determining net input. This is as described in Section 2. As this value is passed directly on as the node's output, we are defining $o_{pj} = net_{pj}$, hence the output function is identity. In order to perform backpropagation on a network such as this, the following equations must be utilized. For the derivation of these equations, see the Appendix.

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi} (1 \pm \sum_{k \neq i} w_{jk} o_{pk}) \tag{4}$$

If u_j is an output unit

$$\delta_{pj} = (t_{pj} - o_{pj}) \tag{5}$$

If u_j is not an output unit

$$\delta_{pj} = \sum_{k_{min}} \delta_{pk} w_{kj} (1 \pm \sum_{i \neq j} w_{ki} o_{pi}) \tag{6}$$

The "Sigma with circle" notation is meant to represent probabilistic sum over the index value, and the $\pm$ notation is shorthand for two separate cases. If $w_{ji}o_{pi} \geq 0$, then $-$ is used, otherwise $+$ is used. The k_{min} subscript refers to the fact that we do not perform this summation for *every* unit k (as in standard backpropagation), but only those units that received some contribution from unit j. Since a unit j may be required to pass through a min or max-node before reaching the next layer (k), it is possible that its value may not reach k.

Using these equations, CFBP performs gradient descent on RAPTURE networks in a manner analogous to the way standard backpropagation works on neural networks. Examples are fed into the network one at a time through its input lines, and weight-adjustment occurs with each example as network error *propagates* down towards input lines. Each complete pass through all of the examples defines one *epoch*, and numerous epochs are run while training the network. This has the effect of adjusting all of the certainty factors in a direction that will locally minimize the mean-squared error over all training examples.

Assuming the target output value is 1 for the correct category and 0 for all other categories, it is virtually impossible to achieve an overall mean-squared error of zero. When combining evidence using probabilistic sum, an output of 1 is achieved only when there exists a rule(s) with a certainty factor of 1 whose antecedents are all true. Such cases are rare in probabilistic classification problems. Because of this, RAPTURE deems a classification correct when the output value for the correct category is greater than that of any other category. No error propagation takes place in this case ($\delta_{pj} = 0$).

One remaining issue is when to stop adjusting weights (i.e., how many epochs of CFBP are to be executed?). The simplest stopping criteria would be when all examples are classified correctly, but this criteria is not used for two reasons. First, backpropagation is not guaranteed to converge to 100% training accuracy—it may reach a local minimum. Further, classifying all of the examples correctly does not mean that further weight adjustment is of no use. Perhaps further adjustment would minimize error to an even greater extent. Correct classification does not imply 0 error, as explained in the preceding paragraph. A more useful criteria for terminating weight adjustment is when overall error seems to have reached a local minimum. RAPTURE checks the total mean-squared error after every 10 epochs, and halts CFBP whenever training accuracy is observed to decrease, or mean-squared error decreases by less than ϵ (.001), without improving training accuracy.

3.4 Changing the Network Architecture

Whenever training accuracy fails to reach 100% through CFBP, it is very likely a sign that the network architecture is inappropriate for the current classification task. To date, RAPTURE has been given two ways of changing network architecture. First, whenever the weight of a link in the network approaches zero ($|\text{weight}| < 0.001$), it is removed from the network along with all of the nodes and links that become detached due to this removal. This is reasonable since any link with zero weight is not contributing towards the classification of any of the examples. Further, whenever an intermediate node loses all of its input links due to link deletion, it too is removed from the network, along with its output link. This link/node deletion is performed immediately following CFBP.

RAPTURE also has a method for adding new nodes into the network. Specific nodes are added in an attempt to maximize the number of training examples that are classified correctly. The problems are in deciding which node to add, where to insert it, and how to link it into the network. The simple solution employed by RAPTURE is to create new input nodes that connect directly, either positively or negatively, to one or more output nodes. These new nodes are created in a way that will best help the network distinguish among training examples that are being misclassified.

RAPTURE achieves this in the following manner. Let us define C as the set of all possible categories into which an example may be classified, and C_i represent the ith category. We define FN_i to be the set of false-negative examples for category C_i. These are those examples whose target (correct) category is C_i, yet the network has classified as $C_{j \neq i}$. In fact, the network may have given a higher certainty factor to several C_j for any given example in FN_i. We can define MC_i as the set of all mistaken categories $C_{j \neq i}$, such that for some example in FN_i, C_j had a higher certainty factor than C_i. Finally, define CE_i as the set of all examples whose target category is among the categories in MC_i.

This leaves us with two disjoint groups of examples. FN_i—the false negatives for C_i, and CE_i—the true positives for all categories that are being confused with C_i. We need to find a feature-value which can discriminate between these groups of examples. Quinlan's ID3 metric (Quinlan, 1986b) has been adopted by RAPTURE as the solution to this problem.

ID3 is designed to build decision trees that classify examples into pre-defined categories. Given a set of examples described as feature vectors, ID3 selects the feature that best partitions these examples into subsets that correlate with an expert's classification. This is done using an information-gain metric. Each node in the tree is the feature with the highest information gain among examples at this node, and each branch beneath this node represents a particular value.

This idea can be easily modified to work with RAPTURE. RAPTURE has two sets of examples to work with (FN_i and CE_i), and needs to find a feature-value pair that is highly prominent in the former, yet lacking in the latter. Once this feature-value has been determined, it can be used as positive evidence for category C_i and negative for all categories in MC_i. ID3 information gain can find this feature-value for us if we consider every possible feature-value pair in the domain as a binary feature. Instead of having a feature `COLOR` with values `RED`, `BLUE`, and `GREEN`, we have 3 features `COLOR=RED, COLOR=BLUE, and COLOR=GREEN`, all with values `NO` or `YES`. By labelling each example from FN_i as *negative*, and those from CE_i as *positive*, RAPTURE can then use information gain to determine which *binary* feature maximally distinguishes these sets of examples. ID3's metric is designed to choose that feature that produces the greatest reduction in the estimated entropy of information of the examples.

Specifically, this metric chooses the feature that minimizes

$$[(P_{yes} + N_{yes}) * INFO(yes) + (P_{no} + N_{no}) * INFO(no)]/N$$

where P_{yes} refers to the number of positive examples with value yes for the feature in question. N_{yes} is the number of negative such examples. $INFO()$ is a function of the value (yes or no) defined as $INFO(\texttt{value}) =$

- `0 if` $P_{value} = 0$ `or` $N_{value} = 0$
- $-ulog_2u - vlog_2v$ `otherwise`,
  `where` $u = P_{value}/(P_{value} + N_{value})$, `and` $v = N_{value}/(P_{value} + N_{value})$.

RAPTURE breaks any ties at random, and checks to make sure that the selected feature is not already being used among the rules for C_i. Once this feature-value has been selected, a new input node is added as positive evidence for C_i. It may be the case that the feature-value selected is *negative* evidence for C_i, meaning that it is highly prominent in CE_i yet lacking in FN_i. In this case, we invert the value of the feature (from `COLOR=RED` to `COLOR≠RED`). Placing a small positive weight (0.1) on the rule gives a rule of the form `COLOR=RED`$\xrightarrow{.1} C_i$, or `COLOR≠RED`$\xrightarrow{.1} C_i$ which serves as new positive evidence for C_i. Similar rules (using the same feature-value pair) are built for each of the categories in set MC_i with negative weights (-0.1). This gives new negative evidence for all of these categories. The examples in FN_i will now be more likely to be classified in category C_i, and less likely to be classified in a category in MC_i.

This is performed for each $C_i \in C$. Note that for some C_i, the false negative list will be empty. This happens whenever every example of this category is classified correctly. This produces no new nodes in the network. False positives are not dealt with explicitly. These are examples that are *not* supposed to be classified as C_i, but are nonetheless. These will turn up as false negatives for some other C_i.

With these new nodes in place, we can now return to CFBP, where hopefully more training examples will be successfully classified. This entire process (CFBP followed by deleting links and adding new nodes) repeats until all training examples are correctly classified. Once this has occurred, the network is considered trained, and testing may begin. An overview of the entire algorithm is depicted in Figure 2.

3.5 Special-Case Handling

There are certain types of examples that are handled specially by RAPTURE. The first of these is examples with noise. If two or more examples are found to have identical feature-value vectors, yet different classifications, only one of them is kept in the set of examples (one with the category shared by the majority). Clearly, no rule base has the ability to distinguish

Loop until 100% training accuracy is achieved.

1. Perform CFBP on the network. Use given training examples, and as many epochs as necessary until mean-squared error decreases by $< \epsilon$. Delete any links whose weights change sign.

2. If not 100% training accuracy, use ID3 information gain to add new input units. Make one new rule for each output unit misclassifying positive examples.

Figure 2: Overview of the RAPTURE Algorithm

---

between two identical examples. Similarly, examples with missing features are handled in a special manner. In order for an example to be classified by the network, a value must be assigned to each feature that is referenced by RAPTURE. An example with no value for a given feature is assigned the value $1/n$ as its certainty factor for *each* of the n possible values for this feature. This was shown to be an effective encoding in Shavlik et al. (1991) A similar idea is utilized when an example is passed into the information-gain metric. The function requires counts of the number of examples with and without a particular feature-value, and a missing-valued example counts as $1/n$th of an example with the value in question (and $(n-1)/n$th of an example without). This is the method used in Quinlan (1986a).

Finally, features that have continuous values (linear features), are handled in a fuzzy-logic manner. A rule of the form `‘‘if [AGE-OF-PATIENT` $<$ `50] then ...’’` is evaluated by using the value $1/(1+e^{AGE-50})$ as its certainty factor. This gives a certainty factor of 0.5 to someone aged exactly 50, and drops to 0 rapidly as age increases. Similarly, the certainty factor goes to 1 as age decreases. This allows for some flexibility in the evaluation of rules with strict threshold values.

4 Experimental Results

To date, RAPTURE has been tested on three real-world domains. The first of these is a domain for recognizing promoter sequences in strings of DNA-nucleotides. The second uses a theory for diagnosing soybean diseases, and the third is a version of the MYCIN knowledge-base, designed to provide consultative advice on diagnosis and therapy for infectious diseases. These data sets are discussed in detail in the following sections.

4.1 DNA Promoter Results

A prokaryotic *promoter* is a short DNA sequence that precedes the beginnings of genes, and are locations where the protein RNA polymerase binds to the DNA structure (Towell et al., 1990). A theory designed to recognize such strings composed of DNA-nucleotides was given to RAPTURE for revision. The data set used for these experiments is one of 106 examples, for which there are 53 positive examples (i.e. promoters), and 53 negative examples. An example consists of a sequence of 57 DNA nucleotides, each of which can take on one of four values—A, C, G, or T. The original theory for this recognition task, based on information provided by O'Neill and Chiafari (1989), was written as a set of propositional Horn-clauses, not as a set of certainty-factor rules. For this reason, the theory had to be modified in order to be utilized by RAPTURE. This theory was modified by breaking up rules with multiple antecedents, into several rules. This was done in order to allow each antecedent the ability to contribute its own evidence towards belief in the consequent. This overcame one of the difficulties with standard Horn-clause representations. If any one of several antecedents to a rule is false, then the rule cannot fire. This representation allows it to fire a little less strongly.

Initial certainty factors were assigned in such a way that if *every* antecedent (from the original rule) were true, a certainty factor of 0.9 would result for the consequent. As an example, one of the (original) rules looks like :

$(p12 = t) \wedge (p11 = a) \wedge (p7 = t) \rightarrow (minus10)$.

In RAPTURE format, this becomes:

$(p12 = t) \stackrel{.536}{\rightarrow} (minus10)$, $(p11 = a) \stackrel{.536}{\rightarrow} (minus10)$, $(p7 = t) \stackrel{.536}{\rightarrow} (minus10)$.

Given an example (DNA strand) that has $(p12 = t, p11 = a, p7 = t)$, RAPTURE will calculate a certainty factor of 0.9 for proposition $minus10$. If, however, only one or two of these nucleotides match the rule, there will still be some evidence contributed towards $minus10$.

Remembering that this problem is a *single-category* problem, where an example either is or is not a promoter, RAPTURE's training and testing schemes have to be slightly adjusted. Instead of assigning an example to the category with the highest certainty factor, a threshold value is used. For training, a certainty factor of 0.9 or greater classifies an example into the promoter category. Less than 0.9 classifies it as a non-promoter. Once the rule base is trained to 100% accuracy (using the 0.9 threshold), the threshold is adjusted for testing. By averaging the certainty factor of the lowest scoring positive example with the highest scoring negative example, a threshold maximizing the distance between the two sets of examples is obtained. The effect of this was minimal, as the new value never differed from the original 0.9 by more than 0.05.

In order to test RAPTURE using this data set, standard training and test runs were performed, which resulted in the learning curve of Figure 3.

This graph is a plot of average performance in accuracy at classifying DNA strings over

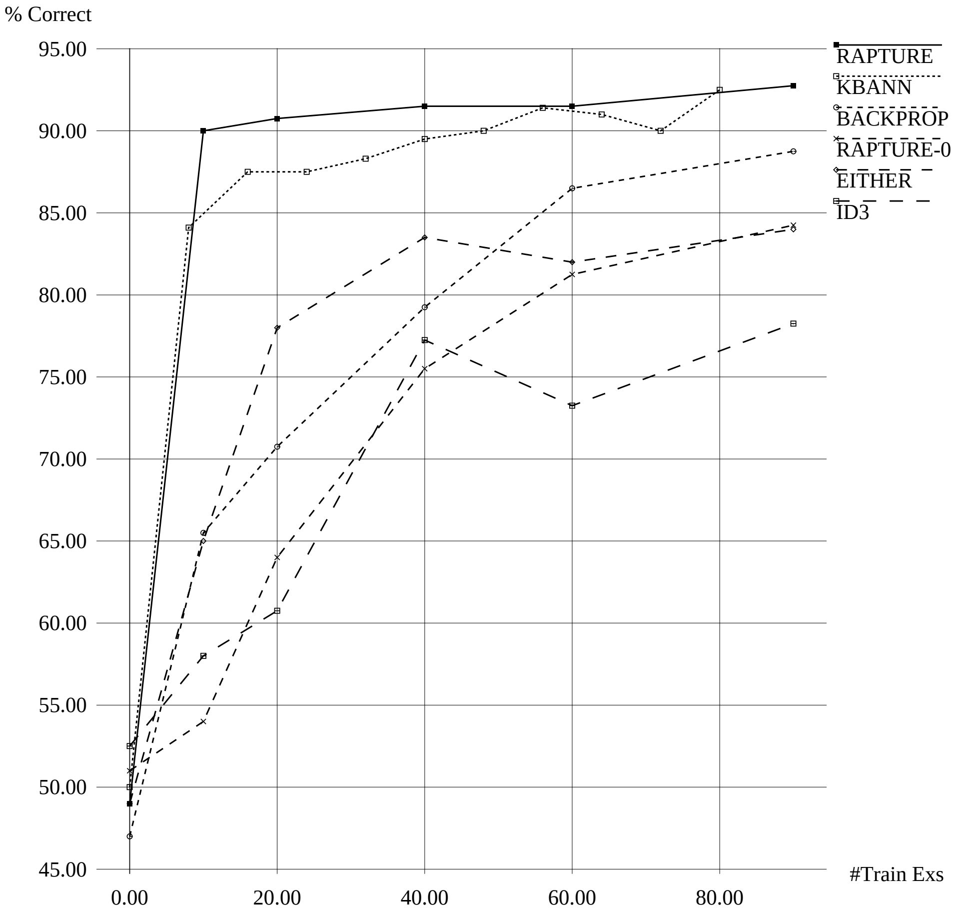

Figure 3: PROMOTER Testing Accuracy

25 independent trials. A single trial consists of providing each system with increasing numbers of examples to use for training, and then seeing how well it can classify unseen test examples. Each system is given the same training examples, and tested on the same unseen examples. This graph clearly demonstrates the advantages of an evidence summing system like Rapture over a pure Horn-clause system such as Either, a pure inductive system such as ID3, or a pure connectionist system, like backprop. Though they all start near 50% accuracy, Rapture is very rapidly over the 90% mark. In fact, running statistical *t-tests* over this data verifies the fact that Rapture outperforms all other systems (except Kbann), at the 0.05 level of significance or better. It was not possible to run t-tests on Kbann due to the fact that it was run on differing splits of the data, which were not available. It does appear, however, that Rapture's performance is superior. Also plotted in the graph, is Rapture-0, which is simply Rapture given no initial theory. This system begins with an

empty network, and adds links one at a time in order to correctly classify the training data. This system is unable to build hidden links, and this, coupled with lack of initial theory, clearly hinders learning.

This particular domain is thought to require M-of-N rules, which fire as long as some number (M) of its N antecedents are true. This is because of the fact that there are several potential sites at which hydrogen bonds can form between DNA and a protein. When enough of these bonds form, promoter activity can occur. This could explain why the connectionist systems KBANN and backprop perform considerably better than EITHER and ID3. This is because connectionist systems receive activation levels from a linear sum of all of their inputs, which include all possible features of the domain, thus producing a combining of evidence effect.

RAPTURE found this data very easy to classify. Figure 4 plots the time to train for the various systems. Only ID3 runs faster, as RAPTURE averaged about 20 seconds to train with 90 examples.

For this data set, CFBP alone was all that was required in order to train the network. The node addition module was never called. This is evidenced in Figure 5 which plots the complexity of the learned rules. This is a count of the number of symbols in the final theory after training with the largest training set (90 examples). RAPTURE's complexity remains constant throughout training, since CFBP did all of the necessary revision. This seems to be an indication that the original theory is relatively accurate, at least in regard to rule structure. This plot shows that only RAPTURE-0 ends up with a simpler theory, though clearly at the expense of predictive accuracy. The EITHER plot is somewhat unfair, as these numbers are based upon an early version of the system, whereby redundant rules were never deleted from the rule base. More recent results would show fewer symbols in the revised theory.

4.2 SOYBEAN Diagnosis Results

The soybean data comes from Michalski and Chilausky (1980), and is a data set of 562 examples of diseased soybean plants. Examples are described by a string of 35 features including the condition of the stem, the roots, the seeds, as well as information such as the time of year, temperature, and features of the soil. An expert classified each example into one of 15 soybean diseases. This data set has been used as a benchmark for a number of learning systems.

The theory used by RAPTURE was a certainty-factor version of the expert theory given in Michalski and Chilausky (1980). Again, the original theory was given as a set of Horn clauses, though it was described in a manner that suggested the use of certainty factors. The expert provided *significant* Horn clause rules for each disease, as well as *confirmatory* rules. The significant rules were meant to imply strong evidence for a particular disease, whereas the confirmatory only added confirming evidence. The values of 90% and 10% as

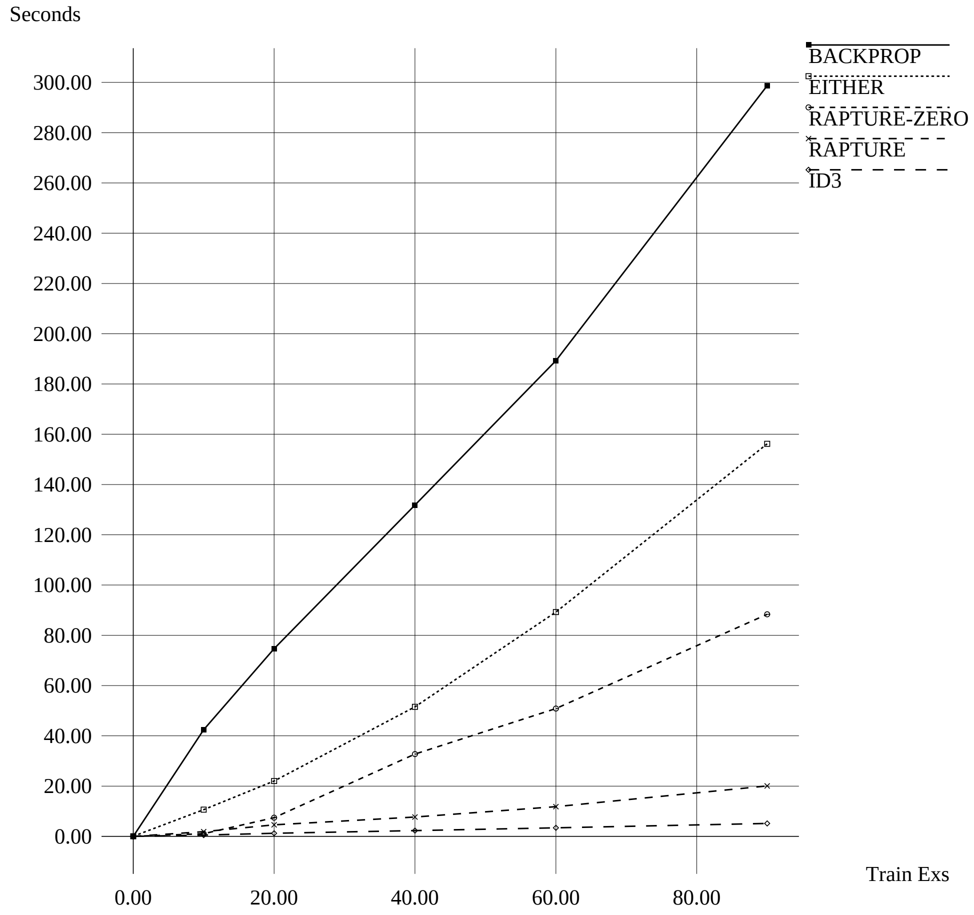

Figure 4: PROMOTER Training Time

the amount of evidence contributed respectively by the significant and confirmatory rules were suggested in Michalski and Chilausky (1980).

Certainty-factor rules for RAPTURE were produced in a manner similar to their production in the DNA domain. Initial certainty factors were assigned to reflect the significant and confirmatory status of the various pieces of evidence. Antecedents from significant rules were given certainty factors high enough to produce a 0.9 degree of belief if each of the conjuncts from the original rule were true. For the confirmatory pieces of evidence, this value was 0.1.

Figure 6 is a learning curve on this data comparing RAPTURE, RAPTURE-0, backpropagation, ID3, and EITHER. As is clear from the graph, RAPTURE is given a considerable head start with its expert-provided initial theory. The head start given to RAPTURE does not last throughout testing in this domain. RAPTURE maintains a statistically significant lead over the other systems (except RAPTURE-0) through 80 examples, but by 150 examples, all

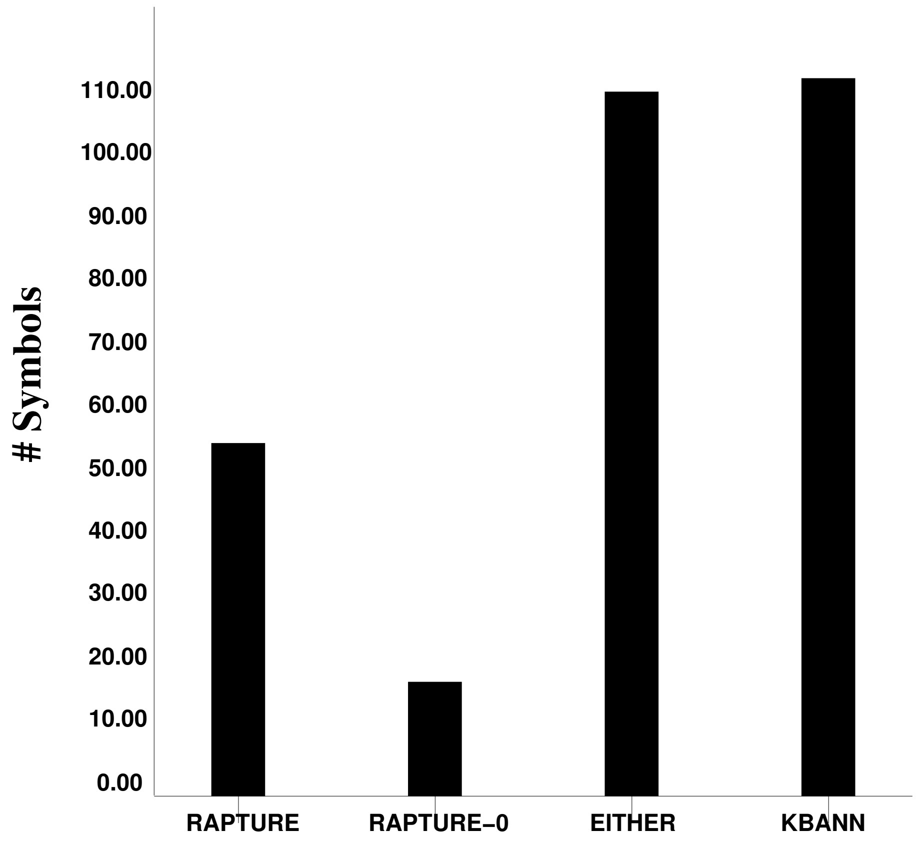

Figure 5: PROMOTER Concept Complexity

systems are performing at equivalent levels. There is perhaps a simple explanation for this phenomena. The initial theory tends to be quite helpful for a number of the easier-to-classify diseases. But there are a few categories of diseases that are difficult to distinguish, and the initial theory is not very helpful here. Because of this, the easier categories are also easy to learn through straight-forward induction. ID3 and backprop both catch up with Rapture after 150 training examples. Rapture-0 catches up a little more quickly (and actually does slightly better). There are no significant differences between any of the systems after 150 training examples. Trials have actually been run out to 300 examples, though all systems are performing at equivalent levels of accuracy.

Either to date has only been run up to 100 examples. Either has also been run using a partial-matching technique, where examples are classified into the category that was closest to firing, in which case its' performance nearly matches that of Rapture (Mooney and Ourston, 1991).

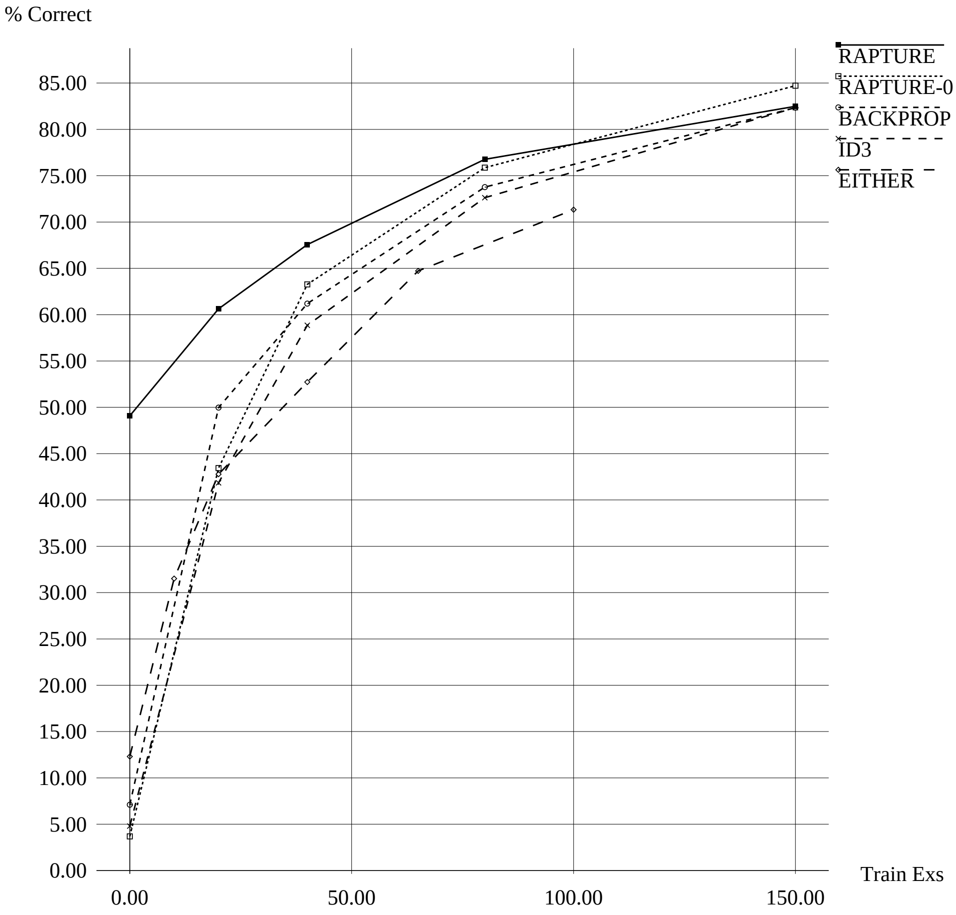

Figure 6: SOYBEAN Testing Accuracy

Training for this data set clearly took much more effort than the others, as seen in Figure 7. This is a plot of the time to train the various systems. It is clear, however, that RAPTURE does indeed perform more efficiently than either backpropagation or EITHER.

4.3 MYCIN Results

Experiments were also run on a version of the MYCIN knowledge-base (Buchanan and E.H. Shortliffe, 1984), which was designed to provide consultative advice on diagnosis and therapy for infectious diseases. This domain consists of 115 examples of solved cases (patients) of infectious diseases drawn from the Stanford Medical Center. Ten diseases are included with this data set. Each example is described with a vector of 264 features ranging from the patient's sex to the duration of any headaches. Many features for each patient are

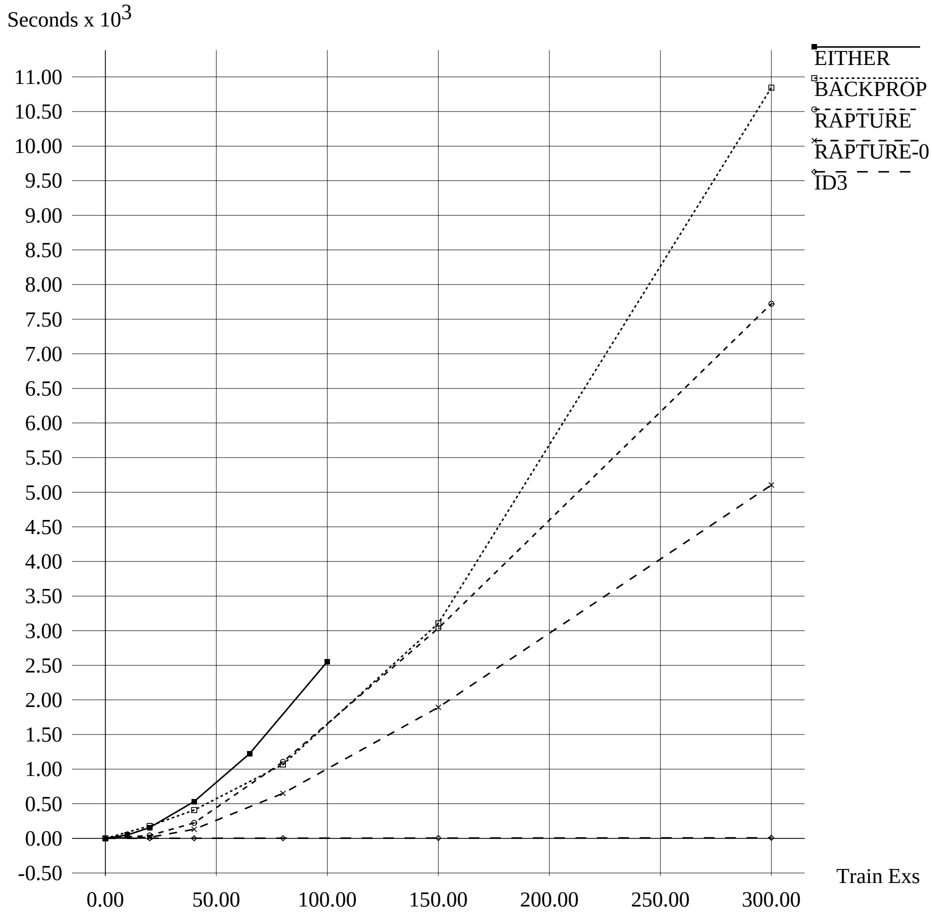

Figure 7: SOYBEAN Training Time

missing, and there are a great number of continuously-valued features.

The rules for the initial theory are provided as part of the Mycin database, and are already in a format acceptable to Rapture (certainty factors). Included are 137 certainty-factor rules for diagnosing the diseases, including a number of intermediate concepts. One of the diseases (primary brain-tumor) is given no initial rules. The learning curves for this data set are shown in Figure 8.

Despite Rapture's significant head start with this data set, backpropagation is doing somewhat better after 40 examples. It should be noted, however, that most of the differences between backpropagation and Rapture are *not* significant due to the variability of the testing. In particular, backpropagation is only significantly better at 40 and 80 test examples. Elsewhere, Rapture is performing at an equivalent or better level.

It is not yet clear why backpropagation does so well on this particular data set; there are

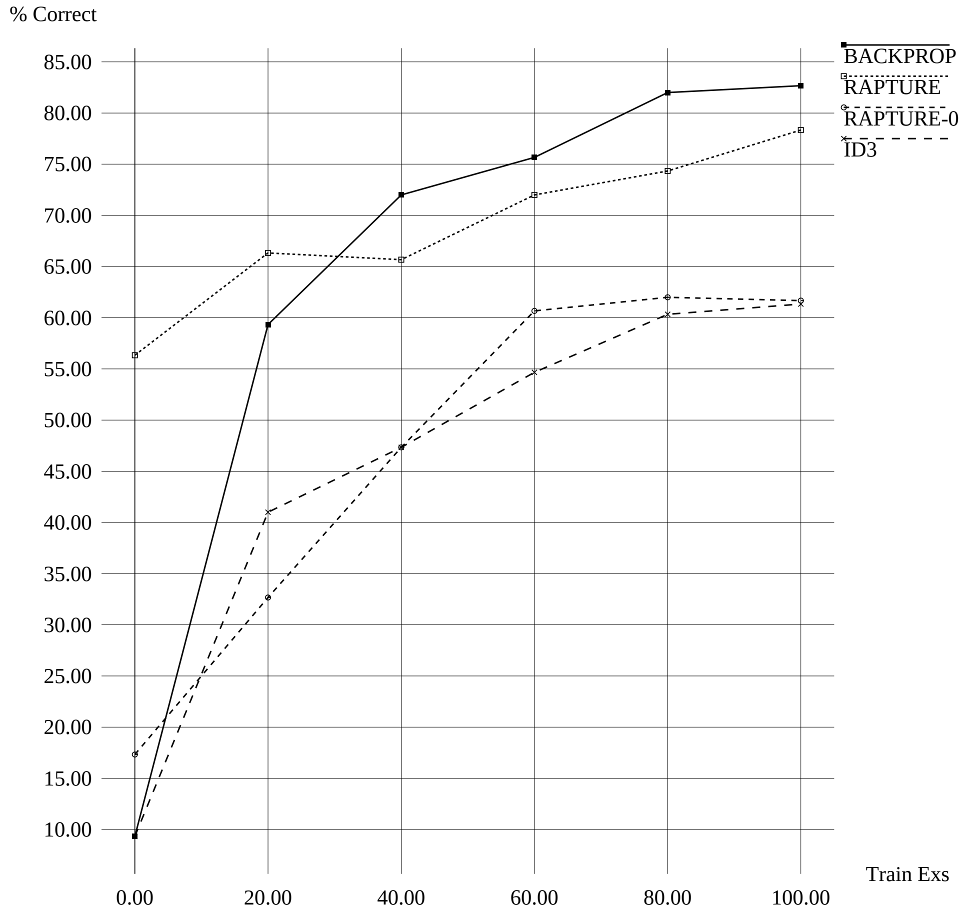

Figure 8: MYCIN Testing Accuracy

a couple of possible explanations. First, backprop is given a hidden layer of units to work with during training, and this is perhaps significant. RAPTURE on the other hand is working with a relatively flat theory (containing few hidden units), and has no means for creating hidden units. Second, backpropagation has available to it every feature from the domain from which to draw evidence. This may be crucial in this domain. Both of these hypotheses will be tested for future work. It should also be noted that in order for backpropagation to produce these results, a good deal of manual learning-rate adjustment was necessary before there was convergence. Also, backprop's training time was an order of magnitude longer than RAPTURE's. Finally, backprop is not capable of producing symbolic rules after network training. This is a significant advantage for RAPTURE, since the expert will be able to examine the refined knowledge base. Backpropagation simply returns a black-box classifier.

The two other systems, Rapture-0 and ID3, appear to be performing at similar levels, significantly less than backprop and Rapture. Rapture is outperforming these systems at the 0.0005 level of significance. This gives further support to the idea that this domain is one in which having every feature available for use is desirable. It is also noteworthy that backpropagation is significantly outperforming ID3. In most domains studies, these two systems generally perform at equivalent levels (Shavlik et al., 1991), and future work will examine why backpropagation is performing so well.

5 Related Work

This section reviews related work in both the connectionist and symbolic areas. Compared to most previous work, Rapture deals with a wider range of theory refinement problems and has been more thoroughly tested on actual expert knowledge bases.

The Seek2 system (Ginsberg et al., 1988) revises rule bases containing *M-of-N* rules, also known as *choice-component* rules. Seek2 uses specific heuristics to revise the threshold, M, of individual rules in order to improve performance on the training data. Unlike Rapture, Seek2 can not modify real-valued weights and contains no means for adding new rules.

Valtorta and Ling (Valtorta, 1988; Valtorta, 1990; Ling and Valtorta, 1991) have examined the computational complexity of various refinement tasks for probabilistic knowledge bases. They have considered networks using various combination functions, such as MIN, MAX, and probabilistic sum. For most combination functions and network architectures, they show that refining the weights to fit a set of training data is an NP-Hard problem. However, they present a simple linear-time algorithm for determining a correct setting of weights (or proving none exist) in the special case of a one-layer network with MAX as the combining function. However, even in this case, they provide no mechanisms for altering the network architecture. Rapture uses heuristic methods such as backpropagation and information gain to refine multi-layer networks that use MIN, MAX, *and* probabilistic sum. Although the method does not guarantee convergence, it performs quite well in practice on fairly large, realistic problems.

Ma and Wilkins (Ma and Wilkins, 1991) have developed methods for improving the accuracy of a certainty-factor knowledge base by deleting rules. They report only modest improvements in the accuracy of the same Mycin rule base (one used in our experiments). Their experiments increase performance from 26.8% to 36.0%. Rapture has the advantage of being able to adjust certainty factors and add rules in addition to deleting rules.

Gallant (Gallant, 1988) was apparently the first to design and implement a system that combines expert domain knowledge with connectionist learning. Given a set of training examples and expert supplied dependency information, his system builds a connectionist network that correctly classifies the training data. However, the training method is a variation of perceptron learning and is not suitable for multi-layer networks or for alternative combination functions like probabilistic sum.

KBANN (Towell et al., 1990; Towell and Shavlik, 1992) uses standard backpropagation to refine a symbolic rule base. A propositional Horn-clause rule base is mapped into a standard neural network, the network is refined using normal backpropagation, and the result is mapped back into rules with real-valued antecedent weights. Unlike RAPTURE, the mapping between the symbolic rules and the network is only an approximation. Also, it is unclear how certainty-factor rules might be mapped into a KBANN network. KBANN allows the learning of new rules by including an underlying fully-connected network of low-weighted links. These links can be "recruited" by backpropagation and eventually mapped back into new rules. Weight decay (Hinton, 1986) is used to keep weights small and therefore help minimize the number of new rules that are eventually introduced. By contrast, RAPTURE uses symbolic methods to add a minimal number of new connections (rules) as needed. The results on promoter recognition in Section 4.1 indicate that the RAPTURE approach produces a simpler and slightly more accurate revised knowledge base than KBANN.

There has been a number of methods for growing a network architecture sufficient to classify a set of training examples, e.g. *cascade correlation* (Fahlman and Lebiere, 1989), the upstart algorithm (Frean, 1990), and the tiling algorithm (Mezard and Nadal, 1989). By contrast, RAPTURE uses methods from decision-tree induction (Quinlan, 1986b) to add units to an existing network. However, the current method is limited to adding new input units that directly feed into the output layer. Appropriately modified versions of connectionist methods for growing networks may prove useful in allowing RAPTURE to add new hidden units.

Feldman's (Feldman, 1993) PTR system takes an initial rule base expressed as a collection of Horn clause rules, along with an expert's confidence values in the accuracy of each of these rules. Unlike in RAPTURE, these values do not represent the strength, or amount of evidence suggested by each of these rules, but rather one's confidence that the rule is correct. By using these values, along with a set of training examples, PTR is able to incrementally reformulate the rule base in such a way that is consistent with the training data, as well as maximizing one's confidence in the rules.

Fu (Fu, 1989) and Lacher (Lacher, 1992) have also used backpropagation techniques to revise certainty factors on rules. Fu has apparently derived formulas for CFBP, although they are not given in the paper. Unlike RAPTURE, Fu's method does not implement complete CFBP, but rather uses it only on every other layer of the network, and uses a different hill-climbing method on the alternate layers. Fu claims he chose this approach because the MIN and MAX functions are not differentiable. In RAPTURE, this does not cause a major problem since although these functions are not *everywhere* differentiable, they are trivially so almost everywhere. When working with real-valued weights, the problem of having two non-zero activation levels that are exactly the same, and both being the minimum value into another node has yet to occur in practice. Lacher apparently concurs with this assessment, and has independently implemented a complete version of CFBP. However, the current publications on these two projects do not address the problem of altering the network architecture (i.e. adding new rules) and do not present results on revising actual expert knowledge bases. Fu does, however, present data on a rule base which is initially 100% accurate, and then corrupted by adding contradictory rules.

6 Future Work

There are several areas in which our current methods and experiments can be improved and extended. This section discusses several of these.

The current method for changing network architecture in RAPTURE is restricted to adding new input units that directly feed the outputs. This method has proven sufficient for refining several real-world knowledge bases; however, it is clearly a significant limitation. As mentioned in the previous section, a number of methods have recently been developed for dynamically adding new hidden units to a connectionist network. Work is currently underway to use the upstart algorithm (Frean, 1990) to allow networks to grow new hidden units when it is deemed appropriate.

Another area requiring further research concerns the differences between certainty-factor networks and traditional connectionist networks. The normal method of training a fixed network with initially random weights is also easily applied to certainty-factor networks. In this situation, what are the relative advantages and disadvantages of probabilistic sum as a combination function compared to the normal thresholded linear sum? Experimental studies on a variety of learning problems could help answer this question.

Further comparison of the RAPTURE and KBANN approaches to knowledge-base refinement are also indicated. Specifically, what are the advantages and disadvantages of adding a background of fully-connected low-weighted links versus dynamically adding new units and links as required. The current results indicate that the RAPTURE approach is capable of producing simpler, more accurate knowledge bases; however, these results are complicated by the differences between probabilistic sum and linear threshold units. A comparison of the current version of RAPTURE to an alternative version that uses an initial background network of low-weighted links could help elucidate the advantages and disadvantages of these two approaches.

In recent years, certainty-factors have been the subject of considerable criticism from researchers in uncertain reasoning (Shafer and J. Pearl, 1990). Certainty factors only have a clear probabilistic semantics if very restrictive assumptions about independence are made (Heckerman, 1986). However, the basic revision framework in RAPTURE should be applicable to other uncertain reasoning formalisms such as Bayesian networks (Pearl, 1988), Dempster-Shafer theory (Shafer, 1976), or fuzzy logic (Zadeh, 1965). As long as the activation functions in the corresponding network implementations of these methods are differentiable, backpropagation techniques should be employable. For example, if Bayesian networks are restricted to being singly-connected, there are linear-time algorithms for calculating the desired output probabilities (Pearl, 1988). The resulting equations appear to be amenable to backpropagation. Recent methods for inducing Bayesian networks from data (Geiger et al., 1990; Cooper and Herskovits, 1992) could also prove useful in making structural changes to an existing network. Regarding fuzzy logic, there has been some recent work in using training data and connectionist learning methods to revise fuzzy controllers (Berenji, 1990), and RAPTURE could potentially be extended to deal with these sorts of problems as well.

7 Conclusions

Automatic refinement of probabilistic rule bases is an under-studied problem with important applications to the development of intelligent systems. This paper has described and evaluated an approach to refining certainty-factor rule bases that integrates connectionist and symbolic learning. The approach is implemented in a system called RAPTURE, which uses a revised backpropagation algorithm to modify certainty factors and ID3's information gain criteria to determine new rules to add to the network. In other words, connectionist methods are used to adjust parameters and symbolic methods are used to make structural changes to the knowledge base.

In domains with limited training data or domains requiring meaningful explanations for conclusions, refining existing expert knowledge has clear advantages. Results on revising three real-world knowledge bases indicates that RAPTURE generally performs better than purely inductive systems (ID3 and backpropagation), a purely symbolic revision system (EITHER), and and purely connectionist revision system (KBANN).

The certainty-factor networks used in RAPTURE blur the distinction between connectionist and symbolic representations. They can be viewed either as connectionist networks or symbolic rule bases. RAPTURE demonstrates the utility of applying connectionist learning methods to "symbolic" knowledge bases and employing symbolic methods to modify "connectionist" networks. Hopefully these results will encourage others to explore similar opportunities for cross-fertilization of ideas between connectionist and symbolic learning.

8 Appendix

Using the designed network, we wish to perform backpropagation on the certainty factors (which are the weights on the links between the nodes). Following the presentation given in Rumelhart et al. (1986), we can define the network error due to input pattern p as

$$E_p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2 \tag{7}$$

and total error is similarly measured as $E = \sum E_p$. For any p-sum unit j, the net input to this unit for input pattern p is defined by

$$net_{pj} = CF(w_{ji}o_{pi})\forall i \tag{8}$$

Defining the certainty factor (CF) as in Shortliffe and Buchanan (1975) as measure of belief (MB)− measure of disbelief (MD), where $0 \leq MD \leq 1$, $0 \leq MB \leq 1$, and -1≤CF≤1, this gives us

$$net_{pj} = \sum_{beliefs} w_{ji}o_{pi} - \sum_{disbeliefs} w_{ji}o_{pi} \tag{9}$$

Note that $\sum x_i$ is shorthand for $x_1 \oplus x_2 \oplus ... \oplus x_n$ (the probabilistic sum of the x_i). Since in our representation, disbeliefs are represented as negative numbers, we can rewrite the above as

$$net_{pj} = \sum_{+inputs} w_{ji}o_{pi} + \sum_{-inputs} w_{ji}o_{pi} \tag{10}$$

Further, since in RAPTURE we define a units output as simply the certainty factor of its inputs, we get

$$o_{pj} = net_{pj} \tag{11}$$

In order to achieve gradient descent, we need to have

$$\Delta_p w_{ji} \propto -\frac{\partial E_p}{\partial w_{ji}} \tag{12}$$

The term on the right can be expressed as

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial net_{pj}} \frac{\partial net_{pj}}{\partial w_{ji}} \tag{13}$$

Looking at the second factor, and applying equation 8 gives us

$$\frac{\partial net_{pj}}{\partial w_{ji}} = \frac{\partial}{\partial w_{ji}} CF(w_{jk}o_{pk})\forall k = \frac{\partial}{\partial w_{ji}} \sum_{+inputs} w_{jk}o_{pk} + \sum_{-inputs} w_{jk}o_{pk} \tag{14}$$

It is clear that each input line ji into node j contributes either positively or negatively to the overall certainty factor of node j, depending upon the product of $w_{ji}o_{pi}$. This tells us that one of the two probabilistic sums does not depend upon w_{ji}. Therefore $\frac{\partial net_{pj}}{\partial w_{ji}}$ breaks down into two cases.

1. $\boldsymbol{w_{ji}o_{pi} \geq \mathbf{o}}$: In this case, the partial for the probabilistic sum of the negative inputs is zero (not effected by w_{ji}). Noting that $\frac{\partial}{\partial x_k} \sum_i x_i = 1 - \sum_{i \neq k} x_i$ gives us

$$\frac{\partial net_{pj}}{\partial w_{ji}} = o_{pi}(1 - \sum_{+inputs \neq i} w_{jk}o_{pk}) \tag{15}$$

2. $\boldsymbol{w_{ji}o_{pi} < \mathbf{o}}$: As in above, this gives us

$$\frac{\partial net_{pj}}{\partial w_{ji}} = o_{pi}(1 + \sum_{-inputs \neq i} w_{jk}o_{pk}) \tag{16}$$

These two equations can be combined into the more compact

$$\frac{\partial netpj}{\partial w_{ji}} = o_{pi}(1 \pm \sum_{\mp inputs \neq i} w_{jk}o_{pk}) \tag{17}$$

where it is understood that this represents the two above cases.

If we now define

$$\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}} \tag{18}$$

and combine this with equations 13 and 17 we get

$$-\frac{\partial E_p}{\partial w_{ji}} = \delta_{pj}o_{pi}(1 \pm \sum_{\mp inputs \neq i} w_{jk}o_{pk}) \tag{19}$$

Therefore, in order to implement gradient descent in total error E, we need to make our weight adjustments according to

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi}(1 \pm \sum_{\mp inputs \neq i} w_{jk}o_{pk}), \tag{20}$$

where η is our learning rate. All that now remains is to calculate δ_{pj} for each unit u_j in the network. This can be done by applying the chain rule to our original definition (18), giving us

$$\delta_{pj} \equiv -\frac{\partial E_p}{\partial net_{pj}} = -\frac{\partial E_p}{\partial o_{pj}}\frac{\partial o_{pj}}{\partial net_{pj}} \tag{21}$$

The rightmost factor is trivial. Since $o_{pj} = net_{pj}$ (11), this results in

$$\frac{\partial o_p j}{\partial net_{pj}} = \frac{\partial x}{\partial x} = 1. \tag{22}$$

In order to calculate the left-hand factor, we need to consider two separate cases. Beginning at the top of the network and working our way down, we first consider that u_j is an output unit. From our definition of E_p (7) we see that

$$\frac{\partial E_p}{\partial o_{pj}} = -(t_{pj} - o_{pj}), \tag{23}$$

and by combining this with equations 21 and 22 we get

$$\delta_{pj} = (t_{pj} - o_{pj}) \tag{24}$$

for any output unit u_j. If u_j is *not* an output unit, then we must remember that the output produced may first pass through a min node before its value can be fed-forward. Therefore, the *effective* output for any such node j is either 0, or the net_{pj} value previously calculated. In other words, a probabilistic-sum unit j's value only feeds forward to certain k's, which I label as k_{min}. Clearly, if j's value does not pass successfully through the min node, then its value is not a factor in the network's output. This can be shown formally by again applying the chain rule.

$$\frac{\partial E_p}{\partial o_{pj}} = \sum_{k_{min}} \frac{\partial E_p}{\partial net_{pk}}\frac{\partial net_{pk}}{\partial o_{pj}} = \sum_{k_{min}} \frac{\partial E_p}{\partial net_{pk}}\frac{\partial}{\partial o_{pj}} CF(w_{ki}o_{pi})_{\forall i} = -\sum_{k_{min}} \delta_{pk} w_{kj}(1 \pm \sum_{i \neq j} w_{ki} o_{pi}) \tag{25}$$

Therefore, in order to modify weights using certainty-factor backpropagation, we need to utilize the following three equations:

$$\Delta_p w_{ji} = \eta \delta_{pj} o_{pi}(1 \pm \sum_{k \neq i} w_{jk} o_{pk}) \tag{26}$$

If u_j an output unit

$$\delta_{pj} = (t_{pj} - o_{pj}) \tag{27}$$

If u_j is not an output unit

$$\delta_{pj} = \sum_{k_{min}} \delta_{pk} w_{kj}(1 \pm \sum_{i \neq j} w_{ki} o_{pi}) \tag{28}$$

While backpropagating, the only difficulty occurs when passing through a min node, and determining which of the connected p-sum nodes actually was sent through. This is easily determined, by checking each p-sum node's value, and seeing which one has the same value. The only problem arises when two or more nodes each have the same minimum value, in which case the min-function is not differentiable. This problem turns out to be negligible, as it occurs rarely ($< 1\%$), and usually this involves two or more 0 values. In these cases, RAPTURE simply picks one of these nodes at random and continues backpropagating.

9 Acknowledgements

This research was supported by the National Science Foundation under grant IRI-9102926, the NASA Ames Research Center under grant NCC 2-629, and the Texas Advanced Research Program under grant 003658114. We wish to thank R.S. Michalski for furnishing the soybean data, M. Noordewier, G.G. Towell, and J.W. Shavlik for supplying the DNA data, and the KBANN results, and Yong Ma for providing the MYCIN data.

References

Berenji, H. (1990). Refinement of approximate reasoning-based controllers by reinforcement learning. In *Proceedings of the Eighth International Workshop on Machine Learning*, 475–479. Evanston, IL.

Buchanan, G., and E.H. Shortliffe, e. (1984). *Rule-Based Expert Systems:The MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading, MA: Addison-Wesley Publishing Co.

Cohen, W. (1992). Compiling prior knowledge into an explicit bias. In *Proceedings of the Ninth International Conference on Machine Learning*, 102–110. Aberdeen, Scotland.

Cooper, G. (1987). Probabilistic inference using belief networks is np-hard. Technical Report KSL-87-27, Medical Computer Science Group, Stanford Univ., Stanford, CA.

Cooper, G. G., and Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347.

Fahlman, S., and Lebiere, C. (1989). The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems 2*, 524–532. Denver, CO.

Feldman, R. (1993). *Probabilistic Revision of Logical Domain Theories*. PhD thesis, Department of Computer Science, Cornell University, Ithaca, NY.

Frean, M. (1990). The upstart algorithm: A method for constructing and training feedforward neural networks. *Neural Computation*, 2:198–209.

Fu, L.-M. (1989). Integration of neural heuristics into knowledge-based inference. *Connection Science*, 1(3):325–339.

Gallant, S. (1988). Connectionist expert systems. *Communications of the Association for Computing Machinery*, 31:152–169.

Geiger, D., Paz, A., and Pearl, J. (1990). Learning causal trees from dependence information. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 770–776. Boston,MA.

Ginsberg, A., Weiss, S. M., and Politakis, P. (1988). Automatic knowledge based refinement for classification systems. *Artificial Intelligence*, 35:197–226.

Heckerman, D. (1986). Probabilistic interpretations for Mycin's certainty factors. In Kanal, L. N., and Lemmer, J. F., editors, *Uncertainty in Artificial Intelligence*, 167–196. Amsterdam: North Holland.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the Eighth Annual Conference of the Cognitive Science Society*, 1–12. Amherst, MA.

Lacher, R. (1992). Node error assignment in expert networks. In Kandel, A., and Langholz, G., editors, *Hybrid Architectures for Intelligent Systems*, 29–48. Boca Raton, FL: CRC Press, Inc.

Ling, X., and Valtorta, M. (1991). Revision of reduced theories. In *Proceedings of the Eighth International Workshop on Machine Learning*, 519–523. Evanston, IL.

Ma, Y., and Wilkins, D. C. (1991). Improving the performance of inconsistent knowledge bases via combined optimization method. In *Proceedings of the Eighth International Workshop on Machine Learning*, 23–27. Evanston, IL.

Mezard, M., and Nadal, J. (1989). Learning in feedforward layered networks: The tiling algorithm. *Journal of Physics*, A22(12):2191–2203.

Michalski, R. S., and Chilausky, S. (1980). 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. *Journal of Policy Analysis and Information Systems*, 4(2):126–161.

Mooney, R. J., and Ourston, D. (1991). A multistrategy approach to theory refinement. In *Proceedings of the International Workshop on Multistrategy Learning*, 115–130. Harper's Ferry, W.Va.

O'Neill, M., and Chiafari, F. (1989). Escherichia coli promoters. *Journal of Biological Chemistry*, 264:5531–5534.

Ourston, D., and Mooney, R. (1990). Changing the rules: a comprehensive approach to theory refinement. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 815–820. Detroit, MI.

Ourston, D., and Mooney, R. J. (in press). Theory refinement combining analytical and empirical methods. *Artificial Intelligence*.

Pazzani, M., and Kibler, D. (1992). The utility of background knowledge in inductive learning. *Machine Learning*, 9:57–94.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo,CA: Morgan Kaufmann, Inc.

Quinlan, J. R. (1986a). The effect of noise on concept learning. In Michalski, R. S., Carbonell, J. G., and Mitchell, T. M., editors, *Machine Learning: An Artificial Intelligence Approach, Volume II*, 149–166. Morgan Kaufman.

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

Rumelhart, D. E., Hinton, G. E., and Williams, J. R. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., and McClelland, J. L., editors, *Parallel Distributed Processing, Vol. I*, 318–362. Cambridge, MA: MIT Press.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton, NJ: Princeton University Press.

Shafer, G., and J. Pearl, e. (1990). *Readings in Uncertain Reasoning*. San Mateo,CA: Morgan Kaufmann, Inc.

Shavlik, J. W., Mooney, R. J., and Towell, G. G. (1991). Symbolic and neural learning algorithms: An experimental comparison. *Machine Learning*, 6:111–143.

Shavlik, J. W., and Towell, G. G. (1989). Combining explanation-based and neural learning: An algorithm and empirical results. *Connection Science*, 1(3):325–339.

Shortliffe, E., and Buchanan, B. (1975). A model of inexact reasoning in medicine. *Mathematical Biosciences*, 23:351–379.

Swartout, W. (1981). Explaining and justifying in expert consulting programs. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, 203–208. Vancouver, BC.

Thompson, K., Langley, P., and Iba, W. (1991). Using background knowledge in concept formation. In *Proceedings of the Eighth International Workshop on Machine Learning*, 554–558. Evanston, IL.

Towell, G., and Shavlik, J. (1992). Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. In Lippmann, R., Moody, J., and Touretzky, D., editors, *Advances in Neural Information Processing Systems*, vol. 4. Morgan Kaufmann.

Towell, G. G. (1991). *Symbolic Knowledge and Neural Networks: Insertion, Refinement, and Extraction.* PhD thesis, University of Wisconsin, Madison, WI.

Towell, G. G., Shavlik, J. W., and Noordewier, M. O. (1990). Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 861–866. Boston, MA.

Valtorta, M. (1988). Some results on the complexity of knowledge-base refinement. In *Proceedings of the Sixth International Workshop on Machine Learning*, 326–331. Ithaca, NY.

Valtorta, M. (1990). More results on the complexity of knowledge-base refinement:belief networks. In *Proceedings of the Seventh International Conference on Machine Learning*, 419–424. Austin, TX.

Zadeh, L. (1965). Fuzzy sets. *Information and Control*, 8:338–353.