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

Refining Domain Theories Expressed as Finite-State Automata

Richard Maclin **Jude W. Shavlik**
Computer Science Dept.
University of Wisconsin
Madison, WI 53706
email: maclin@cs.wisc.edu

Abstract

The KBANN system uses neural networks to refine domain theories. Currently, domain knowledge in KBANN is expressed as non-recursive, propositional rules. We extend KBANN to domain theories expressed as finite-state automata. We apply finite-state KBANN to the task of predicting how proteins fold, producing a small but statistically significant gain in accuracy over both a standard neural network approach and a non-learning algorithm from the biological literature. Our method shows promise at solving this and other real-world problems that can be described in terms of state-dependent decisions.

1 INTRODUCTION

Research in artificial neural networks (ANNs) has until recently largely ignored preexisting knowledge about the task at hand. One recent effort, the KBANN system [14], addresses this problem by using domain knowledge to select a promising configuration for a neural network. This approach uses the domain knowledge to determine the topology of the network and biases the network so that it will start with a "good" set of weights. This approach has been effective in complex domains, such as gene recognition [9, 14], even when the initial domain theory is not particularly correct. In this paper we discuss how KBANN's domain theory vocabulary is extended. Application to the problem of protein folding demonstrates the promise of this approach.

At present, the KBANN algorithm is limited in the types of domain theories that can be represented. Domain theories are written using simple non-recursive propositional rules. This limitation means that many domain theories are too complex for the standard KBANN algorithm. This research extends KBANN to include domain theories represented as finite-state automata. The result is that KBANN can be applied to more complex problems, including ones requiring state-dependent knowledge.

The sample task we examine is predicting the folding of globular proteins. There are a number of reasons we chose this domain. One, it is a real-world problem; at present no simple solution exists that produces highly accurate predictions. Two, there have been a number of attempts to apply standard neural network methods [4, 10], which can be used for comparison purposes. Finally, there are a number of domain theories [1, 7, 11], most notably the Chou-Fasman [1] method, which can not be naturally expressed in simple propositional rules, but which can be expressed as a finite-state automaton.

The next section presents the KBANN method and how it is extended for finite-state automata. This is followed by an experiment involving protein folding. The paper finishes with a short discussion and conclusions.

2 FINITE-STATE DOMAIN THEORIES

This section reviews the basic KBANN algorithm and presents our method for extending it to finite-state automata. We show examples of standard and finite-state KBANN.

2.1 STANDARD KBANN

Standard KBANN [14] takes a domain theory expressed as simple rules and translates it into a corresponding network with initial weights. For example, consider the pair of rules in Figure 1(a).

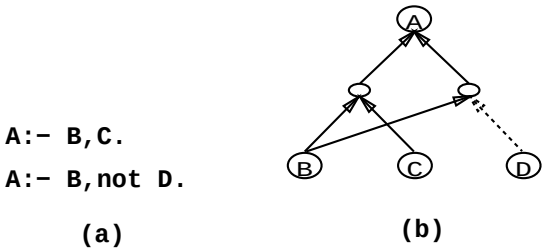


Figure 1: (a) A set of propositional rules and (b) a KBANN translation of these rules.

KBANN translates these rules into the network shown in Figure 1(b). Positive links are solid lines, negative links are dashed lines. B, C, and D are binary input units. The hidden units correspond to the rules; the first is active if B and C are active, the second is active if B is active and D is not active. Unit A is active if either hidden unit is active.

After the domain theory is translated into a network, each unit is connected to the unconnected units at the next lower level using a small weight link. These connections allow the system to learn new antecedents to rules that might not have been part of the original domain theory. The resulting network is trained using backpropagation [12]. Backpropagation refines the domain theory to correctly classify any training examples not already covered.

2.2 FINITE-STATE KBANN

A finite-state automaton (FSA) is represented by a set of nodes and arcs. Each node is a state. Each arc is a transition from one state to another. The FSA, beginning in its start state, scans a series of input values. For each input value, the automaton transitions to a new state by examining the arcs from the present state and following the arcs labeled with the input value. The problems with representing an FSA in KBANN are (1) how to represent the concepts of "scanning" a series of input values and (2) keeping track of the state.

A solution to problem 1 is to "scan" the input as done in the NETtalk system [13]. The input is a window of values around the central value, and the next input is obtained by advancing the window one place.

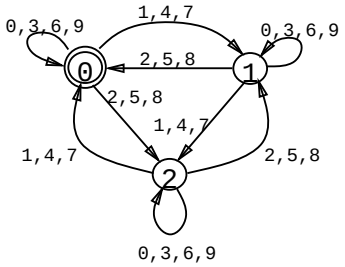
To solve problem 2, the current state of the system is an input to the network at each step, as suggested by Jordan [5] and Elman [3]. After each step, when the window advances, the state from the last step is copied back as the next state. This *copied-back* state makes translation of an FSA a simple extension to the KBANN method. Each arc in the FSA is viewed as a rule where the antecedents are the source state and the input value; the consequent is the destination state. This representation of an FSA in a neural network is very similar to that of Cleeremanns et al. [2].

2.3 A FINITE-STATE KBANN EXAMPLE

As an example of translating an FSA, consider the automaton in Figure 2(a) for recognizing numbers divisible by three. The FSA is translated into a set of rules with current state and input as antecedents and resulting state as consequent.

Finite-state KBANN translates the FSA from Figure 2(a) into rules; examples are shown in Figure 2(b). Applying KBANN to these rules produces the network shown in Figure 2(c). In this network the hidden units act like AND-gates and the output units like OR-gates.

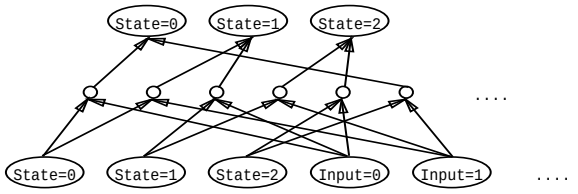
Note that the domain theory in Figure 2 is correct. However, in an imperfect domain theory, the transition rules may be incorrect or there may be missing transitions. Finite-state KBANN uses example sequences to correct these errors. Also, in finite-state KBANN we have generalized the notion of input to a window of input including not only the present input value, but also some number of input values before and after the present input.



(a) An FSA to recognize numbers divisible by three.

```
If      Present State = 0 And Current Input = 0
 Then   Next State = 0
If      Present State = 0 And Current Input = 1
 Then   Next State = 1
If      Present State = 1 And Current Input = 0
 Then   Next State = 1
...
```

(b) Rules derived from (a).



(c) KBANN translation of rules from (b).

Figure 2: Example of the translation process for an FSA.

3 A CASE STUDY IN THE DOMAIN OF PROTEIN FOLDING

This section defines the problem of predicting protein folding and examines standard neural network approaches to this problem. This is followed by a discussion of the Chou-Fasman domain theory used in our approach to this problem. We then report experiments performed to evaluate our approach.

3.1 PROTEIN FOLDING

Proteins are long strings of amino acids. There are twenty amino acids in all (represented by capital letters). The amino acids in a protein are the *primary* structure of the protein. Once a protein forms, it folds into a three-

dimensional shape; the location of the amino acids in this shape is the *tertiary* structure of the protein. Tertiary structure is important because the shape of the protein determines its function.

At present, few means exist to predict tertiary structure of a protein. There are X-ray crystallography and magnetic resonance processes; both are costly and time consuming. An alternative solution is to determine the *secondary* structure of proteins as an approximation to the tertiary structure. Secondary structure is a description of the local structure for each amino acid. A prevalent description of secondary structure divides a protein into three types of structures: (1) α-helix regions, (2) β-sheet regions, and (3) random coils (all other regions). A sample mapping between a protein's primary and secondary structure is shown in Figure 3. A common method used to solve this problem by biologists, the Chou-Fasman method [1], was shown to have a prediction accuracy of 58% for sample protein sequences [8].

Primary	....VYRNNFKSA....
Secondary	....ββccccαα....

Figure 3: Primary and secondary structures of a sample protein.

3.2 STANDARD NEURAL NETWORK APPROACHES TO PREDICTING SECONDARY STRUCTURE

Several researchers have attempted to use neural networks to solve this problem [4, 10]. The neural networks in these efforts had 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. The window was used to predict the secondary structure for the central amino acid. Output from the network was one unit for each type of secondary structure. The networks include some number of hidden units in a single layer between the input and output units (the number was varied in both studies); a general picture of this type of network is shown in Figure 4.

After the output values are determined, the resulting prediction is obtained. For each amino acid, the unit having the highest activation is chosen, with some exceptions made for "short" sequences. "Short" sequences are sequences of amino acids that predict `helix` or `sheet`, but are less than four or two amino acids in length, respectively. Since such "short" sequences do not in general occur in real proteins [6], these predictions are replaced with `coil`. The best test set accuracies reported from these efforts were 63.2% [4] and 62.7% [10].

3.3 THE CHOU-FASMAN DOMAIN THEORY

A standard approach in the biological literature to predicting secondary structure is the Chou-Fasman method [1]. The Chou-Fasman method is similar for both `helix` and `sheet` prediction; we will look at only `helix` prediction. The first step is to find helix *nucleation sites*. Nucleation sites are small regions of amino acids that are likely to be part of α-helix structures, according to the rules reported by Chou-Fasman. From this small region the structure is extended forward and backward along the protein as long as the likelihood of α-helix remains high. After `helix` and `sheet` regions have been predicted, relative likelihoods of the regions are compared to resolve overlaps.

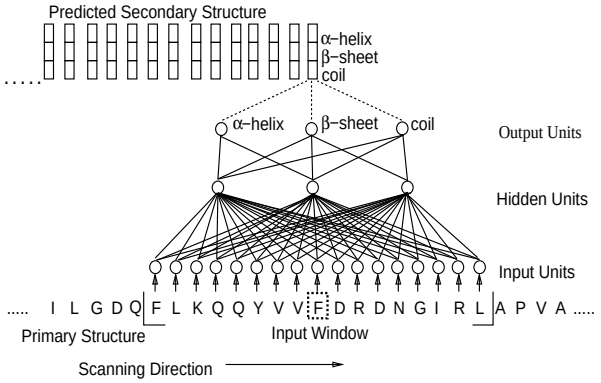


Figure 4: General architecture of networks from [4, 10].

A general picture of the finite-state automaton we used to represent this domain theory appears in Figure 5. Note the notion of transition in this automata is more complex. Each transition is represented by a set of rules applied to the input window and the current state to determine if the transition can be followed. Initial state of the FSA is `coil`.

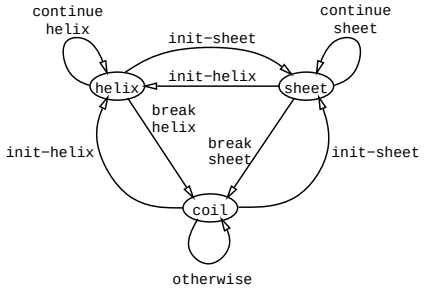


Figure 5: FSA used to represent the Chou-Fasman domain theory.

Sample Chou-Fasman rules are shown in Figure 6. Note the first three rules do not refer to state; these rules are subrules used by the transition rules (the last rule). The

assumption is that these rules are incomplete, incorrect, or inconsistent for some input(s). Finite-state KBANN takes the incorrect transition rules and refines them so that the correct transition is chosen.

```
If      amino acid = E
 Then   the amino acid is a helix-former
If      amino acid = P
 Then   terminate helix is true
If      center amino acid is a helix former And
        3 of 5 following amino acids
            are helix formers And
        conformation value is > 1.13†
 Then   initiate helix
If      Present State = coil And
        initiate helix is true
 Then   Next State = helix
...
```

Figure 6: Rules from the Chou-Fasman domain theory.

Recognizing a nucleation site depends only on information in the input window and can be implemented in regular KBANN. Extending a sequence, on the other hand, requires the finite-state capability of the system, since extending a sequence depends on having recognized a nucleation site in some previous input window. Since sequences are extended in both directions according to the Chou-Fasman method, the system scans a protein twice when it is presented. One scan proceeds forward over the protein and the next scan backward, with the results being combined. In this way the helices and sheets are extended in both directions. Once both scans are performed the results are averaged. A general picture of the type of network used to translate this domain theory is shown in Figure 7.

3.4 EXPERIMENTAL STUDY

We performed experiments with finite-state KBANN to determine if it does better than standard ANNs and the Chou-Fasman method at predicting secondary structure. Testing was done with the set of proteins used by Qian and Sejnowski [10]. The data set consists of 128 proteins. The proteins were randomly divided into training and test sets containing two-thirds (85 proteins) and one-third (43 proteins) of the original proteins, respectively. This process was repeated ten times. Each of the training sets of 85 proteins was further divided into four training sets: the first contained the first 10 of the 85 proteins; the second contained the first 25 of the 85; the third contained the first 50 of the 85; and the fourth had all 85 training proteins.

† The original Chou-Fasman domain theory suggests using a threshold of 1.03. Nishikawa [8] suggested that only the most likely nucleation sites should be chosen, thus the higher threshold.

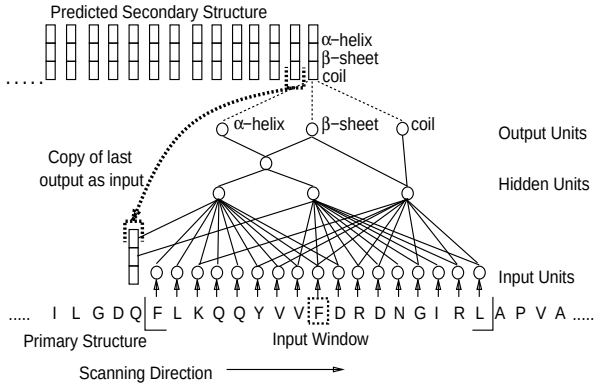


Figure 7: General structure of networks used to represent the Chou-Fasman domain theory.

Each network was trained for 60 epochs on its training set and the percent correctness recorded. The standard ANN used the same number of hidden units (28) as the networks used in the KBANN method. The results averaged over the 10 runs for each training set size are shown in Figure 8. The reason that the accuracy reported for the standard ANN is lower than that reported by Qian and Sejnowski is that they recorded the highest accuracy achieved by the system through all of the epochs, while our results only report the accuracy at the end of the 60 epochs.

3.4.1 Empirical Results

In Figure 8, finite-state KBANN shows a small increase in accuracy over standard ANN. Statistical analysis of the differences in prediction accuracy using a t-test show that the differences, while small, are statistically significant at the 0.5% level (i.e., with 99.5% confidence). For the largest training sets (the ones with 85 proteins), accuracy is 61.9% for finite-state KBANN and 59.8% for standard ANN, these results are also significant at the 0.5% level.

3.4.2 Discussion

While the results presented show improvement over standard neural network solutions and the Chou-Fasman method, the gain at this point is still fairly small. Detailed inspection of the errors made by the neural nets and the Chou-Fasman method show some similarities and differences. All three approaches do poorly in cases where both `helix` and `sheet` are predicted. All three do fairly well at finding the approximate location of α-helix and β-sheet structures, but less well in determining exactly how far these structures extend. The neural network techniques do well at predicting `coil`, but less well at `helix` or `sheet` prediction. This is not surprising since 54% of the secondary structure in the data set is `coil`, with `helix` and `sheet` being roughly equally split among the remaining instances. The KBANN

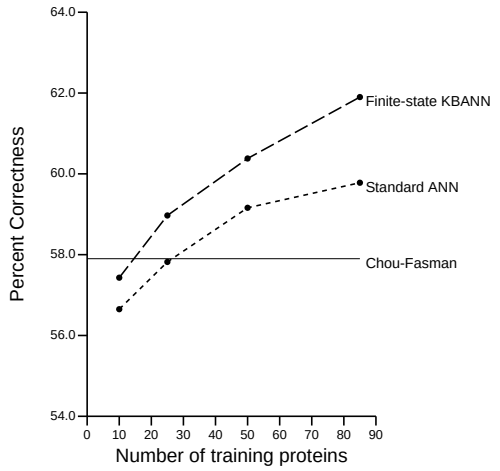


Figure 8: Average accuracy on test set as a function of training set size.

solution does a better job of predicting `helix` and `sheet` than the standard ANN solution, accounting for much of its advantage over the standard ANN method. This again is not surprising since the Chou-Fasman domain theory focuses on predicting `helix` and `sheet`, with `coil` only predicted as a default.

Since the present approach does not achieve high accuracy, a number of extensions to the approach are being considered. A better initial representation of the Chou-Fasman domain theory might raise accuracy since KBANN is dependent on the quality of its initial domain theory. Also, combining Chou-Fasman with other techniques such as the Lim [7] or Robson et al. [11] algorithms might produce better results.

4 CONCLUSIONS

The finite-state KBANN method provides a useful and powerful extension to the standard KBANN technique. Finite-state KBANN represents domain theories expressed as finite-state automata. The state of the automata is remembered in the neural network by copying the state back as input to the network in the next step. The "scanning" of the input string in an FSA is handled by having the network scan the input, activating it with the current input value and the current state to determine the output state. Finite-state KBANN derives a set of rules from an FSA which are translated into a neural network. The state transitions are refined using a set of training examples so that they will be correct for the training examples.

By allowing domain theories that include state-dependent rules, KBANN has been extended to cover additional real-world applications. The application of finite-state KBANN to the protein structure-prediction problem demonstrates that this method may produce improvement on standard neural network methods and on existing non-learning algorithms (e.g. Chou-Fasman). However, more work must be done both in generating better domain theories and improving the network refinement process.

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.

References

1. Chou, P. Y. and Fasman, G. D., "Prediction of the secondary structure of proteins from their amino acid sequence," *Advan. Enzymol. 47*, (1978), 45-148.
2. Cleeremans, A., Servan-Schreiber, D. and McClelland, J. L., "Finite state automata and simple recurrent networks," *Neural Computation 1*, 3 (1989), 372-381.
3. Elman, J. L., "Finding structure in time," *Cog. Sci. 14*, 2 (1990), 179-211.
4. Holley, L. H. and Karplus, M., "Protein structure prediction with a neural network," *Proc. Nat. Acad. Sci. 86*, (1989), 152-156.
5. Jordan, M. I., "Serial order: a parallel distributed processing approach," TR 8604, UCal, Inst. for Cog. Sci., San Diego, 1986.
6. Kapsch, W. and Sander, C., *Biopolymers 22*, (1983), 2577-2637.
7. Lim, V. I., "Algorithms for prediction of α-helical and β-structural regions in globular proteins," *J. Mol. Bio. 88*, (1974), 873-894.
8. Nishikawa, K., "Assessment of secondary-structure prediction of proteins comparison of computerized Chou-Fasman method with others," *Biochim. Biophys. Acta 748*, (1983), 285-299.
9. Noordewier, M. O., Towell, G. G. and Shavlik, J. W., "Training knowledge-based neural networks to recognize genes in DNA sequences," in *Advances in Neural Information Processing Systems*, Vol. 3, Morgan Kaufmann, Denver, CO, 1991.
10. Qian, N. and Sejnowski, T. J., "Predicting the secondary structure of globular proteins using neural network models.," *J. Mol. Bio. 202*, (1988), 865-884.
11. Robson, B. and Suzuki, E., "Conformational properties of amino acid residues in globular proteins," *J. Mol. Bio. 107*, (1976), 327-356.
12. Rumelhart, D. E., Hinton, G. E. and Williams, R. J., "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the microstructure of cognition. Volume 1: Foundations*, Rumelhart, D. E. and McClelland, J. L. (ed.), MIT Press, Cambridge, MA, 1986, 318-363.
13. Sejnowski, T. J. and Rosenberg, C. R., "Parallel Networks that Learn to Pronounce English Text," *Complex Systems 1*, (1987), 145-168.
14. Towell, G. G., Shavlik, J. W. and Noordewier, M. O., "Refinement of approximate domain theories by knowledge-base neural networks," *AAAI90*, 1990, 861-866.