Biochemistry

© Copyright 1992 by the American Chemical Society Volume 31, Number 32 August 18, 1992

*Perspectives in Biochemistry*

# Prediction of Structural and Functional Features of Protein and Nucleic Acid Sequences by Artificial Neural Networks

Jonathan D. Hirst and Michael J. E. Sternberg*

*Biomolecular Modelling Laboratory, Imperial Cancer Research Fund, 44, Lincoln's Inn Fields, P.O. Box 123, London WC2A 3PX, U.K.*

*Received March 12, 1992*

ABSTRACT: The applications of artificial neural networks to the prediction of structural and functional features of protein and nucleic acid sequences are reviewed. A brief introduction to neural networks is given, including a discussion of learning algorithms and sequence encoding. The protein applications mostly involve the prediction of secondary and tertiary structure from sequence. The problems in nucleic acid analysis tackled by neural networks are the prediction of translation initiation sites in *Escherichia coli*, the recognition of splice junctions in human mRNA, and the prediction of promoter sites in *E. coli*. The performance of the approach is compared with other current statistical methods.

Workers far removed from direct research in the field of artificial neural networks are now investigating applications to their own areas. An artificial neural network is, basically, a computer program that can detect patterns and correlations in data. It can learn to recognize a pattern by increasing the emphasis placed on important information and ignoring irrelevant information. The success of this methodology in the recognition and classification of patterns, and the contrast of the parallel learning algorithms with conventional serial computing, has attracted the attention of, among others, scientists interested in structural and functional analyses of protein and nucleic acid sequences.

Originally, research into neural networks was primarily motivated by a desire to model the working of the human brain. The power of the brain is presumed to come from the number of neurons, $10^{14}$, and the high degree of connectivity—$10^3$ connections (synapses) per neuron [see Hubel (1979) for an introduction to neurobiology]. The brain has thus been modeled using aggregates of simple units connected to each other. These models are limited because the numbers of neurons and connections in a neural network are orders of magnitude less than in the human brain. The neurons and the synapses themselves are not precisely modeled, and the learning procedure is not well understood. Despite these shortcomings, not only are neural networks still being used to investigate learning procedures, but the algorithms themselves are being exploited in areas that conventional computing has not been entirely successful.

Current mathematical models stem from the work of McCulloch and Pitts (1943), Hebb (1949), Rosenblatt (1962), and others. Interest in neural networks was curtailed when Minsky and Papert (1969) highlighted a major limitation of the approach, and it was not until the implementation of a new algorithm, called the back-propagation of errors (Rumelhart et al., 1986), that this limitation was widely seen to have been overcome. Although back-propagation is not a plausible model of learning in brains (Rumelhart et al., 1986; Crick, 1989), the prospect of tackling previously unsolved computational problems using the power of back-propagation, and other work in the field, including that of Kohonen (1984), Grossberg (1986), and Hopfield (1982, 1984, 1986), rekindled the excitement about neural networks. Schillen (1991) lists many of the current areas of application, including speech recognition (Sejnowski & Rosenberg, 1987) and vision (Lehky & Sejnowski, 1988), and an extensive list of references can be found in a book by Simpson (1990).

Neural networks have several features that have encouraged their application to the analysis of protein and nucleic acid sequences. They incorporate both positive and negative information—both sequences with the feature of interest and without the feature are used to train the neural network. They are able to detect second- and higher-order correlations in patterns—this approach can find more complex correlations than a method based simply on the frequency of occurrence

0006-2960/92/0431-7211$03.00/0

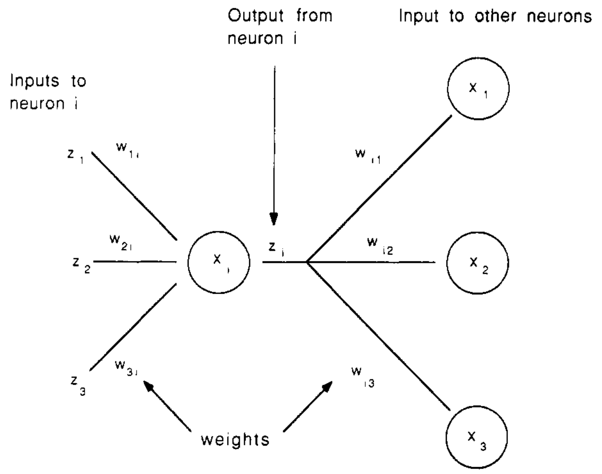

FIGURE 1: Units, $x_i$, of a neural network, based on a simplified model of a biological neuron. The weights connecting units $i$ and $j$ are denoted by $w_{ij}$. The output of a unit, $z_i$, is given by $z_i = 1/(1 + e^{-X})$, where $X = \sum_{h=1}^{n} z_h w_{hi} + \theta_i$, $n$ is the number of inputs received by the unit, and $\theta_i$ is the threshold of the unit.

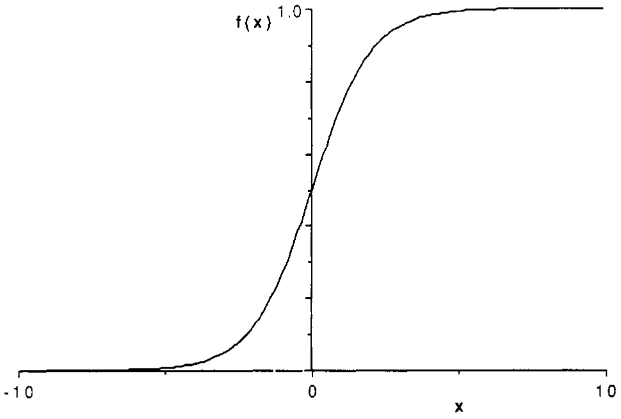

FIGURE 2: Sigmoid function, $f(x) = 1/(1 + e^{-x})$, commonly used as an output transfer function. It provides an approximate model of the firing of a neuron, by converting a high number to 1 and a low number to 0.

of residues (or bases) at certain positions. A preconceived model is not required—the neural network automatically determines which residues and which positions are important. In this paper, the applications of neural networks to the analysis of protein and nucleic acid sequences are reviewed.

## METHODOLOGY

*Introduction to Artificial Neural Networks.* A neural network consists of a number of simple, connected computational units that operate in parallel, and it can be trained to map a set of input patterns onto a set of output patterns. A unit has the basic functionality of a biological neuron: it takes signals from other units; if the sum of these signals is greater than a threshold, it produces a signal, which is passed onto other units (Figure 1). Each unit operates independently, but the units are connected to one another with a weight, which is a real number, and these weights determine the behavior of the neural network. Each unit transmits a signal to its neighbors through the connections. The value of the output signal depends upon the activation of the unit, which is a real number associated with the unit. This dependence is expressed in an output transfer function, most commonly, the sigmoid function (Figure 2). There are three types of units: input units which receive signals from external sources and send signals to other units; output units which receive signals from other units and send signals to the environment; and hidden units which have no direct contact with the environment, and hence, they receive inputs from other units and send their output signals to other units.

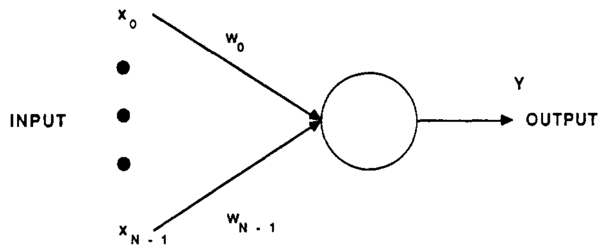

FIGURE 3: Schematic diagram of a two-layer perceptron, with *N* input units, denoted by *x*, *N* weights denoted by *w*, and one output unit, denoted by *Y*.

The architecture of a network is formed by organizing the units into layers. There can be connections between units in the same layer and connections between units in different layers. Interlayer connections can allow propagation of signals in one direction (feedforward) or in either direction (feedback). The network learns by altering the values of the weights in a well-defined manner, described by a learning rule. The general type of learning widely used in sequence analysis is supervised learning, which incorporates an external teacher and requires a knowledge of the desired responses to input signals (i.e., observations about the system). The aim is to minimize the error between the desired and computed output unit values.

The applications of neural networks to sequence analysis have involved mainly feedforward architectures with supervised learning. Even within this subset of neural networks, a number of decisions still have to be made. These include the choice of learning algorithm, the number of input units, the possible use of hidden layers, and the method of encoding sequences. Some of the more complicated neural networks can find arbitrarily complex mappings between input patterns and output classifications, but this process is poorly understood, and so the above choices are not automatic. In the following sections, these choices are considered in more detail.

*Learning Algorithms.* A neural network with no hidden layers can be trained using the perceptron algorithm (Rosenblatt, 1957). For simplicity, consider a two-layer perceptron, i.e., one with no hidden units, that decides whether an input belongs to just one of two classes, denoted A and B (Figure 3). The single output unit computes a weighted sum of the input units, subtracts a threshold, $\theta$, and converts the result to +1 or −1, using an output-transfer function. The decision rule is to respond class A if the output is +1 and class B if the output is −1. The behavior of such networks can be analyzed using a plot of the decision regions created in the multidimensional space spanned by the input variables. These decision regions specify which input values result in a class A and which result in a class B response. The perceptron forms two decision regions separated by a hyperplane (Figure 4), and the equation of the boundary line depends on the connection weights and the threshold. Rosenblatt (1962) proved for two-layer perceptrons that if the inputs presented from the two classes are separable (that is, they fall on opposite sides of a hyperplane), then the perceptron algorithm converges and positions the decision hyperplane between those two classes. Rosenblatt was unable to extend this to perceptrons with three or more layers. Two-layer perceptrons are not appropriate when classes cannot be separated by a hyperplane, as in the exclusive OR problem (Figure 5). For these nonlinearly separable problems multilayer networks trained with a more involved algorithm are required.

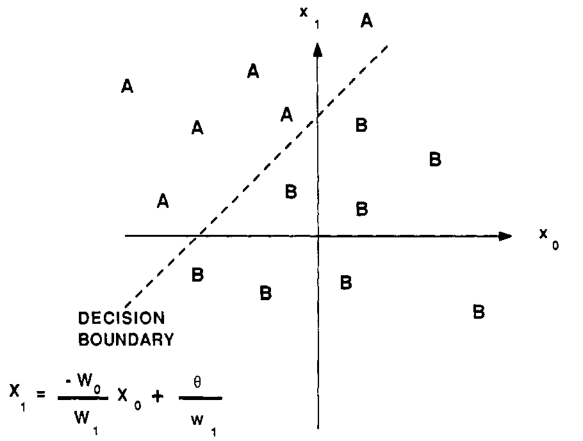

FIGURE 4: Decision boundary formed by a two-layer perceptron separating two classes, A and B, by two input coordinates, $x_0$ and $x_1$. The equation of the line is given as a fun tion of the weights, $w_0$ and $w_1$, and the threshold, $\theta$. The decision boundary can be depicted in two dimensions because there are only two input coordinates. More input units would increase the dimension of the decision space.

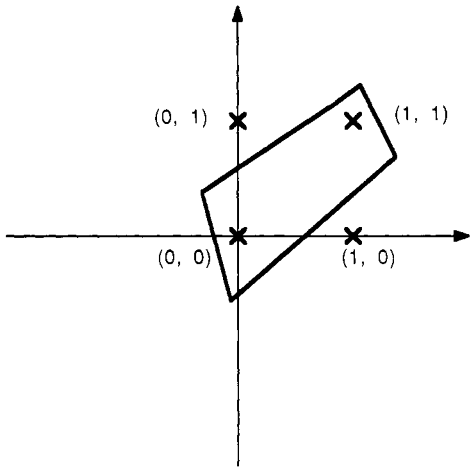

FIGURE 5: Graphical representation of the exclusive OR problem. If the two inputs are (0, 0) or (1, 1), the output is 0, and if the two inputs are (0, 1) or (1, 0), the output is 1. The decision region required to separate the two classes is schematically shown, and it cannot be a single line.

The back-propagation of errors (Rumelhart et al., 1986) is such an algorithm. It performs the input to output mapping by adjusting weight connections according to the difference between the computed and desired output unit values, to minimize a cost function, typically the squared difference between the computed output values and the desired output values, across all the patterns in the data set. The weight adjustments are derived from the change in the cost function with respect to the change in each weight. The back-propagation algorithm is powerful, because this derivation is extended to find the equation for adapting the connections between the input and hidden layers of a multilayer network, as well as the penultimate layer to output layer adjustments. The key to the extension to the hidden layer adjustments is the realization that the error of each unit in the penultimate layer is a proportionally weighted sum of the errors produced at the output layer.

*Sequence Encoding.* The input to a neuron is a number (often a binary number—the input neurons are usually on or off), and the input to a neural network is a sequence of numbers. Thus, a protein sequence or nucleic acid sequence has to be encoded from alphabetic characters to a sequence of binary numbers. There are 20 different amino acids residues—each of these is represented by 19 zeros and a single 1; for example, glycine is coded by 00000010000000000000. Such a sparse representation avoids biochemically unjustified algebraic relationships between the coded forms of the amino acid residues. The encoding of DNA sequences requires less bits, usually four. As well as binary numbers, it is possible to input real numbers (Uberbacher & Mural, 1991).

*Number of Input Units.* In the analysis of protein and nucleic acid sequences by neural networks, the effect of neighboring residues is an important aspect. For instance, a feature of an amino acid, be it secondary structure or surface accessibility, can be influenced by local neighbors (residues that are close in the sequence, as opposed to residues that are spatially close only because of the folding of the sequence). Neural networks have been used to predict features of an amino acid given its context. The length of the segment of sequence presented to the neural network is known as the window size. In protein applications, where each amino acid is represented by a 20-bit binary number, the number of input units is 20 times the window size. If the database is small, the window size and the number of hidden units may be limited by the sparse encoding.

*Hidden Units.* Two-layer neural networks will only separate linearly separable data, whereas neural networks with hidden units can classify nonlinearly separable inputs. If the classification problem is linearly separable, a neural network with hidden layers will not perform better than one without hidden units. The optimal number of hidden layers and the number of units in each hidden layer are determined empirically, although no more than one hidden layer has been used for the direct analysis of protein and nucleic acid sequences.

For secondary structure prediction, the accuracy of the network was almost independent of the number of hidden units (Qian & Sejnowski, 1988), although the learning rate became slower with less hidden units. Qian and Sejnowski concluded that the common features in the proteins are all first-order features and that the higher-order features (the information due to interactions between two or more residues) learned by the network were specific to each individual protein. For the prediction of $\beta$-turns (McGregor et al., 1989), a network with no hidden units was only marginally inferior to one with hidden units. Holbrook et al. (1990) found that the addition of hidden units gave an improvement of just 2% in the prediction of surface accessibility.

A neural network predicting promoter sites in *Escherichia coli* mRNA (Demeler et al., 1991) did not vary in performance with one to ten hidden units. Brunak et al. (1991), predicting donor and acceptor sites in mRNA, found a large increase in performance when hidden units were added, with 40 hidden units giving the best performance. Applications of neural networks to nucleic acid sequence analysis seem more likely to require hidden units and thus to exploit more fully the power of the approach.

*Interpreting Results.* In a two-layer perceptron, the output, and hence the classification of an input pattern, depends on the sum of the products of each weight and the activation of the input unit attached to it. The larger the weight, the greater the influence of its connected input unit on the final classification of the total input pattern. It is possible to identify weights that are important in a classification problem, and they can be correlated to a given residue being required at a certain position. This is often done using a graphical representation of the weights, known as a Hinton diagram. The matrix of weights is represented as a rectangle and is shaded according to magnitude and sign. The weights from

Table I: Neural Network Applications to Structural and Functional Analyses of Protein and Nucleic Acid Sequences[a]

| reference | problem | no. of hidden layers | result | comparison |
|---|---|---|---|---|
| Stormo et al., 1982a | *E. coli* mRNA translational initiation sites | 0 | 70% | 60% (Stormo et al., 1982b) |
| Nakata et al., 1985 | splice junctions in human mRNA | 0 | 73–91% | 61–74% (Fickett, 1982) |
| Nakata et al., 1988 | promoter regions in *E. coli* | 0 | 67% (perceptron)<br>75% (+other information) | |
| Lukashin et al., 1989 | promoter recognition in *E. coli* | 1 | 94–99%<br>2–6% false positives | |
| O'Neill, 1991 | *E. coli* promoter recognition | 1 | 80%<br>0.1% false positives | 70% (O'Neill, 1989) |
| Demeler et al., 1991 | *E. coli* promoter recognition | 1 | 98% | 77% (O'Neill, 1989) |
| Brunak et al., 1991 | human mRNA donor and acceptor sites | 1 | 95%<br>false + ves 0.4% | 95% (Staden, 1984)<br>false + ves 0.7% |
| Uberbacher & Mural, 1991 | protein-coding regions in human DNA | 2 | 92%<br>8% false positives | |
| Qian & Sejnowski, 1988 | secondary structure of globular proteins | 1 | 64% | 53% (Garnier et al., 1978)<br>50% (Chou & Fasman, 1978)<br>50% (Lim, 1974)<br>59% (Levin et al., 1986) |
| Bohr et al., 1988 | protein secondary structure – $\alpha$-helices | 1 | 73% | |
| Holley & Karplus, 1989 | protein secondary structure | 1 | 63% | 48% (Chou & Fasman, 1978)<br>55% (Garnier et al., 1978)<br>54% (Lim, 1974) |
| McGregor et al., 1989 | $\beta$-turns in proteins | 1 | 26% | 21% (Wilmot & Thornton, 1988) |
| Bohr et al., 1990 | 3-D structure of protein backbone | 1 | 3.0 Å rms | 74% homology |
| Wilcox et al., 1990 | protein tertiary structure | 1 | no generalization | |
| Kneller et al., 1990 | protein secondary structure | 1 | 79% (all $\alpha$)<br>70% (all $\beta$)<br>64% ($\alpha$–$\beta$) | 64% (Qian & Sejnowski, 1988) |
| Muskal et al., 1990 | disulfide-bonding state of cysteine | 0 | 81% | |
| Holbrook et al., 1990 | surface exposure of amino acids | 1 | 72% (binary model)<br>54% (ternary model) | 70% (Rose, 1985) |
| Hirst & Sternberg, 1991 | An ATP/GTP-binding motif | 0 | 78% | 80% (Hirst & Sternberg, 1991) |

[a] This table summarizes the problem tackled, the number of hidden layers used, the result, and the comparison, if any, made by the authors to other methods. Nucleic acid applications are given in chronological order followed by protein applications in chronological order.

a neural network with hidden units cannot be interpreted in this way, because their influence on the final output is only indirect through the hidden layer.

A confidence level can be assigned to predictions by a neural network based on the magnitude of the activity of the output unit. A prediction with a high output unit activity is more likely to be correct than one with a lower output unit activity. A significance filter can be used to improve the predictive accuracy of a network, but the number of predictions made decreases (Holley & Karplus, 1989; Muskal et al., 1990).

## APPLICATIONS

Table I summarizes the performances of neural networks applied to various problems and shows the authors' comparisons of the neural network approach with other methods.

*Translational Initiation Sites in E. coli.* The first application of a neural network model to sequence analysis was by Stormo et al. (1982a), who used a perceptron algorithm with no hidden layers to predict translational initiation sites in *E. coli.* Their goal was to define nucleotides that may play a role in the selection of initiation codons by the ribosomes of *E. coli.* The training set consisted of 124 known gene beginnings and 167 false beginnings, as identified by another method (Stormo et al., 1982b). In a test set of ten genes, the perceptron correctly predicted six of the gene beginnings and incorrectly identified five false beginnings. A rule-based approach (Stormo et al., 1982b) only predicted five true gene beginnings and identified twelve false ones.

*Splice Junctions.* There are two basic approaches to the computer prediction of protein-coding regions in DNA. First, coding function constrains a nucleotide sequence, so coding and noncoding sequences can be distinguished using patterns of codon usage (Staden & McLachlan, 1982; Gribskov et al., 1984), positional mono- and oligonucleotide frequencies, and weak three periodicity (Fickett, 1982; Staden, 1984a). Second, the nonuniformity of nucleotide distribution near start codons and splicing sites can be used (Shapiro & Senepathy, 1987; Gelfand, 1989). The more successful prediction schemes have combined the two approaches.

Nakata et al. (1985) predicted splice junctions in human mRNA sequences by discriminant analysis of information including consensus sequence patterns around splice junctions, free energy of snRNA and mRNA base pairing, and base composition and periodicity. Discriminant analysis is a statistical technique based on a comparison of distribution profiles of certain attributes (discriminant variables) for true and false sequences. When the distributions are well separated, the attributes may be used for distinguishing true and false sequences. Information about the consensus sequence was provided by the output activities of two two-layer perceptrons, one trained to recognize exon/intron boundaries and the other intron/exon boundaries. The output activity was termed the perceptron value by Nakata et al. (1985), and it reflects a degree of similarity of the input pattern to the consensus sequence patterns of true sequences. The perceptron was more accurate than Fickett's function, a combined measure of base composition and periodicity (Fickett, 1982), for predicting the start of coding regions (84% versus 74%), the end of coding regions (78% versus 61%), the exon/intron boundary (91% versus 66%), and the intron/exon boundary (82% versus 65%).

Brunak et al. (1991) used neural networks trained by back-propagation to tackle both approaches to the problem of distinguishing coding and noncoding regions and predicted

Table II: Comparison of the Prediction of Exon/Intron Boundaries in Human DNA by a Neural Network, a Weight Matrix Method Based on Data Available When the Method Was First Developed, and a Weight Matrix Method Derived from More Recent Data

| method | no. of false positives at 90% detection of true sites | no. of false positives at 95% detection of true sites |
|---|---|---|
| neural network (Brunak et al., 1991) | 28 | 34 |
| weight matrix (Staden, 1984b) | 49 | 83 |
| weight matrix from training data | 29 | 44 |

human mRNA donor and acceptor sites in DNA, based on the combined result. Multilayer neural networks trained on small sequence segments were used to identify intron/exon and exon/intron boundaries. Other neural networks were trained on large sequence segments to predict the transition between the coding and noncoding regions. The large neural network correctly identified 70% of exons with 2.5% false positives. The neural network trained to recognize exon/intron boundaries correctly detected 94% of unseen boundaries with 0.1% false identification. Intron/exon detection was 87% accurate with 0.2% false identification. The combined method detected 95% of true exon/intron and intron/exon boundaries with false identification at 0.1% and 0.4%, respectively. The weight matrix method of Staden (1984b) gave more false positives (0.7%) for the same level of detection of true boundaries.

The rapid growth of databases necessitates the comparison of neural network results with those of statistical methods developed on the same data to demonstrate that the difference in performance is due to the methodology and not to the database size. We illustrate this by a comparison of the prediction of exon/intron boundaries between weight matrices derived from current data and from the data available when the weight matrix method was first developed. The weight matrix method performs better if the matrix is derived from the more recent data (Table II), although it still does not achieve the accuracy of the neural network.

In contrast to the above applications where neural networks have been trained to examine sequence data directly, a neural network was used by Uberbacher and Mural (1991) to examine sequences indirectly. A network of seven input units, two hidden layers of fourteen and five units and an output unit, was trained to locate protein-coding regions in human DNA sequences. Input to the network was a vector containing the values of seven sensor algorithms calculated for positions at intervals of ten bases along the sequences of interest. The seven sensor algorithms were a frame bias matrix, based on the nonrandom frequency with which each of the four bases occupies each of the three positions within codons; Fickett's function (Fickett, 1982); the dinucleotide fractal dimension (Hsu & Hsu, 1990); and four analyses based on the frequency of occurrence of 6-tuple "words" in the nucleotide sequences. The approach identifies 90% of coding exons greater than 100 bases with less than one false positive coding exon indicated per five coding exons indicated. Shorter exons are harder to detect—only 47% of exons less than 100 bases were detected.

*Promoter Sites in E. coli.* Nakata et al. (1988) adapted the discriminant analysis method used to predict splice junctions (Nakata et al., 1985) to predict promoter regions in *E. coli*. The attributes used for discrimination were the accuracy of consensus sequence patterns measured by the perceptron algorithm, the thermal stability map, the base composition, and the Calladine–Dickerson rules for helical twist, roll angle, torsion angle, and propeller twist angle (Dickerson, 1983). The perceptron on its own predicted promoter regions in *E. coli* with 67% accuracy. Inclusion of the other information in the prediction algorithm increased the predictive ability to 75%.

Lukashin et al. (1989) and Demeler et al. (1991) have both tackled the same problem with more complicated neural networks, obtaining accuracies of 94%–99%, with a 2%–6% chance of false identification. O'Neill (1991) trained a back-propagating network to recognize 80% *E. coli* promoters of the 17 base spacer class with a false positive rate below 0.1%. Lukashin et al. (1989) used two three-layer neural networks to examine the two conservative hexanucleotides that occur 10 base pairs and 35 base pairs upstream from the transcription starting point of the promoter. The outputs from the two networks provided the input to a final unit, whose output corresponded to the classification of the sequence. The training set used by O'Neill (1991) included 5148 58-base sequences drawn from 39 promoters and 4000 random sequences which were 60% in A + T. The group of true sequences was expanded by permuting all possible single base changes in positions other than those known to harbor promoter point mutations. Demeler et al. (1991) optimized the predictive ability of a three-layer neural network trained by back-propagation. This was done by varying the encoding scheme (two bit and four bit), the number of hidden units (one to ten), the ratio of promoter sequences to nonpromoter sequences (1:1 to 1:20), and the extent of training (training was stopped when the error had reached 1, 0.1, 0.01, 0.001, and 0.0001). The combination of parameters that gave the best results on the test set was selected as the optimized neural network. O'Neill (1989), who used six empirically developed tests to filter out false positives, identified, by a consensus sequence match algorithm, 77% of the fully characterized promoters of Hawley and McClure (1983) but noted that the search produces many false positives and also noted that the performance was being assessed on sequences used in training the classification method.

*Protein Secondary Structure Prediction.* Similar approaches were used by Qian and Sejnowski (1988) and Holley and Karplus (1989) to develop neural networks that predicted the secondary structure of an amino acid within a protein (as either $\alpha$, $\beta$, or random) with an accuracy of 63%–64%. A three-layer neural network was trained by back-propagation on contiguous segments of protein sequence to assign the secondary structure, as defined by Kabsch and Sander (1983), of the amino acid at the center of the segment. A window size of 13 was found to be optimal for protein secondary structure prediction (Qian & Sejnowski, 1988) as smaller windows exclude information useful for prediction. There is little useful information outside the window of 13, and irrelevant weights are deleterious to the performance of the network. The accuracy of the neural network was compared with the sequence similarity method of Levin et al. (1986) and with rule-based (Lim, 1974) and statistical methods (Chou & Fasman, 1979; Garnier et al., 1978) and was better in all cases. However, more recently, statistical methods have been improved by Gibrat et al. (1987) and Ptitsyn and Finkelstein (1989), who both obtained an accuracy of 63%. A machine learning method (King & Sternberg, 1990) was 60% accurate. Recent reviews (von Heijne, 1991; Garnier, 1991) suggest that 65% appears to be the maximum attainable performance of a variety of methods of secondary structure prediction.

*Specific Protein Secondary Structure Prediction.* A three-layer neural network trained by back-propagation (Bohr et al., 1988) achieved an accuracy of 73% when predicting the transmembrane $\alpha$-helices in rhodopsin, which was compared qualitatively with the prediction of Argos et al. (1982). This was a two-state prediction ($\alpha$ or not $\alpha$), and the result is not directly comparable to three-state predictions ($\alpha$, $\beta$, or random), as noted by Petersen et al. (1990). Kneller et al. (1990) provided an analysis which shows that a two-state prediction will be 12% better than a three-state prediction, purely because of the change in the number of classes. Thus, to classify residues as either helical or nonhelical at 73% is comparable to a three-state prediction of 61%, which is obtainable by several methods. Using a similar network, Kneller et al. (1990) report an even greater increase in predictive accuracy by preclassifying proteins as all-$\alpha$, all-$\beta$, and mixed $\alpha\beta$. For all-$\alpha$ a 16% improvement (79% compared to 63%) over three-state predictions was attained. They note, however, that their extra 4% improvement was mostly due to the inclusion of homologous proteins in the testing set. When proteins with a greater than 40% sequence identity were removed, the prediction accuracy fell to 76%.

McGregor et al. (1990) improved the prediction of $\beta$-turns from 21% (Wilmot & Thornton, 1988) to 26%. $\beta$-turns are a specific class of chain reversals localized over a four-residue sequence (Richardson, 1981). Wilmot and Thornton (1988) distinguished seven types of turn and one miscellaneous class. The neural network of McGregor et al. (1990), trained using back-propagation, classified four residue segments into four categories: type I turns, type II turns, nonspecific turns, and nonturns.

*Tertiary Protein Structure Prediction.* Bohr et al. (1990) used a neural network to predict the three-dimensional structure of rat trypsin. The output of the neural network was a distance matrix. After steepest descent minimization of the neural network prediction, the predicted structure was within 3 Å rms of the crystal structure. A window size, and hence the width of the diagonal band of the distance matrix, was 61. The training set consisted of the sequences and distance matrices of 13 proteases, including trypsin and subtilisin. There is a significant degree of homology between rat trypsin and the other trypsins in the training set. The binary distance matrix that was generated for rat trypsin used bovine pancreatic $\beta$-trypsin as the starting configuration, which is 74% homologous with rat trypsin. The serine proteases have also been modeled by homology (Greer, 1990). Hubbard and Blundell (1987) superposed pairwise seven different serine proteases giving 21 comparisons. The rms between any two serine proteases ranged from 0.63 to 1.35 Å, with the homology between the sequences varying 19%–66%. The neural network approach thus has yet to match traditional methods of modeling by homology.

Wilcox et al. (1990) found tertiary structure prediction using a neural network very problematic. The training set of 15 proteins, of 140 residues or less, intentionally included some homologous proteins. Each amino acid was encoded using hydrophobicity values adapted from Liebmann et al. (1985) that were normalized into the range −1 to +1. The input layer was thus 140 units. A hidden layer of 15–240 units was used, and the output layer was a window for distance matrices composed of 19 600 (140 by 140) units. The test set consisted of nine proteins, each of which was homologous (either in sequence or function) to one or more of the training proteins. Although the network performed well on the training set (a mean squared error below 2%), there was very little generalization, and this was attributed partly to the small size and the heterogeneity of the training set.

*Other Protein Structure Applications.* Holbrook et al. (1990) used neural networks to classify amino acid residues as either buried or exposed with an accuracy of 72% and as either buried, intermediate, or exposed with an accuracy of 54%. The training classifications were based on fractional accessibilities derived from solvent accessibilities, calculated with the DSSP program (Kabsch & Sander, 1983) and the standard, fully exposed values (Rose et al., 1985). Flanking residues were found to have a small effect on the surface exposure of an amino acid residue. For the two-state problem (buried/exposed) Rose et al. (1985) correctly predicted the surface exposure of 70% of residues using the fractional residue accessibilities.

A two-layer perceptron was trained by Muskal et al. (1990) to predict whether cysteine residues were disulfide-bonded or not. Only the flanking residues were presented to the network, as both classes had a cysteine as the central residue. An accuracy of 81% was achieved, but this was not quantitatively compared with another method.

The importance of a rigorous comparison of the performance of a neural network with the performance of a similarly sophisticated statistical method was illustrated by Hirst and Sternberg (1991), who investigated the recognition of an ATP/GTP-binding motif. A motif is a well-conserved group of amino acids within a specific region of a protein sequence. Other residues outside of this region are usually poorly conserved, so there is low overall homology with other proteins containing the same motif. For this reason, a motif-searching program such as PROMOT (Sternberg, 1991) will generate many false positives. In a scan of the SWISSPROT release 14 database by PROMOT (Sternberg, 1991) for the PROSITE (Bairoch, 1990) ATP/GTP-binding motif, denoted by the one-letter amino acid code [AG]-X-X-X-X-G-K-[ST] (Walker et al., 1982), where X is any residue and [YZ] means "either Y or Z", 193 21-residue segments were correctly identified as exhibiting the motif, along with 156 false positives. These segments, minus the four motif-defining residues which were common to both classes, were the data on which the neural network and a comparative statistical method based on the motif-searching program of AMPS (Barton & Sternberg, 1990) were developed. The two-layer perceptron recognized an ATP/GTP-binding motif with 78% accuracy, but the statistical method was 80% accurate on the same data.

## CONCLUSIONS

Studies that rigorously compare the performance of a neural network with the performance of another state-of-the-art statistical method on the same data set are required to assess the utility of the neural network approach. The assessment is further complicated when one realizes that there are many empirical parameters in the neural network approach (aside from the weights) that can be optimized. Some of these parameters include the window size, the number of hidden units, the learning rate, the sampling of the data, the ratio of true to false examples, and the definition of convergence. While the optimization of these parameters may be permissible, the final choices are sometimes just those that give the best result. It is also unclear, in some cases, whether the optimization has been performed on the test or training set. The test set and training set must be clearly distinct, and there should be no homology between the two. The existence of homology between the test and training sets can increase the performance of the network, by the network learning homology rules, as well as detecting the general features that are of interest.

There are several technical difficulties that can arise in the implementation of a neural network. Kneller et al. (1990)

reported that, during the training of their neural network to predict secondary structure of $\alpha/\beta$ proteins, the network converged to a set of weights that predicted no $\beta$ structure. This may have been due to the network becoming trapped in a local minimum. Whereas the perceptron convergence theorem (Rosenblatt, 1957) guarantees that a two-layer perceptron will converge if the classes are linearly separable, no such theorem exists for back-propagation. Training a neural network by back-propagation is similar to other minimization procedures in that local minima can be a problem. To overcome this, Kneller et al. (1990) set the initial weights, so that the starting point exaggerated the tendencies of $\beta$ structure to be detected.

Even if the training procedure is straightforward, overtraining can occur. As the number of free variables (weights and biases) in the network approaches the number of data elements, the network learns to reproduce most of the training set but in the process loses some of its ability to generalize, and the prediction accuracy goes down. This is known as memorization. By back-propagating the error signal only when the difference between the actual and desired values of the outputs was greater than 0.1, Qian and Sejnowski (1988) ensured that their network did not overlearn on inputs it was already getting correct.

In the application of neural networks to protein structure problems, the attraction must be the possibility that a neural network with hidden units will be able to extract higher than first-order information. Neural networks should be able to learn rules including complex conditional statements, such as "the secondary structure is predicted to be helical if either leucine or valine are neighbors to the residue but random coil if they are both neighbors". Since rules similar to this one are relevant to secondary structure prediction schemes, it has been hoped that the hidden unit layer would be important for such problems. However, several workers have reported that the database is simply too small for second-order features to be exhibited as general features (Qian & Sejnowski, 1988; Rooman & Wodak, 1988). Thus, at present, neural networks are only able to use the first-order information that other predictive methods use, and there is no evidence to suggest that they can use this information better than these other methods. For example, there are now several methods for predicting protein secondary structure that perform as well as the neural network approach (Gibrat et al., 1987; King &Sternberg, 1990; Ptitsyn & Finkelstein, 1989).

Nucleic acid sequence analysis by neural networks appears to have been more successful than protein applications. This may in part be due the greater amount of nucleic acid sequence data and the fact that the bases of DNA can be encoded into a network with only 4 bits, whereas the encoding of amino acids requires 20 bits. The approach of Uberbacher and Mural (1991) perhaps suggests that the indirect analysis of protein sequences by neural networks may be more successful than the direct analyses so far, which have been limited by the size of the database being insufficient to exhibit in a generalizable way information higher than first order.

The applications of neural networks to problems in protein sequence analysis, although interesting, have not yielded significant improvements over other current methodologies. However, the success of the approach in the analysis of nucleic acid sequences and in other fields suggests that as the protein structure database grows and perhaps with the incorporation of other information, the power of neural networks may be exploited in the analysis of protein sequences.

## REFERENCES

Argos, P., Rao, J. K. M., & Hargrave, P. A. (1982) *Eur. J. Biochem. 128*, 565–575.

Bairoch, A. (1990) *Prosite: a Dictionary of Protein Sites and Patterns*, 5th ed., Department de Biochimie Medicale, Universite de Geneve, Geneva.

Barton, G. J., & Sternberg, M. J. E. (1990) *J. Mol. Biol. 212*, 389–402.

Bohr, H., Bohr, J., Brunak, S., Cotterill, R. M. J., Lautrup, B., Norskov, L., Olsen, O. H., & Petersen, S. B. (1988) *FEBS Lett. 241*, 223–228.

Bohr, H., Bohr, J., Brunak, S., Cotterill, R. M. J., Fredholm, H., Lautrup, B.,& Petersen, S. B. (1990) *FEBS Lett. 261*, 43–46.

Brunak, S., Engelbrecht, J., & Knudsen, S. (1991) *J. Mol. Biol. 220*, 49–65.

Chou, P. Y., & Fasman, G. D. (1974) *Biochemistry 13*, 222–245.

Crick, F. (1989) *Nature 337*, 129–132.

Demeler, B., & Zhou, G. (1991) *Nucleic Acids Res. 18*, 1593–1599.

Dickerson, R. E. (1983) *J. Mol. Biol. 166*, 419–441.

Fickett, J. W. (1982) *Nucleic Acids Res. 10*, 5303–5318.

Garnier, J. (1990) *Biochimie 72*, 513–524.

Garnier, J., Osguthorpe, D. J., & Robson, B. (1978) *J. Mol. Biol. 120*, 97–120.

Gelfand, M. S. (1989) *Nucleic Acids Res. 17*, 6369–6382.

Gibrat, J. F., Garnier, J., & Robson, B. (1987) *J. Mol. Biol. 198*, 425–443.

Greer, J. (1990) *Proteins 7*, 317–334.

Gribskov, M., Devereux, J., & Burgess, R. R. (1984) *Nucleic Acids Res. 12*, 539–549.

Grossberg, S. (1986) *The Adaptive Brain*, Vols. I and II, Elsevier Science Publishers, New York.

Hawley, D. K., & McClure, W. R. (1983) *Nucleic Acids Res. 11*, 2237–2255.

Hebb, D. (1949) *Organization of Behaviour*, John Wiley & Sons, New York.

Hirst, J. D., & Sternberg, M. J. E. (1991) *Protein Eng. 4*, 615–623.

Holley, L. H., & Karplus, M. (1989) *Proc. Natl. Acad. Sci. U.S.A. 86*, 152–156.

Hopfield, J. J. (1982) *Proc. Natl. Acad. Sci. U.S.A. 79*, 2554–2558.

Hopfield, J. J. (1984) *Proc. Natl. Acad. Sci. U.S.A. 81*, 3088–3092.

Hopfield, J. J., & Tank, D. W. (1986) *Science 233*, 625–633.

Hsu, K. J., & Hsu, A. J. (1990) *Proc. Natl. Acad. Sci. U.S.A. 87*, 938–941.

Hubbard, T. J. P., & Blundell, T. L. (1987) *Protein Eng. 1*, 159–171.

Hubel, D. H. (1979) *Sci. Am. 241*, 39–46.

Kabsch, W., & Sander, C. (1983) *Biopolymers 22*, 2577–2637.

King, R. D., & Sternberg, M. J. E. (1990) *J. Mol. Biol. 216*, 441–457.

Kneller, D. G., Cohen, F. E., & Langridge, R. (1990) *J. Mol. Biol. 214*, 171–182.

Kohonen, T. (1984) *Self-Organization and Associative Memory*, Springer-Verlag, Berlin.

Lehky, S. R., & Sejnowski, T. E. (1988) *Nature 333*, 452–454.

Levin, J. M., Robson, B., & Garnier, J. (1986) *FEBS Lett. 205*, 303–308.

Liebman, M. N., Venanzi, C. A., & Weinstein, H. (1985) *Biopolymers 24*, 1721–1758.

Lim, V. I. (1974) *J. Mol. Biol. 88*, 873–894.

Lukashin, A. V., Anshelevich, V. V., Amirikyan, B. R., Gragerov, A. I., & Frank-Kamenetskii, M. D. (1989) *J. Biomol. Struct. Dyn. 6*, 1123–1133.

McCulloch, W., & Pitts, W. (1943) *Bull. Math. Biophys. 5*, 115–133.

McGregor, M. J., Flores, T. P., & Sternberg, M. J. E. (1989) *Protein Eng. 2*, 521–526.

Minsky, M., & Papert, S. (1969) *Perceptrons*, MIT Press, Cambridge.

Muskal, S. M., Holbrook, S. R., & Kim, S.-H. (1990) *Protein Eng. 3*, 667–672.

Nakata, K., Kanehisa, M., & DeLisi, C. (1985) *Nucleic Acids Res. 13*, 5327–5340.

Nakata, K., Kanehisa, M., & Maizel, J. V. (1988) *Comput. Appl. Biosci. 4*, 367–371.

O'Neill, M. C. (1989) *J. Biol. Chem. 264*, 5522–5530.

O'Neill, M. C. (1991) *Nucleic Acids Res. 19*, 313–318.

Petersen, S. B., Bohr, H., Bohr, J., Brunak, S., Cotterill, R. M. J., Fredholm, H., & Lautrup, B. (1990) *Trends Biotechnol. 8*, 304–308.

Ptitsyn, O. B., & Finkelstein, A. V. (1989) *Protein Eng. 2*, 443–447.

Qian, N., & Sejnowski, T. J. (1988) *J. Mol. Biol. 202*, 865–884.

Rooman, M. J., & Wodak, S. J. (1988) *Nature 335*, 45–49.

Rose, G. D., Geselowitz, A. R., Lesser, G. J., Lee, R. H., & Zehfus, M. H. (1985) *Science 229*, 834–838.

Rosenblatt, F. (1957) *The perceptron: A perceiving and recognizing automation*, project PARA, Cornell Aeronautical Laboratory Report, 85-460-1.

Rosenblatt, F. (1962) *Principles of Neurodynamics*, Spartan Books, Washington.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986) *Nature 323*, 533–536.

Schillen, T. B. (1991) *Comput. Appl. Biosci. 7*, 417–430.

Sejnowski, T. E., & Rosenberg, C. R. (1987) *Complex Syst. 1*, 145–168.

Shapiro, M. B., & Senepathy, P. (1987) *Nucleic Acids Res. 15*, 7155–7173.

Simpson, P. F. (1990) *Artificial Neural Systems*, Pergamon Press, New York.

Staden, R. (1984a) *Nucleic Acids Res. 12*, 551–567.

Staden, R. (1984b) *Nucleic Acids Res. 12*, 505–519.

Staden, R., & McLachlan, A. D. (1982) *Nucleic Acids Res. 10*, 141–156.

Sternberg, M. J. E. (1991) *Comput. Appl. Biosci. 1*, 256–260.

Stormo, G. D., Schneider, T. D., Gold, L., & Ehrenfeucht, A. (1982a) *Nucleic Acids Res. 10*, 2997–3011.

Stormo, G. D., Schneider, T. D., & Gold, L. M. (1982b) *Nucleic Acids Res. 10*, 2971–2996.

Uberbacher, E. C., & Mural, R. J. (1991) *Proc. Natl. Acad. Sci. U.S.A. 88*, 11261–11265.

von Heijne, G. (1991) *Eur. J. Biochem. 199*, 253–256.

Walker, J. E., Saraste, M., Runswick, M. J., & Gay, N. J. (1982) *EMBO J. 1*, 945–951.

Wilcox, G. L., Poliac, M., & Liebman, M. N. (1990) *Tetrahedron Comput. Methodol. 3*, 191–211.

Wilmot, C. M. & Thornton, J. M. (1988) *J. Mol. Biol. 203*, 221–232.