Appears in *The Intl Symposium on Integrating Knowledge and Neural Heuristics (ISIKNH-94)*, pp. 75-85, Pensacola, FL, May, 1994

Modifying Network Architectures for Certainty-Factor Rule-Base Revision

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

March 11, 1994

Abstract

This paper describes RAPTURE — a system for revising probabilistic rule bases that converts symbolic rules into a connectionist network, which is then trained via connectionist techniques. It uses a modified version of backpropagation to refine the certainty factors of the rule base, and uses ID3's information-gain heuristic (Quinlan, 1986) to add new rules. Work is currently under way for finding improved techniques for modifying network architectures that include adding hidden units using the UPSTART algorithm (Frean, 1990). A case is made via comparison with fully connected connectionist techniques for keeping the rule base as close to the original as possible, adding new input units only as needed.

1 Introduction

The task of rule base (theory) revision is one of taking an approximate domain theory (set of rules given by from an expert), and fine-tuning it in order that it more accurately reflects the expert's decision making process. This is commonly achieved through the use of preclassified examples of the target concept that guide the revision process. Many techniques, both symbolic and connectionist, have recently been developed for precisely this task (Shavlik and Towell, 1989; Towell, 1991; Ourston and Mooney, 1990; Pazzani and Kibler, 1992; Cohen, 1992; Fu, 1993). These methods demonstrate the value of beginning with an initial set of rules, provided by an expert, that may only be partially correct. When compared to pure inductive learning, these knowledge rich methods show superior rates of learning, usually requiring fewer examples.

One of the weaknesses of the connectionist approaches to date is the inability to modify their network architectures. The approach of KBANN (Towell, 1991) is to include every possible feature from the domain in the network, fully connecting, and letting those links that don't significantly contribute drop out, while training the network via backpropagation. In extremely complex domains, this can result in excessively large networks that can become unmanageable. Research presented here indicates that such full connectivity can actually slow down learning, not only in terms of training time, but also in terms of numbers of training examples required.

The system described here, RAPTURE (**R**evising **A**pproximate **P**robabilistic **T**heories **U**sing **R**epositories of **E**xamples), attempts to overcome this problem by adding to and deleting from the initial rule base only as deemed necessary. RAPTURE begins with an initial rule base expressed as certainty-factor rules, which is converted into a connectionist network. Training proceeds with a modified version of backpropagation using pre-classified training examples. When backpropagation fails to train the network to 100% accuracy on the training data, network modification is attempted. Currently, rules are added through the use of ID3's information-gain metric. Rules can also be deleted through weight decay. Work in progress includes attempting to add hidden units to the network using a modified version of the UPSTART (Frean, 1990) algorithm.

In the next section, we describe the RAPTURE system. Results making comparisons to fully connected techniques are presented, as well as preliminary UPSTART results. This is followed by related and future work, and conclusions.

2 The Rapture Algorithm

The RAPTURE algorithm breaks down into three separate modules. First, an initial set of probabilistic rules is converted into a RAPTURE network. This initial set of rules (also known as a domain theory) commonly comes from an expert in the domain, and is quite often incomplete or otherwise flawed. It does, however, contain useful knowledge for solving the problem at hand, which is how it helps the learning process.

These probabilistic rules are of the form

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

where the numbers above the arrows represent certainty factors stating how much evidence each of A, B, and C contribute towards concluding proposition D. These certainty factors range from -1 to $+1$, and combine as in MYCIN, which makes use of probabilistic sum. Probabilistic sum allows two pieces of evidence (x and y) to combine to give resulting evidence

$x + y - x * y$. Further details on the combining functions can be found in (Buchanan and E.H. Shortliffe, 1984).

2.1 Building the Network

Given these probabilistic rules, a RAPTURE network is built from them via a one-to-one mapping. Each rule in the rule base corresponds to one link in the network. The certainty-factor of the rule is the weight that is attached to the link. The nodes on either side of the link represent the rules antecedent and consequent. Each unique proposition in the rule base corresponds to one unique node in the network. In the case of an antecedent containing multiple propositions (that are joined through either conjunction or disjunction), each of the antecedent nodes must pass through a min (or max) node before being linked to the consequent node. The trivial example above is shown in Figure 1(a). Figure 1(b) illustrates the following more complete set of rules.

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

Once built, we have an exact representation of the symbolic rule base. There is no information lost in the conversion. The elegance of this representation is the fact that it is now possible to view the rule base as either a set of symbolic rules, or as a connectionist network. Once the network has been trained, symbolic rules can be read directly off of the network without any need for translation. This approach bridges the gap between symbolic versus connectionist learning.

2.2 Certainty-Factor Backpropagation

The resulting network is then trained using the technique of gradient descent to minimize error in the network. In order to achieve this, standard backpropagation can not be used directly, as the combining functions for certainty-factors are different from those of standard neural networks. Because of this, the gradient descent formulae had to be modified accordingly, resulting in what we label Certainty-Factor Backpropagation (CFBP). Specifically, the formulae used are

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

If u_j is an output unit

$$\delta_{pj} = (t_{pj} - o_{pj}) \tag{2}$$

If u_j is not an output unit

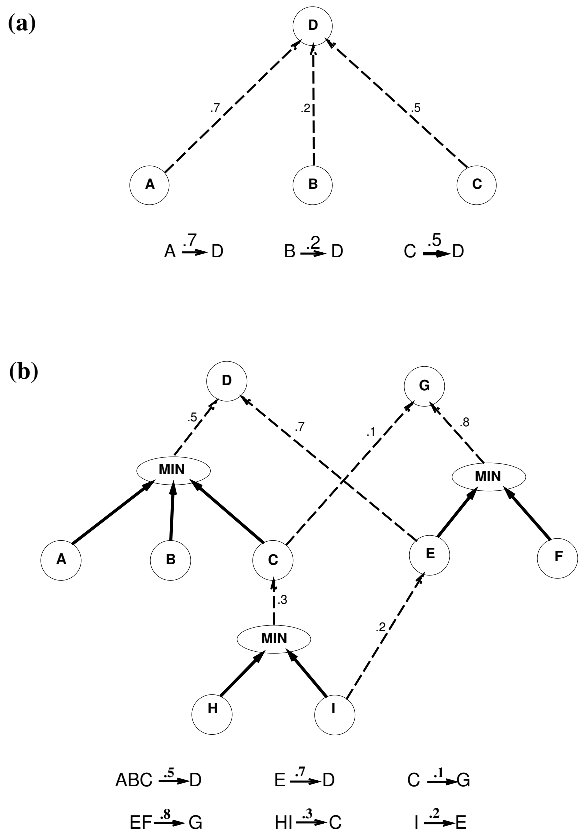

Figure 1: Converting Rules into Rapture-Networks

$$\delta_{pj} = \sum_{k_{min}} \delta_{pk} w_{kj} (1 \pm \sum_{i \neq j} w_{ki} o_{pi}) \tag{3}$$

The "Sigma with circle" notation represents 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.

CFBP is simply gradient descent learning where the standard backpropagation formulae are modified to work with certainty-factor combining functions. Further details, along with a derivation of these formulae can be found in (Mahoney and Mooney, 1993).

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 $< \epsilon$.
2. If not 100% training accuracy, delete zero-valued links from the network. 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

2.3 Architecture Modification

Unfortunately, CFBP alone is often insufficient by itself to fully train the network. RAPTURE-networks are considered fully trained when they achieve 100% accuracy over the training examples. This inability to fully train is usually an indication that the network architecture as given by the domain expert is incomplete or inaccurate, and needs to be altered. RAPTURE contains simple means for adding and deleting links from the network.

Whenever the value of a link comes within 0.01 of zero after training, it is deleted from the network. This is due to the fact that no values are propagating forward through this link, and its removal can have no effect on the network. Similarly, RAPTURE contains a simple mechanism for adding links into the network. This is achieved through utilizing ID3's (Quinlan, 1986) information-gain metric. For each output category with misclassified examples, a search is made for the best feature-value combination (e.g., `COLOR-RED`) that could be used to help correctly classify these examples, using information-gain to discriminate examples. An input node with this feature-value is linked directly into the output node with small positive weight, and similarly with small negative weight into the other output nodes.

Adding links in this fashion has enabled RAPTURE to completely train essentially all of the domains studied to date. However, the original RAPTURE contains no mechanism for creating hidden units in the network, which is the source of current research. An overview of the RAPTURE algorithm is depicted in Figure 2.

3 Comparisons with Fully Connected Systems

Figure 3 presents learning curves for the DNA promoter-recognition problem. 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, 1991). 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

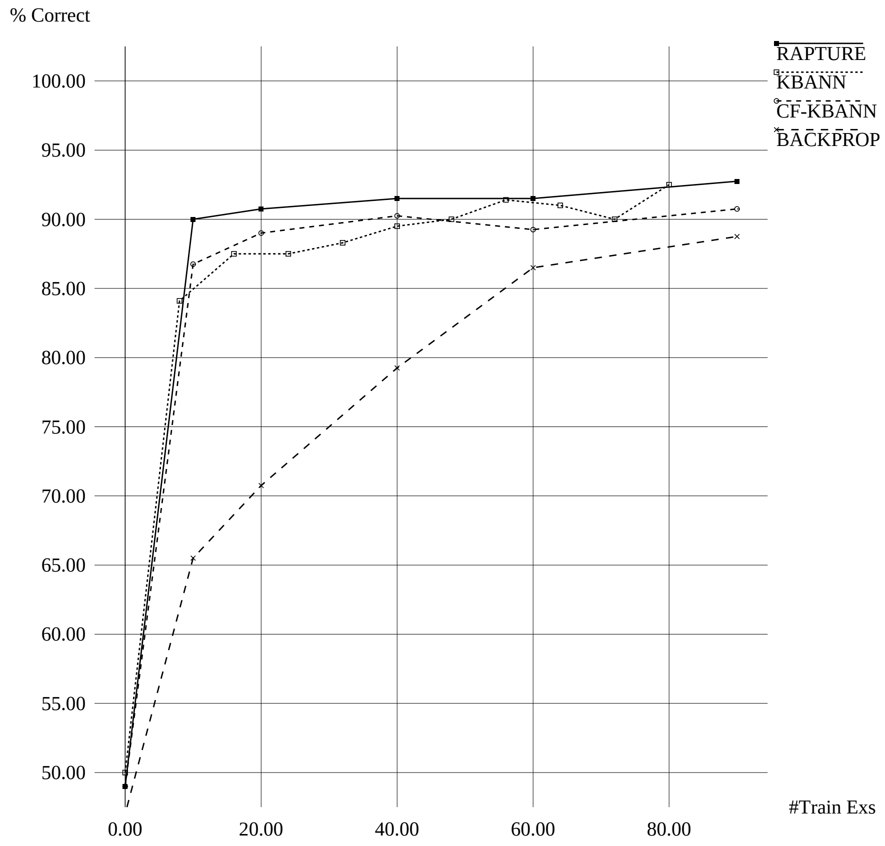

Figure 3: PROMOTER Testing Accuracy

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 used as an initial theory for RAPTURE, KBANN, and CF-KBANN.

This graph is a plot of average performance in accuracy at classifying novel test examples over 25 independent trials. It is significant that RAPTURE is performing better than the fully connected systems. KBANN (Towell, 1991) is a neural network that has been initialized with expert knowledge. CF-KBANN is a hybrid system that is an implementation of KBANN that uses certainty-factor combining functions and CFBP. T-tests confirm that RAPTURE is performing significantly better than the other systems with the exception of KBANN (which was run on differing train and test splits for which we do not have the data). These plots are an indication of the fact that adding in every possible feature and fully connecting can actually hurt performance. It is for this reason that more steamlined methods for adding in new units and modifying network architecture are desired.

4 Creating Hidden Units

One method for creating hidden units that we are currently exploring uses the UPSTART algorithm (Frean, 1990). This is a connectionist technique for adding new hidden units directly below an output unit. For each output unit that has mistaken examples, all of the false negative and false positive examples are collected. Two new hidden nodes are then created, and placed directly below this output unit. These new units are designed to correct the mistaken examples. One of these units is trained to be active for all of the false positives. This unit then connects with negative weight to the output unit. Similarly, the other new unit is trained to be active for the false negatives, and its output is connected positively to the output unit. If an output unit has only false positives (or only false negatives), then only one hidden unit need be created. Assuming the hidden units are trained correctly, the mistaken examples will now be correctly classified.

This algorithm fits quite well into the overall RAPTURE scheme. Once CFBP and ID3 show no improvement in overall mean-squared error, the UPSTART module is called. The algorithm checks each output unit with mistaken examples and applies UPSTART. This simply means that two new hidden units are created, which have no input units, and are connected (one positively, one negatively) to the output unit. The task is now reduced to training each new unit to become active only on its particular examples (say the false negatives). RAPTURE is then called recursively on the new unit, with the false negatives as positive examples, and the rest of the examples as negatives. This node, then begins getting new input features (from the original node addition module), whose weights are then massaged via CFBP. This continues recursively, until finally all training examples are successfully categorized.

To date, experiments with the UPSTART module have only been run on one dataset. This is the soybean dataset. 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 most significant result of this experiment is the decrease in training time. Without UPSTART, convergence to 100% training accuracy does not occur for RAPTURE on training

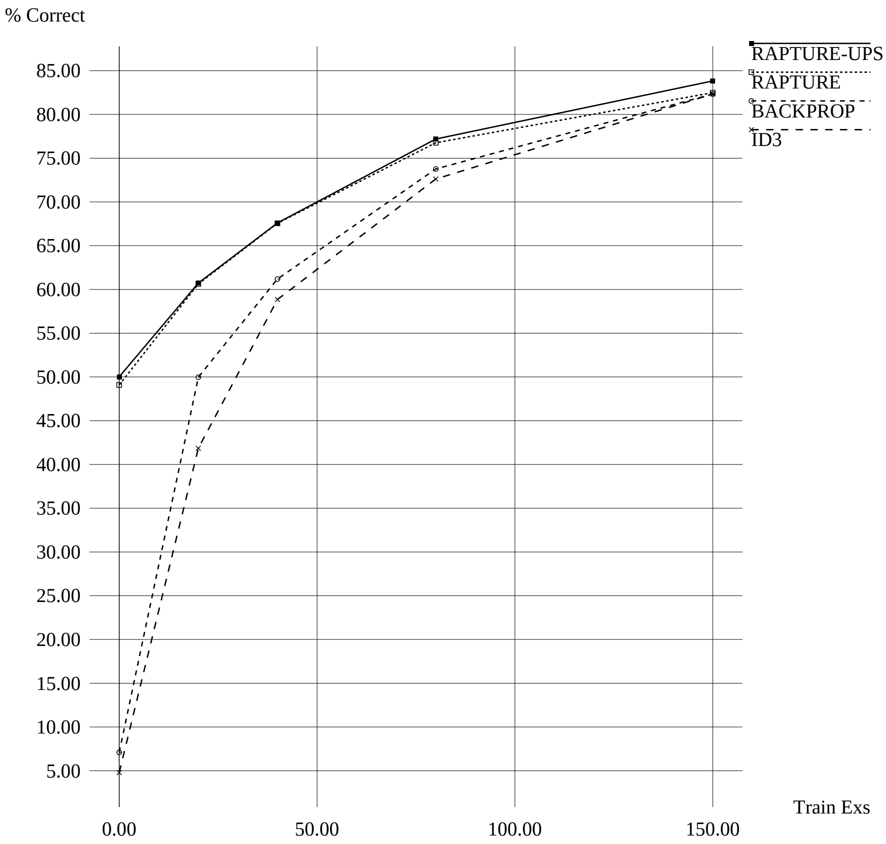

Figure 4: SOYBEAN Testing Accuracy

sets larger than 150 examples. With UPSTART, convergence has been successful up to 300 examples. Unfortunately, this increase in training accuracy does not seem to carry over to generalization accuracy in this domain. This may be an indication that UPSTART is only causing a number of examples to be memorized.

5 Related and Future Work

The most closely related work to this has been done in conjunction with KBANN. Towell's original work contained a suggestion for adding new hidden units into a KBANN-net that would seem to work only in very specialized domains, such as Promoter Recognition. This was achieved through adding "cone" units that would connect contiguous features. Opitz (Opitz and Shavlik, 1993) recently created a means for adding new hidden nodes that link into the nodes in the KBANN network with the highest error rates. These nodes are selected heuristically by measuring their performance on sets of tuning examples. Lacher (Lacher, 1992) has independently developed a very similar backpropagation technique (ENBP) for use with EMYCIN expert networks, though there apparently has been no work on network architecture modification.

Much of the future work for this project will be devoted to discovering new ways to alter network architectures to more accurately classify data. The UPSTART algorithm definitely holds promise, though a search for a good domain to fully demonstrate its potential is currently underway. Work is also underway investigating the differences between standard backpropagation and CFBP. Since the combining functions are significantly different, it is not known how their convergence biases differ.

6 Conclusions

The results to date show that this line of research has much promise. RAPTURE seems to consistently outperform fully connected network approaches. It seems clear that more focus needs to be placed on techniques for modifying network architectures, especially hidden unit creation. The UPSTART algorithm seems to be one approach of merit. In domains studied to date, hidden node creation has not been a factor, as they can be characterized successfully with one-layer networks. We hope to demonstrate UPSTART's full potential through a domain requiring new hidden nodes.

One of the major benefits of the RAPTURE approach is its ability to completely represent a symbolic rule base, while at the same time be viewed as a connectionist system. There is no need for any translation or re-translation as each representation is equivalent to the other.

References

Buchanan, G., and E.H. Shortliffe, e., editors (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.

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

Fu, L. M. (1993). Knowledge-based connectionism for revising domain theories. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(1):173–182.

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.

Mahoney, J. J., and Mooney, R. J. (1993). Combining connectionist and symbolic learning to refine certainty-factor rule-bases. *Connection Science*, 5(3-4):339–364.

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.

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

Opitz, D. W., and Shavlik, J. W. (1993). Heuristically expanding knowledge-based neural networks. In *Proceedings of the Thirteenth International Joint Conference on Artificial intelligence*, 512–517. Chamberry, France.

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.

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

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

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.

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