Machine Learning, 11, 195–215 (1993)
© 1993 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

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

RICHARD MACLIN MACLIN@CS.WISC.EDU
JUDE W. SHAVLIK SHAVLIK@CS.WISC.EDU
Computer Sciences Department, University of Wisconsin, 1210 W. Dayton St., Madison, WI 53706

Abstract. This article describes a connectionist method for refining algorithms represented as generalized finite-state automata. The method translates the rule-like knowledge in an automaton into a corresponding artificial neural network, and then refines the reformulated automaton by applying backpropagation to a set of examples. This technique for translating an automaton into a network extends the KBANN algorithm, 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, FSKBANN is used to improve the Chou–Fasman algorithm, a method for predicting how globular proteins fold. Empirical evidence shows that the multistrategy approach of FSKBANN leads to a statistically-significantly, more accurate solution than both the original Chou–Fasman algorithm and a neural network trained using the standard approach. Extensive statistics report the types of errors made by the Chou–Fasman algorithm, the standard neural network, and the FSKBANN network.

Keywords. Multistrategy learning, theory refinement, neural networks, finite-state automata, protein folding, Chou–Fasman algorithm

1. Introduction

As machine learning has been applied to complex real-world problems, many researchers have found themselves turning to systems that combine knowledge from multiple sources. A standard approach is to incorporate existing knowledge about a domain into an empirical learning system, so as to produce a more accurate solution. Artificial neural networks (ANNs) have been shown to be a powerful technique for empirical learning, but until recently ANNs were largely unable to take advantage of existing problem-specific knowledge. This article describes an extension to the KBANN system (Towell, Shavlik, & Noordeweir, 1990), a connectionist system that refines symbolic domain knowledge. The extended system, called Finite-State KBANN (FSKBANN), translates domain theories that use state information, represented as generalized finite-state automata (FSAs) (Hopcroft & Ullman, 1979), into neural networks. The system then refines these networks using backpropagation (Rumelhart, Hinton, & Williams, 1986) with a set of examples.

The application of KBANN to the domain of gene recognition (Towell et al., 1990; Noordeweir, Towell, & Shavlik, 1991) showed that domain theories refined by neural networks can be more accurate than both the unrefined domain knowledge and neural networks trained in the standard manner. That work demonstrates the promise of a multistrategy approach based on the combination of a symbolic representation of rules with the numeric representation inherent in neural networks. By allowing domain theories to express *state* information, FSKBANN greatly extends the applicability of the KBANN approach. Researchers outside of machine learning generally publish algorithms rather than the sets of rules that machine learning researchers refer to as domain theories. Many of these algorithms maintain some sense of state, so this extension makes it easier to use machine learning to refine existing "real-world" knowledge. We test our extended system by refining the Chou-Fasman algorithm (Chou & Fasman, 1978) 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 or not the light is on. The rules introduced to solve this problem can therefore take into account the state of the problem—rules to turn on the light would only be considered when the state indicated that the light was off. In this style of problem solving, the problem is not solved in one step, but instead as a series of actions, each leading to a new state, that leads to the goal state (turning on the light, navigating to the couch, etc.).

The protein-folding problem is an open problem that is becoming increasingly critical as the Human Genome Project (Watson, 1990) proceeds. The Chou-Fasman algorithm is the focus of this article because it is one of the best-known and widely used algorithms in the field. The protein-folding problem is also of interest because a number of machine learning techniques are currently being applied to this problem, including neural networks (Holley & Karplus, 1989; Qian & Sejnowski, 1988), inductive logic programming (Muggleton & Feng, 1991), case-based reasoning (Cost & Salzberg, 1993), and multistrategy learning (Zhang, Mesirov & Waltz, 1992). Our work shows that a multistrategy approach combining the Chou-Fasman algorithm with a neural network produces a more accurate result than either method alone.

This article presents and empirically analyzes the FSKBANN approach for problem solving in domains where prior state-based knowledge exists. Section 2 presents the basic KBANN algorithm and discusses the extension of the algorithm to handle state information. Section 3 defines the protein-folding problem and reviews previous approaches taken. Experiments that investigate the utility of FSKBANN for this problem follow section 3.

2. Finite-state KBANN

Before describing FSKBANN, we review the basic 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 about a problem.

KBANN takes as input a set of propositional, non-recursive rules, such as those shown in figure 1a. Figure 1b shows the dependencies among the rules. A dependency is indicated by a link between two propositions—arcs show conjunctive dependencies. From the set of dependencies, it is easy to map the rules to a network by replacing each proposition with a corresponding unit (and adding units where conjunctions are combined into disjunctions). Figure 1c displays the resulting network. This network has the same behavior as

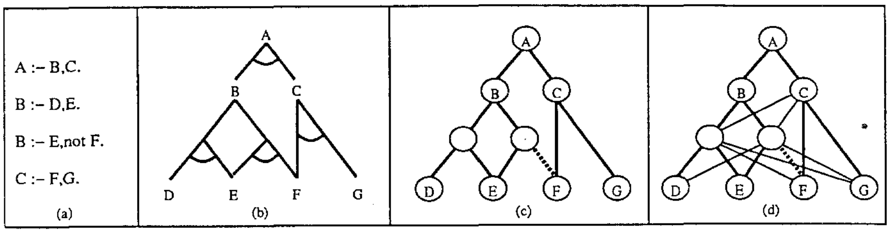

Figure 1. Sample of KBANN: (a) a set of rules; (b) dependencies among the rules; (c) the corresponding neural network; and (d) elaborated network after KBANN connects each unit to unconnected units at the next lower level.

the rules for each possible 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, the present article extends KBANN to translate domain theories represented as *generalized* FSA.[1] The main extension for FSKBANN is the type of network onto which the domain theory is mapped. FSKBANN maps domain theories onto a variant of simple recurrent networks (Jordan, 1986; Elman, 1990), where a subset of the network output is copied back as input to the network in the next step. This copied output represents the current state calculated by the network and can be used in calculating the succeeding state.

Table 1 describes the class of problem solvers to which FSKBANN is applicable. Consider a problem solver that determines the next state on the basis of both externally provided input and its internal representation of the current state of the problem solution. The externally provided input may involve a description of the initial state or the changing measurements of sensors (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 to choose its internal representation of the next state of the problem solution. This process repeats until a termination condition is met (e.g., a goal state is reached).

The description in table 1 is essentially a definition of state-based problem solving. The contribution of FSKBANN is a mechanism for using neural networks to improve a state-

Table 1. The type of problem solving to which FSKBANN is applicable.

```
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  =   results specific to this problem solving step
                                 +
                           next internal representation of the problem-solving state
Until a Termination Criterion is met.
```

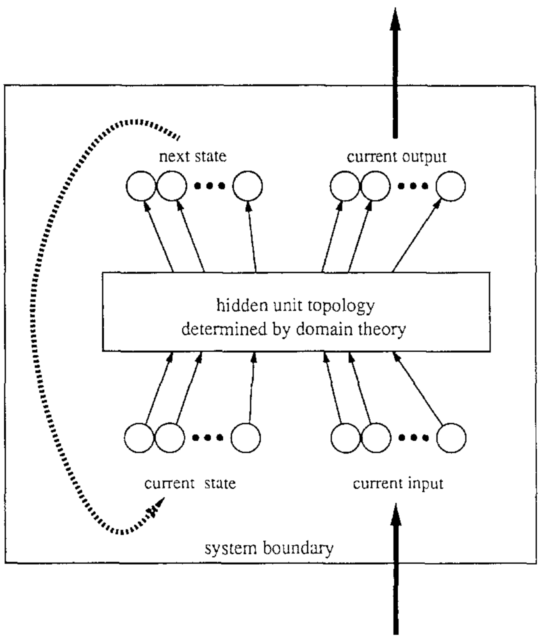

Figure 2. A schematic view of an FSKBANN network.

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 diagram of the type of network produced by FSKBANN.

The FSKBANN approach requires that the user provide sample input/output pairs that can be used to train the network with backpropagation. It also requires that the inputs and the outputs be of bounded size, which means the domain theory can only store a finite amount of state information. Finally, FSKBANN requires the domain theory to be propositional, since no good mechanism exists for dealing with predicate calculus variables in neural networks. Despite these current limitations, it is likely that many "real-world" algorithms can be adequately represented in this finite-state framework. It is an open empirical question whether and where our neural-based approach generalizes better than inductive logic programming approaches (Muggleton, 1992; Quinlan, 1990), which learn a larger class of languages.

3. The protein-folding problem

This section describes the protein-folding problem, an open problem in the field of molecular biology that is being examined by researchers in both the biological and machine learning communities. Following this description is an outline of a standard algorithm used by the biological community to solve this problem, along with a description of how this algorithm can be mapped into section 2's 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 given protein constitutes the *primary* structure of the protein. Once a protein forms, it folds into a three-dimensional shape known as the protein's *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 in the laboratory is costly and time consuming. An alternative solution is to predict the *secondary* structure of a protein as an approximation. The secondary structure of 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 different types of structures: (1) α-helix regions, (2) β-strand regions, and (3) random coils (all other regions). Figure 3 shows the tertiary structure of a protein and how the shape is divided into regions of secondary structure. For our purposes, the secondary structure of a protein is 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 predictive accuracies of some standard algorithms from the biological literature for solving the secondary-structure problem (Chou & Fasman, 1978; Garnier & Robson, 1989; Lim, 1974). In the data sets used to test the algorithms, 54%–55% of the amino acids in the proteins are part of coil structures, so 54% accuracy can be achieved trivially by always predicting coil. It is important to note that many biological researchers believe that algorithms that only take into account local information can only achieve limited accuracy (Wilson et al., 1985), generally believed to be at most 80%–90% (Cohen & Presnell, personal communication).

Another approach to the secondary-structure problem is to use a learning method such as neural networks (Holley & Karplus, 1989; Qian & Sejnowski, 1988). The neural networks

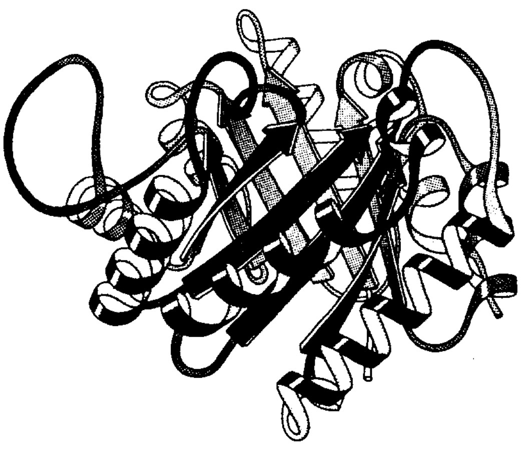

Figure 3. *Ribbon* drawing of the three-dimensional structure of a protein (reprinted from Richardson & Richardson, 1989). The areas resembling springs are α-helix structures, the flat arrows represent β-strands, and the remaining regions are random coils.

Table 2. Primary and secondary structures of a sample protein.

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

Table 3. Accuracies of various (non-learning) prediction algorithms.

Method	Accuracy	Comments
Chou & Fasman (1978)	58%	Data from Qian & Sejnowski (1988)
Lim (1974)	50%	From Nishikawa (1983)
Garnier & Robson (1989)	58%	Data from Qian & Sejnowski (1988)

in these efforts have as input a *window* of amino acids consisting of the central amino acid being predicted, plus some number of the amino acids before and after it in the sequence (similar to NETTALK networks; see Sejnowski & Rosenberg, 1987). The output of the network is the secondary structure for the central amino acid. Figure 4 shows the general structure of this type of network; table 4 presents results from these studies.

Our approach is to combine the knowledge from biological methods with a neural learning method in the hopes of achieving a better solution. We chose the Chou–Fasman algorithm (1978) as our domain theory because it is widely used. The Chou–Fasman approach is to find amino acids that are likely to be part of α-helix and β-strand regions, and then to extend these predictions to neighboring amino acids. Figure 5 provides a schematic overview of the algorithm. The first step of the process is to find *nucleation sites*. Nucleation sites are amino acids that are likely to be part of α-helix or β-strand structures, based on their neighbors and according to the conformation probabilities and rules reported by Chou and Fasman. From these sites, their algorithm extends the structure both forward and

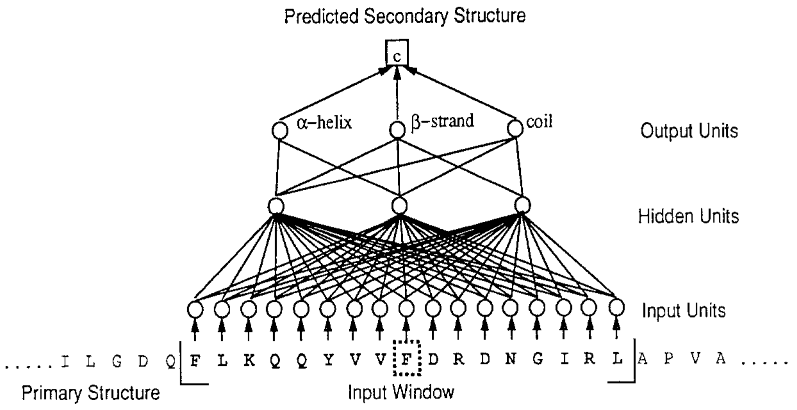
Predicted Secondary Structure
c
α–helix
β–strand
coil
Output Units
Hidden Units
Input Units
.....I L G D Q F L K Q Q Y V V F D R D N G I R L A P V A .....
Primary Structure
Input Window

Figure 4. Neural network architecture used by Qian and Sejnowski (1988).

Table 4. Neural network results for the secondary-structure prediction task.

Method	Testset Accuracy	Number of hidden units	Window size
Holley & Karplus (1989)	63.2%	2	17
Qian & Sejnowski (1988)	62.7%	40	13

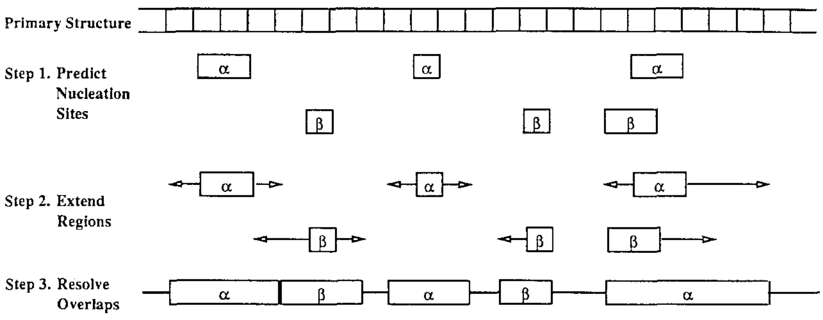

Figure 5. Steps of the Chou–Fasman algorithm.

backward along the protein, as long as the probability of being part of a α-helix structure remains sufficiently high. After predicting both α-helix and β-strand regions, the Chou–Fasman algorithm compares the relative probabilities of regions to resolve predictions that overlap.

This algorithm cannot be easily represented using propositional rules, since the prediction for an amino acid may depend on the predictions for its neighbors. However, one can represent the algorithm with a generalized FSA (see figure 6). The start state of the FSA is *coil*. To make predictions for a protein, the protein is scanned[2]; the input at each step

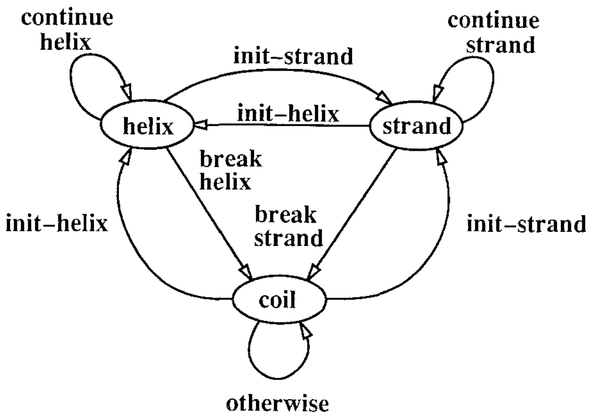

Figure 6. The finite-state automaton interpretation of the Chou–Fasman algorithm.

is the amino acid being classified plus its neighbors (a window). Each prediction is based on the last prediction (or state) and depends on which "transitions" are valid for the current input window. The notion of transition in figure 6's automata is complex. Each transition is actually a set of rules dependent on the input window and the current state. For example, in the FSA there is transition from state *helix* to state *coil* on *break-helix*. This is represented by the rule:

$$coil_i \leftarrow helix_{i-1} \wedge break\text{-}helix.$$

The term *break-helix* is not an input to the network, but is instead itself a predicate derived from the input. *Break-helix* is defined in terms of two other rules:

$$break\text{-}helix \leftarrow helix\text{-}break@0 \wedge helix\text{-}break@1.$$
$$break\text{-}helix \leftarrow helix\text{-}break@0 \wedge helix\text{-}indiff@1.$$

The terms *helix-break@0*, *helix-break@1*, and *helix-indiff@1* are further defined by other rules (see the appendix for the full set of rules defining the FSA).

Table 5 shows how this algorithm fits into the FSKBANN framework of table 1. The resulting network appears in figure 7 (although not shown, recall that the network also contains low-weighted links into hidden units). This network is similar to the one in figure 4, but with two major differences. First, the input to the network includes a copy of the past prediction made by the network—this represents the state of the network. Second, the topology of the hidden units is determined by the rules implementing the Chou-Fasman algorithm.

4. Experimental study

This section reports several experiments on the protein-folding problem that evaluate FSKBANN. They demonstrate that 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. This section also contains an in-depth empirical analysis of the strengths and weaknesses of the different methods.

4.1. Experimental details

The experiments use the data set from Qian and Sejnowski (1988). Their data set consists of 128 segments from 106 proteins with a total of 21,623 amino acids, for an average length

Table 5. Mapping the Chou–Fasman algorithm into the FSKBANN framework.

Domain theory	=	the Chou–Fasma algorithm
Goal	=	assign a secondary structure to each amino acid
External input	=	a sliding window of amino acids
Current state	=	the predicted secondary structure for the previous amino acid
Results	=	the predicted secondary structure for the current amino acid
Next state	=	*ditto*

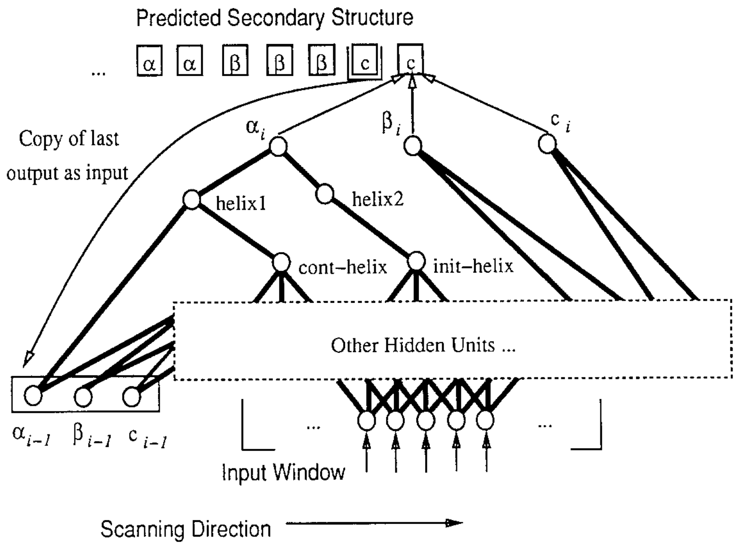

Figure 7. General neural-network architecture used to represent the Chou-Fasman algorithm.

of 169 amino acids per segment. Of these amino acids, 54.5% are part of coil structures, 25.2% are part of α-helix structures, 20.3% are part of β-strand structures. We randomly divided the proteins ten times into disjoint training and test sets, which contained two thirds (85 proteins) and one third (43 proteins) of the original proteins, respectively.

We use backpropagation to train the neural networks in the two approaches (FSKBANN and standard ANNs). Training is terminated using *patience*[3] as a stopping criterion (Fahlman & Lebiere, 1990). During training, we divided the proteins used for training into two portions—a training set and a *tuning* set. We employ 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.

FSKBANN randomly chooses a "representative" tuning set; it considers a tuning set to be representative if the percentages of each type of structure (α, β, and coil) in the tuning set roughly approximate the percentages of all the training proteins. Note that the system does not consider the *testing* set when comparing the percentages. Through empirical testing (not reported here), we found that a tuning set size of five proteins achieves the best results for both FSKBANN and ANNs. It is important to note that this style of training is different than the one reported by Qian and Sejnowski. They tested their network periodically and retained the network that achieved the highest accuracy for the test set.

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 methodology 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 article's experiments, we use 28 hidden units in our standard ANNs. This has the added advantage that the FSKBANN and standard networks contain the same number of hidden units.

4.2. Results and analysis

Table 6 contains results averaged over the 10 test sets. The statistics reported are the percent accuracy overall, the percent accuracy by secondary structure, and the correlation coefficients for each structure.[4] 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 fairly small (only 1.6 percentage points), but this number is somewhat misleading. The correleation 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 still does fairly well in overall accuracy is that it predicts a large number of coil structures (the largest class) and does very 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 approximately the same accuracy for β-strands, but the correlation coefficients demonstrate that the Chou–Fasman algorithm achieves this accuracy by predicting a much larger number of β-strands.

Also shown in table 6 are results for ANNs that included state information—networks similar to Qian and Sejnowski's, but ones in which 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.

To evaluate the usefulness of the domain theory as a function of the number of training examples and to allow us to estimate the value of collecting more proteins, we performed a second series of tests. We divided each of the training sets into four subsets: the first contained the first 10 of the 85 proteins; the second contained the first 25; the third contained the first 50; and the fourth had all 85 training proteins. This process produced 40 training sets. Each of these training sets was then used to train both the FSKBANN and ANN networks. Figure 8 contains the results of these tests. FSKBANN shows a gain in accuracy for each training set size (statistically significant at the 5% level, i.e., with 95% confidence).

The results in figure 8 demonstrate two interesting trends. First, the FSKBANN networks do better no matter how large the training set, and second, the shape of the curve indicates that accuracy might continue to increase if more proteins were used for training. The one anomaly for this curve is that the gain in accuracy for the 10 training proteins is not very

Table 6. Results from different prediction methods.

	Testset accuracy				Correlation coefficients		
Method	Total	Helix	Strand	Coil	Helix	Strand	Coil
Chou–Fasman	57.3%	31.7%	36.9%	76.1%	0.24	0.23	0.26
ANN	61.8	43.6	18.6	86.3	0.35	0.25	0.31
FSKBANN	63.4	45.9	35.1	81.9	0.37	0.33	0.35
ANN (w/state)	61.7	39.2	24.2	86.0	0.32	0.28	0.31

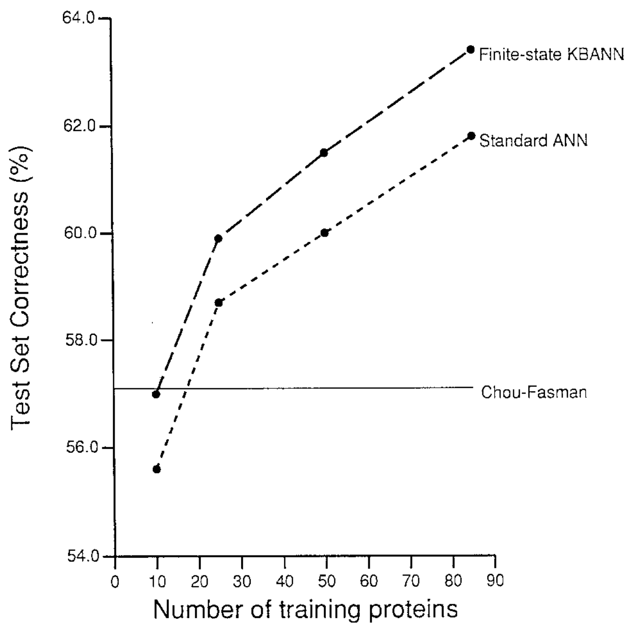

Figure 8. Percent correctness on test proteins as a function of training-set size.

large. One would expect that when the number of training instances is very small, the domain knowledge would be a big advantage. The problem here is that for a small training set it is possible to obtain random sets of proteins that are not very indicative of the overall population. Individual proteins generally do not reflect the overall distribution of secondary structures for the whole population; many proteins have large numbers of α-helix regions and almost no β-sheets, while others have large numbers of β-sheet regions and almost no α-helices. Thus in trying to learn to predict a very skewed population, the network may produce a poor solution. This problem is mitigated as more proteins are introduced, causing the training population to more closely match the overall population.

Finally, to analyze the detailed performance of the various approaches, we gathered a number of additional statistics concerning the FSkbann, ANN, and Chou-Fasman solutions. These statistics analyze the results in terms of *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 exactly the structure within which each amino acid lies. Consider the two predictions in figure 9 (adapted from Cohen et al., 1991). The first prediction completely misses the third α-helix region, so it has four errors. The second prediction is slightly skewed for each α-helix region and ends up having six errors, even though it appears to be a better answer. The statistics we have gathered try to assess how well each solution does on predicting α-helix regions (table 7) and β-strand regions (table 8).

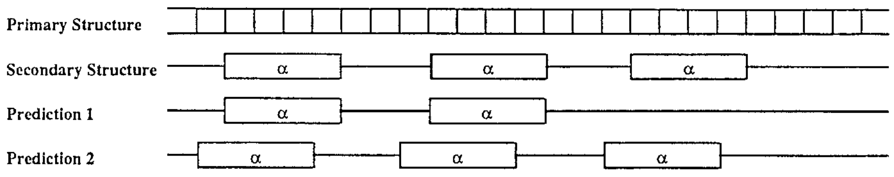

Figure 9. Two possible predictions for secondary structure.

Table 7. Region-oriented statistics for α-helix prediction.

Occurrence		Description	FSkbann	ANN	Chou–Fasman
Actual	α-helix	Average length of an actual helix region (number of regions)	10.17 (1825)	10.17 (1825)	10.17 (1825)
Predicted	α-helix	Average length of predicted helix region (number of regions)	8.52 (1774)	7.79 (2067)	8.00 (1491)
Actual Predicted	α-helix α-helix	Percentage of time an actual helix region is overlapped by a predicted helix region (length of overlap)	67% (6.99)	70% (6.34)	56% (5.76)
Actual Predicted	other α-helix	Percentage of time a predicted helix region does not overlap an actual helix region	34%	39%	36%

Table 8. Region-oriented statistics for β-strand prediction.

Occurrence		Description	FSkbann	ANN	Chou–Fasman
Actual	β-strand	Average length of an actual strand region (number of regions)	5.00 (3015)	5.00 (3015)	5.00 (3015)
Predicted	β-strand	Average length of predicted strand region (number of regions)	3.80 (2545)	2.83 (1673)	6.02 (2339)
Actual Predicted	β-strand β-strand	Percentage of time an actual strand region is overlapped by a predicted strand region (length of overlap)	54% (3.23)	35% (2.65)	46% (4.01)
Actual Predicted	other β-strand	Percentage of time a predicted strand region does not overlap an actual strand region	37%	37%	44%

Table 7 and table 8 give a picture of the strengths and weakness of each approach. Table 7 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 7 also indicates that FSKBANN and ANNs more accurately predict the occurrence of regions than Chou–Fasman does.

Table 8 demonstrates that FSKBANN's predictions overlap a much higher percentage of actual β-strand regions than either the Chou–Fasman algorithm or ANNs alone. The overall accuracy for β-strand predictions is approximately the same for FSKBANN and the Chou–Fasman method, because the length of overlap for the Chou–Fasman method is much longer than for FSKBANN (at the cost of predicting much longer regions). The ANN networks do extremely poorly at 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 that more work needs to be done on developing methods of evaluating solution quality. A simple position-by-position count of correct predictions does not capture adequately the desired behavior. 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 of using neural learning to refine an existing algorithm can be worthwhile.

5. 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, because training will start from a "better" location. Information in tables 7 and 8 indicate the weaknesses of the present system. With this knowledge, we plan to develop domain-theory extensions addressing these weaknesses by studying the biological literature.

A second method of augmenting the knowledge is to use a more complex encoding scheme. At present each amino acid is represented by a single input unit. Hunter (1991) suggests a more complex encoding scheme that encodes a number of properties of each of the amino acids. Our preliminary tests with this encoding scheme on standard neural networks showed promising results. More recent domain theories (Garnier & Robson, 1989; Prevelige & Fasman, 1989) also include knowledge about a fourth type of secondary structure, β-turns, which in our data set are classified as random coils. This knowledge can be added to the present networks as a partial domain theory for coils, or in networks trying to predict the four different classes.

One interesting property of FSKBANN's networks is that the magnitude of each output is correlated with how accurate the prediction is. We plan to use this information in a more complex method: instead of predicting all of the protein's structure in one scan, we will predict only the strongest activated areas first, and then will feed these predictions back into the network for the next scan. In the first pass over the protein, the system could mark those amino acids with the largest predictions, and then these could be used in predicting for the next step. This is different from the existing method in that during the second pass the system would not only know the structure for the previous amino acid, but also might know what the structure is for the next amino acid, the amino acid two positions ahead, etc.

A basic problem with the KBANN approach is extracting information in a human-readable form from trained networks (Giles et al., 1992; Towell & Shavlik, 1992). We plan to address rule extraction for the augmented networks of FSKBANN by extending the existing method for extracting rules from KBANN networks (Towell & Shavlik, 1992). Our extension will extract refined FSAs rather than rules.

Jacobs et al. (1991) proposed a method for learning how to combine knowledge from a number of neural networks to produce a better solution. Since a number of different approaches to protein structure prediction have been investigated, we are working on a method to combine these strategies into a single prediction method. The combined solution will aggregate predictions from a number of different neural networks, plus the output of other machine learning approaches and biological approaches.

Another important area of focus is applying FSKBANN in other domains. We plan to apply FSKBANN to problems in molecular biology, such as the splice-junction problem (Noordewier et al., 1991), where the states of the network are *intron* and *exon*. Also of interest is evaluating this approach for problems in other fields, such as natural language (as in Elman, 1990). The task could involve learning to recognize simple sentences involving a simple (regular) grammar where some information about the grammar is known, but some is missing or incorrect.

6. Related research

Our work shares similarities with research in three areas: algorithms for predicting protein structure, neural networks that use state information, and systems that combine strategies for solving problems.

6.1. Methods of predicting protein secondary structure

There have been a number of algorithms proposed for predicting protein secondary structure. These can loosely be divided into those that use biological knowledge (and are non-learning methods) and those that use a learning mechanism.

6.1.1. Non-learning methods

The three most widely used approaches in the biological literature (Fasman, 1989) for predicting protein secondary structure are the Chou-Fasman (Chou & Fasman, 1978; Prevelige & Fasman, 1989), Robson (Garnier & Robson, 1989; Robson & Suzuki, 1976), and Lim (1974) algorithms. The Robson and Suzuki (1976) and the later GorII and GorIII (Garnier & Robson, 1989) solutions are based on information theory. These approaches, like neural networks, base prediction on a window of information around a central amino acid (from position −8 to +8 around the central amino acid). For every window position, the Robson algorithm determines the relevance of the amino acid for predicting each type of secondary structure. The computerized versions of the Chou–Fasman and Robson techniques that we implemented and tested on the Qian and Sejnowski test data exhibit 58% accuracy (see table 3). The Lim (1974) method is the only one that tries to account for long-range interactions; it uses a stereochemical theory of the secondary structure of globular proteins. Later solutions (Garnier & Robson, 1989; Prevelige & Fasman, 1989) include theories of a fourth type of secondary structure, β-turns, which usually are classified as coils. The main advantage of FSKBANN over these algorithms is that while FSKBANN contains biological information, it also has a mechanism for learning.

6.1.2. Learning methods

A number of investigators have used learning algorithms and sample folded proteins to try to predict secondary structure of new proteins. Holley and Karplus (1989) and Qian and Sejnowski (1988) use simple one-hidden-layer neural networks to try to predict secondary structure. Both studies focus on varying the hidden unit size and window size, achieving very different results (as shown in table 4) for these parameters, though both report test-set accuracies around 63%. Qian and Sejnowski also use a cascaded architecture, which produces a 1.6-percentage-point improvement in accuracy over their single-network results. Stolorz, Lapedes, and Xia (1991) use a perceptron architecture to evaluate a different error function, *mutual information*, which produces a one-percentage-point gain in accuracy over a standard perceptron. The interesting thing about the Stolorz measure is that it improves helix and strand prediction at the expense of coil prediction, a desirable effect, since coil (making up 54% of the training data) tends to be overpredicted in many other neural-network techniques.

Zhang et al. (1992) also uses machine learning. This method combines information from a statistical technique, a memory-based reasoning algorithm, and a neural network. Zhang and co-workers divide the training set into halves, where each of the three components is trained on one half of the training set. Further training is done using the other half of the training set to learn to combine results from the three different components, using a second neural network. The best results they report are 66.4% for a training set of 96 proteins (Zhang et al. (1992).

Another learning technique applied to this problem is the nearest-neighbor algorithm PEBLS (Cost & Salzberg, 1993). These authors report accuracy of approximately 64% for a training set similar in size to the one we used. Work by Muggleton and King (1991) applying inductive logic programming produced test-set results of 81% accuracy for proteins containing only α-helix and coil regions. Kneller, Cohen, and Langridge (1990) produced similar results (79% accuracy) on proteins consisting of α-helices and coils. Our domain theory, trained to solve a similar problem (predicting either α-helix or not), also showed approximately 81% accuracy.

The major difference between these learning approaches and FSKBANN is that only FSKBANN incorporates a complete algorithm. FSKBANN also differs from the above approaches in that the neural networks used in our studies incorporate state information.

6.2. Methods of representing state information in neural networks

Several researchers have proposed neural-network architectures for incorporating information about state. The idea of retaining a state or context across training patterns occurs primarily in work addressing natural language problems (Cleeremans, Servan-Schreiber, & McClelland, 1989; Elman, 1990). These approaches provide a mechanism for preserving one or more of the past activations of some units to use in processing of the next input.

Jordan (1986) and Elman (1990) introduced the particular recurrent network topology we use in FSKBANN. Their networks have a set of hidden units called *context units* that preserve the state of the network. At each time step, the previous value of the context units are copied back as input to the system. These networks allow for the possibility of keeping multiple past contexts as input to the system.

The idea of using the type of network introduced by Jordan to represent a finite-state automaton was discussed by Cleeremans et al. (1989). They show that this type of network can perfectly learn to recognize a grammar derived from a finite-state automaton. Giles et al. (1992) used a more complex recurrent network both to learn an FSA and to extract the learned FSA. The major difference between our research and that of Cleeremans et al. and Giles et al. is that we focus on using an initial domain theory expressed as a finite-state automaton, rather than attempting to learn it from scratch.

6.3. Methods of multistrategy learning

A number of researchers have recently addressed the problem of blending different strategies to produce more effective learners. Ourston and Mooney's (1990) EITHER system uses a domain theory to focus the corrections that an inductive learning system performs. Tecuci (this issue) describes a system that includes a number of different mechanisms such as induction, deduction, analogy, and abduction to correct a domain theory. In Pazzani's (this issue) Theory-Driven Learning system, the search for rules to explain incorrect examples is constrained by regularities observed between rules in the domain theory. Saitta and Botta (this issue) developed a system that combines abductive learning based on a domain model with inductive learning based on sample instances. The major difference between the above work and FSKBANN is that the above systems only use symbolic reasoning to learn. Each of these systems works directly on rules in some form. FSKBANN allows the integration of a symbolic representation with neural learning to take advantage of the generality and power of this type of learning.

7. Conclusion

This article presents and evaluates FSKBANN, a system that provides a mechanism for combining the knowledge from domain theories represented as generalized finite-state automata into neural networks. These networks can be further trained, using backpropagation, to refine the initial domain knowledge. The extension of KBANN to domain theories that include knowledge about state significantly enhances the power of the KBANN approach; 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, a task that is becoming a "challenge problem" in the machine learning and computational biology communities. The FSKBANN multistrategy approach of combining domain knowledge with a neural network proved to be more accurate than either a standard neural network approach or the non-learning Chou–Fasman algorithm. The FSKBANN solution proved even more effective when considered in terms of how well it does for each class of secondary structure.

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

Appendix. The Chou–Fasman domain theory

The Chou–Fasman algorithm (1978) involves three activities: (1) recognizing nucleation sites, (2) extending sites, and (3) resolving overlapping predictions. This appendix provides more details of these three steps and describes the representation of their algorithm as a collection of rules.

To recognize nucleation sites, Chou and Fasman assign two *conformation* values to each of the 20 amino acids. The conformation values represent how likely an amino acid is to be part of either a helix or strand structure, with higher values being more likely. They also group the amino acids into classes of similar conformation value. The classes for helix are *formers*, *high-indifferent*, *indifferent*, and *breakers*; those for strand are *formers*, *indifferent*, and *breakers*. Table 9 defines the values for the various types of breakers and formers.

Table 10 contains the rules that represent the Chou–Fasman algorithm; x@N is true if x is the amino acid N positions from the one whose secondary structure the algorithm is predicting. The rules predict an α-helix nucleation site if for some consecutive set of six amino acids, at least four are helix formers and fewer than two are helix breakers. (Two helix high-indifferent amino acids count as a helix former.) A rule to determine if a location is a nucleation site simply adds the helix-former and helix-breaker values for a window six amino acids wide. If the totals are greater than four and less than two, respectively, the rule predicts a helix nucleation site (proposition init-helix in the rules). Nucleation of β-strands is similar to α-helix nucleation, except that the window is only five amino acids wide and a strand nucleation site is predicted if there are at least three strand formers and fewer than two strand breakers.

The third step of the algorithm—resolving overlaps—is the reason we use the numbers in table 9 rather than making the formers and breakers Boolean properties. Chou and Fasman suggest that the conformation values of regions be compared to resolve overlaps. This is done in FSKBANN's networks by weighting the links from various amino acids according to the numbers in table 9. For example, a combination of four alanines (A's) will produce a higher activation of the init-helix unit than a combination of four phenylalanines (F's).

Table 9. Former and breaker values‡ for the amino acids.

helix-former(E)	= 1.37	helix-former(A)	= 1.29	helix-former(L)	= 1.20
helix-former(H)	= 1.11	helix-former(M)	= 1.07	helix-former(Q)	= 1.04
helix-former(W)	= 1.02	helix-former(V)	= 1.02	helix-former(F)	= 1.00
helix-former(K)	= 0.54	helix-former(I)	= 0.50		
helix-former(*others*)	= 0.00				
helix-breaker(N)	= 1.00	helix-breaker(Y)	= 1.20	helix-breaker(P)	= 1.24
helix-breaker(G)	= 1.38				
helix-breaker(*others*)	= 0.00				
strand-former(M)	= 1.40	strand-former(V)	= 1.39	strand-former(I)	= 1.34
strand-former(C)	= 1.09	strand-former(Y)	= 1.08	strand-former(F)	= 1.07
strand-former(Q)	= 1.03	strand-former(L)	= 1.02	strand-former(T)	= 1.01
strand-former(W)	= 1.00				
strand-former(*others*)	= 0.00				
strand-breaker(K)	= 1.00	strand-breaker(S)	= 1.03	strand-breaker(H)	= 1.04
strand-breaker(N)	= 1.14	strand-breaker(P)	= 1.19	strand-breaker(E)	= 2.00
strand-breaker(*others*)	= 0.00				

‡We produced these values using the tables reported by Chou and Fasman (1978, p. 51). We normalized the values for formers by dividing the conformation value of the given former by the conformation value of the weakest former. So for example, the helix former value of alanine (A) is 1.29, since the helix conformation value of alanine is 1.45 and the conformation value of the weakest helix former phenylalanine (F) is 1.12. Breaker values work similarly except that the value used to calculate the breaker value is the multiplicative inverse of the conformation value.

We did not directly use the values of Chou and Fasman for two reasons. One, we wanted smaller values in order to decrease the number of times that three very strong helix-formers would add up to more than 4 (and similarly for strands). Two, breaker conformation values tend to be numbers between 0 and 1 with the stronger breakers being close to 0. We wanted the breaker value to be larger the stronger the breaker, so we used the inverse of the breaker's conformation value (restricting the result to not exceed 2).

The Chou–Fasman algorithm continues to predict α-helices as long as the predicate `cont-helix` is true. The rules define `cont-helix` mostly in terms of helix-breaking rules—a helix continues as long as a break region is not encountered. An α-helix break region occurs when a helix-breaker amino acid is immediately followed by either another helix-breaker or a helix-indifferent amino acid. A helix is also broken when encountering the amino acid proline (`P`). The process of extending β-strand structures works similarly. The algorithm predicts coil as the default.

Acknowledgments

This research was partially supported by National Science Foundation Grant IRI-9002413 and Office of Naval Research Grant N00014-90-J-1941. The authors would like to thank Terrence Sejnowski for providing the protein data used in testing. The domain theory and data used in testing are available by anonymous ftp from the University of California-Irvine archive maintained by David Aha and Patrick Murphy.

Table 10. The Chou–Fasman algorithm expressed as inference rules.

Rules for recognizing nucleation sites

$$\textit{init-helix} \leftarrow \left[\sum_{position=0}^{5} \textit{helix-former}\ (\textit{amino-acid@position}) \right] > 4$$

$$\wedge \left[\sum_{position=0}^{5} \textit{helix-breaker}\ (\textit{amino-acid@position}) \right] < 2$$

$$\textit{init-strand} \leftarrow \left[\sum_{position=0}^{4} \textit{strand-former}\ (\textit{amino-acid@position}) \right] > 3$$

$$\wedge \left[\sum_{position=0}^{4} \textit{strand-breaker}\ (\textit{amino-acid@position}) \right] < 2$$

Rules for pairs of amino acids that terminate helix structures

$\textit{helix-break@0} \leftarrow N@0 \vee Y@0 \vee P@0 \vee G@0$

$\textit{helix-break@1} \leftarrow N@1 \vee Y@1 \vee P@1 \vee G@1$

$\textit{helix-indiff@1} \leftarrow K@1 \vee I@1 \vee D@1 \vee T@1 \vee S@1 \vee R@1 \vee C@1$

$\textit{break-helix} \leftarrow \textit{helix-break@0} \wedge \textit{helix-break@1}$

$\textit{break-helix} \leftarrow \textit{helix-break@0} \wedge \textit{helix-indiff@1}$

Rules for pairs of amino acids that terminate strand structures

$\textit{strand-break@0} \leftarrow K@0 \vee S@0 \vee H@0 \vee N@0 \vee P@0 \vee E@0$

$\textit{strand-break@1} \leftarrow K@1 \vee S@1 \vee H@1 \vee N@1 \vee P@1 \vee E@1$

$\textit{strand-indiff@1} \leftarrow A@1 \vee R@1 \vee G@1 \vee D@1$

$\textit{break-strand} \leftarrow \textit{strand-break@0} \wedge \textit{strand-break@1}$

$\textit{break-strand} \leftarrow \textit{strand-break@0} \wedge \textit{strand-indiff@1}$

Rules for continuing structures

$\textit{cont-helix} \leftarrow \neg P@0 \wedge \neg \textit{break-helix}$

$\textit{cont-strand} \leftarrow \neg P@0 \wedge \neg E@0 \wedge \neg \textit{break-strand}$

Rules for predicting a-helix: either by nucleation or propagating from the last state

$\textit{helix}_i \leftarrow \textit{init-helix}$

$\textit{helix}_i \leftarrow \textit{helix}_{i-1} \wedge \textit{cont-helix}$

Rules for predicting β-strand: either by nucleation or propagating from the last state

$\textit{strand}_i \leftarrow \textit{init-strand}$

$\textit{strand}_i \leftarrow \textit{strand}_{i-1} \wedge \textit{cont-strand}$

Rules for predicting coil (the default)

$\textit{coil}_i \leftarrow \textit{helix}_{i-1} \wedge \textit{break-helix}$

$\textit{coil}_i \leftarrow \textit{strand}_{i-1} \wedge \textit{break-strand}$

$\textit{coil}_i \leftarrow \textit{coil}_{i-1}$

Notes

1. The notion of an 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.
2. FSKBANN actually scans each protein twice: from left-to-right and from right-to-left. It then sums the results to simulate extending nucleation sites in both directions.
3. The patience criterion states that training should continue until the error rate has not decreased for some number of training cycles. For this study we set the number of epochs to be four (a value determined by empirical testing).
4. The following formula defines the correlation coefficient for the secondary structure problem (Mathews, 1975):

$$C = \frac{P * N - F * M}{\sqrt{(P + F)(P + M)(N + F)(N + M)}}$$

where C is calculated for each structure separately, and P, N, F, and M are the number of true positives, true negatives, false positives, and misses for each structure, respectively.

References

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

Cleeremans, A., Servan-Schreiber, D., & McClelland, J. (1989). Finite state automata and simple recurrent networks. *Neural Computation,* (*3*), 372–381.

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

Cost, S., & Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning, 10(1),* 57–78.

Elman, J. (1990). Finding structure in time. *Cognitive Science, 14*(*2*), 179–211.

Fahlman, S., & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. Touretzky (Ed.), *Advances in neural information processing systems,* Vol. 2 (pp. 524-532). Denver, CO: Morgan Kaufmann.

Fasman, G. (1989). The development of the prediction of protein structure. In Fasman, G. (Ed.), *Prediction of protein structure and the principles of protein conformation.* New York: Plenum Press.

Garnier, J., & Robson, B. (1989). The GOR method for predicting secondary structures in proteins. In G. Fasman (Ed.), *Prediction of protein structure and the principles of protein conformation.* New York: Plenum Press.

Giles, C., Miller, C., Chen, D., Chen, H., Sun, G., & Lee, Y. (1992). Learning and extracting finite state automata with second-order recurrent neural network. *Neural Computation, 4,* 393–405.

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

Hopcroft, J., & Ullman, J. (1979). *Introduction to automata theory, languages, and computation.* Reading, MA: Addison Wesley.

Hunter, L. (1991). Representing amino acids with bitstrings. *Proceedings of the AAAI-91 Workshop on Artificial Intelligence Approaches to Classification and Pattern Recognition in Molecular Biology* (pp. 110–117). Anaheim, CA.

Jacobs, R., Jordan, M., Nowlan, S., & Hinton, G. (1991). Adaptive mixtures of local experts. *Neural Computation, 3(1),* 79–87.

Jordan, M. (1986). *Serial order: A parallel distributed processing approach* (Technical Report 8604). San Diego: University of California, Institute for Cognitive Science.

Kneller, D., Cohen, F., & Langridge, R. (1990). Improvements in protein secondary structure prediction by an enhanced neural network. *Journal of Molecular Biology, 214,* 171–182.

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

Mathews, B. (1975). Comparison of the predicted and observed secondary structure of T4 Phage Lysozyme. *Biochimica et Biophysica Acta*, *405*, 442–451.

Muggleton, S. (Ed.). (1992). *Inductive logic programming*. London: Academic Press.

Muggleton, S., & Feng, R. (1991). *Predicting protein secondary-structure using inductive logic programming*. (Technical Report). Glasgow, Scotland: Turing Institute.

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.

Noordewier, M., Towell, G., & Shavlik, J. (1991). Training knowledge-based neural networks to recognize genes in DNA sequences. In R. Lippmann, J. Moody, & D. Touretzky, (Eds.), *Advances in neural information processing systems*, Vol. 3. Denver, CO: Morgan Kaufmann.

Ourston, D., & Mooney, R. (1990). Changing the rules: A comprehensive approach to theory refinement. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 815–820). Boston, MA: MIT Press.

Pazzani, M. (1993). Learning causal patterns: Making a transition from data-driven to theory-driven learning. *Machine Learning*, *11* (this issue).

Prevelige, P. Jr., & Fasman, G. (1989). Chou–Fasman prediction of the secondary structure of proteins: The Chou–Fasman–Prevelige algorithm. In G. Fasman (Ed.), *Prediction of protein structure and the principles of protein conformation*. New York: Plenum Press.

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

Quinlan, J. (1990). Learning logical definitions from relations. *Machine Learning*, *5(3)*, 239–266.

Richardson, J., & Richardson, D. (1989). Principles and patterns of protein conformation. In G. Fasman (Ed.), *Prediction of protein structure and the principles of protein conformation*. New York: Plenum Press.

Robson, B., & Suzuki, E. (1976). Conformational properties of amino acid residues in globular proteins. *Journal of Molecular Biology*, *107*, 327–356.

Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning internal representations by error propagation. In D. Rumelhart, & J. McClelland (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition. Volume 1: Foundations* (pp. 318–363). Cambridge, MA: MIT Press.

Saitta, L., & Botta, M. (1993). Multistrategy learning and theory revision. *Machine Learning*, *11* (this issue).

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

Stolorz, P., Lapedes, A., & Xia, Y. (1991). Predicting protein secondary structure using neural net and statistical methods. *Journal of Molecular Biology*, *225*, 363–377.

Tecuci, G. (1993). Plausible justification trees: A framework for deep and dynamic integration of learning strategies. *Machine Learning*, *11* (this issue).

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

Towell, G. (1991). *Symbolic knowledge and neural networks: Insertion, refinement and extraction*. Doctoral dissertation, Department of Computer Science, University of Wisconsin, Madison.

Towell, G., & Shavlik, J. (1992). Interpretation of artificial neural networks: Mapping knowledge-based neural networks into rules. In R. Lippmann, J. Moody, & D. Touretzky (Eds.), *Advances in neural information processing systems*, Vol. 4. Denver, CO: Morgan Kaufmann.

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

Wilson, I., Haft, D., Getzoff, E., Tainer, J., Lerner, R., & Brenner, S. (1985). Identical short peptide sequences in unrelated proteins can have different conformations: A testing ground for theories of immune recognition. *Proceeding of the National Academy of Sciences (USA)*, *82*, 5255–5259.

Zhang, X., Mesirov, J. & Waltz, D. (1992). Hybrid system for protein secondary structure prediction. *Journal of Molecular Biology*, *225*, 1049–1063.

Received December 11, 1991
Accepted January 16, 1992
Final Manuscript February 28, 1992