The Effect of Reasoning with State Information

Cathie LeBlanc
Department of Computer Science
Keene State College
Keene, NH 03435
leblanc@cs.fsu.edu

Susan I. Hruska
Department of Computer Science
Florida State University
Tallahassee, FL 32306-4019
hruska@cs.fsu.edu

Abstract

In this article, we describe a method for the incremental development of knowledge-based neural systems which incorporate state information about a problem domain. Human expert domain knowledge is encoded in the architecture of a computational network and data-driven techniques are used to refine the encoded knowledge. State information, in the form of state nodes and recurrent network connections, is then added in order to augment the refined knowledge even further. The methodology is tested in the domain of protein secondary structure prediction, where a set of domain rules, the Chou-Fasman algorithm, is implemented in phases as an expert network. The domain rules are then refined using a backpropagation-like learning algorithm. Finally, state information is added to the resulting network and is shown to increase prediction accuracy.

1 Introduction

Hybrid systems which combine rule-based knowledge about a problem domain with data-driven knowledge discovery methods have been implemented in a number of ways, including *expert networks* [9]. An expert network is a computational network whose topology is derived from an underlying knowledge base. Each of the nodes in the network has a unique functionality according to the role the node plays in encoding the knowledge base [9]. Such systems allow human expert knowledge to be explicitly encoded in the system and then refined by automated examination of data. The topologies and learning algorithms developed for expert networks prior to the work described in this paper have been strictly feed-forward.

Some classification tasks, however, are difficult to complete using a strictly feed-forward architecture. In particular, the solution to many problems requires that *state* information be maintained. State information is the context in which the problem is currently being solved. The context of a problem solution will change as the solution proceeds and certain sets of rules need only be considered in certain contexts. In order to maintain such state information, we have added recurrent connections to the topologies and modified the learning algorithms allowed for expert networks.

A particular problem domain for which state information is important is the problem of predicting the secondary structure of a protein from its primary sequence. Algorithms developed for prediction of protein secondary structure do not take into account information concerning interactions between amino acids which occur far apart along the protein primary sequence although many studies (*e.g.*, [13]) have shown that such interactions play a significant role in the formation of actual protein structure. Information concerning these long-range interactions constitutes the state information necessary to adequately address this prediction problem.

We have encoded the rules from the Chou-Fasman [4] protein secondary structure prediction algorithm into an expert network and augmented this expert network with knowledge about longer-range amino acid interactions. Results of protein secondary structure prediction experiments show that the addition of state knowledge to existing algorithms encoded as expert networks aids in the development of better prediction tools.

2 Expert Networks

Expert networks have been described in detail by various researchers at Florida State University [8] [10]. Early exploration of expert networks showed how to create the acyclic topology of an expert network using the rules from a rule-based expert system [9] and how to use a backpropagation-like learning algorithm to extract knowledge from data [8]. Later work removed the need for an existing rule base for the creation of the expert network topology [11] [12].

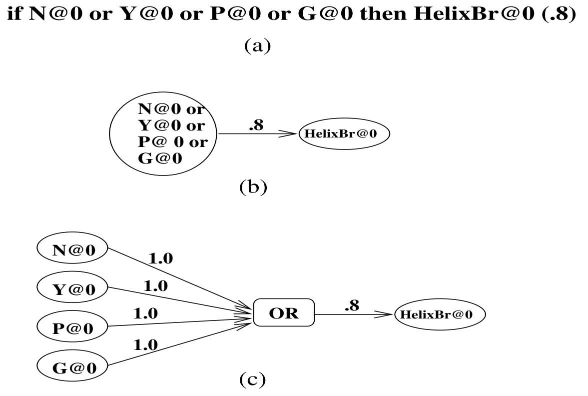

Figure 1. Transforming a Rule Base into an Expert Network

The transformation from rules to the architecture of the expert network occurs in two steps [9]. In the first step, each assertion in the rule is represented by a node in the expert network. Since a rule is simply a relationship between two or more assertions, the implication of the rule is represented by the weighted connections between the nodes representing its assertions. The second step of the transformation concerns the assertions that contain logical operations, such as AND, OR, or NOT. In this step, the logical operation is separated from the elements that the operation connects. Figure 1 shows the transformation process.

Note that transformation from expert system rules results in a somewhat non-traditional network, one which has no identifiable layer structure and is sparsely connected. In addition, the combining, firing and output functions for each node in the expert network are derived from the inference engine dynamics of the expert system shell in which the expert system was defined [9]. For example, if the expert system shell is M.1 [19], a node which corresponds to an AND would have a firing function which outputs the minimum of its inputs because this is how the inference engine of M.1 implements AND.

Recently, researchers at Florida State University have shown that one need not start with a rule base in order to build an expert network [12]. Instead, one only needs expert knowledge to build an expert network. The key point is that the architecture of the network is determined by the expert knowledge base that is to be encoded in it.

3 Recurrent Expert Networks

Many problems cannot be solved adequately using strictly feed-forward network architectures. In particular, problems in which elements of the input patterns interact with each other over uneven distances are difficult, if not impossible, to solve using feed-forward architectures. For example, in Dutch, the infinitive form of any verb must be placed at the end of the sentence. Therefore, if one wants to say "I want to go there," in Dutch, one would actually say "I want there to go." If the task is to understand the meaning of the Dutch sentence, one must connect the verb "want" with what one wants to do, which is "to go." The distance between the two related verbs is variable in any Dutch sentence. If a neural network-like tool is developed for understanding Dutch sentence, it would likely have to be a recurrent network of some sort in order to keep information in the system long enough for it to be related to the later information. In fact, much of the research done on recurrent neural networks has been focused on the problem domain of natural language understanding [*e.g.,* [5]].

Recurrent neural networks can be categorized as *partially recurrent* networks, which are recurrent in their activation only, and *fully recurrent* networks, which are recurrent in both their activation and their learning [7]. Fully recurrent networks allow arbitrary connections among nodes and implement learning algorithms which allow learning of the weights on all connections in the network. These learning algorithms are often based on backpropagation of error with an additional *relaxation step* [1]. The algorithms require long training periods which make them impractical to use in real applications. On the other hand, partially recurrent networks have a small, carefully chosen set of recurrent connections whose weights are fixed and cannot be learned. Because these weights are fixed, standard backpropagation of error can be used to learn the weights on the other feed-forward connections. Because of the difficulties associated with fully recurrent neural networks, we have chosen to implement partially recurrent expert networks, taking advantage of the ability to use a backpropagation-like algorithm for training.

Figure 2 shows a schematic view of recurrent architectures implemented to allow state information to be maintained. In these architectures a set of *input nodes* receives external input from an input pattern file. These input nodes are connected, consistent with the knowledge found in the underlying knowledge base, to a subset of the *hidden nodes*. The hidden nodes have a variety of connections among them, again in a manner which encodes the knowledge from the knowledge base. A subset of the hidden nodes is then connected to the *output nodes*, which receive external information concerning the ideal outputs for each of the nodes for each input pattern. For each output node, a *state node* node and a synapse between the output node and the state node are created. Each of the state nodes is copied a number of times. The outputs of all state nodes corresponding to a particular output node are combined according to additional domain knowledge and the combined value is sent into the corresponding output node. These connections allow each output node to receive information

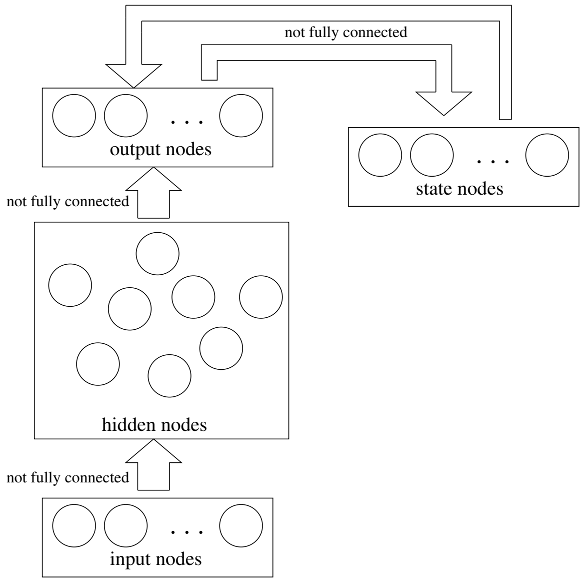

Figure 2. Recurrent Expert Network Architectures

concerning previous network predictions.

4 Protein Secondary Structure Prediction

The problem of predicting the folded structure of a protein solely from the linear sequence of its building blocks (the amino acids) is the major outstanding problem in structural molecular biology today [3]. The sequence of amino acids that makes up the protein is called the *primary sequence* of the protein. The role that the protein will play in the living cell is determined by its *tertiary structure*, that is, the linear sequence of amino acids folded as a three-dimensional entity. The folding of a protein is called its *conformation*. The *secondary structure* of a protein refers to arrangements of relatively short amino acid subsequences into intermediate-level conformations that appear over and over in many folded proteins. Secondary structure conformations include α-*helices*, which are spirals of amino acids in three dimensions, and β-*strands*, which bind to each other to form sheets of amino acids [3]. Amino acids not participating in helices or strands are considered to be participating in *coil* conformations.

In order to determine the function of a protein, its tertiary structure must be found. In 1961, Anfinsen and his colleagues demonstrated that the structure (and therefore, the function) of a protein is determined by its primary sequence of amino acids [2]. Many researchers have approached the problem of predicting the tertiary structure of a protein as a problem of first predicting the secondary structure of the protein and then assembling the secondary structure elements into a tertiary structure [6]. Thus, many have viewed the prediction of the secondary structure of a protein as a necessary step toward prediction of its tertiary structure.

Many algorithms have been developed for protein secondary structure prediction. The Chou-Fasman algorithm [4] is a widely used algorithm for protein secondary structure prediction. Its prediction accuracy, however, is estimated to be between 50% [3] and 58% [16].

4.1 The Chou-Fasman Algorithm

In the Chou-Fasman approach to protein secondary structure prediction, two conformational parameters, a helix propensity value, P_α, and a strand propensity value, P_β, are calculated for each amino acid from the relative frequency of the amino acid in helices and strands [17]. From the helix propensity value, each amino acid is classified as a strong helix former (H_α), a helix former (h_α), a weak helix former (I_α), an indifferent helix former (i_α), a weak helix breaker (b_α) or a strong helix breaker (B_α). Similarly, from the strand propensity value, each amino acid is classified as a strong strand former (H_β), a strand former (h_β), a weak strand former (I_β), an indifferent strand former (i_β), a weak strand breaker (b_β) or a strong strand breaker (B_β). The Chou-Fasman algorithm then uses these conformational parameters in a simple set of rules to predict the secondary structure designation of a region in a primary sequence. The set of Chou-Fasman rules described in Maclin and Shavlik [14] is used as the basis for the expert networks for protein secondary structure prediction described in the next section of this paper.

4.2 Correlation Coefficients

In prediction of the secondary structure of proteins, there are three possible output classes: *helix*, *strand*, or *coil* [3]. Every amino acid falls into one of these three categories. In nature many more amino acids participate in coil structures than in either helices or strands. Of the 21,623 amino acids in the two datasets created by Qian and Sejnowski [18], 55% participate in coil structures while only 25% are helical and 20% are strands. By predicting all amino acids to be coil, a prediction algorithm can easily achieve 55% prediction accuracy. Such an algorithm is useless, however, for actual prediction of protein secondary structure.

As a result of the high number of coil amino acids in proteins, other measurement tools for prediction accuracy have been developed. Of these tools, Matthews' correlation coefficient is the most commonly used [15]. In this method for determining prediction accuracy, the correlation between the observed structure for the protein and the predicted structure for the protein is calculated. The

Experiment	*Before Training*	*After Training*	*Average*
Helix Finding	.226055	.247928	NA
Strand Finding	.206484	.188305	NA
Synthesis of SubNetworks	H: .170847 S: .266290 C: .292876	H: .170847 S: .266290 C: .292876	.243338
State with Difficult Coils	H: .170847 S: .134265 C: 0.0	H: .246673 S: .202298 C: .292876	.247282
State with Easy Helices Network A	H: .168548 S: .147477 C: 0.0	H: .254184 S: .192497 C: .288846	.245509
State with Easy Helices Network B	H: .168548 S: .147477 C: 0.0	H: .168548 S: .266290 C: .288846	.241228

Figure 3. Results of Experiments

correlation coefficient for each secondary structure type is measured separately and algorithms with good prediction accuracy will do equally well on all three structure types. A correlation coefficient of 1 means a perfect correlation between the observed secondary structure and the predicted secondary structure, a coefficient of -1 means no correlation whatsoever has been found and a coefficient of 0 means the correlation is what one would expect from a completely random prediction. Values between .2 and .4 are typical for secondary structure prediction algorithms.

5 Experiments

The benchmark protein datasets created by Qian and Sejnowski [18] were used in all the studies reported here. These proteins are split into a training dataset of 111 proteins and a test dataset of 17 proteins. We chose a random subset of the training data as our training dataset, which contains 1965 helical amino acids, 1135 strand amino acids and 2989 coil amino acids taken from 35 proteins. The test dataset contains 849 helical amino acids, 746 strand amino acids and 1923 coil amino acids taken from 17 proteins. All training proceeded using the ΔW averaging technique [12], which averages weight changes by the number of patterns of a particular example type in order to prevent large numbers of one example type from overwhelming the training process. We used a learning rate of .0001. Each input pattern contains information about the Chou-Fasman conformational parameters of the local region of the amino acid sequence along with information about the actual amino acids found in that region. The ideal output associated with a particular input pattern corresponds to the observed secondary structure of the amino acid located in the center of the local region.

5.1 Helix Finding and Strand Finding Expert Sub-Networks

The first step in developing a recurrent expert network for protein secondary structure prediction is to use the Chou-Fasman algorithm rules to create two sub-networks and train those networks using example data. The first sub-network encodes the rules for finding helices and, thus, makes predictions about whether a particular amino acid participates in a helix or not. The second sub-network encodes the rules for finding strands. The weights on all synapses were set initally to 1.0 and therefore, in Figure 3, the column showing the correlation coefficient before training represents the extent to which the original Chou-Fasman rules accurately predict helices and strands. The networks were then trained separately for 2000 iterations and the results of the trained networks on the test set are also shown in Figure 3.

5.2 Synthesis of the Two Sub-Networks

A network for finding helices is of little practical use in predicting the secondary structure of actual proteins without some integration of the network into a larger system for prediction of all secondary structure. The same holds true for the strand finding network. Therefore, the helix finding and strand finding expert network predictions must be combined in order to create a system which will predict overall secondary structure for proteins. The most difficult part of this task comes in determining how to resolve conflicts between the outputs of the two networks. Figure 4 shows the method for synthesis of the helix finding and strand finding sub-networks. In the network shown in this figure, the synapses in the helix and strand finding sub-networks were fixed so that they could no longer be changed. The synapses represented by dotted lines in Figure 4 were updated during training of this synthesis network while those represented by solid lines remained unchanged with a value 1.0.

In the synthesis network shown in Figure 4, the helix finding sub-network feeds into an ADD node with a threshold of 0.6. If the output of the helix finding sub-network is above 0.6, the ADD node outputs a 1. Otherwise, the ADD node outputs a 0. The strand finding sub-network feeds into a similar ADD node. The threshold of 0.6 was determined experimentally. If the helix finding network predicts `not helix` and the strand finding network predicts `not strand`, the overall network predicts `coil`. If the helix finding network predicts `helix` and the strand finding network predicts `strand`, the overall network predicts both as well, and an outside observer must choose the prediction with the highest output value. The prediction results of the synthesis network is shown in Figure 3. The column *Before Training* represents the prediction results

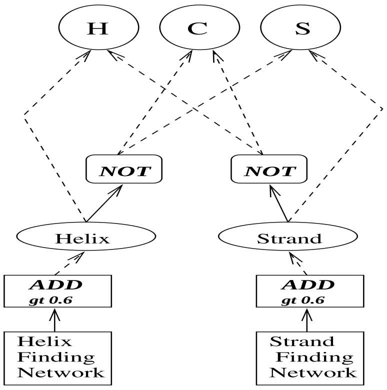

Figure 4. Synthesis of the Helix and Strand Finding Sub-Networks

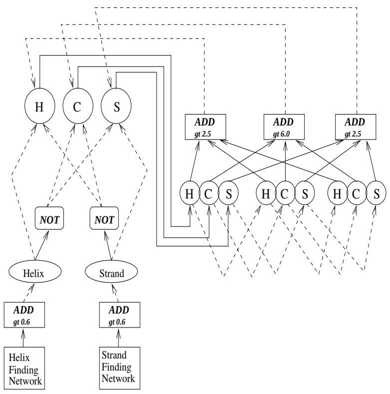

Figure 5. Network Incorporating State Knowledge and Requiring More Evidence for Coil Prediction

of the synthesis network before training of the synapses shown as dotted lines in Figure 4 while the column *After Training* represents the coefficients after training of those synapses. Each of these two columns shows the correlation coefficients for all three secondary structure types, with H representing the coefficient for helices, S representing the coefficient for strands and C representing the coefficient for coils. The third column in Figure 3 shows the average correlation coefficient of the trained network for the three secondary structure types on the test dataset.

5.3 Adding State Nodes and Additional Domain Knowledge

Figure 5 shows a recurrent expert network which incorporates state information. In this network, we have added three sets of state nodes which maintain information about network predictions for previous amino acids in the protein. The set of state nodes for the prediction of helices all feed into an ADD node with a threshold of 2.5, which means that if the sum of the outputs of the helix state nodes is greater than 2.5, the ADD node outputs a 1 and outputs a 0 otherwise. The set of strand state nodes feeds into a similar ADD node with a threshold of 2.5. Because of the observation that the prediction of coil begins to overwhelm all the other predictions during the course of training, the threshold of the ADD node for the coil state nodes is 6.0. This increases forces a higher level of evidence to be present in order for the network to predict coil. The results of using this network for prediction are shown in Figure 3 in the row labeled *State with Difficult Coils*. When untrained, this network does not predict any coils. Therefore, the correlation coefficients in the untrained network are extremely low. On the other hand, the trained network has an average correlation coefficient of .247282, which is significantly higher than the trained network without any state information. The fact that this network has a higher average correlation coefficient than the network without state information suggests that state information is useful in increasing prediction accuracy.

A final experiment at maintaining state information during the course of the problem solution attempts to make helices easier to predict by changing the threshold for the ADD node into which helix finding sub-network feeds from 0.6 to 0.4. The network is shown in Figure 6 and the changed is highlighted. The results of two different training experiments using this initial network are shown in Figure 3, in the two rows labeled *State with Easy Helices, Network A* and *State with Easy Helices, Network B*. Network A has an average correlation coefficient of .245509. Network B has a lower average correlation coefficient of .241228. One can see that both trained networks have the same coil correlation coefficient but Network A has a higher helix correlation coefficient than Network B and Network B has a higher strand correlation coefficient than Network A. The disparity between helix and strand prediction accuracy is higher in Network B than in Network A.

By examining the weights on the synapses in the trained network, it is apparent that the Network B has learned to weight the state information less than other information. By weighting this state information less, the prediction accuracy of the network drops. For Network B, the weights on the synapses labeled $H1$, $H2$, and $H3$ in Figure 6 are all just about equal, with values between 0.96 and 0.97. The weights on the synapses between the state nodes, however, have widely varying values. In the trained network, the weight on the synapse between the first helix state node and the second helix state node, labeled $H4$ in Figure 6, is 0.54. In the Network A, the weight on this synapse is 0.83 while the weights on synapses $H1$, $H2$, and $H3$ are

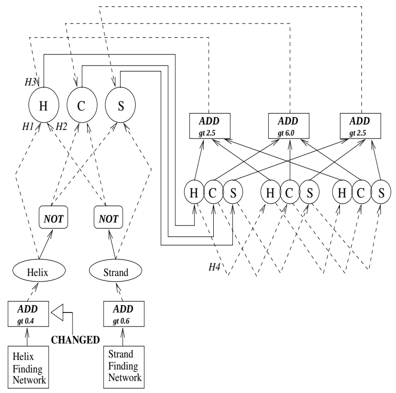

Figure 6. Network Incorporating State Knowledge and Requiring Less Evidence for Helix Prediction

between 0.96 and 0.97. The benefit of state information in prediction accuracy can be seen from the fact that when the network weights the state information less strongly, the prediction accuracy of the network drops.

6 Conclusion

In this paper, we present a methodology for encoding existing domain knowledge in a network architecture and then refining that knowledge via data-driven methods. Many problems are not adequately addressed by feed-forward networks. Such a problem, the protein secondary structure prediction problem, is addressed via an expert network. Since this problem is known to be significantly affected by state information, the knowledge encoded in the expert network is augmented via recurrent connections in the network architecture. These recurrent connections capture the state information and are shown to improve the prediction accuracy of the network.

References

[1] L. Almeida. Backpropagation in perceptrons with feedback. In R. Eckmiller and C. von der Malsburg, editors, *Neural Computers*, pages 199–208, Berlin, 1988. Springer-Verlag.

[2] C. Anfinsen, E. Haber, M. Sela, and F. White. The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. *Proceedings of the National Academy of Sciences, USA*, 47:1309–1314, 1961.

[3] C. Branden and J. Tooze, editors. *Introduction to Protein Structure*. Garland Publishing, Inc., New York, 1991.

[4] P. Y. Chou and G. D. Fasman. Prediction of protein conformation. *Biochemistry*, 13(2):222–245, 1974.

[5] J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[6] G. D. Fasman, editor. *Prediction of protein structure and the principles of protein conformation*. Plenum Press, New York, 1989.

[7] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the theory of neural computation*. Addison-Wesley Publishing Company, Redwood City, California, 1991.

[8] S. Hruska, D. Kuncicky, and R. Lacher. Hybrid learning in expert networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume II, pages II117–II120, Piscataway, NJ, July 1991. IEEE.

[9] D. Kuncicky, S. Hruska, and R. Lacher. Hybrid systems: the equivalence of expert system and neural network inference. *International Journal of Expert Systems*, 4:281–297, 1992.

[10] R. Lacher. Expert networks: Paradigmatic conflict, technological rapprochement. *Minds and Machines*, 3:53–71, 1993.

[11] A. Levis, R. Timpany, W. Austad, J. Elling, J. Ferguson, D. Klotter, and S. Hruska. Application of knowledge-based network processing to automated gc data interpretation. In *Proceedings of SPIE AeroSense '95*, volume 2492, pages 294–302, Bellingham, WA, April 1995. International Society for Optical Engineering.

[12] A. P. Levis, K. L. Adair, and S. I. Hruska. A table-based approach to expert fault diagnosis networks. In *Proceedings of ANNIE'96*. Artificial Neural Networks in Engineering, 1996.

[13] T. Logan, Y. Theriault, and S. Fesik. Structural characterization of the FK506 binding protein unfolded in urea and guanidine hydrochloride. *Journal of Molecular Biology*, 236:637–648, 1994.

[14] R. Maclin and J. W. Shavlik. Using knowledge-based neural networks to improve algorithms: Refining the Chou-Fasman algorithm for protein folding. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 165–170, Menlo Park, CA, July 1992. American Association for Artificial Intelligence, AAAI Press/MIT Press.

[15] B. Matthews. Comparision of the predicted and observed secondary structure of T4 phage lysozyme. *Biochimica and Biophysica Acta*, 405:442–51, 1975.

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

[17] P. Prevelige, Jr. and G. D. Fasman. Chou-Fasman prediction of secondary structure. In G. D. Fasman, editor, *Prediction of Protein Structure and the Principles of Protein Conformation*, pages 391–416, New york, 1989. Plenum Press.

[18] N. Qian and T. Sejnowski. Predicting the secondary structure of globular proteins using neural network models. *Journal of Molecular Biology*, 202:865–884, 1988.

[19] Teknowledge. *M.1 Reference Manual*. Teknowledge, Palo Alto, CA, 1986.