Appears in the Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90, pp. 861–866)

Refinement of Approximate Domain Theories by Knowledge-Based Neural Networks*

Geoffrey G. Towell **Jude W. Shavlik** **Michiel O. Noordewier**

University of Wisconsin — Madison
1210 West Dayton Street
Madison, Wisconsin 53706

Abstract

Standard algorithms for explanation-based learning require complete and correct knowledge bases. The KBANN system relaxes this constraint through the use of empirical learning methods to refine approximately correct knowledge. This knowledge is used to determine the structure of an artificial neural network and the weights on its links, thereby making the knowledge accessible for modification by neural learning. KBANN is evaluated by empirical tests in the domain of molecular biology. Networks created by KBANN are shown to be superior, in terms of their ability to correctly classify unseen examples, to randomly initialized neural networks, decision trees, "nearest neighbor" matching, and standard techniques reported in the biological literature. In addition, KBANN's networks improve the initial knowledge in biologically interesting ways.

Introduction

Explanation-based learning (EBL) (Mitchell *et al.* 1986; DeJong & Mooney 1986) provides a way of incorporating pre-existing knowledge into a learning system. However, the basic algorithms suffer from the fact that the pre-existing knowledge cannot contain imperfections (Mitchell *et al.* 1986). Conversely, empirical learning is a method for learning solely from training examples (e.g., Quinlan 1986). Empirical learning systems have problems such as misclassification due to spurious correlations in the training data.

Recent work (e.g., Flann & Dietterich 1989; Shavlik & Towell 1989) combines empirical and explanation-based learning to overcome the problems of each approach by using training examples to inductively refine pre-existing knowledge. Beyond overcoming the problems of each approach, *hybrid* systems should, after training, be superior to EBL systems in terms of the range of examples over which they are correct. Moreover, given the same set of training examples, hybrid systems should be superior, in terms of classification accuracy, to empirical learning systems.

*This research was partially supported by a grant from the University of Wisconsin Graduate School.

This paper describes the KBANN *(Knowledge-Based Artificial Neural Networks)* hybrid learning system and demonstrates its superiority to empirical and explanation-based learning systems along these dimensions. Briefly, KBANN uses a knowledge base of hierarchically-structured rules which may be both incomplete and incorrect to form an artificial neural network (ANN). In so doing, KBANN makes it possible to apply neural learning techniques to the empirical, incremental improvement of knowledge bases.

At present, KBANN is restricted to non-recursive, propositional (i.e., variable-free) domain theories. Under these restrictions, the ability of EBL to speedup a problem solver (Minton 1988) is not utilized. While this speedup is the primary strength of EBL, the secondary strengths of this form of learning *are* directly applicable. Specifically, the domain theory indicates the features which are believed to be important to an example's classification. The theory also specifies important derived features; through their deduction the complexity of an ANN's final decision is reduced.

The following section presents the KBANN algorithm. In the subsequent section, KBANN is applied to a real-world problem in the domain of molecular biology. KBANN is shown to produce results better than those reported in the biological literature. Additionally, KBANN's results are shown to be superior to randomly started ANNs, ID3 (Quinlan 1986) - a symbolic empirical learning system, and "nearest neighbor" classification. Moreover, ANNs created by KBANN are shown to have improved upon the original domain theory in biologically interesting ways. The paper concludes with a discussion of research related to KBANN and the areas which our research is currently pursuing.

The KBANN Algorithm

KBANN uses a knowledge base of domain-specific inference rules in the form of PROLOG-like clauses to define what is initially known about a topic. The knowledge base need be neither complete nor correct; it need only support approximately correct explanations. KBANN translates the knowledge base into an ANN in which units and links[1] in the ANN correspond to parts of the knowledge base, as described in Table 1. The next section presents the approach KBANN uses to translate rules into neural networks. Subsequent sections present the KBANN algorithm and provide an example of its operation.

Table 1: Knowledge Base – ANN Correspondences

Knowledge Base	*Neural Network*
Final Conclusions	Output Units
Supporting Facts	Input Units
Intermediate Conclusions	Hidden Units
Dependencies	Weighted Connections

Translation of rules

This section describes how KBANN translates rules containing AND, OR and NOT into an ANN. Rules are assumed to be conjunctive, nonrecursive and variable-free; disjuncts are encoded as multiple rules. (To simplify discussion in this section, only binary-valued features are assumed to exist. Handling of non-binary features is described on the next page.)

The KBANN method sets weights on links and biases of units so that units have significant activation only when the corresponding deduction could be made using the knowledge base. For example, assume there exists a rule in the knowledge base with n *mandatory* antecedents (i.e., antecedents which must be true) and m *prohibitory* antecedents (i.e., antecedents which must not be true). The system sets weights on links in the ANN corresponding to the mandatory and prohibitory dependencies of the rule to ω and $-\omega$, respectively. The bias on the unit corresponding to the rule's consequent is set to $n * \omega - \phi$. ϕ is a parameter chosen so that units have *activation* ~ 0.9 when their antecedents are satisfied and *activation* ~ 0.1 otherwise.[2]

This mapping procedure is sufficient only for a purely conjunctive knowledge base. Disjuncts cannot be handled because there is no way to set the bias of a unit that can be "deduced" in multiple ways such that no unintended combinations are allowed. For example, assume there exists a consequent Υ which can be proven by two rules, R_1 and R_2. Further assume, that there are 7 antecedents (labeled to $0, \ldots, 6$) to Υ and that antecedents [0 1 2] are mandatory for R_1 while antecedents [3 4 5 6] are mandatory for R_2. If the antecedents of R_1 and R_2 are all connected to Υ such that either [0 1 2] or [3 4 5 6] can activate Υ, then there is no way to set the bias of Υ such that unwanted combinations (e.g., [0 1 3 4]) cannot also activate Υ.

KBANN handles disjuncts by creating units Υ_1 and Υ_2, which correspond to R_1 and R_2, using the approach for conjunctive rules described above. These units will only be active when their corresponding rule is true. KBANN then connects Υ_1 and Υ_2 to Υ by a link of weight ω and sets the bias of Υ to $\omega - \phi$. Hence, Υ will be active when either Υ_1 or Υ_2 is active.

[1] *Unit* refers to a processing element in a neural network. *Link* refers to a connection between units.

[2] Currently, we use $\omega = 3.0$ and $\phi = 2.3$, values empirically found to work well on several domains.

Algorithm specification

Three additional steps are required to complete ANN following the initial translation of the knowledge base. First, input units corresponding to features of the environment that do not appear as an antecedent of any rule must be added to the network. These units are necessary because an approximately correct knowledge base may not have used some features that are necessary to accurately express a concept. Second, links must be added to the network to give existing rules access to items not mentioned in the knowledge base. These links initially have weight equal to zero. They are placed by grouping units according to their maximum path length from an input unit and adding links between all units in successive groups. Third, the network must be perturbed by adding random numbers within ϵ of zero to all link weights and biases to avoid symmetry breaking problems (Rumelhart *et al.* 1986).[3]

The KBANN algorithm is summarized in Table 2. Once the network is produced, it is refined by providing training examples which are processed using backpropagation (Rumelhart *et al.* 1986).

Table 2: Overview of the KBANN Algorithm

1.	Translate rules to set initial network structure.
2.	Add units not specified by translation.
3.	Add links not specified by translation.
4.	Perturb the network by adding near zero random numbers to all link weights and biases.

Example of the algorithm

As an example of the KBANN method, consider the artificial knowledge base in Figure 1a which defines membership in category A. Figure 1b represents the hierarchical structure of these rules: solid and dotted lines respectively represent necessary and prohibitory dependencies. Figure 1c represents the ANN that results from the translation into a neural network of this knowledge base. Units X and Y in Figure 1c do not correspond directly to consequents in the knowledge base. Rather, the units are introduced to handle the disjunction in the knowledge base as described above. The thick lines in Figure 1c represent the links in the ANN that correspond to dependencies in the explanation. Thus, with $\omega = 3$, the weight on thick solid lines is 3, while the weight on thick dotted lines is -3. The lighter solid lines represent the links added to the network to allow refinement of the domain theory.

[3] We currently use $\epsilon = 0.01$.

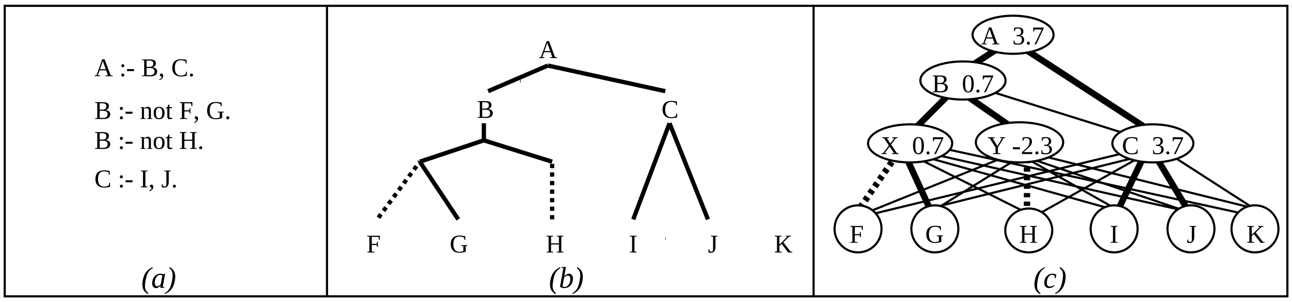

Figure 1: Translation of a Knowledge Base into an ANN

Numbers beside the unit names in Figure 1c are biases of the units. So, with $\phi = 2.3$, the bias of unit B is set to 0.7 so it is activated when either Y or Z is active. The bias of unit A is set to 3.7 so it is active only when both C and B are active. Conversely, the bias of unit X is set to 0.7 so that it will be active when input is received from unit G and not from unit F. Lacking any mandatory antecedents, the bias of Y is set to -2.3. Hence, Y will be active except when H is active.

Handling non-binary features

Currently, the system can handle three types of features: nominal, linear and hierarchical. Discussions of the exact approach used to handle these feature types, and the added information they require, follows.

Nominal Nominally valued features (i.e., features whose possible values can be listed and have no structure) are handled by assigning one input unit to each value of the feature. To do this, KBANN must be given a list of the possible values a feature can have. For example, if the feature color is stated to have three values: *red*, *green* and *blue*, then three input units: *color-is-blue*, *color-is-red* and *color-is-green*, will be created.

Linear Linear features may take on an infinite number of values within some region of the number line. As a result, the method for handling nominal features cannot be used. Instead, KBANN requires a user-defined partitioning of the range over which the feature varies. The partitioning provides a nominal representation (e.g., *small*, *medium*, and *large*) which may be used by rules in a knowledge base. In addition, each partition is assigned an input unit in the ANN. For values within the partition of a particular unit, the unit has an activation of one. Outside the partition, the unit should still be active according to how close the value is to the partition. This allows the network to learn subranges that are not among those initially specified. To implement this, units for linear features have the activation function $\Psi(MAX(0, (ABS(Midpoint - Actual) - Range/2))/Range)$ where: *Midpoint* is the midpoint of the subrange, *Range* is the width of the subrange, *Actual* is the the exact value of the feature, and Ψ is a function based upon the *standard normal* distribution.[4]

Hierarchical Hierarchical features are handled, with one exception, as if a set of rules defined the ISA hierarchy. The exception is that whenever a rule in the knowledge base refers to an element in a hierarchy, in addition to the high weight link from that element, low weight links are created from all ancestors and descendants of the element. So, looking at Figure 2, if a rule contains *non-insulating* as an antecedent, the unit corresponding to the consequent of this rule would be given low weight links to *material*, *paper* and *ceramic*. In this way, the network is given the capability to specialize or generalize the initial rule according to the hierarchy.

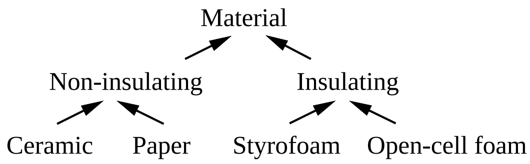

Figure 2: A Hierarchy of Cup Materials

Experimenting with KBANN

This section reports a study of the utility of the KBANN algorithm. The real-world task of recognizing biological concepts in DNA sequences was investigated. In particular, the task was to recognize *promoters* in strings that represent nucleotides (one of A, G, T, or C). A promoter is a genetic region which initiates the first step in the expression of an adjacent gene (*transcription*).

Table 3 contains the initial domain theory used in the promoter recognition task. The first rule says that a promoter involves two subcategories: a `contact` and a `conformation` region. The second rule states that a contact involves two regions, while subsequent rules define alternative ways these regions can appear. This set of rules was easily derived, by one of us (Noordewier, who is also a biologist) from the biological literature (Harley & Reynolds 1987; Hawley & McClure 1983). It

[4] The *standard normal* distribution is a common statistical probability distribution.

should be noted that this domain theory fails to correctly classify any positive example in the training set.

Table 3: A Domain Theory For Promoters

promoter	:-contact, conformation.
contact	:-minus_35, minus_10.
minus_35	:-@-37 "cttgac".
minus_35	:-@-36 "ttgxca".
minus_35	:-@-36 "ttgaca".
minus_35	:-@-36 "ttgac".
minus_10	:-@-14 "tataat".
minus_10	:-@-13 "taxaxt".
minus_10	:-@-13 "tataat".
minus_10	:-@-12 "taxxxt".
conformation	:-@-45 "aaxxa".
conformation	:-@-45 "axxxa", @-4 "t", @-28 "txxxtxaaxxtx".
conformation	:-@-49 "axxxxt", @-1 "a", @-27 "txxxxaxxtxtg".
conformation	:-@-47 "caaxttxac", @-22 "gxxxtxc", @-8 "gcgccxcc".

The input features are 57 sequential DNA nucleotides. A special notation is used to simplify specifying locations in the DNA sequence. The biological literature counts locations relative to the site where transcription begins. Fifty nucleotides before and six following this location constitute an example. When a rule's antecedents refer to input features, they first state the starting location, then list the sequence that must follow. In these specifications, "x" indicates that *any* nucleotide will suffice. Hence, the first rule for `conformation` says that there must an "a" 45 nucleotides *before* the site where transcription begins. Another "a" must be at position -44, then any two nucleotides can appear, and finally there must be a "t" at location -41.

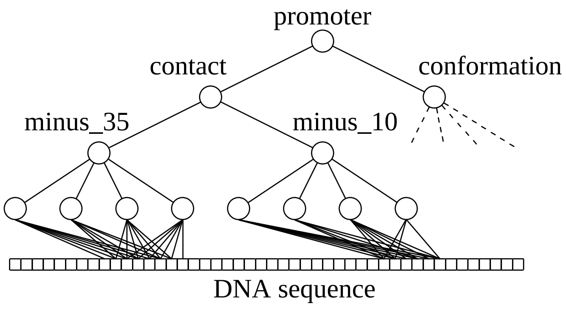

Figure 3: The Initial ANN for Promoter Recognition

This domain theory is translated by KBANN into a neural network with the topology shown in Figure 3. Recall that the algorithm adds additional low-weighted links (not shown) so that if additional sequence information is relevant, the algorithm can capture that information during training.

Fifty-three sample promoters and 53 nonpromoter sequences were used to refine the initial neural network. The 53 sample promoters were obtained from a compilation produced by Hawley and McClure (1983). An initial concern of ours was the construction of negative training examples (i.e., sequences that contained no promoters). Most studies randomly permute sequences in an effort to derive examples that do not meet consensus criteria described below, but nonetheless retain the correct nucleotide frequencies (Lapedes *et al.* 1989). DNA, however, is known to be highly non-random. Negative training examples were thus derived by selecting contiguous substrings from a 1.5 kilobase sequence provided by Prof. T. Record of the Univ. of Wisconsin's Chemistry Dept. This sequence is a fragment from *E. coli* bacteriophage *T7* isolated with the restriction enzyme *Hae*III. By virtue of the fact that the fragment does not bind RNA polymerase, it is believed to not contain any promoter sites [Record, personal communication].

In order to get an estimate of how well the algorithm learned the concept of promoter, a standard experimental methodology called "leave-one-out" (or "cross-validation") was used. This technique operates by training using $N-1$ examples, then testing using the example left out. The procedure is repeated N times ($N = 106$ here), so that each example is excluded once from the training set. The error rate is the number of errors on the single test cases, divided by N. This procedure was repeated 10 times for neural-based algorithms because they use random numbers during initialization and training.

Using the same methodology, three other learning algorithms were applied: standard backpropagation, Quinlan's ID3 (1986), and "nearest neighbor." For standard backpropagation, the same number of hidden units (16) was used as was used in the ANNs created by KBANN .[5] All of the input units were connected to each hidden unit and every hidden unit was connected to the output unit. All weights were randomly initialized to a number near zero. ID3 is a non-connectionist empirical learning algorithm. It uses training data to construct a decision tree for determining the category of an example. At each step, a new node is added to the decision tree by partitioning the training examples based on their value along a single, statistically most-informative feature. "Nearest neighbor" compares the current instance to all known instances, locating exact matches or the k most similar. The classification of the instance is the classification of the majority of the k most similar neighbors. With distance defined as the number of mismatched nucleotides, $k = 3$ was found to work best on this task.

Table 4 contains the number of errors on the 106

[5] Networks with 16 hidden units were locally superior to networks with a greater (21) or lesser (11) number of hidden units.

training examples for the three learning algorithms.[6] In all cases, each algorithm correctly classified all members in the training sets. Hence, although each algorithm fully accounted for the training data, KBANN did a better job of *generalization*, in that its error rate on previously unseen examples was substantially lower. Finally, Table 4 contains the results of O'Neill's (1989b) *ad hoc* partial pattern matching approach for promoter recognition that is the best method reported in the biological literature.

Table 4: Error Rates in the Promoter Experiment

System	*Error Rates*
KBANN	4/106
Standard Backpropagation	8/106
O'Neill	12/106
Nearest Neighbor	13/106
ID3	19/106

As KBANN and standard backpropagation were each run 10 times, the respective error rates can be statistically compared. The t-test indicates that on individual runs KBANN is superior to Standard Backpropagation with 99.95% certainty ($t = 5.29, d.f. = 18$).

Human inspection of the network after learning clearly showed which input positions were most important for recognizing promoters. Combinations of as few as six nucleotides were found to be sufficient to discriminate promoters from nonpromoters. By contrast, a recent study using more conventional techniques suggested using a minimum of twelve nucleotides (O'Neill 1989a). This "consensus sequence" was determined by noting which positions displayed the same base in greater than 50% of the class of promoters under study. Unfortunately, such a consensus fails to recognize any true promoters, due to excessive stringency if exact matches are required at each position. Furthermore, KBANN's neural network assigned particular importance to bases in certain positions. These highlighted positions correspond exactly to the most conserved bases in (Hawley & McClure 1983). Finally, the network learned that certain values for some base pairs indicate that a promoter is probably not present. For instance, a C in base pair -8 and an A in base pair -36 both strongly suggest that a promoter is not present. This ability may be useful to address the problem that promoters lose their biological activity when specific single nucleotides are mutated (Youderian *et al.* 1982). O'Neill notes that this is an unresolved problem for consensus methods, since the alteration of a single base does not degrade the quality of the match very much. A neural network, on the other hand, is capable of severely penalizing individual bases, by attaching large negative weights to the input units representing those positions.

This experiment demonstrates, using an important real world problem, the promise of the KBANN approach. It produced a more accurate recognizer of promoters, demonstrating the value of incorporating preexisting knowledge about the task being learned.

[6] Rather than simply taking the average of the error rates over 10 runs for the neural learning algorithms, the activation of the output for each test example in each of the 10 runs was averaged. This average output was then used to determine the classification of each example. This technique slightly reduced the error rates of both neural-based approaches.

Related Work

This paper extends and realistically tests the ideas first presented in (Shavlik & Towell 1989).

One problem, specific to neural networks, addressed by KBANN is topology determination. In relatively early work on ANNs, topological decisions were restricted to the size of a single layer of hidden units in fully-connected networks (e.g., Rumelhart *et al.* 1986). This decision is important, because an ANN with too few units will be unable to learn a concept, and an ANN with too many hidden units may generalize poorly (Kruschke 1988). More recently, full connectivity has been shown to hinder learning on some tasks (Rueckl *et al.* 1988). Moreover, different random settings of link weights can result in radically different learning rates and generalization (Shavlik *et al.* in press). Thus, determining the topology of an ANN requires deciding about: the pattern of connectivity, the number and distribution of hidden units, and the link weights.

In general, two approaches have been taken to this problem. The first approach, similar in spirit to KBANN, makes most or all topological decisions prior to training (Rueckl *et al.* 1988; Katz 1989). The second approach modifies network structure as a part of the learning process. This approach includes recruitment learning (e.g., Honavar & Uhr 1988) in which hidden units are added to the network as during learning and methods for removing excess hidden units (e.g., Kruschke 1988).

A second problem specific to neural networks is the integration of existing information into the network. Complex, hand-designed networks (e.g., Rueckl *et al.* 1988) can be viewed as an attempt to give networks some implicit knowledge of a problem domain. However, little work other than KBANN, has been done on how to explicitly give ANNs background information. The work that has been done is similar in approach to KBANN but does not focus on improving incorrect domain theories. For example, Katz' (1989) work stresses improving the execution speed of neural networks by adding links that effectively reduce the depth of the network.

ANNs have been essentially unused as a tool for improving approximately correct domain theories. However, much work has been done on the use of other empirical learning techniques to modify and correct domain theories. For instance, the IOE system (Flann & Dietterich 1989) uses conventional inductive learning to empirically analyze a collection of explanations, thereby refining the domain theory.

Current Research Issues

An extension to KBANN being pursued is automatic interpretation of networks after training. As pointed out in the molecular biology experiments, interpretation of ANNs after learning can be helpful in understanding why the ANN behaves as it does. Automatic translation is expected to take advantage of the meaningful starting configuration of the ANN to allow the post-learning ANNs to be understood. Preliminary investigations suggest that analysis of the changes in link weights and biases in combination with observation of activations over several inputs can provide an accurate picture of how the network arrives at its conclusions.

Another extension currently receiving attention is the use of reasoning by explanation failure to constrain error propagation in the network. The method, based upon work by Hall (1988), makes directed changes to link weights when false negative answers are generated.

A further extension to KBANN is the addition of hidden units to the network beyond those specified by the knowledge translation. These added units would allow the network to learn relations not anticipated in the pre-existing knowledge. Currently we are considering adding hidden units as a fixed percentage of the existing hidden units at each layer in the ANN. Other methods for adding hidden units such as recruitment learning (e.g., Honavar & Uhr 1988) are also being investigated.

Conclusions

The KBANN approach has been shown to make it possible to use ANNs to refine pre-existing knowledge. In addition, it was demonstrated that the KBANN method can automatically generate ANNs that are well-suited to the task they are intended to learn. KBANN does this by using a knowledge base of approximately correct, domain-specific rules to determine the structure and set the initial weights for an ANN.

An experiment in molecular biology demonstrated the effectiveness of the KBANN approach. Taking advantage of a knowledge-based initialization, networks created by KBANN were superior in terms of their generalization ability to randomly initialized networks, classification trees, "nearest neighbor" methods, and the best technique reported in the biological literature. Further, neural learning improved the accuracy of the provided domain theory. Thus, the KBANN method provides a technique both for automatically generating ANNs with good initial topologies and for empirically improving domain theories.

References

DeJong, G. and Mooney, R. 1986. Explanation-based learning: An alternative view. *Machine Learning*, 1:145–176.

Flann, N. and Dietterich, T. 1989. A study of explanation-based methods for inductive learning. *Machine Learning*, 4:187–226.

Hall, R. 1988. Learning by failing to explain: Using partial explanations to learn in incomplete or intractable domains. *Machine Learning*, 3:45–77.

Harley, C. and Reynolds, R. 1987. Analysis of E. coli promoter sequences. *Nucleic Acids Research*, 15:2343–2361.

Hawley, D. and McClure, W. 1983. Compilation and analysis of Escherichia coli promoter DNA sequences. *Nucleic Acids Research*, 11:2237–2255.

Honavar, V. and Uhr, L. 1988. A network of neuron-like units that learns to perceive by generation as well as reweighting of links. In *Proc. Connectionist Models Summer School*, pages 472–484.

Katz, B. 1989. EBL and SBL: A neural network synthesis. In *Proc. Eleventh Conference of the Cognitive Science Society Conference*, pages 683–689.

Kruschke, J. 1988. Creating local and distributed bottlenecks in hidden layers of back-propagation networks. In *Proc. 1988 Connectionist Models Summer School*, pages 357–370.

Lapedes, A.; Barnes, C.; Burkes, C.; Farber, R.; and Sirotkin, K. 1989. Application of neural networks and other machine learning algorithms to DNA sequence analysis. In *Computers and DNA, SFI Studies in the Science of Complexity VII*. Addison-Wesley, Reading, MA.

Minton, S. 1988. Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, 42:363–391.

Mitchell, T.; Keller, R.; and Kedar-Cabelli, S. 1986. Explanation-based generalization: A unifying view. *Machine Learning*, 1:47–80.

O'Neill, M. 1989a. Escherichia coli promoters: I. Consensus as it relates to spacing class, specificity, repeat substructure, and three dimensional organization. *Journal of Biological Chemistry*, 264:5522–5530.

O'Neill, M. 1989b. Escherichia coli promoters: II. A spacing class-dependent promoter search protocol. *Journal of Biological Chemistry*, 264:5531–5534.

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

Rueckl, J.; Cave, K.; and Kosslyn, S. 1988. Why are "what" and "where" processed by separate cortical visual systems? *Journal of Cognitive Neuroscience*, 1(2).

Rumelhart, D.; Hinton, G.; and Williams, J. 1986. Learning internal representations by error propagation. In Rumelhart, D. and McClelland, J., editors, *Parallel Distributed Processing, Vol. 1*, pages 318–362. MIT Press, Cambridge, MA.

Shavlik, J. and Towell, G. 1989. An approach to combining explanation-based and neural learning algorithms. *Connection Science*, 1:233–255.

Shavlik, J.; Mooney, R.; and Towell, G. in press. Symbolic and neural net learning algorithms: An empirical comparison. *Machine Learning*. Forthcoming.

Youderian, P.; Bouvier, S.; and Susskind, M. 1982. Sequence determinants of promoter activity. *Cell*, 10:843–853.