Appears in *Proceedings of the 1992 Machine Learning Workshop on Integrated Learning in Real Domains,* Aberdeen, Scotland, July, 1992

Combining Symbolic and Neural Learning to Revise Probabilistic Theories

J. Jeffrey Mahoney
Dept. of Computer Sciences
University of Texas
Austin, TX 78712
mahoney@cs.utexas.edu

Raymond J. Mooney
Dept. of Computer Sciences
University of Texas
Austin, TX 78712
mooney@cs.utexas.edu

Abstract

This paper describes RAPTURE — a system for revising probabilistic theories that combines symbolic and neural-network learning methods. RAPTURE uses a modified version of backpropagation to refine the certainty factors of a Mycin-style rule-base and it uses ID3's information gain heuristic to add new rules. Results on two real-world domains demonstrate that this combined approach performs as well or better than previous methods.

1 Introduction

Automatically revising an imperfect knowledge base to correctly classify a set of training examples (theory refinement) has proven to be an important task that lends itself to integrating analytical and empirical learning methods. Several theory refinement systems have been successfully applied to real-world problems [Ginsberg, 1990; Ourston and Mooney, 1990; Towell *et al.*, 1990]. To date, theory refinement has largely focused on revising logical Horn-clause theories. However, many real-world domains require some form of probabilistic reasoning [Shafer and J. Pearl, 1990]. The primary advantage of probabilistic methods is their ability to combine evidence from several sources and draw a conclusion based on the total amount of evidence for each possible decision. For instance, knowing a general rule: $a \wedge b \wedge c \wedge d \rightarrow C_1$ for classifying an example into category C_1, an expert might still be inclined to conclude C_1 if there was strong evidence for $a \wedge b \wedge c$ even though there was no evidence for d. This is especially true in cases where the example does not exactly fit into *any* predefined category. An automated classification system using pure Horn-clause logic is not capable of reasoning in this manner. Once d is found to be *false*, the rule is ignored.

Flexible matching methods in inductive rule learning [Michalski and Chilausky, 1980; Michalksi *et al.*, 1986] and theory revision [Mooney and Ourston, 1991] are one way to deal with this problem. In this approach, a score is calculated measuring how well an example matches each symbolic rule and the rule with the greatest score is invoked. Another approach is to use N-of-M rules [Towell and Shavlik, 1991; Ginsberg *et al.*, 1988], which fire if at least N of their M antecedents are satisfied, e.g. 3 of $a, b, c, d \rightarrow C_1$. Unfortunately, this approach does not consider the relative strengths of each of the antecedents. Neural networks, on the other hand, use connection weights to encode relative strengths. By representing the theory as a neural network, standard backpropagation [Rumelhart *et al.*, 1986] can be used to modify the weights representing rule strengths [Fu, 1989; Towell *et al.*, 1990; Lacher, 1992]. Since a unit's activation-level depends upon a linear sum of all incoming activations, this is an effective approach to combining evidence.

In this paper, we describe the RAPTURE system (Revising Approximate Probabilistic Theories Using Repositories of Examples), which combines symbolic and neural approaches. RAPTURE takes a Mycin-style probabilistic theory [Shortliffe and Buchanan, 1975] and converts it into a network. The certainty factors on the rules are mapped into weights of 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, in a RAPTURE network, the total input is the *probabilistic* sum of the incoming activations. Unlike standard neural networks, no thresholding output function is needed since the probabilistic sum already provides the needed non-linearity.

Once the network is built, it is modified to properly classify a repository of training examples. The 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 to perform gradient descent for certainty-factor output functions such

as probabilistic-sum, MIN, and MAX. If all examples can be classified correctly through back-propagation 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, 1986] 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 the network has been trained, the revised rules can be read directly off of the network – no retranslation is necessary. Unlike KBANN [Towell and Shavlik, 1991], the direct correspondence between weighted links and probabilistic rules removes any distinction between the symbolic and connectionist representations.

The rest of this paper is organized as follows. Section 2 presents the RAPTURE algorithm in some detail. Section 3 shows some preliminary results on a couple of real-world data-sets. Section 4 discusses future work, Section 5 discusses related work, and we conclude in section 6.

2 The RAPTURE Approach

Mycin-style rules are of the form: $A \xrightarrow{0.8} B$, stating that belief in proposition A gives a 0.8 *Measure of Belief* in proposition B. The certainty-factor formalism was chosen for a variety of reasons. First, it is perhaps the simplest method that retains the evidence-summing aspect of probabilistic reasoning. As each rule fires, it contributes a piece of evidence for its consequent. All evidence is then *probabilistically* summed, giving a total measure of belief in the consequent. Mycin's use of probabilitic sum ($a \oplus b \equiv a + b - ab$) enables many small pieces of evidence to add up to significant evidence. This is lacking in formalisms that use MIN or MAX to combine evidence [Ling and Valtorta, 1991]. Second, probabilistic sum is a simple, differentiable, non-linear function. This is crucial for implementing gradient descent using backpropagation. Finally, and perhaps most significantly, is the popularity of certainty factors. Numerous knowledge-bases have been implemented using the Mycin model, which immediately gives our approach a wide degree of applicability.

2.1 Building the Network

The process of converting a Mycin-style probabilistic theory into a network is straightforward. The result will be called a *conceptual network*, in accordance with [Fu, 1989]. Building the conceptual network begins by mapping all identical symbols in the rules to the same node in the network. Input features (those only appearing as rule-antecedents) become input nodes, and are 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 of the links adjoining nodes. Networks for classification problems contain one output for each category. When an example is presented, the measure of belief in each of the categories is computed and the example is assigned to the category with the highest value.

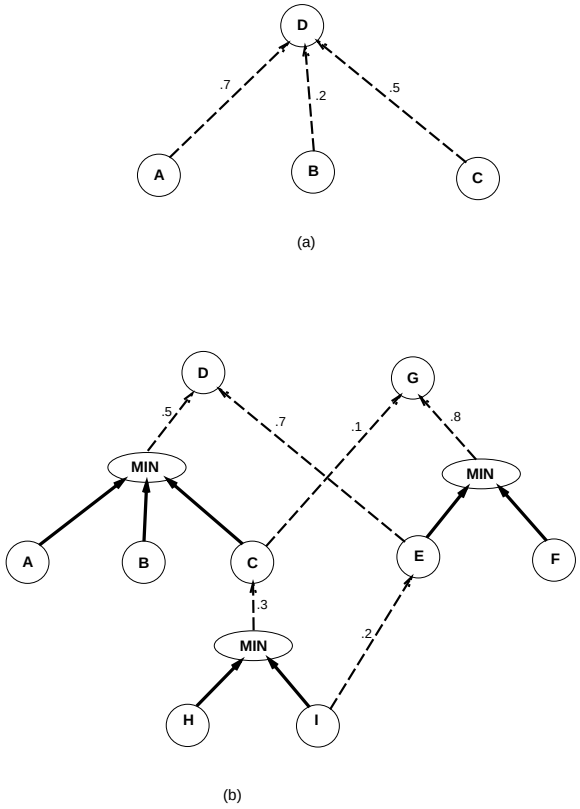

Figure 1: Building Simple Conceptual Networks

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 conceptual network for this rule set. These rules state that A,B, and C all contribute evidence towards D in varying degrees. Thus to calculate our Measure of Belief in D based on these rules, we probabilistically sum the input activations (A,B,C) multiplied by their corresponding certainty factors. If, for example, our current belief in A is 1, B is 0, and C is .6, then our current belief in D is $(1 \times .7) \oplus (0 \times .2) \oplus (.5 \times .6) \equiv .7 \oplus 0 \oplus .3 = .79$.

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. The standard Mycin operator

for handling explicit conjunctions and disjunction are the MIN and MAX functions, respectively. Figure 1(b) illustrates the following more complete set of rules.

$$ABC \xrightarrow{.5} D \qquad E \xrightarrow{.7} D \qquad C \xrightarrow{.1} G$$
$$EF \xrightarrow{.8} G \qquad HI \xrightarrow{.3} C \qquad I \xrightarrow{.2} E$$

As shown in the network, conjuncts must first pass through a MIN node before any activation reaches the consequent node. Therefore, in the first rule, only one of the activations from among A,B, and C (the one with the minimum value) will multiply by the .5 certainty factor, giving evidence for D. Note that each of the conjuncts is *directly* connected (i.e., has weight 1) to the corresponding MIN mode. This link is non-adjustable, and simply passes the full activation value to the input of the MIN node. MAX nodes are used analogously to represent antecedent disjunction. Thus a rule of the form $A \vee B \vee C \xrightarrow{x} D$ would require that each of A,B, and C's activations pass through a MAX node before being multiplied by the certainty factor.

This construction shows how easily a neural-network can model a Mycin rule-base. In fact, they are isomorphic! Each representation can be converted into the other, without any loss of information. They are two equivalent representations of the same rule-base.

2.2 Certainty Factor Back-Propagation

Using the constructed conceptual network, we desire to minimize the overall error at each of the output nodes with respect to the training examples. By cycling through the examples and for each one slightly adjusting all of the 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.

2.2.1 Backpropagation

Backpropagation (also referred to as the generalized delta rule) is the standard algorithm for approximating gradient descent in a multi-layer network, i.e. a network with hidden units that are neither inputs nor outputs. It replaced the earlier perceptron learning algorithm [Rosenblatt, 1962]) which performed the same function on single-layer networks (perceptrons). With no hidden units, gradient descent is *guaranteed* to find a solution, if one exists. Unfortunately, no such claim is possible for multi-layer networks.

For each example, the normal backpropagation 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. In a normal neural network, 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 (usually a thresholding 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

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

2.2.2 Using Certainty Factors

We cannot, however, use these standard back-propagation formulas for our system, since we are trying to model Mycin's summation of evidence. In order to achieve gradient descent on this network, it is necessary to first derive the corresponding formulas for Certainty Factor Back-Propagation (CFBP).

The main distinction is the manner in which inputs to a node combine to give the total net input (net_{pj}). A conceptual network uses probabilistic sum instead of normal addition to determine the net input. First, all positive inputs ($w_{ji} o_{pi} \geq 0$) are combined with $\sum w_{ji} o_{pi}$ (where $\sum$ represents a probabilistic sum: $a \oplus b \equiv a + b - ab$). This results in a number between 0 and 1. Similarly, all negative inputs are combined, where probabilistic sum states $a \oplus b \equiv a - b - ab$. This results in a number between -1 and 0. These two numbers correspond to Shortliffe's Measures of *Belief* (MB) and *Disbelief* (MD) [Shortliffe and Buchanan, 1975]. The net input is calculated as MB+MD, which represents the certainty factor of the symbol (node) concluded from the combination of input rules, and can take on values between -1 and +1 inclusive. As this value is passed directly on as the nodes output, we are defining $o_{pj} = net_{pj}$, hence the output function is identity. In order to perform back-propagation on a network such as this, the following equations must be utilized. For the derivation of these equations, see [Mahoney, 1992].

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

If u_j is an output unit

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

If u_j is not an output unit

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

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-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 conceptual networks exactly as standard backpropagation does for neural networks. This adjusts all of the certainty factors to locally minimize the mean-squared error over all training examples.

Since 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 only achieved when there is a rule with a certainty factor of 1 whose antecedents all have a certainty factor of 1. 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 any other category. No error propagation takes place in this case ($\delta pj = 0$).

One remaining issue is when to stop adjusting weights. The easiest stopping criteria would be when all examples are classified correctly, but this criteria is not used for two reasons. First, backpropogation is not guaranteed to converge to 100% training accuracy — it may reach a local minimum. Further, just because all of the examples are classified correctly does not mean that further weight adjustment is useless. Perhaps further adjustments will minimize the error to an even greater extent. Correct classification does not mean 0 error. A better criteria is to stop weight adjustment when the error has bottomed out. RAPTURE checks the total mean-squared error every 10 epochs, and if the error ever decreases by less than ϵ (.001), CFBP halts.

2.3 Changing the Network Architecture

Whenever training accuracy fails to reach 100% through CFBP, it is possibly a sign that the network architecture is insufficient for the classification task at hand. To date, we have implemented two ways of changing the architecture. First, whenever the weight of a link goes to zero or changes sign, it is removed from the network and all lower nodes that become detached from the rest of the network are also removed. Deletion is performed dynamically during CFBP, where links are checked for deletion after every 10 epochs. If a link weight goes to zero, then any node beneath this link cannot contribute to the output except through another path. Further, if the weight actually crosses zero, then a rule which was proposed by the expert as positive (negative) evidence is now being used as negative (positive) evidence, indicating that something is terribly wrong with the rule.

RAPTURE also has a method for adding new nodes to the network. Specific nodes are added in an attempt to get more of the training examples classified correctly. The problems are deciding which node to add, and where to insert it in the network. Since we want new evidence that will make one or more output nodes classify a group of examples differently, it makes sense to add new input nodes that connect directly, either positively or negatively to one or more output nodes. These new nodes should be designed to help the network distinguish between the training examples that are being misclassified.

Let us define C as the set of all possible categories into which an example may be classified. We define S_i to be the set of false negative examples for category C_i. These are those examples whose target category is C_i, yet the network 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 S_i. We can define L_i as the set of all $C_{j \neq i}$ such that for some example in S_i, C_j had a higher certainty factor than C_i. Finally, define T_i as the set of all examples whose target category is in L_i.

Now we have two disjoint groups of examples. S_i—the false negatives for C_i, and T_i—the true positives for all categories that are being confused with C_i. How can we better discriminate between these two sets of examples? By utilizing Quinlan's ID3 metric [Quinlan, 1986], we can find the best feature-value combination of the examples that discriminate these examples. Once we have this, add it into the network.

Information gain, as calculated by ID3, works by looking at features in the domain of examples, and determines how well this feature separates the examples in accordance with expert classification. It then selects the feature that best discriminates the examples. This works well for building a decision-tree, but is not exactly what is needed for our conceptual network. What we have is two sets of examples, and we need to discover a feature that is highly prominent in one of the sets, yet lacking in the other. ID3 information gain can easily handle this, if we consider every possible feature-value pair in the domain of the examples as a binary feature. By labelling each example from S_i as *negative*, and those from T_i as *positive*, then for every feature-value pair, we can determine the percentage of examples from each of the two sets that do and do not have this feature-value pair. Information gain is then calculated for each of these pairs, and the one with maximal gain is selected. Ties are broken at random.

Once this feature-value has been selected, we build a new node to use it 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 T_i yet lacking in S_i. In this case, we invert the value of

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

the feature. Placing a small positive weight on the rule gives us a rule of the form $COLOR = RED \stackrel{.1}{\rightarrow} C_i$, or $COLOR \neq RED \stackrel{.1}{\rightarrow} C_i$ which serves as new positive evidence for C_i. We build similar rules (using the same feature-value pair) for each of the categories in set L_i with small negative weights. This gives new negative evidence for all of these categories. The examples in S_i will now be more likely to be classified in category C_i.

This is performed for each $C_i \in C$. Note that for some C_i, the false negative list will be empty, as every example will be classified correctly. This produces no new nodes in the network. We do not worry about false positives. These are examples that are *not* supposed to be classified as C_i, but are nonetheless. These will turn up as false negatives for another 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 adding new nodes) repeats until all training examples are correctly classified. Once this has occurred, the network is considered trained, and testing may begin.

3 Current Experimental Results

So far, the RAPTURE system has been tested on two real-world domains. The first uses a theory for diagnosing soybean diseases. The second is a domain for recognizing DNA-promoter sequences from strings of nucleotides. These datasets are discussed in more detail in the following sections.

3.1 SOYBEAN Results

The Soybean Data comes from [Michalski and Chilausky, 1980] and is a dataset of 562 examples of diseased soybeans. Examples were 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 dataset has been used as a benchmark

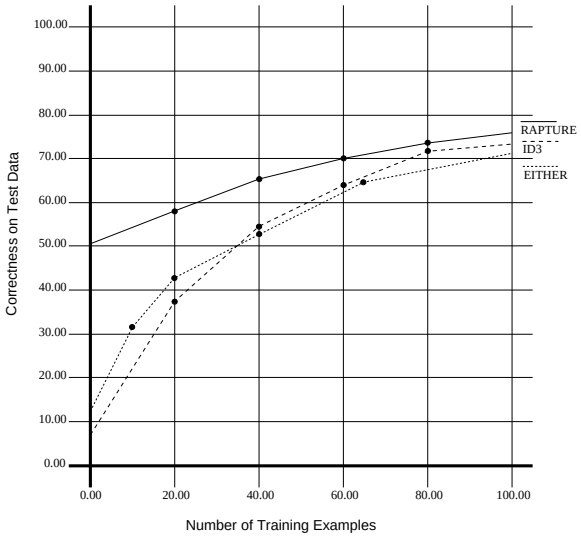

Figure 3: Learning Curves for Soybean Data

for a number of learning systems. Figure 2 is a learning curve on this data comparing RAPTURE, EITHER, and ID3. As is clear from the graph, RAPTURE is given a considerable headstart with its initial theory that was provided by the experts. The theory used by RAPTURE was a certainty factor version of the expert theory given in [Michalski and Chilausky, 1980]. Instead of using Horn clauses, each conjunct in a rule's antecedent was used as separate evidence for its disease. The original theory contains both rules labelled as significant, that highly indicate a particular disease, and others labelled confirmatory, which only corroborated other evidence. Initial certainty factors were assigned to reflect this. Pieces of evidence from significant rules were given certainty factors high enough to give a .9 degree of belief if each of the conjuncts from the original rule were true. For the confirmatory pieces of evidence, this value was .1.

Training proceeded slowly, as CFBP by itself usually peaked out at 90% training accuracy, and three or four rounds of node addition were common. Training 100 examples commonly took 4 hours of computation time. It was interesting to see that the same two or three diseases were constantly being confused, and RAPTURE focused much of its time getting these correct. About 2/3 of the diseases had all or their examples correctly classified after the first application of CFBP.

RAPTURE maintains its headstart throughout testing. Through 60 examples, a t-test shows that RAPTURE is performing significantly better than ID3 (at the .05 level of significance), and continues to hold a slight lead through 100 examples. As this data was the average over only 10 trials, the last two points on the plot (80 and 100 examples) show no significant difference between RAPTURE and ID3, though it is suspected

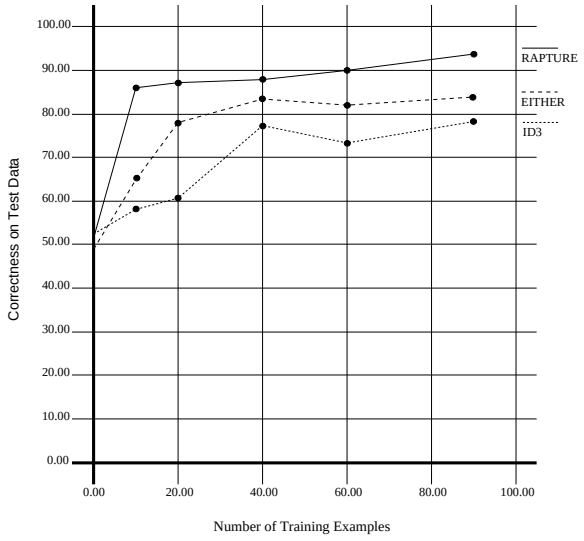

Figure 4: Learning Curves for DNA Data

that more trials would reveal a significant difference. EITHER was also tested on this dataset, though it was run using different training and testing sets. Its performance is clearly worse than RAPTURE's. This was a result of many examples not being classified into *any* category. By resorting to a partial-matching technique, and all examples were assigned to the category that was most nearly fired, EITHER's performance matches that of RAPTURE, almost identically[Mooney and Ourston, 1991].

3.2 DNA Results

The DNA theory is designed to recognize prokaryotic *promoters* in strings composed of nucleotides. A 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]. This is a dataset of 106 examples, for which 53 are examples of promoters, and 53 are not. Each example consists of a sequence of 57 DNA nucleotides. The original theory for this recognition task comes from [O'Neill and Chiafari, 1989], and is described as an N-of-M theory, which should be beneficial for a probabilistic system such as RAPTURE. Once again, this theory was modified for RAPTURE so that if every conjunct of a rule fired, this would lead to a .9 certainty factor. Figure 3 provides a learning curve for this data against EITHER. This graph clearly demonstrates the advantages of an evidence summing system over a pure Horn-clause system such as EITHER. Though they both start near 50% accuracy, RAPTURE very rapidly is up over the 90% mark. ID3 has similar difficulty in learning this N-of-M type of theory. This data has also been run on K-BANN, and although the training and test sets are different comparisons, K-BANN runs about 1-percentage point higher. Whereas RAPTURE's accuracy is at 93.8% after 90 training examples, K-BANN is slightly over 95%[Towell and Shavlik, 1991]. It is not known whether or not this difference is statistically significant.

Interestingly, RAPTURE found this data very easy to classify. CFBP alone did all of the training, as the node addition module was never called. Generally training completed in the order of 20 to 30 epochs, and usually took less than a minute.

4 Future Work to be Done

At this point, we are quite pleased with the CFBP stage of the algorithm. We feel that given the correct network architecture, CFBP will properly adjust the weights in a manner that will generalize well to unseen examples. This was confirmed by the DNA results. Much remaining work is yet to be done, however, on restructuring the network architecture.

We have begun looking into new ways to add new nodes into the network to better classify troublesome examples. Currently Cascade-Correlation [Fahlman and Lebiere, 1989] and the Tiling algorithm [Mezard and Nadal, 1989] seem to hold promise.

Of further interest is in how conceptual networks perform on standard neural network problems. Comparison studies will be done to see if gradient descent on a neural network performs comparably to CFBP on a conceptual network. We also hope to see how well RAPTURE performs when starting with a K-BANN network, where all available features are already in the network with low weights.

Other than running RAPTURE on a number of other domains, we will explore the limitations of this symbolic-connectionist problem-solving technique. We hope to show that this idea is extendible to other probabilistic formalisms such as Bayes-Nets, Dempster-Shafer, or fuzzy logic.

5 Related Work

Much work has already been done in revising probabilistic knowledge bases. Ginsberg [Ginsberg *et al.*, 1988] was among the first to explore the idea of using N-OF-M rules, and his SEEK algorithms devised various ways of adjusting N and M.

Valtorta [Ling and Valtorta, 1991] has looked into rules with varying strengths, and building networks out of these rules. Most of his work has been in proving that the problem of finding a correct set of strengths to exactly match the expert is NP-Hard.

Closer to this project, Gallant [Gallant, 1988] has explored using a connectionist network as an expert sys-

tem for solving diagnosis problems. Towell [Towell and Shavlik, 1991] has built neural networks out of symbolic rules, where all features of the examples are present in small weighted links. Symbolic rules are then extracted in an N-OF-M manner. Fu [Fu, 1989] and Lacher [Lacher, 1992] have both designed algorithms for creating conceptual networks out of Mycin-style rules. Fu only performs a partial back-prop, then resorts to a hill-climbing on classification accuracy. Lacher has done nothing in the area of node addition, and presents no results.

6 Conclusions

The ability to revise probabilistic theories is critical to applying theory revision to many real-world problems. This paper has described and evaluated an approach to revising certainty-factor rule bases that integrates neural and symbolic learning methods. This 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 theory.

Initial results in two real-world domains indicate that RAPTURE performs better than a standard inductive system (ID3) and a logic-based theory revision system (EITHER), and performs as well as a more traditional neural network approach (KBANN) and a logic-based system (EITHER) enhanced with partial matching.

7 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 also wish to thank M. Noordewier, G.G. Towell, and J.W. Shavlik for supplying the DNA data.

References

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

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

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

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

[Ginsberg, 1990] A. Ginsberg. Theory reduction, theory revision, and retranslation. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 777–782, Detroit, MI, July 1990.

[Lacher, 1992] R.C. Lacher. Expert networks: Paradigmatic conflict, technological rapprochement. *Neuroprose FTP Archive*, pages 1–24, 1992.

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

[Mahoney, 1992] J. Mahoney. Combining symbolic and neural learning for theory revision. *Dissertation Proposal–To be completed*, 1992.

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

[Michalksi *et al.*, 1986] R.S. Michalksi, I. Mozetic, J. Hong, and N. Lavrac. The multi-purpose incremental learning system AQ15 and its testing application to three medical domains. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 1041–1045, Philadelphia, PA, Aug 1986.

[Michalski and Chilausky, 1980] R. S. Michalski and S. Chilausky. 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, 1980.

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

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

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

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

[Rosenblatt, 1962] F. Rosenblatt. *Principles of Neurodynamics*. Spartan, New York, 1962.

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

[Shafer and J. Pearl, 1990] G. Shafer and eds. J. Pearl. *Readings in Uncertain Reasoning*. Morgan Kaufmann, Inc., San Mateo:CA, 1990.

[Shortliffe and Buchanan, 1975] E.H. Shortliffe and B.G. Buchanan. A model of inexact reasoning in medicine. *Mathematical Biosciences*, 23:351–379, 1975.

[Towell and Shavlik, 1991] G. Towell and J. Shavlik. Refining symbolic knowledge using neural networks. In *Proceedings of the International Workshop on Multistrategy Learning*, pages 257–272, Harper's Ferry, W.Va., Nov. 1991.

[Towell *et al.*, 1990] G. G. Towell, J. W. Shavlik, and Michiel O. Noordewier. Refinement of approximate domain theories by knowledge-based artificial neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 861–866, Boston, MA, July 1990.