Appears in *Machine Learning: Proceedings of the Eighth International Workshop*,
Birnbaum, L. and Collins, G. (eds.), Morgan Kaufmann, San Mateo, Ca., 1991

Constructive Induction in Knowledge-Based Neural Networks

Geoffrey G. Towell **Mark W. Craven** **Jude W. Shavlik**
Department of Computer Sciences
University of Wisconsin
Madison, Wisconsin 53706
email: {towell, craven, shavlik}@cs.wisc.edu

Abstract

Artificial neural networks have proven to be a successful, general method for inductive learning from examples. However, they have not often been viewed in terms of constructive induction. We describe a method for using a knowledge-based neural network of the kind created by the KBANN algorithm as the basis of a system for constructive induction. After training, we extract two types of rules from a network: modified versions of the rules initially provided to the knowledge-based neural network, and rules which describe newly constructed features. Our experiments show that the extracted rules are more accurate, at classifying novel examples, than the trained network from which the rules are extracted.

1 INTRODUCTION

Artificial neural networks (ANNs) have proven to be a powerful and general technique for machine learning. For example, a host of empirical comparisons indicate that ANNs are at least as effective at generalizing from training to testing examples as any of several common symbolic machine learning algorithms [Atlas89, Fisher89, Shavlik91, Weiss89]. This generalization aptitude results from the ability of ANNs to train hidden units to form useful intermediate representations. In other words, the success of ANNs results from their ability to use hidden units as loci for constructive induction. However, ANNs tend to be "black boxes" after training; their intermediate representations are not easily comprehended by humans. As a result, features constructed by ANNs cannot be passed along for use on related problems.

This paper describes work aimed at making the constructed features of an ANN available for human inspection and understanding. The method begins by using the KBANN algorithm [Towell90] to supply a *Knowledge-based Neural Network* (KNN) with much of the intermediate information that it would otherwise have to derive in order to correctly solve a problem. Hence, KBANN reduces the scope of the problem that must be solved, and simplifies the task of understanding the features discovered during training. The KNN is then trained using a set of classified examples and standard neural learning methods [Rumelhart86]. Finally, the NOFM algorithm (described below), which extracts rules from trained KNNs, is used to understand the new feature representations that arise during training and how the initial knowledge provided to the KNN has been adapted. We present results of using KNNs as a tool for constructive induction in the molecular genetics problem of recognizing *splice junctions*.

2 THE KBANN ALGORITHM

The KBANN algorithm uses a knowledge base of domain-specific inference rules (a *domain theory*), in the form of PROLOG-like clauses, to determine the topology and initial weights of a KNN. The domain theory need be neither complete nor correct; it need only support approximately correct reasoning. KBANN translates a domain theory into a KNN in which units and links correspond to parts of the domain theory. A detailed explanation of the procedure used by KBANN to translate rules into an ANN can be found in [Towell90].

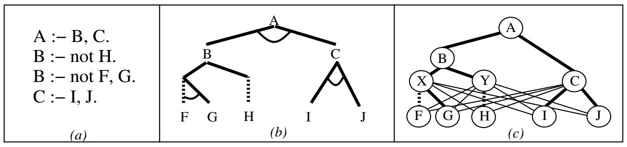

Figure 1: Translation of a Domain Theory into a KNN

As an example of KBANN, consider the artificial domain theory in Figure 1a, which defines membership in category A. Figure 1b represents the hierarchical structure of these rules: solid and dotted lines represent necessary and prohibitory dependencies, respectively. Figure 1c represents the KNN that results from the translation of this domain theory into a neural network. Units X and Y in Figure 1c are introduced into the KNN to handle the disjunction in the rule set. Otherwise, each unit in the KNN corresponds to a consequent or an antecedent in the domain theory. The

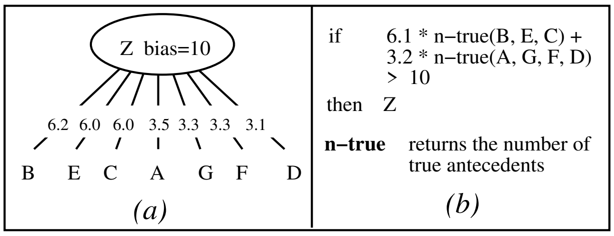

Figure 2: Example of the NOFM Algorithm

thick lines in Figure 1c represent heavily-weighted links in the KNN that correspond to dependencies in the domain theory. The thin lines represent the links added to the network to allow refinement of the domain theory.

3 RULE EXTRACTION

This section briefly presents the NOFM method, an algorithm for extracting rules from trained KNNs [Towell91]. The method makes two assumptions about these networks. First, that training a KNN does not significantly shift the meaning of its units. By making this assumption, the methods are able to attach labels to extracted rules that correspond to consequents in the symbolic knowledge upon which the KNN is based, thereby enhancing the comprehensibility of the rules. Second, that the units in a trained KNN either have activation near one) or near zero. By making this assumption, each non-input unit in a trained KNN can be treated as a step function or a Boolean rule. Observation of trained KNNs suggests that both of these assumptions are valid.

The NOFM method searches for antecedents of the form:
if (N of these M antecedents are true) then . . .
NOFM was suggested by experiments that indicate neural networks are good at learning N-of-M concepts [Fisher89] and our observation that rule sets extracted by methods similar to [Saito88] often contain subsets which can be compactly expressed as N-of-M concepts. The NOFM method finds groups of links that have approximately the same weight and treats each group as if all of the weights in it were the same.

As an example of the NOFM algorithm, consider Figure 2a that shows a unit of a trained KNN with seven heavily-weighted antecedents (lowly-weighted antecedents are not shown). The rules extracted by NOFM are similar to Figure 2b. It is often possible to simplify these rules, removing the weights and thresholds. However, even without such simplification, extracted rules are normally much easier to comprehend than networks.

A primary advantage of the NOFM algorithm is that it does not need to exhaustively search through combinations of links to find rules. As a result, NOFM runs in polynomial time with respect to the number of incoming links to a unit whereas other methods [Saito88] require exponential time. Also, NOFM does not require alterations to the structure of networks to allow extraction [Sestito90]. The rules derived by NOFM, however, may lack the simplicity of straightforward, PROLOG-like rules extracted using other methods.

4 THE SPLICE-JUNCTION PROBLEM

Splice junctions are the points on a DNA sequence at which “superfluous” DNA is removed during the process of protein creation. The problem we have addressed is to recognize the boundaries between *exons* (the parts of the DNA sequence retained after splicing) and *introns* (the parts of the DNA sequence that are spliced out) in a given sequence of DNA. Figure 3 illustrates how splicing occurs during the process of protein creation. This problem consists of two subtasks: recognizing exon/intron boundaries (referred to as E/I), and recognizing intron/exon boundaries (referred to I/E).

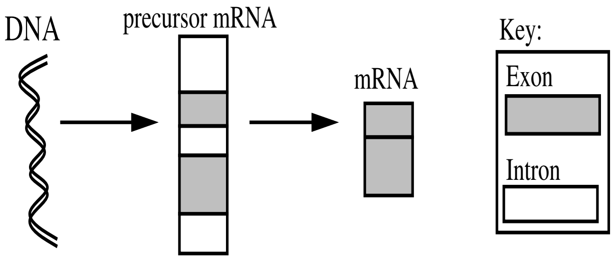

Figure 3: The Organization of Genes in Higher Organisms

To approach this problem, we use a dataset initially described in [Noordewier91]. This dataset contains 3190 examples, of which 25% are *I/E*, 25% are *E/I* and the remaining 50% are *neither*. Each example consists of a 60 nucleotide-long DNA sequence categorized according to the type of boundary at the center of the sequence. A domain theory that classifies 61% of the examples correctly was derived from the biological literature [Watson87]. An abstracted view of the KNN resulting from the theory is depicted in Figure 4.

Previously reported experiments [Noordewier91] indicate that this domain theory does not include all of the intermediate terms necessary to learn the concept of a splice-junction. Therefore, the KNN requires hidden units in addition to those specified by the domain theory to achieve a high level of accuracy. These extra hidden units enable the network to constructively induce the requisite terms not captured in the domain theory. The following sections discuss some of the problems that arise in incorporating hidden units not specified by a domain theory into a KNN, and how these problems can be addressed

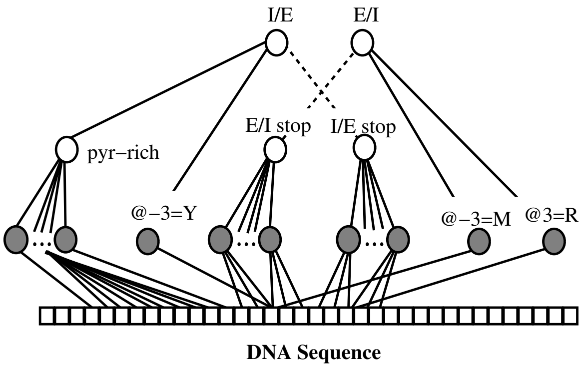

Figure 4: Initial Splice-Junction KNN
Shaded units represent "definitional" rules in the domain theory. Neither incoming weights nor the bias can change during training for these units.

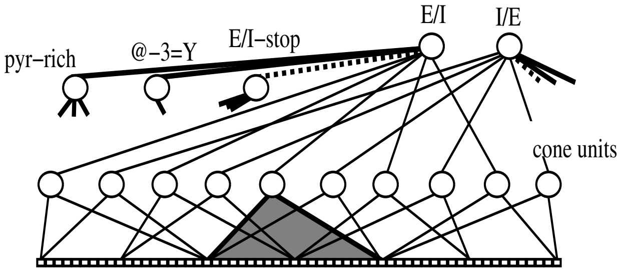

Figure 5: Splice-Junction Network with Cone Units

5 ADDING UNITS FOR CONSTRUCTIVE INDUCTION

While the performance of a neural network is dependent on its topology [Kolen90, Shavlik91], how to organize the topology of a network in order to best learn a particular task is an open question. By using the domain knowledge of a problem to specify network topology, KBANN directly addresses this issue and eliminates the need to search network-topology space. However, adding extra hidden units to capture features not expressed in the domain theory reopens the problem of topology determination. Specifically, the number and connectivity of the added hidden units must be determined.

A guiding principle used to determine the number and connectivity of the added hidden units was to, whenever possible, make decisions about topology to simplify the problem of interpreting trained KNNs. A second idea that guided topology determinations was an analogy between vision and the scanning of DNA sequences. This analogy suggests the use of *recognition cones* [Honavar88] for scanning a DNA sequence. That is, individual hidden units should be connected only to input units representing a short, contiguous subsequence of the DNA strand. The analogy to recognition cones is supported by the biology of DNA, which suggests that certain localized regions on a DNA sequence are key in recognizing important biological signals. Moreover, this analogy at least partially resolves both the number and connectivity issues raised by the added units.

In the experiments reported below, each "cone unit" covered a sequence 20 nucleotides long. This cone size is slightly longer than several DNA *features* which are specified by the splice-junction domain theory. Cones were given 50% overlap. Hence, except at the ends of the sequence, where important information is least likely to be found, every input unit is covered by two cones. Networks with more cones of shorter length were tried, as were architectures which varied the amount of overlap between cones. None of the variations had a significant effect on network performance.

Each cone in the network is connected directly to only one of the output units rather than to an intermediate layer of hidden units. (Figure 5 is a schematic representation of the splice junction network with cones added.) This simplifies interpretation because it does not allow multiple output units to encode different features with a shared cone. Supporting the decision to connect cones units to only one output unit is Dietterich's suggestion that sharing of hidden units is not an important contributor to the classification abilities of ANNs [Dietterich90] .

The input and output weights and the biases of the cone units were initialized to random, near-zero values. The near-zero values of these parameters reflect the fact that, unlike the units that are specified by the domain theory, the roles of the cone units are completely unknown before training.

6 INTERPRETING ADDED HIDDEN UNITS

Three problems arise in trying to apply the NOFM interpretation algorithm to KNNs which include cone units. First, units in KNNs are labeled as a result their initial correspondence to parts of a domain theory. The cone units, however, have no similar labels. Until the biological significance of a feature measured by a cone unit can be determined, the region of the DNA sequence measured by the unit serves as a consequent name in extracted rules.

The second problem is that the rules extracted from cone units often contain unnecessary double negatives. This occurs because the biases of the cone units and the links into and out of the units are initialized to near-zero values. As a result, the signs on the links and biases of the cone units after training are meaningful in relation to each other, but arbitrary with respect to the concepts with which the units are associated. Hence, the result of simply applying NOFM to the cone units is rules which contain many unnecessary double negatives. To alleviate this problem, cones with negative biases were inverted prior to interpretation. That is, the signs of the bias and all input and output links were reversed. The biases at all receivers of links from these reversed units were adjusted to reflect the changed signal.

The third problem is related to the way in which the NofM algorithm approximates each network unit with a linear threshold unit (LTU). As previously discussed, the NofM method assumes that the units in a KNN tend have activations near zero or one. When this condition of activation bifurcation is true, NofM is able to accurately approximate each unit with a LTU. While activation bifurcation almost always holds true after training in those KNN units that are specified by the domain theory, it is not as likely to hold true for cone units, since they are not initially configured to approximate LTUs. Therefore, it was occasionally necessary to coax the cone units into approximating LTUs by progressively steepening the logistic activation function for these units during training.

With each of these issues addressed, the NofM method can be used to interpret the rules encoded by the added hidden units, thereby making the constructive inductions of KNNs available for human review.

7 EXPERIMENTAL RESULTS

The utility of using KNNs as a method for constructive induction can be measured on at least two independent axes: (1) the ability of the rules to retain the accuracy of the KNN from which they were extracted, and (2) the comprehensibility of the extracted rules. Figure 6 addresses the first issue, plotting the accuracy of extracted rules versus the accuracy of the networks from which the rules were derived. All results in Figure 6 are for a 1000 example subset chosen randomly from the 3190 available examples. Testing was done using repeated 10-fold cross-validation [Weiss90]. While the extracted rules do not perform as well as networks on the training examples, the error rates of the rules are slightly better than those of the networks on the testing examples. This suggests that the extracted rules capture meaningful regularities in the training set while avoiding spurious correlations that account for the performance of the networks on the training sets. In addition, Figure 6 presents results for KNNs that do not have cone units. Clearly, the addition of cone units improved the accuracy of both the network and the extracted rules.

There are two aspects to the comprehensibility of constructive inductions: are the individual rules extracted from a network understandable, and are nearly the same sets of rules derived when training is slightly modified? Table 1, which presents a representative set of rules extracted from cone units by NofM from one of the trials of a ten-fold cross-validation test[1] contains information relevant to each of these questions. Looking first at the comprehensibility of individual rules, the largest of these rules has six an-

[1]Ten-fold cross-validation provides a good way to approach the latter issue because each training set has 89% of the examples in common with every other training set.

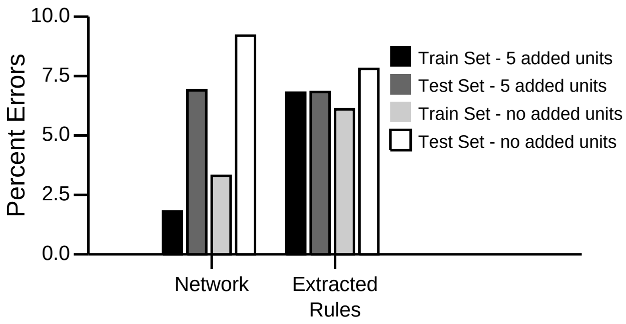

Figure 6: Error Rates on Training and Testing Examples

tecedents. Recall that the number of antecedents does not tell the whole comprehensibility story as each antecedent is weighted. While weights are not shown here for clarity, rules rarely had antecedents with more than two different weights. Hence, the rules extracted from a *single* training episode are certainly comprehensible.

Table 1: Typical Rules Extracted From Cone Units

consequent	*antecedents*
E/I[-20..-1]	@-3=C, @-2=A, @-1=G.
E/I[-10..10]	@-9=t, @-3=C, @-2=A, @-1=G, @5=c.
I/E[-10..10]	@-5=c, @-1=g, @1=G, @2=T, @3=a, @5=G.
I/E[1..20]	@1=G, @2=T, @3=A, @5=G, @14=c, @18=g.

The notation E/I[-20..-1] signifies a cone connected to the E/I output unit that looks at a subsequence starting 20 nucleotides before, and ending at, the splice-junction. Antecedents in upper-case appear seven or more times during a single 10-fold cross-validation test.

Moreover, as indicated by the antecedents shown in upper-case, the extracted rules are quite consistent. Fourteen of the twenty antecedents in Table 1 appear in at least six of the other nine trials.

8 DISCUSSION

On each of the measures described above, the rules extracted from the cone units are successful. The rules are both short and consistent, and hence quite comprehensible. The extracted rules are more accurate at classifying testing examples than the network from which they are derived. This improvement in accuracy result from the extracted rules picking up the important generalizations made by the network, while ignoring small correlations that allow the network to perform well on the training set. Furthermore, the rules constructed by the cone units seem to identify biologically-significant structures. We are currently working on more detailed assessments of the "interestingness" of the constructed rules.

It is important to note that the rules extracted from the cone units are not, in themselves, sufficient for the recognition of either E/I or I/E boundaries. Instead, the constructed rules lend additional evidence to support predicates in the initial domain theory. This supports a suggestion made previously that KNNs are particularly well suited to constructive induction because the presence of knowledge in the KNN considerably reduces the problems that must be resolved through feature construction.

Our plans for enhancing the utility of KNNs for performing constructive induction are primarily aimed at improving methods for the extraction of rules from trained KNNs. We are currently working on a rule-extraction algorithm that operates *during* the training of a KNN. Rather than allowing link weights to freely take on arbitrary values, this algorithm periodically rounds each link weight into a member of a predetermined, small set of values. Instead of undergoing the transitions from an interpretable set of rules to a black-boxish KNN and back to an interpretable set of rules, the rounding algorithm should preserve the comprehensibility of the knowledge base during training.

9 CONCLUSIONS

We have described a method for using neural networks to perform constructive induction. By using the KBANN system to form a knowledge-based neural network, we have significantly reduced the difficulty of learning and interpreting the necessary intermediate representations formed by hidden units. Our technique for deriving rules from a trained KNN is able to express the information captured by the network in a manner easily comprehended by humans.

To make use of this interpretation technique, we specialized the computer vision concept of recognition cones to apply to DNA sequence analysis. This use of recognition cones constrained the possible inductions of the hidden units not specified by the domain theory, thereby facilitating interpretability of the constructed rules.

Our efforts to promote the interpretability of the constructed rules were successful. The trained network constructed four new rules, each of which identified significant aspects of the domain we studied. Additionally, the derived rules were easily comprehensible, averaging fewer than six antecedents. Moreover, the rules extracted from the trained network (in which the constructed rules play an integral part) are more accurate on sets of testing examples than the networks from which they were extracted.

Acknowledgements
This work was partially supported by Office of Naval Research Grant N00014-90-J-1941 and National Science Foundation Grant IRI-9002413 and Department of Energy Grant DE-FG02-91ER61129.

References

[Atlas89] Atlas, L., Cole, R., Connor, J., El-Sharkawi, M., Marks II, R. J., Muthusamy, Y., and Barnard, E. (1989). Performance comparisons between backpropagation networks and classification trees on three real-world applications. In *Advances in Neural Information Processing Systems*, volume 2, pages 622–629, Denver, CO.

[Dietterich90] Dietterich, T. G., Hild, H., and Bakiri, G. (1990). A comparative study of ID3 and backpropagation for English text-to-speech mapping. In *Proceedings of the Seventh International Conference on Machine Learning*, pages 24–31, Austin, TX.

[Fisher89] Fisher, D. H. and McKusick, K. B. (1989). An empirical comparison of ID3 and back-propagation. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 788–793, Detroit, MI.

[Honavar88] Honavar, V. and Uhr, L. (1988). A network of neuron-like units that learns to perceive by generation as well as reweighting of its links. In Hinton, G. E., Sejnowski, T. J., and Touretzky, D. S., editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 472–484. Morgan Kaufmann, San Mateo, CA.

[Kolen90] Kolen, J. F. and Pollack, J. B. (1990). Back-propagation is sensitive to initial conditions. In *Advances in Neural Information Processing Systems*, volume 3, Denver, CO. Morgan Kaufmann.

[Noordewier91] 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*, volume 3, Denver, CO. Morgan Kaufmann.

[Rumelhart86] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E. and McClelland, J. L., editors, *Parallel Distributed Processing: Explorations in the microstructure of cognition. Volume 1: Foundations*, pages 318–363. MIT Press, Cambridge, MA.

[Saito88] Saito, K. and Nakano, R. (1988). Medical diagnostic expert system based on PDP model. In *Proceedings of IEEE International Conference on Neural Networks*, volume 1, pages 255–262.

[Sestito90] Sestito, S. and Dillon, T. (1990). Using multi-layered neural networks for learning symbolic knowledge. In *Proceedings of the 1990 Australian Artificial Intelligence Conference*, Perth, Australia.

[Shavlik91] Shavlik, J. W., Mooney, R. J., and Towell, G. G. (1991). Symbolic and neural net learning algorithms: An empirical comparison. *Machine Learning*, 6:111–143.

[Towell91] Towell, G. G., Shavlik, J. W., and Craven, M. W. (1991). Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. Technical report, Computer Sciences Department, University of Wisconsin, Madison, WI.

[Towell90] Towell, G. G., Shavlik, J. W., and Noordewier, M. O. (1990). Refinement of approximately correct domain theories by knowledge-based neural networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 861–866, Boston, MA.

[Watson87] Watson, J. D., Hopkins, N. H., Roberts, J. W., Steitz, J. A., and Weiner, A. M. (1987). *Molecular Biology of the Gene*. pages 634–647.

[Weiss89] Weiss, S. M. and Kapouleas, I. (1989). An empirical comparison of pattern recognition, neural nets, and machine learning classification methods. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 688–693, Detroit, MI.

[Weiss90] Weiss, S. M. and Kulikowski, C. A. (1990). *Computer Systems that Learn*. Morgan Kaufmann, San Mateo, CA.