Appears in *Machine Learning: Proceedings of the Tenth International Conference*,
P. E. Utgoff (editor), Morgan Kaufmann, San Mateo, CA, 1993

Learning Symbolic Rules Using Artificial Neural Networks

Mark W. Craven and Jude W. Shavlik
Computer Sciences Department
University of Wisconsin
1210 West Dayton St.
Madison, WI 53706
email: {craven, shavlik}@cs.wisc.edu

Abstract

A distinct advantage of symbolic learning algorithms over artificial neural networks is that typically the concept representations they form are more easily understood by humans. One approach to understanding the representations formed by neural networks is to extract symbolic rules from trained networks. In this paper we describe and investigate an approach for extracting rules from networks that uses (1) the NofM extraction algorithm, and (2) the network training method of soft weight-sharing. Previously, the NofM algorithm had been successfully applied only to *knowledge-based* neural networks. Our experiments demonstrate that our extracted rules generalize better than rules learned using the C4.5 system. In addition to being accurate, our extracted rules are also reasonably comprehensible.

1 INTRODUCTION

Artificial neural networks (ANNs) have been successfully applied to real-world problems as varied as steering a motor vehicle (Pomerleau, 1991) and learning to pronounce English text (Sejnowski & Rosenberg, 1987). In addition to these practical successes, several empirical studies have concluded that neural networks provide performance comparable to, and in some cases, better than common symbolic learning algorithms (Atlas et al., 1989; Fisher & McKusick, 1989; Mooney et al., 1989). A distinct advantage of symbolic learning algorithms, however, is that the concept representations they form are usually more easily understood by humans than the representations formed by neural networks. In this paper we describe and investigate an approach for extracting symbolic rules from trained neural networks. Our approach uses the NofM algorithm (Towell & Shavlik, 1991) to extract rules from networks that have been trained using Nowlan and Hinton's (1992) method of *soft weight-sharing*. Although soft weight-sharing was designed as a technique for improving generalization in neural networks, we explore it here as a means for facilitating rule extraction. We present experiments that demonstrate, for two difficult learning tasks, our method learns rules that are more accurate than rules induced by Quinlan's (1993) C4.5 system. Furthermore, the rules that are extracted from our trained networks are comparable to rules induced by C4.5 in terms of complexity and understandability.

Towell and Shavlik (1991) demonstrated that concise and accurate symbolic rules can be extracted in the restricted case of *knowledge-based* neural networks. In a knowledge-based network, the topology and initial weights of the network are specified by a domain theory consisting of symbolic inference rules. Since these networks initially encode symbolic rules, training is more a process of rule refinement than of *tabula rasa* learning. This paper describes work that involves using Towell and Shavlik's NofM algorithm to extract rules from ANNs which have *not* been initialized by a domain theory. Because the NofM algorithm assumes that the weights in a trained network are clustered, we modify the training process to encourage such a network state after training. Previously, Towell (1991) reported that the NofM algorithm failed to extract accurate rules from conventional networks.

We use two problem domains to investigate the effectiveness of our approach. The first domain involves recognizing *promoters* in DNA (Towell et al., 1990). Promoters are short nucleotide sequences that occur before genes and serve as binding sites for the protein *RNA polymerase* during gene transcription. Identifying promoters is an important step in locating genes in DNA sequences. The second problem domain that we investigate is a simplified version of the NETTALK task of mapping English text to its pronunciation (Sejnowski & Rosenberg, 1987). Our scaled-down version of this domain involves learning only the stresses (but not the phonemes) from a corpus of the 1000 most common English words.

2 EXTRACTING RULES FROM NEURAL NETWORKS

An important criterion by which a machine learning algorithm should be judged is the comprehensibility of the representations formed by the algorithm. That is, does the algorithm encode the information it learns in such a way that it may be inspected and understood by humans? There are at least five reasons why this is an important criterion.

- *Validation.* If the designers and end-users of a learning system are to be confident in the performance of the system, then they must understand how it arrives at its decisions.
- *Discovery.* Learning algorithms may discover salient features in the input data whose importance was not previously recognized. If the representations formed by the learner are comprehensible, then these discoveries can be made accessible to human review.
- *Explanation.* If the representations are understandable, then an explanation of the classification made on a particular case can be garnered.
- *Improving generalization.* The feature representation used for an inductive learning task can have a significant impact on generalization performance. Understanding learned concept representations may facilitate the design of a better feature representation for a given problem.
- *Refinement.* Some researchers use inductive learning systems to refine approximately-correct domain theories (Ourston & Mooney, 1990; Pazzani & Kibler, 1992; Towell et al., 1990). When a learning system is used in this way, it is important to understand the changes to the knowledge base that have been imparted during the training process.

2.1 RULE EXTRACTION METHODS

A significant limitation of artificial neural networks is that the concepts they learn are usually impenetrable to human understanding because concepts are represented by a large number of real-valued parameters: the weights and biases of the network. One approach toward understanding the representations formed by a neural network is to extract symbolic rules from the network (Fu, 1991; McMillan et al., 1991; Saito & Nakano, 1988).

The underlying premise of these rule-extraction methods is that each hidden and output unit in the network can be thought of as implementing a symbolic rule. The concept associated with each unit is the consequent of the rule, and certain subsets of the units that feed into this unit represent the antecedents of the rule. As shown in Figure 1, the process of rule extraction involves finding the *sufficient* conditions for each consequent. In order to find such sets of sufficient conditions, rule-extraction methods assume that, after training, hidden and output units tend to be either maximally active (i.e., have activation near one), or inactive (i.e., have activation near zero). Given this assumption, a rule-extraction algorithm can search for minimal sets of antecedent units that, when maximally active, cause the consequent unit to become maximally active. The process of searching for rules is problematic because of the combinatorics involved. The complexity of this search is $O(2^n)$ where n is the number of connections impinging on the consequent unit. Moreover, these algorithms tend to extract a large number of rules, even for networks of moderate complexity.

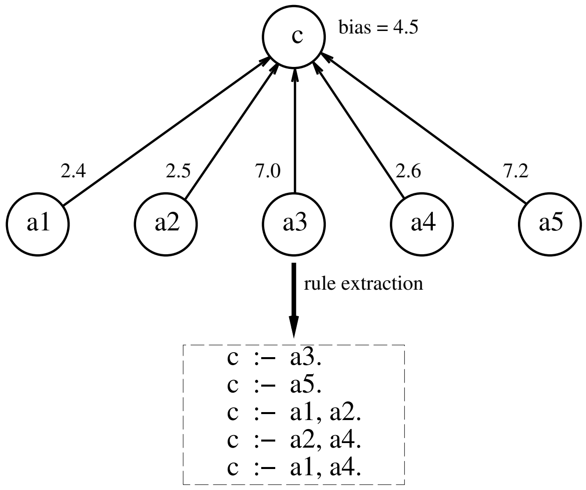

Figure 1: **Extracting Rules from a Unit in a Neural Network.** The extracted rules show the combinations of antecedent units which must be active for the consequent unit's bias to be exceeded.

2.2 THE NofM ALGORITHM

Towell and Shavlik previously described an algorithm, called NofM, that avoids the combinatoric and rule-set size problems of other rule-extraction algorithms by clustering weights into equivalence classes. They have demonstrated that their NofM algorithm is able to extract accurate and concise rules from trained knowledge-based neural networks; that is, networks for which the topology and initial weights have been specified by an approximately-correct domain theory. The algorithm is called NofM because it explicitly searches for rules of the form:

If (N of the M antecedents are true) then ...

The NofM algorithm comprises six steps:

1. **Clustering.** The weights impinging on each hidden and output unit of the trained network are

grouped into clusters. Initially, each weight is treated as a cluster. The two nearest clusters are successively merged until no pair of clusters is closer than a preselected distance. The distance metric used for clustering is the difference in the means of the weight magnitudes for two clusters. Additionally, weights with small magnitudes are pruned from the network at this step.

2. **Averaging.** The magnitude of each weight is set to the average value of the weights in its cluster.
3. **Eliminating.** Weight clusters that are not needed in order to correctly activate a unit are eliminated. Two elimination procedures are applied: one algorithmic and one heuristic. The algorithmic elimination procedure identifies clusters that *cannot* have an effect on whether or not a unit's bias is exceeded. The heuristic elimination step eliminates clusters that do not have such an effect for any of the training examples.
4. **Optimizing.** The unit biases are retrained to adapt the network to the changes that been imparted by the previous steps.
5. **Extracting.** Each hidden and output unit is translated into a rule with weighted antecedents such that the consequent is true if the sum of the weighted antecedents exceeds the bias.
6. **Simplifying.** Weights and thresholds are eliminated and rules are expressed in the NofM format.

Figure 2 illustrates the application of the NofM to the unit shown in Figure 1. The weights have been grouped into two clusters, and each weight has been set to the average value of its cluster. One of the extracted rules is expressed in the NofM format; the other two rules are trivial NofM cases (1 of 1). The *eliminating* and *optimizing* steps are not depicted in this example.

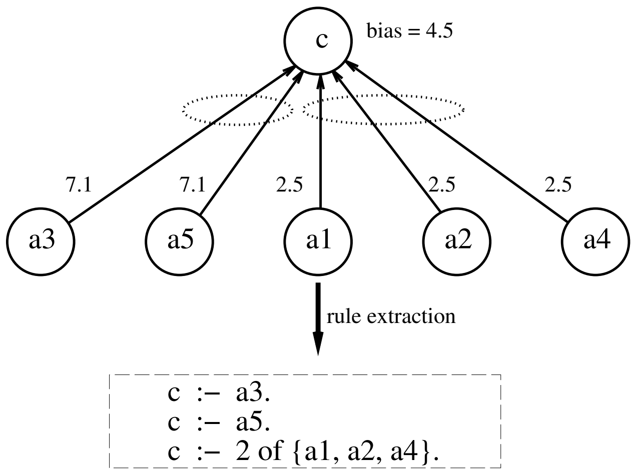

Figure 2: **Extracting Rules using the NofM Method.** The dotted ovals illustrate how the weights have been grouped into clusters. Each weight has been set to the average value of its cluster.

3 EXTENDING NofM WITH SOFT WEIGHT-SHARING

An underlying assumption of the NofM method is that the distribution of weights in the network will be conducive to forming a small number of clusters for each hidden and output unit. For knowledge-based neural networks, this is a reasonable assumption since the weights are clustered before training. For example, using the KBANN algorithm (Towell et al., 1990) to map a set of symbolic rules into a knowledge-based network, the weights that are specified by the domain theory have values of approximately 4 and -4, whereas the rest of the weights have values near 0. Experimental evidence indicates that the weights tend to be fairly well clustered after training as well (Towell, 1991).

The applicability of the NofM method might seem to be limited to knowledge-based networks, since in conventional neural networks there is usually not a bias that leads weight values to be clustered after training. In fact, Towell (1991) reported that NofM did not extract small sets of accurate rules from conventional networks. However, the approach that we explore in this paper does not rely on the network weights being initially clustered, but instead encourages clustering *during* network training. We use a method developed by Nowlan and Hinton (1992), termed *soft weight-sharing*, that encourages weights to form clusters during the training process. Although their method was motivated by the desire for better generalization, we explore it here as a means for facilitating rule extraction.

In the spirit of the minimum-description-length principle (Rissanen, 1978), soft weight-sharing uses a cost function that penalizes network complexity. Thus, during training the network tries to find an optimal tradeoff between data-misfit (i.e., the error rate on the training examples) and complexity. The complexity term in soft weight-sharing models the distribution of weights in the network as a mixture of multiple Gaussians. A set of weights is considered to be simple if the weights have high probability densities under the mixture model. Specifically, the cost function in soft weight sharing is the following:

$$C = \lambda E - \sum_{i \in wgts} \log \left[\sum_{j \in Gauss} \pi_j \; p_j(w_i) \right]$$

where E is the data-misfit term, λ is a parameter used to balance the tradeoff between data misfit and complexity, w_i is a weight in the network, $p_j(w_i)$ is the density value of w_i under the jth Gaussian, and π_j is the mixing proportion of the jth Gaussian. A mixing proportion is a weight that determines the influence of a particular Gaussian. The mixing proportions are constrained to sum to 1.

Neural learning algorithms perform gradient descent to locally minimize this cost function. The partial derivative of the cost function with respect to each weight is the sum of the usual error derivative plus a term due to the complexity cost of the weight:

$$\frac{\partial C}{\partial w_i} = \lambda \frac{\partial E}{\partial w_i} - \sum_{j \in Gauss} r_j(w_i) \frac{(\mu_j - w_i)}{\sigma_j^2}$$

Here μ_j and σ_j^2 are the mean and variance, respectively, of the jth Gaussian, and $r_j(w_i)$ is the conditional probability that w_i is being modelled by the jth Gaussian:

$$r_j(w_i) = \frac{\pi_j \; p_j(w_i)}{\sum_{k \in Gauss} \pi_k \; p_k(w_i)}$$

Thus, the effect of each Gaussian is to pull each weight toward the mean of the Gaussian with a force proportional to the density of the Gaussian at the value of the weight. When weights are pulled tightly around the means of the Gaussians, the network is similar to one that has fewer free parameters than connections (i.e., ordinary weight sharing). The parameter settings of each Gaussian – the mean μ_j, standard deviation σ_j, and mixing proportion π_j – are learned simultaneously with the weights during training.

Our approach to rule extraction involves training networks using a variant of soft weight-sharing and then applying the NofM algorithm to the trained networks. Although the NofM method was designed for knowledge-based neural networks, we hypothesized that it could be successfully applied to conventional networks, provided that the weights of the networks were grouped into clusters during training.

Whereas the NofM algorithm works best when the weights impinging on *each unit* form clusters, standard soft weight-sharing tends to *globally* cluster network weights. Our implementation of soft weight-sharing hence assigns a local set of Gaussians to each unit. The complexity cost of a given weight is calculated with respect to only the Gaussians associated with the unit to which the weight connects.

4 DATA SETS

Our experiments address the hypothesis that soft weight-sharing is able to cluster the network weights during training such that NofM is able to extract a small set of accurate rules. In order to evaluate the effectiveness of our approach, we use two problem domains to compare the accuracy and succinctness of our extracted rules against rules induced by the C4.5 system. Both problem domains involve predicting a class given a fixed-length "window" onto a string of interest. In the case of the *promoter* domain, the string is a DNA sequence, and in the NETTALK domain, the string is an English word (or part of one).

The *promoter* data set comprises 468 examples,[1] half of which are positive examples (i.e., promoters). Each example has 57 features which represent the DNA sequence. A single strand of DNA is a linear chain composed from the four nucleotides represented by the letters {A, G, C, T}. Thus all of the features for this problem are nominal features that can take on the values *A*, *G*, *C*, *T*, or *unknown*. Each example is a member of one of two classes: *promoter* or *non-promoter*. The positive examples for this data set are aligned such that the gene following each promoter begins in the seventh position from the right end of the window. Thus the leftmost 50 window positions are labelled `-50` to `-1`, and the rightmost seven are labelled `1` to `7`.

For the neural networks, a local representation is used for the *promoter* features. For each feature there are four input units – one corresponding to each of the nucleotides. When the value of a feature is known, the input unit corresponding to the value is given an activation of 1, and the other three units for the feature are given activations of 0. When a feature value is unknown, all four input units are given activations of 0.25.

Our simplified NETTALK data set consists of 5438 examples taken from the 1000 most common English words. Each example has seven features which represent the letters in the input window. Each feature can take on one of 27 values. There is a value corresponding to each letter of the alphabet, and a value to represent the absence of a letter. Since each example is formed from only a single word, when the window overhangs a word, the overhanging window positions are set to the "space" value. In the original NETTALK domain, the task involved predicting both a phoneme and a stress for each window position. In our experiments we have simplified the problem so that the classifiers are trained only to predict a stress (from five disjoint classes).

5 EXPERIMENTAL RESULTS

In this section we evaluate our approach to rule extraction by comparing the accuracy and comprehensibility of (1) rules extracted from neural networks, and (2) rules learned using the C4.5 system. The comprehensibility of a set of rules is difficult to measure. We measure the syntactic complexity of the rule sets and

[1]Note that this data set is larger and more biologically complicated than the one that was used by Towell et al. (1990). The latter data set is available by anonymous ftp from the UC-Irvine Repository of Machine Learning Databases and Domain Theories (ftp.ics.uci.edu).

use this as a crude proxy for comprehensibility. Specifically, we consider the number of rules and antecedents as measures of syntactic complexity.

5.1 THE PROMOTER DATA SET

For the *promoter* problem, we use a ten-fold cross-validation methodology[2] to assess the ability of our approach to extract accurate, comprehensible rules from trained networks. Our reported results represent averaged values for the ten runs.

The neural networks used for the *promoter* domain have fully-connected hidden units in a single layer. The number of hidden units used in each network is determined by cross-validation *within* the training set. That is, for each training set, networks with 20, 15, 10, 5, and no hidden units are trained, and cross-validation is used to pick the network that is to be trained on all of the data in the training set. After the number of hidden units is selected for each network, a similar cross-validation procedure is used to determine the λ parameter for soft weight-sharing. We use a conjugate-gradient learning algorithm (Kramer & Sangiovanni-Vincentelli, 1989) to train the weights and the Gaussian parameters of the networks. Each hidden and output unit has five local Gaussians which act on the weights feeding into the unit.

Decision trees are induced, and rules extracted from them, using Quinlan's (1993) C4.5 system. Cross-validation within each training set is used to determine the confidence levels for both tree pruning and rule pruning. The confidence level selected for tree pruning does not affect the rule-extraction results since the C4.5 rule-induction program operates on unpruned trees and performs its own pruning independently. Thus, the confidence level for tree pruning affects the generalization only of the decision trees. For each training set, we test confidence levels ranging from 5% to 95% and separately select tree-pruning and rule-pruning levels.

Table 1 shows the test set error rates on the *promoter* data set for the decision trees, rules extracted from the trees, neural networks, and rules extracted from the networks. As can be seen in the table, neural networks perform significantly better on this task than decision trees or the rules extracted from them. Additionally, the performance of the symbolic rules extracted from the neural networks is fairly close to the performance of the networks themselves, and better than the rules extracted from the decision trees. The difference in error rates between the rules extracted from networks and the C4.5 rules is significant at the 0.05 level using a paired, 1-tailed t-test.

[2] In ten-fold cross-validation, the available data is partitioned into ten sets. Classifiers are trained using examples from nine of the sets and tested on examples from the tenth set. This procedure is repeated ten times so that each set is used as the testing set once.

Table 1: **Generalization on the *Promoter* Data.**

approach		% test set error
C4.5	decision trees	16.9
	extracted rules	13.5
ANNs	networks	7.9
	extracted rules	11.1

Table 2: **Rule-set Sizes for the *Promoter* Data.**

approach	# rules	# antecedents
C4.5	23.2	47.3
ANNs	8.2	119.6

Table 2 shows the average number of rules and antecedents for the rules extracted from our networks and the rules induced by C4.5. The values for antecedents indicate the *total* number in a rule set. The rules extracted by the C4.5 algorithm are purely-conjunctive rules that tend to have few antecedents. Thus, the sets extracted from the decision trees contain more rules but fewer antecedents than those extracted from networks. The additional complexity of the rules extracted from networks, however, results in a significant gain in accuracy. Moreover, the rules extracted from networks have only 14.6 antecedents per rule on average, and these antecedents often refer to localized, contiguous parts of the DNA sequence. We feel, therefore, that their complexity is within the bounds of what biologists can readily understand.

Table 3 shows a set of rules extracted from one of the *promoter* networks. In addition to the NofM-style rules, this rule set has been expressed using a new predicate we call `more_than`. The `more_than` predicate has the following form:

$$N \texttt{ more\_than}(Pos\_Set, \; Neg\_Set)$$

where N is an integer, and *Pos_Set* and *Neg_Set* are sets of positive and negated antecedents, respectively. The predicate returns true if the number of satisfied antecedents in *Pos_Set* minus the number of satisfied antecedents in *Neg_Set* is greater than N. This predicate provides a succinct way of expressing rules that have many negated antecedents. Without such a predicate, negated antecedents tend to result in a large number of mostly-redundant rules. Since the knowledge-based networks to which the NofM algorithm was previously applied had very few negated antecedents, this predicate was not previously necessary.

The rule set shown in Table 3 exhibits several interesting characteristics. First, the rules abstract away

Table 3: **Rules Extracted from a *Promoter* Network.** The predicate `more_than` returns true if the number of satisfied antecedents in the first set minus the number of satisfied antecedents in the second set is greater than the supplied threshold. The notation `@-36` indicates the starting position of a given sequence; in this case the position is 36 nucleotides before the start of a putative gene. Dashes represent placeholders, so {`@-36 '--AG----A'`} has only three antecedents. The letter `S` is an ambiguity code that biologists use to represent (`C` $\vee$ `G`).

```
promoter :-
    hidden2, not (hidden1), not (hidden4).

promoter :-
    hidden3, not (hidden1), not (hidden4).

promoter :-
    hidden2, hidden3,
    not 2 of {hidden1, hidden4}.

hidden1 :-
    5 more_than( {@-40 'C----C-G-C-G',
                  @-13 '-SG---', @-1'G'},
                 {@-40 'A---T--A----',
                  @-13 '-T---T'} ).

hidden1 :-
    not ({@-40 '-----T------'}),
    3 more_than( {@-40 'C----C-G-C-G',
                  @-13 '-SG---', @-1'G'},
                 {@-40 'A---T--A----',
                  @-13 '-T---T'} ).

hidden2 :-
    5 more_than( {@-40 'A---T-GA-A', @-13 '-T-'},
                 {@-40 'C----C-G--', @-13 '-SG'} ).

hidden2 :-
    {@-40 '-----T----'},
    3 more_than( {@-40 'A---T-GA-A', @-13 '-T-'},
                 {@-40 'C----C-G--', @-13 '-SG'} ).

hidden3 :-
    4 more_than( {@-40 'A---T-GA-AT',
                  @-13 '-T-A-T'},
                 {@-40 '-----C-G-C-', @-20 'G--G',
                  @-13 'GSGG'} ).

hidden3 :-
    {@-40 '-----T-----'},
    2 more_than( {@-40 'A---T-GA-AT',
                  @-13 '-T-A-T'},
                 {@-40 '-----C-G-C-', @-20 'G--G',
                  @-13 'GSGG'} ).

hidden4 :-
    4 of {@-44 'G', @-40 '----AC-G-C',
          @-24 'A--G', @-13 '-SG'}.
```

a significant amount of the complexity of the network from which they are extracted. There are only ten rules and a total of 106 antecedents. Six of the hidden units and more than 2400 of the weights that were present in the neural network are not represented in the rules. A second observation is that the rules focus on what are known by biologists to be the most significant regions of the DNA sequence. In particular, a domain theory developed by Michiel Noordewier (Towell et al., 1990) identifies the `-14` to `-7` and the `-37` to `-31` regions as containing the most important features of a promoter. These are termed the *contact* regions. The rules extracted from all of the hidden units specify antecedents primarily in these areas.

Although we have used the number of rules and antecedents as a basis for comparing the comprehensibility of extracted rule sets, it is important to note that this comparison does not take into account the semantic differences between the two types of rules. Whereas we have employed the N-`of`-M and `more_than` constructs in the network-extracted rules, C4.5 rules are purely conjunctive. A second difference is that the rules extracted from networks identify intermediate concepts between the input features and the output classes; these are the rules whose consequents correspond to hidden units. Although it is difficult to attach meaningful labels to the concepts represented by hidden units, we believe that in some cases, these intermediate terms might lead to rule sets that are more comprehensible than those which simply include the input and output terms. A third difference is that the network-extracted rules define only the positive (promoter) class, and employ the closed-world assumption to classify examples as negative.

5.2 THE NETTALK DATA SET

For the NETTALK domain, classifiers are trained on half of the example set and tested on the other half. Ten runs of this procedure are performed, and the reported results represent averaged values. As with the *promoter* domain, cross validation is used to determine the λ parameter and the number of hidden units for the networks, and the confidence levels for pruning C4.5 trees and rules. Five local Gaussians are used for each hidden and output unit, and a conjugate-gradient algorithm is used to learn all of the network parameters.

Since the NETTALK domain involves five classes, the rules extracted from trained networks are not necessarily mutually exclusive and exhaustive. In other words, a given input sequence may satisfy more than one of the class rules, or alternatively, it may satisfy none of the class rules. The C4.5 rule-extraction method also faces this complication when it prunes antecedents and rules from its rule set. C4.5 handles this problem in two ways: (1) rules are ordered by class, and the first rule to match a given instance determines the predicted class; (2) a default rule is used to classify in-

Table 4: **Generalization on the NETTALK Data.**

approach		% test set error
C4.5	decision trees	19.1
	rules	20.1
ANNs	networks	13.0
	rules	17.0

Table 5: **Rule-set Sizes for the NETTALK Data.**

approach	# rules	# antecedents
C4.5	233.5	466.5
ANNs	17.5	661.9

stances that do not satisfy any of the other rules. We employ the same policy in classifying instances using our network-extracted rules. The rules are ordered to minimize false-positive errors on the training set, and the default rule predicts the class which has the most training examples not covered by any rule.

Table 4 shows the test set error rates on the NETTALK data set for the decision trees, rules extracted from the trees, neural networks, and rules extracted from the networks. The results in this table indicate that the neural networks and the rules extracted from them outperform C4.5 decision trees and rules. The difference in error rates between the rules extracted from networks and the decision trees is significant at the 0.005 level using a paired, 1-tailed t-test.

Table 5 shows the average number of rules and antecedents in the extracted rule sets. The rule sets extracted from the neural networks contain far fewer rules than the rules generated from the decision trees, although the network rules have far more antecedents. This result points out an interesting tradeoff regarding comprehensibility: is it easier to understand a large body of simple rules or a small collection of complex rules?

5.3 THE EFFECT OF SOFT WEIGHT-SHARING

In order to evaluate the contribution of soft weight-sharing to the results presented in this section, we trained *promoter* and NETTALK networks using the same methodology as the previous experiments, except that soft weight-sharing was not employed. Surprisingly, the results we obtained for the *promoter* domain, in terms of rule generalization and comprehensibility, were essentially the same. The results for the NETTALK domain are presented in Table 6. Although soft weight-sharing does not seem to affect the generalization ability of the extracted rules, it has a significant impact on the concision of the rules. The rules extracted from the networks that employed soft weight-sharing had, on average, 108 fewer antecedents. This difference is significant at the 0.05 level using a paired, 1-tailed t-test.

Table 6: **The Effect of Soft Weight-Sharing (SWS) for the NETTALK Domain.**

approach	% error	# rules	# antes
ANNs w/ SWS	17.0	17.5	616.1
ordinary ANNs	16.9	16.0	724.5

In our experiments, we have evaluated our rule-extraction approach along two dimensions: generalization and comprehensibility. For neural networks, it is clearly more difficult to control the performance of a rule-extraction method along the dimension of comprehensibility. A rule set that generalizes as well as the network from which it was extracted can be obtained by simply replicating the network as a large set of rules. The results presented in Table 6 indicate that soft weight-sharing can act to significantly improve the performance of the NOFM algorithm along this dimension – comprehensibility – which is the most formidable dimension.

We are currently conducting experiments to evaluate the effectiveness of soft weight-sharing as we vary such parameters as network size, training-set size, number of Gaussians, output representation, etc.

6 CONCLUSIONS

We have demonstrated that small sets of accurate, reasonably concise symbolic rules can be extracted from ordinary artificial neural networks. Our approach to this problem involves exploiting the effectiveness of the NOFM algorithm by encouraging weight clustering during training. For two difficult problem domains, recognizing promoters in DNA, and mapping English text to stress patterns, our approach was able to induce rules that resulted in better generalization than rules learned using C4.5.

There are a number of issues regarding our approach that we plan to pursue in further research. One such issue is adapting the method so that it can extract concise rule sets from networks that have learned distributed representations. In a distributed representation, each concept at the hidden-unit level may be encoded by the activations of many hidden units, and each unit may play a part in representing many different concepts. Distributed representations tend to result in rule sets that are verbose and difficult to understand. The NOFM algorithm makes the assumption that each hidden unit corresponds to a meaningful concept, and thus it searches for rules by considering each hidden unit independently. Our proposed approach involves partitioning the space of hidden unit activations and then searching for rules that explain particular regions of this space.

A second area that we plan to investigate in future research is to employ a weight-pruning method, such as *Optimal Brain Damage* (LeCun et al., 1990), during learning. The effectiveness of the NofM algorithm is partly due to the weight pruning that it performs during its clustering step. The expected advantage of pruning weights *during* the learning process is that the remaining weights are able to adapt to the changes imparted by the pruning operation.

A third issue to be investigated is rule-pruning strategies. The C4.5 system, for example, incorporates clever techniques for pruning antecedents and for dropping entire rules. These methods improve both the generalization ability and the comprehensibility of the extracted rules. We plan to investigate the use of these methods in our system.

Extracting accurate, comprehensible rules from neural networks is an important problem in machine learning. We have described an approach that employs the NofM algorithm and soft weight-sharing, and demonstrated that it is able to extract accurate, comprehensible rules from networks trained on a difficult real-world problem. These promising results indicate that the problem of understanding representations learned by artificial neural networks may be tractable.

Acknowledgements

This work was partially supported by Department of Energy Grant DE-FG02-91ER61129 and National Science Foundation Grant IRI-9002413. Rich Maclin provided helpful comments on an earlier version of this paper.

References

Atlas, L., Cole, R., Connor, J., El-Sharkawi, M., Marks II, R. J., Muthusamy, Y., & Barnard, E. (1989). Performance comparisons between backpropagation networks and classification trees on three real-world applications. In Touretzky, D., editor, *Advances in Neural Information Processing Systems (volume 2)*. Morgan Kaufmann, San Mateo, CA.

Fisher, D. H. & McKusick, K. B. (1989). An empirical comparison of ID3 and back-propagation. In *Proc. of the 11th IJCAI*, (pp. 788–793), Detroit, MI.

Fu, L. M. (1991). Rule learning by searching on adapted nets. In *Proc. of the 9th Nat. Conf. on Artificial Intelligence*, (pp. 590–595), Anaheim, CA.

Kramer, A. H. & Sangiovanni-Vincentelli, A. (1989). Efficient parallel learning algorithms for neural networks. In Touretzky, D., editor, *Advances in Neural Information Processing Systems (volume 1)*. Morgan Kaufmann, San Mateo, CA.

LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. In Touretzky, D., editor, *Advances in Neural Information Processing Systems (volume 2)*. Morgan Kaufmann, San Mateo, CA.

McMillan, C., Mozer, M., & Smolensky, P. (1991). The connectionist scientist game: Rule extraction and refinement in a neural network. In *Proc. of the 13th Conf. of the Cognitive Science Society*, Chicago, IL. Erlbaum.

Mooney, R., Shavlik, J., Towell, G., & Gove, A. (1989). An experimental comparison of symbolic and connectionist learning algorithms. In *Proc. of the 11th IJCAI*, (pp. 775–780), Detroit, MI. (A longer version appears in *Machine Learning*, 6).

Nowlan, S. J. & Hinton, G. E. (1992). Simplifying neural networks by soft weight-sharing. *Neural Computation*, 4:473–493.

Ourston, D. & Mooney, R. J. (1990). Changing the rules: A comprehensive approach to theory refinement. In *Proc. of the 8th Nat. Conf. on Artificial Intelligence*, (pp. 815–820), Boston, MA.

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

Pomerleau, D. A. (1991). Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3:88–97.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA.

Rissanen, J. (1978). Modeling by shortest data description. *Automatica*, 14:465–471.

Saito, K. & Nakano, R. (1988). Medical diagnostic expert system based on PDP model. In *Proc. of the IEEE International Conf. on Neural Networks*, (pp. 255–262), San Diego, CA. IEEE.

Sejnowski, T. & Rosenberg, C. (1987). Parallel networks that learn to pronounce English text. *Complex Systems*, 1:145–168.

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

Towell, G. G. & Shavlik, J. W. (1991). Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. In Moody J., Hanson S. and Lippman, R., editors, *Advances in Neural Information Processing Systems (volume 4)*. Morgan Kaufmann, San Mateo, CA. (A longer version will appear in *Machine Learning*).

Towell, G. G., Shavlik, J. W., & Noordewier, M. O. (1990). Refinement of approximately correct domain theories by knowledge-based neural networks. In *Proc. of the 8th Nat. Conf. on Artificial Intelligence*, (pp. 861–866), Boston, MA. MIT Press.