CHAPTER III
MATERIALS AND METHODS

3.1 Materials

3.1.1 Protein Database

Information on protein sequence and secondary structure was obtained from Brookhaven's Protein Data Bank (PDB).

URL = http://pdb.pdb.bnl.gov/

URL = Gopher://pdb.pdb.bnl.gov : 70/00/PDB/Entries

URL = Gopher://pdb.pdb.bnl.gov : 77/xfullindex/full

3.1.2 Computer Hardware

All computations were performed on the COMPAQ PRESARIO 4712 Series Personal Computer, with the following specification :

System Processor : Intel is Pentium®

System Clock Speed 166 MH_z

Hard Disk : 2.5 GB

Cache Memory 32 MB

Main memory : 32 MB of RAM

3.1.3 Computer Software

<u>Linux</u>

Linux is an operating system which shares many features of UNIX system V, but with many enhancements. It has become a widely popular version of UNIX for use in personal computers. Linux can be obtained from a variety of sources. Two of the most popular locations to find Linux on the internet are :

sunsite.unc.edu (152.2.22) in the */pub/Linux* directory

tsx-11.mit.edu (18.86.0.44) in the */pub/linux* directory

In this study, the neural network, SNNS, program was run on Linux version 2.0.0. Some tools and applications of Linux such as X-window, C++ compiler, vi - editor were also used.

SNNS (Stuttgart Neural Network Simulator)

SNNS is a simulator for neural networks that consists of two main components, the simulator kernel and the graphical user interface (Figure3.1). The SNNS was developed at the Institute for Parallel and Distributed High Performance System at Stuttgart University (SNNS, 1989). The SNNS simulator can be obtained as a free software via anonymous ftp from host

ftp.informatik.uni-stuttgart.de (129.69.211.2)

in the subdirectory

/pup/SNNS

as file

SNNSv3.3.tar.Z

After successful transmission the file was moved into the target directory and uncompressed with the Unix command

uncompress SNNSv3.3.tar.z

followed by

tar -xvf SNNSv3.3.tar

A simple network (Figure.3.2) that can be generated by the SNNS simulator consists of *units* and directed, weighted *links* (connection) between them. Depending on their function in the net, one can distinguish three types of units :

- *input units*, the units whose activation is the problem input for the net.
- *output units*, the units whose outputs represent the output of the net.
- *hidden units*, the remaining units that are not visible from the outside

In most neural network models the type correlates with the topological position of the unit in the net : If a unit does not have input connections but only output connections then it is an input unit. If it lacks output connections but has input links, it is an output unit, if it has both types of connections it is a hidden unit. The actual information processing within the units is modeled in the SNNS simulator with the *activation function* and the *output function*. The activation function first computes the net input of the unit from the weighted output values of prior units. It then computes the new activation from this net input. The output function takes this result to generate the output of the unit.

In contrast to other network simulators where the bias (threshold) of a unit is simulated by a link weight from a special 'on '-unit, SNNS represents it as a unit parameter. In the standard version of SNNS the bias determines where the activation function has its steepest ascent. Learning procedure like back-propagation change the bias of a unit like a weight during training.

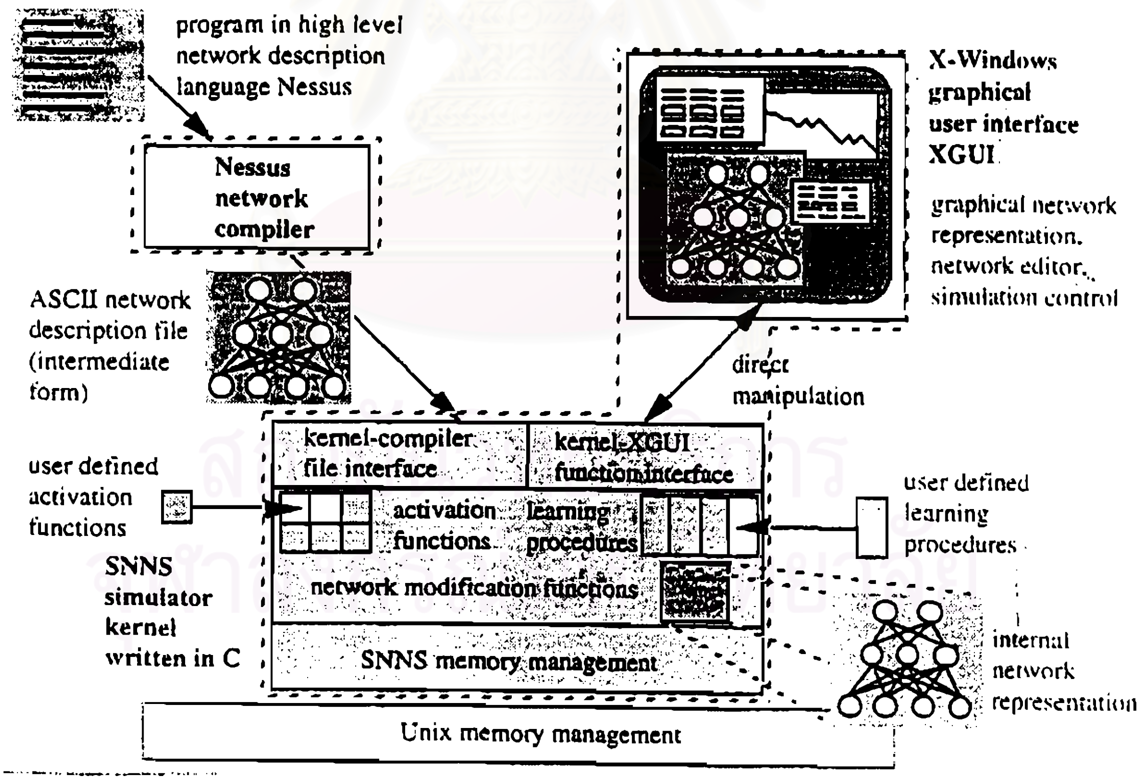

program in high level network description language Nessus
Nessus network compiler
ASCII network description file (intermediate form)
X-Windows graphical user interface XGUI
graphical network representation, network editor, simulation control
direct manipulation
kernel-compiler file interface
kernel-XGUI function interface
user defined activation functions
activation functions
learning procedures
user defined learning procedures
SNNS simulator kernel written in C
network modification functions
SNNS memory management
internal network representation
Unix memory management

Figure 3.1 SNNS components.

Source: Andrease et al., 1989. *SNNS User Manual, Version 3.3.* p. 2.

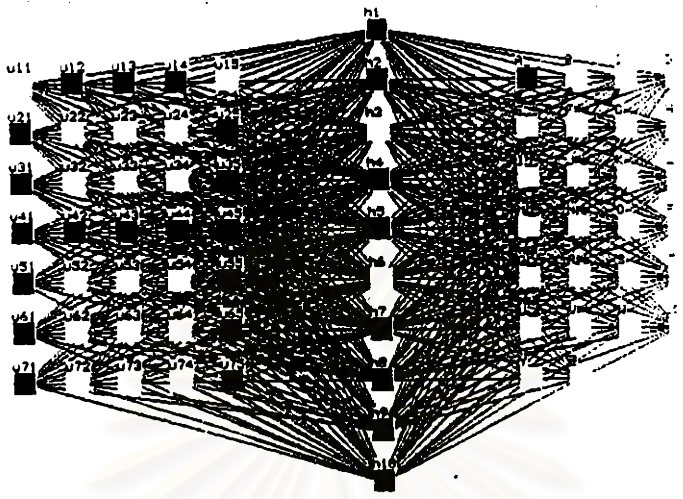


Figure 3.2 A simple network, it is a feed-forward net with three layer of units.
Source: Andrease et al., 1989. *SNNS User Manual, Version 3.3.* p. 22.

SNNS Algorithm.

Activation function : A new activation is computed from the output of preceeding units, usually multiplied by weights connecting these predecessor units with the current unit, the old activation and its bias. The SNNS default activation function *Act_logistic*

$$a_j(t+1) = \frac{1}{1 + e^{-(\sum w_{ij} o_i(t) - \theta_j)}}$$

where $a_j(t)$ is activation of unit j in step t

$O_i(t)$	is output of unit i in step t
W_{ij}	is weight of the link from unit i to unit j
θ_j	is threshold or bias of unit j
j	is index for some unit in the net
i	is index of a predecessor of the unit j

Output function : The output function computes the output of every unit from the current activation of this unit. The output function is in most cases identity function

(SNNS: Out_identity). This is the default in SNNS. The output creates the possibility to process the activation before an output occurs.

$$O_j(t) = f_{out}(a_j(t))$$

where :

$a_j(t)$	is activation of unit j in step t
$O_j(t)$	is output of unit j in step t
J	is index for all units of the net

Learning in Neural networks. An important focus of neural network research is the question of how to adjust the weights of link to get the desired system behavior. This modification is very often based on the Hebb-rule, which states that between two units is strengthened, if both units are active at the same time. In its general form is :

$$\Delta w_{ij} = g(a_j(t), t_i)\, h(O_i(t), w_{ij})$$

where:

Δw_{ij}	is weight of the link from unit *i* to unit *j*
$a_j(t)$	is activation of unit *j* in step *t*
t_i	is teaching input, in general the desired output of unit *j*
$O_i(t)$	is output of unit *i* at time *t*
$g(\ldots)$	is function, depending on the activation of the unit and the teaching input
$h(\ldots)$	is function, depending on the output of the preceding element and the current weight of the link

Training a feedforward neural network with supervised learning consists of following procedure :

1. An input pattern is presented to the network. The input is then propagated forward in the net until activation reaches the output layer. This constitutes the so called *forward propagation phase*.

2. The output of the output layer is then compared with the teaching input. The error, δ_j between the output O_j and the teaching input t_j of a target output unit j is then used

together with the output O_i of the source unit *i* to compute the necessary changes of the link w_{ij}. To compute the deltas of the following layer, which are already computed, are used in a formula given below . In this way the error are propagated backward, so this phase is called ***backward propagation.***

3. The weight change Δw_{ij} are cumulated for all patterns in the training file and the sum of all changes is applied after one full cycle (epoch) through the training pattern file.

The most famous learning algorithm which works in the manner described is backpropagation. The backpropagation weight updated rule, also called ***generalization delta rule*** reads as follows :

$$\Delta w_{ij} = \eta \, \delta_j \, O_i$$

$$\delta_j = \begin{cases} f'_j(net_j)(t_j - O_j) & \text{if unit j is a output - unit} \\ f'_j(net_j) \sum_k \delta_k w_{jk} & \text{if unit j is a hidden - unit} \end{cases}$$

Where :

η learning factor eta (a constant)

δ_j error (difference between the real output and the teaching input) of unit *j*

t_j teaching input of unit *j*

O_i output of the preceding unit *i*

i index of a predecessor to the current unit *j* with link w_{ij} from *i* to *j*

j index of the current unit

k index of a successor to the current unit *j* with link w_{jk} from *j* to *k*

There are several backpropagation algorithms supplied with SNNS. A simple version called Std_Backpropagation was used in this study.

Molecular visualization

Rasmol (the UC Regent/Modular CHEM Consortium version 2.6 ucb.) was used to visualize molecular information obtained primarily from PDB (Protein Data Bank, Brookhaven National Laboratory).

3.2 Methods

3.2.1 Protein database

The database of protein sequences and associated 2' structure used in this study was complied from information PDB (*gopher:// www.pdb.bnl.gov*). These collected protein was based on the classification scheme of Pascarella and Argos (1992) that categorizes the majority of known proteins (245 proteins) into 83 classes : 38 classes have two or more proteins members whereas the other 45 classes have only a single protein example. The data are based on superpositions among protein structures with similar main-chain folds and contain a large number of protein families with low sequence homology. The average sequence identity over all possible aligned PDB sequence pairs is 15%. Only 98 proteins (table 3.1) which have one chain of amino acid sequence were used in this study. These proteins were randomly grouped into two groups of training set and testing set. The number of proteins in the training and testing sets are 70 and 28 respectively.

3.2.2 Properties of amino acid used

A sequence of amino acids as input were replaced by a sequence of symbols representing properties. In the first set of trials, eight symbols were used to represent the following properties :

1. Aliphatic side chain	=	Gly, Ala, Val, Ile, Leu	symbol = 0.1
2. Aromatics side chain	=	Phe, Tyr, Trp	symbol = 0.2
3. Imino side chain	=	Pro	symbol = 0.3
4. Sulfur	=	Cys, Met	symbol = 0.4
5. Hydroxy	=	Ser, Thr	symbol = 0.5
6. Basics	=	Lys, Arg, His	symbol = 0.6
7. Acidics	=	Asp, Glu	symbol = 0.7
8. Amides	=	Asn, Gln	symbol = 0.8

In the second set of trials, the hydropathy of each amino acid residue was used as the sole attribute. The amino acid residues were divided into 2 groups (Table 3.2) and 5 groups (Table 3.3) by these hydropathies scale.

In the third set of trials, the relative hydrophobicity of amino acid residues was chosen as the sole attribute. The amino acid residues were classified into three groups (Chothia and Finkelstein, 1990).

Polar group	: Arg, Lys, Glu, Asp, Gln and Asn	symbol = 0.1
Neutral group	: Gly, Ala, Ser, Thr, Pro, His and Tyr	symbol = 0.2
Hydrophobic group:	Cys, Val, Leu, Ile, Met, Phe and Trp	symbol = 0.3

In the final set of the trials, the attribute chosen was the relative helical tendencies of amino acid measurement in one peptide. From this property, the amino acid was divided into 5 groups as shown in Table 3.4.

In summary, amino acid residues were divided into eight groups based on (1) property of amino acid side chain (Aliphatic, Aromatics, Imino, Sulfur, Hydroxy, Basics, Acidis, Amides), (2) hydropathy scale of each amino acid (2 groups) (3) hydrophobicity (3 groups) and (4) five groups based on relative helical tendencie. These properties were substituted in place amino acid residues symbols in the primary structures of proteins which were used for training and testing by Neural Networks in this study.

Table 3.1 Database of protein(s) used in protein structures prediction by NNs.

Input No.	Code	Protein name	Resolution	%H	%S	%T
1	1ALC	Calcium binding protein	1.7	21	6.5	0
2	1BP2	Hydrolase	1.7	55	48	33
3	1CA2	Lyase (oxo-acid)	2	16	23	9
4	1CDH	T-cell surface glycoprotein	2.3	0	65	0
5	1CMS	Hydrolase (acid proteinase)	2.3	12	46	0
6	1CTX	Toxin	2.8	0	23	23
7	1ECA	Oxygen transport	1.4	78	0	0
8	1GOX	Oxidoreductase (oxygen(A))	2	39	9	0
9	1HIP	Electron transfer (Iron -sulfur protein)	2	11	16	0
10	1HOE	Glycosidase inhibitor	2	0	77	16
11	1I1B	Cytokine	2	0	69	31
12	1LDM	Oxidoreductase (CHOH(D)-NAD(A))	5	33	21	31
13	1MBA	Oxygen storage	1.6	73	0	0
14	1P2P	Carboxylic ester hydrolase	2.6	52	8	33
15	1PHH	Oxidoreductase	2.3	26	26	0
16	1PLC	Electron transport	1.33	4	48	45
17	1PYP	Acid anhydride hydrolase	3	19	47	31
18	1R69	Gene regulating protein	2	59	0	0
19	1RHD	Transferase (thiosulfate, cyanide sulfur)	2.5	37	15	26
20	1SGT	Hydrolase (serine proteinase)	1.7	13	36	29
21	1TON	Hydrolase (serine proteinase)	1.8	12	30	24
22	1UBQ	Chromosomal protein	1.8	21	43	47
23	1UTG	Steroid binding	1.34	76	0	6
24	2LDX	Oxidoreductase (CHOH(D)-NAD(A))	2.96	45	17	0
25	2LH1	Oxygen transport	2	69	0	0
26	2LIV	Periplasmic binding protein	2.4	41	25	0
27	2LZ2	Hydrolase (o-glycosyl)	2.2	28	16	31
28	2MHR	Oxygen binding	1.7/1.3	64	0	32
29	2OVO	Proteinase inhibitor (KAZAL)	1.5	21	18	29

Table 3.1 (continued)

Input No.	Code	Protein name	Resolution	%H	%S	%T
30	2PAZ	Electron transfer (cuproprotein)	2	17	40	16
31	1ALC	Calcium binding protein	1.7	21	6.5	0
32	1BP2	Hydrolase	1.7	55	48	33
33	1CA2	Lyase (oxo-acid)	2	16	23	9
34	1CDH	T-cell surface glycoprotein	2.3	0	65	0
35	1CMS	Hydrolase (acid proteinase)	2.3	12	46	0
36	1CTX	Toxin	2.8	0	23	23
37	2TS1	ligase (synthetase)	2.3	35	7	0
38	3APP	Hydrolase (acid proteinase)	1.8	15	53	0
39	3BLM	Hydrolase	2	42	18	0
40	3C2C	Electron transport protein (cytochrome)	1.68	54	0	18
41	3CLA	Transferase (acyltransferase)	1.75	31	29	0
42	3CLN	Calcium binding protein	2.2	64	29	11
43	3CNA	Lectin (agglutinin)	2.4	0	50	0
44	3DFR	Oxido-reductase	1.7	25	37	37
45	3EST	Hydrolase (serine proteinase)	1.65	8	56	0
46	3GRS	Oxydoreductase (flavoenzyme)	1.54	38	32	0
47	3HVP	Hydrolase (acid proteinase)	2.8	9	57	10
48	3ICB	Calcium binding protein	2.3	77	0	0
49	3LZM	Hydrolase (o-glycosyl)	1.7	62	11	9
50	3PFK	Transferase (phosphotransferase)	2.4	50	20	0
51	3PGM	Transferase (phosphoryl)	2.8	32	10	12
52	3SSI	Serine protease inhibitor	2.3	16	32	0
53	451C	Electron transport	1.6	51	0	24
54	4APE	Hydrolase (acid proteinase)	2.1	11	62	7
55	4FXC	Electron transport	2.5	12	24	0
56	4FXN	Electron transport	1.8	38	27	14
57	4GCR	Eye lens protein	1.47	7	16	0
58	4PEP	Hydrolase (acid proteinase)	1.8	13	48	28

Table 3.1 (continued)

Input No.	Code	Protein Name	Resolution	%H	%S	%T
59	4TNC	Contractile system protein	2	64	0	0
60	5CPV	Calcium binding protein	1.6	55	6	15
61	1CCR	Electron transport (cytochrome)	1.5	57	0	28
62	2-Apr	Hydrolase (aspartic proteinase)	1.8	17	82	53
63	3B5C	Electron transport	1.5	38	27	26
64	1NXB	Neurotoxin (post-synaptic)	1.38	0	47	32
65	5CPA	Hydrolase (c-terminal peptidase)	1.54	34	15	42
66	8TLN	Hydrolase (metalloproteinase)	1.6	42	23	26
67	5PTI	Proteinase inhibitor (trypsin)	1	28	26	0
68	6RXN	Electron transfer (Iron -sulfur protein)	1.5	0	26	76
69	7RSA	Hydrolase (phosphoric diester)	1.26	27	77	0
70	8ADH	Oxidoreductase (NAD(A)-CHOH(D))	2.4	36	0	21
71	2CAB	Hydro-lyase	2	20	28	17
72	2CDV	Heme protein of electron transport	1.8	18	6	37
73	2CRO	Gene regulating protein	2.35	59	0	0
74	2FXB	Electron transport	2.3	22	9	15
75	2LBP	Periplasmic binding protein	2.4	40	33	0
76	2LDB	Oxidoreductase (CHOH(D)-NAD(A))	3	40	24	0
77	1MBS	Oxygen transport	2.5	73	0	0
78	1MBD	Oxygen storage	1.4	79	0	0
79	1CY3	Electron transport (heme protein)	1.7	22	3	44
80	2RNT	Hydrolase (endoribonuclease)	1.8	16	35	38
81	2STV	Virus	2.5	14	47	3
82	2TMV	Virus	2.9	12	22	18
83	5TNC	Contractile system proteins	2	65	0	0
84	2CYP	Oxidoreductase (H2o2(A))	1.7	49	0	10
85	2GBP	Periplasmic binding protein	1.9	45	25	48
86	2LHB	Oxygen transport	2	76	0	2
87	2PTN	Hydrolase (serine proteinase)	1.55	13	0	0

Table 3.1 (continued)

Input No.	Code	Protein name	Resolution	%H	%S	%T
88	1CYC	Electron transport	2.3	45	0	12
89	1LLC	Oxidoreductase (CHOH(D)-NAD(A))	3	41	23	0
90	1LZ1	Hydrolase (o-glycosyl)	1.5	26	15	56
91	1SBT	Hydrolase (serine proteinase)	2.5	31	10	0
92	2CI2	Proteinase inhibitor (chymotrypsin)	2	16	27	24
93	2TAA	Hydrolase (o-glycosyl)	3	21	25	0
94	5CYT	Electron transport (heme protein)	1.5	51	0	19
95	3PGK	Phosphotransferase (carboxyl as acceptor)	2.5	36	13	12
96	7PCY	Electron transport protein	1.8	7	62	29
97	8DFR	Oxidoreductase (CHOH(D)-NAD(A))	1.7	27	60	17
98	9PAP	Hydrolase (sulfhydryl proteinase)	1.65	27	17	28

Table 3.2 Two groups of amino acids which were divided by hydropathy property.

AMINO ACIDS	HYDROPATHY	SYMBOL
Ile	4.5	
Val	4.2	
Leu	3.8	
Phe	2.8	0.1
Cys	2.5	
Met	1.9	
Ala	1.8	
Gly	-0.4	
Thr	-0.7	
Ser	-0.8	
Trp	-0.9	
Tyr	-1.3	
Pro	-1.6	
His	-3.2	0.2
Glu	-3.5	
Gln	-3.5	
Asp	-3.5	
Asn	-3.5	
Lys	-3.9	
Arg	-4.5	

Table 3.3 Seven groups of amino acids which were divided by the hydropathy property.

AMI NO ACIDS	HYDROPATHY	SYMBOL
Ile	4.5	
Val	4.2	0.1
Leu	3.8	
Phe	2.8	
Cys	2.5	0.2
Met	1.9	
Ala	1.8	0.3
Gly	-0.4	
Thr	-0.7	
Ser	-0.8	0.4
Trp	-0.9	
Tyr	-1.3	
Pro	-1.6	0.5
His	-3.2	
Glu	-3.5	
Gln	-3.5	0.6
Asp	-3.5	
Asn	-3.5	
Lys	-3.9	
Arg	-4.5	0.7

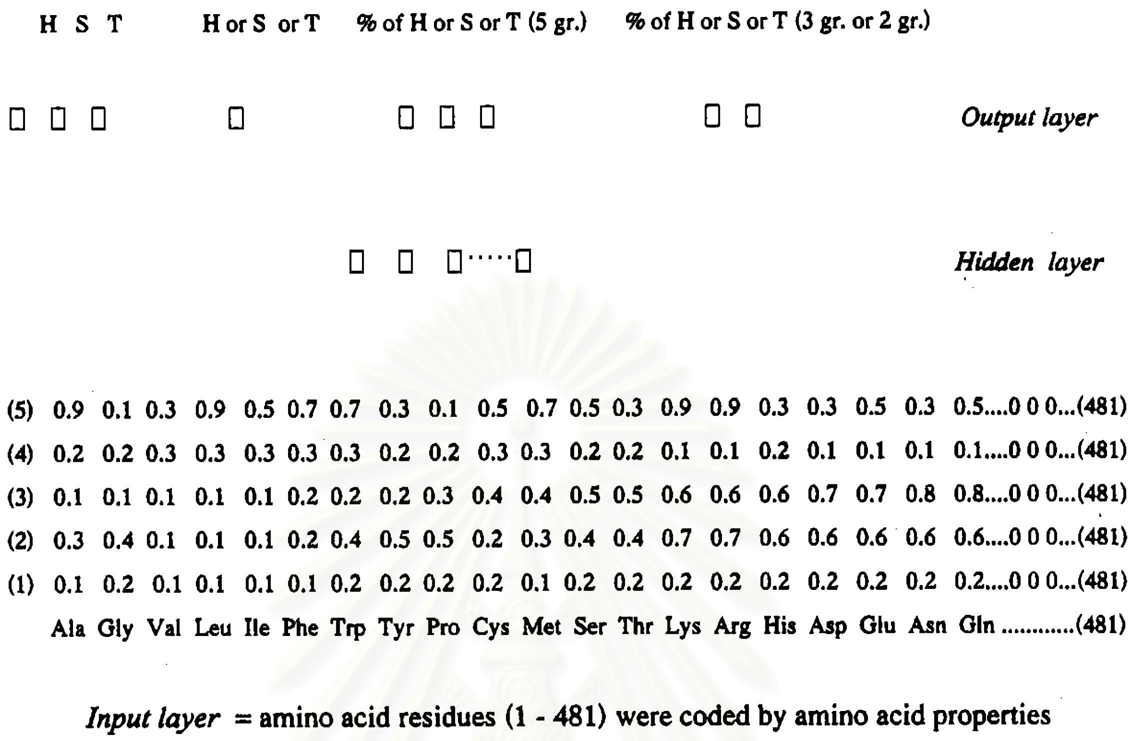
Table 3.4 Five groups of amino acids which were divided by relative helical tendencies.

Symbol	Amino acid residue	Relative stabilization of α-helical conformation (kcal/mol)
0.1	Pro	0
	Gly	~3
0.3	His	-0.06
	Asn	-0.07
	Thr	-0.11
	Val	-0.14
	Asp	-0.15
	Tyr	-0.17
0.5	Ile	-0.23
	Cys	-0.23
	Glu	-0.27
	Gln	-0.33
	Ser	-0.35
0.7	Phe	-0.41
	Trp	-0.45
	Met	-0.50
0.9	leu	-0.62
	Lys	-0.65
	Arg	-0.68
	Ala	-0.77

3.2.3 Input pattern and output pattern construction

The input pattern to the neural network amino acid sequence of each protein, after substitution by properties as previously described. First of all, the amino acid sequences were extracted from PDB files and replaced with properties using the amino acid residues symbols with computer programs SEQaa, SEQh2, SEQh7, SEQpho and SEQhe. These programs were used for replacing the amino acid residue symbols with seven groups of amino acid side chain property, two groups of hydropathy, seven groups of hydropathy , three groups of hydrophobicity and five groups of helical tendency respectively. To conform with PDB format, the input pattern was set to 13 columns of properties vector and the number of rows were dependent on the number of amino acid residues in each protein. Since the protein which has the longest amino acid sequence in this study was 481 amino acid residues in length, if follow that the number of input unit was 481 units. Because the number of input units should be a constant value, all input units in all input patterns for training and testing by Neural network should have the same number of input units. Thus, all input units in this study had 481 units. Any amino acid sequences that had shorter lengths than 481 residues were added with zeroes to make such sequences 481 residue long. Some examples of input patterns in this study are shown in Figure 3.3

The output patterns consist of 3 classes of secondary structure, helix, sheet and turn. These secondary structure data were obtained from the PDB file of each protein. For prediction of existence of these secondary structures in an amino acid sequence, the output layer of the networks consisted of 1 unit where corresponding to helix, sheet or turn. For prediction number of amino acid residue which should be helix, sheet or turn, the output depends on the range of the number of each secondary structure. For example, the number of helical residue of proteins in this study were in the range between 1- 100, the output has 5 groups with the range 1-20, 21-40, 41-60, 61-80 and 81-100 of the number of amino acid residues. This output had 3 units for each group of range.


H S T
H or S or T
% of H or S or T (5 gr.)
% of H or S or T (3 gr. or 2 gr.)
Output layer
Hidden layer
Input layer = amino acid residues (1 - 481) were coded by amino acid properties

(5)	0.9	0.1	0.3	0.9	0.5	0.7	0.7	0.3	0.1	0.5	0.7	0.5	0.3	0.9	0.9	0.3	0.3	0.5	0.3	0.5....0 0 0...(481)
(4)	0.2	0.2	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.3	0.3	0.2	0.2	0.1	0.1	0.2	0.1	0.1	0.1	0.1....0 0 0...(481)
(3)	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.4	0.4	0.5	0.5	0.6	0.6	0.6	0.7	0.7	0.8	0.8....0 0 0...(481)
(2)	0.3	0.4	0.1	0.1	0.1	0.2	0.4	0.5	0.5	0.2	0.3	0.4	0.4	0.7	0.7	0.6	0.6	0.6	0.6	0.6....0 0 0...(481)
(1)	0.1	0.2	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.1	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2....0 0 0...(481)
	Ala	Gly	Val	Leu	Ile	Phe	Trp	Tyr	Pro	Cys	Met	Ser	Thr	Lys	Arg	His	Asp	Glu	Asn	Gln ....(481)

Figure 3.3 Protein structure prediction networks. Units in the network are represented by squares, connection between units by solid lines. In input layer, shown at the bottom of the figure, there are 481 input units which were amino acid sequence of each proteins coded by amino acid properties : (1) two groups of hydropathy coded, (2) five groups hydropathy coded, (3) eight groups of amino acid side chain properties coded, (4) three groups of hydrophobicity coded and (5) three groups of helical tendencies coded. All input units are connected to every hidden units which were also connected to all output units. The networks with 3 output units were used for prediction of the existence of helix&sheet&turn in the same network and percent (5 gr.) of helix or sheet or turn. The networks with 2 output units were used for prediction of percent (3 groups) of helix or sheet or turn and percent (2 groups) of helix or sheet. The networks with one output unit were used for prediction of the existence of helix or sheet or turn in separate networks.

3.2.4 Neural Networks

All neural networks model used in this study are three-layer feed forward neural networks the SNNS neural network simulator software. The networks are fully connected from one layer to the next. The first layer, the middle layer and the last layer are input, hidden and output layers respectively. Each unit in the neural network accepts a number of inputs from previous layer or from external data in the case of the input layer. Each input unit is multiplied by a weight *Wij*, which represents the strength of the connection between 2 units *i* and *j* , and the total is offset by the bias , b_i, of the unit :

$$input_i = \Sigma_j W_{ij} + b_i \qquad (1)$$

The output is a result from input processing. This processing is a continuous nonlinear activation function that switches between 0 and 1:

$$output = \frac{1}{1 + e^{-inputi}} \qquad (2)$$

The independent variables in these functions are the biases of individual units and the weights between every pair of units in adjacent layers. The initiation values of these variables was randomly picked.

The input patterns for training and testing in were the amino acid sequences substituted by properties and the output pattern was the secondary structure of each protein as previously described. For training, the networks were trained by back-propagation which is used for adjusting the weights and biases, using 70 input patterns (70 amino acid sequences) and 481 units for each pattern. After training, the test set (28 amino acid sequences) was examined and the predicted outputs were compared with the observed outputs from PDB.

To determine the suitable number of hidden units which would produce the most accurate results, 7, 35, 70, 100, 120 and 140 hidden units were employed in each set of trial.

3.2.5 Training and Testing

The input and output patterns as described in 3.2.3 were saved as a learn pattern file and test pattern file (t.pat and te.pat). The learn pattern files were used for training the networks, while, the test pattern files were used for testing.

The network files for training were created by "bignet" program in SNNS. This program is a generator for special 3 layered feedforward networks. After training the trained networks were saved as "file.net"

3.2.6 Measurement of accuracy prediction.

$$\text{prediction accuracy} = \frac{\text{the total number of protein predicted correctly}}{\text{total number of proteins for testing}} \times 100 \text{ percent}$$

3.3 Structure Prediction Problem as a Classification Problem

3.3.1 Geometical Meaning of Neural function

Based on the output function of implemented by the sigmoid function $\frac{1}{1 + e^{-\Sigma w_i x_i}}$, the function of a neuron can obviously be considered as a hyperplane whose location in the n-dimensional space is captured by $\Sigma w_i x_i$. Each input pattern is, therefore, a vector in the n-dimensional space. Hence, the hidden neurons are acting as a set of separating hyperplanes.

For examples, in a two-dimensional space, suppose we have these input vectors (1,2), (2,2), (5,5) and (4,5).

Vectors (1,2) and (2,2) are in class A. Vector (5,5) and (4,5) are in class B.

To classify these vectors into their correct classes (A and B) by a neuron, a good separating line is required. The value of each w_i can be obtained as follows.

From linear line equation $y = ax + b$

From Figure 3.4 $y = 4 \qquad x = 6$

$$\therefore \quad 4 = b$$

$$\therefore \quad 0 = 6a + 4$$

$$\therefore \quad y = -\frac{4}{6}x + 4$$

$$\therefore \quad f = y + \frac{4}{6}x - 4$$

$$w_1 = 1 \qquad w_2 = \frac{2}{3} \qquad bias = -4$$

From the examples, the actual meaning of learning of each neuron is finding the appropriate vale of each w_i to locate the hyperplane in between two separated classes. Thus, the learning process is the adjustment of a and b using x and y for teaching Where, w_1, w_2 and bias are a, b and c in the linear line equation respectively.

$$L_1 = ax_1 + bx_2 + c$$

$$L_2 = w_1x_1 + w_2x_2 + bias$$

In case of complex inputs pattern such a protein structure prediction, we need more than one separating line to classify the desired outputs or hyperplanes.

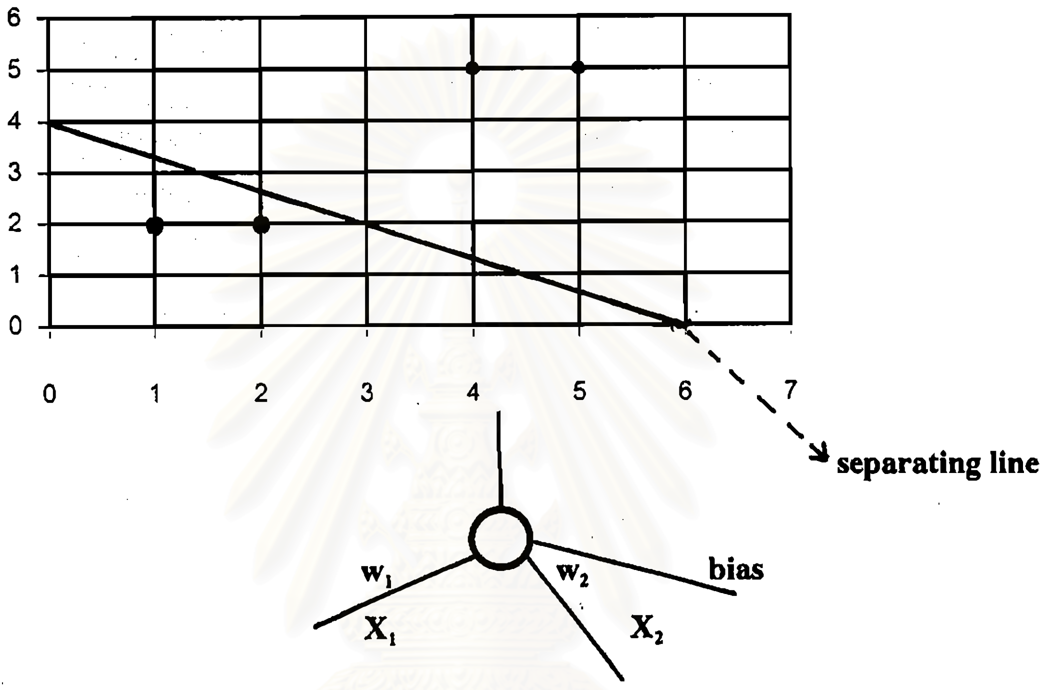


Figure 3.4 Vectors in 2-dimensional space which are separated into hyperplane by a separating line.

3.3.2 Classification of Protein Structures

We can transform the problem of structure prediction to the problem of classifying the difference structures into their corresponding classes based on their essential properties. Protein structures acting as a set of hyperplanes. Thus the protein properties are collected to form a vector in n-dimensional space. Neural Network is separating hyperplanes. The suitable features are required for training the network to separate these proteins to the correct classes of structures. Thus, the actually main problem is the extraction of protein properties to obtained the desired structures. If we have the suitable properties, it mean that the different protein structures are obviously separated into the different dimensional space. On the other hand, the neuron can have a good example for learning and give rise a correct answer.