J. Mol. Biol. (1992) **225**, 713–727

Predicting Protein Secondary Structure Content

A Tandem Neural Network Approach

Steven M. Muskal† and Sung-Hou Kim

Department of Chemistry and Lawrence Berkeley Laboratory
University of California, Berkeley, CA 94720, U.S.A.

(Received 5 August 1991; accepted 9 January 1992)

A priori knowledge of secondary structure content can be of great use in theoretical and experimental determination of protein structure. We present a method that uses two computer-simulated neural networks placed in "tandem" to predict the secondary structure content of water-soluble, globular proteins. The first of the two networks, NET1, predicts a protein's helix and strand content given information about the protein's amino acid composition, molecular weight and heme presence. Because NET1 contained more adjustable parameters (network weights) than learning examples, this network experienced problems with memorization, which is the inability to generalize onto new, never-seen-before examples. To overcome this problem, we designed a second network, NET2, which learned to determine when NET1 was in a state of generalization. Together, these two networks produce prediction errors as low as 5·0% and 5·6% for helix and strand content, respectively, on a set of protein crystal structures bearing little homology to those used in network training. A comparison between three other methods including a multiple linear regression analysis, a non-hidden-node network analysis and a secondary structure assignment analysis reveals that our tandem neural network scheme is, indeed, the best method for predicting secondary structure content. The results of our analysis suggest that the knowledge of sequence information is not necessary for highly accurate predictions of protein secondary structure content.

Keywords: secondary structure content; neural network; memorization; protein structure prediction; protein folding

1. Introduction

While it is generally accepted that an amino acid sequence defines a protein's structure (Anfinsen, 1973), our present knowledge is still insufficient to predict correctly the three-dimensional fold of a protein. Though model-building procedures offer some insight into protein structure, unless there exists a highly homologous sequence(s) of known structure (Blundell *et al.*, 1987, 1988; Greer, 1981, 1990), a less than satisfactory secondary structure prediction (Chou & Fasman, 1974; Garnier *et al.*, 1978; Levin & Garnier, 1988; Lim, 1974; Qian & Sejnowski, 1988; Holley & Karplus, 1989) will have to suffice. With the hope of improving secondary structure prediction performance, some have included *a priori* information of protein function (Nishikawa & Ooi, 1982; Nishikawa *et al.*, 1983), protein class (Kneller *et al.*, 1990) and overall secondary structure content (Garnier *et al.*, 1978). Because the *a priori* knowledge of secondary structure content can provide useful boundary conditions for the theoretical as well as the experimental determination of protein structure, we have focused on predicting secondary structure content utilizing readily available information such as amino acid sequence, amino acid composition, molecular weight and heme group presence or absence. Our primary tool has been the computer-simulated neural network.

Computer-simulated neural networks have recently gained much attention (Crick, 1989). The application of neural network models toward a variety of problems associated with protein structure prediction (Qian & Sejnowski, 1988; Holley & Karplus, 1989; Kneller *et al.*, 1990; Bohr *et al.*, 1988; McGregor *et al.*, 1989; Bohr *et al.*, 1990; Holbrook *et al.*, 1990; Muskal *et al.*, 1990) provides testimony to their versatility. However, because a supervised hidden-node network can be trained to map any set of input patterns to any set of output patterns, feed-forward network computing often falls into the trap of providing a set of parameters that perform

† Present address: Molecular Design Ltd., 2132 Farallon Drive, San Leandro, CA 94577, U.S.A.

0022-2836/92/110713-15 $03.00/0

very well on the database of learning examples, but very poorly on examples outside the learning set. In neural network terminology, this is called "memorization". In general, memorization will result when the number of adjustable parameters (network weights) is large relative to the number of learning examples or observations.

In our approach to predicting secondary structure content, we were forced to confront the issue of memorization because our hidden-node neural networks contained a large number of weights relative to learning examples. Network training is an iterative process. Therefore, neural networks typically learn general features early on and absorb more specific details in the later stages of training (Hertz *et al.*, 1991). Memorization can be thought of as overloading on specifics to the extent that a neural network loses both its interpolation and extrapolation ability. At this point, a network becomes nothing more than a database look-up device.

To address this issue of memorization, we employed a second neural network (NET2), which learned to determine when our first network (NET1) was in a state of generalization. This "tandem" network approach not only outperforms other means of predicting secondary structure content, such as counting secondary structure elements from a secondary structure prediction scheme (Qian & Sejnowski, 1988; Pascarella & Bossa, 1989) and a multiple linear regression analysis (Davies, 1964; Krigbaum & Knutton, 1973), but also is comparable to the accuracy of the most popular method of secondary structure assignment given knowledge of protein structure (Kabsch & Sander, 1983).

2. Methods

(a) *Database*

The database of 104 protein crystal structures similar to those chosen by Qian & Sejnowski (1988) and Kneller *et al.* (1990) was used to optimize our networks (Table 1). We selected 15 additional proteins that have secondary structure content ranging from nearly all helix to all strand (Table 2). We shall refer to the database of 104 crystal structures, which was used for network optimizations, as OPTBASE† and the database of 15 crystal structures, which was used for final evaluation, as EVALBASE. Homologies between proteins within OPTBASE were desired since it was necessary to learn the effects of small changes in amino acid composition, molecular weight and heme group presence with respect to secondary structure content. It should be noted, however, that proteins in OPTBASE have little homology with those found in EVALBASE (see Table 2).

In both databases, amino acid compositions were normalized to values between 0·0 and 1·0, molecular weights were divided by 1×10^4, the presence or absence of bound heme groups (determined by HETATM records in the protein databank entries) were represented by 1·0 and 0·0, respectively, and the secondary structure assignments determined by DSSP (⟨H⟩: $\alpha+3_{10}$-helix; ⟨E⟩: β-strand (Kabsch & Sander, 1983)) were counted and normalized to values between 0·0 and 1·0. While DSSP was known to be rather conservative in its secondary structure assignment, we deemed it necessary to quantify the differences between assignments by DSSP and those found in the HELIX and SHEET records provided by the authors. For this analysis, we required that entries that contained HELIX and/or SHEET records did not claim to use DSSP for secondary structure assignments and had clear, non-redundant assignments for each chain considered in Table 1. Table 3 shows the average difference and standard deviation between assignments by DSSP and authors for 80 proteins in OPTBASE. We suspect the true differences to be slightly larger since it is likely that some of the authors actually used DSSP for their secondary structure assignments without noting it in the REMARK records. This would result in an artificially high number of identical matches between assignments by authors and DSSP, therefore decreasing the average difference.

(b) *Neural network training procedure*

The process of training a feed-forward network begins with a set of input/output patterns (the "training" or "learning" set). For our network that predicted secondary structure content (NET1), the input pattern was a string of real numbers representing a protein's amino acid composition, molecular weight and heme group presence or absence and the output pattern was the protein's secondary structure content (Fig. 1(a)). Network training then involved determining a set of weights that minimized the error between the network's prediction and the true secondary structure content for each protein found in OPTBASE. More precisely, given the following definitions:

$i = 1 \ldots \text{ninputs};$
$j = 1 \ldots \text{nhidden};$
$k = 1 \ldots \text{noutputs};$
$e = 1 \ldots \text{numexamp};$

I_{ie}: input value i, for example, e (i.e. amino acid comp., mol. wt., heme);
O_{ke}: network output value k, for example, e (i.e. predicted %helix, %strand);
T_{ke}: target output value , for example, e (i.e. observed %helix, %strand);
ε: learning rate;

the goal of network training is to minimize the following equation, referred to as the totalerror:

$$E(w) = (1/2 \cdot \text{numexamp}) \sum_e \sum_k (T_{ke} - O_{ke})^2, \qquad (1)$$

where w is a vector representation of the network weights.

Our networks were fully connected. Therefore, a hidden-node network had weights connecting input-to-hidden and hidden-to-output layers, as well as a bias term for each hidden and output node:

WtI_H_{ji}: weights connecting input-to-hidden nodes;
WtH_O_{kj}: weights connecting hidden-to-output nodes;
$BiasHid_j$: hidden node biases;
$BiasOut_k$: output node biases;

† Abbreviations used: OPTBASE, database of 104 protein crystal structures used for optimization; EVALBASE, database of 15 protein crystal structures used for evaluation; DSSP, (Kabsch & Sander, 1983); m.l.r., multiple linear regression.

Table 1

OPTBASE: database of protein(s) used to optimize secondary structure composition prediction methods

Code	Protein name	Resolution	*Ni*	%*H*	%*E*
1PPD	2-Hydroxyethylthiopapain-crystal form D	2·0		26·4	17·0
4APE	Acid proteinase (endothiapepsin)	2·1		9·4	45·8
2APP	Acid proteinase (penicillopepsin)	1·8		13·9	45·5
2APR	Acid proteinase (rhizopuspepsin)	1·8		13·8	44·9
2ACT	Actinidin (sulfhydryl proteinase)	1·7		30·3	18·3
1ACX	Actinoxanthin	2·0		0·0	43·9
3ADK	Adenylate kinase	2·1		54·6	12·9
1CTX	Alpha cobratoxin	2·8		5·6	22·5
2ALP	Alpha-lytic protease	1·7		7·1	52·5
2LDX	Apo-lactate dehydrogenase isoenzyme C4	2·96		38·4	16·9
8ATC	Aspartate carbamoyltransferase	2·5	AB	32·7	20·2
1PPT	Avian pancreatic polypeptide	1·37		50·0	0·0
1AZU	Azurin	2·7		11·3	25·8
2AZA	Azurin (oxidized)	1·8	A	16·3	33·3
1REI	Bence–Jones immunoglobulin rei var. portion	2·0	A	2·8	47·7
2RHE	Bence–Jones protein (lambda, var. domain)	1·6		2·6	43·0
4CPV	Calcium-binding parvalbumin	1·5		56·5	0·0
3ICB	Calcium-binding protein	2·3		57·3	0·0
2CAB	Carbonic anhydrase form B	2·0		15·6	30·9
2CA2	Carbonic anhydrase II (carbonate dehydratase)	1·9		16·4	28·9
5CPA	Carboxypeptidase alpha (Cox)	1·54		38·1	16·3
8CAT	Catalase	2·5	A	32·5	15·5
4CTS	Citrate synthase	2·9	A	52·4	4·1
3CNA	Concanavalin A	2·4		0·0	40·5
1CRN	Crambin	1·5		47·8	8·7
2SOD	Cu, Zn superoxide dismutase	2·0	O	2·0	38·4
2B5C	Cytochrome *b*5 (oxidized)	2·0		40·0	24·7
1CCR	Cytochrome *c* (rice)	1·5		42·3	0·0
2CYP	Cytochrome *c* peroxidase (baker's yeast)	1·7		50·2	5·5
2CCY	Cytochrome *c'* (*Rhodospirillum molischianum*)	1·67	A	74·8	0·0
3C2C	Cytochrome *c*2 (reduced)	1·68		42·9	0·0
1CY3	Cytochrome *c*3 (*Desulfovibrio desulfuricans*)	2·5		24·6	0·0
2CDV	Cytochrome *c*3 (*Desulfovibrio vulgaris miyazaki*)	1·8		28·0	9·3
1CC5	Cytochrome *c*5 (oxidized)	2·5		47·0	0·0
451C	Cytochrome c_{551} (reduced)	1·6		50·0	0·0
4MDH	Cytoplasmic malate dehydrogenase	2·5	A	42·6	18·9
3GPD	D-glyceraldehyde-3-phosphate dehydrogenase	3·5	R	27·2	21·0
4DFR	Dihydrofolate reductase	1·7	A	26·4	30·8
4FD1	Ferredoxin (*Azotobacter vinelandii*)	1·9		34·0	13·2
1FDX	Ferredoxin (*Peptococcus aerogenes*)	2·0		14·8	7·4
3FXC	Ferredoxin (*Spirulina platensis*)	2·5		13·3	15·3
1CYC	Ferrocytochrome *c*	2·3		34·0	0·0
1FX1	Flavodoxin	2·0		32·0	21·8
4FXN	Flavodoxin (semiquinone form)	1·8		36·2	21·0
2GCH	Gamma chymotrypsin A	1·9		9·7	33·1
1GCR	Gamma-II crystallin	1·6		7·5	44·3
2GN5	Gene *5* DNA-binding protein	2·3		0·0	4·6
1GCN	Glucagon (pH 6–pH 7 form)	3·0		48·3	0·0
1GP1	Glutathione peroxidase	2·0	AB	32·2	15·8
3GRS	Glutathione reductase	1·54		34·3	24·1
1HMQ	Hemerythrin (Met)	2·0	A	69·9	0·0
3HHB	Hemoglobin (deoxy, human)	1·74	AB	78·7	0·0
1FDH	Hemoglobin (deoxy, human fetal)	2·5	AG	72·1	0·0
1ECD	Hemoglobin (erythrocruorin, deoxy)	1·4		76·5	0·0
2DHB	Hemoglobin (horse, deoxy)	2·8	AB	64·5	0·0
1HDS	Hemoglobin (sickle cell)	1·98	AB	56·3	0·0
2LHB	Hemoglobin V (cyano, Met)	2·0		75·2	0·0
6ADH	Holo-liver alcohol dehydrogenase	2·9	A	17·9	19·3
1FC2	Immunoglobulin Fc	2·8	D	8·7	45·1
2IG2	Immunoglobulin G1	3·0	LH	6·6	41·0
2MCP	Immunoglobulin MC/PC603	3·1	LH	3·8	47·7
1PYP	Inorganic pyrophosphatase	3·0		15·0	10·0
2INS	Insulin	2·5	AB	56·0	4·0
1GF1	Insulin-like growth factor I	NA		37·1	0·0
1GF2	Insulin-like growth factor II	NA		38·8	6·0
2KAI	Kallikrein A	2·5	AB	9·1	32·8
1ABP	L-Arabinose-binding protein	2·4		36·6	5·9
5LDH	Lactate dehydrogenase H4 and S-LAC-/NAD^+	2·7		39·0	9·3
2LH1	Leghemoglobin (acetate, Met)	2·0		77·8	0·0
2LZM	Lysozyme (*E. coli* infected with bacteriophage T4)	1·7		66·5	8·5

Table 1 *(continued)*

Code	Protein name	Resolution	*Ni*	%*H*	%*E*
1LZ1	Lysozyme (human)	1·5		39·2	7·7
1LZT	Lysozyme triclinic crystal form (hen egg white)	1·97		42·6	6·2
2MLT	Melittin	2·0	A	92·3	0·0
7API	Modified alpha 1-antitrypsin	3·0	AB	27·5	35·5
1MBD	Myoglobin (deoxy, pH 8·4)	1·4		77·8	0·0
1MBS	Myoglobin (Met)	2·5		72·5	0·0
1NXB	Neurotoxin B	1·38		0·0	41·9
1HIP	Oxidized high potential iron protein	2·0		22·4	10·6
1PFC	PFC′ fragment of an IgGI	3·125		3·6	30·6
3PGK	Phosphoglycerate kinase	2·5		34·5	11·1
3PGM	Phosphoglycerate mutase	2·8		30·0	6·5
1BP2	Phospholipase A2 (bovine pancreas)	1·7		48·8	6·5
1P2P	Phospholipase A2 (porcine pancreas)	2·6		43·5	4·8
3PCY	Plastocyanin (HG2+substituted)	1·9		10·1	35·4
2PAB	Prealbumin (human plasma)	1·8	A	7·0	51·8
2SGA	Proteinase A	1·5		9·9	54·1
3SGB	Proteinase B (*Streptomyces griseus*)	1·8	E	6·5	51·9
3RP2	Rat mast cell protease II	1·9	A	8·0	37·1
1RHD	Rhodanese	2·5		29·7	10·9
1RN3	Ribonuclease A	1·45		21·0	38·7
5RXN	Rubredoxin (oxidized, Fe(III))	1·20		16·7	14·8
2STV	Satellite tobacco necrosis virus	2·50		11·4	44·6
1SN3	Scorpion neurotoxin (variant 3)	1·8		12·3	18·5
4SBV	Southern bean mosaic virus coat protein	2·8	A	15·1	35·2
2SNS	Staphylococcal nuclease	1·5		20·6	19·9
2SSI	*Streptomyces* subtilisin inhibitor	2·6		15·9	24·3
2SBT	Subtilisin novo	2·8		21·5	13·8
2TAA	Taka-amylase A	3·0		26·2	14·4
3TLN	Thermolysin	1·6		41·5	16·5
2TBV	Tomotao bushy stunt virus	2·90	A	4·1	30·4
1EST	Tosyl-elastase	2·5		10·4	34·2
1TIM	Triose phosphate isomerase	2·5	A	45·7	17·0
5PTI	Trypsin inhibitor (crystal form II)	1·0		20·7	24·1
1TGS	Trypsinogen	1·8	Z	10·2	37·8

Proteins were selected from the January, 1991 release of the Brookhaven Protein Databank (Bernstein *et al.*, 1977). Protein databank code (Code), name (Protein name), resolution, chains used (*Ni*), and helix (%*H*) and strand (%*E*) composition of chains used are tabulated. The program DSSP (Kabsch & Sander, 1983) was used to calculate helix ($\alpha+3_{10}$) and strand compositions. For these proteins, the average helix composition is 31·0% ($\sigma = 22·4$) and strand composition is 19·3% ($\sigma = 16·5$).

Table 2

EVALBASE: database of proteins used to evaluate secondary structure composition prediction methods

Code	Protein name	Resolution	*Ni*	%*H*	%*E*	Homology
1LDB	Apo-L-lactate dehydrogenase	2·8		41·8 (50·1)	19·1 (17·2)	5LDH (32%)
	Aspartate receptor (ligand domain)	2·5		78·2 (82·7)	0·0 (1·3)	2TAA (17%)
2P21	C-H-*ras* P21 protein catalytic domain	2·2		33·9 (31·9)	24·6 (15·2)	5LDH (15%)
3GAP	Catabolite gene activator protein	2·5	AB	32·7 (45·1)	13·8 (14·0)	4CTS (12%)
2GBP	D-Galactose/D-glucose-binding protein	1·9		43·0 (35·6)	18·5 (6·7)	1ABP (16%)
4XIA	D-Xylose isomerase	2·3		47·8 (44·7)	8·9 (14·3)	7API (13%)
4TS1	Tyrosyl-transfer RNA synthetase	2·5	AB	51·7 (57·7)	9·5 (16·6)	8CAT (13%)
1GOX	Glycolate oxidase	2·0		44·3 (51·5)	12·6 (10·2)	7API (13%)
1HNE	Human neutrophil elastase	1·8		8·3 (8·8)	33·9 (37·3)	1EST (33%)
2LTN	Pea lectin	1·7	AB	3·5 (8·2)	47·4 (61·5)	3CNA (19%)
3PFK	Phosphofructokinase	2·4		46·1 (48·1)	18·5 (17·7)	3GRS (15%)
2PRK	Proteinase K	1·5		26·9 (26·8)	21·5 (20·9)	2SBT (30%)
1R08	Rhinovirus 14	3·0	123	11·3 (7·3)	31·4 (35·8)	2IG2 (12%)
1THI	Thaumatin I	1·5		12·1 (7·2)	34·8 (41·4)	2ALP (14%)
1SGT	Trypsin	1·7		12·1 (20·7)	34·5 (20·3)	1EST (25%)

Protein databank code (Code), name (Protein name), resolution, chains used (*Ni*), helix (%*H*) and strand (%*E*) composition of chains used (NET1+NET2 predictions enclosed in parentheses), and the protein in OPTBASE (Table 1) with greatest sequence homology (percent homology is enclosed in parentheses) are tabulated. The program DSSP (Kabsch & Sander, 1983) was used to calculate helix ($\alpha+3_{10}$) and strand compositions. The program CLUSTAL (Higgins & Sharp, 1988) was used to calculate sequence homology. For these proteins, the average helix composition is 32·9% ($\sigma = 20·7$) and strand composition is 21·9% ($\sigma = 12·5$).

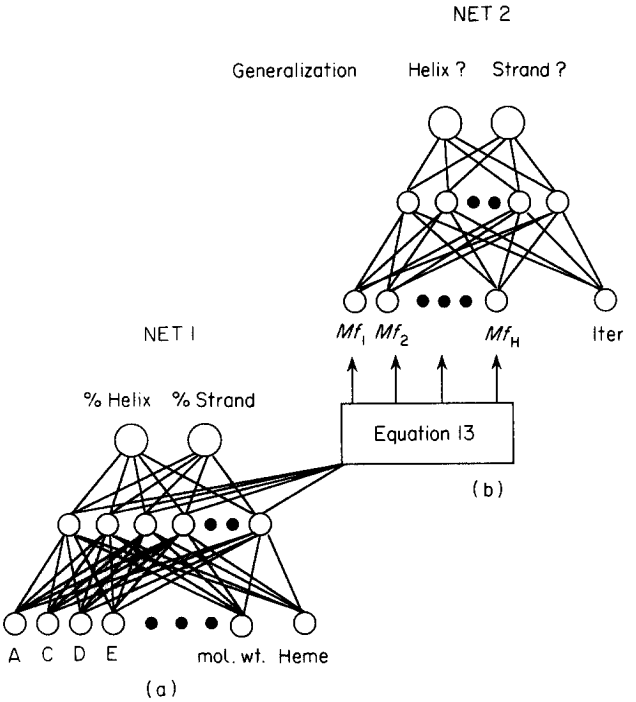


Figure 1. The "tandem" neural network approach to secondary structure composition prediction. To use this strategy, an example protein's amino acid composition, molecular weight and heme presence/absence are supplied as input to NET1 (part (a), lower left). NET1 makes a secondary structure composition prediction after every 30 iterations of training on the examples in OPTBASE (Table 1). Memory factors (*Mf*) are computed using eqn (13) and supplied to NET2 along with the number of training iterations (Iter). NET2 predicts a memory state for each secondary structure prediction at each stage of NET1 training (part (b), upper right). With this tandem network strategy, NET2 can be used to determine the best secondary structure composition prediction of NET1.

which were vectorized as follows:

$$w = \langle \{BiasHid_j, (WtI\_H_{ji}) i = 1 \ldots \text{ninputs}) j = 1 \ldots \text{nhidden}, \{BiasOut_k, (WtH\_O_{kj}) j = 1 \ldots \text{nhidden}\} k = 1 \ldots \text{noutputs}\rangle. \quad (2)$$

Given these weights, a fully connected hidden-node network will produce output node activities (secondary structure content predictions, generalization state, etc.) with the following equations:

$$O_{ke} = \text{SIG}\left(\sum_j (WtH\_O_{kj} * H_{je}) + BiasOut_k\right) \quad (3)$$

and

$$H_{je} = \text{SIG}\left(\sum_i (WtI\_H_{ji} * I_{ie}) + BiasHid_j\right), \quad (4)$$

where SIG is a non-linear activation function:

$$\text{SIG}(x) = 1/\{1 + \exp(-x)\}. \quad (5)$$

As stated, our goal was to train a network to predict secondary structure content, or to mathematically minimize eqn (1) over all the proteins in OPTBASE. While Rumelhart's adaptation of gradient descent (Rumelhart *et al.*, 1986*a*,*b*) could have been used for network training, its slow convergence rate would have made our experimentation impractical. We implemented a more powerful error-minimization procedure, the method of conjugate gradient back propagation. This method has been applied in a neural network context by Kramer & Sangiovanni-Vincentelli (1989) and Makram-Ebeid *et al.* (1989) and has been discussed theoretically by Hertz *et al.* (1991). We provide the method in an explicit form that is easy to program.

The conjugate gradient method of minimizing eqn (1) requires the use of a direction vector d, which parallels eqn (2):

$$d = \langle \Delta BiasHid_j, (\Delta WtI\_H_{ji}) i = 1 \ldots \text{ninputs}\} j = 1 \ldots \text{nhidden}, \{\Delta BiasOut_k, (\Delta WtH\_O_{kj}) j = 1 \ldots \text{nhidden}\} k = 1 \ldots \text{noutputs}\rangle. \quad (6)$$

The vector elements of d are defined by:

$$\Delta BiasHid_j = \varepsilon * \sum_e \delta hid_{je}, \quad (7)$$

$$\Delta WtI\_H_{ji} = \varepsilon * \sum_e \delta hid_{je} * I_{ie}, \quad (8)$$

$$\Delta BiasOut_k = \varepsilon * \sum_e \delta out_{ke}, \quad (9)$$

$$\Delta WtH\_O_{kj} = \varepsilon * \sum_e \delta out_{ke} * H_{je}, \quad (10)$$

with

$$\delta hid_{je} = H_{je} * (1 - H_{je}) * \sum_k (\delta out_{ke} * WtH\_O_{kj}), \quad (11)$$

$$\delta out_{ke} = O_{ke} * (1 - O_{ke}) * (T_{ke} - O_{ke}). \quad (12)$$

Table 3

Average differences ($\bar{X}$), standard deviation of differences (σ) and highest signed differences between "observed" secondary structure compositions

Secondary structure	$\bar{X} = \frac{\sum_{i=1}^{N} \lvert A_i - D_i \rvert}{N}$	$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (\bar{X} - \lvert A_i - D_i \rvert)^2}{N-1}}$	Highest difference $(A_i - D_i)$
Helix	5·2	5·4	1GCN (24·1)
Strand	4·4	5·5	3PCY (26·3)

Secondary structure elements provided by HELIX and SHEET records in the protein databank entries (Bernstein *et al.*, 1977) and those made by the program DSSP (Kabsch & Sander, 1983) were used to calculate A_i (author) and D_i (DSSP), respectively. Only 80 (N) of the 104 proteins in OPTBASE had clear, non-redundant secondary structure assignments in HELIX and SHEET records and did not claim to use DSSP. DSSP assigned more secondary structure in 32 of the 80 proteins (i.e. in 40% of the structures, $(A_i - D_i) < 0{\cdot}0$). The largest differences between helix and strand assignments were glucagon (1GCN) and plastocyanin (3PCY), respectively. All values listed are in percent.

The conjugate gradient method effects a series of line minimizations along a current direction vector, d_{old}. Mathematically, this involves finding λ such that $E(w+\lambda * d_{old})$ is minimized. After λ is determined, the network weights are changed by $\lambda * d_{old}$ and a new direction vector d is calculated using eqns (3) to (5) and (7) to (12). Because of the line minimization, d is necessarily perpendicular to d_{old} (Polak, 1971). If this process were repeated, a zigzag path to the minimum would result. Therefore, a better approach provides a compromising constant β that blends the gradient information contained in the old direction vector d_{old} and the newly calculated direction vector d. This compromise is the basis of conjugate gradient methods and attempts to prevent the new direction vector from spoiling the minimization achieved by the old one. Our training procedure instituted the Polak–Ribiere method (Press *et al.*, 1990) for computing β. In practice, successive line minimizations result in a deterioration in β; therefore after every "restart" iterations, the newly calculated direction vector d was used without the compromising constant β.

In summary, the conjugate gradient method for network training is as follows.

(1) Initialize and vectorize weights$\rightarrow w$.
(2) Calculate direction vector, $d \rightarrow d_{old} = d$.
(3) Line minimization: find λ that minimizes:

$$E(w+\lambda * d_{old}).$$

(4) Update weights:

$$w = w + \lambda * d_{old}.$$

(5) Calculate new direction vector, d and total error, $E(w)$.
(6) If $E(w)$ converged, stop.
(7) If reset gradient (restart), then let $d_{new} = d$, else

$$\beta = (d - d_{old}) \cdot d / (d_{old} \cdot d_{old}); \quad d_{new} = d + \beta d_{old}.$$

(8) $d_{old} = d_{new}$.
(9) Go to step (3).

We used the procedures LINMIN, MNBRAK and BRENT developed by Press *et al.* (1990) for the line minimization of eqn (1) in step (3).

We have developed an interactive, programmable neural network simulator (BIOPROP), which gives the user the ability to define variables, test and loop on these and other internal variables during a network simulation. Such an environment enables "jack-knife" training procedures (see below), random-sample training and testing, architecture optimization, etc., BIOPROP can use either gradient or conjugate-gradient descent procedures for network training.

(c) *Network optimization*

The 1st step in optimizing a feedforward network is to determine the best architecture for the task at hand. For 3-layer networks, this involves determining the optimal number of hidden nodes. In most applications, the procedure of choice is to vary systematically the number of hidden nodes, train the network to convergence, fix the weights and evaluate the network. The architecture with the best performance statistics is chosen for future experimentation. In our mapping, however, networks with greater than 4 hidden nodes contained more weights than examples in OPTBASE and the problem of memorization could not be avoided.

We used the small database of examples to our advantage and instituted jack-knife training protocols. A jack-knife procedure systematically extracts 1 example from a database, derives prediction parameters (network weights, regression coefficients, etc.) on the remaining set of examples, replaces the example, and repeats until every example has been extracted, much like pulling out and replacing every blade in a Swiss Army knife. A jack-knife procedure not only has the advantage of increasing the effective learning set size, but it also provides a wealth of statistical information after considering every example in a database as a true testing example. Even with the benefit of a jack-knife training protocol, however, our networks could still memorize the remaining proteins in OPTBASE.

To overcome this problem, we employed a "best-case" method for network architecture determination. For each extraction in a jack-knife training procedure, we found the network architecture that performed best on each extracted example. Here, every reasonable architecture was considered for every extracted example. The architecture that performed best on the extracted example was noted, the example was replaced, and a new example was selected. This process was repeated through the entire learning set, after which, the best performing architectures were counted and the most frequently occurring optimal architecture was selected. This method is referred to as best case because network training was stopped at the point of maximum generalization. While it is not statistically valid to develop a prediction method that requires knowledge of the correct answer, this best-case method was useful for determining a network architecture that had the greatest potential for maximizing prediction accuracy.

For situations in which memorization was not a problem, a simple sampling procedure was used for architecture determination. In this case, a random set of examples was extracted from the database of learning examples. A network architecture was trained on the remaining examples until eqn (1) was minimized, the weights were fixed and the network was evaluated on the extracted examples. This procedure was repeated numerous times for each architecture so that average and standard deviations of performances could be calculated for each architecture. The architecture with the best performance statistics was the architecture of choice.

Virtually every error minimization procedure is plagued with multiple minima. Neural network training is no exception. Multiple minima can be spotted after a network is repeatedly trained on a fixed database, each time reinitializing the starting network weights. If the final totalerrors (eqn (1)) for all trials differ significantly from one another, then multiple minima do, indeed, exist.

While one could continue to reinitialize the network weights, train until convergence, and repeat until the lowest possible totalerror is achieved, a better approach is to slightly perturb the converged set of weights in random directions (von Lehman *et al.*, 1988), train until totalerror convergence and repeat. The weights with the lowest totalerror will be the weights of choice. The big advantage of this approach is that after each weight perturbation, the number of training iterations until convergence is significantly smaller than the number required with a completely randomized set of weights. Even more sophisticated methods such as the Metropolis algorithm (Metropolis *et al.*, 1953) can be implemented to accept and reject weight sets with different converged totalerrors. The conjugate gradient method of error minimization, coupled with weight perturbation is rapid enough to invest a sizeable amount of time searching for the global minimum of eqn (1).

3. Results

(a) *NET1: predicting secondary structure content*

To predict secondary structure content, we designed a network (NET1) with a $22:H:2$ architecture (22 input, H hidden and 2 output nodes) depicted in Figure 1(a). The information of a protein's amino acid composition, molecular weight and heme group presence or absence was supplied to NET1 as a string of 22 real numbers, and the network was trained to produce two real numbers describing helix and strand content at its output nodes. Because it is difficult for the sigmoid function (eqn (5)) to produce values near 0·0 and 1·0, we did not expect very accurate predictions on proteins with extremes in secondary structure content. The number of weights or adjustable parameters for this architecture with H hidden notes (eqn (2)) was $H+22H+2+2H$ $(=25*H+2)$ weights.

For a reasonable number of hidden nodes (such as $H>4$), NET1 contained more weights than examples in OPTBASE. Therefore, we determined optimal network architecture of NET1 by employing a best-case method. For each extraction in a jack-knife training procedure, we varied the NET1 architecture between five and ten hidden nodes. Table 4 shows the results of this procedure. Because we utilized the knowledge of the correct answer for a stopping criterion in training NET1, the data in Table 4 only suggest an architecture with the highest prediction potential. Before NET1 can be used for secondary structure content predictions, a statistically acceptable criterion for halting NET1 training process was necessary.

Table 4

NET1 architectures obtained after training on OPTBASE proteins

Hidden nodes, H	N	H, $\bar{X}$ (σ)	E, $\bar{X}$ (σ)
5	17	5·3 (5·4)	2·6 (2·6)
6	18	9·0 (10·7)	4·7 (5·3)
7	14	4·1 (5·2)	3·5 (5·7)
8	**25**	**2·4 (3·3)**	**3·6 (4·2)**
9	16	4·4 (7·0)	6·8 (7·4)
10	14	7·8 (13·8)	5·3 (4·7)
Totals	104	5·3 (8·1)	4·4 (5·2)

NET1 architectures, obtained after performing a best-case jack-knife training procedure on the proteins in OPTBASE, are listed. In this procedure, an example was extracted from OPTBASE, and each of the 6 network architectures ($22:5:2 \rightarrow 22:10:2$) was trained on the remaining set of 103 examples. The architecture that performed best on the extracted example was chosen. The frequency that a particular architecture was chosen (N), average errors ($\bar{X}$), and standard deviations of errors (σ) in helix (H) and strand (E) predictions with that architecture are listed. The network architecture that had the highest value of N and the lowest prediction errors for H and E was chosen as the network architecture with the greatest prediction potential. This network architecture is highlighted in boldface.

(b) *NET2: determining NET1's state of memorization*

NET2 was designed to determine when NET1 training should be halted. Unfortunately, hidden-node NET1 networks provided the best predictions at different points in training depending on the testing example and the number of training iterations (see Fig. 2(a) to (c)). We noted, however, that poor predictive performance usually resulted when the hidden node activities differed significantly from the distribution of hidden node activities utilized on examples in the learning set. We therefore defined a memory factor (Mf_j) for each hidden node j in NET1:

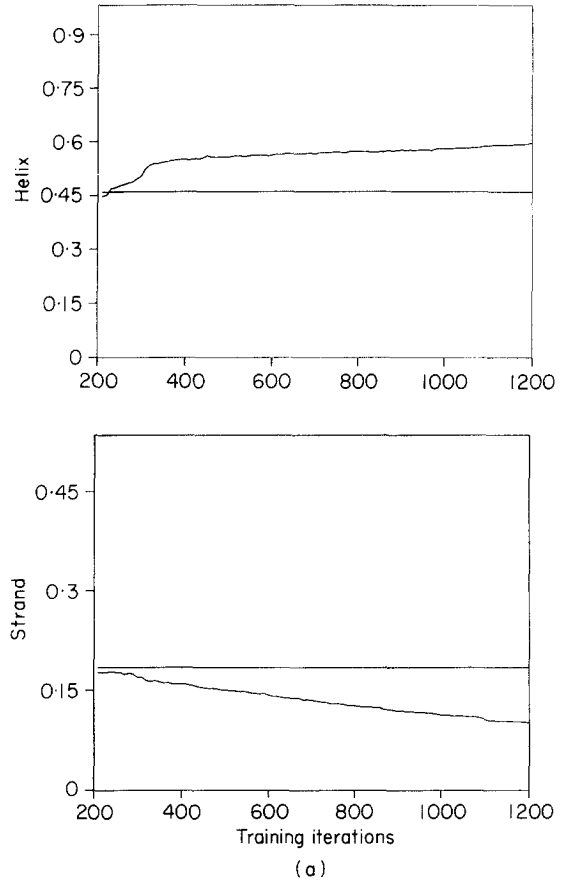
$$Mf_j = \left[\frac{(\overline{H_j} - H_j)}{\sigma_j}\right]^2, \qquad (13)$$

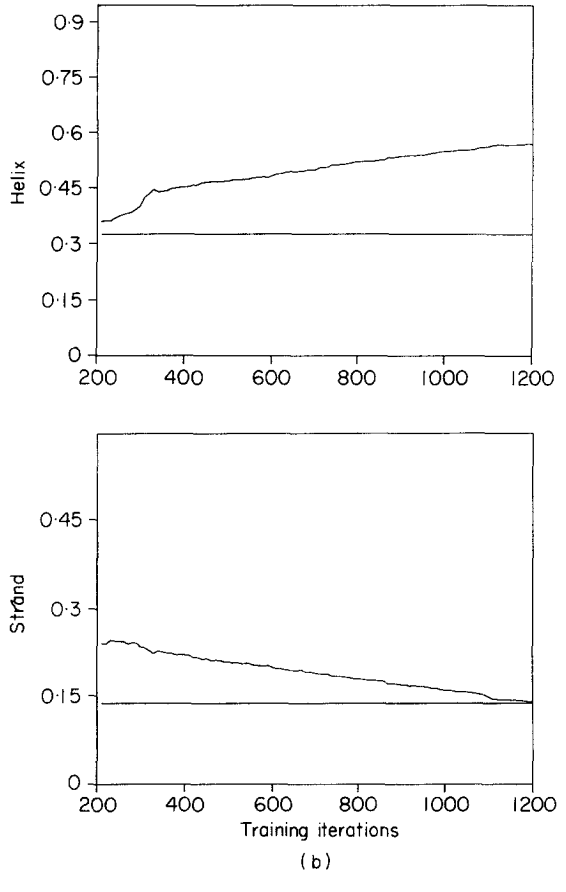
where $\overline{H_j}$ represents the average activity of hidden node j (learning set), H_j represents the activity of hidden node j (testing example) and σ_j is the standard deviation of activities of hidden node (learning set).

Defined as such, a hidden node's memory factor will be large when its activity on a testing example significantly differs from its distribution of activities used for learning set examples. Similarly, a hidden node's memory factor will be small when its activity falls within the distribution of training set activities.

The relationship between memory factors of NET1 and its state of generalization was not simply that small memory factors imply good generalization. So, to learn the relationship between NET1 memory factors and its generalization ability, we designed a second network, NET2 (see Fig. 1(b)). NET2 had a $(H+1):H':2$ architecture (H hidden nodes from NET1 + number of training iterations, H' of its own hidden nodes and 2 output nodes). NET2 was a simple classification network designed to accept a set of memory factors at a stage of NET1 training as input, and produce the answers to the following two yes-or-no questions as output: Is NET1 in a state of generalization for its helix prediction? Is NET1 in a state of generalization for its strand prediction? Designed as such, NET2 could account for situations in which NET1 was in a state of generalization for one structural element while still learning the other (see Fig. 2(a) to (c)).

NET1 was considered to be in a state of generalization if, after 200 training iterations, its structural content predictions were in error by no more than 5%. After this, NET1 was considered to be on the course to memorization; or alternatively, NET1 was not in a state of generalization. We generated training information for NET2 by monitoring the training of NET1: (1) an example was extracted from our OPTBASE as in a jack-knife procedure; (2) NET1 was trained for 30 iterations with conjugate–gradient descent, and then tested both on the extracted example and on the remaining set of 103 examples; (3) NET1 memory factors (eqn (13)) were calculated and stored along with the





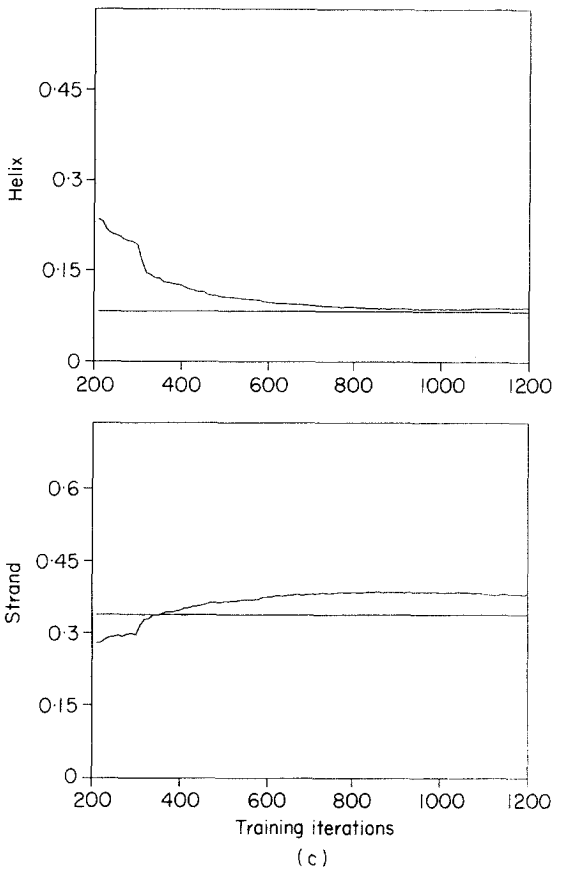


Figure 2. NET1 helix and strand composition predictions as a function of its OPTBASE training for 3 EVALBASE examples: (a) phosphofructokinase (3PFK); (b) catabolite gene activator protein (3GAP); and (c) human neutrophil elastase (1HNE). The horizontal lines indicate the observed secondary structure content as assigned by the program DSSP (Kabsch & Sander, 1983). The curved lines indicate the NET1 predictions. The points of greatest generalization occur at different points for each secondary structure composition of each protein.

number of training iterations; (4) steps (2) to (3) were repeated until NET1 training converged (approx. 2000 iterations), after which the extracted testing example was replaced and another selected. The procedure (steps (1) to (4)) was repeated until every example in OPTBASE had been extracted. This procedure for generating input/output information for NET2 resulted in approximately 90 examples per protein. With 104 proteins in OPTBASE, over 9000 examples were generated for NET2 training. Given that H' varied from five to eleven, we did not expect NET2 to possess the capacity for memorization, even for its largest possible architecture (a $9:11:2$ architecture will contain 134 weights).

Optimal architecture of NET2 ($9:H':2$) was determined by a simple sampling procedure. One hundred examples were randomly selected from the learning set generated for NET2, NET2 was trained until convergence on the remaining set of examples, and then tested on the 100 extracted examples. This

Table 5
Performance of NET2

Hidden nodes, H	H, $\bar{X}$ (σ)	E, $\bar{X}$ (σ)
5	0·35 (0·13)	0·44 (0·06)
6	0·36 (0·05)	0·49 (0·06)
7	0·39 (0·12)	0·56 (0·05)
8	0·42 (0·08)	0·57 (0·08)
9	0·41 (0·09)	0·57 (0·08)
10	**0·66 (0·11)**	**0·68 (0·07)**
11	0·51 (0·09)	0·59 (0·06)

For each architecture, 100 examples were selected at random from the generated NET2 database (see text), NET2 was trained until convergence on the remaining examples and tested on the 100 extracted examples. This process was repeated 10 times for each architecture. The values reported are correlations between NET2 output node activities and the NET1 state of generalization for helix (H) and strand (E) predictions. Values enclosed in parentheses are standard deviations of correlations over the 10 trials. The architecture in boldface was chosen for NET2.

procedure was repeated ten times, each time with a new set of 100 examples (see Table 5).

Because of the sizeable spreads in performance correlations in Table 5, we suspected a multiple minimum problem with NET2 optimization. As a control, we repeated the training of the optimal NET2 architecture (9 : 10 : 2, $H' = 10$ hidden nodes), each time with a new set of initial weights. As seen in Figure 3, NET2 converged to a variety of different totalerrors. Note how small changes in totalerrors result in rather large changes in prediction performances. Clearly, the lowest totalerror is desired. To fully optimize NET2, we implemented the method of weight perturbation to escape from any possible local minima in the NET2 training procedure. After numerous training trials, we obtained what appeared to be a minimum totalerror for NET2. At this point, we considered NET2 optimized.

Once optimal architectures were determined for NET1 and NET2, NET1 was trained on the complete set of proteins in OPTBASE and its weight sets were saved every 30 iterations. A true evaluation of our "tandem-network" method was made by performing predictions on each of the 15 examples in EVALBASE in the following manner: (1) amino acid composition, molecular weight and heme content of each protein were provided to NET1 as input; (2) NET1 performed helix- and strand-content predictions on this information using each of the weight sets previously saved; (3) for each prediction (representing a stage of training), memory factors were computed and given to NET2 as input information; (4) NET2 outputs (each a numerical value between zero and unity) describing the NET1 state of generalization for helix and strand predictions were stored. After this procedure was repeated for each of the NET1 weight sets, the predictions made by NET1 with the strongest NET2 outputs were selected. This seemingly complex procedure accounted for the different points of memorization that NET1 experienced for helix and strand predictions, respectively (see Fig. 2(a) to (c)).

Despite what appear to be moderate correlations in the optimization of NET2 in Table 5, after escaping multiple minima, NET2 did, indeed, learn when NET1 was in a state of generalization for both helix and strand predictions. Figure 4 displays the performance of NET1 and NET2 in tandem. Using this tandem-network approach, the average prediction errors for proteins within OPTBASE were 4·1% ($\sigma = 4·5$) and 4·1% ($\sigma = 3·4$) for helix and strand content, respectively, and the average prediction errors for proteins within EVALBASE were 5·0% ($\sigma = 3·4$) and 5·6% ($\sigma = 4·9$) for helix and strand content, respectively.

(c) *Non-hidden node network*

We designed a network similar to NET1 but without a layer of hidden nodes. This fully connected non-hidden-node network only contained

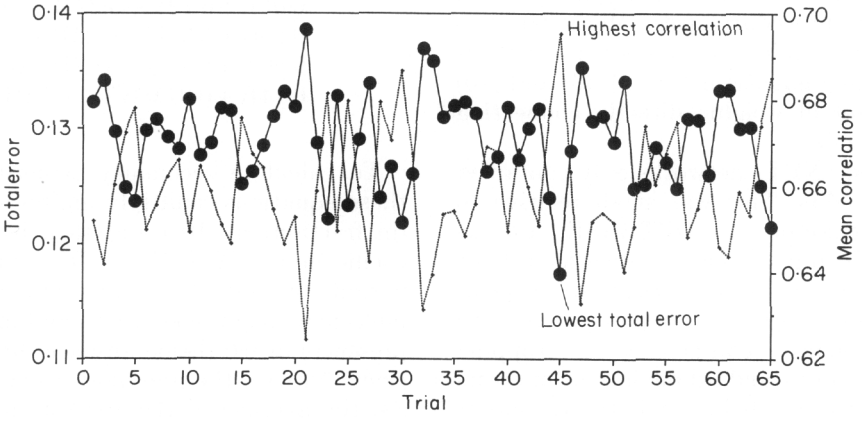


Figure 3. Multiple minima in training of NET2. The complete set of examples generated for NET2 training was used in this experiment. For each trial, NET2 weights were initialized to random values. NET2 was trained with conjugate gradient descent until the totalerror (eqn (1)) converged to a minimum. (—●—), the various totalerrors at convergence. (· · · + · · ·), the mean correlation ($\sqrt[n]{\prod_k \text{Corr}_k}$) between NET2 outputs and its target outputs at convergence.

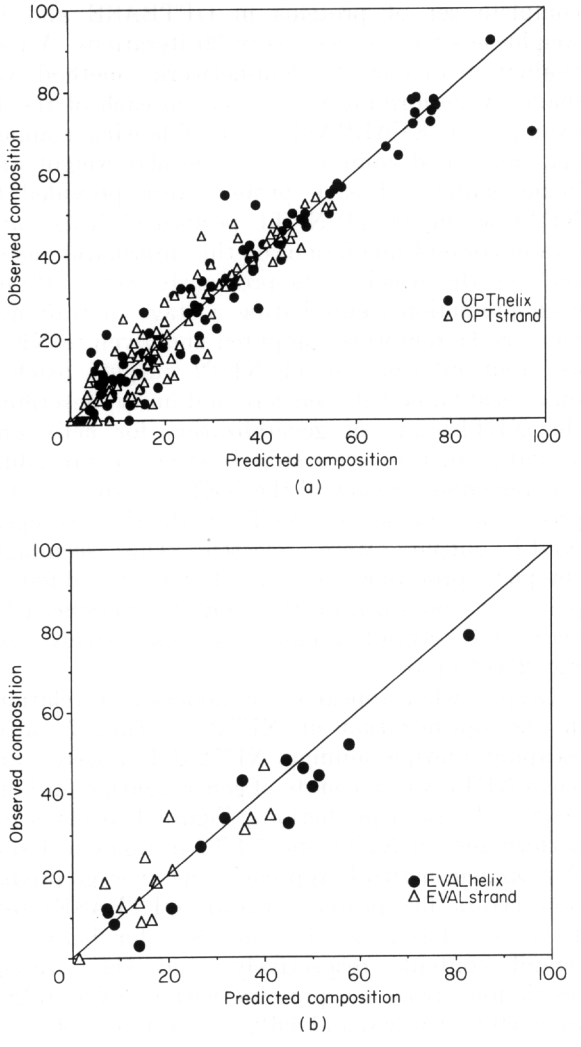


Figure 4. Observed and predicted secondary structure compositions using the tandem network method (a) on examples used in network optimization (OPTBASE), and (b) on true testing examples (EVALBASE). Circles represent helix and triangles represent strand compositions.

weights connecting input-to-output nodes and a bias for each output node. The conjugate gradient descent training procedure (see Methods) was adpated for this non-hidden node network analysis by simply redefining equations (2), (3), (6) and (10) to consider the set of weights directly connecting input-to-output nodes.

Non-hidden node NET1 networks only contained $22*2+2=46$ weights. This was small relative to the number of examples in OPTBASE; and as expected, non-hidden node NET1 networks did not experience problems with memorization. A jack-knife training procedure was implemented with totalerror convergence (eqn (1)) as a stopping criterion. Average and standard deviations of errors were calculated from the helix- and strand-content predictions after treating each extracted example as a true testing example. This procedure resulted in a set of 46 network weights for each of the 104 proteins in OPTBASE. The average of these weight sets (Fig. 5) was used for the evaluation on EVALBASE proteins. As expected, this average weight set was almost identical with the weight set obtained after training on all the examples in OPTBASE.

The non-hidden node network prediction errors, after considering every example in OPTBASE as a true testing example (i.e. the jack-knife training of OPTBASE), were 12·5% ($\sigma=10{\cdot}9$) and 9·6% ($\sigma=8{\cdot}7$) for helix- and strand-content predictions, respectively. On EVALBASE, the average non-hidden node network weights (Fig. 5) achieved prediction errors of 11·8% ($\sigma=9{\cdot}8$) and 9·5% ($\sigma=4{\cdot}1$) for helix- and strand-content predictions, respectively.

(d) *Multiple linear regression*

Davies (1964), as well as Krigbaum & Knutton (1973), noted the correlation between amino acid composition and secondary structure content using the technique of multiple linear regression (m.l.r.). We computed m.l.r. coefficients on our considerably larger OPTBASE using the capabilities of the program LOTUS 1-2-3 release 3 (1989). We performed a jack-knife procedure within the confines of a LOTUS 1-2-3 spreadsheet. As in the non-hidden-node network analysis, the average and standard deviations of errors were calculated from the helix- and strand-content predictions after treating each extracted example as a true testing example. This procedure resulted in a set of 23 coefficients (1 intercept + 20 amino acid coefficients + 1 molecular weight coefficient + 1 heme coefficient) for each of the 104 proteins in OPTBASE. The average of the 104 sets of coefficients derived in this jack-knife m.l.r. analysis are as follows:

$$\%H = -8{\cdot}67 + 1{\cdot}04\mathrm{W} + 1{\cdot}08\mathrm{I} + 1{\cdot}04\mathrm{Y} + 1{\cdot}09\mathrm{F} + 1{\cdot}18\mathrm{L} + 0{\cdot}94\mathrm{V} + 0{\cdot}86\mathrm{M} + 0{\cdot}99\mathrm{C} + 0{\cdot}97\mathrm{A} + 0{\cdot}75\mathrm{G} + 0{\cdot}77\mathrm{H} + 0{\cdot}62\mathrm{P} + 0{\cdot}74\mathrm{S} + 0{\cdot}78\mathrm{T} + 0{\cdot}80\mathrm{N} + 0{\cdot}86\mathrm{Q} + 0{\cdot}87\mathrm{D} + 0{\cdot}77\mathrm{E} + 0{\cdot}93\mathrm{K} + 0{\cdot}94\mathrm{R} + 0{\cdot}01\mathrm{MWT} + 0{\cdot}26\mathrm{HEME} \quad (14)$$

and

$$\%E = -1{\cdot}36 + 0{\cdot}15\mathrm{W} + 0{\cdot}10\mathrm{I} + 0{\cdot}01\mathrm{Y} + 0{\cdot}12\mathrm{F} - 0{\cdot}03\mathrm{L} + 0{\cdot}12\mathrm{V} + 0{\cdot}22\mathrm{M} + 0{\cdot}06\mathrm{C} + 0{\cdot}10\mathrm{A} + 0{\cdot}26\mathrm{G} + 0{\cdot}32\mathrm{H} + 0{\cdot}23\mathrm{P} + 0{\cdot}30\mathrm{S} + 0{\cdot}31\mathrm{T} + 0{\cdot}20\mathrm{N} + 0{\cdot}19\mathrm{Q} + 0{\cdot}07\mathrm{D} + 0{\cdot}27\mathrm{E} + 0{\cdot}06\mathrm{K} + 0{\cdot}14\mathrm{R} + 0{\cdot}01\mathrm{MWT} - 0{\cdot}15\mathrm{HEME} \quad (15)$$

To calculate secondary structure content from these equations, multiply amino acid compositions (in fraction) by 10·0 and then by their respective coefficients (W, I, Y, . . ., R), divide molecular weight by $1{\cdot}0\times10^4$ and then multiply by the molecular weight coefficient, and multiply 1·0 (heme presence) or 0·0 (heme absence) by the heme coefficient. As with the non-hidden node network analysis, the average of the 104 sets of coefficients was almost identical with the set of coefficients obtained after performing m.l.r. on all the samples in OPTBASE. These average coefficients achieve prediction errors of 9·8% ($\sigma=7{\cdot}8$) and 6·9%

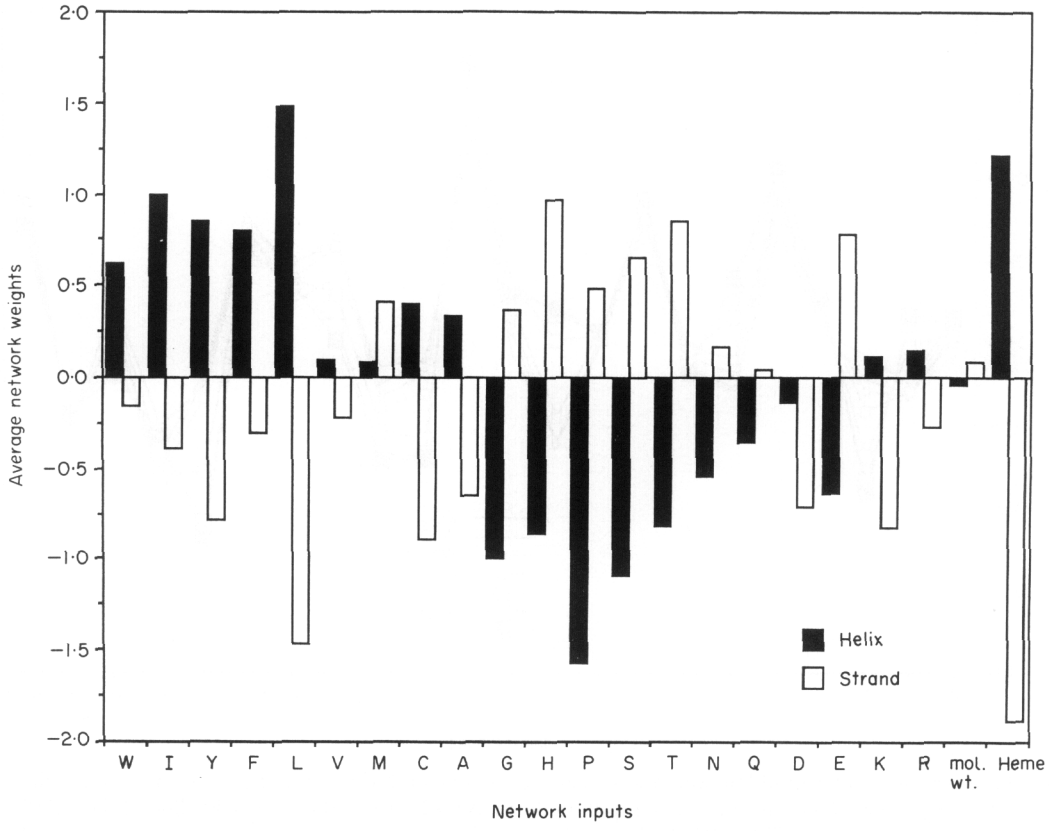


Figure 5. The average of the non-hidden-node network weight sets obtained by performing a jack-knife procedure on the examples in OPTBASE. Filled bars indicate network weights connecting input elements to the helix output node, and open bars indicate weights to the strand output node. Ordered horizontally are the one-letter amino acid codes in decreasing hydrophobicity, molecular weight (mol. wt.), and heme presence (Heme). These weights will achieve errors of 9·5% ($\sigma = 7{\cdot}5$) and 7·4% ($\sigma = 5{\cdot}8$) for helix and strand composition predictions, respectively, on the examples in OPTBASE, and 11·8% ($\sigma = 9{\cdot}8$) and 9·5% ($\sigma = 4{\cdot}1$) for helix and strand composition predictions, respectively, on the examples in EVALBASE.

($\sigma = 6{\cdot}1$) for helix- and strand-content predictions, respectively, on the examples in OPTBASE.

The prediction errors of the m.l.r. analysis treating every example in OPTBASE as a true testing example (i.e. the jack-knife m.l.r. analysis) were 12·9% ($\sigma = 11{\cdot}1$) for helix- and 9·0% ($\sigma = 8{\cdot}5$) for strand-content predictions. Using the average m.l.r. coefficients (eqns (14) and (15)) on EVALBASE, the average prediction errors were 12·8% ($\sigma = 11{\cdot}1$) and 10·6% ($\sigma = 4{\cdot}6$) for helix- and strand-content predictions, respectively.

(e) *Secondary structure predictions*

The method of Qian & Sejnowski (1988) was used to make secondary structure predictions on the examples in EVALBASE. This method assumes that conformation of a central amino acid depends both on its identity and on its flanking amino acid sequence (±6 residues). Qian & Sejnowski (1988) used a database very similar to our OPTBASE to derive their network weights without including any examples similar to those in EVALBASE. Their method appears to be one of the better secondary structure prediction methods (Garnier, 1990). Using this method, predicted secondary structure elements were counted and normalized to make an indirect prediction of secondary structure content. It should be noted that the indirect helix-content predictions utilizing the neural network of Qian & Sejnowski (1988) included only α-helix elements, whereas the previously described methods included both α- and 3_{10}-helix elements in the prediction of helix content.

The average prediction errors for this counting method on the proteins in OPTBASE were 12·5% ($\sigma = 12{\cdot}0$) and 10·2% ($\sigma = 9{\cdot}2$) for helix- and strand-content predictions, respectively, whereas the average prediction errors on proteins in EVALBASE were 11·9% ($\sigma = 7{\cdot}0$) and 8·6% ($\sigma = 7{\cdot}7$) for helix- and strand-content predictions, respectively.

4. Discussion

As seen in Table 3, the average secondary structure content differences after counting the assignments by the authors and by DSSP for 80 of the 104 proteins in OPTBASE are 5·2% and 4·4% for helix- and strand-content, respectively. Clearly, there is a degree of uncertainty in secondary structure assignment, even with known protein structures. Indeed,

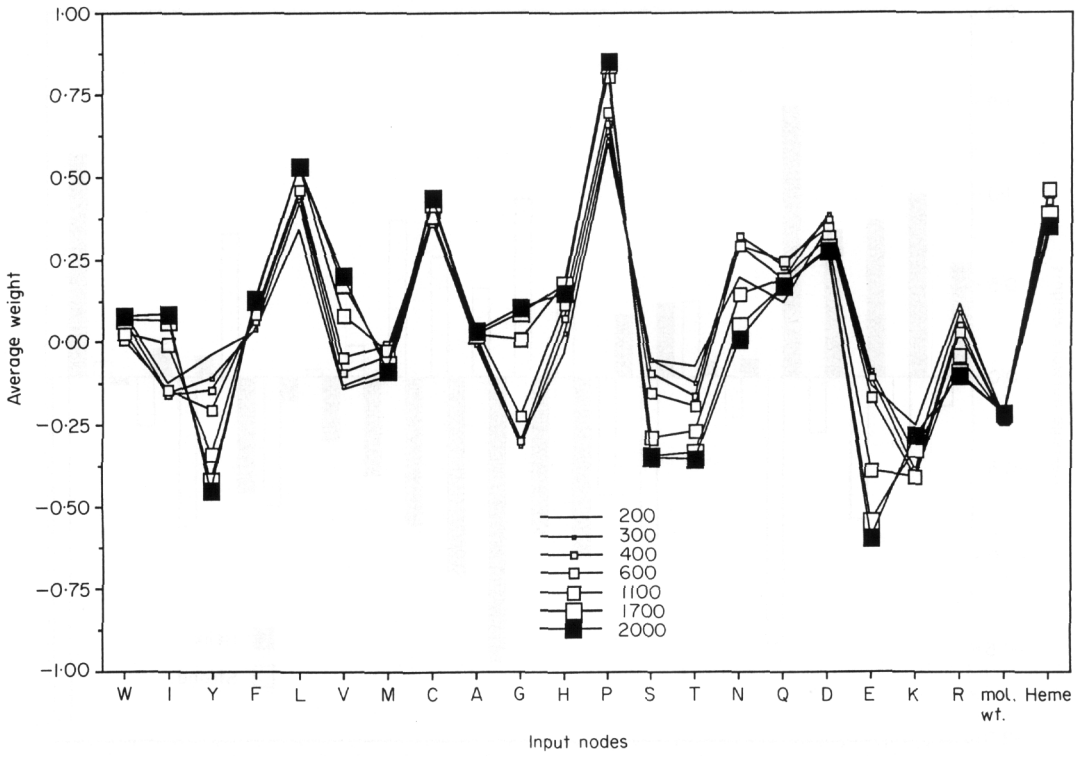


Figure 6. The average input-to-hidden weights at various points of NET1 training. Weights connecting input-to-hidden nodes (WtI_H_{ji}) were averaged for each input node in NET1 after 200, 300, 400, 600, 1100, 1700 and 2000 training iterations on examples in OPTBASE. Amino acid composition is represented with the respective one-letter codes W, I, Y, etc., the molecular weight with mol. wt., and Heme presence or absence with Heme.

if a method of predicting secondary structure content performs with average errors in the range of 0 to 10% (1σ) for both helix and strand content, then the method will approach the uncertainty associated with secondary structure assignment.

The performances of all four methods are summarized in Table 6. While the non-hidden-node network performs slightly better than the m.l.r. coefficients, both performances still border outside the acceptable (0 to 10%) error range. The COUNT method, which considers local sequence information, performs better than both the m.l.r. and non-hidden-node network analysis; but because of the inaccuracies associated with secondary structure prediction, the COUNT method also borders outside the acceptable error range. The tandem network method clearly outperforms the three other methods with prediction errors as low as those errors associated with secondary structure assignment.

Table 6

Comparison of secondary structure predictions for the 15 proteins in EVALBASE

Structure	COUNT	m.l.r.	NHN	TN
Helix, $\bar{X}$ (σ)	11·9 (7·0)	12·8 (11·1)	11·8 (9·8)	5·0 (3·4)
Strand	8·6 (7·7)	10·6 (4·6)	9·5 (4·1)	5·6 (4·9)

Under COUNT, the secondary structure composition was computed by counting the secondary structure elements determined by the method of Qian & Sejnowski (1988). Under m.l.r. (multiple linear regression), NHN (non-hidden-node network, 22:0:2), and TN (tandem networks 22:8:2|9:10:2, i.e. NET1+NET2), secondary structure composition was calculated directly from amino acid composition, molecular weight and heme presence. Average and standard deviation (enclosed in parentheses) of errors between prediction and assignment by DSSP (Kabsch & Sander, 1983) of secondary structure composition are tabulated in percent.

A physical basis for the success of secondary structure content predictions can be derived from an analysis of the prediction parameters. While we can directly interpret the non-hidden-node network weights and the m.l.r. coefficients, hidden-node networks are less conducive to such analyses. We can, however, compare input-to-hidden-node weights in hidden-node networks to determine the relative contribution of each input element.

The non-hidden-node network weights in Figure 5 reveal an ambivalence to molecular weight information, though strand content seems a bit more influenced by molecular weight information than helix content. Equations (14) and (15) are similar in this sense. Though molecular weight information appears to have little effect on the non-hidden-node network weights and the m.l.r. coefficients, the weights that connect the molecular weight input-to-hidden-nodes in NET1 during its training appear to contribute as much information as the other input-to-hidden weights (see Fig. 6). In fact, the average weights assigned to molecular weight information are actually greater than those considering Trp, Ile, Phe, Met, Ala, His, Gln and Arg compositions. That is, molecular weight information has just as much

influence on protein secondary structure content as many of the amino acid compositions. Also note that, as shown in Figure 6, the average weights to molecular weight information change very little throughout the process of training NET1. This suggests that the influence of molecular weight was learned early in NET1 network training without changing as more specific rules were encompassed.

For globular proteins, molecular weight is correlated directly to size and accessible surface area (Chothia, 1975; Janin, 1976). Therefore, given the amino acid distribution and molecular weight of a protein, much can be inferred about its size and shape; but for a method to consider the effects of protein size along with the quantities of each amino acid, it would have to possess the ability to learn from "higher-order information", or combinations of informational elements. The techniques of m.l.r. and non-hidden-node networks do not possess this ability. Hidden-node networks, on the other hand, do possess this ability. Indeed, it seems likely that the differences between the tandem network prediction errors and those of the three other methods in Table 6 are a result of the ability to capture higher-order information.

In both the non-hidden-node network and the m.l.r. analyses, the coefficients on presence or absence of a heme group are quite large relative to the other coefficients. As with molecular weight information, the heme input-to-hidden weights in NET1 appear to contribute a great deal of information (see Fig. 6). In fact, the average input-to-hidden weights to heme have greater influence than most of the amino acid compositions (exceptions are Leu and Pro). The m.l.r. coefficients directly suggest that the presence of a heme group increases helix content by 26%. This is consistent with optical rotatory dispersion experiments, which have indicated that apomyoglobin has approximately 20% less helix content than myoglobin (Harrison & Blout, 1965; Expand & Scheraga, 1968; Hermans & Puett, 1971). Likewise, the loss of a heme group from cytochrome *c* has been shown to disrupt the protein structural integrity altogether (Dickerson & Timkovich, 1975).

Heme groups occur in hemoglobins, *b*-type and *c*-type cytochromes, and catalases and peroxidases (Schulz & Schirmer, 1988). In hemoglobins, the function of the heme group and surrounding peptide chain is to protect the liganded ferrous ion from oxidation and to give iron its O_2-binding properties (Wang, 1970), whereas cytochromes use the Fe^{3+}/Fe^{2+} redox system within the heme group for electron transport (Dickerson & Timkovich, 1975). For both of these proteins, the heme group sits in a crevice lined with apolar amino acids. In cytochrome C3 molecules (1CY3 and 2CDV), over 20 amino acids interact with a single heme group (Dickerson & Timkovich, 1975), and in one subunit of hemoglobin there are some 60 atoms making van der Waals' contacts with the porphyrin ring of the heme group (Perutz, 1970).

Clearly, the presence or absence of a heme group has a significant effect on structure and function of a protein. The non-hidden-node network weights, m.l.r. coefficients and the hidden-node network weights all seemed to capture the relationship between presence of a heme group and secondary structure content of the protein. Perhaps further functional information combined with amino acid composition and molecular weight will improve prediction performances.

In general, the weights in Figure 5 and equations (14) and (15) suggest that hydrophobic residues appear more influential on helix content and hydrophilic residues appear more influential on strand content. Similar to their helix propensities (Chou & Fasman, 1974), Leu contributes strongly towards helix content and Pro strongly against (Fig. 5). Also note that, as shown in Figure 6, Leu and Pro have the largest input-to-hidden weights throughout the NET1 training. Indeed, the properties of Leu and Pro are quite significant in the formation/disruption of protein secondary structure. It should be emphasized that the average input-to-hidden weights in Figure 6 only suggest which informational elements contribute more than others. Because the hidden-to-output weights further complicate the picture, at present no hard and fast rules relating input-to-output information can be extracted from hidden-node networks.

From Figure 5 and equations (14) and (15), the residues contributing most strongly to strand formation are His, Thr, Glu, Ser and Pro. With the exception of Thr, these residues do not have large strand propensities (Chou & Fasman, 1974). In general, the non-hidden-node network weights in Figure 5 and the coefficients in equations (14) and (15) have little correlation with the helix and strand propensities (Chou & Fasman, 1974) of the respective amino acid residues (the correlation between Chou–Fasman propensities and our non-hidden-node network weights are 0·42 and −0·19 for helix and strand, respectively). One explanation for these low correlation values is that, unlike the statistics describing helix and strand propensities, these methods relate *global* amino acid composition to *global* secondary structure content. Whereas most probabilistic methods utilize secondary structure propensities of amino acids in their local context, our m.l.r. coefficients and non-hidden-node network weights will consider the effects of all the amino acid residues working together, both locally and globally, to build a particular amount of each secondary structure.

Because of the complexity of hidden-node networks (eqns (3) to (5)), any further weight interpretation is beyond the scope of our present capabilities. Coupled with the effects of memorization, it seems very unlikely that a complete physical interpretation can be made from the weights contained within NET1 + NET2. Indeed, this is perhaps the most disappointing feature of neural network application, a highly successful prediction without the luxury of a comprehensive explanation. Despite our inability to explain completely the

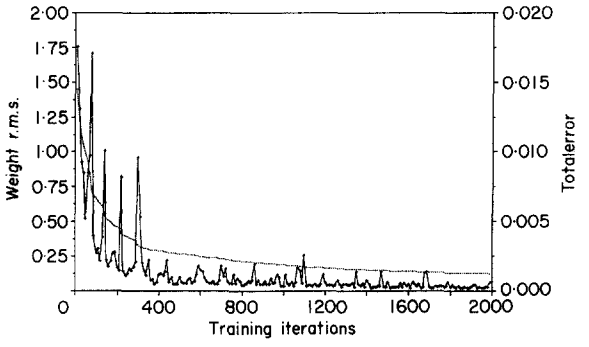


Figure 7. The root-mean-square (r.m.s.) difference between successive weight sets in NET1 training procedure (solid line) and the totalerror over examples in OPTBASE (broken line). Weights were compared after every 10 training iterations. The complete set of proteins in OPTBASE was used in network training.

success of our prediction method, we can report with confidence that amino acid composition, molecular weight and presence or absence of a heme group are correlated strongly with protein secondary structure content.

While our tandem network approach is highly accurate, it may appear that the total number of parameters is large. Given a NET2 architecture of 9:10:2 (122 weights) plus multiple sets of NET1 architectures of 22:8:2 (202 weights) representing 60 stages of training (between 200 and 2000 iterations with steps of 30→60 stages) will result in over 12,200 parameters! With only 104 proteins, this may appear to be an exorbitant number of parameters.

Recall, however, that we generated over 9000 examples for the NET2 training. This decreases the ratio of observations to prediction parameters. Also, the weights sets representing a stage of NET1 training are not independent. As seen in Figure 7, the r.m.s. deviation of weight sets between training epochs (10 iterations per epoch) approaches 0·0 quite smoothly with some spiked feature. The largest output activities of NET2 (i.e. NET1 predicted to be in its greatest state of generalization) often occurred on one of the peaked epochs in Figure 7. Because choices made by NET2 often fell on one of these spikes, the overall parameter set can be drastically reduced. Regardless of the ratio between observations and parameters, however, our tandem network method demonstrated highly accurate predictions on examples quite different from those used in network optimizations. Indeed, our tandem network approach truly demonstrated inductive learning.

5. Conclusion

The results of our experiments have demonstrated good accuracies in predicting secondary structure content. We have compared four methods, including a tandem neural network approach, a non-hidden-node network analysis, a multiple linear regression analysis and structural element count after a local secondary structure prediction. Our tandem neural network approach outperformed the three other methods with prediction errors as low as 5·0% and 5·6% for helix and strand content, respectively. These average prediction errors are as low as those associated with secondary structure assignment.

An interesting implication of this mapping is that, although amino acid sequence is important for local secondary structure predictions, amino acid composition is a more important determinant of secondary structure content. For a protein of N residues and D_i copies of amino acid i, a single composition of amino acids will have $N!/\prod_i D_i!$ possible sequences. In this vast sequence space, only a small number of sequences will likely fold into compact three-dimensional structures. Our mapping suggests that all those sequences that fold will contain the same quantity of secondary structure, the implication being that the amino acid sequence determines whether or not a protein will fold and the amino acid composition determines the final quantity of secondary structure.

One practical application of this mapping is for improving secondary structure prediction. Garnier (1978) demonstrated an increase in secondary structure prediction given knowledge of secondary structure content. Recently, Kneller *et al.* (1990) demonstrated a marked increase in secondary structure prediction when they considered tertiary structural class. Their best prediction increases were for all α- and all β-proteins and less so for α/β proteins. All α- and all β-poteins can be classified as such if our method suggests high helix and high strand respectively. Therefore, after combining our method with other existing methods of secondary structure prediction, it is likely that secondary structure prediction accuracies will increase.

We acknowledge the support of the University of California Regents, the Health Effects Research Division, Health and Environmental Research, Office of Energy Research of the U.S. Department of Energy and National Science Foundation.

References

Anfinsen, C. G. (1973). Principles that govern folding of protein chains. *Science*, **181**, 223.

Blundell, T. L., Sibanda, B. L., Sternberg, M. J. & Thornton, J. M. (1987). Knowledge-based prediction of protein structures and the design of novel molecules. *Nature (London)*, **326**, 347–352.

Blundell, T., Carnery, D., Gardner, S., Hayes, F., Howlin, B., Hubbard, T., Overington, J., Singh, D. A., Sibanda, B. L. & Sutcliffe, M. (1988). 18th Sir Hans Krebs lecture. Knowledge-based protein modelling and design. *Eur. J. Biochem.* **172**, 513–520.

Bohr, H., Bohr, J., Brunak, S., Cotterill, R. M., Lautrup, B., Norskov, L., Olsen, O. H. & Petersen, S. B. (1988). Protein secondary structure and homology by neural networks: the alpha-helices in rhodopsin. *FEBS Letters*, **241**, 223–228.

Bohr, H., Bohr, J., Brunak, S., Cotterill, R. M. J., Fredholm, H., Lautrup, B. & Petersen, S. B. (1990).

A novel approach to prediction of the 3-dimensional structures of protein backbones by neural networks. *FEBS Letters,* **261**, 43–46.

Chothia, C. (1975). Structural invariants in protein folding. *Nature (London),* **254**, 304–308.

Chou, P. Y. & Fasman, G. D. (1974). Conformational parameters for amino acids in helical, β-sheet and random coil regions from proteins. *Biochemistry,* **13**, 211.

Crick, F. (1989). The recent excitement about neural networks. *Nature (London),* **337**, 129–132.

Davies, D. (1964). A correlation between amino acid composition and protein structure. *J. Mol. Biol.* **9**, 605–609.

Dickerson, R. E. & Timkovich, R. (1975). *Enzymes,* **11**, 397–547.

Expand, R. M. & Scheraga, H. A. (1968). The influence of long-range interactions on the structure of myoglobin. *Biochemistry,* **7**, 2864–2872.

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

Garnier, J., Osguthorpe, D. J. & Robson, B. (1978). Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. *J. Mol. Biol.* **120**, 97.

Greer, J. (1981). Comparative model-building of the mammalian serine proteases. *J. Mol. Biol.* **153**, 1027–1042.

Greer, J. (1990). Comparative modeling methods: application to the family of the mammalian serine proteases. *Proteins,* **7**, 317–334.

Harrison, S. C. & Blout, E. R. (1965). Reversible conformational changes of myoglobin and apomyoglobin. *J. Biol. Chem.* **240**, 299–303.

Hermans, J. J. & Puett, D. (1971). Relative effects of primary and tertiary structure on helix formation in myoglobin and α-lactalbumin. *Biopolymers,* **10**, 895–914.

Hertz, J., Krogh, A. & Palmer, R. G. (1991). In *Introduction to the Theory of Neural Computation* (Vol. 1), pp. 145–156, 124–129, Addison-Wesley Publishing Company, Santa Fe, CA.

Higgins, D. G. & Sharp, P. M. (1988). CLUSTAL: a package for performing multiple sequence alignment on a microcomputer. *Gene,* **73**, 237–244.

Holbrook, S. R., Muskal, S. M. & Kim, S. H. (1990). Predicting surface exposure of amino acids from protein sequence. *Protein Eng.* **3**, 659–665.

Holley, L. H. & Karplus, M. (1989). Protein secondary structure prediction with a neural network. *Proc. Nat. Acad. Sci., U.S.A.* **86**, 152–156.

Janin, J. (1976). Surface area of globular proteins. *J. Mol. Biol.* **105**, 13–14.

Kabsch, W. & Sander, C. (1983). Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers,* **22**, 2577–2637.

Kneller, D. G., Cohen, F. E. & Langridge, R. (1990). Improvements in protein secondary structure prediction by an enhanced neural network. *J. Mol. Biol.* **214**, 171–182.

Kramer, A. H. & Sangiovanni-Vincentelli, A. (1989). Efficient parallel learning algorithms for neural networks. In *Advances in Neural Information Processing Systems I* (Denver 1988, Touretzky, D. S., ed.), pp. 40–48, San Mateo, Morgan Kaufmann, Denver, CO.

Krigbaum, W. R. & Knutton, S. P. (1973). Prediction of the amount of secondary structure in a globular protein from its amino acid composition. *Proc. Nat. Acad. Sci., U.S.A.* **70**, 2809–2813.

Levin, J. M. & Garnier, J. (1988). Improvements in a secondary structure prediction method based on a search for local sequence homologies and its use as a model building tool. *Biochim. Biophys. Acta,* **955**, 283–295.

Lim, V. I. (1974). Algorithms for predictions of α-helical and β-structural regions in globular proteins. *J. Mol. Biol.* **88**, 873.

Lotus Development Corporation. (1989). Lotus 1-2-3 release 3 Reference Manual, Lotus Development Corporation, Cambridge, MA.

Makram-Ebeid, S., Sirat, J.-A. & Viala, J.-R. (1989). A rationalized back-propagation learning algorithm. In *International Joint Conference on Neural Networks* (Washington 1989), pp. 373–380, IEEE, New York.

McGregor, M. J., Flores, T. P. & Sternberg, M. J. (1989). Prediction of β-turns in proteins using neural networks. *Protein Eng.* **2**, 521–526.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A. & Teller, E. (1953). Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**, 1087.

Muskal, S. M., Holbrook, S. R. & Kim, S. H. (1990). Prediction of the disulfide-bonding state of cysteine in proteins. *Protein Eng.* **3**, 667–672.

Nishikawa, K. & Ooi, T. (1982). Correlation of the amino acid composition of a protein to its structural and biological characteristics. *J. Biochem.* **91**, 1821–1824.

Nishikawa, K., Kubota, Y. & Ooi, T. (1983). Classification of proteins into groups based on amino acid composition and other characters: I. Angular distribution. *J. Biochem.* **94**, 981–995.

Pascarella, S. & Bossa, F. (1989). PRONET: a microcomputer program for predicting the secondary structure of proteins with a neural network. *CABIOS,* **5**, 319–320.

Perutz, M. F. (1970). Stereochemistry of cooperative effects in haemoglobin. *Nature (London),* **228**, 726–739.

Polak, E. (1971). In *Computational Methods in Optimization* (92.3), Academic Press, New York.

Press, W. H., Flannery, B. P., Teukolsky, S. A. & Vetterling, W. T. (1990). In *Numerical Recipes: The Art of Scientific Computing* (Chap. 10). Cambridge University Press, Cambridge.

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

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986*a*). Learning representations by back-propagating errors. *Nature (London),* **323**, 533–536.

Rumelhart, D. E., McClelland, J. L. & Group, T. P. R. (1986*b*). In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* (Vol. 1, Chap. 8), MIT Press, Cambridge, MA.

Schulz, G. E. & Schirmer, R. H. (1988). In *Principles of Protein Structure,* New York, 1988, pp. 211–220, Springer-Verlag, New York.

von Lehman, A., Paek, E. G., Liao, P. F., Marrakchi, A. & Patel. J. S. (1988). In *Factors Influencing Learning by Back-propagation,* New York, IEEE, San Diego and New York.

Wang, J. H. (1970). Synthetic biochemical models. *Accts Chem. Res.* **3**, 90–97.

Edited by F. E. Cohen