Appears in the Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92).

Using Knowledge-Based Neural Networks to Improve Algorithms: Refining the Chou-Fasman Algorithm for Protein Folding

Richard Maclin and **Jude W. Shavlik**
Computer Sciences Department
University of Wisconsin
Madison, Wisconsin 53706
email: maclin@cs.wisc.edu

Abstract

We describe a method for using machine learning to refine algorithms represented as generalized finite-state automata. The knowledge in an automaton is translated into an artificial neural network, and then refined with backpropagation on a set of examples. Our technique for translating an automaton into a network extends KBANN, a system that translates a set of propositional rules into a corresponding neural network. The extended system, FSKBANN, allows one to refine the large class of algorithms that can be represented as state-based processes. As a test, we use FSKBANN to refine the Chou-Fasman algorithm, a method for predicting how globular proteins fold. Empirical evidence shows the refined algorithm FSKBANN produces is statistically significantly more accurate than both the original Chou-Fasman algorithm and a neural network trained using the standard approach.

Introduction

As machine learning has been increasingly applied to complex real-world problems, many researchers have found themselves turning to systems that refine existing theories rather than building theories from scratch. Ignoring existing knowledge is dangerous, since the resulting learned concept may not contain important factors already identified in earlier work. Our research extends the KBANN system (Towell et al., 1990). KBANN uses knowledge represented as simple, propositional rules (a *domain theory*) to create an initial neural network containing the knowledge from the rules. Work in the domain of gene recognition (Towell, 1991) shows knowledge-based neural networks are more effective than randomly configured networks – even when the original domain theory is not good at solving the problem. This paper describes an addition to KBANN that extends it for domain theories that employ *state* information.

State is important because researchers outside machine learning generally publish algorithms, rather than the sets of rules which machine learning researchers call domain theories. Many algorithms maintain some sense of state, so our extension makes it easier to use machine learning to refine existing "real-world" knowledge. We test our extended system by refining the Chou-Fasman (1978) algorithm for predicting (an aspect of) how globular proteins fold, an important and particularly difficult problem in molecular biology.

State in a domain theory represents the context of the problem. For example, if the problem is to find a path across a room, the state variables may include whether the light is on. The rules introduced to solve this problem take into account the state of the problem – rules to turn on the light would only be considered when the state indicates the light is off. In this style of problem solving, the problem is not solved in one step, but as a series of actions, each leading to a new state, leading to a goal state (turning on the light, navigating to the couch, etc.). The extended KBANN system, called Finite-State KBANN (FSKBANN), translates domain theories that use state information, represented as generalized finite-state automata (FSAs). As in KBANN, FSKBANN translates the state-based domain theory into a neural network, and refines the network using backpropagation (Rumelhart et al., 1986).

The protein-folding problem is an open problem that is increasingly critical as the Human Genome Project (Watson, 1990) proceeds. The Chou-Fasman algorithm is a well-known and widely-used solution. The protein-folding problem is also interesting because many machine learning techniques have been applied to it, including neural networks (Holley & Karplus, 1989; Qian & Sejnowski, 1988), inductive logic programming (Muggleton & King, 1991), case-based reasoning (Cost & Salzberg, in press), and multistrategy learning (Zhang, 1990). Our work combines the Chou-Fasman algorithm with a neural network to achieve a more accurate result than either method separately.

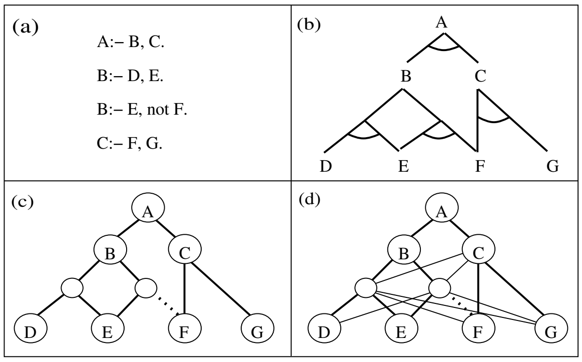

Figure 1: Sample of KBANN: (a) set of rules, (b) dependencies among the rules, (c) corresponding network, and (d) network with near-zero weights added.

This paper presents and empirically analyzes the FSKBANN approach for problem solving in domains where prior state-based knowledge exists. The next section presents the KBANN algorithm and discusses how we extended it to handle state information. The third section defines the protein-folding problem and reviews previous approaches taken. Following that are experiments we performed to test the utility of FSKBANN.

Finite-State KBANN

Before describing FSKBANN, we review the KBANN (for Knowledge-Based Artificial Neural Networks) algorithm (Towell et al., 1990). KBANN translates a domain theory represented as simple rules into a promising initial neural network. This technique allows neural networks to take advantage of pre-existing knowledge.

KBANN takes as input a set of propositional, nonrecursive rules, such as those shown in Figure 1a. Figure 1b shows dependencies among the rules. A dependency is a link between two propositions – arcs show conjunctive dependencies. From the set of dependencies, it is easy to map to a network by replacing each proposition with a unit (and adding units where conjunctions are combined into disjunctions). Figure 1c displays the resulting network. This network has the same behavior as the rules for every input vector. After setting the weights and biases of the units in the network, KBANN connects each unit to any unconnected units at the next lower level in the network using a small-weight link (the resulting network appears in Figure 1d). KBANN adds these connections so that it can learn new dependencies during backpropagation learning. For further details see Towell (1991).

To handle a wider class of problems, we extended KBANN to domain theories represented as *generalized* FSAs[1]. The main extension is the type of network onto which the domain theory is mapped. FSKBANN maps domain theories onto a variant of simple recurrent networks (Elman, 1990), where a subset of the network output is copied back as input of the network in the next step. The copied output represents the previous state calculated by the network and can be used in calculating the next succeeding state.

[1] The notion of FSA in FSKBANN is generalized in that rather than taking a single input value at each step, the FSA may take a *set* of input values.

Table 1: Type of problem solving addressed by FSKBANN.

Given: a *state-dependent domain theory* and
a *goal description*
Repeat
Set *input* = *externally-provided information*
\+
current *internal representation of the problem-solving state*
Produce, using the domain theory and goal description,
output = *result specific to this problem solving step*
\+
next *internal representation of the problem-solving state*
Until a *Termination Criterion* is met.

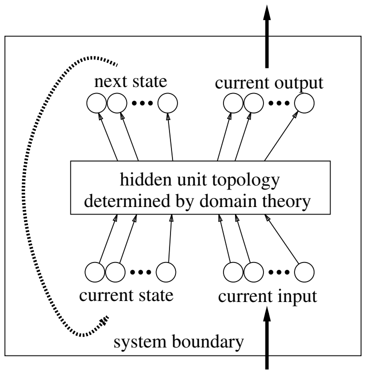

Figure 2: A schematic view of an FSKBANN network.

Table 1 describes the class of problem solvers to which FSKBANN is applicable. Consider a problem solver that determines the next state from externally-provided input and its internal representation of the current state. The externally-provided input may involve a description of the initial state or the measurements of sensors (e.g., as in a reactive planner). The task of the problem solver is to produce the appropriate output for this step in the problem solution (e.g., the operator to apply), as well as choose its internal representation of the next state. This process repeats until a termination condition is met (e.g., a goal state is reached).

The description in Table 1 is a definition of state-based problem solving. The contribution of FSKBANN is a mechanism for using neural networks to *improve* an existing state-dependent domain theory. The inputs and outputs in Table 1 directly map to input and output units in a neural network, and the basic KBANN algorithm uses the domain theory to determine the number and connectivity of the hidden units. Figure 2 shows a

Table 2: Sample primary and secondary structures.

Primary (20 possible amino acids)	S V F L F P P K P K ...
Secondary (three possible structures)	c β β β β c c c α α ...

diagram of the type of network produced by FSkbann.

FSkbann requires the user to provide sample input/output pairs which are used to train the network. It also requires inputs and outputs to be of bounded size, which means the domain theory can only store a finite amount of state. Finally, FSkbann requires the domain theory be propositional, since no good mechanism exists for dealing with predicate calculus variables in neural networks. While these are currently limitations, we posit that many "real-world" algorithms can be represented in this finite-state framework.

The Protein-Folding Problem

This section introduces the protein-folding problem, describes an algorithm from the biological community to solve this problem, and shows how the algorithm is mapped into the above framework. Proteins are long strings of amino acids, several hundred elements long on average. There are 20 amino acids in all (represented by different capital letters). The string of amino acids making up a protein is the *primary* structure of the protein. Once a protein forms, it folds into a three-dimensional shape, known as its *tertiary* structure. Tertiary structure is important because the form of the protein strongly influences its function.

At present, determining the tertiary structure of a protein is costly and time consuming. An alternative approach is to predict the *secondary* structure of a protein as an approximation. Secondary structure in a protein is a description of the local structure surrounding each amino acid. One prevalent system of determining secondary structure divides a protein into three types of structures: (1) α-helix regions, (2) β-strand regions, and (3) random coils (all other regions). For our purposes, we can think of the secondary structure of a protein as simply a sequence corresponding to the primary sequence. Table 2 shows a sample mapping between a protein's primary and secondary structures.

Table 3 contains results of some standard algorithms for solving the secondary-structure problem from the biological literature (Chou & Fasman, 1978; Garnier & Robson, 1989; Lim, 1974). Figure 3 shows the general structure of this type of network. In the data sets used to test the algorithms, 54-55% of the amino acids are part of coil structures, so 54% accuracy can be achieved trivially by predicting coil. Note, many biological researchers believe algorithms which use only local information can achieve at best 80-90% accuracy (Cohen & Presnell, personal communication, 1991).

Table 3: Results of non-learning prediction algorithms.

Method	Accuracy	Comments
Chou & Fasman	58%	data from Qian & Sejnowski (1988)
Garnier & Robson	58%	data from Qian & Sejnowski (1988)
Lim	50%	from Nishkawa (1983)

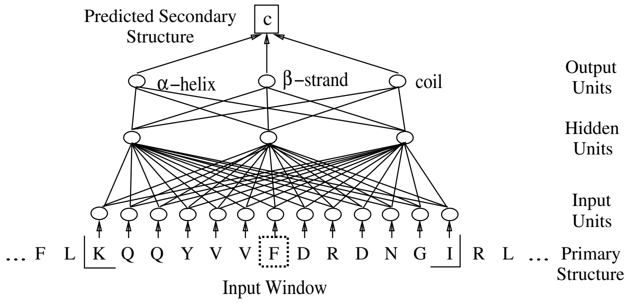

Figure 3: Neural network used by Qian & Sejnowski.

Table 4: Neural network results for structure prediction.

Method	Accuracy	Hidden Units	Window Size
Holley & Karplus	63.2%	2	17
Qian & Sejnowski	62.7%	40	13

A second approach to the secondary-structure problem is to use a neural network (Qian & Sejnowski, 1988; Holley & Karplus, 1989). The neural networks in these efforts have as input a *window* of amino acids consisting of the central amino acid being predicted, plus several amino acids before and after it in the sequence (similar to NETTALK networks, Sejnowski & Rosenberg, 1987). The output of the network is the secondary structure for the central amino acid. Table 4 presents results from these studies, which used different data sets.

Our approach is to combine the knowledge from biological methods into a neural learning method to achieve a better solution. We chose as our biological method the Chou-Fasman algorithm, since this algorithm is widely used. The Chou-Fasman approach finds amino acids that are likely part of α-helix and β-strand regions, and extends these predictions to neighboring amino acids. This algorithm cannot be easily represented using propositional rules, since the prediction for an amino acid may depend on the predictions for its neighbors, but the algorithm can be represented as a generalized FSA (see Maclin & Shavlik, to appear). The resulting network is similar to the network shown in Figure 3 with two major differences. One, the input to the network includes three extra units that contain the past output of the network – the state of the network. Two, the topology of the hidden units is determined by FSkbann, analogously to Figure 1, using the Chou-Fasman algorithm as the domain theory. Table 5 shows how the Chou-Fasman algorithm maps into the FSkbann framework of Table 1.

Table 5: Mapping the Chou-Fasman algorithm into FSKBANN (2° = secondary structure, A.A. = amino acid).

domain theory =	the Chou-Fasman algorithm
goal =	assign a 2° to each A.A.
external input =	a sliding window of A.A.s
current state =	the predicted 2° for the previous A.A.
results =	the predicted 2° for the current A.A.
next state =	*ditto*

Experimental Study

We performed several experiments on the protein problem to evaluate FSKBANN. They demonstrate FSKBANN has a small but statistically-significant gain in accuracy over both standard artificial neural networks (ANNs) and over the non-learning Chou-Fasman algorithm. We also show in-depth empirical analyses of the strengths of the different methods.

Experimental Details

We performed our experiments using the data from Qian and Sejnowski (1988). Their data set consists of 128 segments from 106 proteins with a total of 21,623 amino acids. 54.5% of the amino acids are part of coil structures, 25.2% part of α-helix structures, and 20.3% part of β-strand structures. Ten times we divided the proteins randomly into training and test sets containing two-thirds (85 proteins) and one-third (43 proteins) of the original proteins, respectively.

We used backpropagation to train the neural networks in the two network approaches (FSKBANN and standard ANNs). Training was terminated using *patience*[2] as a stopping criterion. During training, we divided the proteins used for training into two portions – a training set and a *tuning* set. We use the training set to train the network and the tuning set to estimate the generalization of the network. For each epoch, the system trains the network on each of the amino acids in the training set; it then assesses accuracy on the tuning set. We retain the set of weights achieving the highest accuracy for the tuning set and use this set of weights to measure test set accuracy. The system randomly chooses a "representative" tuning set; a tuning set is representative if the percentages of each type of structure (α, β, and coil) in the tuning set approximate the percentages for the training proteins. Note the system does not consider the *testing* set when comparing the percentages. Through empirical testing, we found a tuning set size of five proteins achieved the best results for both FSKBANN and ANNs. Note that this style of training is different from that reported by Qian and Sejnowski. They tested their network periodically, retaining the network that achieved the highest accuracy for the test set.

[2]The patience criterion (Fahlman & Lebiere, 1990) states that training continues until the error rate has not decreased for several training cycles. In this study we set the criterion to be four epochs.

Table 6: Test set accuracies for prediction methods.

Method	Total	Helix	Strand	Coil
Chou-Fasman	57.3%	31.7%	36.9%	76.1%
ANN	61.8%	43.6%	18.6%	86.3%
- w/ state	61.7%	39.2%	24.2%	86.0%
FSKBANN	63.4%	45.9%	35.1%	81.9%
- w/o state	62.2%	42.4%	26.3%	84.6%

Table 7: Correlation coefficients from prediction methods.

Method	Helix	Strand	Coil
Chou-Fasman	0.24	0.23	0.26
ANN	0.35	0.25	0.31
- w/ state	0.32	0.28	0.31
FSKBANN	0.37	0.33	0.35
- w/o state	0.35	0.28	0.32

FSKBANN uses 28 hidden units to represent the Chou-Fasman domain theory. Qian and Sejnowski report that their networks generalized best when they had 40 hidden units. Using the method outlined above, we compared standard ANNs containing 28 and 40 hidden units. We found that networks with 28 hidden units generalized slightly better; hence, for this paper's experiments we use 28 hidden units in our standard ANNs. This also has the advantage that the FSKBANN and ANNs use the same number of hidden units.

Results and Analysis

Tables 6 and 7 contains results averaged over the 10 test sets. The statistics reported are percent accuracy overall, percent accuracy by secondary structure, and correlation coefficients for each structure. The correlation coefficient is good for evaluating the effectiveness of the prediction for each of the three classes separately. The resulting gain in overall accuracy for FSKBANN over both ANNs and the non-learning Chou-Fasman method is statistically significant at the 0.5% level (i.e. with 99.5% confidence) using a t-test.

The apparent gain in accuracy for FSKBANN over ANN networks appears small (only 1.6 percentage points), but this number is somewhat misleading. The correlation coefficients give a more accurate picture. They show that the FSKBANN does better on both α-helix and coil prediction, and much better on β-strand prediction. The reason that the ANN solution does well in overall accuracy is it predicts many coil structures (the largest class) and does well on these predictions.

The gain in accuracy for FSKBANN over the Chou-Fasman algorithm is fairly large and exhibits a corresponding gain in all three correlation coefficients. It is interesting to note that the FSKBANN and Chou-Fasman solutions produce almost the same accuracy for

Table 8: Region-oriented prediction statistics.

Occurrence	Description	FSKBANN	ANN	Chou
α-helix	Average length of predicted helix regions (number of regions).	8.52 (1774)	7.79 (2067)	8.00 (1491)
α-helix / α-helix	Percentage actual helix regions overlap predicted helix regions (length of overlaps).	67% (6.99)	70% (6.34)	56% (5.76)
other / α-helix	Percentage predicted helix regions do not overlap actual helix regions.	34%	39%	36%
β-strand	Average length of predicted strand regions (number of regions).	3.80 (2545)	2.83 (1673)	6.02 (2339)
β-strand / β-strand	Percentage actual strand regions overlap predicted strand regions (length of overlaps).	54% (3.23)	35% (2.65)	46% (4.01)
other / β-strand	Percentage predicted strand regions do not overlap actual strand regions.	37%	37%	44%

β-strands, but the correlation coefficients demonstrate that the Chou-Fasman algorithm achieves this accuracy by predicting more β-strands.

Also shown in Tables 6 and 7 are results for ANNs that included state information – networks similar to Qian and Sejnowski's but where the previous output forms part of the current input vector. These results show that state information alone is not enough to increase the accuracy of the network prediction.

The final row in Tables 6 and 7 shows results for ANNs that use the non-state knowledge from the domain theory. These networks are simple feedforward networks where the topology of the network is determined by the knowledge from the domain theory that does not involve state. These results show state information is integral to the domain theory since networks without could only do as well as standard ANNs.

Finally, to analyze the detailed performance of the various approaches, we gathered additional statistics about the FSKBANN, ANN, and Chou-Fasman solutions. These statistics analyze the results by *regions.* A *region* is a consecutive sequence of amino acids with the same secondary structure. We consider regions because the measure of accuracy obtained by comparing the prediction for each amino acid does not adequately capture the notion of secondary structure as biologists view it (Cohen et al., 1991). For biologists, knowing the number of regions and the approximate order of the regions is nearly as important as knowing the exact structure for each amino acid. The statistics assess how well each solution does on predicting α-helix regions and β-strand regions (see Table 8).

Table 8 gives a picture of the strengths and weakness of each approach. It shows that the FSKBANN solution overlaps slightly fewer actual α-helix regions than the ANNs, but that these overlaps tend to be somewhat longer. On the other hand, the FSKBANN networks *overpredict* fewer regions than ANNs (i.e. predict fewer α-helix regions that do not intersect actual α-helix regions). Table 8 also indicates FSKBANN and ANNs more accurately predict the occurrence of regions than Chou-Fasman approach does.

Table 8 demonstrates that FSKBANN's predictions overlap a much higher percentage of actual β-strand regions than the Chou-Fasman algorithm or ANNs. The ANNs do extremely poorly at predicting overlapping actual β-strand regions. The FSKBANN networks do as well as the ANNs at not overpredicting β-strands, and both do better than the Chou-Fasman method. Taken together, these results indicate that the FSKBANN solution does significantly better than the ANN solution on predicting β-strand regions without having to sacrifice much accuracy in predicting α-helix regions.

Overall, the results suggest more work needs to be done on developing methods of evaluating solution quality. Solutions that find approximate locations of α-helix and β-strand regions and those that accurately predict all three classes should be favored over solutions that only do well at predicting the largest class. Most importantly, the results show that for difficult problems, such as the protein-folding problem, the FSKBANN approach can be worthwhile.

Future Work

FSKBANN uses a domain theory to give a network a "good" set of initial weights, since search starts from that location in weight space. Therefore augmenting the Chou-Fasman domain theory with other information may increase the solution's accuracy. Information in Table 8 indicates current weaknesses. With this knowledge, domain theory extensions addressing these weaknesses can be developed by studying the biological literature.

An interesting property of the networks is that the magnitude of predictions is correlated with their accuracy. This information could be used in a more complex prediction method: instead of predicting all of the protein's structure in one scan, predict only the strongest activated areas first, then feed these predictions back into the network for the next scan.

Finally, a problem with the KBANN approach is extracting information in a human-readable form from the trained network (Towell & Shavlik, 1992). We need to address rule extraction for the augmented networks of FSKBANN to extract learned FSAs.

Related Research

As has been mentioned, the architecture used by FSKBANN, simple recurrent networks, is discussed by Elman (1990). The idea of using this type of network to represent an FSA has been explored by Cleeremans et al. (1989) and Giles et al. (in press). Cleeremans et al. showed that this type of network can perfectly learn to recognize a grammar derived from an FSA. The major difference between FSKBANN and other research on learning FSAs is we focus on using an initial FSA domain theory, rather than learning it from scratch.

Zhang (1990) also applies machine learning to the secondary-structure problem. His method combines information from a statistical technique, a memory-based reasoning algorithm, and a neural network. The best results he reports are 66.4% for a training set size of 96 proteins (Zhang, personal communication, 1991). Another learning technique applied to this problem is the nearest-neighbor algorithm PEBLS (Cost & Salzberg, in press). They report approximately 64% accuracy for a training set similar in size to the one we used.

Conclusions

We present and evaluate FSKBANN, a system that broadens the KBANN approach to a richer, more expressive vocabulary. FSKBANN provides a mechanism for translating domain theories represented as generalized finite-state automata into neural networks. The extension of KBANN to domain theories that include knowledge about state significantly enhances the power of KBANN; rules expressed in the domain theory can take into account the current problem-solving context (i.e. the state of the solution).

We tested FSKBANN by refining the non-learning Chou-Fasman algorithm for predicting protein secondary structure. The FSKBANN-refined algorithm proved to be more accurate than both standard neural network approaches to the problem and a non-learning version of the Chou-Fasman algorithm.

The success of FSKBANN on the secondary-structure problem indicates it may be a useful tool for addressing other tasks including state information. However, work must be done both in improving the neural-network refinement process and the extraction of symbolic knowledge from the trained network.

Acknowledgments

This research was partially supported by NSF Grant IRI-9002413 and ONR Grant N00014-90-J-1941.

References

Chou, P. & Fasman, G. (1978). Prediction of the secondary structure of proteins from their amino acid sequence. *Advanced Enzymology*, 47:45–148.

Cleeremans, A., Servan-Schreiber, D., & McClelland, J. (1989). Finite state automata and simple recurrent networks. *Neural Computation*, 1:372–381.

Cohen, B., Presnell, S., Cohen, F., & Langridge, R. (1991). A proposal for feature-based scoring of protein secondary structure predictions. In *Proc. of the AAAI-91 Workshop on AI Approaches to Classification and Pattern Recognition in Molecular Biology*, (pp. 5–20).

Cost, S. & Salzberg, S. (in press). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*.

Elman, J. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Fahlman, S. & Lebiere, C. (1990). The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems (volume 2)*, (pp. 524–532).

Garnier, J. & Robson, B. (1989). The GOR method for predicting secondary structures in proteins. In Fasman, G., editor, *Prediction of Protein Structure and the Principles of Protein Conformation*. Plenum Press, New York.

Giles, C., Miller, C., Chen, D., Chen, H., Sun, G., & Lee, Y. (in press). Learning and extracting finite state automata with second-order recurrent neural networks. *Neural Computation*.

Holley, L. & Karplus, M. (1989). Protein structure prediction with a neural network. *Proc. of the National Academy of Sciences (USA)*, 86:152–156.

Lim, V. (1974). Algorithms for prediction of α-helical and β-structural regions in globular proteins. *Journal of Molecular Biology*, 88:873–894.

Maclin, R. & Shavlik, J. (to appear). Using knowledge-based neural networks to improve algorithms: Refining the Chou-Fasman algorithm for protein folding. *Machine Learning*.

Muggleton, S. & King, R. (1991). Predicting protein secondary-structure using inductive logic programming. Technical report, Turing Institute, Glasgow, Scotland.

Nishikawa, K. (1983). Assessment of secondary-structure prediction of proteins: Comparison of computerized Chou-Fasman method with others. *Biochimica et Biophysica Acta*, 748:285–299.

Qian, N. & Sejnowski, T. (1988). Predicting the secondary structure of globular proteins using neural network models. *Journal of Molecular Biology*, 202:865–884.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In Rumelhart, D. & McClelland, J., editors, *Parallel Distributed Processing, Volume 1*. MIT Press.

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

Towell, G. (1991). *Symbolic knowledge and neural networks: Insertion, refinement and extraction*. PhD thesis, Dept. of Computer Sciences, Univ. of Wisconsin, Madison, WI.

Towell, G. & Shavlik, J. (1992). Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. In *Advances in Neural Information Processing Systems (volume 4)*.

Towell, G., Shavlik, J., & Noordewier, M. (1990). Refinement of approximate domain theories by knowledge-based neural networks. In *Proc. of the Eighth National Conference on Artificial Intelligence*, (pp. 861–866).

Watson, J. (1990). The Human Genome Project: Past, present, and future. *Science*, 248:44–48.

Zhang, X. (1990). *Exploration on protein structures: Representation and prediction*. PhD thesis, Dept. of Computer Sciences, Brandeis Univ., Waltham, MA.