Appears in *Proceedings of the Eleventh International Workshop on Machine Learning*
pp. 173-180, Rutgers, NJ, July, 1994

Comparing Methods for Refining Certainty-Factor Rule-Bases

J. Jeffrey Mahoney and Raymond J. Mooney
Department of Computer Sciences
The University of Texas
Austin, TX 78712
mahoney@cs.utexas.edu, mooney@cs.utexas.edu

Abstract

This paper compares two methods for refining uncertain knowledge bases using propositional certainty-factor rules. The first method, implemented in the RAPTURE system, employs neural-network training to refine the certainties of existing rules but uses a symbolic technique to add new rules. The second method, based on the one used in the KBANN system, initially adds a complete set of potential new rules with very low certainty and allows neural-network training to filter and adjust these rules. Experimental results indicate that the former method results in significantly faster training and produces much simpler refined rule bases with slightly greater accuracy.

1 INTRODUCTION

Developing systems for automatically refining incorrect and incomplete knowledge bases is the focus of a growing amount of machine learning research, e.g. (Ourston and Mooney, 1990; Towell et al., 1990; Wogulis and Pazzani, 1993; Koppel et al., 1994). However, most of this research focuses on revising logical, Horn-clause, domain theories. This paper, by contrast, focuses on methods for refining uncertain knowledge bases employing rules with certainty factors (Buchanan and Shortliffe, 1984). Since many applications require uncertain reasoning, developing refinement methods for such knowledge bases is an important extension.

RAPTURE is a theory refinement system that combines symbolic and neural-network methods to revise a propositional certainty-factor rule base (Mahoney and Mooney, 1993). It combines the ability of neural-network methods to effectively adjust numerical parameters with the ability of symbolic methods to make concise structural changes. Specifically, it first uses a modified version of backpropagation (Rumelhart et al., 1986) to adjust the certainty factors of existing rules in order to improve classification performance on a set of training examples. During this process, rules whose certainty falls below a certain threshold are deleted. If simply adjusting the certainty of existing rules is insufficient to correctly classify all of the training examples, RAPTURE uses a version of ID3's information gain heuristic (Quinlan, 1986) to determine new features that best discriminate the misclassified examples and includes them in new rules which are then added to the knowledge base. Backpropagation is then used again to adjust the certainties of these new rules. Backpropagation and rule addition/deletion continue in a cycle until all of the training examples are correctly classified or a local maximum is reached. Using this approach, RAPTURE has successfully revised several real-world knowledge bases.

However, a purely neural-network approach to revising certainty-factor knowledge bases is also possible by adopting the approach used in KBANN (Towell et al., 1990; Towell and Shavlik, 1993). KBANN revises logical theories by mapping the rules into an equivalent neural network, revising the weights of the network using backpropagation, and optionally re-extracting symbolic rules from the revised network. KBANN obtains the effect of adding new rules to the theory by including an additional set of low-weighted links in the initial network. Specifically, all of the available features are added to the input layer, and low-weighted links are added to fully connect each layer of the network to the layer immediately above. During backpropagation, the weights on these extra links may be adjusted to improve performance on the training data. During retranslation, the additional links recruited by backpropagation are translated into new rules or new features on existing rules. Although KBANN uses logical theories and linear-threshold networks, the same approach can be used to refine certainty-factor rule bases by adding analogous low-certainty rules and allowing modified backpropagation to adjust their certainties. This approach will be referred to as RAPTURE–KBANN.

We chose not to take this approach in RAPTURE since we believed that simply adding a large number of additional low-weighted rules would greatly increase training time and potentially result in an overly-complex rule-base with decreased accuracy. Consequently, RAPTURE only adds new rules when they are required to correctly classify the training data, and then uses symbolic methods to add a much smaller number of new rules whose features best discriminate the problematic cases. In this paper, we present experiments directly comparing RAPTURE and RAPTURE–KBANN in order to demonstrate the advantages of employing symbolic methods to add new rules, compared to using neural-network methods to select from a large number of initially added rules.

The remainder of the paper is organized as follows. Section 2 presents background information on certainty factors and KBANN. Section 3 presents a summary of RAPTURE and RAPTURE–KBANN. Section 4 presents experimental comparisons on revising two real-world knowledge bases, DNA promoter recognition and a version of the MYCIN rule base. Section 5 discusses related work, Section 6 discusses future work, and Section 7 presents our conclusions.

2 BACKGROUND

2.1 CERTAINTY FACTORS

RAPTURE uses propositional certainty-factor rules to represent knowledge. These rules have the following form: $A \stackrel{0.8}{\rightarrow} D$, representing that belief in proposition A gives a 0.8 measure of belief in proposition D. Certainty factors can range in value from -1 to $+1$, representing how much the rule increases or decreases belief in the consequent. A certainty factor of $+1$ represents absolute certainty (true), whereas one of -1 represents total disbelief (false).

Rules combine evidence via *probabilistic sum*, defined for positive evidence as $a \oplus b \equiv a + b - ab$. Assume we also have the rule: $B \stackrel{0.5}{\rightarrow} D$. Given that A and B are true, our measure of belief in D becomes 0.9 ($= 0.8 + 0.5 - 0.8 \times 0.5$). Negative certainty factors combine using $a \oplus b \equiv a + b + ab$. All positive evidence is combined to determine the *measure of belief* (MB) of a proposition and all negative evidence is combined to obtain a *measure of disbelief* (MD). The certainty factor is then calculated using $CF = (MB + MD)/\min(MB, MD)$.

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

2.2 KBANN

As described in the introduction, KBANN is a theory refinement system which translates a rule base into a neural network and then refines it using backpropagation. First, a logical circuit is created using the AND-OR graph of the theory, and the weights of the units in the network are set to simulate AND and OR gates. In addition, all of the features in the data are added to the input layer. The network is then fully connected by adding low-weighted links from every node in layer n to every node in layer $n + 1$.

Once built, the network is trained using backpropagation, and links whose weights fall below a given threshold are deleted. To help minimize the size of the network, weight-decay (Hinton and Sejnowski, 1986) is utilized. By adjusting each weight in the network slightly towards zero after each weight update, links that are not contributing to the network are eliminated.

After training, symbolic rules can be extracted from the network. By analyzing the weights of the incoming links, each unit is translated into a set of *M-of-N rules*. Such rules are satisfied if at least M of their N antecedents are true. The resulting rulebase is generally much simpler than the revised network; however, there is no guarantee that the two representations are identical.

3 RAPTURE

3.1 THE RAPTURE ALGORITHM

This section summarizes the refinement algorithm used by RAPTURE which is outlined in Figure 1. Further details can be found in (Mahoney and Mooney, 1993). After acquiring a probabilistic rule-base from an expert, the rules are mapped into an equivalent network. The certainty factors of the rules are mapped to 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, the total input is the probabilistic sum of the incoming activations. No thresholding output function is needed since the probabilistic sum already provides the necessary non-linearity.

Once built, the network is refined to correctly classify a set of training examples. Network training is performed in two phases. In the first phase, a modified version of backpropagation is used to adjust the certainty factors on existing rules. The normal backpropagation equations are altered in order to perform gradient descent for certainty-factor combination functions such as probabilistic sum, MIN, and MAX. This process is called CFBP (certainty-factor backpropagation). After training with CFBP, nodes and links whose weights have dropped below 0.01 are removed.

CFBP alone, however, may fail to train the network to 100% accuracy. This is an indication that the network architecture needs modification. Two techniques are used to add new rules. First, new input features are added to the network. Specifically, ID3's information-gain metric is used to find features that will most improve classification accuracy. For each output category with mistaken examples, information gain is used to find the feature that best distinguishes between 1) those examples that belong to this output category, but are being misclassified into other "bad" categories (e.g. false negatives), and 2) all positive examples (true positives) of the bad categories of set 1. All features are converted to binary, resulting in features such as `COLOR = RED (yes/no)` or `AGE < 40 (yes/no)`. The chosen new feature is inserted into the network as positive evidence for the correct category, and as negative evidence for the bad categories. This has the desired effect of making correct classification of the originally mis-classified examples more likely. All new links are directly connected to the corresponding output node with small weights (0.1).

Once new links are in place (one for every output category that has mis-classified examples), the network is once again trained via CFBP. This cycle of CFBP followed by feature addition continues until either 100% training accuracy is achieved, or there are no gains in either classification accuracy or network error. In the latter case, the algorithm resorts to a modified version of the UPSTART algorithm (Frean, 1990), a neural-network technique for adding new hidden units. Beneath every output unit that contains incorrectly classified examples, two new hidden units are built. One of these units is designed to learn the false negative examples of the output unit, and the other is designed to learn the false positives. The former is connected to the output unit with positive weight, and the latter with negative weight. Assuming the new units can be successfully trained, all of the examples will be correctly classified. These new units are recursively trained using RAPTURE.

REPEAT

1. Perform CFBP on the network. Use given training examples, and as many epochs as necessary until classification accuracy and network mean-squared error cease to improve.
2. When $<$ 100% training accuracy, use ID3 information gain to add new input units. Make one new rule for each output unit misclassifying positive examples.

UNTIL classification accuracy and network mean-squared error cease to improve.
IF $<$ 100% training accuracy, begin UPSTART.

1. For each output unit with mistaken examples, build two new hidden units.
2. Use RAPTURE to recursively train one of these units to learn the false negatives of the output unit. This unit connects with positive weight to the output unit.
3. Use RAPTURE to recursively train the other hidden unit to learn the false positives for the output unit. This unit connects with negative weight to the output unit.

Figure 1: Overview of the RAPTURE Algorithm

Once the system has converged on the training data, the revised rules can be read directly off the network. In RAPTURE, the direct correspondence between weighted links and probabilistic rules removes any distinction between the symbolic and connectionist representations.

3.2 RAPTURE–KBANN

RAPTURE–KBANN is essentially KBANN using certainty-factor networks. Given an initial certainty-factor rule base, a certainty-factor network is built as described above. Before training this network, all of the input features are added with low weighted links (0.1) as in KBANN. In order to fully connect the network, each node must belong to a unique layer of the network. This is done by assigning to every node the layer that is equal to the length of the shortest path from this node to the input layer. The network is then completed by fully connecting—inserting links (where not already present) between each node of layer n with each node of layer $n+1$.

Once built, CFBP can be used to train the network to (hopefully) 100% training accuracy, or to a local error-minimum. A weight-decay parameter of 0.05 is included during training to help eliminate inessential links. There is no need to include RAPTURE's feature addition step since all possible features are already included in the network. However, as in RAPTURE, nodes and links are eliminated from the network whenever their weights drop below 0.01.

4 EXPERIMENTAL RESULTS

4.1 DNA PROMOTER-RECOGNITION 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 consists of 106 examples, 53 positive and 53 negative. 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. These had to be converted to certainty factor rules for use by RAPTURE. The theory was modified by breaking rules with multiple antecedents into multiple rules, one for each antecedent. This was done in order to allow each antecedent the ability to contribute its own evidence towards belief in the consequent. Certainty-factors on the individual rules were set so that if all of the antecedents of the original Horn clause were true, the result is a total certainty of 0.9 for the consequent.

RAPTURE and RAPTURE–KBANN were run on this dataset and the results compared with KBANN's results from Towell (1991). Standard training and test runs were performed, resulting in the learning curves shown in Figure 2. For the two RAPTURE systems, this graph is a plot of average classification accuracy over 25 independent trials. A single trial consists of providing each system with an increasing number of training examples and then testing on the same disjoint test set. The KBANN results were run independently on the same dataset, though using different training and test splits. The results are roughly comparable, although RAPTURE is clearly outperforming the other systems. Comparing RAPTURE with RAPTURE–KBANN, a t-test shows significant differences at 60 and 90 training examples. We do not have the necessary data to run t-tests with KBANN, although it appears RAPTURE is maintaining a slight advantage.

Results on training time are shown in Figure 3. While

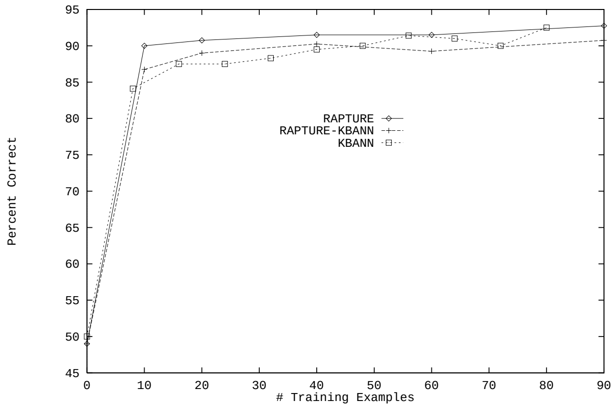

Figure 2: PROMOTER Test Accuracy

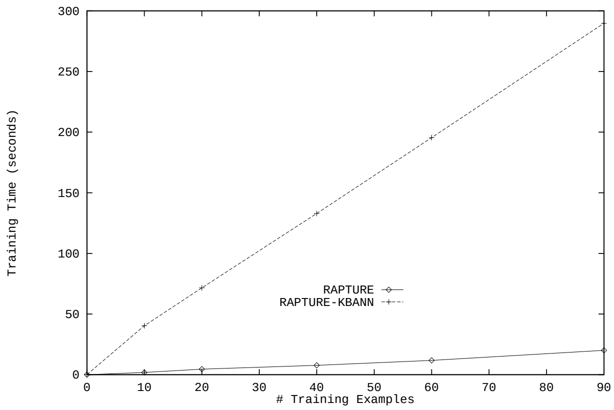

Figure 3: PROMOTER Training Time

RAPTURE trained on 90 examples in an average of 20 seconds, RAPTURE–KBANN took nearly 5 minutes, or almost 15 times as long. This is a clear indication that the addition of all of the extra nodes and links in the network causes a significant computational increase without improved generalization.

Comparisons were also made with regard to the complexity of the resulting rule-bases. Figure 4 is a plot of the number of symbols in the rule-bases that resulted from training. It is clear from the graphs that the addition of all possible features greatly increases the network size. We have also included graphs of RAPTURE–KBANN with and without weight decay for comparison. Although weight-decay helps, the resulting rule base is still overly complex.

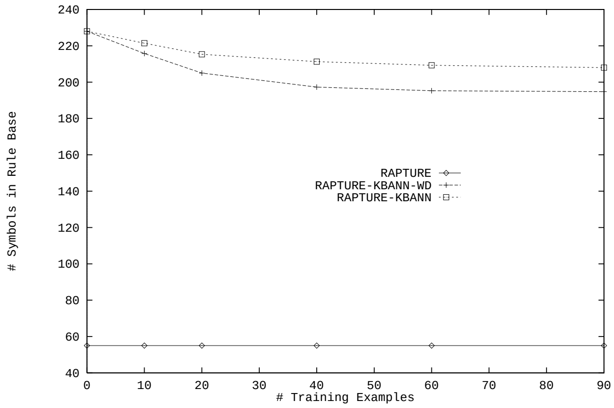

Figure 4: PROMOTER Rule-Base Complexity

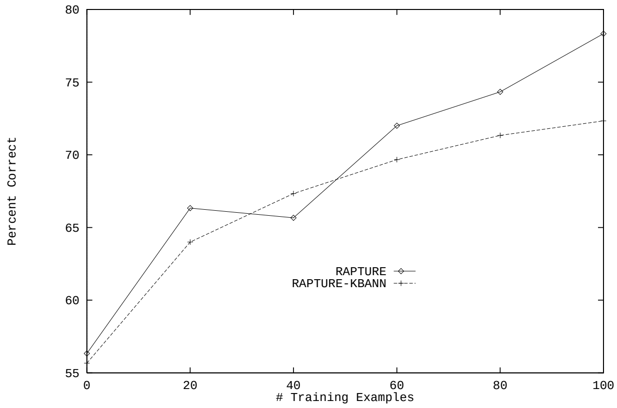

Figure 5: MYCIN Testing Accuracy

4.2 MYCIN RESULTS

Experiments were also run on a version of the Mycin knowledge-base (Buchanan and 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 headache duration. Many features for each patient are 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 in a format acceptable for Rapture (certainty factors). Included are 137 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 5 and are averaged over 20 trials. To date, Kbann has not been run on this dataset, and our comparisons here are between Rapture and Rapture–Kbann. The results are again not too far apart, though Rapture does gain an advantage after it has seen enough examples. T-tests confirm that from 60 examples onward, Rapture is performing significantly better.

Results for training time are shown in Figure 6. For 100 examples, Rapture averages just over 31 minutes compared to Rapture–Kbann which took nearly 4 times as long, or slightly more than 2 hours. This is a clear indication that with a domain of this complexity (264 features), including every possible feature in the network and fully connecting becomes computationally expensive.

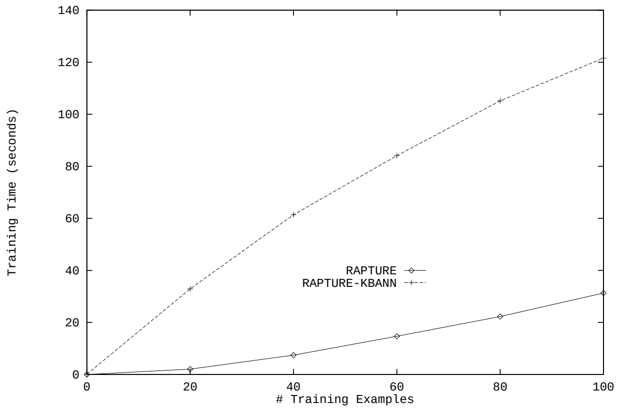

Figure 6: MYCIN Training Time

The resulting rule-base complexities are presented in Figure 7. It is again apparent that with the addition of every feature in the domain, a much larger rule base results and that weight decay gives only a marginal improvement. Weight decay did, however, provide slight improvement in training time. This was not plotted as the improvements were minimal. Further, weight decay had no effect upon generalization accuracy.

5 RELATED WORK

Although most research in theory refinement has focussed on revising logical theories, there have been several other projects on revising uncertain knowledge bases. This section reviews these and other related projects.

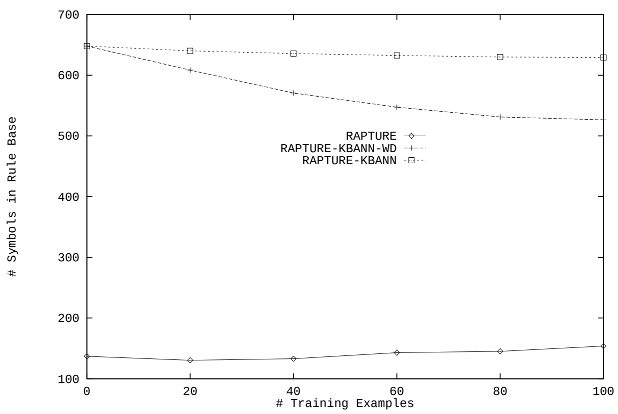

Figure 7: MYCIN Rule-Base Complexity

SEEK2 (Ginsberg et al., 1988) revises rule bases containing *M-of-N* rules, also known as *choice-component* rules. It 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.

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 used in our experiments, increasing accuracy 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.

Fu (Fu, 1989) and Lacher (Lacher, 1992) have also used backpropagation techniques to revise certainty factors. 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 is not a problem since although these functions are not *everywhere* differentiable, they are trivially so almost everywhere. The problem of having two non-zero activation levels that are exactly the same and that are the minimum values 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.

Schwalb (Schwalb, 1993) has shown how the parameters of Bayesian networks can also be refined by mapping them into neural networks and performing backpropagation. However, his method creates a neural network whose size is exponential in the fan-in of the Bayesian network, does not address the issue of adding new features or hidden units, and was not tested on revising actual knowledge bases.

PTR (Feldman, 1993) revises a theory expressed as a collection of Horn-clause rules including numerical parameters representing the expert's confidence in the accuracy of the rule. Unlike certainty factors in RAPTURE, these values do not represent the strength, or amount of evidence suggested by the rule, but rather the user's confidence that the rule is correct.

TOPGEN (Opitz and Shavlik, 1993) is a method for adding new hidden units to a KBANN-network. By keeping track of of the false negative and false positives for which each node is responsible, the algorithm locates areas of the network requiring additional units. However, when TOPGEN adds a new hidden unit, it adopts the KBANN approach of fully connecting it to the input layer. By contrast, RAPTURE uses information gain to add new input features only as needed.

There has been a number of methods for growing neural-network architectures 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). However, these methods also employ full-connectivity to all input features.

6 FUTURE WORK

We are currently in the process of comparing RAPTURE, RAPTURE–KBANN, and other inductive learning and theory refinement algorithms on additional tasks such as soybean disease diagnosis (Michalski and Chilausky, 1980) and recognizing DNA splice-junctions (Noordewier et al., 1991) in order to determine RAPTURE's relative performance in these domains.

Unfortunately, in the existing experiments, hidden-unit addition has not proven particularly useful since RAPTURE is generally able to reach convergence using only backpropagation and feature addition. We are also currently exploring criteria for adding hidden units earlier in the training process in order to reduce the number of new input features that need to be added. The early results on this approach are encouraging.

We are also planning experiments directly comparing CFBP and backpropagation for purely inductive learning. The standard neural-network method of training a fixed network initialized with random weights is also easily applied to certainty-factor networks. We hope to discover the relative advantages and disadvantages of probabilistic sum as a combination function compared to the normal linear-threshold.

Although they have proven quite useful in practice, certainty factors have frequently been criticized as *ad hoc* and restrictive (Shafer and J. Pearl, 1990). Actually, certainty factors have been shown to have a clear probabilistic semantics, but only under very restrictive independence assumptions (Heckerman, 1986). Nevertheless, 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). Although Schwalb's approach to revising Bayesian networks is intractable in the general case (Schwalb, 1993), it may be useful for networks with limited fan-in; and perhaps similar, more efficient, heuristic methods could be developed for more complex networks. In addition, techniques for inducing Bayesian networks from data (Cooper and Herskovits, 1992) could potentially be used to refine the underlying causal structure as well. Finally, there has also been some recent work on combining symbolic and neural-network methods to revise fuzzy-logic controllers (Berenji, 1990).

7 CONCLUSIONS

This paper has demonstrated some advantages to combining symbolic and neural-network methods for refining uncertain knowledge bases. Specifically, a symbolic method for adding new rules to a certainty-factor knowledge base was compared to a neural-network method based on KBANN. Instead of making every feature immediately available to the network, we have shown that by judiciously selecting key features as necessary, training time can be greatly reduced and a simpler and slightly more accurate rule base can be created. This is particularly true when there are a large number of input features that can be recruited by backpropagation but are not strictly necessary for correct classification.

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 neural-network learning methods to "symbolic" knowledge bases and employing symbolic methods to modify "neural" networks. Hopefully these results will encourage others to explore similar opportunities for cross-fertilization of ideas between neural and symbolic learning.

Acknowledgements

This research was supported by the National Science Foundation under grant IRI-9102926 and the Texas Advanced Research Program under grant 003658114. We wish to thank M. Noordewier, G.G. Towell, and J.W. Shavlik for supplying the the KBANN results and promoter theory and data, 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 Shortliffe, E., editors (1984). *Rule-Based Expert Systems:The MYCIN Experiments of the Stanford Heuristic Programming Project.* Reading, MA: Addison-Wesley Publishing Co.

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.

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., and Sejnowski, T. (1986). Learning and relearning in boltzmann machines. In Rumelhart, D. E., and McClelland, J. L., editors, *Parallel Distributed Processing, Vol. I*, 282–317. Cambridge, MA: MIT Press.

Koppel, M., Feldman, R., and Segre, A. M. (1994). Bias-driven revision of logical domain theories. *Journal of Artificial Intelligence Research*, 1:1–50.

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.

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.

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.

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.

Noordewier, M. O., Towell, G. G., and Shavlik, J. W. (1991). Training knowledge-based neural networks to recognize genes in DNA sequences. In *Advances in Neural Information Processing Systems*, vol. 3. San Mateo, CA: Morgan Kaufman.

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.

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

Quinlan, J. R. (1986). 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.

Schwalb, E. (1993). Compiling Bayesian networks into neural networks. In *Proceedings of the Tenth International Conference on Machine Learning*, 291–297. Amherst, MA.

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.

Towell, G., and Shavlik, J. (1993). Extracting refined rules from knowledge-based neural networks. *Machine Learning*, 13(1):71–102.

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.

Wogulis, J., and Pazzani, M. (1993). A methodology for evaluating theory revision systems: Results with Audrey II. In *Proceedings of the Thirteenth International Joint Conference on Artificial intelligence*, 1128–1134. Chambery, France.

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