

# Classification of Transmembrane Helices and Signal Peptides using Neural Networks \*

Austin Che  
austin@cs.stanford.edu

June 9, 2000

## Abstract

The problem of predicting structure from sequence alone is exceedingly complex. Experimenters usually simplify the problem by concentrating on identifying very broad groups and classes of structures, for instance,  $\alpha$ -helices,  $\beta$ -strands, or loops. In this paper, I will evaluate the applicability and efficiency of one method involving neural networks in classifying sequences as signal peptides or transmembrane helices.

## 1 Introduction

Various methods have been developed to identify features of a protein solely from their sequence. Most of these involve taking a group of sequences with the desired feature and finding patterns in the sequences using methods such as multiple sequence alignments, motifs, or position scoring matrices. Here I will consider another method involving training neural networks to recognize the patterns in the training sequences.

Neural networks have been used to identify transmembrane helices with a high success rate (Rost 1995). Neural networks have also been

---

\*Thanks to Doug Brutlag for his support and ideas.

used to differentiate signal peptides from non-signal peptides (Nielsen 1997). Signal peptides are important since they control where proteins are secreted. They are usually cleaved off when the protein goes through the membrane. The neural network method presented by Nielsen, et. al., is able to identify signal peptides from non-signal peptides and also is able to identify the cleavage sites for signal peptides.

Using a similar neural network method used by both Nielsen and Rost, this paper attempts to classify both signal peptides and transmembrane helices. The approach presented in this paper is therefore not a new one. Instead of identifying either signal peptides or transmembrane helices, the goal was to see if it was possible to train a neural network to identify between these two classes of sequences. In doing so, it was possible to evaluate the viability of using neural networks in solving simple but biologically important questions, and in extending the method to more complicated problems.

## 2 Data

The data set was generated by combining data from several places. Signal peptide data came from the SignalP server set up by Nielsen and mentioned in his paper. This signal peptide data came from Swiss-Prot version 29 and further information about how this data was generated is described by Nielsen. The transmembrane helix sequences were obtained from TMbase which is described by Hofmann. The version used, TMbase25, is based on Swiss-Prot version 25.

To generate the data set, the first 20 amino acids of each sequence were used to make the data uniformly the same length. There were 4 types of data included: normal transmembrane helices, transmembrane helices that were labeled as signal anchors in the comments field of its Swiss-Prot entry, signal peptides, and as background, sequences that were in none of the above categories. In all, there were 10,560 sequences put into the full data set.

## 3 Methods

### 3.1 Signal Anchors

Signal anchors are also referred to as uncleaved signal peptides, and are in some respects similar to signal peptides. In the neural network developed by Nielsen, their classification would often mistake signal anchors for signal peptides. Therefore, in an attempt to avoid part of this problem and to decrease the error, the neural network was trained on sequences of signal anchors and was told to identify it as such. Thus the neural network was told to distinguish between four groups: transmembrane helices that were not signal anchors, transmembrane helices that were signal anchors, signal peptides, and none of the above.

### 3.2 Neural Networks

A feed forward neural network was used and trained using the back propagation algorithm (Russell 1995). The activation function for each unit in the neural network was a sigmoid function and the weights were updated according to the standard error function. Since the data set was generated so that all sequences were 20 amino acids long, it was easy to encode this information into the structure of a neural network.

Each amino acid was converted into a set of features. For this, ProtScale on the ExPASy server was used. The number and type of features were varied to find the minimal number that would give good results. After the features were chosen (see experiments below), the numbers were all normalized to be between -1 and 1. For each amino acid, there was one input node for each feature of that amino acid. There were three output nodes: one corresponding to transmembrane helix (non-anchor), one to transmembrane helix (anchor), and one to signal peptide. The output was determined by the node with the highest value. However, if the maximum value was not greater than the second greatest value by a certain threshold (currently at 0.1), then the neural network returns no classification. Thus, only if the neural network is in some sense “sure” of itself, does it return a classification.

The entire data set was randomly split into a training set and a

test set. The training set was picked by randomly choosing 75% of the entire database and the test set was picked by randomly choosing 25% of the entire database. Each training iteration consisted of picking a fraction of the training set, usually set to 60%, and the neural network was trained on that subset. This randomness was shown to get better results than training on the full set.

Initially, the neural network started with random weights. In fact, this net reached over 50% accuracy! This was because it classified everything as a transmembrane helix, and over 50% of the training set was transmembrane helices. After 1 training iteration, this dropped to 14% and the real training began. This also suggested that tests should be made to ensure that a high accuracy is not just due to chance errors like this, and that it is actually due to some actual identification of peptide sequences. The accuracy on each group for the following experiments was tested and compared to the overall accuracy. With very few exceptions, the total accuracy was a good indicator of the accuracy in predicting signal peptides and transmembrane helices. The accuracy on signal anchors was often significantly lower, most likely due to the relatively small proportion of the training set that were signal anchors.

As an example of one training session with a neural network, a net with two hidden layers, one with 100 units and one with 10 units was used. Initially, the test set was identified at 1.4%. After 5000 iterations of training on the training set, the new trained neural net reached 98% accuracy on the test set and 99.94% accuracy on the training set.

## 4 Experiments

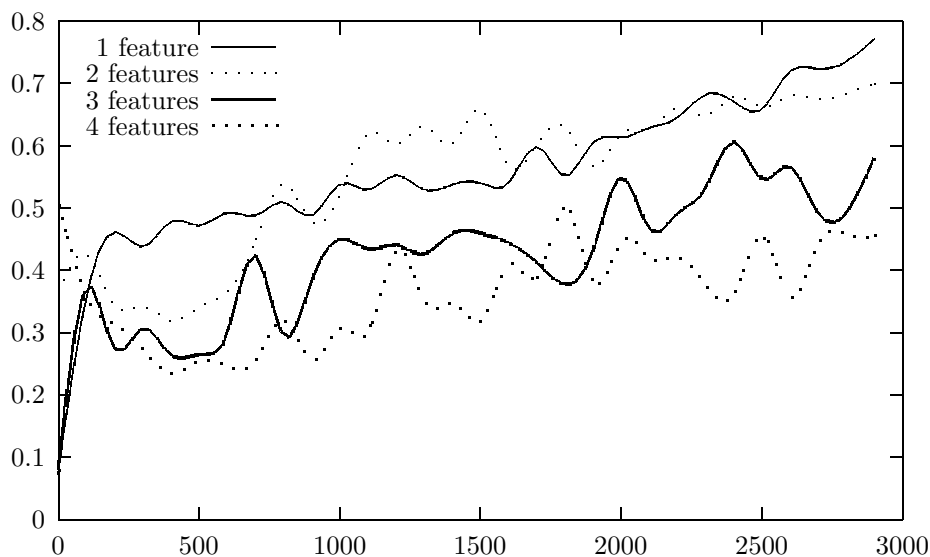
Deciding on a neural network method still left many parameters that could be tweaked to optimize the performance. Here, a couple of the more important parameters affecting performance are considered.

## 4.1 Feature test

The first important aspect that had to be determined was the number and the type of amino acid features to use. A neural net with 100 nodes in one hidden layer was trained for 100 iterations with different number and type of features. There were four features that were taken from ProtScale: the average flexibility index, the polarity, the bulkiness, and the hydrophobicity.

Various combinations of these features were tested. First, each of these features were tested separately. On the test set, the one that was trained on hydrophobicity scored marginally better than the ones trained on polarity or bulkiness, and slightly better than the one trained on flexibility.

Next the number of features was tested. The vertical axis is the accuracy on a test set and the horizontal axis represents the number of iterations.



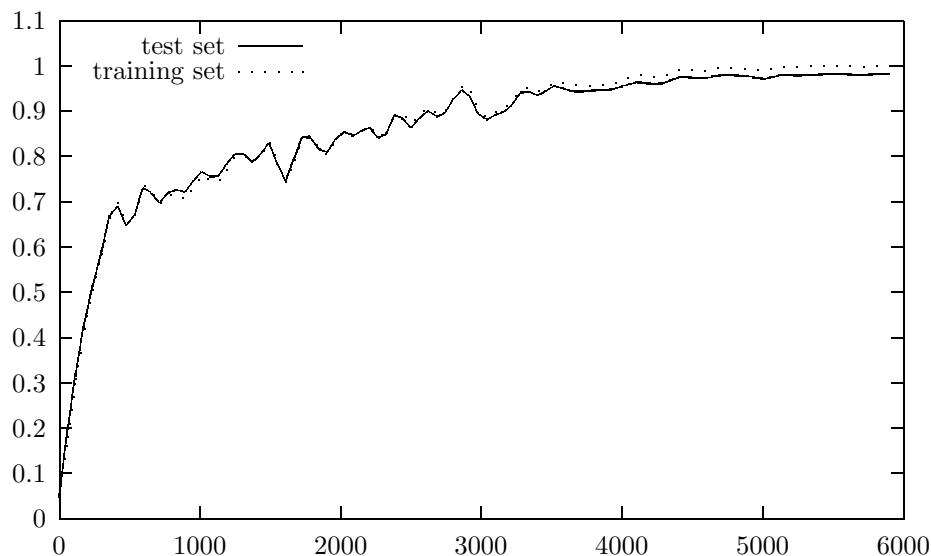
From this it initially seemed that the fewer features that one uses, the better accuracy that one gets. However, further tests show this not to be the case. A neural net with one hidden layer with 100 units was trained for 5000 iterations using all 4 of the features. After train-

ing, the trained neural net reached 97% accuracy on the test set and 98.3% accuracy on the training set. Another similar neural network was trained using only one feature – hydrophobicity. After 5000 training iterations, it only reached 92% accuracy on the test set and 94% accuracy on the training set.

These results show that using fewer features, namely only one feature, does better in the short run than using multiple features. However, after many iterations, using more features allows for more accuracy in the long run, and converges faster also. It took over 8000 iterations for a neural net using only one feature to correctly classify 100% of the training data while a neural net with four features could do it in under 6000 iterations.

## 4.2 Number of Iterations

In order to test how the number of training iterations influences the accuracy of the trained neural network on the test set, a neural net with one hidden layer of 100 nodes was trained. Here is a plot of the accuracy of the network as a function of the number of iterations.

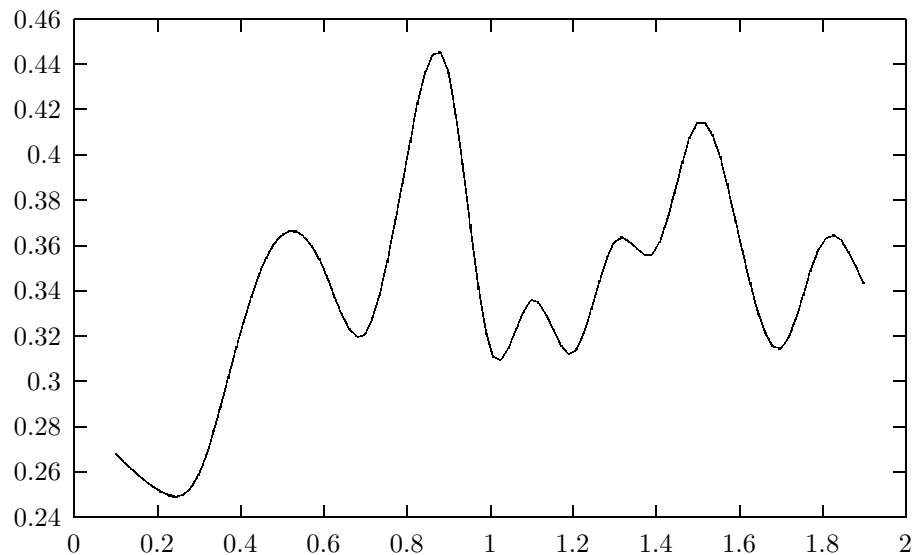


Not surprisingly, the neural network classifies the test data more

accurately as the number of training iterations increases. The major concern when doing too many iterations is that of over-fitting. However, as can be seen here, the accuracy on the test data continues to go up until it hits 98%. There does not seem to be any drop in the accuracy on the test set even though the accuracy on the training set reaches 100%. One of the methods used to prevent over-fitting and to get the graph shown here, was to automatically throw away data that was already classified well enough. That is, the neural network only picks data that gives it new information instead of trying to match perfectly with each sequence. This allows the neural network to reach a point where it stops changing. Here this occurred after 5800 iterations at which point it could correctly classify 100% of the training data.

### 4.3 Alpha test

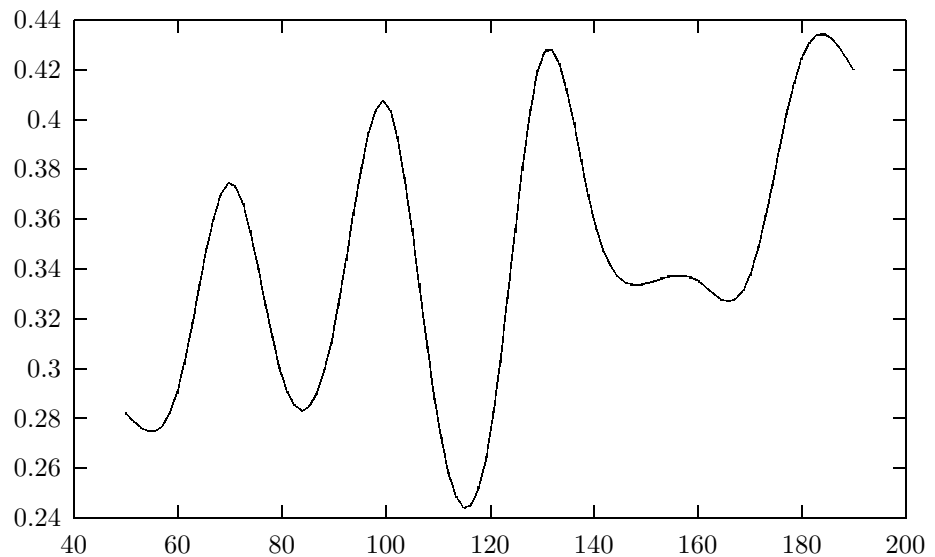
Alpha is the learning rate for the neural network. The higher alpha is, the “faster” the network learns. Using a higher alpha allows quicker convergence but also may introduce over-fitting. Here we see the accuracy on test data plotted against various alpha, with each neural net being trained for 100 iterations.



As can be seen, it seems that there are some good values of alpha and some bad values of alpha. *Unfortunately, there seems to be no visible pattern of good or bad alpha.* This as we will see is a significant problem of neural networks that has to be overcome to make neural networks a useful learning tool.

#### 4.4 Complexity test

A neural net with one hidden layer was trained for 100 iterations. The number of units in the one layer was varied. Here are the results on the test set.



Once again, it appears that although there is some correlation between complexity of the network and the accuracy, it would be extremely difficult to predict a reasonably good structure for a neural network given a specific problem. Also, this experiment only tried a small set of possibilities for a neural network fixed at one hidden layer. Any number of hidden layers could be added with completely different results. The problem, and from where neural networks gets its strength and power, is the numerous connections between nodes. Adding a new node does not just add to the existing neural network; it completely

changes the network.

## 5 Future Directions

The experiments performed showed definite promise for neural network methods. In a very short time, it was possible to train a neural network to distinguish between transmembrane helices and signal peptides up to 98% accuracy. However while this is a reasonable accuracy, there is still room for improvement.

An ideal classification method would be flexible and intuitive to use. Unfortunately neural networks, while purporting to simulate the way neurons in the human brain are connected, have many shortcomings, that our brains don't. If neural networks hope to reach the level of proficiency of the greatest thinking devices that are known, it may as well take its emulation one step further, since at least we know that our brains do work.

### 5.1 Neural Networks with Memory

One of the major problem with the method used here is that each training data is treated as a separate instance; multiple sequences and patterns between sequences are not used to help the neural network generalize. Our brain contains a memory and a history of previous things seen. When new data comes in, we can compare it to everything in memory and use that to learn. Neural networks, on the other hand, takes each training instance as separate. True, neural networks keep some form of memory in the weights, but it is not the same kind of memory that our brains use. Not being able to look at more than one sequence at a time severely limits the generalization that can be done.

One way that has been tried by others to improve accuracy and to get around this problem of only looking at one piece of data at a time is to first perform a multiple sequence alignment among similar sequences and then to feed this alignment into the neural network. This method while still only giving one piece of data to the neural network at any one time, allows the one piece of data to contain information from

multiple sequences. Any method like this that generalizes a group of sequences should help performance. A neural network is a great device for arbitrary learning, but isn't necessarily as great at finding general patterns. Performing a generalizing step outside of the neural network training process also minimizes the effect of over-fitting since the data that the neural network sees is already a general version of the actual data. Unfortunately, this does not really solve any problems. If we can generalize outside of a neural network, why do we even need the neural network?

Ideally we would like to build some memory into our neural network. Each training instance can then be processed in multiple steps. The new instance could be compared with the elements in the memory to find similar sequences and this extra knowledge can be incorporated into the network *in addition* to the original sequence. While this significantly complicates the idea of a neural network, the current neural network design can only go so far before its accuracy stops going up.

## 5.2 Flexibility

Another major problem is with the flexibility of neural networks. There are many variables such as learning rate, encoding amino acids with various features, size of the neural net, and other parameters that are not easily changed or set to their optimal values. For example, it was shown above, that the learning rate can have a dramatic impact on performance, yet it can be very difficult to know beforehand what a reasonable value would be. As another example, different applications dictate that different amino acid features should be used, but in many cases, not even a human expert knows what features are good. A good method would ideally be able to adapt automatically to the optimal parameters, instead of requiring the user or programmer to encode that information. If we assume that the neurons in our brains work the same way that neural networks do, somehow we learn reasonable values for all of these parameters since we do not have a programmer around fiddling with our minds. The flexibility of neural networks can also be improved if connections can be added and removed dynamically

rather than having a static structure. It is not possible that anyone can know beforehand what the ideal structure of the network would be. In fact, the optimal structure undoubtedly changes over time.

### **5.3 Understandability**

Ideally, a method that could be used to perform classifications or solve other problems would have a language that would make it easy to communicate with humans. While a program that can always give the right answer is nice, it would be even more useful for us if we could understand why and how it finds the right answers. Neural networks however are not particularly intuitive or easy to understand. It is not possible to tell the difference between a trained neural network and a random set of weights. Since all nodes in one layer are connected to all nodes in the next layer, no information is present in the connections. Rather, everything is in the weights which while convenient for a computer is not easily parsed by a human. This is not that much of a problem if we think about how little we understand how the neurons in our brains work. However, we would prefer a method of representation that can be shared between human and computer.

## **6 Conclusion**

Neural networks have been shown to be a reasonable method for solving some simple problems, and are able to go relatively far very easily. But the difficulty and real test of its robustness and usefulness lies in the last 5%. Whether neural networks become useful enough to solve important biological problems, rather than the simple toy problem presented here, depends on if it is feasible to make this last step.

## 7 References

Selected references:

Hofmann, K. and Stoffel, W. (1993). *TMBASE - A database of membrane spanning protein segments*. Biol. Chem. Hoppe-Seyler 374,166.

Jones, D. T., Taylor, W. R. and Thornton, J. M. (1994). *A model recognition approach to the prediction of all-helical membrane protein structure and topology*. Biochemistry, 33(10), 3038-49.

Nielsen, H., Brunak, S. and Gunnar von Heijne. (1999). *Machine learning approaches to the prediction of signal peptides and other protein sorting signals*. Protein Engineering 12, 3-9.

Nielsen, H., Engelbrecht, J., Brunak, S., and G. von Heijne. (1997). *Identification of prokaryotic and eukaryotic signal peptides and prediction of their cleavage sites*. Protein Engineering 10, 1-6.

Rost, B. (1996). *Predicting 1D protein structure by profile based neural networks*. Meth. in Enzym, 266, 525-539.

Rost, B. (1997). *Learning From Evolution To Predict Protein Structure*. 'Bio-Computing and Emergent Computation', Sweden Sep 1-2.

Rost, B., Casadio, R., Fariselli, P. and Sander, C. (1995). *Trans-membrane helices predicted at 95% accuracy*. Protein Sci, 4(3), 521-33.

Rost, B. and Sander, C. (1993). *Improved prediction of protein secondary structure by use of sequence profiles and neural networks*. Proc Natl Acad Sci U S A, 90(16), 7558-62.

Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*.