Use of Neural Networks to Model Molecular Structure and Function

A thesis submitted for the degree of Doctor of Philosophy of the University of London

Jonathan Darrell Hirst

October 1993

Biomolecular Modelling Laboratory
Imperial Cancer Research Fund
44 Lincoln's Inn Fields
London WC2A 3PX

and

The Department of Biochemistry and Molecular Biology
University College London
Gower Street
London WC1E 6BT

Abstract

This thesis is a study of some applications of neural networks - a recent computer algorithm - to modelling the structure and function of biologically important molecules.

In Chapter 1, an introduction to neural networks is given. An overview of quantitative structure activity relationships (QSARs) is presented. The applications of neural networks to QSAR and to the prediction of structural and functional features of protein and nucleic acid sequences are reviewed. The neural network algorithms used are discussed in Chapter 2.

In Chapter 3, a two-layer feed-forward neural network has been trained to recognise an ATP/GTP-binding local sequence motif. A comparably sophisticated statistical method was developed, which performed marginally better than the neural network.

In a second study, described in Chapters 4 and 5, one of the largest data sets available for developing a quantitative structure activity relationship - the inhibition of dihydrofolate reductase by 2,4-diamino-6,6-dimethyl-5-phenyldihydrotriazine derivatives - has been used to benchmark several computational methods. A hidden-layer neural network, a decision tree and inductive logic programming have been compared with the more established methods of linear regression and nearest neighbour. The data were represented in two ways: by the traditional Hansch

parameters and by a new set of descriptors designed to allow the formulation of rules relating the activity of the inhibitors to their chemical structure.

The performance of neural networks has been assessed rigourously in two distinct areas of biomolecular modelling: sequence analysis and drug design. The conclusions of these studies are presented in Chapter 6.

Contents

List of Figures

List of Tables

List of Abbreviations

Å	1 Ångstrom = 0.1 nanometers
ATP	adenosine triphosphate
CASE	computer automoated structure evaluation
CoMFA	comparative molecular field analysis
DHFR	dihydrofolate reductase
DNA	deoxyribonucleic acid
E. coli	*Escherichia coli*
FORTRAN	formula translation language
fp	false positives
GOR	Garnier Osguthorp Robson
GTP	guanosine triphosphate
Ig	immunoglobulin
ILP	inductive logic programming
LR	linear regression
mRNA	messenger ribonucleic acid
MSA	molecular shape analysis
MSD	minimal steric difference
MTD	minimal topological difference
NMR	nuclear magnetic resonance
PCA	physicochemical attribute
QSAR	quantitative structure-activity relationship
RNA	ribonucleic acid
rms	root mean square
snRNA	small nuclear ribonucleic acid

Single letter code for DNA bases

A	adenosine
C	cytosine
G	guanine
T	thymine

Single letter codes for amino acids

A	alanine
C	cysteine
D	asparatic acid
E	glutamic acid
F	phenylaniline
G	glycine
H	histidine
I	isoleucine
K	lysine
L	leucine
M	methionine
N	asparagine
P	proline
Q	glutamine
R	arginine
S	serine
T	threonine
V	valine
W	tryptophan
Y	tyrosine

Acknowledgements

I thank my supervisor, Mike Sternberg, for his invaluable support and interest in my work, and for providing an excellent environment in which to work.

I thank the other members of the Biomolecular Modelling Laboratory: Alexei Adzhubei, Paul Bates, Paul Harrison, Suhail Islam, Richard Jackson, Ross King, Richard Lewis, Stephen Pickett, Mansoor Saqi, Paul Snape, Peter Walls for their advice and friendship. The machine learning studies used as a comparison in Chapters 4 and 5 were done by Dr. Ross King.

I am grateful to the Imperial Cancer Research Fund for a generous three-year studentship.

I thank my supervisor at University College, Janet Thornton, for her support and interest.

My parents and my wife, Pilar, have been full of support and love over the years.

Chapter 1

Introduction

1.1 Synopsis

In this thesis, empirical modelling by neural networks is investigated, with particular reference to quantitative structure-activity relationships (QSARs), where the drug activity is related to chemical structure, and biomolecular sequence analysis, where structure and function are related to primary sequence. These two areas are reviewed in this chapter, with a general overview of QSAR and a more specific discussion of sequence analysis based on neural network applications. The concepts underlying neural networks are introduced.

1.2 Introduction

A fundamental objective of scientific research is the recognition of unifying relationships among data. Such relationships may be developed from theories of molecular behaviour, such as the ideal gas law or the Schrödinger equation. However, the complexity of biochemical processes often precludes theoretical calculation and also direct experimental measurement. Empirical models are thus especially important in the biological sciences.

This thesis will consider two areas of active research, where empirical modelling is of particular interest: the study of quantitative structure-activity relationships (QSARs) and the analysis of biomolecular sequences. In QSAR, the activity of a drug is predicted from its chemical structure, through the analysis of drugs with similar modes of action and known activity. In sequence analysis, structure or function is predicted from the primary sequences of proteins or nucleic acids, through the analysis of sequences with known structure or function. The aim of this thesis is to investigate the use of neural networks for modelling molecular structure and function; sequence analysis and QSAR studies serve as illustrative and pertinent examples. The major part of the thesis focuses on QSAR, so after an introduction to the neural network methodology, a short overview of QSAR is presented. Biomolecular sequence analysis is then discussed with specific reference to neural network applications.

1.3 Methodology

1.3.1 Introduction to neural networks

A neural network is, basically, a computer program that can detect patterns and correlations in data. Fundamental to the approach is the concept of parallel processing - many units performing simple tasks in unison. The success of this methodology in the recognition and classification of patterns, and the contrast of these parallel learning algorithms with conventional serial computing has attracted the attention of, amongst others, scientists interested in biomolecular modelling.

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

Current mathematical models stem from the work of McCulloch and Pitts (1943), Hebb (1949), Widrow (1960), Rosenblatt (1962), and others. Interest in neural networks was curtailed when Minsky and Papert (1969) highlighted a major limitation of the approach, proving that only problems with linearly separable solution spaces could be solved by the neural network algorithms of the time. It was not until the implementation of a new algorithm, called the backpropagation of errors (Rumelhart *et al.*, 1986a), that this limitation was widely seen to have been overcome. Although backpropagation is not a plausible model of learning in brains (Rumelhart *et al.*, 1986a; Crick, 1989), the prospect of tackling previously unsolved computational problems using the power of backpropagation and other work in the field, including that of Kohonen (1984), Grossberg (1986), and Hopfield (1982, 1984, 1986), rekindled the excitement about neural networks. Schillen (1991) lists many of the current areas of application, including speech recognition (Sejnowski and Rosenberg, 1987; Clarke *et al.*, 1991) and vision (Lehky and Sejnowski, 1988), and an extensive list of references can be found in a book by Simpson (1990).

Neural networks have several potential advantages that have encouraged their application in many fields. They incorporate both positive and negative information - both data with the feature of interest and without that feature are used to train the neural network. They are able to detect second- and higher-order correlations in patterns, *i.e.*, they are non-linear. A preconceived model is not required - the neural network automatically determines which input variables are important.

A neural network consists of a number of simple, connected computational units that operate in parallel and can be trained to map a set of input patterns on to a set of output patterns. This computational paradigm is based on a simplified model of a biological neuron. A modelled neuron (or unit) has the basic functionality of a biological neuron: it takes signals from other units, if the sum of these signals is greater than a threshold, it produces a signal, which is passed on to other units (Figure 1.1). Each unit operates independently, but the units are connected to one another with a weight, which is a real number, and these weights determine the behaviour of the neural network. Each unit transmits a signal to its neighbours through the connections. The value of the output signal depends upon the activation (or state) of the unit, which is a real number associated with the unit. This dependence is expressed in an output transfer function, most commonly, a sigmoid function, such as the logistic function (Figure 1.2). The activation of a unit is a function of the outputs of the units to which it is connected. There are three types of unit: input units which receive signals from external sources and send signals to other units; output units which receive signals from other units and send signals to the environment; and hidden units which have no direct contact with the environment and, hence, they receive inputs from other units and send their output signals to other units.

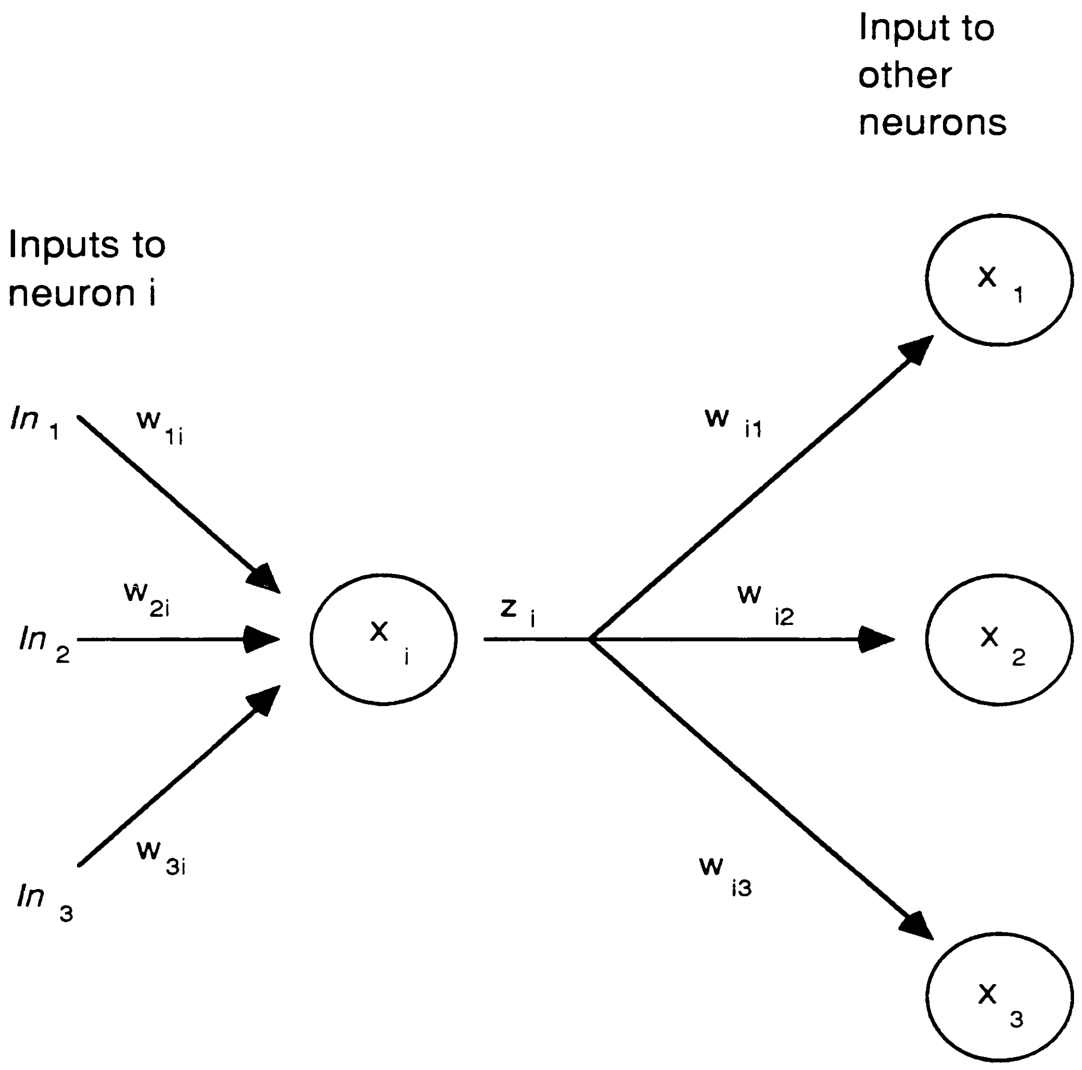


Figure 1.1 A simplified model of a biological neuron. The activation of an input neuron is represented by In_i, the weights connecting units i and j are denoted by w_{ij}; the neurons are labelled x_i, x_1, x_2, and x_3.

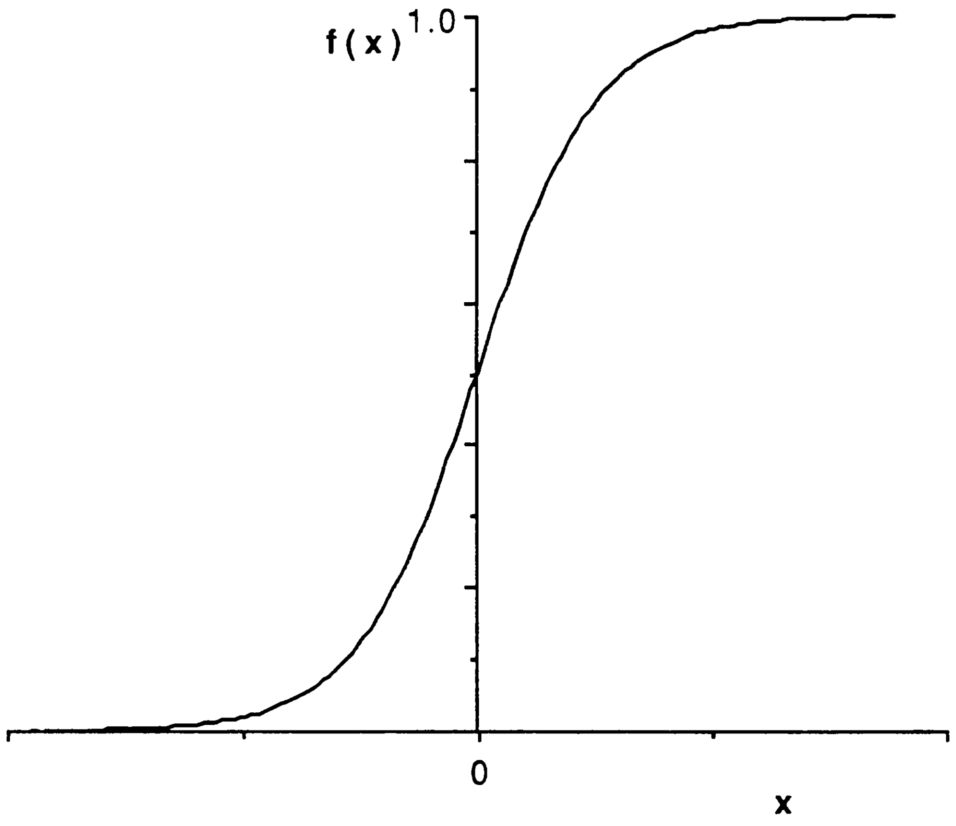


Figure 1.2 The logistic function, $f(x) = \dfrac{1}{1 + e^{-x}}$.

The architecture (or topology) of a network is formed by organising the units into layers. There can be connections between units in the same layer, and connections between units in different layers. Inter-layer connections can allow propagation of signal in one direction (feed-forward) or in either direction (feedback). The neural network learns by altering the values of the weights in a well defined manner, described by a learning rule. There are two general types of learning. Supervised learning incorporates an external teacher and requires a knowledge of the desired responses to input signals. The aim is to minimise the error between the desired and computed output unit values. In statistics, regression and discrimination are of this type. Unsupervised learning uses no external teacher and is based upon local information only. It self-organises data presented to the network and detects the emergent collective properties (Kohonen, 1984; Hopfield, 1982). The analogous paradigms in statistics are clustering and classification.

In this thesis, it has only been possible to study a small number of the diverse range of neural networks. All the applications have used supervised learning, and none of them have employed feed-back architectures. Even within this subset of neural networks, a number of decisions still have to be made. These include the choice of learning algorithm, the architecture of the neural network, the number of input units, the possible use of hidden layers, and the method of encoding data. Some of the more complicated neural networks can find arbitrarily complex mappings between input patterns and output classifications, but this process is poorly understood, and, as a result, the above choices are not automatic. In the following sections, these choices are considered in more detail.

1.3.2 Learning algorithms

1.3.2.1 The perceptron algorithm

A neural network with no hidden layers can be trained using the perceptron algorithm (Rosenblatt, 1957). For simplicity consider a two-layer perceptron, *i.e.*, one with no hidden units, that decides whether an input belongs to just one of two classes, denoted A and B (Figure 1.3). The single output unit computes a weighted sum of the input units, subtracts a threshold, θ, and converts the result to +1 or -1, using an output transfer function. The decision rule is to respond class A if the output is +1 and class B if the output is -1. The behaviour of such networks can be analysed using a plot of the decision regions created in the multi-dimensional space spanned by the input variables (Lippman, 1987). These decision regions specify which input values result in a class A and which result in a class B response. The perceptron forms two decision regions separated by a hyperplane (Figure 1.4), and the equation of the boundary line depends on the connection weights and the threshold. The perceptron algorithm is given in Chapter 2. Rosenblatt (1962) proved for two-layer neural networks that if the inputs presented from the two classes are separable (that is they fall on opposite sides of a hyperplane), then the perceptron algorithm converges and positions the decision hyperplane between those

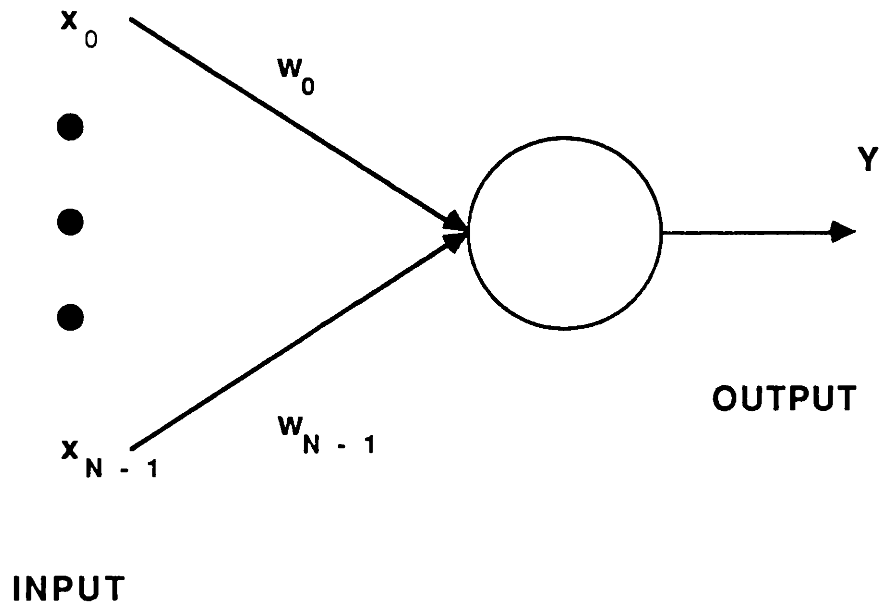


Figure 1.3 A schematic diagram of a two-layer perceptron, with N input units, denoted by x , N weights denoted by w, and one output unit, denoted by Y.

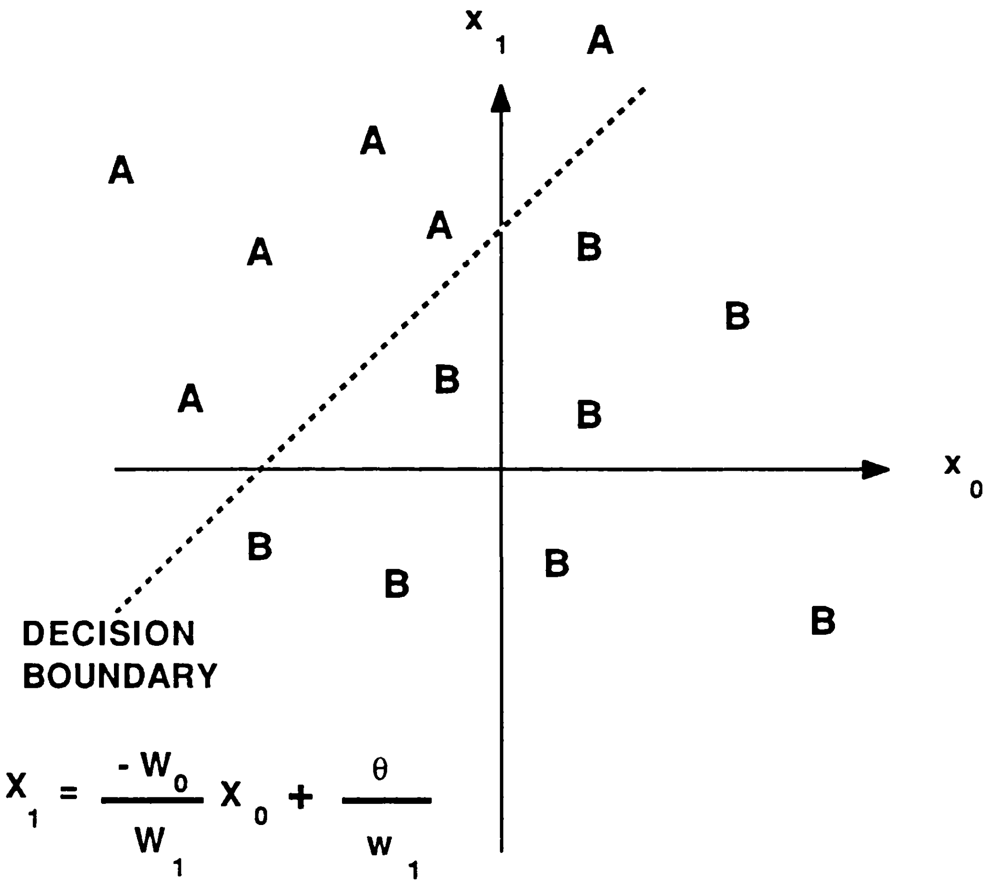


Figure 1.4 The decision boundary formed by a two-layer perceptron separating two classes, A and B, by two input co-ordinates, x_0 and x_1. The equation of the line is given as a function of the weights w_0 and w_1, and the threshold, θ.

two classes. Rosenblatt was unable to extend this to architectures with three or more layers. Two-layer neural networks are not appropriate when classes cannot be separated by a hyperplane, as in the exclusive OR problem (Figure 1.5). For these non-linearly separable problems multi-layer networks trained with a more involved algorithm are required.

1.3.2.2 The backpropagation of errors algorithm

The backpropagation of errors (Rumelhart *et al.*, 1986a) is such an algorithm. It performs the input to output mapping by adjusting weight connections according to the difference between the computed and desired output unit values. A cost function is minimised, typically the squared difference between the computed output values and the desired output values, across all the patterns in the data set. The weight adjustments are derived from the change in the cost function with respect to the change in each weight. The backpropagation algorithm is powerful, because this derivation is extended to find the equation for adapting the connections between the input and hidden layers of a multi-layer network, as well as the penultimate layer to output layer adjustments. The extension to the hidden layer adjustments is based on the realisation that the error of each unit in the penultimate-layer is a proportionally weighted sum of the errors produced at the output layer. The basic algorithm for the three-layer elementary backpropagation topology (Figure 1.6) is outlined in Chapter 2, along with some considerations with regard to its implementation.

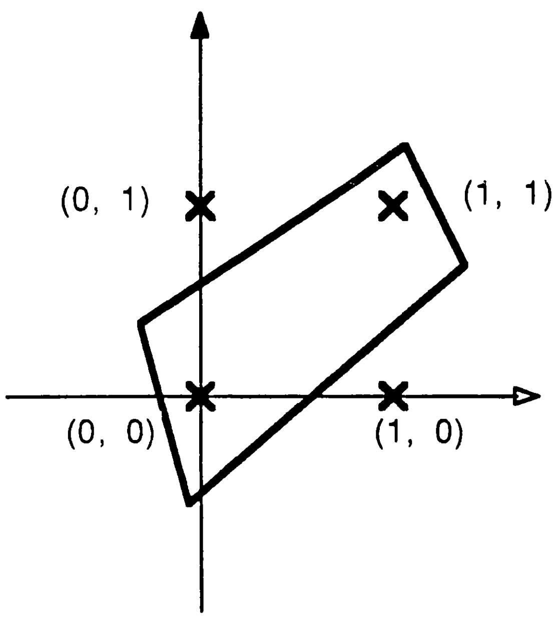


Figure 1.5 A graphical representation of the exclusive OR problem - if the two inputs are (0,0) or (1,1), the output is 0, and if the two inputs are (0,1) or (1,0), the output is 1. The decision region required to separate the two classes is schematically shown, and it cannot be a single line.

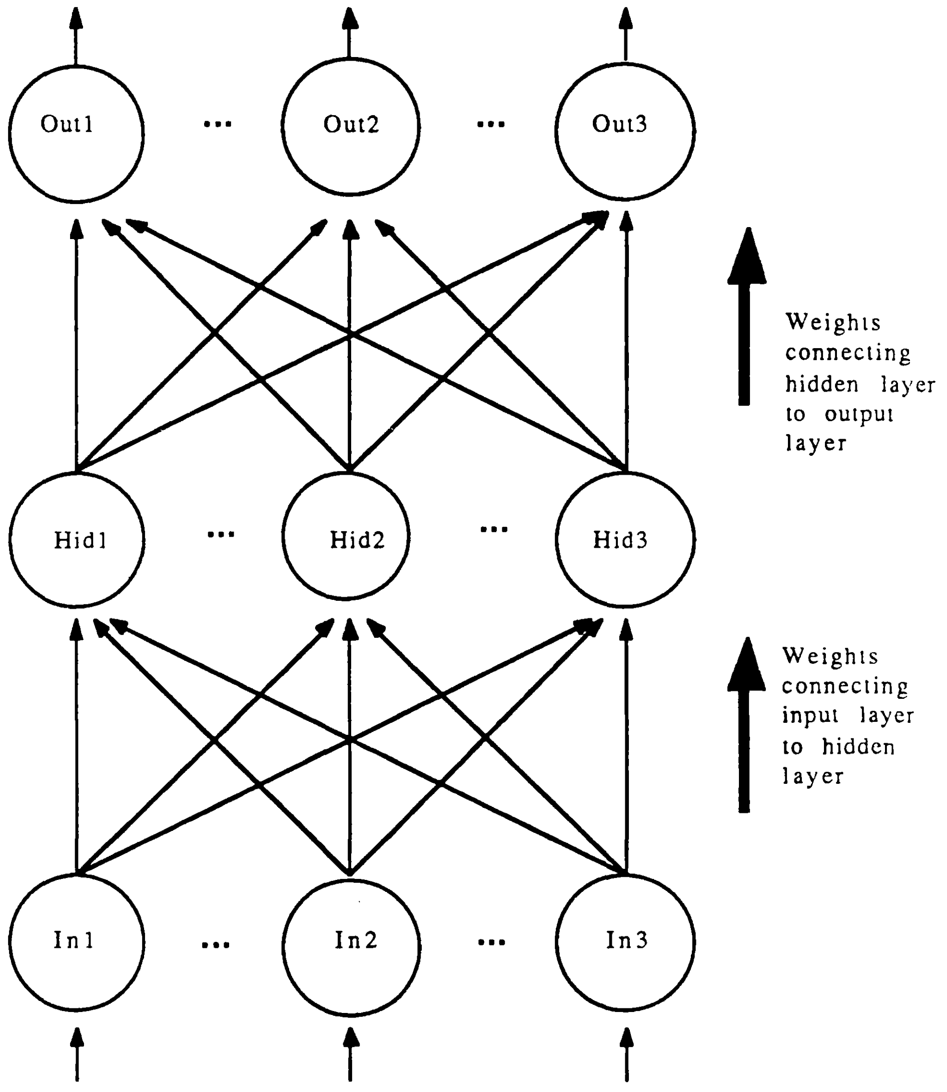

Out1
Out2
Out3
Weights connecting hidden layer to output layer
Hid1
Hid2
Hid3
Weights connecting input layer to hidden layer
In1
In2
In3

Figure 1.6 The elementary backpropagation topology.

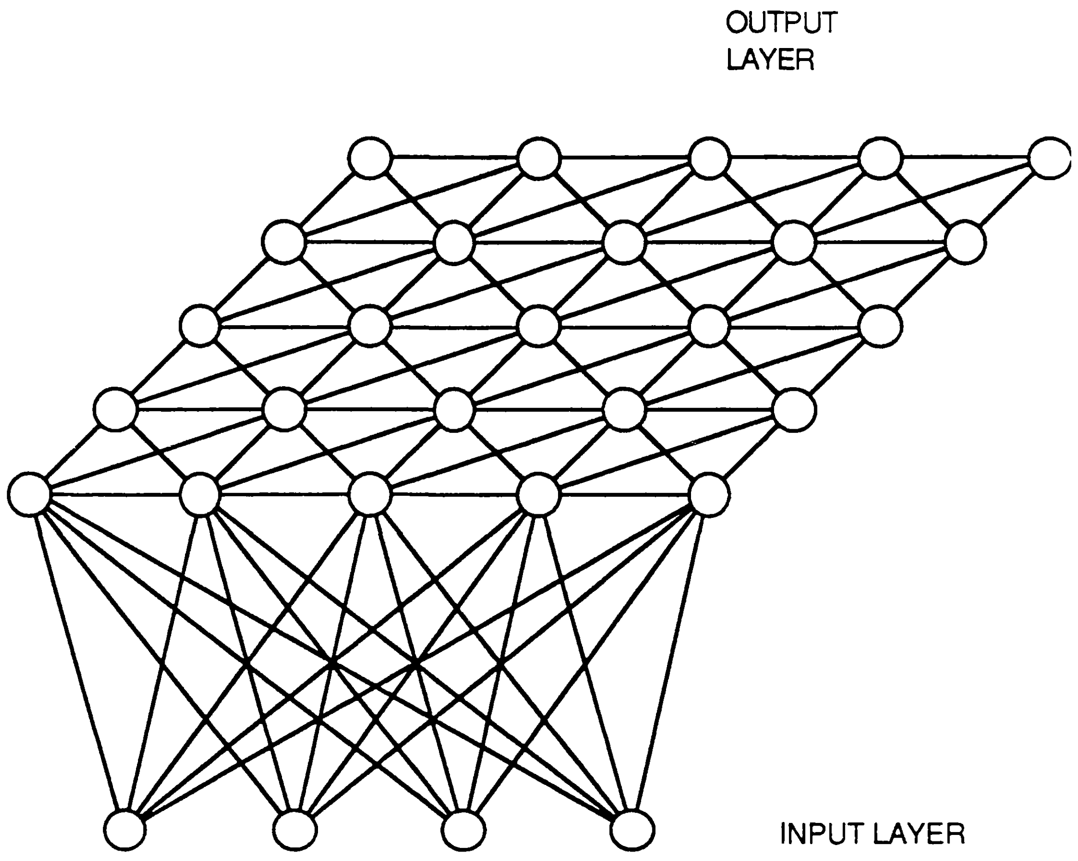


Figure 1.7 A schematic representation of a Kohonen self organising feature map. The output layer is a two-dimensional array of units. For clarity, not all the connections between output units are shown, and only input connections to the first row of output units are shown.

1.3.2.3 The Kohonen net

The other general type of learning, unsupervised, is used in the Kohonen self organising feature map (Kohonen, 1984). The output units are arranged in a two dimensional grid and extensively interconnected (Figure 1.7). Every output unit is also connected to every input unit. Continuous-valued input patterns are presented sequentially in time without specifying the desired output. After enough input patterns have been presented, weights will specify cluster or vector centres that sample the input space such that the point density function of the vector centres tends to approximate the probability density function of the input vectors (Kohonen, 1984). In addition, the weights will be organised such that topologically close units are sensitive to inputs that are physically similar. Despite the importance of the Kohonen network, it has not been studied in this thesis and will not be discussed here in detail.

1.4 An overview of QSAR

1.4.1 The aim of QSAR

The drug design problem that QSAR studies ultimately seek to answer is "How does one increase the activity of a drug, by systematic modification of its chemical structure?". A QSAR, therefore, attempts to describe the activity within a set of compounds by a mathematical formalism which incorporates structure-dependent parameters. QSAR studies require the synthesis and characterisation of a number of related molecules (congeners), which have the same basic structures but differ, for instance, in the substituents on aromatic rings. Although the classical QSAR approaches were introduced empirically, they can be derived in terms of an extrathermodynamic approximation - addivity of substituent effects and separability of different effects (Fujita, 1990).

1.4.2 The Hammett equation

Characterisation of substituent effects and the use of this information to analyse chemical reactions dates back to Hammett (Hammett, 1940). Hammett correlated the rate of hydrolysis of *meta*-substituted and *para*-substituted benzoates with σ, calculated from the dissociation constants of the corresponding benzoic acids:

$$\log(K_X/K_H) = \rho\sigma_X, \qquad \text{[eqn. 1.1]}$$

where K_H is the rate constant for the unsubstituted molecule, K_X is the rate constant for the derivative. σ_X refers to the electronic effect of the substituent relative to hydrogen and is a parameter applicable to many different types of reaction - characterised by different values of ρ - whose relative rates depend on the degree of electron release or withdrawal by that substituent. Taft (Taft, 1952; Taft, 1953) added a steric parameter, E_S, to the Hammett equation, to obtain a relationship that could be applied to *ortho*-substituents as well.

1.4.3 The Hansch Approach

In 1962, Hansch *et al.* correlated the biological activity of phenoxyacetic acids with Hammett substituent constants and partition coefficients. The use of partition coefficients has developed (Hansch and Fujita, 1964; Hansch, 1969; Leo *et al.*, 1971; Hansch and Leo, 1979; Hansch, 1981; Blaney *et al.*, 1984) to become probably the most popular method of QSAR. In most applications, the Hansch equation has the form:

$$\log(1/C) = \sum c_{0,j} + c_{1,j}\sigma + c_{2,j}\pi + c_{3,j}\pi^2 + c_{4,j}E_S, \qquad \text{[eqn. 1.2]}$$

where C is the drug concentration for a chosen standard biological effect; c_{ij} are regression coefficients to be determined by iterative curve fitting by a least squares procedure, π is the substituent hydrophicity constant, σ is the Hammett substituent constant and E_S is the Taft steric parameter; the summation over j indicates that there are terms for each substituent. The Hansch approach is, thus, based on the formation of an empirical model of drug action that uses parameters related to linear free energy as the independent variables. The basic assumption is that the variations in biological activity arising from the modifications of molecular structures within a congeneric series can be correlated with the resulting changes in physicochemical properties - comprising hydrophobic, electronic and steric components. The Hansch approach is discussed further in Chapter 4, where it is used as a benchmark.

1.4.4 Free-Wilson Analysis

The Free-Wilson method (Free and Wilson, 1964) also assumes that biological activity is dependent on the additive properties of the substituents on a parent molecular structure. In the Fujita-Ban modification of this method (Fujita and Ban, 1971):

$$\log(1/C) = \Sigma a_i X_i + \mu_0, \qquad \text{[eqn. 1.3]}$$

where C is as previously defined, a_i is the group contribution of the i^{th} substituent to the activity of the substituted molecule, X_i is unity if substituent i is present and zero otherwise, and $\mu_0 = 1/C$ for the parent compound. A least squares procedure is used to determine a_i and μ_0; no physicochemical parameters are employed. Indicator variables are used in multiple linear regression analysis to model specific features that cannot be described by continuous variables. They take the value of one or zero, depending on the presence or absence of the feature. Free-Wilson analysis can be considered as a regression analysis approach using only indicator variables (Kubinyi, 1990).

1.4.5 Principal component analysis

Principal component analysis is a technique for reducing the effective dimensionality of a dataset, and can be of use in QSAR for variable selection (Martin, 1978). It treats all variables in the analysis equally, unlike regression, where a single dependent variable is to be explained by one or more independent variables.

Given a set of n variables, where $n \geq 2$, principal component analysis rotates these variables in the n-dimensional parameter space to map them onto a new set of n variables, such that the first variable in this new set contains the greatest possible fraction of the total variance, the second contains the greatest possible fraction of the remaining variance, and so on. The dimensionality of the dataset is reduced by retaining only those principal components which contain a significant fraction of the original variance. The rotation matrix required is the matrix of eigenvectors of the covariance matrix.

1.4.6 Three-dimensional QSAR

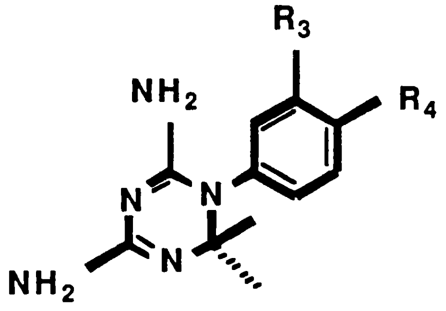
Several approaches have extended the traditional methods of deriving QSARs, by modelling the drugs using more complicated three-dimensional descriptions.

1.4.6.1 Minimal Steric Difference (MSD)

This method, developed by Simon (1974), is based on the assumption that ligand-site interaction is a linearly decreasing function of the steric misfit of the ligand and the site acceptor cavity. An approximation of the shape of the cavity, called the standard, is the natural effector molecule or the most active structure in the set of compounds under study. The structural formulae of the other molecules are superimposed on the standard. The MSD value of a structure is the number of non-superimposable atoms, neglecting hydrogen, with atoms from the third row of the periodic table weighted by a factor of 1.5, and higher period elements by a factor of two. A modified version of the MSD procedure, the minimal topological difference (MTD) method (Simon, 1977), defines the standard as the hypermolecule formed by the superimposition of all the structures under consideration, ignoring hydrogen atoms. This works best when there are clear steps in activity. Small changes may be controlled primarily by changes in electrostatic complementarity or conformational space.

1.4.6.2 Molecular Shape Analysis (MSA)

Information relating to the three-dimensional structure of the drugs is used to compare differences in volumes and fields of ligands in the molecular shape analysis (MSA) method (Hopfinger, 1980). The most stable conformers of the congeners in the dataset are determined by molecular mechanics. In a study of the inhibition of dihydrofolate reductase (DHFR) by triazines (I) (Hopfinger, 1981), the general measure of shape



I

similarity was the common overlap steric volume, V_0, between pairs of C_6H_5X fragments, when the respective two identical triazine rings were superimposed. V_0 was the sum of the van der Waals sphere intersection volumes between pairs of non-hydrogen atoms. It was concluded from a regression analysis that the most active molecules would adopt conformations such that the angle between the planes of the triazine ring and the benzene ring was 310°. More recent applications include the molecular shape analysis of a series of indanone-benzylpiperidine inhibitors of acetylcholinesterase (Cardozo *et al.*, 1992).

1.4.6.3 Distance geometry

The distance geometry method (Ghose and Crippen, 1983; Ghose and Crippen, 1990) uses the three-dimensional structure and atom-based physicochemical properties of the ligand molecules to develop a model for the binding site cavity. The distance geometry representation expresses the flexibility of a molecule by a distance range matrix showing the upper and lower bounds on the distance between atom pair. The underlying idea is based on the following consideration. Suppose there are two flexible ligand molecules m and n, and the atoms m_i and m_j of molecule m and atoms n_i and n_j of molecule n occupy the same respective regions of the active site. The distance between the i^{th} and j^{th} atoms in the two molecules must be very close in their active conformations. Since in the distance geometry representation of the flexible molecules atomic distances have ranges, the active conformations should be represented by a common distance range. If there are several molecules, such comparisons will gradually decrease the range, and better define the possible conformational region. Ultimately, analysis of these distances will give the three-dimensional structure of the site pockets accommodating the ligand atoms.

1.4.6.4 Comparative Molecular Field Analysis (CoMFA)

Comparative molecular field analysis (CoMFA) compares molecules on the basis of the field that they present to their surroundings by mapping the field on a grid (Cramer *et al.*, 1988). The procedure can be summarised as:

(1) Postulate a set of orientation rules.

(2) Align the set of molecules and establish a lattice which surrounds the set in potential receptor space.

(3) For each molecule calculate the field which a probe atom would experience at each lattice point.

(4) Use partial least squares statistics to determine a minimal set of lattice points necessary to distinguish the set of compounds according to their measured activities.

(5) Check the predictive value of the lattice model by successively eliminating observations and determine the predictive value of the newly derived model.

(6) Repeat steps (4) and (5) to find a model of high predictive value.

More traditional physical data may be used to augment the steric and electrostatic field generated by CoMFA (McFarland, 1992). It is difficult, however, to appropriately weight the electrostatic and steric variables. Also, the superposition is crucial.

1.4.6.5 Molecular similarity

As is evident from the above discussion, methods of comparing molecules are central to three-dimensional QSAR. Similarity indices may be based on electron density calculated *ab initio* (Bowen-Jenkins *et al.*, 1985) or using semi-empirical methods (Hodgkin and Richards, 1986; Burt and Richards, 1990; Good *et al.*, 1993). Molecular size and shape are defined by electron density, in that the nuclear positions determine the electron density. Atomic co-ordinates may be used directly to provide measures of similarity (Meyer and Richards, 1991). The application of simulated annealing algorithms (Kirkpatrick, 1983) to calculating molecular similarity based on atomic positions has also been investigated (Barakat and Dean, 1990a, 1990b, 1991; Papadopoulos and Dean, 1991).

1.4.7 Non-parametric techniques

The approaches discussed so far involve parametric regression analysis. One of the common assumptions of parametric methods is that the data are normally distributed. This assumption is avoided in non-parametric techniques, many of which originate from the fields of pattern recognition and artificial intelligence.

1.4.7.1 Pattern recognition

Pattern recognition techniques seek to detect and predict obscure properties of objects from indirect measurements made on those objects. In an early application of pattern recognition, the odour of a molecule was predicted from its shape, as modelled by the silhouette of a scale molecular model (Amoore *et al.*, 1967). The silhouettes were scanned with 4096 random lines which were assigned a binary number depending on the number of intersections the line made with the silhouette. This representation was used to calculate the similarity between unknown patterns and learned examples.

The interpretation of chemical data using pattern recognition has been discussed by Kowalski and Bender (1972), who subsequently analysed two hundred drugs tested for activity in the solid tumour adenocarcinoma 755 screening system (Kowalski and Bender, 1974). Three pattern recognition methods were used, with approximately 90% correct responses, although the selection and representation of the data were later criticised (Mathews, 1975). A pattern recognition study relating the pharmacological activity of a compound to its mass spectrum (Ting *et al.*, 1973) also attracted some criticism (Perrin, 1974).

A linear learning machine, a forerunner of current neural networks, was compared with a nearest neighbour algorithm (see Chapter 4) for the prediction of antitumour activity of a structurally diverse set of compounds tested in an experimental mouse brain tumour system. Stuper and Jurs (1975) used a linear learning machine to classify psychotropic drugs as sedatives or tranquillisers with a predictive ability on unknowns of about 90%. This work and the field as a whole has been extensively reviewed (Stuper *et al.*, 1979; Jurs, 1986).

1.4.7.2 Artificial intelligence methods

Several artificial intelligence approaches have been developed for the manipulation and evaluation of chemical structures. In the CASE method (Computer Automated Structure Evaluation, Klopman *et al.*, 1984; Klopman and Ptchelintsev, 1993), substructural units of ten atoms or so are used to find structural features which may be correlated to biological activity. A symbolically based program, WIZARD (Dolata *et al.*, 1987; Leach *et al.*, 1988), searches the conformational space of a molecule to identify conformations near energy minima.

The machine learning program FLEMING, based on inductive logic, was used to predict inhibitors of thermolysin (Bolis, 1991). Inductive logic involves the formulation of rules that are consistent with the data, whereas deductive logic formulates relationships that must follow from initial axioms. A sample of active and inactive compounds, viewed as a set of positive and negative examples, permits the induction of a molecular model characterising the interaction between the drugs and the target molecule. Rule-induction has been suggested as complementary technique to conventional QSAR methods (A-Razzak and Glen, 1992). Here a modified ID3 algorithm (Quinlan, 1986) constructs a simple decision tree from a number of objects.

Another computer learning method, inductive logic programming (ILP), has been used to model the QSAR of trimethoprim analogues binding to DHFR from *E. coli* (King *et al.*, 1992). Physicochemical attributes (PCAs) were assigned heuristically to substituents, and were chosen to make the approach generally applicable to drug design problems. While not significantly better than the traditional QSAR, this method also produced rules that could provide insight into the stereochemistry of drug-DHFR interactions.

1.4.8 Neural network applications

In the last couple of years, work applying backpropagating neural networks to QSAR has considered the description of drug molecules in the formalism of Hansch (Hansch, 1969; Hansch *et al.*, 1962), which is simpler than some of the above approaches. The input is generally the parameters used by Hansch: molar refractivity and hydrophobic constants, for the relevant substituents. These values are usually scaled to lie between zero and unity. The output is the activity of the molecules for a given assay. Most applications have used hidden units.

Neural networks have been used to derive the QSAR of 16 carboquinone derivatives and their anticarcinogenic activity (Aoyama and Ichikawa, 1992; Aoyama *et al.*, 1990a; Tetko *et al.*, 1993), and the QSAR of the antihypertensive activity of 29 derivatives of arylacryloylpiperazine (Aoyama *et al.*, 1990b). This study was extended to the QSAR of 39 carboquinones, and the QSAR of 60 benzodiazepines (Aoyama *et al.*, 1990a). In the benzodiazepine study, three different assays were used for most of the drugs, giving 163 data examples. The neural network was compared to a regression analysis, and gave better results in 96 cases, worse results in 62 cases, and comparable results in 5 cases. Neural network analyses of the QSAR of 2,4-diamino-5-(substituted-) pyrimidines as dihydrofolate reductase (DHFR) inhibitors (So and Richards, 1992) and the QSAR of 2,4-diamino-6,-dimethyl-5-phenyldihydrotriazines as DHFR inhibitors (Andrea and Kalayeh, 1991) have suggested that neural networks can perform better than traditional regression methods, because they introduce cross-terms into the Hansch equation.

A cautionary note has been sounded by Livingstone and co-workers, who have discussed the dangers of over-fitting due relatively small data sets and large numbers of parameters (Livingstone and Salt, 1992; Livingstone and Mallanack, 1993). Wikel and Dow (1993) have used neural networks for selecting the variables to be considered in a QSAR. Neural networks have also been used to reduce the dimensionality of a data representation, by mapping the input onto itself via a smaller number of hidden units (Livingstone, 1991; Good *et al.*, 1993). If two hidden units are used, then the activity of these units is readily shown in graphical form.

1.4.9 Quantum theoretical methods

Various molecular properties may be calculated by quantum mechanics, which provides energies and wavefunctions for small molecules. The energies of different conformers may be determined. The wavefunction may be used to calculate electron density, electrostatic potential, multipole moments and other properties. The type of quantum mechanics calculation that can be performed depends on the number of electrons in the system and the available computer resources. For small systems it may be feasible to perform *ab initio* calculations; for larger systems, there are many semi-empirical methods (Richards, 1989); calculations including proteins involve further approximations, such as the use of partial charges (Hayes and Kollman, 1976). A more detailed survey of this huge field is given by Loew and Burt (1990).

1.4.10 Structure-based strategies

All the approaches reviewed so far assume no knowledge of the receptor site. If the structure of the receptor is known, from X-ray crystallography or NMR experiments, then drug design can be tackled using more sophisticated approaches (Goodford, 1984; Kuntz, 1992; McCammon, 1987). Graph theory as applied to molecules can be used to generate fragments that fit into the binding site (Lewis and Dean, 1989a, 1989b; Lewis, 1992; Chau and Dean, 1992a, 1992b, 1992c). A molecular docking program, DOCK (Kuntz, 1982), has recently been applied to discover inhibitors of thymidylate synthase (Shoichet *et al.*, 1993). The program GRID (Goodford, 1985), which determines probable interaction sites between probes with various functional group characteristics and the enzyme surface, was used to design sialidase-based inhibitors of influenza virus replication (von Itzstein, 1993). However, despite the elegance and increasing use of structure-based drug design methods, drug design in the absence of the structure of the receptor remains an important and, in some ways, more difficult problem.

1.5 Sequence analysis

Chapter 3 presents an application of neural networks to a sequence analysis problem. In this section, sequence analysis by neural networks is reviewed, to provide a background on sequence analysis generally, and to introduce considerations on the implementation of neural networks, some of which are important in applications other than sequence analysis.

1.5.1 Nucleic acid sequence analysis by neural networks

Table 1.1 summarises the performances of neural networks applied to various problems, and shows the authors' comparisons of the neural network approach with other methods. The results and comparisons are discussed more fully below.

1.5.1.1 Translational initiation sites in *E. coli*

The first application of a neural network model to sequence analysis was by Stormo *et al.* (1982a), who used a perceptron algorithm with no hidden layers to predict translational initiation sites in *E. coli*. Their goal was to define nucleotides that may play a role in the selection of initiation codons by the ribosomes of *E. coli*. The training set consisted of 124 known gene beginnings and

Reference	Problem	No. hidden layers	Result	Comparison
Stormo *et al.*, 1982a	*E. coli* mRNA translational initiation sites	0	70%	60% (Stormo *et al.*, 1982b)
Nakata *et al.*, 1985	Splice junctions in human mRNA	0	73-91%	61-74% (Fickett, 1982)
Nakata *et al.*, 1988	*E. coli* promoter recognition	0	67% (perceptron) 75% (+ other info.)	
Lukashin *et al.*, 1989	*E. coli* promoter recognition	1	94-99% 2-6% fp	
Lapedes *et al.*, 1990	Transription and translation of *E. coli* DNA	1	> 90%	85% (Fickett, 1982)
O'Neill, 1991	*E. coli* promoter recognition	1	80% 0.1% fp	70% (O'Neill & Chiafari, 1989)
Demeler *et al.*, 1991	*E. coli* promoter recognition	1	98%	77% (O'Neill & Chiafari, 1989)
Brunak *et al.*, 1991	Human mRNA Donor and Acceptor Sites	1	95% 0.4% fp	95% (Staden, 1984) 0.7% fp
Uberbacher & Mural, 1991	Protein-coding regions in human DNA	2	92% 8% fp	
O'Neill, 1992	*E. coli* promoter recognition	1	80 - 100% 0.5% fp	
Farber *et al.*, 1992	Eukaryotic protein coding regions	1	99%	91% (Farber *et al.*, 1992)
Horton & Kanehisa, 1992	*E. coli* promoter recognition	0	81%	81% (Mulligan *et al.*, 1984)
Synder & Stormo, 1992	Coding regions in DNA	1	92%	91% (Uberbacher & Mural, 1992

Table 1.1 Neural network applications to analyses of nucleic acid sequences. The table summarises the problem tackled, the number of hidden layers, the result (fp = % false positives) and the comparison, if any, made by the authors to other methods.

167 false beginnings, as identified by another method (Stormo *et al.*, 1982b). In a test set of ten genes, the perceptron correctly predicted six of the gene beginnings and incorrectly identified five false beginnings. A rule based approach (Stormo *et al.*, 1982b) only predicted five true gene beginnings and identified twelve false ones.

1.5.1.2 Splice junctions

Nucleotide segments that code for amino acids are called exons; those segments that are not translated are known as introns. Splice junctions are the boundaries between intron and exon segments. The discrimination between introns and exons is vital for determining what proteins are encoded in a nucleotide sequence.

There are two basic approaches to the computer prediction of protein-coding regions in DNA. Firstly, coding function constrains a nucleotide sequence, so coding and non-coding sequences can be distinguished using patterns of codon usage (Shepherd, 1981; Staden and McLachlan, 1982; Gribscov *et al.*, 1984; Tramontano and Macchiato, 1986; Trifonov, 1987), positional mono- and oligonucleotide frequencies and weak 3-periodicity (Fickett, 1982; Staden, 1984a). Secondly, the non-uniformity of nucleotide distribution near start codons and splicing sites can be used (Shapiro and Senepathy, 1987; Ohshima and Gotoh, 1987; Iida, 1987; Gelfand, 1989). The more successful prediction schemes have combined the two approaches.

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

Brunak *et al.* (1991) used neural networks trained by backpropagation to tackle both approaches to the problem of distinguishing coding and non-coding regions, and, based on the combined result, predicted human mRNA donor and acceptor sites in DNA. Multi-layer neural networks trained on short sequence segments were used to identify intron/exon and exon/intron boundaries. Other neural networks were trained on long sequence segments to predict the transition between the coding and non-coding regions. The neural network trained on long segments correctly identified 70% of exons with 2.5% false positives. The neural network trained to recognise exon/intron boundaries correctly detected 94% of unseen boundaries with 0.1% false identification. Intron/exon detection was 87% accurate with 0.2% false identification. The combined method detected 95% of true exon/intron and intron/exon boundaries with false identification at 0.1% and 0.4% respectively. The weight matrix method of Staden (1984b) gave more false positives (0.7%) for the same level of detection of true boundaries. Work in a similar vein has been applied to *E. coli* coding regions (Lapedes *et al.*, 1990; Farber *et al.*, 1992).

The rapid growth of databases necessitates the comparison of neural network results with those of statistical method developed on the same data to demonstrate that the difference in performance is due to the methodology and not to the database size. This is illustrated by a comparison of the prediction of exon/intron boundaries between weight matrices derived from current data and from the data available when the weight matrix method was first developed. The weight matrix method performs better if the matrix is derived from the more recent data (Table 1.2), although it still does not achieve the accuracy of the neural network.

In contrast to the above applications where neural networks have been trained to examine sequence data directly, a neural network was used by Uberbacher and Mural (1991) to

Method	No. false positives at 90% detection of true sites	No. false positives at 95% detection of true sites
Neural network (Brunak *et al.*, 1991)	28	34
Weight matrix (Staden, 1984b)	49	83
Weight matrix from training data	29	44

Table 1.2 A comparison of the prediction of exon/intron boundaries in human DNA by a neural network, a weight matrix method based on data available when the method was first developed, and a weight matrix method derived from more recent data.

examine sequences indirectly, using different properties of the sequence. The network was trained to locate protein-coding regions in human DNA sequences. It consisted of seven input units, two hidden layers of fourteen and five units and an output unit. Input to the network was a vector containing the values of seven sensor algorithms calculated for positions at intervals of ten bases along the sequences of interest. The seven sensor algorithms were: a frame bias matrix - based on the nonrandom frequency with which each of the four bases occupies each of the three positions within codons; Fickett's function (Fickett, 1982); the dinucleotide fractal dimension (Hsu and Hsu, 1990); and four analyses based on the frequency of occurrence of different small segments in the nucleotide sequences. The approach identifies 90% of coding exons of greater than 100 bases with less than one false positive coding exon per five coding exons indicated. Shorter exons are harder to detect - only 47% of exons less than 100 bases were detected. A similar combination of statistical measures have been used in an approach combining dynamic programming and neural networks (Synder and Stormo, 1992).

1.5.1.3 Promoter sites in *E. coli*

A promoter is a segment of DNA sequence that is recognised by an RNA polymerase as a signal to start RNA synthesis. In general, *E. coli* promoters are positioned just before the starting site for transcription. Nakata *et al.* (1988) adapted the discriminant analysis method used to predict splice junctions (Nakata *et al.*, 1985) to predict promoter regions in *E. coli*. The attributes used for discrimination were the accuracy of consensus sequence patterns measured by the perceptron algorithm, the thermal stability map, the base composition, and the Calladine-Dickerson rules for helical twist, roll angle, torsion angle and propeller twist angle (Dickerson, 1983). The perceptron on its own predicted promoter regions in *E. coli* with 67% accuracy. Inclusion of the other information in the prediction algorithm increased the predictive ability to 75%. A perceptron, with no hidden units predicted *E. coli* promoter sites from a binary representation of the sequence with 81% accuracy, using a novel method of reducing the number of input units (Horton and Kanehisa, 1992).

Lukashin *et al.* (1989) and Demeler *et al.* (1991) have both tackled the problem with more complicated neural networks, obtaining accuracies of 94% - 99%, with a 2% - 6% chance of false identification. O'Neill (1991) trained a backpropagating network to recognize 80% *E. coli* promoters of the 17 base spacer class with a false positive rate below 0.1%. This has been extended to the 16 and 18 base classes (O'Neill, 1992). Lukashin *et al.* (1989) used two three-layer neural networks, trained by simulated annealing, to examine the two conservative hexanucleotides that occur 10 base pairs and 35 base pairs upstream from the transcription starting point of the promoter. The outputs from the two networks provided the input to a final unit, whose output corresponded to the classification of the sequence. The training set used by O'Neill (1991) included 5148 58-base sequences drawn from 39 promoters and 4000 random sequences which were 60% in A + T. The group of true sequences was expanded by permuting all possible single base changes in positions other than those known to harbour promoter point mutations. Demeler *et al.* (1991) optimised the predictive ability of a three-layer neural network trained by backpropagation. This was done by varying the encoding scheme (two bit and four bit), the number of hidden units (one to ten), the ratio of promoter sequences to non-promoter sequences (1:1 to 1:20), and the extent of training (training was stopped when the error had reached 1, 0.1, 0.01, 0.001, and 0.0001). The combination of parameters that gave the best results on the test set was selected as the optimised neural network. O'Neill (1989), who used six empirically developed tests to filter out false positives, identified, by a consensus sequence match algorithm, 77% of the fully characterised promoters of Hawley and McClure (1983), but noted that the search produces many false positives, and also noted that the performance was being assessed on sequences used in training the classification method.

1.5.2 Protein sequence analysis by neural networks

The literature on neural networks applied to the analysis of protein sequences is more extensive than that on nucleic acid sequence analysis. Table 1.3 summarises the applications of neural networks to protein sequence analysis. The results are discussed in the following sections.

1.5.2.1 Protein secondary structure prediction

The aim of protein secondary structure prediction is to identify each residue in the protein as forming part of an α-helix,

Application	References
Secondary structure prediction (α-, β- or coil; 3-state)	Qian & Sejnowski, 1988 Holley & Karplus, 1989, 1991 Vieth & Kolinski, 1991 Vieth *et al.*, 1992 Stolorz *et al.*, 1992 Sasagawa & Tajima, 1993 Rost & Sander, 1993a, 1993b
Secondary structure content	Muskal & Kim, 1992 Pancoska *et al.*, 1992
α-helix prediction (2-state)	Bohr *et al.*, 1988 Kneller *et al.*, 1990 Hayward & Collins, 1992
Prediction of three-dimensional structure	Bohr *et al.*,1990 Wilcox *et al.*, 1990 Bohr *et al.*, 1993
Prediction of structural class/fold	Wu *et al.*, 1992 Dubchak *et al.*, 1993 Metfessel *et al.*, 1993
β-turns	McGregor *et al.*, 1989, 1990
Disulphide-bonding state of cysteine	Muskal *et al.*, 1990
Surface exposure of amino-acids	Holbrook *et al.*, 1990
Immunoglobulin domain recognition	Bengio & Pouliot, 1990
Recognition of an ATP-/GTP-binding motif	Hirst & Sternberg, 1991
Signal peptide recognition	Ladunga *et al.*, 1991
Conserved motif recognition	Frishman & Argos, 1992
Water-binding sites	Wade *et al.*, 1992
Signal peptidase cleavage sites	Schneider & Wrede, 1992
Secondary structure of membrane proteins	Fariselli *et al.*, 1993

Table 1.3 Applications of neural networks to protein sequence analysis

a β-strand or coil region, *i.e.*, the conformation of the main chain is predicted, and neither the tertiary fold or the orientation of side chains are considered. The first widely used prediction method was the Chou-Fasman algorithm (Chou and Fasman, 1974). This employed conformational parameters, $P_{i,x}$, for each amino acid i for the conformation x, where

$$P_{i,x} = (n_{i,x}/n_i)/(n_x/N), \qquad \text{[eqn. 1.4]}$$

with $n_{i,x}$ the number of observed amino acids i in conformation x, n_x the number of residues observed in conformation x, n_i the number of amino acids i and N the number of amino acids in the database. The other popular, early method was GOR, from the names of the authors, Garnier, Osguthorp and Robson (Garnier *et al.*, 1978). Here, using information theory, the conformation of a residue at position j is predicted from parameters based on the residue types at positions j-8 to j+8, derived from statistical analysis of the sequences of proteins of known secondary structure. The Chou-Fasman method and GOR are just two of many approaches; there are plenty of reviews of the extensive literature (Kabsch and Sander, 1983; Garnier and Levin, 1991; Sternberg, 1992; Rost *et al.*, 1993).

Qian and Sejnowski (1988) and Holley and Karplus used similar approaches (1989, 1991) to develop neural networks that predicted the secondary structure of an amino acid within a protein (as either α-helix, β-strand, or coil) with an accuracy of 63 - 64%. A three-layer neural network was trained by backpropagation on contiguous segments of protein sequence to assign the secondary structure, as defined by Kabsch and Sander (1983), of the amino acid at the centre of the segment. The accuracy of the neural network was compared with the sequence similarity method of Levin *et al.* (1986) and with rule based (Lim, 1974) and statistical methods (Chou and Fasman, 1978; and Garnier *et al.*, 1978), and was better in all cases. However, more recently, statistical methods have been improved by Gibrat *et al.* (1987) and Ptitsyn and Finkelstein (1989), who both obtained an accuracy of 63%. A machine learning method (King and Sternberg, 1990) had an accuracy of 60%. A Bayesian method, which assumed that the probability of an amino acid occurring in each position of the protein was independent of the other amino acids, performed only marginally worse than a neural network (Stolorz *et al.*, 1992). Recent reviews (von Heijne, 1991; Garnier and Levin, 1991; Garnier, 1991) suggest that 65% appears to be maximum attainable performance of a variety of methods of secondary structure prediction.

Very recently, Rost and Sander (1993) have incorporated evolutionary information in the form of multiple sequence alignments to develop a neural network method that predicts secondary structure with an accuracy of 71%. Three levels of neural networks were used. In the first level, the frequency of occurrence of each of the 20 amino acids at one position in the alignment is computed for each residue in a 13-residue window, to give the input. The target output is the secondary structure class of the central window. The second level was used to take into account the fact that the secondary structure of consecutive residues are correlated. The input to the second-level neural network was the three outputs from the first neural network for 17 consecutive residues. The target output was again the secondary structure of the central residue. The first two stages were performed in nine different ways, with variations in the number of hidden units and training procedures. The outputs from these nine different schema were used as input to a third neural network, which performed a final prediction, in what was called a jury decision. Alignment of homologous sequences have been used previously for the prediction of secondary structure with an accuracy of 66% (Zvelebil *et al.*, 1987), but not within a neural network framework.

1.5.2.2 Specific protein secondary structure prediction

A three-layer neural network trained by backpropagation (Bohr *et al.*, 1988) achieved an accuracy of 73% when predicting the transmembrane α-helices in rhodopsin, which was compared qualitatively with the prediction of Argos *et al.* (1982). This was a two state prediction (α or not α) and the result is not directly comparable to three state predictions (α, β, or coil), as noted by Petersen *et al.* (1990). Kneller *et al.* (1990) suggested that a two state prediction using a simplistic method would be 12% better than a three state prediction, purely because of the change in the number of classes. Thus, to classify residues as either helical or non-helical at 73% is comparable to a three state prediction of 61%, which is obtainable by several methods. Using a similar network, Kneller *et al.* (1990) report an even greater increase in predictive accuracy by pre-classifying proteins as all-α, all-β, and mixed αβ. For all-α a 16% improvement (79% compared to 63%)

over three state predictions was attained. They note, however, that their extra 4% improvement was mostly due to the inclusion of homologous proteins in the testing set. When proteins with a greater than 40% sequence identity were removed, the prediction accuracy fell to 76%.

Two state predictions depend on a prior classification of the protein as either all-α, all-β or a mixture of α-helix and β-strands. It is usually suggested that an experimental technique such as circular dichroism can provide this information. Neural networks, themselves, have actually been used to predict secondary structure content (Muskal and Kim, 1992). Crystallographic data sets have been analysed using neural networks to investigate the relationships between secondary structure fractions (Pancoska *et al.*, 1992).

McGregor *et al.* (1990) improved the prediction of β turns from 21% (Wilmot and Thornton, 1988) to 26%. β turns are a specific class of chain reversals localised over a four-residue sequence (Richardson, 1981). Wilmot and Thornton (1988) distinguished seven types of turn and one miscellaneous class. The neural network of McGregor *et al.* (1990), trained using backpropagation, classified segments of four residues into the categories: type I turns, type II turns, non-specific turns, and non-turns.

1.5.2.3 Tertiary protein structure prediction

Bohr *et al.* (1990, 1993) used a neural network to predict the three-dimensional structure of rat trypsin, based on homology modelling. The training set consisted of the sequences and distance matrices of 13 proteases, including trypsin and (incorrectly) subtilisin. There was a significant degree of homology between rat trypsin and the other trypsins in the training set. The binary distance matrix that was generated for rat trypsin, used bovine pancreatic beta trypsin as the starting configuration, which is 74% homologous with rat trypsin. The output of the neural network was an α-carbon-α-carbon distance matrix. After steepest descent minimisation of the neural network prediction, the predicted structure was within 3 Å rms of the crystal structure. A window size, and hence the width of the diagonal band of the distance matrix was 61. The serine proteases have also been modelled by homology (Greer, 1990). Hubbard and Blundell (1987) superposed, pairwise, seven different serine proteases giving twenty-one comparisons. The rms between any two serine proteases ranged from 0.63 Å to 1.35 Å, with the homology between the sequences varying from 19% to 66%. The neural network approach thus has yet to match modelling by homology.

Wilcox *et al.* (1990) analysed protein tertiary structure using a neural network. The training set of fifteen proteins, of 140 residues or less, intentionally included some homologous proteins. Each amino acid was encoded using hydrophobicity values adapted from Liebmann *et al.* (1985) that were normalised into the range -1 to +1. The input layer was thus 140 units. A hidden layer of 15 to 240 units was used, and the output layer was a window for distance matrices composed of 19,600 (140 by 140) units. The test set consisted of nine proteins each of which was homologous (either in sequence or function) to one or more of the training proteins. Although the network performed well on the training set (a mean squared error below two percent), it was found that generalisation was largely unsuccessful, and this was attributed partly to the small size and the heterogeneity of the training set.

Bengio *et al.* (1990) used a neural network to recognise immunoglobulin (Ig) domains from their amino acid sequences. Ig domains are sets of β sheets, usually bound by disulphide bonds, which exhibit a characteristic tertiary structure. These domains contain well-conserved groups of amino acids in four specific regions. Other residues outside these regions are poorly conserved, so there is low overall homology between Ig domains, even though they are clearly members of the same superfamily. Bengio *et al.* (1990) scanned input sequences with a window of five residues using a neural network trained to recognise the four conserved subregions. The sequence was then classified by a dynamic programming algorithm (Fourney, 1973), which checked if the four subregions had been detected by the neural network and if they occurred in the correct order, using the constraints on the subregions and the distances between them. The accuracy was 98% with 7% false positives, but this was not quantitatively compared with any other methods.

1.5.2.4 Prediction of structural class/fold

Some recent applications of neural networks have attempted to predict the fold a protein sequence will adopt or the family of proteins to which it belongs (Wu *et al.*, 1992). Metfessel *et al.* (1993) found no significant difference between statistical clustering and neural network for predicting structural class from amino acid composition and hydrophobic pattern frequency information. Four different folding classes (4α-helical bundle, parallel $(\alpha/\beta)_8$ barrels, nucleotide binding fold, immunoglobulin fold) were predicted from amino acid composition, using a neural network by Dubchak *et al.* (1993).

Distant familial proteins can be identified by the recognition of conserved motifs using neural networks (Frishman and Argos, 1992). First, a neural network was used to identify the most conserved regions in a given multiple alignment. Then, several neural networks were trained to recognise a number of motifs, using the most conserved regions identified in the first step. These neural networks were then used to scan a sequence database. A neural network that specifically recognises an ATP-/GTP-binding motif (Hirst and Sternberg, 1991) is discussed in Chapter 3.

1.5.2.5 Other protein sequence applications

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

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

Ladunga *et al.* (1991) found the recognition of signal peptides from their sequence to be worse using a neural network than by a statistical method (von Heijne, 1986). A combination of the two methods was better than either of the individual methods. Amino-terminal segments were classified as either signal peptides or cytosolic proteins. The combined method correctly classified 93% of 116 signal peptides (neural network 88%; von Heijne's method 77%) and 97% of 343 cytosolic proteins (neural network 74%; von Heijne's method 84%).

1.6 Implementation and evaluation

Questions of implementation are addressed in this section, and some of the problems involved in developing neural networks are discussed.

1.6.1 Sequence encoding

The input to a neuron is a number (often a binary number - the input neurons are usually on or off), and the input to a neural network is a sequence of numbers. Thus, a protein sequence or nucleic acid sequence has to be encoded from alphabetic characters to a sequence of binary numbers. There are twenty different amino acids residues - each of these is represented by nineteen zeros and a single 1, for example, glycine is coded by 00000010000000000000. Such a sparse representation avoids biochemically unjustified algebraic relationships between the coded forms of the amino acid residues.

The encoding of DNA sequences requires less bits. Stormo *et al.* (1982a) used four bits to encode each base (A -> 1000; C -> 0100; G -> 0010; T -> 0001). Lukashin *et al.* (1989) used a two bit encoding scheme (A -> 00; C -> 11; G -> 10; T -> 01). A two bit encoding scheme requires less input units, but it introduces some ambiguities into the interpretation of results, because each input unit is involved in representing two bases and so the significance of a particular weight is no longer attributable to the importance of one specific base. For the prediction of *E. coli* promoter sites by a neural network four bit encoding gave better results than two bit encoding (O'Neill, 1991; Demeler *et al.*, 1991). It appears to be important that encoding of sequences maximises the orthogonality between the input items.

As well as binary numbers, it is possible to input real numbers. Kneller *et al.* (1990) added to a neural network to predict protein secondary structure an extra input unit which received the magnitude of the helix hydrophobic moment as measured by Eisenberg *et al.* (1982). Uberbacher and Mural (1991) trained a neural network to distinguish coding regions of genes (exons) from noncoding regions (introns) using the results of seven statistical tests on the sequences (each a real number).

1.6.2 The number of input units

In the analysis of protein and nucleic acid sequences by neural networks, the affect of neighbouring residues is an important aspect. For instance, a feature of an amino acid, be it secondary structure or surface accessibility, can be influenced by local neighbours (residues that are close in the sequence, as opposed to residues that are spatially close only because of the folding of the sequence). Neural networks have been used to predict features of an amino acid given its context. The network is presented with the amino acid and some of its neighbours. The length of the segment of sequence presented is known as the window size. The amino acid of interest is usually at the centre of the window. A window size of nine would correspond to four neighbours either side. In protein applications, where each amino acid is represented by a twenty bit binary number, the number of input units is twenty times the window size. If the database is small, the window size and the number of hidden units maybe limited by the sparse encoding.

A window size of 13 was found to be optimal for protein secondary structure prediction (Qian and Sejnowski, 1988), as smaller windows exclude information useful for prediction. There is little useful information outside the window size of 13, and irrelevant weights are deleterious to the performance of the network. Holley and Karplus (1989) report that the error over the training set was least for a window size of 15, but a window size of 17 gave optimal performance on test data. The GOR method (Garnier *et al.*, 1978; Gibrat *et al.*, 1987) employs a window size of 17. For the prediction of distance plots of protein backbones, where a larger window size is required, as non-local interactions are being modelled, Bohr *et al.* (1990) used a window size of 61.

1.6.3 Hidden units

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

For secondary structure prediction, the accuracy of the network was almost independent of the number of hidden units (Qian and Sejnowski, 1988), although the learning rate became slower with less hidden units. Qian and Sejnowski concluded that the common features in the proteins are all first order features, and that the higher order features (the information due to interactions between two or more residues) learned by the network were specific to each individual protein. Holley and Karplus (1989) found two hidden units gave optimal prediction performance, but a network with no hidden units achieved an accuracy that was very close to the optimum. For the prediction of β-turns (McGregor *et al.*, 1989), a network with no hidden units was only marginally inferior to one with hidden units. Holbrook *et al.* (1990) found that the addition of hidden units gave an improvement of two percent, in the prediction of surface accessibility. Bengio and Pouliot (1990) found that a three-layer neural network recognised subregions of the Ig domain with an error two percent less than a neural network without hidden units.

A neural network predicting promoter sites in *E. coli* mRNA (Demeler *et al.*, 1991) did not vary in performance with one to ten hidden units. Brunak *et al.* (1991) predicting donor and acceptor sites in mRNA found a large increase in performance when hidden units were added, with 40 hidden units giving the best performance. This was the case for different window sizes.

The much larger size of DNA databases suggests that applications of neural networks to nucleic acid sequence analysis may be more likely to require hidden units and thus to exploit more fully the power of the approach.

1.6.4 Interpreting results

In a two-layer neural network, the output, and hence the classification of an input pattern, depends on the sum of the products of each weight and the activation of the input unit attached to it. The larger the weight, the greater the influence of its connected input unit on the final classification of the total input pattern. It is possible to identify weights that are important in a classification problem, and they can be correlated to a given input being required at a certain position. This is often done using a graphical representation of the weights, known as a Hinton diagram (Qian and Sejnowski, 1988). The matrix of weights is represented as a rectangle and is shaded according to magnitude and sign. The weights from a neural network with hidden units cannot be interpreted in this way, because their influence on the final output is only indirect through the hidden layer.

A confidence level can be assigned to predictions by a neural network based on the magnitude of the activity of the output unit. A prediction with a high output unit activity is more likely to be correct than one with a lower output unit activity. A significance filter, in which only arbitrarily high or low outputs are used as predictions and intermediate outputs are not used, can improve the predictive accuracy of a network, but the number of predictions made decreases (Holley and Karplus, 1989; Muskal *et al.*, 1990).

1.6.5 Multiple minima

Kneller *et al.* (1990) reported that during the training of their neural network to predict secondary structure of α/β proteins, the network converged to a set of weights that predicted no β structure. This may have been due to the network becoming trapped in a local minimum. Whereas the perceptron convergence theorem (Rosenblatt, 1957) guarantees that a two-layer perceptron will converge if the classes are linearly separable, no such theorem exists for backpropagation. Training a neural network by backpropagation is similar to other minimisation procedures in that local minima can be a problem. To overcome this, Kneller *et al.* (1990) set the initial weights, so that the starting point exaggerated the tendencies of β structure to be detected. Demeler *et al.* (1991) tackled the problem by continuously adjusted the training rate as a function of the derivative of the error function:

$$\text{training rate} = \frac{k}{E_{i-1} - E_i} , \qquad \text{[eqn 1.3]}$$

where k is a constant and was set to 0.000001, E is the error and i is the iteration. If the training process is trapped in a local minimum, the total error is larger than 0, and the derivative of the error function approaches 0 asymptotically. In such a case, increasing the training rate increases the step size allowed in the gradient descent which aids ascent from the local minimum.

Oscillatory behaviour is another problem that can occur during training. Qian and Sejnowski (1988) trained their neural network by randomly sampling the training data, to prevent erratic oscillations in the performance that occurred when the amino acids were sequentially sampled. However, with a slightly different implementation, Holley and Karplus (1989) sampled amino acids sequentially, and did not report any oscillatory behaviour.

1.6.6 Data presentation

Sometimes the data rather than the network behaviour can complicate the training procedure. Due to a lack of data, Bengio and Pouliot (1990) generated pseudo-sequences by substituting residues in the variable positions of the domain with variable residues found elsewhere in the sequence, to increase the training set. Demeler *et al.* (1991) optimised the performance of their network to predict *E. coli* promoter sites by varying the number of randomly generated counter-examples presented to the network. McGregor *et al.* (1989) also enhanced performance by training the network with more examples from classes that the network was having difficulty predicting. A lack of data can exclude the use of more involved architectures.

1.6.7 Memorisation

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

1.6.8 Testing protocols

Problems in testing procedures are perhaps less apparent. The presence of homologous proteins in data is a major problem that is not always addressed. Training and testing on homologous proteins improves the performance of a neural network by about 10% (Qian and Sejnowski, 1988). It is still not as successful as aligning the two proteins and assigning to the amino acids of the test protein the corresponding secondary structures in the training protein. Bohr *et al.* (1988) report that if the network is trained on one member of a pair of homologous proteins and it is then tested on the other member of the pair, one can obtain a measure of the degree to which the primary-to-secondary mapping in the first protein resembles the corresponding mapping in the second protein.

1.7 Scope of Thesis

Neural networks are being applied to a growing number of different fields. Applications to QSAR and biomolecular modelling have been reviewed in this chapter. There is little doubt that neural networks can often provide good empirical models. In this thesis, the predictive performance of neural networks is carefully benchmarked against more traditional statistical techniques, and the usefulness of the approach, in terms of affording insight into the particular modelling applications, is examined.

The algorithms used in this thesis are presented in Chapter 2. In Chapter 3, a perceptron-type neural network trained to recognise an ATP-/GTP-binding motif is compared with a consensus sequence method. In Chapters 4 and 5, the QSARs of the inhibition of DHFR by pyrimidines and triazines, respectively, are derived by several methods, including neural networks and a contemporary machine learning method. The conclusions from these studies are given in Chapter 6.

Chapter 2

Theory of neural networks

2.1 Synopsis

This chapter presents the algorithms used in the thesis: the elementary perceptron algorithm, the backpropagation of errors for neural networks with hidden layers, and the Gear algorithm for solving ordinary stiff differential equations. An expository example illustrates the application of backpropagation.

2.2 The elementary perceptron algorithm

Figure 2.1 depicts an elementary perceptron. There are n input units and, for simplicity, one output unit, although, in general, there could be any number of output units. The lines connecting units represent the neural network weights, w_i, where i takes the values 1 to n. Training the neural network requires a set of input data and their corresponding classifications, *i.e.*, the value to be shown at the output unit. Each of the examples in the training set, thus, comprises an n-component input vector and a desired value for the output.

The training procedure begins with the initialisation of the weights. The value of each weight is set to be a small random number, usually between -1 and +1. The first training example is presented to the neural network, so the activations of the input layer, In_i (i = 1 to n), are set to the values of the n-component input vector. The input neurons do not perform any further processing. The activations of the input layer are then propagated forward to the output unit. The activation of the output unit, A_{out}, is calculated by:

$$A_{out} = f(X_{out}), \qquad \text{[eqn. 2.1]}$$

where,

$$X_{out} = \sum_{i=1}^{n} w_i \, In_i + B \qquad \text{[eqn. 2.2]}$$

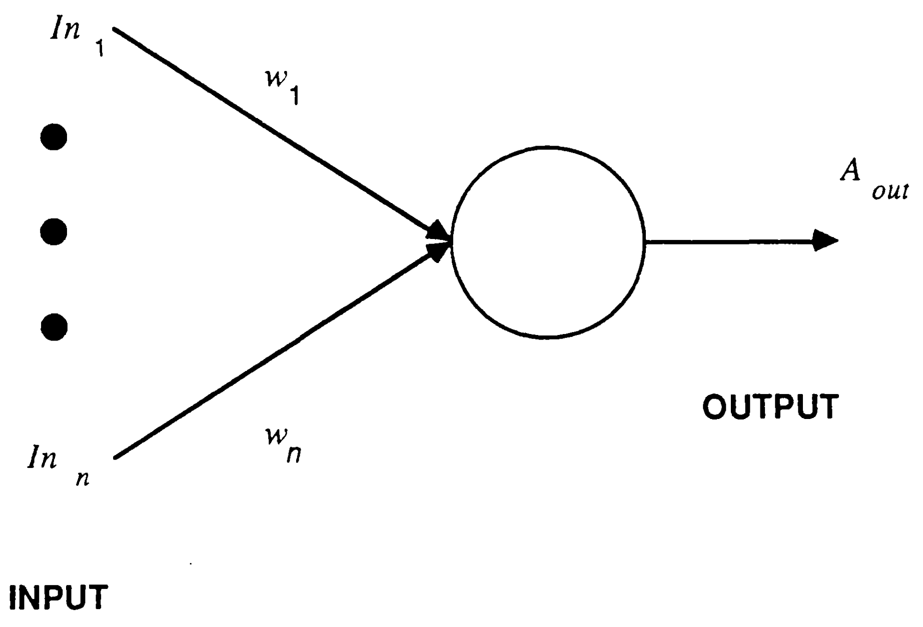


Figure 2.1 An elementary perceptron

where B is a constant, called the bias, and

$$f(X_{out}) = 1, \text{ if } X_{out} \geq 0.0, \qquad \text{[eqn. 2.3]}$$

$$f(X_{out}) = 0, \text{ if } X_{out} < 0.0 \qquad \text{[eqn. 2.4]}$$

The difference, δ, between the actual output, A_{out}, and the desired output, D_{out}, often called the error, is computed:

$$\delta = A_{out} - D_{out} \qquad \text{[eqn. 2.5]}$$

The value of the weights are then altered using the delta rule, which gives the change in the value, Δw_i

$$\Delta w_i = \eta \; \delta \; In_i \qquad \text{[eqn. 2.6]}$$

where η, the learning rate, is an empirically chosen positive constant, which scales the magnitude of the weight change. The value of the bias is changed as if it were one of the weights.

This procedure is repeated for each example in the training set. The input units are given the values of the next training example; these are propagated via the updated weights to the output unit; the error is calculated; the weights are altered accordingly, using the delta rule [eqn. 2.6]; the input units are given the values of the next training example; and so on, until an arbitrary convergence criterion is satisfied. Each example in the training set is presented many times. The perceptron convergence theorem by Rosenblatt (1962) proves that this procedure will find a solution in finite time for any linearly separable problem.

2.3 The backpropagation of errors algorithm

For simplicity, a three layer neural network with one output unit will be considered. In general, there may be any number of hidden layers and any number of output units. The difference between this algorithm and the elementary perceptron algorithm arises from the presence of a layer of units between the input and output layers, called the hidden layer. The backpropagation of errors algorithm is required, because the perceptron convergence theorem can not be extended from two to three layers.

Let the number of input units be n, and the number of hidden units be m; only one output unit is considered. The activations of the input units are In_i, where $i = 1$ to n; the activations of the hidden units are Hid_j, where $j = 1$ to m. The activation of the output unit is A_{out}. The weights connecting the input units to the hidden units are w_{ij}, where w_{ij} is the weight connecting the i^{th} input unit to the j^{th} hidden unit. The weights connecting the hidden units to the output units are w_{j1}. For more involved architectures, the weights are referenced by three indices: one index for the layer, one for number of units in that layer, and one for the number of units in the preceding layer.

The training procedure (see Figure 2.2) is similar in outline to the elementary perceptron algorithm. The first training

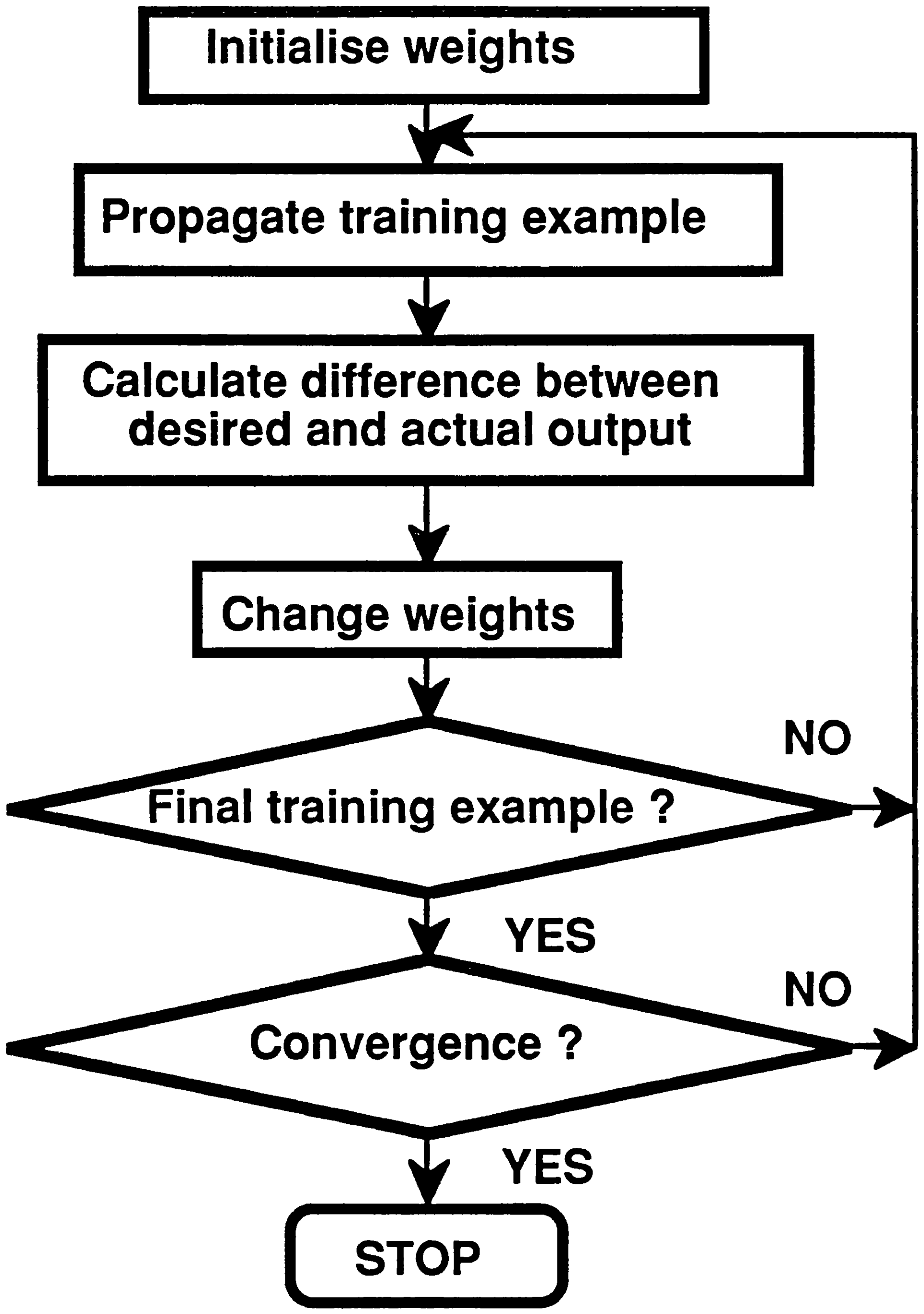


Figure 2.2 Outline of the learning algorithms

example is presented to the neural network, by setting the values of the input units to those of the training example. These are propagated forward to the hidden layer: the activations of the hidden layer are given by:

$$Hid_j = 1/(1 + \exp[-H_{out}]), \qquad \text{[eqn. 2.7]}$$

where,

$$H_{out} = \sum_{i=1}^{n} w_{ij}\, In_i + B \qquad \text{[eqn. 2.8]}$$

where B is a constant, called the bias. Here, the activation function is the logistic function, which is a continuous and nonlinear function. The hidden layer activations are propagated forward to the output unit:

$$A_{out} = 1/(1 + \exp[-X_{out}]), \qquad \text{[eqn. 2.9]}$$

where

$$X_{out} = \sum_{j=1}^{m} w_{j1}\, Hid_j + B' \qquad \text{[eqn. 2.10]}$$

where B' is the bias associated with the hidden layer units.

The error, δ, the difference between the actual output, A_{out}, and the desired output, D_{out}, is calculated by:

$$\delta = A_{out}(1 - A_{out})(D_{out} - A_{out}) \qquad \text{[eqn. 2.11]}$$

The error at the output unit(s) is required to calculate the changes to be made to the weights connecting the hidden layer to the output layer. The key step in the backpropagation of errors algorithm is the calculation of the errors at the hidden layer units, which permits the calculation of the changes to the weights connecting the input layer to the hidden layer. The errors at the hidden layer units, $\delta_{hid,j}$, are calculated by:

$$\delta_{hid,j} = Hid_j(1 - Hid_j) \sum_{j=1}^{m} w_{j1}\ \delta \qquad \text{[eqn. 2.12]}$$

The algorithm takes its name from this step, where the errors are backpropagated from the output layer to the hidden layer. The changes to the weights connecting the hidden layer to the output layer are calculated by:

$$\Delta w_{j1} = \eta\ Hid_j\ \delta \qquad \text{[eqn. 2.13]}$$

The bias, B', is changed as if it were one of these weights. The changes to the weights connecting the input layer to the hidden layer are:

$$\Delta w_{ij} = \eta\ In_i\ \delta_{hid,j} \qquad \text{[eqn. 2.14]}$$

The bias, B, is changed as if it were one of these weights. Although the learning rates are shown to be the same here, each set of weights may have a different learning rate.

As for the perceptron algorithm, the training procedure is performed on each example of the training set, and is repeated until a convergence criterion is satisfied. The algorithm can be shown to be minimising the sum of the squares of the differences between the desired outputs and actual outputs (Rumelhart *et al.*, 1986b). The minimisation procedure is a steepest gradient descent. FORTRAN code for a general backpropagation neural network, *i.e.*, any number of layers, and any number of units in these layers, was written and is given in Appendix 2.

2.3.1 Considerations for implementation

Rumelhart *et al.* (1986a) presented an improvement to the basic algorithm, which accelerates the convergence. The modified algorithm for changing the weights is:

$$\Delta w(n+1) = \eta \ \delta Act + \alpha \Delta w(n), \qquad \text{[eqn. 2.15]}$$

where δ is the error between the computed and desired values of the unit, Act is the appropriate activation, n indexes the presentation number, η is the learning rate, and α is a constant that determines the effect of past weight changes on the current direction of movement in weight space. This provides a momentum in weight space that effectively filters out high-frequency variations of the error surface in the weight space.

Sejnowski and Rosenberg (1987) suggest that to reduce the average error for all the input patterns, the gradients (the errors) should be averaged over all the training patterns before updating the weights, but that in practice, it is sufficient to average over several inputs before updating. Rumelhart *et al.* (1986a) indicate that the weights can be changed after every input-output case, and this has the advantage that no memory is required for the errors. However, they accumulated the gradients over all the input-output cases before updating the weights.

2.4 An expository problem

Consider a series of eight drugs with activities of 1 or 0. Each drug has a benzyl ring that can be substituted at three positions. (see Table 2.1). There is no simple linear or polynomial equation that fits the data. Therefore, classical methods would not be able to predict correctly these data. The simplest equation that fits the data has a cross-product term:

$$\text{Activity} = A_2 + A_3 - (2 \times A_2 \times A_3), \qquad \text{[eqn. 2.16]}$$

where A_2 indicates a substituent at position 2 and A_3 indicates a substituent at position 3.

It could be argued that these data are more naturally considered as a discrimination problem, where instead of predicting a real number, one only discriminates between drugs with high activity (1) and low activity (0). However, again there are difficulties for linear methods, as no linear discriminant can

Drug	Substituent at position 2	Substituent at position 3	Substituent at position 4	Activity
drug1	1	1	1	0
drug2	1	1	0	0
drug3	1	0	1	1
drug4	0	0	1	0
drug5	0	0	0	0
drug6	0	1	0	1
drug7	0	1	1	1
drug8	1	0	0	1

Table 2.1 A simple drug design problem with eight drugs. Each drug can have a substituent at 3 positions (1 = present; 0 = absence). Each drug can have an activity of 0 or 1.

separate the drugs with high activity from those of low activity. If in Table 2.1, 1 and 0 are interpreted to mean true and false respectively, the function for deciding activity can be interpreted as being exclusive OR (XOR). A drug will have high activity if it has either a substituent at position 2 and not at position 3, or a substituent at position 3 and not at position 2. Such a situation could arise in practice if a substituent at position 2 could fit into the active site, as could a substituent at position 3, but when they are both present they interfere with each other and neither can enter the active site.

A neural network trained using backpropagation is shown in Figure 2.3. There are three neurons in the input layer, In_1, In_2, and In_3, two neurons in the hidden layer, Hid_1 and Hid_2, and one output neuron, Out_1. The numbers in each neuron are biases, and the numbers next to the arrows are weights. In the first level of weights there is near symmetry, and in the second level, the numbers are almost complements of each other. Taking drug2 as an example to show how the neural network predicts the activity of a drug, the input to In_1 is 0.00, to In_2 is 1.00, and In_3 is 0.00. These numbers are propagated to Hid_1 and Hid_2:

$$\begin{aligned} Hid_1 &= -2.89 + (6.13 \times In_1) + (6.12 \times In_2) + (0.00 \times In_3) \\ &= 3.23 \end{aligned} \qquad \text{[eqn. 2.17]}$$

$$\begin{aligned} Hid_2 &= -6.09 + (4.09 \times In_1) + (4.09 \times In_2) + (0.00 \times In_3) \\ &= -2.00 \end{aligned} \qquad \text{[eqn. 2.18]}$$

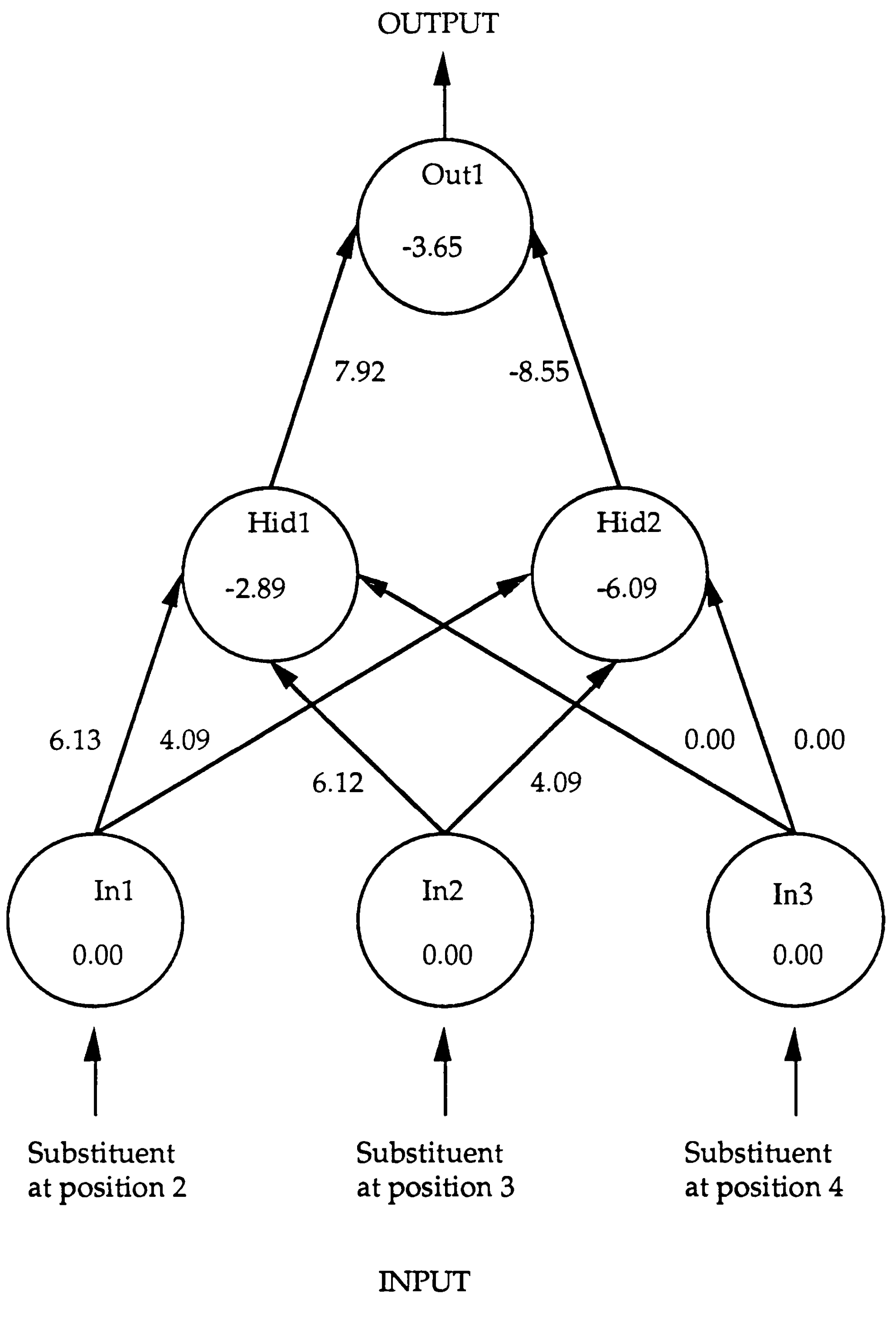


Figure 2.3 A neural network solution to a simple drug design problem

The logistic function is then taken of these values, $f(Hid_1)$ = 0.962, $f(Hid_2)$ = 0.119. These numbers are then propagated through to the output layer:

$$Out_1 = -3.65 + (7.92 \times 0.962) + (-8.55 \times 0.119)$$
$$= 2.95. \qquad \text{[eqn. 2.19]}$$

Finally the logistic function is taken of this value, $f(Out_1)$ = 0.95. This is the outputted predicted activity of drug2.

2.5 The Gear algorithm

Owens and Filkin (1989) showed that the training time of a backpropagating neural network could, for some applications, be considerable reduced by calculating of the weight changes by solving a set of stiff coupled differential equations, using the Gear algorithm (Gear, 1971). This algorithm has been implemented as an option and used in Chapters 4 and 5. The FORTRAN code is not presented in this thesis as it was taken directly from the original book by Gear, but the casting of the learning algorithm as a system of stiff coupled ordinary differential equations is discussed here.

2.5.1 Fixed step steepest gradient descent

The backpropagating algorithm is based on steepest gradient descent. The steepest gradient descent method is illustrated here for a neural network with no hidden units. The generalisation of this method to neural networks with hidden layers is given by Rumelhart *et al.* (1986a, 1986b) and is not presented. The objective is to minimise E, the sum of the squares of the errors:

$$E = (1/2)\sum_{j=1}^{T} (D_{out} - A_{out})^2 \qquad \text{[eqn. 2.20]}$$

where the sum is over T, the number of examples in the training set. The change to the weights is calculated from the gradient of E with respect to the weights, w (the subscripts are omitted for clarity):

$$\Delta w = -\partial E/\partial w \qquad \text{[eqn. 2.21]}$$

Applying the chain rule of calculus:

$$\Delta w = -(\partial E/\partial A_{out})(\partial A_{out}/\partial w) \qquad \text{[eqn. 2.22]}$$

From [eqn. 2.20], for each example in the training set:

$$\partial E/\partial A_{out} = -(D_{out} - A_{out}) \qquad \text{[eqn. 2.23]}$$

Now,

$$A_{out} = [1 + \exp(- \sum_{j=1}^{m} w_j\, In_j + B)]^{-1}, \qquad \text{[eqn. 2.24]}$$

where m is the number of input units. Differentiating using the product rule gives:

$$\partial A_{out}/\partial w = -[\ 1 + \exp(-\sum_{j=1}^{m} w_j\, In_j + B)]^{-2} \text{ x}$$
$$-\exp(-\sum_{j=1}^{m} w_j In_j + B) \text{ x } In_j \qquad \text{[eqn 2.25]}$$

Rearranging, using [eqn. 2.24] gives:

$$\partial A_{out}/\partial w = A_{out}(1 - A_{out})In_j \qquad \text{[eqn. 2.26]}$$

Thus, from [eqn. 2.22], [eqn. 2.23] and [eqn. 2.26]:

$$\Delta w = (D_{out} - A_{out})A_{out}(1 - A_{out})In_j, \qquad \text{[eqn. 2.27]}$$

which is the result given by [eqn. 2.11] and [eqn. 2.13], not including the learning rate.

The minimisation can be viewed as a search for the lowest point in the nearest valley of an N-dimensional space, where N is the number of adjustable weights and biases and the surface is the sum of the squares of the prediction errors. The backpropagation of errors algorithm searches this space by taking a fixed step in the direction of the steepest downhill gradient. The Gear algorithm improves the efficiency of the search by using a variable step size, and maintains the stability of the solution.

2.5.2 Stiff coupled ordinary differential equations

The solution of a system of ordinary differential equations involves integration. Generally, one determines the values of the dependent variables for any value of the independent variable, given the values at a specified value of the independent variable. Differential equations only involving functions of a single variable are termed ordinary differential equations. A system of differential equations is coupled if the equations cannot be solved independently. A system of differential equations is said to be stiff if there are time constants that differ greatly in magnitude, *i.e.*, there are some variables that change much more rapidly than other variables. In backpropagation, this corresponds to having some weights that change very slowly compared to other weights. It can be inefficient to solve a system of stiff differential equations, using a fixed step method, because the rapidly changing component will require a small step size for a stable solution (a solution that is not sensitive to errors in the initial values), and this small step size will be inefficient for the slowly changing component.

The backpropagation of errors algorithm is recast as a system of stiff coupled ordinary differential equations:

$$\mathrm{d}w/\mathrm{d}t = \Delta W, \qquad \text{[eqn. 2.28]}$$

where ΔW represents the weight changes calculated using the backpropagation of errors algorithm [eqn. 2.27] and is nonlinear function of all the weights (as is clear from [eqn. 2.24]). There is one ordinary differential equation for each adjustable weight and bias. Each unit of time, t, corresponds to one presentation of the training set. The initial conditions are small random weights. The system of equations is integrated numerically using the Gear algorithm (Gear, 1971).

2.5.3 Outline of the Gear algorithm

The general problem under consideration may be cast as:

$$dy/dt = f(y), \qquad \text{[eqn. 2.29]}$$

where y is a vector with components y_1, y_2, y_3, *etc.*, and f is a function. Explicitly, the ordinary differential equations are: $dy_1/dt = f_1(y)$, $dy_2/dt = f_2(y)$, $dy_3/dt = f_3(y)$, and so on. The form of [eqn. 2.29] is the same as that of [eqn. 2.28]. The solution to the problem involves numerical integration of the equation, to give the values of y at any time t, given the values of y at $t = 0$.

One value methods provide an approximation of $y(t)$ at $t = t_n$, given $y(t)$ at $t = t_{n-1}$, where n indexes the time steps. The Gear algorithm is a multivalue method, because it uses the values of the dependent variable, y, and its derivative at several values of t, the independent variable.

A multivalue method consists of two processes. In the prediction process, an approximation to y_n is computed by linear extrapolation from y_{n-1} (which is taken to include the first derivatives). This approximation, $y_{n,(0)}$, is given by:

$$y_{n,(0)} = By_{n-1}, \qquad \text{[eqn. 2.30]}$$

where B is any suitable matrix of constants (discussed in detail by Gear, 1971). The prediction process does not make use of the differential equations in anyway, so the correction process corrects the approximate values if they do not satisfy the differential equation at $t = t_n$. The amount by which $y_{n,(0)}$ does not satisfy the differential equations is $G(y_{n,(0)})$. A vector multiple of this scalar is added to $y_{n,(0)}$ to correct it:

$$y_{n,(1)} = y_{n,(0)} + cG(y_{n,(0)}) \qquad \text{[eqn. 2.31]}$$

This can be repeated by:

$$y_{n,(m+1)} = y_{n,(m)} + cG(y_{n,(m)}) \qquad m = 1, 2, \ldots \qquad \text{[eqn. 2.32]}$$

The value used for y_n is then $y_{n,(M)}$, where M, the number of corrector iterations, is either fixed, or large enough to give acceptable convergence.

Important properties of general multivalue methods include mechanisms for changing the step size and estimation of errors. The basic step control mechanism is to execute one step and to test if the estimate of the truncation error (the difference between the exact solution and the numerical solution) is less than an arbitrary limit. If it is, then the step is accepted; otherwise, it is rejected. The step size to use for the next step or to repeat the rejected step can then be estimated (details in Gear, 1971).

2.5.4 Trials using the Gear algorithm

The utility of the Gear algorithm is illustrated in Figure 2.4, where the decrease in the training time required for the expository problem discussed in section 2.4 is clearly evident. Similar benefits were found for the QSAR studies in Chapters 4 and 5 (results not shown).

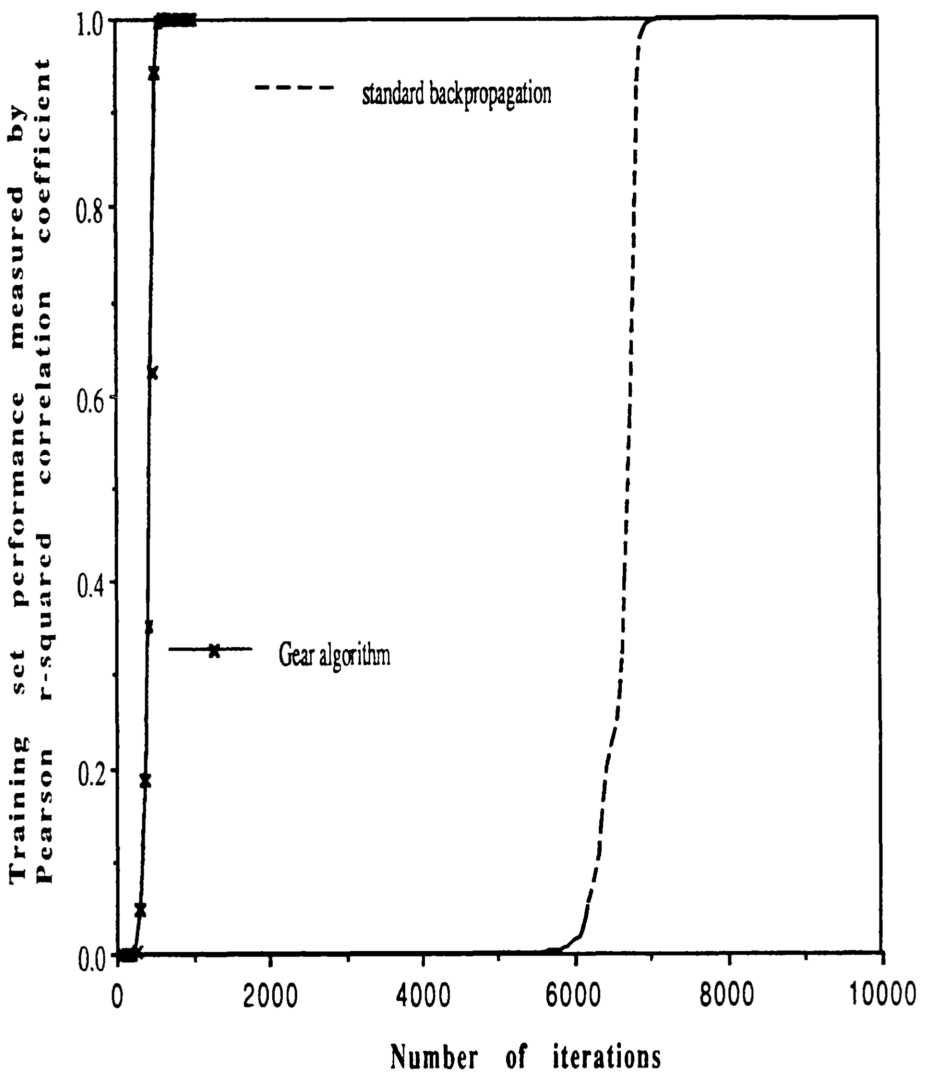


Figure 2.4 Example of speed up in training using the Gear on the expository drug design problem discussed in section 2.4

Chapter 3

Prediction of an ATP/GTP-binding motif:
a comparison of a perceptron type neural network
and a consensus sequence method

3.1 Synopsis

Neural networks have been applied to a number of protein structure problems. In some applications, their success has not been substantiated by a comparison with the performance of a suitable alternative statistical method on the same data. In this chapter, a two-layer feed-forward neural network has been trained to recognise an ATP/GTP-binding local sequence motif. The neural network correctly classified 78% of the 349 sequences used. This was much better than a simple motif-searching program. A more sophisticated statistical method was developed, however, which performed marginally better (84% correct classification) than the neural network. The neural network and the statistical method performed similarly on sequences of varying degrees of homology. These results do not imply that neural networks, especially those with hidden layers, are not useful tools, but they do suggest that two-layer networks in particular should be carefully tested against other statistical methods.

3.2 Introduction

Proteins often contain a local sequence motif - a well-conserved group of amino acids in a specific region. Other residues outside of this region are usually poorly conserved, so there is low overall homology with other proteins containing the same motif. Many of these motifs are involved in catalysis and ligand-binding (Bairoch, 1990; Hodgman, 1989; Sternberg, 1991a). Their presence in a protein can indicate putative function and so it is useful to scan amino acid sequences databases to identify proteins motifs of interest. The ATP/GTP-binding site motif A (Walker *et al.*, 1982) was chosen for the development of a neural network to recognise motifs, because of the relatively large number of known examples. Many ATP- and GTP-binding proteins have a phosphate-binding loop (P-loop) - a glycine-rich sequence followed by a conserved lysine and a serine or a threonine (Walker *et al.*, 1982). The P-loop is preceded by a β-strand and followed by an α-helix (Figure 3.1). Some proteins with this motif bind ATP and others bind GTP; other proteins bind ATP or GTP, but do not have this motif, for example, the glycolytic kinases, E1/E2-type ATPases, actin, tubulin and aminoacyl-tRNA synthetases (Saraste *et al.*, 1990). The structures of several ATP- and GTP-binding proteins containing P-loops have been solved and are discussed in a recent review (Saraste *et al.*, 1990).

The development of the learning by back-propagation of errors algorithm (Rumelhart *et al.*, 1986a) has lead to a resurgence of interest in neural networks. The first biochemical

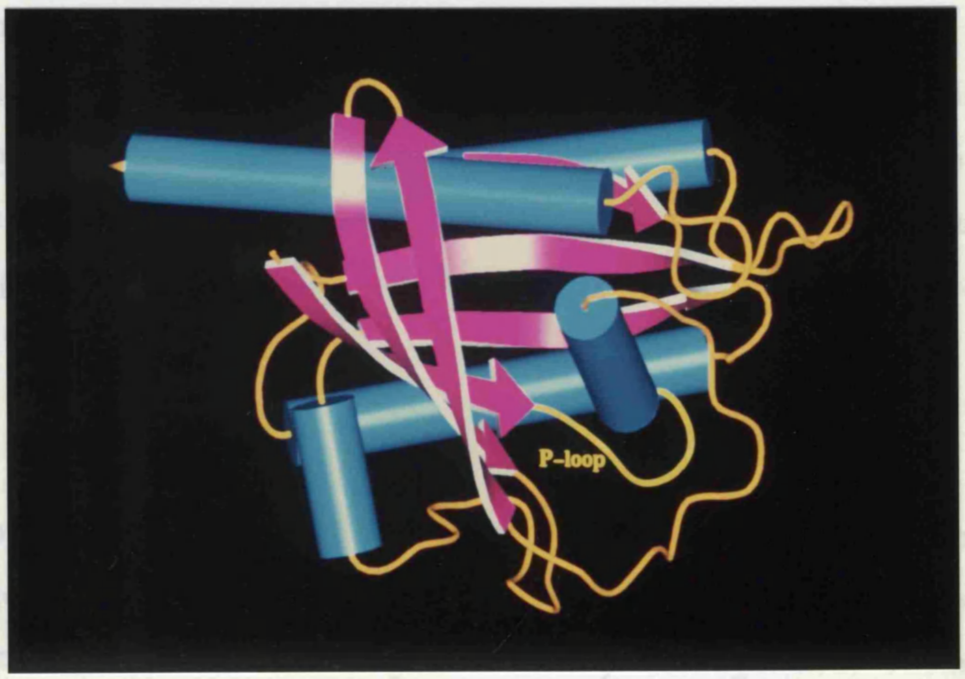


Figure 3.1 P-loop in the human p21 ras protein (Tong *et al.*, 1990), displayed using the graphics program PREPI by Dr. Suhail Islam.

application of neural networks was the prediction of protein α- and β-secondary structures (Bohr *et al.*, 1990; Bohr *et al.*, 1988; Holley and Karplus, 1989; Kneller *et al.*, 1990; Qian and Sejnowski, 1988). Other applications include the prediction of β-turns (McGregor *et al.*, 1989) and the recognition of immunoglobulin domains (Bengio and Pouliot, 1990). The domain recognition employed a neural network with hidden units, which performed better than a simpler two-layer network. Its performance, however, was not contrasted with that of a statistical technique on the same data. Two-layer feed-forward neural networks have been trained, using back-propagation, to find the disulphide-bonding state of cystines, and the surface exposure of amino acids (Holbrook *et al.*, 1990; Muskal *et al.*, 1990).

It is relatively easy to generate a neural network that works. It is, apparently, less easy to assess its performance rigorously. Well defined testing procedures should be followed, with the test and training sets clearly distinguished. Further, a statistical method of similar sophistication to the neural network should be tested on the same data. In some applications, homologous proteins in the training and testing sets can give misleading results. A prime motivation for the study presented here was to determine if the neural network would perform better than a comparable statistical method.

In this chapter, a two-layer feed-forward a neural network was used to recognise an ATP/GTP-binding motif in proteins. Despite using a motif that occurs in many proteins, there were not sufficient data to investigate networks with hidden layers. The performance of the neural network was compared with that of a statistical method based on the motif searching program of the AMPS package (Barton and Sternberg, 1990). This type of statistical measure is widely used in profile/pattern recognition programs (Bashford *et al.*, 1987; Gribskov *et al.*, 1987; Taylor, 1986).

3.3 **Method**

A feed-forward network with no hidden units was used. Between each unit in the input layer and each unit in the output layer is a weighted connection. Each output has an additional weight called the bias. Each output unit takes a weighted sum of its inputs:

$$X_O = \sum_i (W_{o,i} \,.\, I_i) + B_o \qquad \text{[eqn. 3.1]}$$

where the $W_{o,i}$ is the weight to output unit o from input unit i, I_i is the value at the input unit i (either 0.0 or 1.0) and B_o is the output bias. The activities, A_o, of the output units are a function of an arbitrary, constant activation threshold T:

$$X_o \geq T, A_o = 1.0 \qquad \text{[eqn. 3.2]}$$

$$X_o < T, A_o = 0.0 \qquad \text{[eqn. 3.3]}$$

The threshold used in this work was $T = 0.0$. The weights were adjusted by back-propagation, so as to minimise the difference between the actual output A_o and the desired output D_o. The weight change is defined by

$$\Delta W_{o,i} = -\eta \, . \, v_o \, . \, X_o \quad , \qquad \text{[eqn. 3.4]}$$

where

$$v_o = (D_o - A_o) \qquad \text{[eqn. 3.5]}$$

and η is the constant learning rate, chosen in this work to be 0.05. The training of the neural network is such as to minimise the total error,

$$E = \sum (D_o - A_o)^2. \qquad \text{[eqn. 3.6]}$$

The network is trained until this total error converges. If it converges to zero, the network perfectly classifies the training set. In this work, the network could not be trained to perfection.

ATP/GTP-binding proteins were extracted from the SWISSPROT release 14 database by PROMOT (Sternberg, 1991b), a computer program that searches databases for sequences exhibiting a chosen motif. The PROSITE (Bairoch, 1990) ATP/GTP-binding site motif A used was [AG]-X(4)-G-K-[ST] (Walker *et al.*, 1982), where X is any residue and X(*n*) is *n* such residues, [AG] means either A or G is acceptable and [ST] means either S or T is acceptable. This gave 798 matches. These were classified into

ATP/GTP-binding and not ATP/GTP-binding proteins, by reference to known biochemical properties, with any ambiguous sequences being discarded. After identical sequences were removed, the resulting classes contained 197 and 152 examples respectively. These proteins are listed in Tables 3.1a and 3.1b.

PROMOT (Sternberg, 1991b) generated many false positives and compared poorly with both the neural network and the more sophisticated statistical method employed in this chapter. The neural network was developed as a more sensitive method, that would distil the coarser classification of PROMOT (Sternberg, 1991b), perhaps by detecting weaker patterns in the ATP/GTP-binding sequences, such as the β-strand and the α-helix that occur immediately before and after the P-loop. A major appeal of the neural network approach is that information about non-binding sequences is readily included.

The network was trained on sequences of 17 residues, *i.e.*, a window size of 17. This was the maximum window size possible, while the number of sequences remained greater than the number of weights. Table 3.2 shows the variation of the performance of both the statistical method and the neural network with the window size. These 17 residues did not include the four specified in the ATP/GTP-binding motif used in PROMOT (Sternberg, 1991b), because they were common to all the sequences, binding or non-binding, and therefore had no

194K_TRVSY	899	POTENTIAL 194 KD PROTEIN (PUTATIVE REPLICASE) (CONTAINS: 134 KD PROTEIN).
ARF1_YEAST	19	ADP-RIBOSYLATION FACTOR 1.
ARF2_BOVIN	19	ADP-RIBOSYLATION FACTOR 2.
AROL_ECOLI	4	SHIKIMATE KINASE II (EC 2.7.1.71) (SKII).
AROL_ERWCH	4	SHIKIMATE KINASE (EC 2.7.1.71).
ARSA_ECOLI	10	ARSENICAL PUMP-DRIVING ATPASE.
ARSA_ECOLI	329	as above
ATP0_MAIZE	166	ATP SYNTHASE ALPHA CHAIN, MITOCHONDRIAL (EC 3.6.1.34).
ATPA_ANASP	166	ATP SYNTHASE ALPHA CHAIN (EC 3.6.1.34).
ATPA_BOVIN	207	ATP SYNTHASE ALPHA CHAIN, LIVER ISOFORM, MITOCHONDRIAL PRECURSOR (EC 3.6.1.34) (FRAGMENT).
ATPA_ECOLI	164	ATP SYNTHASE ALPHA CHAIN (EC 3.6.1.34).
ATPA_MAIZE	165	ATP SYNTHASE ALPHA CHAIN (EC 3.6.1.34).
ATPA_MARPO	165	ATP SYNTHASE ALPHA CHAIN (EC 3.6.1.34).
ATPA_RHOBL	164	ATP SYNTHASE ALPHA CHAIN (EC 3.6.1.34).
ATPA_RHORU	164	ATP SYNTHASE ALPHA CHAIN (EC 3.6.1.34).
ATPA_SULAC	228	MEMBRANE-ASSOCIATED ATPASE ALPHA CHAIN (EC 3.6.1.34) (SUL-ATPASE ALPHA).
ATPA_THEP3	164	ATP SYNTHASE ALPHA CHAIN (EC 3.6.1.34).
ATPA_VIBAL	164	ATP SYNTHASE ALPHA CHAIN (EC 3.6.1.34).
ATPB_ANASP	157	ATP SYNTHASE BETA CHAIN (EC 3.6.1.34).
ATPB_BACFR	152	ATP SYNTHASE BETA CHAIN (EC 3.6.1.34).
ATPB_BACME	153	ATP SYNTHASE BETA CHAIN (EC 3.6.1.34).
ATPB_ECOLI	145	ATP SYNTHASE BETA CHAIN (EC 3.6.1.34).
ATPB_RHORU	147	ATP SYNTHASE BETA CHAIN (EC 3.6.1.34).
ATPB_THEP3	153	ATP SYNTHASE BETA CHAIN (EC 3.6.1.34).
ATPB_VIBAL	144	ATP SYNTHASE BETA CHAIN (EC 3.6.1.34).
ATPB_YEAST	185	ATP SYNTHASE BETA CHAIN, MITOCHONDRIAL PRECURSOR (EC 3.6.1.34).
CARB_ECOLI	860	CARBAMOYL-PHOSPHATE SYNTHASE LARGE CHAIN (EC 6.3.5.5) (CARBAMOYL- PHOSPHATE SYNTHETASE AMMONIA CHAIN).
CDR1_SCHPO	5	MITOSIS INDUCER PROTEIN KINASE CDR1 (EC 2.7.1.-) (NIM1+) .
CFTR_HUMAN	453	CYSTIC FIBROSIS TRANSMEMBRANE CONDUCTANCE REGULATOR (CFTR).
CFTR_HUMAN	1239	as above
CIN1_RAT	903	SODIUM CHANNEL PROTEIN I, BRAIN.
CIN2_RAT	894	SODIUM CHANNEL PROTEIN II, BRAIN.
CIN3_RAT	846	SODIUM CHANNEL PROTEIN III, BRAIN.
CINS_RAT	707	SODIUM CHANNEL PROTEIN, SKELETAL MUSCLE ALPHA-SUBUNIT (MU-1).
CLPA_ECOLI	209	ATP-DEPENDENT CLP PROTEASE ATP-BINDING SUBUNIT (EC 3.4.21.-) (CASEINOLYTIC PROTEASE) (PROTEASE TI).
CLPA_ECOLI	490	as above
CLPA_RHOBL	239	PROBABLE ATP-DEPENDENT CLP PROTEASE ATP-BINDING SUBUNIT (EC 3.4.21.-).
CLPA_RHOBL	520	as above
CLPB_ECOLI	201	ATP-DEPENDENT CLP PROTEASE ATP-BINDING SUBUNIT-LIKE PROTEIN (HEAT SHOCK PROTEIN F84.1).
CLPB_ECOLI	511	as above
CLPB_ECOLI	600	as above
CN37_BOVIN	32	2',3'-CYCLIC NUCLEOTIDE 3'-PHOSPHODIESTERASE (EC 3.1.4.37) (CNP).
CN37_HUMAN	32	2',3'-CYCLIC NUCLEOTIDE 3'-PHOSPHODIESTERASE (EC 3.1.4.37) (CNP).
CN37_RAT	32	2',3'-CYCLIC NUCLEOTIDE 3'-PHOSPHODIESTERASE (EC 3.1.4.37) (CNPI).
CYAA_BORPE	344	CALMODULIN-SENSITIVE ADENYLATE CYCLASE PRECURSOR (EC 4.6.1.1) (ATP PYROPHOSPHATE-LYASE) (ADENYLYL CYCLASE) (CYCLOLYSIN) (CONTAINS: HEMOLYSIN).
DP3X_BACSU	40	DNA POLYMERASE III SUBUNITS GAMMA AND TAU (EC 2.7.7.7).
DP3X_ECOLI	40	DNA POLYMERASE III SUBUNITS GAMMA AND TAU (EC 2.7.7.7).
DPKI_VZV4	14	DEOXYPYRIMIDINE KINASE (EC 2.7.1.-) (PYRIMIDINE DEOXYRIBONUCLEOSIDE KINASE).
DPOL_EBV	807	DNA POLYMERASE (EC 2.7.7.7).
DPOL_HPBV4	225	DNA POLYMERASE (EC 2.7.7.7).
DPOL_HPBVR	225	DNA POLYMERASE (EC 2.7.7.7).
EF11_DROME	9	ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA) (50 KD FEMALE-SPECIFIC PROTEIN).
EF11_RHIRA	9	ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA).
EF11_RHIRA	213	as above
EF1A_EUGGR	9	ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA).
EF1A_YEAST	213	ELONGATION FACTOR 1-ALPHA (EF-1-ALPHA).
EF2_CRIGR	21	ELONGATION FACTOR 2 (EF-2).
EF2_DICDI	21	ELONGATION FACTOR 2 (EF-2).
EF2_DROME	21	ELONGATION FACTOR 2 (EF-2).
EF2_HALHA	23	ELONGATION FACTOR 2 (EF-2).
EF2_METVA	23	ELONGATION FACTOR 2 (EF-2).
EF2_METVA	270	as above
EFG_ECOLI	11	ELONGATION FACTOR G (EF-G).
EFG_MICLU	10	ELONGATION FACTOR G (EF-G).
EFG_SPIPL	12	ELONGATION FACTOR G (EF-G).
EFG_THETH	14	ELONGATION FACTOR G (EF-G).
EFTU_ASTLO	14	ELONGATION FACTOR TU (EF-TU).
EFTU_ASTLO	386	as above
EFTU_ECOLI	13	ELONGATION FACTOR TU (EF-TU).
EFTU_HALMA	7	ELONGATION FACTOR TU (EF-TU).
EFTU_METVA	9	ELONGATION FACTOR TU (EF-TU).
EFTU_MICLU	14	ELONGATION FACTOR TU (EF-TU).
EFTU_MYCGE	14	ELONGATION FACTOR TU (EF-TU).

Table 3.1a The segments with the ATP/GTP-binding site motif A are listed below. The SWISSPROT release 17 identity code is given, followed by the residue number of the start of the segment, and then a brief description of the protein.

EFTU_SPIPL	14	ELONGATION FACTOR TU (EF-TU).
EFTU_THEMA	14	ELONGATION FACTOR TU (EF-TU) (FRAGMENT).
EFTU_THETH	14	ELONGATION FACTOR TU (EF-TU).
EFTU_YEAST	50	ELONGATION FACTOR TU, MITOCHONDRIAL PRECURSOR.
ERA_ECOLI	10	GTP-BINDING ERA PROTEIN.
F261_RAT	43	6-PHOSPHOFRUCTO-2-KINASE (EC 2.7.1.105) / FRUCTOSE-2,6-BISPHOSPHATASE (EC 3.1.3.46) LIVER ISOZYME 1 (L-TYPE).
FLIC_SERMA	213	FLAGELLIN.
GBA1_YEAST	43	GUANINE NUCLEOTIDE-BINDING PROTEIN GP1-ALPHA.
GBA2_DICDI	33	GUANINE NUCLEOTIDE-BINDING PROTEIN ALPHA-2.
GBA2_YEAST	125	GUANINE NUCLEOTIDE-BINDING PROTEIN GP2-ALPHA.
GBAS_BOVIN	42	GUANINE NUCLEOTIDE-BINDING PROTEIN G(S), ALPHA SUBUNIT (ADENYLATE CYCLASE-STIMULATING).
GBI1_BOVIN	35	GUANINE NUCLEOTIDE-BINDING PROTEIN G(I), ALPHA-1 SUBUNIT (ADENYLATE CYCLASE-INHIBITING).
GBI1_XENLA	35	GUANINE NUCLEOTIDE-BINDING PROTEIN G(I), ALPHA-1 SUBUNIT (ADENYLATE CYCLASE-INHIBITING).
GST1_HUMAN	76	GST1-HS GTP-BINDING PROTEIN.
GTR1_HUMAN	106	GLUCOSE TRANSPORTER PROTEIN, ERYTHROCYTE/BRAIN.
IF2_BACST	246	INITIATION FACTOR IF-2.
IF2_ECOLI	395	INITIATION FACTOR IF-2.
IF42_MOUSE	72	EUKARYOTIC INITIATION FACTOR 4A-II (EIF-4A-II).
KGUA_YEAST	3	GUANYLATE KINASE (EC 2.7.4.8) (GMP KINASE).
KIPN_BPT4	4	POLYNUCLEOTIDE KINASE (EC 2.7.-.-) (PNK).
KITH_BPT4	4	THYMIDINE KINASE (EC 2.7.1.21).
KITH_CHICK	21	THYMIDINE KINASE, CYTOSOLIC (EC 2.7.1.21).
KITH_CRIGR	21	THYMIDINE KINASE, CYTOSOLIC (EC 2.7.1.21).
KITH_EBV	286	THYMIDINE KINASE (EC 2.7.1.21).
KITH_FOWPV	6	THYMIDINE KINASE (EC 2.7.1.21).
KITH_HSV11	51	THYMIDINE KINASE (EC 2.7.1.21).
KITH_HSVE1	27	THYMIDINE KINASE (EC 2.7.1.21).
KITH_HSVF	22	THYMIDINE KINASE (EC 2.7.1.21).
KITH_HSVMR	12	THYMIDINE KINASE (EC 2.7.1.21).
KITH_HSVTF	12	THYMIDINE KINASE (EC 2.7.1.21).
KITH_HUMAN	106	THYMIDINE KINASE, CYTOSOLIC (EC 2.7.1.21).
KITH_MONPV	6	THYMIDINE KINASE (EC 2.7.1.21).
KITH_SFVKA	6	THYMIDINE KINASE (EC 2.7.1.21).
KKIT_HUMAN	488	KIT PROTO-ONCOGENE TYROSINE KINASE PRECURSOR (EC 2.7.1.112).
KKIT_MOUSE	491	KIT PROTO-ONCOGENE TYROSINE KINASE PRECURSOR (EC 2.7.1.112).
KPC1_HUMAN	509	PROTEIN KINASE C, BETA-I TYPE (EC 2.7.1.-) (PKC-BETA-1).
KPCA_BOVIN	506	PROTEIN KINASE C, ALPHA TYPE (EC 2.7.1.-) (PKC-ALPHA).
KPCG_BOVIN	508	PROTEIN KINASE C, GAMMA TYPE (EC 2.7.1.-) (PKC-GAMMA) (FRAGMENT).
KPPR_SPIOL	58	PHOSPHORIBULOKINASE PRECURSOR (EC 2.7.1.19) (PHOSPHOPENTOKINASE) (PRKASE) (PRK).
KTHY_VACCV	6	THYMIDYLATE KINASE (EC 2.7.4.9) (DTMP KINASE).
KTHY_YEAST	7	THYMIDYLATE KINASE (EC 2.7.4.9) (DTMP KINASE).
MDR1_HUMAN	422	MULTIDRUG RESISTANCE PROTEIN 1 (P-GLYCOPROTEIN 1).
MDR1_HUMAN	1065	as above
MDR_PLAFF	408	MULTIDRUG RESISTANCE PROTEIN (CHLOROQUINE RESISTANCE PROTEIN).
MDR_PLAFF	1156	as above
MYS1_YEAST	175	MYOSIN II (MYOSIN-LIKE HEAVY CHAIN).
MYSA_CAEEL	174	MYOSIN HEAVY CHAIN A (MHC A).
MYSA_CAEEL	641	as above
MYSA_RAT	55	MYOSIN HEAVY CHAIN, CARDIAC MUSCLE ALPHA ISOFORM.
MYSA_RAT	172	as above
MYSB_CAEEL	172	MYOSIN HEAVY CHAIN B (MHC B).
MYSB_CAEEL	639	as above
MYSB_HUMAN	56	MYOSIN HEAVY CHAIN, CARDIAC MUSCLE BETA ISOFORM.
MYSC_ACACA	96	MYOSIN IC HEAVY CHAIN.
MYSC_CAEEL	642	MYOSIN HEAVY CHAIN C (MHC C).
MYSD_CAEEL	634	MYOSIN HEAVY CHAIN D (MHC D).
MYSG_CHICK	171	MYOSIN HEAVY CHAIN, GIZZARD SMOOTH MUSCLE.
MYSH_BOVIN	96	MYOSIN I HEAVY CHAIN-LIKE PROTEIN (MIHC).
MYSN_ACACA	177	MYOSIN II HEAVY CHAIN, NON MUSCLE.
MYS_DICDI	174	MYOSIN HEAVY CHAIN.
MYS_DROME	174	MYOSIN HEAVY CHAIN, MUSCLE (FRAGMENT).
NIF1_AZOVI	4	NITROGENASE IRON PROTEIN (EC 1.18.6.1) (NITROGENASE COMPONENT II) (NITROGENASE REDUCTASE).
NIF1_CLOPA	3	NITROGENASE IRON PROTEIN (EC 1.18.6.1) (NITROGENASE COMPONENT II) (NITROGENASE REDUCTASE).
NIF1_CLOPA	19	as above
NIF2_METIV	4	NITROGENASE IRON PROTEIN (EC 1.18.6.1) (NITROGENASE COMPONENT II) (NITROGENASE REDUCTASE).
NIF2_METTL	3	NITROGENASE IRON PROTEIN (EC 1.18.6.1) (NITROGENASE COMPONENT II) (NITROGENASE REDUCTASE).
NIF3_AZOVI	4	NITROGENASE IRON PROTEIN (EC 1.18.6.1) (NITROGENASE COMPONENT II) (NITROGENASE REDUCTASE).
NIF3_CLOPA	4	NITROGENASE IRON PROTEIN (EC 1.18.6.1) (NITROGENASE COMPONENT II) (NITROGENASE REDUCTASE).
NIF6_CLOPA	3	NITROGENASE IRON PROTEIN (EC 1.18.6.1) (NITROGENASE COMPONENT II) (NITROGENASE REDUCTASE).
NIFA_RHILE	489	NIF-SPECIFIC REGULATORY PROTEIN.
NIFH_ANASP	8	NITROGENASE IRON PROTEIN (EC 1.18.6.1) (NITROGENASE COMPONENT II) (NITROGENASE REDUCTASE).
NIFH_FRASR	3	NITROGENASE IRON PROTEIN (EC 1.18.6.1) (NITROGENASE COMPONENT II) (NITROGENASE REDUCTASE).
NIFH_METVO	3	NITROGENASE IRON PROTEIN (EC 1.18.6.1) (NITROGENASE COMPONENT II) (NITROGENASE REDUCTASE).
NIFH_THIFE	9	NITROGENASE IRON PROTEIN (EC 1.18.6.1) (NITROGENASE COMPONENT II) (NITROGENASE REDUCTASE).
P23_RAT	4	RAS-RELATED PROTEIN P23 (RAS-RELATED PROTEIN BRL-RAS).
P25A_BOVIN	23	GTP-BINDING PROTEIN SMG P25A.
P25B_BOVIN	24	GTP-BINDING PROTEIN SMG P25B.
P29_MYCHR	37	PROTEIN P29.
PABP_YEAST	153	POLYADENYLATE-BINDING PROTEIN, CYTOPLASMIC AND NUCLEAR (PABP).
PPCK_YEAST	244	PHOSPHOENOLPYRUVATE CARBOXYKINASE (ATP) (EC 4.1.1.49).
PRIA_RHORU	248	POSSIBLE PRIMOSOMAL PROTEIN N' (ATP SYNTHASE SUBUNITS REGION ORF 2).
RA50_YEAST	29	DNS REPAIR PROTEIN RAD50 (153 KD PROTEIN).
RAB2_HUMAN	8	RAS-RELATED PROTEIN RAB-2.
RAB4_RAT	10	RAS-RELATED PROTEIN RAB-4.
RAC1_HUMAN	5	RAS-RELATED C3 BOTULINUM TOXIN SUBSTRATE 1 (SMALL G PROTEIN) (GX).
RAD9_YEAST	1238	PROTEIN RAD9.
RADH_YEAST	30	PUTATIVE DNA HELICASE.
RAL_CALJA	16	RAS-RELATED PROTEIN RAL.

Table 3.1a (cont.)

RAP2_HUMAN	5	RAS-RELATED PROTEIN RAP-2A.
RAPA_HUMAN	5	RAS-RELATED PROTEIN RAP-1A (C21KG) (KREV-1 PROTEIN) (SMG-21) (G-22K).
RAS1_DICDI	5	TRANSFORMING PROTEIN P23.
RAS1_YEAST	12	TRANSFORMING PROTEIN HOMOLOG H-RAS-1.
RAS2_DROME	7	RAS-LIKE PROTEIN 2.
RAS2_YEAST	12	TRANSFORMING PROTEIN HOMOLOG H-RAS-2.
RAS3_DROME	5	RAS-LIKE PROTEIN 3 (ROUGHENED PROTEIN).
RASH_MSV	5	TRANSFORMING PROTEIN P21.
RASH_MSVHA	57	TRANSFORMING PROTEINS P21 AND P29.
RASK_MSVKI	5	TRANSFORMING PROTEIN 21.
RASK_RAT	5	TRANSFORMING PROTEIN P21/K-RAS (FRAGMENT).
RAS_CARAU	5	TRANSFORMING PROTEIN RAS (FRAGMENT).
RAS_SCHPO	10	TRANSFORMING PROTEIN HOMOLOG RAS.
RHO1_YEAST	12	TRANSFORMING PROTEIN HOMOLOG RHO1.
RHO2_YEAST	9	TRANSFORMING PROTEIN HOMOLOG RHO2.
RHOB_HUMAN	7	TRANSFORMING PROTEIN RHOB (H6).
RHO_APLCA	7	TRANSFORMING PROTEIN RHO.
RPOU_METTH	286	DNA-DIRECTED RNA POLYMERASE SUBUNIT B' (EC 2.7.7.6).
RRAS_HUMAN	31	RAS-RELATED PROTEIN R-RAS.
RRP2_INBAC	287	RNA-DIRECTED RNA POLYMERASE SUBUNIT P2 (EC 2.7.7.48) (P2 OR PA PROTEIN).
RSR1_YEAST	5	RAS-RELATED PROTEIN RSR1.
SEC4_YEAST	4	RAS-RELATED PROTEIN SEC4.
SEC4_YEAST	22	as above
STE6_YEAST	387	MATING FACTOR A SECRETION PROTEIN STE6 (MULTIPLE DRUG RESISTANCE PROTEIN HOMOLOGUE) (P-GLYCOPROTEIN).
STE6_YEAST	1082	as above
TALA_BFDV	348	LARGE T ANTIGEN.
TALA_POVBK	423	LARGE T ANTIGEN.
TALA_POVHA	543	LARGE T ANTIGEN.
TALA_POVJC	422	LARGE T ANTIGEN.
TALA_POVLY	484	LARGE T ANTIGEN.
TALA_POVM3	570	LARGE T ANTIGEN.
TALA_SV40	421	LARGE T ANTIGEN.
VAT1_DAUCA	247	VACUOLAR ATP SYNTHASE 69 KD SUBUNIT (EC 3.6.1.34).
VAT1_NEUCR	241	VACUOLAR ATP SYNTHASE 67 KD SUBUNIT (EC 3.6.1.34).
YPT1_SCHPO	13	YPT1-RELATED PROTEIN.
YPT1_YEAST	10	RAS-RELATED PROTEIN YP2.

Table 3.1a (cont.)

A2AA_HUMAN	92	ALPHA-2A ADRENERGIC RECEPTOR (SUBTYPE C10).
ACHE_BOVIN	176	ACETYLCHOLINE RECEPTOR PROTEIN, EPSILON CHAIN PRECURSOR.
AEP_YARLI	231	ALKALINE EXTRACELLULAR PROTEASE PRECURSOR (EC 3.4.21.14) (AEP).
AGI3_WHEAT	42	AGGLUTININ ISOLECTIN 3 PRECURSOR (WGA3) (FRAGMENT).
AGI_HORVU	68	ROOT-SPECIFIC LECTIN PRECURSOR.
ALK1_HUMAN	17	ANTILEUKOPROTEINASE 1 (ALP) (HUSI-1) (SEMINAL PROTEINASE INHIBITOR) (SECRETORY LEUKOCYTE PROTEASE INHIBITOR) (BLPI) (MUCUS PROTEINASE INHIBITOR) (MPI).
AMYA_DROME	452	ALPHA-AMYLASE A PRECURSOR (EC 3.2.1.1) (1,4-ALPHA-D-GLUCAN GLUCANOHYDROLASE).
AMYH_SACDI	274	GLUCOAMYLASE S1 PRECURSOR (EC 3.2.1.3) (GLUCAN 1,4-ALPHA-GLUCOSIDASE) (1,4-ALPHA-D-GLUCAN GLUCOHYDROLASE).
AMY_BACSU	349	ALPHA-AMYLASE PRECURSOR (EC 3.2.1.1) (1,4-ALPHA-D-GLUCAN GLUCANOHYDROLASE).
AMY_STRLM	276	ALPHA-AMYLASE PRECURSOR (EC 3.2.1.1) (1,4-ALPHA-D-GLUCAN GLUCANOHYDROLASE).
APB_HUMAN	2842	APOLIPOPROTEIN B-100 PRECURSOR (APO B-100/APO B-48).
AROA_ECOLI	27	3-PHOSPHOSHIKIMATE 1-CARBOXYVINYLTRANSFERASE (EC 2.5.1.19) (5- ENOLPYRUVYLSHIKIMATE-3-PHOSPHATE SYNTHASE) (EPSP SYNTHASE).
AROA_SALTY	27	3-PHOSPHOSHIKIMATE 1-CARBOXYVINYLTRANSFERASE (EC 2.5.1.19) (5- ENOLPYRUVYLSHIKIMATE-3-PHOSPHATE SYNTHASE) (EPSP SYNTHASE).
ARRS_BOVIN	130	ARRESTIN (RETINAL S-ANTIGEN) (48 KD PROTEIN) (S-AG).
ARRS_HUMAN	134	ARRESTIN (RETINAL S-ANTIGEN) (48 KD PROTEIN) (S-AG).
ARRS_RAT	131	ARRESTIN (RETINAL S-ANTIGEN) (48 KD PROTEIN) (S-AG).
BGAL_ECOLI	836	BETA-GALACTOSIDASE (EC 3.2.1.23) (LACTASE).
BGLS_BUTFI	328	BETA-GLUCOSIDASE A (EC 3.2.1.21) (GENTIOBIASE) (CELLOBIASE) (BETA-D- GLUCOSIDE GLUCOHYDROLASE).
CA1Y_HUMAN	843	COLLAGEN ALPHA 1(XI) CHAIN PRECURSOR.
CA2Y_HUMAN	93	COLLAGEN ALPHA 2(XI) CHAIN (FRAGMENT).
CA2Y_HUMAN	609	as above
CADH_PHAVU	410	CINNAMYL-ALCOHOL DEHYDROGENASE (EC 1.1.1.195) (CAD).
CAO4_CANMA	471	ACYL-COENZYME A OXIDASE (EC 1.3.3.6) (AOX).
CATA_ECOLI	260	CATALASE HPI (EC 1.11.1.6) (HYDROPEROXIDASE I).
CATG_HUMAN	208	CATHEPSIN G PRECURSOR (EC 3.4.21.20).
CCPR_YEAST	236	CYTOCHROME C PEROXIDASE PRECURSOR (EC 1.11.1.5) (CCP).
CD11_MOUSE	187	T-CELL SURFACE GLYCOPROTEIN CD1.1 PRECURSOR (CD1.1 ANTIGEN).
CD12_MOUSE	187	T-CELL SURFACE GLYCOPROTEIN CD1.2 PRECURSOR (CD1.2 ANTIGEN).
CMGA_HUMAN	310	CHROMOGRANIN A PRECURSOR (CGA) (CONTAINS: PANCREASTATIN AND WE-14) (PITUITARY SECRETORY PROTEIN I) (SP-I).
CMGA_PIG	2	CHROMOGRANIN A PRECURSOR (CGA) (CONTAINS: PANCREASTATIN AND WE-14) (FRAGMENT).
CO3_HUMAN	311	COMPLEMENT C3 PRECURSOR.
CO3_MOUSE	312	COMPLEMENT C3 PRECURSOR.
COGS_HYPLI	162	COLLAGENASE (EC 3.4.21.-).
COPB_PSESM	158	COPPER RESISTANCE PROTEIN B PRECURSOR.
CTR2_VESCR	33	CHYMOTRYPSIN II (EC 3.4.21.1).
CTR2_VESOR	33	CHYMOTRYPSIN II (EC 3.4.21.1).
CTRB_BOVIN	132	CHYMOTRYPSINOGEN B (EC 3.4.21.1).
CTRB_RAT	150	CHYMOTRYPSINOGEN B PRECURSOR (EC 3.4.21.1).
CYC_EISFO	1	CYTOCHROME C.
CYPH_ECHGR	62	PEPTIDYL-PROLYL CIS-TRANS ISOMERASE (EC 5.2.1.8) (PPIASE) (ROTAMASE) (CYCLOPHILIN) (CYCLOSPORIN A-BINDING PROTEIN) (FRAGMENT).
CYPH_NEUCR	120	PEPTIDYL-PROLYL CIS-TRANS ISOMERASE PRECURSOR (EC 5.2.1.8) (PPIASE) (ROTAMASE) (CYCLOPHILIN) (CYCLOSPORIN A-BINDING PROTEIN) (CPH).
CYPH_YEAST	63	PEPTIDYL-PROLYL CIS-TRANS ISOMERASE (EC 5.2.1.8) (PPIASE) (ROTAMASE) (CYCLOPHILIN) (CYCLOSPORIN A-BINDING PROTEIN) (CPH).
DCUP_HUMAN	286	UROPORPHYRINOGEN DECARBOXYLASE (EC 4.1.1.37).
DDLA_SALTY	1	D-ALANINE--D-ALANINE LIGASE A (EC 6.3.2.4) (D-ALANYLALANINE SYNTHETASE).
DHAM_HORSE	131	ALDEHYDE DEHYDROGENASE, MITOCHONDRIAL (EC 1.2.1.3) (CLASS 2).
DHOM_CORGL	101	HOMOSERINE DEHYDROGENASE (EC 1.1.1.3) (HDH).
ETC1_STAAU	124	ENTEROTOXIN TYPE C-1 PRECURSOR (SEC1).
EX5A_ECOLI	166	EXODEOXYRIBONUCLEASE V (EC 3.1.11.5), 67K POLYPEPTIDE (EXONUCLEASE V, ALPHA CHAIN).
EX5B_ECOLI	18	EXODEOXYRIBONUCLEASE V (EC 3.1.11.5), 135K POLYPEPTIDE (EXONUCLEASE V, 135 KD POLYPEPTIDE).
EXO2_BPT4	31	EXONUCLEASE SUBUNIT 2 (EC 3.1.11.-) (PROTEIN GP46).
EXO2_BPT5	30	POSSIBLE EXONUCLEASE SUBUNIT 2 (EC 3.1.11.-) (D13).
EXON_VZVD	174	ALKALINE EXONUCLEASE (EC 3.1.11.-).
FA8_HUMAN	214	COAGULATION FACTOR VIII PRECURSOR (PROCOAGULANT COMPONENT).
FDHF_ECOLI	24	FORMATE DEHYDROGENASE (EC 1.2.1.2), FORMATE-HYDROGEN-LYASE-LINKED, SELENOCYSTEINE-CONTAINING POLYPEPTIDE (FORMATE DEHYDROGENASE-H ALPHA SUBUNIT).
FLAV_AZOVI	78	FLAVODOXIN.
FRIL_HUMAN	80	FERRITIN LIGHT CHAIN.
GAG_FIVSD	15	GAG POLYPROTEIN (CONTAINS: CORE PROTEINS P15, P24 AND P10).
GAG_HIV2D	405	GAG POLYPROTEIN (CORE PROTEINS P16 AND P26).
GAG_HIV2I	405	GAG POLYPROTEIN (CORE PROTEINS P16 AND P26).
GAG_HIV2N	404	GAG POLYPROTEIN (CORE PROTEINS P16 AND P26).
GAG_SIVS4	409	GAG POLYPROTEIN (CORE PROTEINS P17 AND P24).
GLTB_ECOLI	70	GLUTAMATE SYNTHASE (NADPH) LARGE CHAIN PRECURSOR (EC 1.4.1.13) (NADPH-GOGAT).
GPDA_DROME	263	GLYCEROL-3-PHOSPHATE DEHYDROGENASE, CYTOPLASMIC (EC 1.1.1.8) (ALPHA- GPDH-M).
GPDA_DROVI	271	GLYCEROL-3-PHOSPHATE DEHYDROGENASE, CYTOPLASMIC (EC 1.1.1.8) (ALPHA- GPDH-M).

Table 3.1b The segments incorrectly identified by PROMOT as having the ATP/GTP-binding site motif A are listed below. The SWISSPROT release 17 identity code is given, followed by the residue number of the start of the segment, and then a brief description of the protein.

GPDA_MOUSE	270	GLYCEROL-3-PHOSPHATE DEHYDROGENASE, CYTOPLASMIC (EC 1.1.1.8) (ALPHA- GPDH-M).
GUND_CLOTM	117	ENDOGLUCANASE D PRECURSOR (EC 3.2.1.4) (EGD) (ENDO-1,4-BETA-GLUCANASE) (CELLULASE).
H1B_STRPU	85	HISTONE H1-BETA, LATE EMBRYONIC.
H1G_STRPU	85	HISTONE H1-GAMMA, LATE.
H1_LYTPI	85	LATE HISTONE H1.
H2A1_TETPY	1	HISTONE H2A.1.
H2A_SEPOF	1	HISTONE H2A.
H2A_SIPNU	1	HISTONE H2A.
H2B1_HUMAN	16	HISTONE H2B.1.
HBA_ELEEL	50	HEMOGLOBIN ALPHA CHAIN.
HBB4_SALIR	54	HEMOGLOBIN BETA-IV CHAIN.
HBG4_HUMAN	175	TRANSFORMING PROTEIN HST/KS3 (HBGF-4).
HEMA_IAFPR	261	HEMAGGLUTININ PRECURSOR.
HEMA_IAPUE	147	HEMAGGLUTININ PRECURSOR.
HEMA_IATAI	147	HEMAGGLUTININ PRECURSOR (FRAGMENT).
HEMA_PI1HW	210	HEMAGGLUTININ-NEURAMINIDASE (EC 3.2.1.18).
HEMA_PI3B	208	HEMAGGLUTININ-NEURAMINIDASE (EC 3.2.1.18).
HEMA_PI3H4	208	HEMAGGLUTININ-NEURAMINIDASE (EC 3.2.1.18).
HEMA_PI3HW	208	HEMAGGLUTININ-NEURAMINIDASE (EC 3.2.1.18).
HISQ_SALTY	174	HISTIDINE PERMEASE MEMBRANE Q PROTEIN.
HLY2_ECOLI	497	HAEMOLYSIN SECRETION PROTEIN, CHROMOSOMAL.
HMCA_DROME	228	HOMEOTIC CAUDAL PROTEIN.
HO1_RAT	245	HEME OXYGENASE 1 (EC 1.14.99.3) (HO-1).
HS70_ONCMY	126	HEAT SHOCK 70 KD PROTEIN (HSP70) (FRAGMENT).
HS70_PLAFA	138	HEAT SHOCK 70 KD PROTEIN (HSP70) (CYTOPLASMIC ANTIGEN) (74.3 KD PROTEIN).
HS70_THEAN	126	HEAT SHOCK 70 KD PROTEIN (HSP 70.1).
HS7C_HUMAN	126	HEAT SHOCK COGNATE 71 KD PROTEIN.
HSF_YEAST	317	HEAT SHOCK FACTOR PROTEIN (HSF) (HEAT SHOCK TRANSCRIPTION FACTOR) (HSTF).
IGA_NEIGO	1470	IGA PROTEASE PRECURSOR (EC 3.4.24.13).
ILVD_ECOLI	141	DIHYDROXY-ACID DEHYDRATASE (EC 4.2.1.9).
ITAV_HUMAN	912	VITRONECTIN RECEPTOR ALPHA SUBUNIT PRECURSOR (INTEGRIN ALPHA-V) (CD51).
ITB2_MOUSE	384	CELL SURFACE ADHESION GLYCOPROTEINS LFA-1, CR3 AND P150,95, BETA- SUBUNIT PRECURSOR (CD18 ANTIGEN BETA SUBUNIT) (INTEGRIN BETA-2).
KEX1_KLULA	257	KEX1 PROTEASE PRECURSOR (EC 3.4.21.14).
LAC1_NEUCR	72	LACCASE PRECURSOR (EC 1.10.3.2) (BENZENEDIOL:OXYGEN OXIDOREDUCTASE) (URISHIOL OXIDASE) (ALLELE OR).
LDHH_HUMAN	31	L-LACTATE DEHYDROGENASE H CHAIN (EC 1.1.1.27) (LDH-B).
LDHH_PIG	31	L-LACTATE DEHYDROGENASE H CHAIN (EC 1.1.1.27) (LDH-B).
LEGA_PEA	314	LEGUMIN A PRECURSOR.
LIG2_PHACH	236	LIGNINASE H2 PRECURSOR (EC 1.11.1.-) (LIGNIN PEROXIDASE) (LG4).
LIG8_PHACH	24	LIGNINASE H8 PRECURSOR (EC 1.11.1.-) (LIGNIN PEROXIDASE).
LIPP_HUMAN	323	TRIACYLGLYCEROL LIPASE PRECURSOR (EC 3.1.1.3) (LIPASE, PANCREATIC).
LIP_STAAU	380	LIPASE PRECURSOR (EC 3.1.1.3) (GLYCEROL ESTER HYDROLASE).
LIP_STAHY	337	LIPASE PRECURSOR (EC 3.1.1.3) (TRIACYLGLYCEROL LIPASE).
MBHA_MYXXA	235	MYXOBACTERIAL HEMAGGLUTININ (LECTIN).
MX1_MOUSE	38	INTERFERON INDUCED MX1 PROTEIN (INFLUENZA RESISTANCE PROTEIN).
MYBC_MAIZE	40	ANTHOCYANIN REGULATORY C1 PROTEIN.
MYB_CHICK	29	MYB PROTO-ONCOGENE PROTEIN.
MYB_DROME	61	MYB PROTO-ONCOGENE PROTEIN.
MYB_DROME	603	as above
MYB_HUMAN	29	MYB PROTO-ONCOGENE PROTEIN.
NCA1_CHICK	659	NEURAL CELL ADHESION MOLECULE, LARGE ISOFORM PRECURSOR (N-CAM) (FRAGMENTS).
NCA1_CHICK	729	as above
NCA1_HUMAN	180	NEURAL CELL ADHESION MOLECULE, CLONE LAMBDA-4.4 (N-CAM) (FRAGMENT).
NCA1_XENLA	665	NEURAL CELL ADHESION MOLECULE, LARGE ISOFORM PRECURSOR (N-CAM).
NPRE_BACAM	144	NEUTRAL PROTEASE PRECURSOR (EC 3.4.24.4).
NPRE_BACAM	223	as above
NPRE_BACSU	144	NEUTRAL PROTEASE PRECURSOR (EC 3.4.24.4) (MCP 76).
NTRC_RHOCA	166	NITROGEN REGULATION PROTEIN NIFR1.
NU5M_BOVIN	212	NADH-UBIQUINONE OXIDOREDUCTASE CHAIN 5 (EC 1.6.5.3).
NYLA_PSES8	105	6-AMINOHEXANOATE-CYCLIC-DIMER HYDROLASE (EC 3.5.2.12) (NYLON OLIGOMERS DEGRADING ENZYME EI).
OCD_AGRT5	228	ORNITHINE CYCLODEAMINASE (EC 4.3.1.12) (OCD).
ODP2_ECOLI	306	DIHYDROLIPOAMIDE ACETYLTRANSFERASE COMPONENT (E2) OF PYRUVATE DEHYDROGENASE COMPLEX (EC 2.3.1.12).
OPS1_DROME	345	OPSIN RH1 (OUTER R1-R6 PHOTORECEPTOR CELLS OPSIN).
P1P_LACLC	1006	PI-TYPE PROTEINASE PRECURSOR (EC 3.4.21.14) (WALL-ASSOCIATED SERINE PROTEINASE).
PBPA_ECOLI	175	PENICILLIN-BINDING PROTEIN 1A (PBP-1A).
PGLR_LYCES	77	POLYGALACTURONASE 2A PRECURSOR (EC 3.2.1.15) (PG-2A) (PECTINASE).
PHCA_ANASP	71	C-PHYCOCYANIN ALPHA CHAIN.
PHCA_MASLA	70	C-PHYCOCYANIN ALPHA CHAIN.
PHLC_CLOPE	218	PHOSPHOLIPASE C PRECURSOR (EC 3.1.4.3) (PLC) (PHOSPHATIDYLCHOLINE CHOLINEPHOSPHOHYDROLASE) (ALPHA-TOXIN) (HEMOLYSIN).
PIPA_DROME	335	1-PHOSPHATIDYLINOSITOL-4,5-BISPHOSPHATE PHOSPHODIESTERASE (EC 3.1.4.11) (PHOSPHOINOSITIDE-SPECIFIC PHOSPHOLIPASE C).
PORI_NEUCR	35	OUTER MITOCHONDRIAL MEMBRANE PROTEIN PORIN.
PPB_ECOLI	234	ALKALINE PHOSPHATASE PRECURSOR (EC 3.1.3.1) (APASE).
PROS_HUMAN	195	PROSTATE SPECIFIC ANTIGEN PRECURSOR (EC 3.4.21.-) (PSA) (GAMMA- SEMINOPROTEIN).
PROV_ECOLI	56	PERIPHERAL MEMBRANE PROTEIN PROV.
PULE_KLEPN	250	PULLULANASE SECRETION PROTEIN PULE.
QA1S_NEUCR	103	QUINATE REPRESSOR.
RBSB_ECOLI	220	D-RIBOSE-BINDING PERIPLASMIC PROTEIN PRECURSOR.
RCA_ARATH	160	RIBULOSE BISPHOSPHATE CARBOXYLASE/OXYGENASE ACTIVASE PRECURSOR (RUBSICO ACTIVASE).
RCA_SPIOL	158	RIBULOSE BISPHOSPHATE CARBOXYLASE/OXYGENASE ACTIVASE PRECURSOR (RUBSICO ACTIVASE).
RNAS_ASPGI	32	RIBONUCLEASE ALPHA-SARCIN PRECURSOR (EC 3.1.27.-).
RNBR_BACAM	85	RIBONUCLEASE PRECURSOR (EC 3.1.27.-) (BARNASE).
RN_BACIN	37	RIBONUCLEASE PRECURSOR (EC 3.1.27.-) (BINASE).
SODF_ECOLI	39	SUPEROXIDE DISMUTASE (FE) (EC 1.15.1.1).
SYK1_ECOLI	296	LYSYL-TRNA SYNTHETASE (EC 6.1.1.6) (LYSINE--TRNA LIGASE).
SYLM_NEUCR	916	LEUCYL-TRNA SYNTHETASE, MITOCHONDRIAL PRECURSOR (EC 6.1.1.4) (LEUCINE--TRNA LIGASE).
TFE2_HUMAN	90	TRANSCRIPTION FACTOR E2-ALPHA.
THA1_RAT	293	THYROID HORMONE RECEPTOR ALPHA-1.
THER_BACCE	69	THERMOLYSIN (EC 3.4.24.4) (NEUTRAL PROTEASE) (NP).
TPIS_ECOLI	127	TRIOSEPHOSPHATE ISOMERASE (EC 5.3.1.1) (TIM).

Table 3.1b (cont.)

Win. size	No. seqs	No. wts	Performance of statistical method			Performance of neural network		
			Binding seqs	Non-binding seqs	All seqs	Binding seqs	Non-binding seqs	All seqs
17	349	340	82%	86%	84%	81%	74%	78%
16	345	320	83%	86%	84%	79%	71%	75%
15	335	300	79%	89%	84%	80%	73%	77%
14	331	280	77%	91%	83%	81%	71%	76%
13	325	260	83%	86%	84%	72%	72%	72%
12	318	240	81%	86%	83%	75%	71%	74%

Table 3.2 The variation of the performance of the neural network and the statistical program with window size (column headed win. size). The window size is the number of residues in the sequences presented to the network and the statistical program. The number of non-identical sequences (column headed no. seqs) falls, and the number weights required in the network (column headed no. wts) decreases with decreasing window size. The overall performances, and the performances on the binding and non-binding sequences are indicated.

predictive value after the initial PROMOT (Sternberg, 1991b) screen. Excluding conserved residues is a common procedure (Muskal *et al*, 1990). For each window position there were 20 input units, one per amino acid type. Hence, the input layer consisted of 340 units. The output layer was a single unit. Activation of the output unit was a prediction that the sequence would bind ATP or GTP.

A jack-knife procedure was used to test the neural network. At the start of each simulation, the network weights were assigned random values between -1.0 and 1.0. One of the sequences out of the 349 (197 binding and 152 non-binding) was removed for testing, and the network was trained on the remaining 348 sequences. Every example in the training set was presented to the network and the weights were updated according to the delta rule [eqn. 3.4]. Once the total error over all the examples in the training set was minimised (typically 30 learning steps were required), the weights were fixed, and the test sequence presented to the network. This was repeated for each sequence. Thus, the neural network was tested on all the sequences.

A statistical method, based on the motif searching program of the AMPS package (Barton and Sternberg, 1990), was compared with the neural network. In this statistical program, the PAM250 matrix (Dayhoff, 1978) measures the similarity between two protein sequences. A pattern can be defined by a set of aligned sequences. A score between the test sequence and each of the pattern-defining sequences is computed, summed over all the sequences in the training set and averaged. This score reflects the degree of homology between the test sequence and the set of pattern-defining sequences.

The statistical method was also tested using a jack-knife procedure. One of the 197 ATP/GTP-binding proteins was taken as the test sequence and the ATP/GTP-binding pattern was defined using the remaining 196 sequences. A score was generated for the one test ATP/GTP-binding sequence, using the statistical program. This was repeated for each of the ATP/GTP-binding proteins. Thus a set of scores for ATP/GTP-binding proteins was obtained. The ATP/GTP-binding pattern was then defined on all 197 sequences, and a score was calculated for each of the 152 sequences that do not bind ATP or GTP. All 349 sequences were sorted with respect to their score. If the statistical program perfectly classified ATP/GTP-binding proteins, then all of the ATP/GTP-binding sequences would score more highly than any of the proteins that do not bind ATP or GTP. However, it does not do this, and the boundary between binding and non-binding sequences is not clearly defined, *i.e.*, some binding sequences have a low score and some non-binding sequences have a high score. The predictive performance of the statistical method was calculated by choosing a cut-off score, such that the number of sequences incorrectly classified was minimised. Any sequence above this score was predicted to be ATP/GTP-binding and below this score to not bind ATP/GTP. The choice of cut-off score, -6.4, was obvious.

The training set does contain homologous proteins, but this is the case for both the methods, so neither is advantaged. The performance of the two methods was analysed with respect to the homology between the test sequence and the ATP/GTP-binding sequences. The homology score was generated by the statistical program. Greater homology was indicated by a higher score.

The neural network was simulated using code written in FORTRAN and implemented on a VAX 8700 under VMS V5.3. The statistical programs, AMPS (Barton and Sternberg, 1990) and PROMOT (Sternberg, 1991b) also are in FORTRAN.

3.4 **Results**

The overall performance of the neural network was 273 correct predictions out of a possible 349, *i.e.*, 78%, compared with 292 out of 349 (84%) for the statistical program. Table 3.3 shows the performance of the two methods in more detail. Compared to the statistical approach, the neural network predicts more of the ATP/GTP-binding proteins to be binding, but less of the non-binding proteins to be non-binding.

Figure 3.2 shows the distribution of homology (as calculated by the statistical method) in the 349 proteins with respect to the ATP/GTP-binding proteins, and it shows how the performances of the neural network and the statistical program

	Number predicted "binding"		Number predicted "non-binding"	
	neural network	statistical program	neural network	statistical program
"binding"	160	162	37	35
"non-binding"	39	22	113	130

Table 3.3 The performance of the neural network and the statistical program

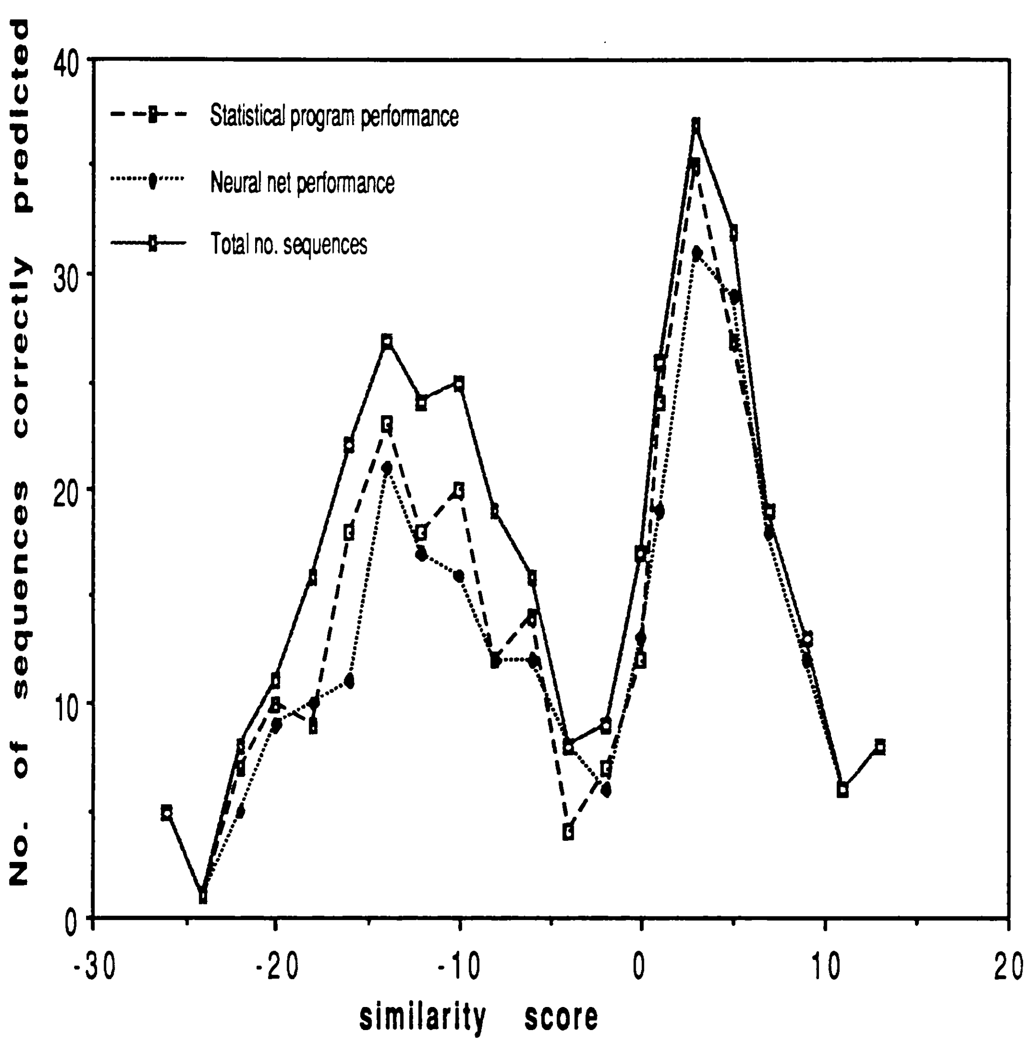


Figure 3.2 Performance of neural network and statistical program: For each method, the number of sequences correctly predicted is plotted as a function of the homology score calculated by the statistical program.

vary. The homology score is the similarity as calculated by the statistical method using the Dayhoff matrix. The two methods perform similarly on all the data. Some of the ATP/GTP-binding proteins have a degree of homology with each other and so there is peak at the high homology end of Figure 3.2. The broader peak, at the lower end, is mainly due to the non-binding sequences, which have little homology with the binding sequences.

Figures 3.3 and 3.4 show which data the two methods classify differently, and why their overall performances are similar. Figure 3.3 shows that the statistical method will predict that a sequence will bind ATP or GTP if its homology score is greater that a given value. The sequences that do bind ATP or GTP, but have low homology scores are incorrectly classified. Conversely, Figure 3.4 shows that highly scoring non-binding sequences are also incorrectly classified. This behaviour is due to the definition of the method. The neural network does have a more even distribution of incorrect predictions, but if the network is less likely to classify a high scoring non-binding sequence as binding, it is more likely to classify a high scoring binding sequence as non-binding. This explains why the neural network and the statistical method perform similarly, as shown in Figure 3.2.

The weights of a two-layer network have a well defined meaning. Each weight corresponds to a particular amino acid at a certain position. The pattern that the neural network is searching against is defined by the large positive and negative weights. The positive weights define the binding sequences and the negative

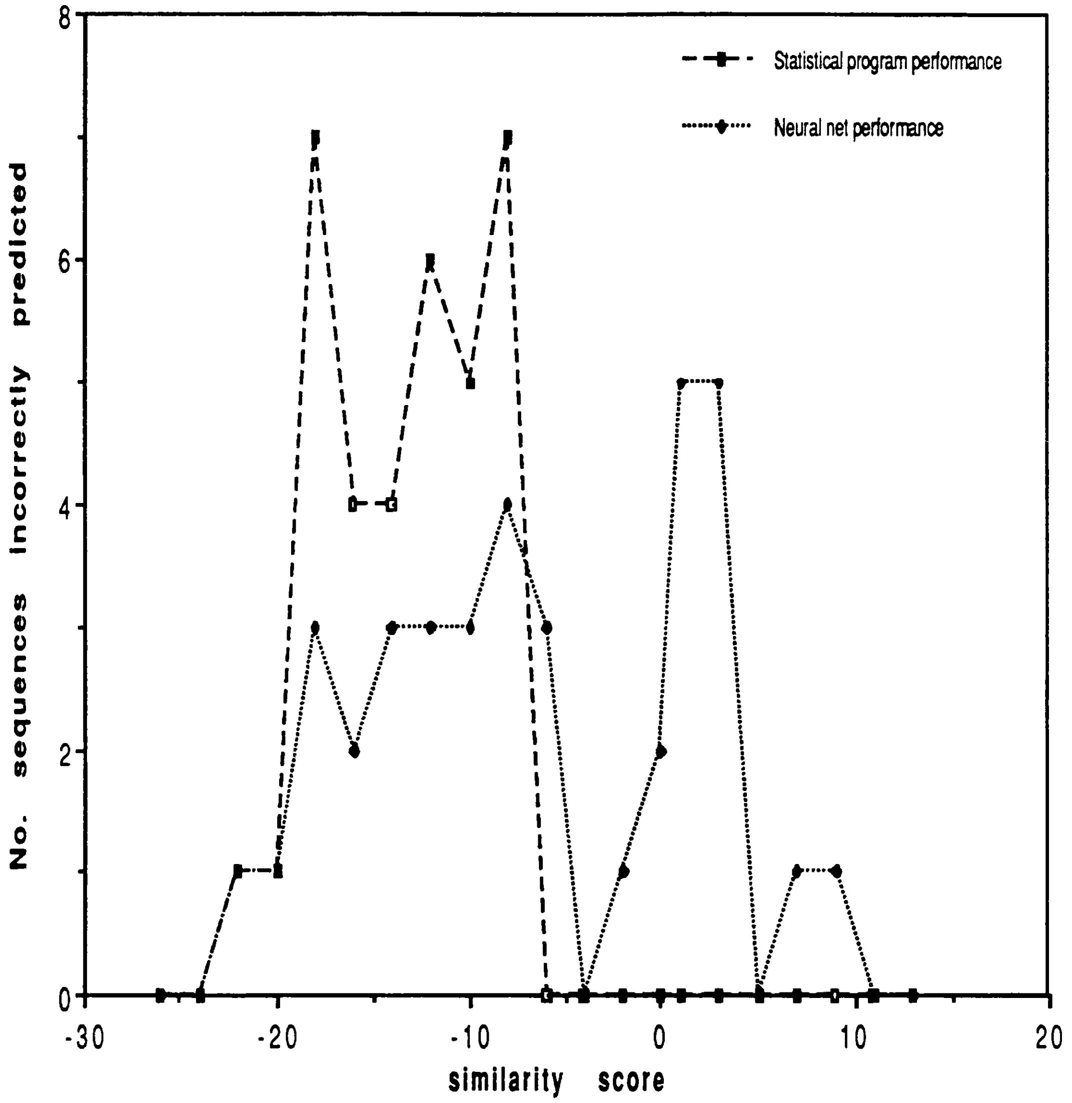


Figure 3.3 Number of ATP/GTP-binding proteins incorrectly predicted as non-binding: The number of sequences incorrectly classified as non-binding by each method is plotted as a function of the similarity score calculated by the statistical program.

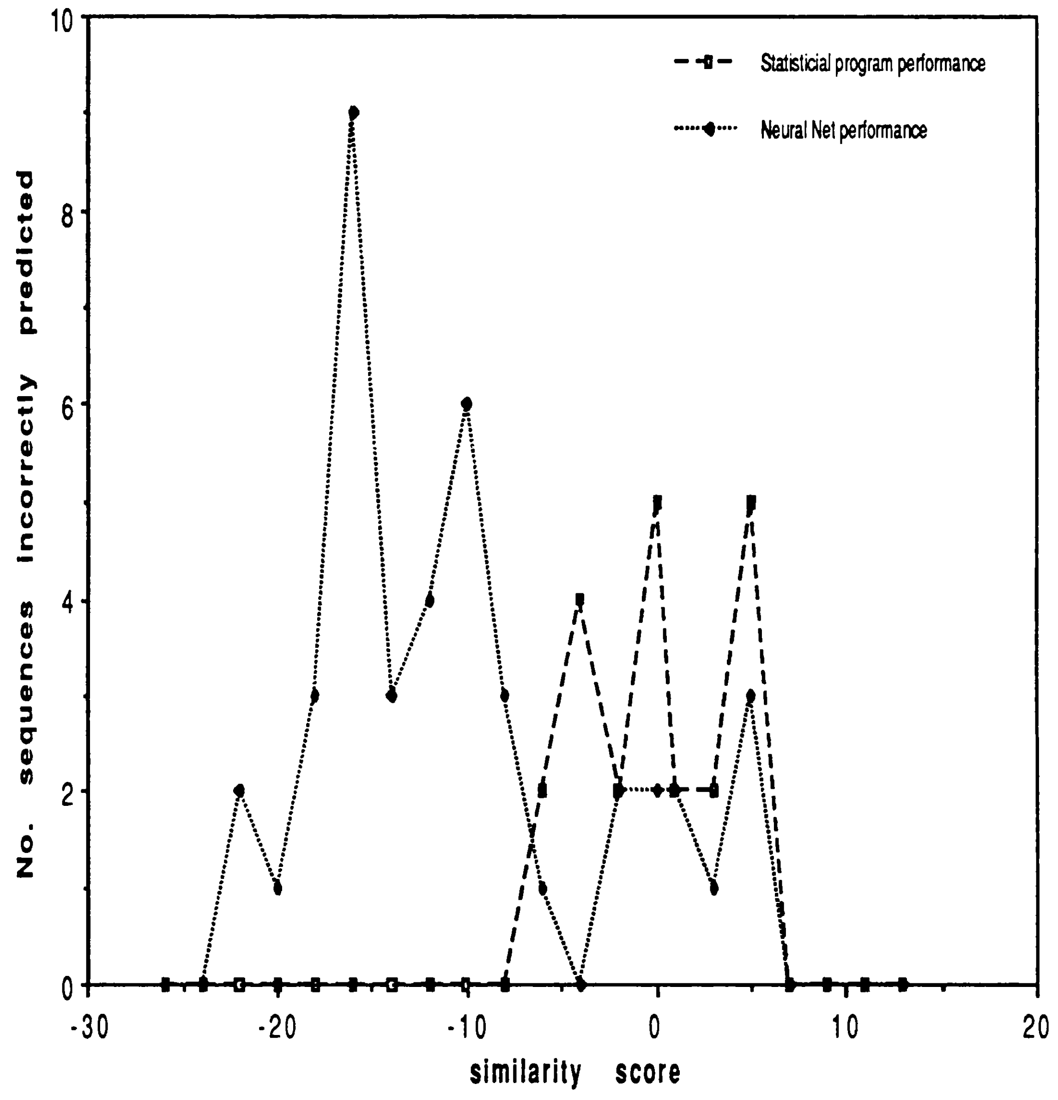


Figure 3.4 Number of non-binding-ATP/GTP proteins incorrectly predicted as binding: The number of sequences incorrectly classified as non-binding by each method is plotted as a function of the similarity score calculated by the statistical program.

	A	E	Q	D	N	L	G	K	S	V	R	T	P	I	M	F	Y	C	W	H
1	-04	53	20	-33	-00	16	-29	19	-39	20	-14	-28	-26	19	-41	06	25	-13	19	11
2	23	24	-20	-10	-04	38	-05	19	-05	-29	-13	-21	-24	19	-13	15	-00	-20	09	-12
3	-17	-10	-04	-27	-30	44	-05	-34	-55	31	00	26	-27	13	14	53	15	16	03	-11
4	-25	05	-01	42	-25	-34	-11	14	21	25	-10	00	-19	60	-13	00	18	-14	-23	-15
5	[AG]																			
6	04	-04	-14	06	10	-07	-22	-25	09	-05	-61	-04	-00	-25	39	-02	25	-32	-00	38
7	-39	-16	-14	-07	-59	-33	52	-36	00	30	44	00	-00	29	13	-01	19	-29	-06	17
8	-01	17	-13	-00	59	-26	47	33	-51	-09	03	-19	01	00	10	-25	-52	-23	-16	-25
9	11	-18	-40	05	22	-13	-44	-05	-31	29	15	-20	-39	21	33	02	19	14	-12	16
10	G																			
11	K																			
12	[ST]																			
13	22	06	-08	-35	-29	17	-03	-37	53	08	-37	78	-03	-28	04	-23	-11	-17	-16	-20
14	10	-00	-14	-15	05	48	16	-07	07	01	05	-16	-13	26	11	07	-25	-35	-10	-38
15	-11	-18	-26	-18	-27	35	-55	-06	28	-26	-04	09	24	-21	56	66	-39	14	24	-26
16	-12	-49	15	-19	-15	02	34	-03	-24	05	-04	-15	-10	01	43	-03	-46	28	16	-05
17	-30	35	-21	-19	02	-64	34	-20	09	04	34	10	-21	33	-07	11	-15	13	-10	-00
18	-83	-26	-31	13	-02	56	-42	-25	-16	27	-13	24	18	-15	36	56	27	05	-08	-08
19	20	-35	-22	-15	-54	-02	-30	61	-27	23	53	-04	-13	83	16	-33	-26	-12	-06	-26
20	15	-26	-14	-30	-05	-03	-01	23	-47	64	13	-49	-30	14	27	-02	-06	16	-14	16
21	-33	23	-23	-24	32	-20	-26	25	-01	-63	28	-13	-25	32	13	-20	49	-18	-06	-17

Average bias = 50

Table 3.4 Weight matrix generated by averaging 100 networks trained on all 349 examples. The columns are headed by the one letter amino acid codes that the weights correspond to. The rows are labelled by the position in the sequence of the residue that the weights correspond to. [AG], [ST], G and K are the four residues that define the ATP-binding motif used in PROMOT (Sternberg, 1991b). All numbers have been multiplied by 100.

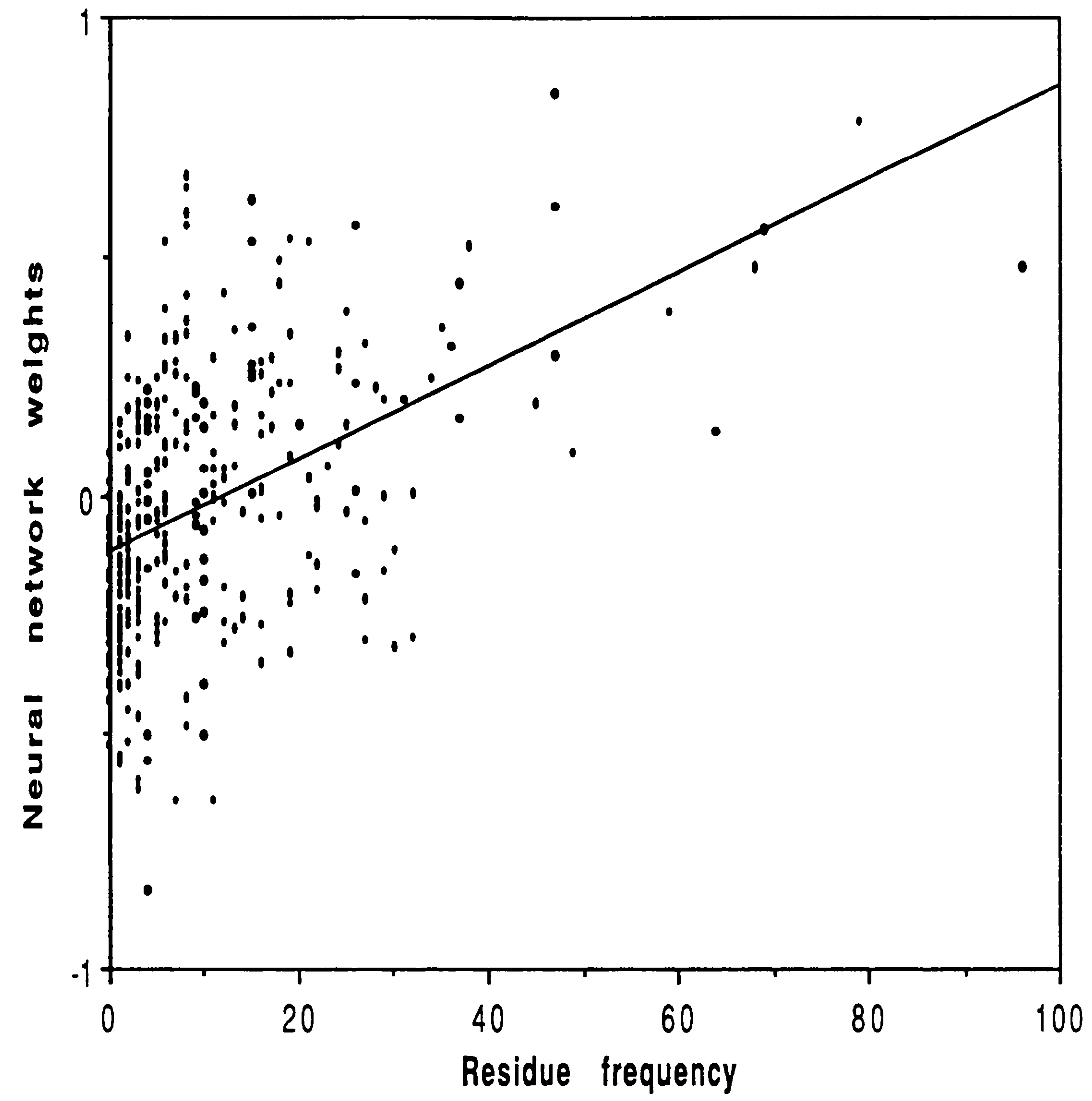


Figure 3.5 Comparison of the neural network weights and the residue frequency: The residue frequency is plotted against the neural network weights. Each of the weights corresponds to a particular amino acid at a certain position. The weight is plotted against the number of times that amino acid appears at that position in the ATP/GTP-binding proteins.

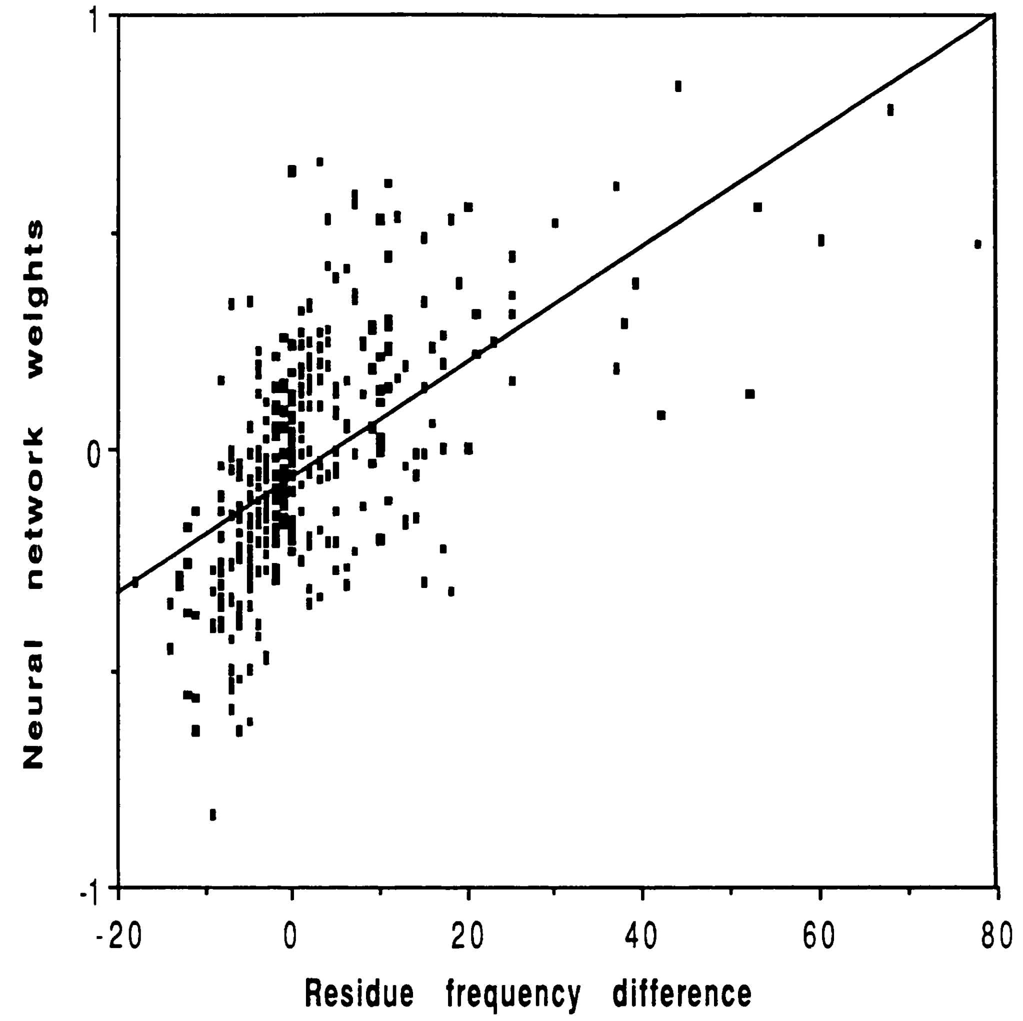


Figure 3.6 Comparison of neural network weights and the residue frequency difference: The residue frequency difference is plotted against the neural network weights. Each of the weights corresponds to a particular amino acid at a certain position. The weight is plotted against the number of times that amino acid appears at that position in the ATP/GTP-binding proteins minus the number of times that amino acid appears at that position in the non-binding proteins.

weights define the non-binding sequences. The weight matrix of the neural network, given in Table 3.4, was generated by running the network 100 times, training on all 349 sequences, so there were no randomly high weights. No clear pattern was deducible from the weights.

In Figure 3.5, the frequency of each amino acid at each position in the ATP/GTP-binding proteins (the residue frequency) is plotted against the corresponding neural network weight. There are 20 different amino acids and 17 different positions, so there are 340 points. Figure 3.6 is a similar plot, but the residue frequency of the non-binding proteins has been subtracted from the residue frequency of the binding proteins, to give the residue frequency difference. The correlation coefficient in Figure 3.5 is 0.22 and Figure 3.6 this rises to 0.36. The higher correlation coefficient for Figure 3.6 indicates that the weights of the neural network are more correlated to the residue frequency difference than to the residue frequency. This suggests that the neural network has incorporated information about the non-binding sequences, as well as the binding sequences, although, as neither coefficient is large, it suggests that the neural network approach and the statistical method are not identical.

3.5 Conclusion

The results show that the neural network is slightly worse than the alignment by homology method, although comparable. It would appear that, if sufficient trouble is taken over developing a statistical method, that it will perform as well, or better than a simple two-layer feed-forward neural network. Perceptron type neural networks can only distinguish linearly separable functions. Thus, if the data are represented in two dimensions, a perceptron can classify the data only if it is possible to separate the two classes by a straight line. More involved architectures may perform better, but the development of these is problematic in the case of motif recognition, because the small number of known protein structures limits the number of weights that may be used in the neural network. Two-layer feed-forward networks have been used in other applications and their 'success' proclaimed as vindication of the neural network approach, when, in fact, alternative statistical methods have not be tried on the same data.

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

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

Nucleic acid sequence analysis by neural networks appears to have been more sucessful than protein applications. The detection and prediction by neural networks of translational initiation sites (Stormo, *et al.*, 1982) and promoter regions in *E. coli* (Nakata *et al.*, 1988, Lushakin *et al.*, 1989; O'Neill, 1991; Demeler *et al.*, 1991), and splice junctions (Nakata *et al.*, 1985; Brunak *et al.*, 1991) and coding regions in human DNA (Uberbacher and Mural, 1991) are signifcantly better than other statistical approaches. This may in part be due the greater amount of nucleic acid sequence data and the fact that the bases of DNA can be encoded into a network with only four bits, whereas the encoding of amino acids requires twenty bits. The approach of Uberbacher and Mural (1991) suggests that the indirect analysis of protein sequences by neural networks may be more successful than the direct analyses so far, which have been limited by the size of the database being insufficient to exhibit in a generalisable way information higher than first order.

The use of aligned sequences in neural network applications, as outlined in this chapter, has been developed in more recent work by Frishman and Argos (1992) on motif recognition and Rost and Sander (1993a, 1993b) on secondary structure prediction. It is difficult to tell how much the success of these studies is due the use of aligned sequences and how much is due to the use of neural networks. In applications to problems in protein sequence analysis, it is still unclear if neural networks have yielded significant improvements over other current methodologies. However, the success of the approach in the analysis of nucleic acid sequences and in other fields suggests that as the protein structure database grows and perhaps with the incorporation of other information, the power of neural networks may be more fully exploited in the analysis of protein sequences.

Chapter 4

Quantitative structure-activity relationships:
neural networks and inductive logic programming
compared to statistical methods.
The inhibition of dihydrofolate reductase by pyrimidines

4.1 Synopsis

Recent innovations in methodology and in data representation applied to quantitative structure-activity relationship (QSAR) analysis have been evaluated. Neural networks and inductive logic programming (ILP) have been compared to the traditional statistical techniques of linear regression, nearest neighbour algorithms and decision trees. A new representation of drugs by physicochemical attributes (PCAs) (King *et al.*, 1992) has been extended to cover more substituents. The PCA representation has been benchmarked against the widely used Hansch parameters for the QSAR of the inhibition of *E. coli* dihydrofolate reductase (DHFR) by 2,4-diamino-5-(substituted-benzyl) pyrimidines, and, in the subsequent chapter, the inhibition of rodent DHFR by 2,4-diamino-6,6-dimethyl-5-phenyl-dihydrotriazines. Neural networks and ILP perform better than the traditional statistical techniques on the PCA representation, but the difference is not statistically significant. The PCA representation does not consistently give more accurate QSARs than Hansch parameters, but does allow the formulation of rules relating the activity of the inhibitors to their chemical structure.

4.2 **Introduction**

In this chapter, the recent innovations in method and data representation in QSAR analysis have been assessed, using the inhibition of *E. coli* dihydrofolate reductase (DHFR) by 2,4-diamino-5-(substituted-benzyl) pyrimidines (Figure 4.1). Neural networks and inductive logic programming (ILP) have been compared against the traditional statistical techniques of linear regression, nearest neighbour algorithms and decision trees. A new representation of drugs by PCAs (King *et al.*, 1992) has been extended to cover many more substituents, for use in QSAR analysis. This chapter and the subsequent chapter provide a thorough comparison of neural networks and ILP against traditional statistical methods in QSAR, and with this framework, the new PCA representation has been evaluated. The work presented here on ILP and decision trees was done by Dr. Ross King.

4.3 **Methods**

4.3.1 Data

The data used in the study were 74 2,4-diamino-5-(substituted-benzyl) pyrimidines (I), whose substituents are

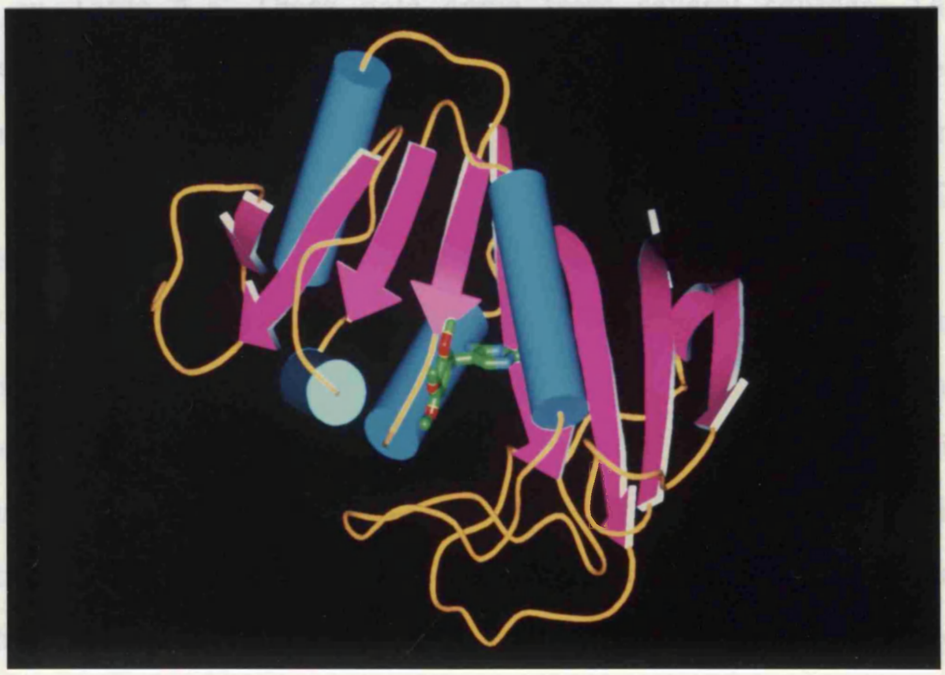
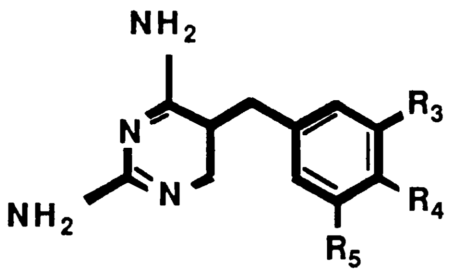


Figure 4.1 Model of trimethoprim bound to DHFR, displayed using the graphics program PREPI by Dr. Suhail Islam.



I

listed in Table 4.1. These data come from several sources. The first 44 drugs (numbers 1 - 44 in Table 4.1) have been analysed by linear regression (Hansch *et al.*, 1982); these 44 and 11 more from Roth and coworkers (Roth *et al.*, 1981; Roth *et al.*, 1987) (numbers 1 - 55) were used in an ILP study (King *et al.*, 1992); 43 of the 44, and another 25 were used in a more recent linear regression study (Selassie *et al.*, 1991) (numbers 2 - 44, and numbers 56 - 74) and in a subsequent neural network analysis (So and Richards, 1992). Six of the 25 have not been included in this comparison, because the complete set of Hansch parameters for the substituents were not available at the time of this study. Biological activities had been measured by the association constant to DHFR from MB1428 *E. coli* (Li *et al.*, 1982). The substituents of the phenyl ring vary in the 3-, 4- and 5-positions. Not only has this reasonably large data set been extensively studied by QSAR methods, but there are also crystallographic studies of the complex formed between trimethoprim (2,4-diamino-5-(3,4,5-trimethoxybenzyl) pyrimidine) and DHFR from *E. coli* (Champness *et al.*, 1986; Matthews *et al.*, 1985). It is possible therefore, in this test case,

Index	Activity	Substituent		
No.	($\log K_i$)	3 -	4 -	5 -
01	3.04	OH	H	OH
02	5.60	H	$O(CH_2)_6CH_3$	H
03	6.07	H	$O(CH_2)_5CH_3$	H
04	6.18	H	H	H
05	6.20	H	NO_2	H
06	6.23	F	H	H
07	6.25	$O(CH_2)_7CH_3$	H	H
08	6.28	CH_2OH	H	H
09	6.30	H	NH_2	H
10	6.31	CH_2OH	H	CH_2OH
11	6.35	H	F	H
12	6.39	$O(CH_2)_6CH_3$	H	H
13	6.40	H	$OCH_2CH_2OCH_3$	H
14	6.45	H	Cl	H
15	6.46	OH	OH	H
16	6.47	OH	H	H
17	6.48	H	CH_3	H
18	6.53	$OCH_2CH_2OCH_3$	H	H
19	6.55	$CH_2O(CH_2)_3CH_3$	H	H
20	6.57	OCH_2CONH_2	H	H
21	6.57	H	OCF_3	H
22	6.59	CH_2OCH_3	H	H

Table 4.1. Pyrimidines used in this study.

23	6.65	Cl	H	H
24	6.70	CH_3	H	H
25	6.78	H	$N(CH_3)_2$	H
26	6.82	H	Br	H
27	6.82	H	OCH_3	H
28	6.82	$O(CH_2)_3CH_3$	H	H
29	6.86	$O(CH_2)_5CH_3$	H	H
30	6.89	H	$O(CH_2)_3CH_3$	H
31	6.89	H	$NHCOCH_3$	H
32	6.92	OSO_2CH_3	H	H
33	6.93	OCH_3	H	H
34	6.96	Br	H	H
35	6.97	NO_2	$NHCOCH_3$	H
36	6.99	$OCH_2C_6H_5$	H	H
37	7.02	CF_3	H	H
38	7.22	$OCH_2CH_2OCH_3$	$OCH_2CH_2OCH_3$	H
39	7.23	I	H	H
40	7.69	CF_3	OCH_3	H
41	7.72	OCH_3	OCH_3	H
42	8.35	OCH_3	$OCH_2CH_2OCH_3$	OCH_3
43	8.38	OCH_3	H	OCH_3
44	8.87	OCH_3	OCH_3	OCH_3
45	7.56	CH_3	OH	CH_3
46	7.74	CH_3	OCH_3	CH_3
47	7.87	OCH_3	$O(CH_2)_5CH_3$	OCH_3
48	7.87	OCH_3	$O(CH_2)_7CH_3$	OCH_3

Table 4.1 (cont.)

49	8.42	OCH_3	$OCH_2C_6H_5$	OCH_3
50	8.57	OCH_3	CH_3	OCH_3
51	8.82	I	OCH_3	I
52	8.82	I	OH	I
53	8.85	Br	NH_2	Br
54	8.87	Cl	NH_2	Cl
55	8.87	Cl	NH_2	CH_3
56	6.45	H	OH	H
57	6.60	H	OSO_2CH_3	H
58	6.84	OH	OCH_3	H
59	6.89	H	$OCH_2C_6H_5$	H
60	6.93	H	C_6H_5	H
61	7.04	CH_3	H	CH_3
62	7.13	OCH_2O	H	OCH_2O
63	7.16	$O(CH_2)_7CH_3$	OCH_3	H
64	7.20	OCH_3	$O(CH_2)_7CH_3$	OCH_3
65	7.41	OC_3H_7	H	OCH_3H_7
66	7.53	OCH_3	$OCH_2C_6H_5$	H
67	7.54	OCH_3	OH	H
68	7.66	$OCH_2C_6H_5$	OCH_3	H
69	7.71	OCH_3	$N(CH_3)_2$	OCH_3
70	7.77	OCH_3	$OCH_2CH_2OCH_3$	H
71	7.80	OSO_2CH_3	OCH_3	H
72	7.82	CH_2CH_3	CH_2CH_3	CH_2CH_3
73	7.94	OCH_3	OSO_2CH_3	H
74	8.18	OCH_3	Br	OCH_3

Table 4.1 (cont.)

to compare the QSAR models with the X-ray stereochemistry of interaction, although it must be stressed that the QSAR methods examined here are for application to drug design problems where there is no receptor structure. The structure information about DHFR is only used to evaluate the interpretive results of the methods, and is not used in anyway for the construction of the QSARs.

The QSAR methods studied in this chapter have been assessed using five-fold cross-validation, in which the 55 drugs (numbers 1 - 55) from the machine learning study (King *et al.*, 1992) were randomly divided into five equal sets each containing 11 drugs. Each of these sets were used as a test set, with other four sets forming corresponding training sets, each of 44 drugs. Each of the 55 pyrimidines, in the cross-validation study appears once only in only one of the test sets, so all the drugs are tested. The 19 more recently characterised derivatives (Selassie *et al.*, 1991) (numbers 56 - 74), were used as an additional test set. This division of the data was chosen to maintain consistency with the earlier QSAR studies (Hansch *et al.*, 1982; King *et al.*, 1992), and allows further assessment of this work. The training and test sets are given in Table 4.2.

4.3.2 Hansch parameters

Linear regression studies (Hansch *et al.*, 1982; Li and Poe, 1988; Selassie *et al.*, 1991) have correlated the activity of the pyrimidines to the chemical properties of the 3-, 4-, and 5-substituents of the phenyl ring. The activity was measured by log

$1/K_i$, where K_i is the inhibition constant as experimentally assayed (Dietrich *et al.*, 1980; Li *et al.*, 1982; Roth *et al.*, 1981; Roth *et al.*, 1987). The chemical properties of the substituents were represented by the hydrophobic parameter, π and the molar refractivity, MR, where π is derived from partition coefficients between 1-octanol and water (Leo *et al.*, 1971), and MR is related to the size of the substituent.

$$\pi_X = \log P_X - \log P_H \ , \qquad \text{[eqn. 4.1]}$$

where P_X is the octanol/water partition coefficient of a derivative, and P_H is that of the parent compound.

$$MR = \frac{MW(n^2 - 1)}{d(n^2 + 2)} \ , \qquad \text{[eqn. 4.2]}$$

where n is the refractive index, MW the molecular weight, and d the density of the substance. MR is also related to polarisability, α, by:

$$MR = \frac{4\pi N\alpha}{3} \ , \qquad \text{[eqn. 4.3]}$$

where N is Avogadro's number. Previous analyses (Hansch *et al.*, 1982; King *et al.*, 1992; Selassie *et al.*, 1991; So and Richards, 1992), have used truncated MR values, which have an upper limit of 0.79. To maintain consistency, only truncated MR values have been used in this study. There are other physicochemical properties which also may be used in a QSAR (Hansch and Leo,

Split number	index numbers[a]
set 1	11, 31, 34, 42, 20, 24, 30, 23, 37, 08, 39
set 2	16, 54, 43, 19, 22, 10, 41, 33, 35, 01, 09
set 3	14, 03, 55, 06, 04, 47, 28, 25, 27, 50, 07
set 4	51, 49, 53, 46, 12, 21, 44, 52, 38, 15, 18
set 5	36, 45, 26, 32, 02, 17, 40, 13, 48, 29, 05
set 6	56 - 74
train set 1	sets 2, 3, 4, 5
train set 2	sets 1, 3, 4, 5
train set 3	sets 1, 2, 4, 5
train set 4	sets 1, 2, 3, 5
train set 5	sets 1, 2, 3, 4

[a]The numbers in this column correspond to those in column 1 of Table 4.1.

Table 4.2. Training and testing sets

1979). The hydrophobic constants and molar refractivities for all the substituents in this study are listed in Table 4.3.

4.3.3 Physicochemical attributes (PCAs)

The representation of data is an important issue in any data analysis, regardless of the method used, and QSAR is no exception. One way to describe molecules is to use graph theory to represent atomic connectivity by set of attributes (Elrod *et al.*, 1990; Kvasnicka *et al.*, 1993; Luce and Govind, 1990; Ugi *et al.*, 1979), which may also include some physicochemical descriptors. However, in QSAR, molecules are generally described in lower resolution, by grosser properties of substituents as embodied in the Hansch parameters. These parameters encode much structural information. For example, lipophilicity, the logarithm of partition coefficients, can be decomposed into a volume term and terms reflecting electrostatic interactions such as polarisability, hydrogen-bond donor acidity and hydrogen-bond acceptor basicity (El Tayar *et al.*, 1992).

The PCA representation has been developed, by adapting an approach from machine learning (King *et al.*, 1992), to describe explicitly some of these volume and electrostatic interactions. The substituents of the drugs were described by nine PCAs assigned heuristically: the polarity (the amount of residual charge on the α and β atoms of the substituent), size, flexibility, the number of hydrogen-bond donors and the number of hydrogen-bond acceptors, the presence and strength of π-acceptors and π-donors, the polarisability of the molecular

Substituent	hydrophobic parameter, π	truncated molar refractivity, MR
OCH_2CONH_3	-1.37	0.79
NH_2	-1.23	0.54
CH_2OH	-1.03	0.72
$NHCOCH_3$	-0.97	0.79
OSO_2CH_3	-0.88	0.79
CH_2OCH_3	-0.78	0.79
OH	-0.67	0.28
$OCH_2CH_2OCH_3$	-0.40	0.79
NO_2	-0.28	0.74
OCH_3	-0.02	0.79
H	0.00	0.10
F	0.14	0.10
$N(CH_3)_2$	0.18	0.79
CH_3	0.56	0.57
Cl	0.71	0.60
$CH_2O(CH_2)_4CH_3$	0.84	0.79
Br	0.86	0.79
CF_3	0.88	0.79
OCF_3	1.04	0.79
I	1.12	0.79
$O(CH_2)_4CH_3$	1.55	0.79
$OCH_2C_6H_5$	1.66	0.79

Table 4.3. Hansch parameters of the substituents.

$O(CH_2)_5CH_3$	2.63	0.79
$O(CH_2)_6CH_3$	3.17	0.79
$O(CH_2)_7CH_3$	3.71	0.79
CH_2CH_3	0.86	0.79
OC_3H_7	1.05	0.79
OCH_2O	0.00	0.45
C_6H_5	0.00	0.79

Table 4.3 (cont.)

orbitals, and the σ-effect. Each property of each substituent was represented by an integer value; originally, the PCA representation was designed for the inductive logic program (GOLEM) (Muggleton and Feng, 1990), which uses integer values. The PCAs have been rescaled to lie between 0.1 and 0.9 for the neural network, linear regression and nearest neighbour analyses. The attributes are not calculated automatically, which is a limitation of the representation to be addressed in further work. The pyrimidines are substituted at a maximum of three positions, and so each drug is represented by 27 PCAs (three positions and nine PCAs per position). The attributes assigned to 114 different fragments, used in this chapter and the subsequent chapter, are given in Table 4.4.

4.3.4 Linear regression

The Hansch parameters π_3, π_4, π_5, MR_3, MR_4 and MR_5 were assigned to the drugs in the five cross-validation training sets. To provide a benchmark on the data in this study, a stepwise linear regression was performed on these variables and their squares; the regression equation was derived fully automatically, using the STEP command in Minitab release 7.2 VAX/VMS version (C) copyright (1989) Minitab, Inc. Stepwise regression, which is based on the principle of least squares, is used to identify a useful subset from a collection of predictor variables, using the maximum F-statistic criterion. The F-statistic is the ratio of the variance of one sample to the variance of another sample (see Press *et al.*, 1992 for detailed discussion).

Group	PL	SZ	FL	HD	HA	πD	πA	PO	σ	BR[a]
$(CH_2)_2$	1	2	2	0	0	0	0	1	0	0
$(CH_2)_2COCH_2Cl$	1	3	3	0	1	0	0	1	0	1
$(CH_2)_2CON(CH_2CH_2)_2O$	1	6	2	0	1	0	0	1	0	1
$(CH_2)_2CONEt_2$	1	5	2	0	1	0	0	1	0	2
$(CH_2)_2CONMe_2$	1	4	2	0	1	0	0	1	0	2
$(CH_2)_3CH(CH_2NHCOCH_2Br)$	1	6	6	0	1	0	0	1	0	0
$(CH_2)_3O$	1	4	4	0	1	0	0	1	0	0
$(CH_2)_4$	1	4	4	0	0	0	0	1	0	0
$(CH_2)_4COCH_2Cl$	1	5	5	0	1	0	0	1	0	1
$(CH_2)_6$	1	4	6	0	0	0	0	1	0	0
$(CH_2)_2CON(C_3H_7)_2$	2	5	3	0	1	0	0	1	0	2
Br	3	1	0	0	0	1	0	2	2	0
C_6H_5	0	3	0	0	0	2	0	3	0	1
CF_3	3	1	0	0	0	0	1	0	3	1
$CH(CH_2NHCOCH_2Br)CH_2$	2	5	2	0	1	0	0	1	0	2
CH_2	1	1	0	0	0	0	0	1	0	0
CH_2CH_3	1	2	2	0	0	0	0	1	0	0
CH_2CN	2	2	1	0	1	0	0	1	0	0
$CH_2CON(CH_2CH_2)_2O$	2	5	1	0	2	0	0	1	0	2

Table 4.4. Physicochemical attributes (PCAs) of fragments: polarity (PL), size (SZ), flexibility (FL), number of hydrogen-bond donors (HD), number of hydrogen-bond acceptors (HA), strength and presence of π-donors (πD) and π-acceptors (πA), polarisability (PO), σ-effect (σ), and branching (BR)

CH_2CONEt_2	2	4	3	0	1	0	0	1	0	2
CH_2CONMe_2	2	3	2	0	1	0	0	1	0	2
$CH_2N(Me)CO(CH_2)_2$	2	6	4	0	1	0	0	1	0	2
$CH_2N(Me)COCH_2$	2	5	3	0	1	0	0	1	0	2
$CH_2NHCOCH_2Br$	2	4	3	1	1	0	0	1	0	1
$CH_2NHCONEt_2$	2	5	3	1	2	0	0	1	0	2
$CH_2NHCON(CH_2CH_2)_2O$	2	5	1	1	3	0	0	1	0	2
CH_2O	2	2	2	0	1	0	0	1	0	0
$CH_2O(CH_2)_3CH_3$	1	4	6	0	1	0	0	1	0	0
CH_2OCH_3	1	2	3	0	1	0	0	1	0	0
CH_2OH	2	2	2	2	2	0	0	1	0	0
CH_3	1	1	0	0	0	0	0	1	0	0
Cl	3	1	0	0	0	1	0	1	3	0
CN	4	1	0	0	1	0	2	2	0	0
$COCH_2Cl$	3	2	1	0	1	0	1	1	0	1
$CON(CH_2)_4$	3	4	0	0	1	0	1	1	0	2
$CON(CH_2)_5$	3	5	0	0	1	0	1	1	0	2
$CON(CH_2CH_2)_2O$	3	5	0	0	2	0	1	1	0	2
$CONEt_2$	3	4	0	0	1	0	1	1	0	2
$CONH(CH_2)_2$	3	3	2	1	1	0	1	1	0	1
$CONH(CH_2)_4O$	3	5	4	1	2	0	1	1	0	1
$CONHCH_2$	3	2	1	1	1	0	1	1	0	1
$CONMe_2$	3	3	0	0	1	0	1	1	0	2
F	5	1	0	0	1	0	0	0	5	0
H	1	0	0	0	0	0	0	0	0	0

Table 4.4 (cont.)

I	2	1	0	0	0	1	0	3	1	0
$N(CH_3)_2$	1	2	0	0	1	2	0	1	1	1
$N(CH_3)COCH_2O$	2	4	2	0	2	2	0	1	1	2
$N(Me)CO(CH_2)_2$	2	4	2	0	1	2	0	1	1	2
$N(Me)COCH_2$	2	3	1	0	1	2	0	1	1	2
NH_2	2	1	0	2	0	2	0	0	1	0
NHCO	2	2	0	1	1	2	0	1	1	2
$NHCO(CH_2)_2$	2	3	2	1	1	1	0	1	1	1
$NHCO(CH_2)_2S$	2	4	3	1	1	1	0	1	1	1
$NHCO(CH_2)_3$	2	4	3	1	1	1	0	1	1	1
$NHCO(CH_2)_3O$	2	5	4	1	2	1	0	1	1	1
$NHCO(CH_2)_4O$	2	5	5	1	1	1	0	1	1	1
$NHCOCH(\alpha\text{-}C_{10}H_7)CH_2$	2	8	1	1	1	1	0	1	1	1
$NHCOCH(CH_2CH_2Ph)CH_2$	2	8	1	1	1	1	0	1	1	3
$NHCOCH(CH_3)O$	2	4	2	1	2	1	0	1	1	2
$NHCOCH(Me)CH_2$	2	4	2	1	1	1	0	1	1	2
$NHCOCH(Ph)CH_2$	2	6	2	1	1	1	0	1	1	3
$NHCOCH(Ph\text{-}2''\text{-}CH_3)CH_2$	2	6	1	1	1	1	0	1	1	4
$NHCOCH(Ph\text{-}2''\text{-}OCH_3)CH_2$	2	7	1	1	2	1	0	1	1	4
$NHCOCH(Ph\text{-}3''\text{-}CH_3)CH_2$	2	6	1	1	1	1	0	1	1	4
$NHCOCH(Ph\text{-}3''\text{-}OCH_3)CH_2$	2	7	1	1	2	1	0	1	1	4
$NHCOCH(Ph\text{-}4''\text{-}CH_3)CH_2$	2	6	1	1	1	1	0	1	1	4
$NHCOCH_2$	2	3	1	1	1	1	0	1	1	1
$NHCOCH_2Br$	2	3	1	1	1	1	0	1	1	1
$NHCOCH_2CH(CH_3)$	2	3	2	1	1	1	0	1	1	2

Table 4.4 (cont.)

$NHCOCH_2CH(Ph)$	2	6	2	1	1	1	0	1	1	3
$NHCOCH_2CH(Ph)CH_2$	2	6	3	1	1	1	0	1	1	3
$NHCOCH_2O$	2	3	2	1	2	1	0	1	1	1
$NHCOCH_2S$	2	3	2	1	1	1	0	2	1	1
$NHCOCH_3$	2	2	0	1	1	1	0	1	1	0
NHCOCHCH	2	3	0	1	1	1	0	1	1	1
$NHCONH(CH_2)_2O$	2	5	3	1	3	1	0	1	1	1
$NHCONH(CH_2)_3O$	2	6	4	2	3	1	0	1	1	1
$NHCONH(CH_2)_4O$	2	6	5	2	3	1	0	1	1	1
$NHCONHCH_2$	2	4	1	2	2	1	0	1	1	1
NO_2	5	2	0	0	0	0	2	0	3	1
O	2	1	0	0	1	1	0	2	1	0
$O(CH_2)_2$	2	3	2	0	1	1	0	1	1	0
$O(CH_2)_2O$	2	3	3	0	2	1	0	1	1	0
$O(CH_2)_2O(CH_2)_2O$	2	4	6	0	3	1	0	1	1	0
$O(CH_2)_3CH_3$	2	3	4	0	1	1	0	1	1	0
$O(CH_2)_3O$	2	3	4	0	2	1	0	1	1	0
$O(CH_2)_4$	2	4	4	0	1	1	0	1	1	0
$O(CH_2)_4O$	2	4	5	0	2	1	0	1	1	0
$O(CH_2)_5CH_3$	2	5	6	0	1	1	0	1	1	0
$O(CH_2)_5O$	2	5	6	0	2	1	0	1	1	0
$O(CH_2)_6O$	2	5	7	0	2	1	0	1	1	0
$O(CH_2)_6CH_3$	2	5	7	0	1	1	0	1	1	0
$O(CH_2)_7CH_3$	2	5	8	0	1	1	0	1	1	0
OC_3H_7	2	4	3	0	1	1	0	1	1	0

Table 4.4 (cont.)

OCF_3	4	2	0	0	0	0	2	0	3	1
$OCH_2C_6H_{10}CH_2O$	2	6	2	0	1	1	0	1	1	1
$OCH_2C_6H_5$	2	4	2	0	1	1	0	1	1	1
$OCH_2CH_2OCH_3$	2	3	4	0	2	1	0	1	1	0
OCH_2CH_3	2	3	2	0	1	1	0	1	1	0
$OCH_2CON(CH_2)_4$	2	5	1	0	2	1	0	0	1	2
$OCH_2CON(CH_2)_5$	2	5	1	0	2	1	0	0	1	2
$OCH_2CON(CH_2CH_2)_2O$	2	5	2	0	3	1	0	0	1	2
OCH_2CONMe_2	2	4	1	0	2	1	0	0	1	1
$OCH_2CONMePh$	2	6	1	0	2	1	0	0	1	1
OCH_2CONEt_2	2	5	2	0	2	1	0	0	1	2
OCH_2CONH_2	2	3	1	1	2	1	0	0	1	1
OCH_3	2	2	0	0	1	1	0	1	1	0
OCH_2O	2	3	3	0	2	1	0	1	1	0
OH	3	1	0	2	2	2	0	1	2	0
$ON(CH_3)COCH_2O$	3	4	2	0	3	1	0	1	2	2
OSO_2CH_3	4	3	1	0	0	0	1	2	2	2
SO_2F	4	2	0	0	0	0	1	2	2	2
$SO_2NH(CH_2)_2$	4	4	2	0	1	0	1	2	2	2
SO_2NMe_2	4	4	0	0	1	0	1	2	2	3

[a]The PCA representing branching was not used in this study

Table 4.4 (cont.)

More complex relationships have been derived from the consideration of thousands of possible correlation equations. The equation:

$$\log(1/K_i) = 0.75\pi_{3,4,5} + 1.36MR_{3,5} + 0.88MR_4 - 1.07\log_{10}(\beta.10^{\pi_{3,4,5}} + 1) + 6.20, \qquad \text{[eqn. 4.4]}$$

where $\log_{10}\beta = 0.12$, $\pi_{3,4,5}$ is the sum of π_3, π_4 and π_5, $MR_{3,5}$ is the sum of MR_3 and MR_5, and all the MR values are truncated, was derived by Hansch *et al.* (1982), and used as benchmark in the machine learning study (King *et al.*, 1992). This equation was not automatically determined, and it is therefore infeasible to determine equations of this form for each split of the data here. However, the performance of this equation has been evaluated on the training and testing data, but the results are not directly comparable, because it was derived from a different training set. Stepwise linear regression analyses were also done using the PCA representation of the drugs; one regression used the 27 attributes and the other used the 27 attributes and their squares, analogous to the approach using the Hansch parameters.

4.3.5 Nearest neighbour

A simple nearest neighbour algorithm was written in the programming language C, under UNIX, on a SUN Sparc workstation. Each drug in the data set is represented by an n-dimensional vector, where, in this case, n is either 6, when the drugs are described by the Hansch parameters π_3, π_4, π_5, MR_3, MR_4, and MR_5, or n is 27, when the drugs are described by the

PCA representation. The activity of a drug in the test set is predicted to be the activity of the closest drug in the training set, where the "closeness" is the Euclidean distance between the vectors that represent the drugs, *i.e.*, the minimum distance, *d*, is found, where,

$$d = \{\sum (\mathrm{train}[i] - \mathrm{test}[i])^2\}^{1/2}, \qquad \text{[eqn. 4.5]}$$

with train[*i*] and test[*i*] denoting the vectors representing a drug in the training set and a drug in the test set.

4.3.6 Neural networks

The backpropagation of errors algorithm (Rumelhart *et al.*, 1986a), has been implemented here, as in the previous neural network study of the QSAR between triazines and DHFR (Andrea and Kalayeh, 1991), using an approach to increase the speed of learning of the neural network (Owens and Filkin, 1989), where the changes to the weights are calculated by solving a set of stiff differential equations (Gear, 1971). Code was written in FORTRAN to simulate the neural network, incorporating the original code of Gear (Gear, 1971), and was implemented on a VAX 4600, and also on a SUN Sparc workstation. The algorithm is explained in detail in Chapter 2, and some of the code is given in Appendix 2.

The neural network methodology has several parameters that are determined empirically, and which may vary from application to application. This is perhaps a weakness of the approach, and in this work, one of these empirically determined parameters - the convergence criterion - has been addressed. The predictive performance of a neural network on unseen test data can be sensitive to when the learning phase of the neural network is terminated. The problem of memorisation is well documented and occurs when the neural network is overtrained (discussed in section 1.6.7). If the convergence criterion is too stringent, *i.e.*, training is continued until the change in the error on the training set is too small, then the performance on the test set will be impaired, although the training set performance may increase a little. The neural network can fail to generalise well to unseen data, even though it has modelled training data well. Thus, method is required that, *a priori*, gives the convergence criterion, without examination of the test data.

In this work, the convergence criterion has been determined by the performance of the neural network on a third set of data. The data were divided into three sets: training, monitoring, and testing. The neural network was trained directly on the training data, and the test data remained unseen during the learning phase. The performance of the neural network, in terms of generalisation to unseen data, was monitored using the monitor set. Throughout the learning phase the performance of the neural network on the monitor set was calculated; training was stopped when the performance on the monitor set was maximum.

The neural network was trained using both representations of the data. In an approach similar to that of So and Richards (1992), a neural network with six input units, a varying number of hidden layer units and one output unit was trained to predict the activity of drugs given $\pi_3, \pi_4, \pi_5, MR_3, MR_4, MR_5$. Previously (Hansch *et al.*, 1982; So and Richards, 1992), a slightly different set of Hansch parameters were used, but the most straightforward set has been used here, for benchmarking purposes - both sets give similar performances (results not shown). There are a few minor differences in implementation of the neural network. Here, a speed-up algorithm (Owens and Filkin, 1989) has been used; a monitor set was used to determine when to stop training; and different splits of the data were used. The data were split as for the other methods, with five-fold cross-validation on 55 drugs (44 drugs in the training set and 11 in the test set) and further independent test set of 19 drugs. For each of the 30 drugs in a test set (11 from the cross-validation trial and 19 from the independent test), one was used as the completely unseen test set, and 29 were used to monitor the neural network. This was repeated for each drug in the test set in turn. This procedure meant that the neural network was only trained once per test set of 30, rather than being trained 30 times. This use of some of the test set for monitoring is legitimate, because the activity of the test drug is not used in training, but the performance of the neural network may be slightly over-estimated compared to the other methods. A neural network with 27 input units, a varying number of hidden units and one output unit (Figure 4.2) was trained on the same data using the PCA representation, with the same monitoring procedure.

4.3.7 Inductive logic programming

The work discussed in this section and the following section (4.3.8) was done by Dr. Ross King. GOLEM is a machine learning program that uses ILP. The ILP methodology (Muggleton and Feng, 1990) is, in theory, well suited for drug design problems as it is designed specifically to learn relationships between objects, *e.g.*, molecular structures. In ILP, a subset of predicate calculus is used to express learned rules. This language is expressive enough to describe most mathematical concepts and has a strong link with natural language, leading to ease of comprehension. GOLEM is written in the programming language C, but implements predicate logic in the language Prolog. The performance of ILP is dependent on the problem; if the problem is not particularly structural other machine learning methods may be expected to do as well or better (King *et al.*, 1993).

QSAR is generally a regression problem, in which a real number is predicted from the description of a compound. However, GOLEM is designed to carry out classification (discrimination) tasks in which a small number of discrete classes are predicted. To overcome this difficulty pairs of drugs are considered, and their activities compared. The output is a set of rules that predicts which of a pair of drugs has higher activity. Paired comparisons are then converted to a ranking by the method of David (David, 1987).

GOLEM takes three types of facts: positive, negative, and background, as input. The positive facts are the paired examples

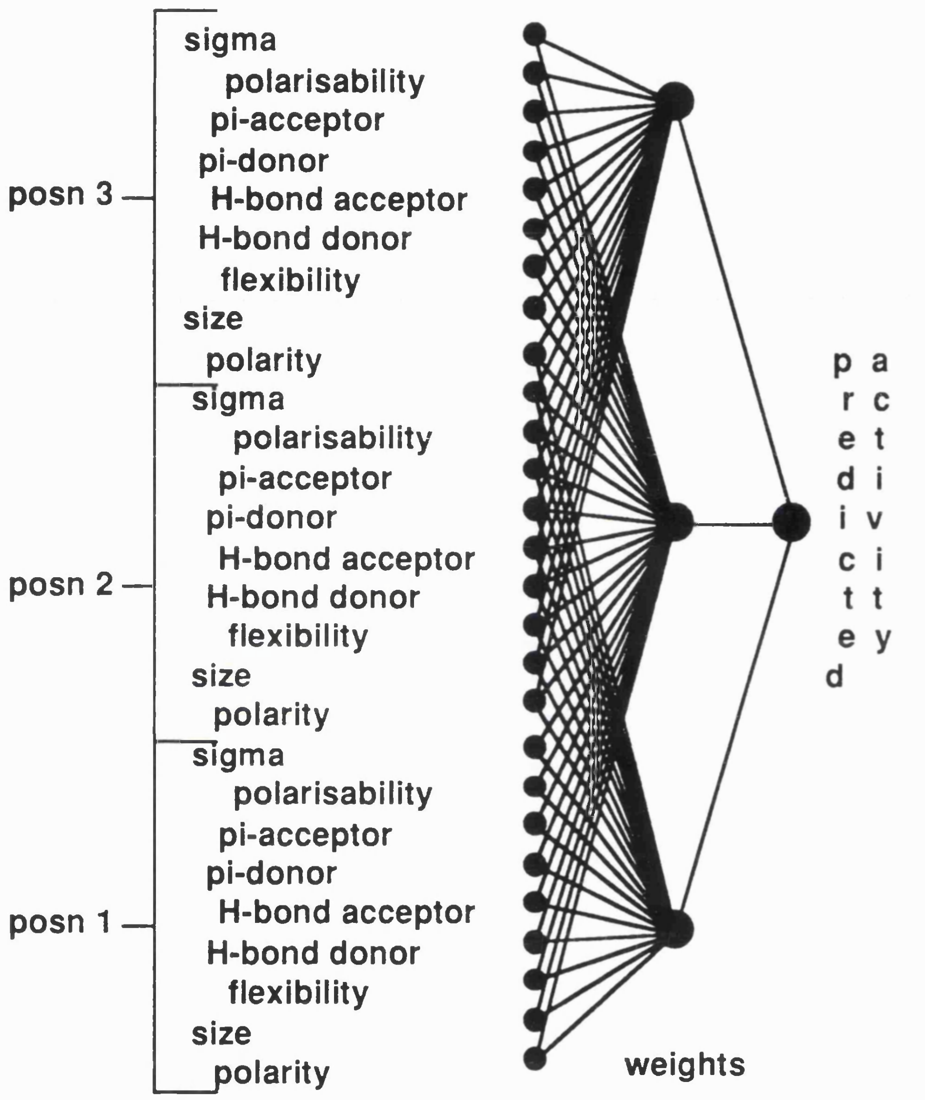


Figure 4.2 Schematic representation of the neural network trained using the PCA representation. All the weights are not shown, but each input unit is connected to each of the three hidden units by a weight.

of greater activity, *e.g.*, great(d50, d9), which states that drug number 50 has higher activity than drug number 9. The paired examples of lower activity, *e.g.*, great(d9, d50), are negative facts (or false statements). GOLEM requires both positive and negative facts to give balanced generalisation.

The background facts are the chemical structures of the drugs and the properties of the substituents. Chemical structure is represented in the form: struc(d35, NO_2, $NHCOCH_3$, H). This Prolog representation of drug number 35 states that the drug has NO_2 substituted at position 3, $NHCOCH_3$ substituted at position 4, and no substituent at position 5. By convention, if only one of positions 3 and 5 is substituted, as in drug number 35, the position with no substitution is assumed to be position 5.

The representation used by GOLEM is similar to that described by King *et al.* (1992) for the modelling of the QSAR of trimethoprim analogues binding to DHFR. The PCA representation was used, with different predicates for each attribute. For example, polar(Cl, polar3) states that Cl has polarity of value 3. Information was also given about the relative values of the properties for the substituent group, *e.g.*, great_polar(polar3, polar2) states that a polarity of 3 is greater than a polarity of 2. This information has to be provided because arithmetic is not encoded into GOLEM.

For the five fold cross-validation study, involving 55 drugs, there were 2198 background facts, 1394 positive facts, and 1394 negative facts. In the separate test set of 19 drugs, there were

2388 background facts (a super-set of the previous 2198 with the addition of the new groups), 965 positive facts, and 965 negative facts. Rules were selected sequentially, using the "minimal description length" principle to avoid overfitting the data (as implemented in the compression model (Muggleton *et al.*, 1992)). The following parameter settings for GOLEM were used: depth, i = 5; clause parameter, j = 5; error level, *noise* = 2; sample size, *rlggsample* = 8 (as defined in the original GOLEM work (Muggleton and Feng, 1990)).

4.3.8 Decision tree

Classification And Regression Trees, CART, is a collection of binary decision tree growing algorithms for use in classification (discrimination) and regression (Breiman *et al.*, 1984). To compare directly a propositional learning algorithm (CART) with a predicate logic algorithm (GOLEM), CART was used in its most common form as a binary classification tree generator. A binary classification tree consists of nodes, and each node has two branches. There is a single test (or decision) on each node, splitting the node into two subtrees. Depending on whether the result of a test is true or false, the tree will branch to left or right, and the splitting process continues recursively. At each leaf node a decision is made on the class assignment.

Only the PCA representation of the data was used, with each drug described by the 27 attributes. Paired comparisons were then created, as described for GOLEM. This produced training examples with 54 attributes, which belonged to class 1 if the first drug was more active than the second, and to class 0 if the second drug was more active than the first. For CART the following settings were used: the Gini splitting criteria, ten fold cross validation, and the 0-SE rule for tree pruning. The version of CART used was INDCART (a free version of CART supplied by Wray Buntine of NASA's Ames Laboratories).

4.4 Results

The performances of the methods on the cross-validation training data and testing data, as measured by the Spearman rank correlation coefficient, are given in Tables 4.5 and 4.6 respectively. The Spearman rank correlation coefficient, r_s, is given by:

$$r_s = \frac{1 - \frac{6}{N^3 - N}\left[D + \frac{1}{12}\sum_k (f_k^3 - f_k) + \sum_m (g_m^3 - g_m)\right]}{\left[1 - \frac{\sum_k (f_k^3 - f_k)}{N^3 - N}\right]^{1/2}\left[1 - \frac{\sum_m (g_m^3 - g_m)}{N^3 - N}\right]^{1/2}} \qquad \text{[eqn.4.6]}$$

where, for N pairs of measurements (x_i, y_i), D is the sum of the squares of the differences in ranks, f_k is the number of ties in the kth group of ties among the ranks of the x_i's, and g_m is the number of ties in the mth group of ties among the ranks of the y_i's. The Spearman rank correlation coefficient allows a direct comparison between all the methods. There are five sets of test data, each with 11 drugs. Each of the 55 pyrimidines, in the cross-validation study appears once only in only one of the test sets, so all the drugs are tested. The respective training sets comprise the other four sets of cross-validation data. Each of these cross-validation training sets has also been tested with the independent test set of 19 drugs (Table 4.7).

4.4.1 Linear regression

For each representation of the data, five regression equations have been derived - one for each set of training data. To summarise the results, these have been combined to give representative correlation equations, in which contains the mean values of the coefficients of any variable that appears in more than one of the five regression equations. For each representative correlation equation, n is the number of training set examples, r^2 is the mean of the Pearson correlation coefficient for the five cross-validation training sets and σ is the mean of the standard deviation from the regression (also known as the residual standard deviation) for the five cross-validation training sets - r^2 and σ are mean values, which do not characterise the representative correlation equation. The stepwise linear regression on the Hansch parameters and their squares yields the following representative correlation equation:

$$\log(1/K_i)_N = 0.32 - 1.62MR_5 + 0.84\pi_5 + 0.09MR_3{}^2 + 0.05MR_4{}^2 + 1.09MR_4{}^2 - 0.20\pi_4{}^2 - 0.23\pi_5{}^2. \quad \text{[eqn. 4.7]}$$

$n = 44$; $r^2 = 0.85$; $\sigma = 0.06$,

Method	Set 1	Set 2	Set 3	Set 4	Set 5	Mean[a] (σ)
LR on Hansch + squares[b]	0.883	0.832	0.846	0.830	0.773	0.833 (0.035)
LR on PCAs + squares	0.941	0.922	0.899	0.881	0.796	0.888 (0.050)
nearest neighbour on Hansch parameters	1.000	1.000	1.000	1.000	1.000	1.000 (0.000)
nearest neighbour on PCAs	1.000	1.000	1.000	1.000	1.000	1.000 (0.000)
neural network on Hansch parameters	0.835	0.886	0.937	0.781	0.759	0.840 (0.066)
neural network on PCAs	0.926	0.906	0.937	0.834	0.866	0.894 (0.038)
CART on PCAs	0.990	0.980	0.981	0.980	0.980	0.982 (0.004)
GOLEM on PCAs	0.966	0.929	0.952	0.952	0.943	0.948 (0.012)

[a]Each method was trained on the five cross-validation training sets. The mean and the standard deviation (σ) of the five performances are given.

[b]Linear regression (LR) on the Hansch parameters and their squares.

Table 4.5 Cross-validation training set performances as measured by the Spearman rank correlation coefficients.

Method	Set 1	Set 2	Set 3	Set 4	Set 5	Mean[a] (σ)
LR on Hansch + squares[b]	0.717	0.478	0.521	0.845	0.905	0.693 (0.170)
LR on PCAs + squares	0.654	0.506	0.694	0.819	0.596	0.654 (0.104)
nearest neighbour on Hansch parameters	0.700	0.694	0.879	0.814	0.724	0.762 (0.073)
nearest neighbour on PCAs	0.405	0.336	0.532	0.741	0.729	0.549 (0.165)
neural network on Hansch parameters	0.387	0.582	0.497	0.720	0.773	0.592 (0.142)
neural network on PCAs	0.702	0.418	0.843	0.784	0.836	0.717 (0.158)
CART on PCAs	0.247	0.055	0.774	0.772	0.645	0.499 (0.294)
GOLEM on PCAs	0.751	0.574	0.753	0.757	0.627	0.692 (0.077)

[a]Each method was trained on the five cross-validation training sets. The mean and the standard deviation (σ) of the five performances are given.

[b]Linear regression (LR) on the Hansch parameters and their squares.

Table 4.6 Cross-validation test set performances as measured by the Spearman rank correlation coefficients.

Method	Mean performance[a] (σ)
linear regression on Hansch parameters + squares	0.657 (0.036)
linear regression on PCAs + squares	0.509 (0.152)
nearest neighbour on Hansch parameters	0.655 (0.063)
nearest neighbour on PCAs	0.497 (0.074)
neural network on Hansch parameters	0.634 (0.041)
neural network on PCAs	0.530 (0.037)
CART on PCAs	0.536 (0.157)
GOLEM on PCAs	0.738 (0.095)

[a]Each method was trained on the five cross-validation training sets. For each training set, a corresponding performance on the independent test set of 19 drugs was calculated. The mean and the standard deviation (σ) of the five performances are shown here.

Table 4.7 The mean Spearman rank correlation coefficients on the independent test set of 19 drugs.

where $\log(1/K_i)_N$ is $\log(1/K_i)$ rescaled to lie between 0.1 and 0.9. The Hansch parameters and PCAs were similarly rescaled. Rescaling maintained consistency with the neural network analysis, which required inputs between 0.1 and 0.9. The more complex equation of Hansch *et al.* (1982) gave an average cross-validation test set Spearman rank correlation coefficient of 0.859 and on the independent test set 0.794, but these results are not directly comparable because some of the test data was used in the derivation of the equation.

The correlation equation derived from a linear regression on the PCAs gave relatively poor predictions, which was improved by the inclusion of the squares of the PCAs in the regression. Linear regression on the 27 PCAs and their squares gives:

$$\begin{aligned}\log(1/K_i)_N = {} & 0.44 + 0.66PO_3 - 0.82PO_3^2 + 0.11\pi A_4 \\ & + 0.79SZ_5 - 0.94FL_5 + 1.10\pi A_5 - 0.18\pi A_5^2 - \\ & 0.12SZ_3^2 - 0.05FL_3^2 - 0.17SZ_4^2 \qquad \text{[eqn. 4.8]}\end{aligned}$$

$n = 44$; $r^2 = 0.93$; $\sigma = 0.04$.

The PCAs are represented by their two letter abbreviations, with the subscripts denoting the position of substitution.

4.4.2 Nearest neighbour

The nearest neighbour analyses guarantee 100% accuracy on the training data, by the definition of the method. A training example will always be nearest to itself. The simple implementation here is essentially a look-up table, and, as such, provides no insight as to which features are of importance.

4.4.3 Neural networks

A neural network with no hidden units performed as well as neural networks with hidden units, when the data were represented using the Hansch parameters. This permits a more straightforward analysis of the weights of the neural network than when there are hidden units (So and Richards, 1992), because without hidden units, the neural network is essentially just performing a weighted sum. Large positive weights indicate that the respective input should be high for high activity; large negative weights indicate that the respective input should be low for high activity. A separate trial was performed for the analysis of the weights, in which the weights giving optimal performance on the test set were averaged for the five cross-validation data sets. This would not be legitimate for evaluating the predictive performance, but, for the purposes of analysis, provides a well-defined set of weights; the alternative would have been to average the 30 sets of weights from the monitoring procedure (the length of training depends on the monitoring procedure, and so may vary for each drug in the test set). The following relationship was indicated:

$$\log(1/K_i) \propto 0.40\pi_4 + 0.29\pi_5 + 0.51MR_3 - 0.48MR_5. \quad \text{[eqn. 4.9]}$$

This is broadly in accord with a previous neural network analysis using hidden units (So and Richards, 1992), which found that higher MR_3 values improved activity and a more complex dependence on MR_5; negative π_3 values decreased activity, and π_4 and π_5 had not been used as inputs.

The neural network trained on the PCAs, also showed no improvement in test set performance with the addition of hidden units. The results of a similar analysis of the weights using the PCA representation are shown in Figure 4.3. This suggests that

$$\log(1/K_i) \propto - 0.22HA_3 + 0.16\pi A_3 + 0.25\sigma_3 + 0.20\pi D_4 + 0.19\pi A_4 + 0.13\sigma_4 + 0.24SZ_5 + 0.28FL_5. \quad \text{[eqn. 4.10]}$$

4.4.4 Inductive logic programming

GOLEM has only been assessed using the PCA representation of the data. Although it is possible to use the Hansch parameters as input to GOLEM, this is not a particularly useful comparison, as the major motivation for using GOLEM is to generate rules, which the Hansch parameters do not facilitate. An example of a rule generated by GOLEM is given in Table 4.8. These rules have the form of Prolog clauses, and can be translated into English. In total, 59 rules were found for the five cross-validation runs. From these rules, seven consensus rules were formed manually (Table 4.9), by selecting the most

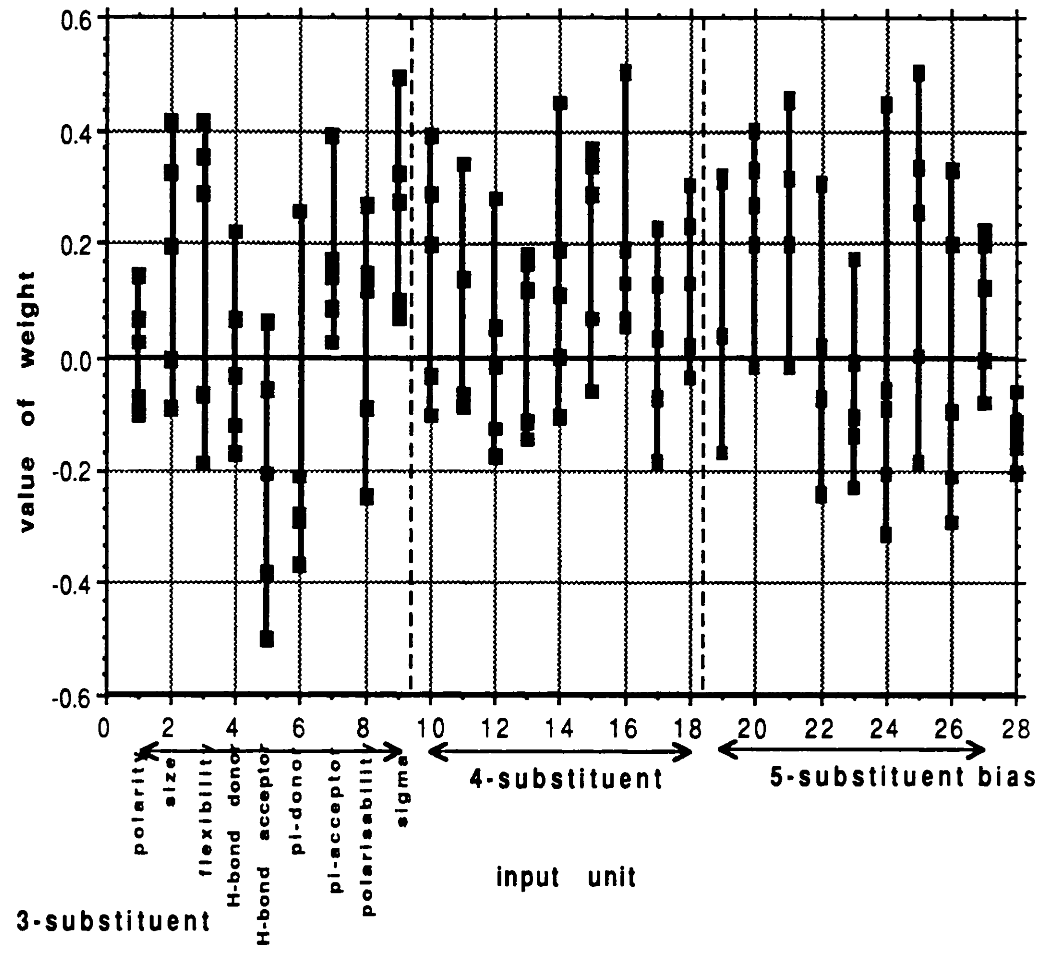


Figure 4.3 Analysis of the weights of neural network with no hidden units trained on the PCA representation. The five values for each weight correspond to the five cross-validation trials. The weights giving the best test set performance are shown.

Test accuracy 0.918, coverage[a] 440

Prolog format:

```
great(A,B) :-
        struc(A,Pos_a3,Pos_a4,_), struc(B,_,_,h),
        h_donor(Pos_a3,h_don0),
        pi_acceptor(Pos_a3,pi_acc0),
        polar(Pos_a3,Pol_a3), great0_polar(Pol_a3),
        size(Pos_a3,Siz_a3), less3_size(Siz_a3),
        polar(Pos_a4,Pol_a4).
```

English translation:

Drug A is better than drug B if:-

drug A has a substituent at position 3 with

hydrogen-bond donor = 0 and

π-acceptor = 0 and

polarity > 0 and

size < 3 and

drug A has a substituent at position 4 (*i.e.*, not H) and

Drug B has no substituent at position 5.

[a]The coverage is the number of pair comparisons that the rule covers, and the accuracy is the number of cases for which the rule is correct divided by the coverage. Both are based on the new 19 drugs.

Table 4.8 Example of a GOLEM Rule

Rule a5[a]: accuracy[b] 0.824, coverage 393

Rule a4[c]: accuracy 0.760, coverage 459

Rule a3: accuracy 0.860, coverage 392

```
great(A,B) :-
      struc(B,h,_,_), not struc(A,h,_,_).
```

English translation - Drug A is better than drug B if:-

drug A has a substituent at position 3 and drug B does not.

Rule b5: accuracy - , coverage 0

Rule b4: accuracy 1.0, coverage 32

Rule b3: accuracy 0.967, coverage 30

```
great(A,B) :-
      struc(A,S1,_,_), struc(B,S2,_,_),
      polar(S1,P1), gt(polar5,P1),
      polar(S2,P2), not gt(polar5,P2).
```

English translation - Drug A is better than drug B if:-

drug A has a substituent at position 3 with
polarity < 5 and drug B does not.

Rule c5: accuracy 0.563 , coverage 16

Rule c3: accuracy 0.651, coverage 215

```
great(A,B) :-
      struc(A,S1,_,_), struc(B,S2,_,_),
      size(S1,Sz1), gt(size3,Sz1),
      size(S2,Sz2), not gt(size3,Sz2).
```

Table 4.9 Consensus Rules[a]

English translation - Drug A is better than drug B if:-

drug A has a substituent at position 3 with

size < 3 and drug B does not.

Rule d5: accuracy 0.883, coverage 383

Rule d4: accuracy 0.783, coverage 511

Rule d3: accuracy 0.917, coverage 468

```
great(A,B) :-
        struc(A,S1,_,_), struc(B,S2,_,_),
        h_donor(S1,h_don0), not h_donor(S2,h_don0).
```

English translation - Drug A is better than drug B if:-

drug A has a substituent at position 3 with

hydrogen-bond donor = 0 and drug B does not.

Rule e5: accuracy 0.872, coverage 289

Rule e3: accuracy 0.841, coverage 510

```
rule(53,A,B) :-
        struc(A,S1,_,_), struc(B,S2,_,_),
        pi_donor(S1,pi_don1), not pi_donor(S2,pi_don1).
```

English translation - Drug A is better than drug B if:-

drug A has a substituent at position 3 with

π-donor = 1 and drug B does not.

Table 4.9 (cont.)

Rule f5: accuracy -, coverage -

Rule f3: accuracy 1.0, coverage 15

```
rule(63,A,B) :-
        struc(A,S1,_,_), struc(B,S2,_,_),
        sigma(S1,G1), gt(sigma5,G1),
        sigma(S2,G2), not gt(sigma5,G2).
```

English translation - Drug A is better than drug B if:-

drug A has a substituent at position 3 with
sigma < 5 and drug B does not.

Rule g5: accuracy 0.563, coverage 16

Rule g3: accuracy 0.698, coverage 189

```
rule(73,A,B) :-
        struc(A,S1,_,_), struc(B,S2,_,_),
        flex(S1,F1), gt(flex3,F1),
        flex(S2,F2), not gt(flex3,F2).
```

English translation - Drug A is better than drug B if:-

drug A has a substituent at position 3 with
flexibility < 3 and drug B does not.

[a]Because of bond rotation, positions 3 and 5 are symmetrical, so each rule for position 3 has a symmetrical copy at position 5, *e.g.* rule a5 has the same conditions as rule a3, but refers to position 5 (struc(B,_,_,h), not struc(A,_,_,h), drug A has a substituent at position 5 and drug B does not).

Table 4.9 (cont.)

[b]The coverage is the number of pair comparisons that the rule covers, and the accuracy is the number of cases for which the rule is correct divided by the coverage. Both are based on the new 19 drugs.

[c]Position 4 is not structurally symmetrical to position 3, but for the cases where a rule was found for position 4 (a, b, and d), it was the same as the rule for position 3.

Table 4.9 (cont.)

commonly found features. These consensus rules, which are consistent with previous work (King *et al.*, 1992), have a simpler form than the automatically generated GOLEM rules, making them easier to understand. In forming these consensus rules, the substitutions at each position were assumed to be independent. The consensus rules were tested against the five cross-validation data sets, giving an average Spearman rank correlation of 0.845. On the separate test set of 19 drugs, the Spearman rank correlation coefficient was 0.793. The rank correlations are similar across training and test sets, which indicates that the rules are not over-fitting the data.

The consensus rules may be used to generate the best predicted drug or drugs. Considering positions 3 and 5, the only possible substituents with the conjunction of: polarity < 5 (Table 4.9: rules b3 and b5), size < 3 (rules c3 and c5), hydrogen-bond donor $= 0$ (rules d3 and d5), π-donor $= 1$ (rules e3 and e5) , sigma < 5 (rules f3 and f5), flexibility < 3 (rules g3 and g5), are OCH_3, I, Cl, and Br (O is excluded, because it is a linking moiety). These are, therefore, the substituents recommended for positions 3 and 5; the rules do not distinguish between these groups. There are fewer constraints at position 4, only the conjunction of: polarity $<$ 5 (rule b4), and hydrogen-bond donor $= 0$ (rule d4). There are 68 substituents with these PCAs (the substituents suggested for positions 3 and 5 are a subset of this set).

4.4.5 Decision tree

CART was implemented as a benchmark for GOLEM, to compare directly a propositional learning algorithm with a predicate logic algorithm, and so only the PCA representation of the data was used. For each of the five cross-validation trials, a separate decision tree was generated. The mean number of leaves for these trees was 49.4. There are quite large variations in the test rank correlations, from 0.774 for split 3 to 0.055 for split 2. The difference in test accuracy was not so large, 85% were examples correct for split 3 and 68% for split 2, showing that a few crucially wrong predictions can ruin the correlation. By examining the most important leaves (the ones with the most examples), it is possible to get some impression what the CART trees mean. These show that it is important to have substitutions at positions 4 and 5, that hydrogen-bond donor should be 0 at position 3, and polarity should not be 0 at positions 3 and 4.

4.5 **Discussion**

The five-fold cross-validation trial gave a range of performances, from a Spearman rank correlation coefficient of 0.499 for CART to one of 0.762 for a nearest neighbour algorithm using the Hansch parameters. The independent test set of 19 drugs gave a similar range, of 0.497 for a nearest neighbour algorithm using PCAs to 0.738 for GOLEM. The data sets are not sufficiently large to allow significant discrimination between the different methods or the different representations. The largest difference between the representations was for the nearest neighbour algorithm which performed better on the Hansch parameters, but the difference was not quite significant at the five per cent level (p = 0.052), as determined by a Fisher's z test (two-tail) (Kendall and Stuart, 1977). The largest difference in methods was between neural networks and CART on the PCAs, with neural networks better, but again not significantly at the five per cent level (p = 0.0854). The neural network results are in agreement with those of the previous study on these data (So and Richards, 1992). The GOLEM results are better than the previous machine learning study (King *et al.*, 1992), which gave a Spearman rank correlation coefficient of 0.457. However, that study used a more difficult test set, which contained the 11 highest activity drugs.

All the methods performed below average on the second split of the data, which suggests that the second test set was not representative of the whole data set. In fact, the second test set contains one data point [3,5-$(OH)_2$] which Hansch *et al.* (Hansch *et al.*, 1982) found to be 6000 times less active than expected. Excluding this point improved their regression from a Pearson r^2 of 0.650, to one of 0.815. Omission of this data point gives similar improvements in the performances of all the methods using the PCA representation, with the Spearman rank correlation coefficient on the second test set rising by 0.163 for linear regression, 0.271 for nearest neighbour, 0.218 for neural networks, 0.072 for CART and 0.040 for GOLEM. In the independent test set of 19 drugs, the 3,5-dimethoxy, 4-(dimethylamino) derivative (drug number 69) is over predicted by all the methods. For this derivative, it has been suggested that the amino substituent would be forced to lie almost perpendicular to the phenyl ring, and the projection above and below the ring plane reduces the expected activity of this drug (Selassie *et al.*, 1991). An analysis of outliers for the cross-validation trial is shown in Figure 4.4 and for the independent test set of 19 drugs in Figure 4.5. As would be expected, there is a tendency for low activities to be over-predicted, and high activities to be under-predicted. In general, all the methods over and under-predict on the same drugs.

Both the linear regression and the neural network methods suggest that the 5-substituent should have a low MR value, that the 3-substituent should have a high MR value, and that π_3 does not have a large effect on activity, but the analyses disagree on the importance of MR_4, π_4 and π_5. The methods using the PCA representation generate a variety of possible influences of structure on activity. Linear regression and GOLEM both indicate that the 3-substituent should have a small size and a low flexibility, and that the 5-substituent should have low flexibility. Linear regression and neural networks suggest that the 4-substituent should be involved in π-bonding. The σ-effect of the 3-substituent is identified as influencing factor by neural networks and GOLEM. Analysis of the crystal structure of the complex formed between trimethoprim and DHFR (Matthews *et al.*, 1985) indicates that both meta sites, *i.e.*, the 3- and 5-substituents, are buried in a hydrophobic environment (Figure 4.6, and restrictions on size and flexibility are consistent with this. It has

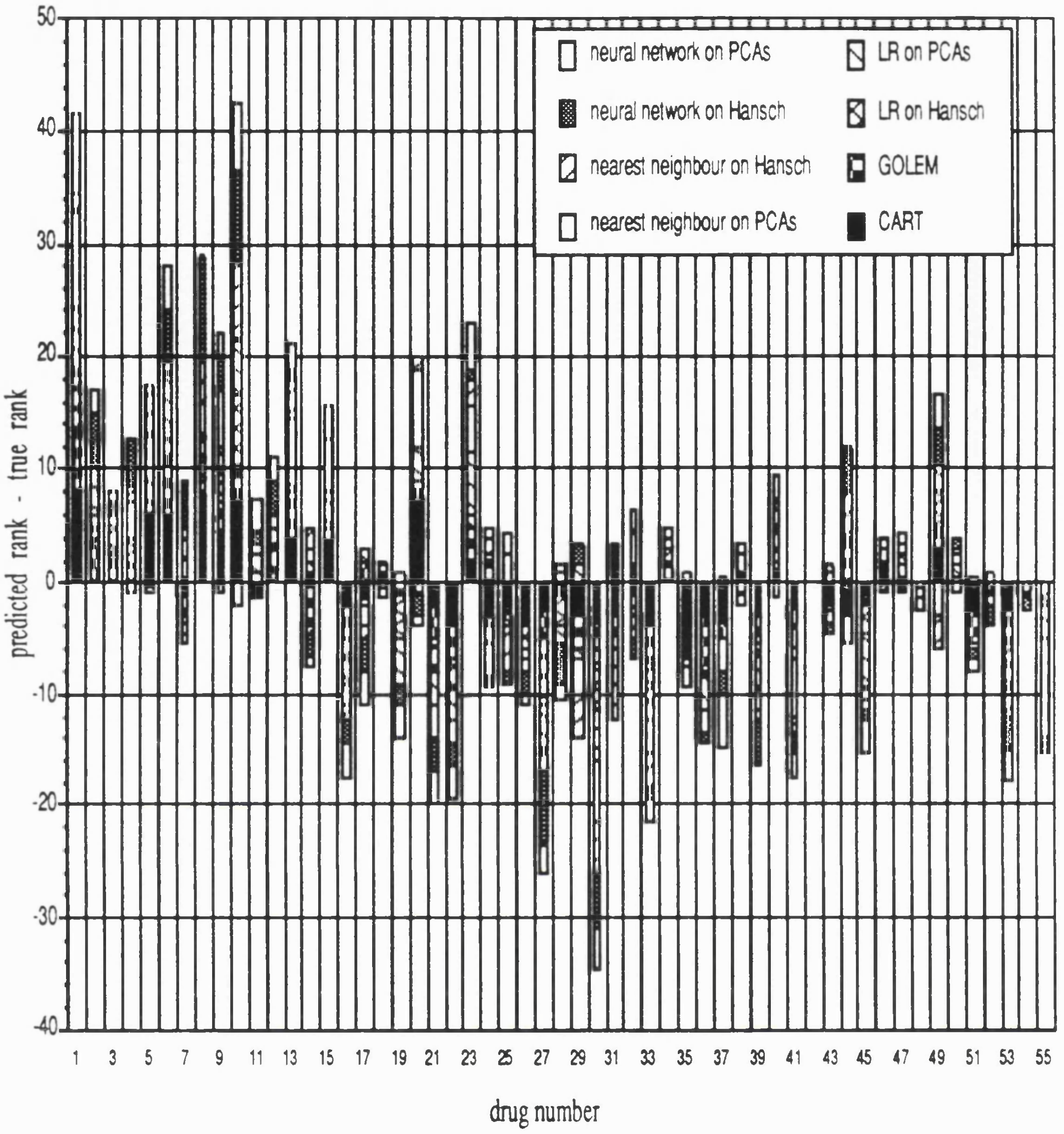


Figure 4.4 Outlier analysis of the cross-validation trial. For each drug, the differences between the test set predicted rank and the true rank for each method is shown as a stacked column, giving the eight components of the sum over all the methods. Linear regression is denoted by LR. The drug number corresponds to the index number in column 1 of Table 4.1.

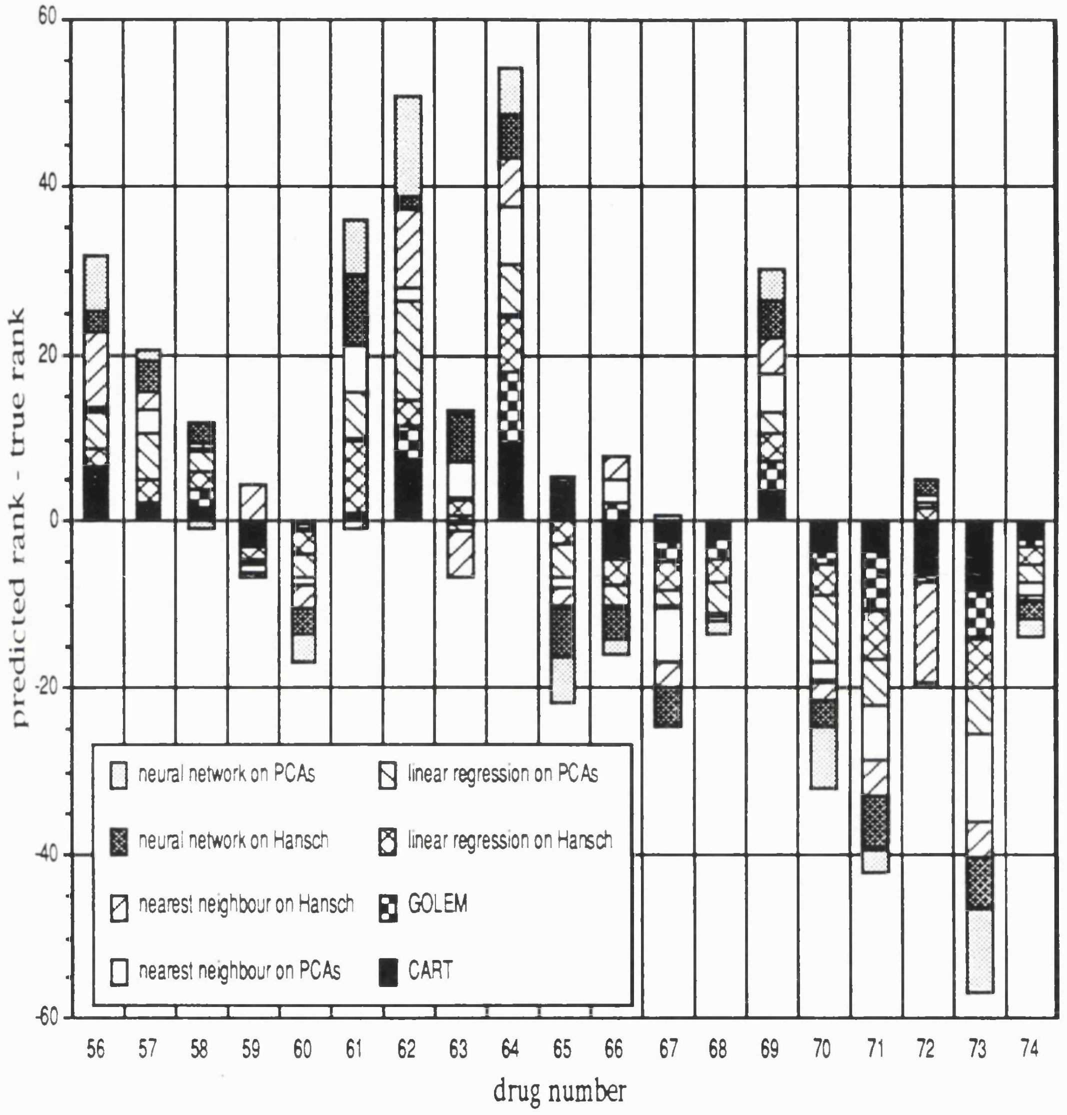


Figure 4.5 Outlier analysis of the independent test set. For each drug, the differences between the test set predicted rank and the true rank for each method is shown as a stacked column, giving a sum over all the methods. Linear regression is denoted by LR. The drug number corresponds to the index number in column 1 of Table 4.1.

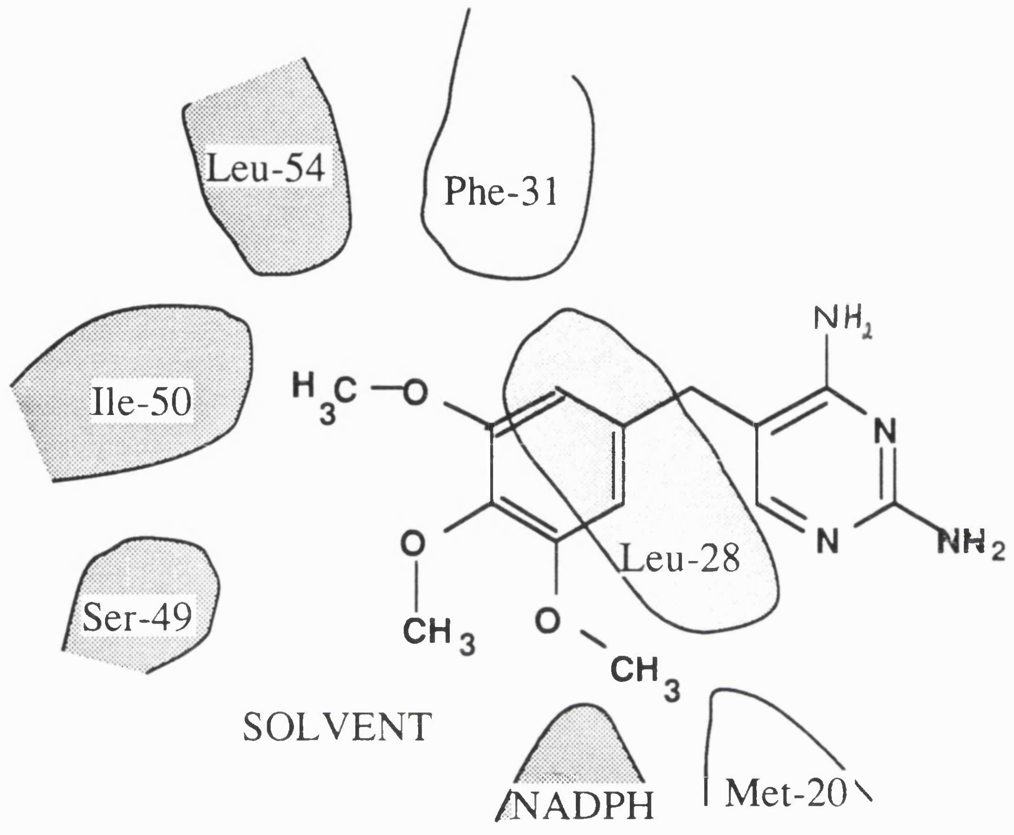


Figure 4.6 Cartoon of the interaction of trimethoprim with DHFR, from X-ray structures (Matthews *et al.*, 1985; Champness *et al.*, 1986). Faint stippling indicates that the residue lies below the plane of the phenyl ring; darker stippling indicates that the atoms are above.

also been proposed that the 4-substituent should lie in the plane of the aromatic ring (Roth *et al.*, 1987), and a substituent involved in π-bonding would have such a constraint.

The relationships generated using the PCAs illustrate the potential of the representation as a QSAR tool providing insight into drug-receptor interactions. The small size of the data set in statistical terms, although not in QSAR terms, has not allowed discrimination between the methods. The fact that the addition of hidden units to neural network did not give an improvement, even though the linear regression analysis suggested that squared PCA terms were important, indicates that the power of neural networks may not be fully exploited using the PCA representation, especially for small data sets. In the following chapter, a larger data set of DHFR-inhibitors provides a more statistically sensitive test of the methods and representations, and demonstrates the general applicability of PCAs with a wide extension of the representation to many more substituents.

Chapter 5

Quantitative structure-activity relationships: neural networks and inductive logic programming compared to statistical methods. The inhibition of dihydrofolate reductase by triazines

5.1 Synopsis

One of the largest available data sets for developing a quantitative structure-activity relationship (QSAR) - the inhibition of dihydrofolate reductase (DHFR) by 2,4-diamino-6,6-dimethyl-5-phenyl-dihydrotriazine derivatives - has been used for a six-fold cross-validation trial of neural networks and inductive logic programming (ILP), against the more traditional statistical methods of linear regression, nearest neighbour algorithms and decision trees. The representation of drugs by physicochemical attributes (PCAs) (King *et al.*, 1992) has been extended to cover a wide range of substituents. The PCA representation has been compared against the established Hansch parameters. Methods performed quite similarly, with test set performances, as measured by the Spearman rank correlation coefficient, ranging from 0.272 to 0.635. This suggests that even the largest data sets available for QSAR analysis are not of sufficient size to allow significant discrimination between current QSAR methods or between the Hansch parameter and PCA representations. However, the generality of PCAs has been demonstrated and the representation is useful in providing readily understandable rules about drug-receptor interactions.

5.2 Introduction

In the last chapter, the representation of drugs by physicochemical attributes (PCAs) (King *et al.*, 1992) has been shown to be useful in the analysis of quantitative structure-activity relationships (QSARs), in particular, for the generation of rules concerning the interaction between pyrimidines and dihydrofolate reductase (DHFR). In this chapter, the general applicability of the PCAs is demonstrated by the extension of the representation to a wider range of substituents. Further, the use of one of the largest data sets available for QSAR analyses - the inhibition of dihydrofolate reductase (DHFR) by 2,4-diamino-6,6-dimethyl-5-phenyl-dihydrotriazine derivatives (Figure 5.1) - provides greater statistical sensitivity for benchmarking the PCA representation and the newer methodologies. The methods applied in the preceding chapter: neural networks, inductive logic programming (ILP), linear regression, a nearest neighbour algorithm, and a decision tree, have been used again in this study, with only minor alterations in implementation. The representations (Hansch parameters and PCAs) of the data have also been kept as similar as possible. This chapter and the preceding one illustrate the usefulness and provide a thorough evaluation of the PCA representation and a careful comparison of QSAR methods.

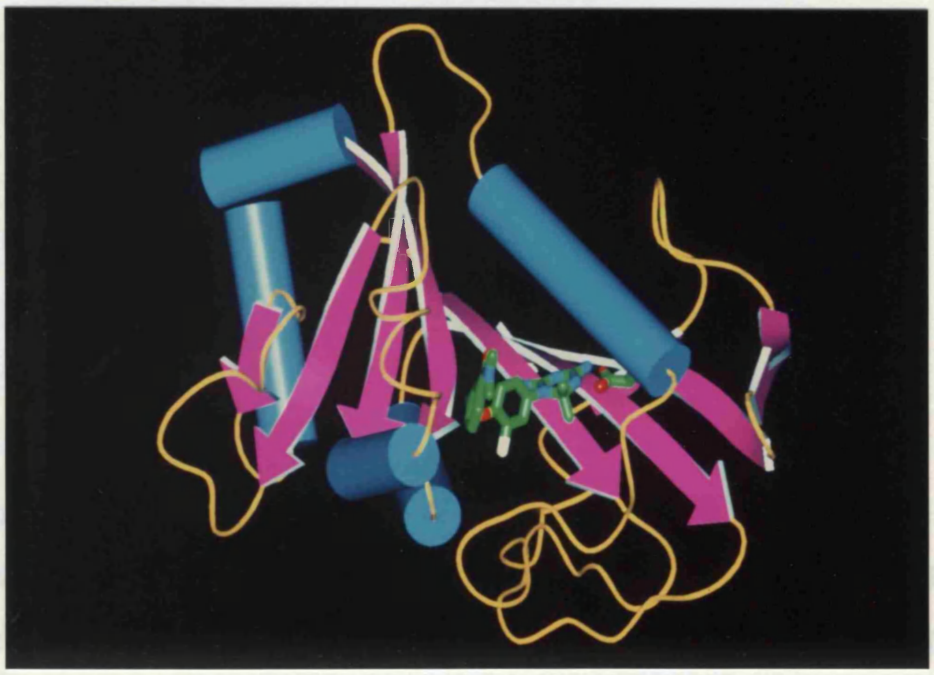
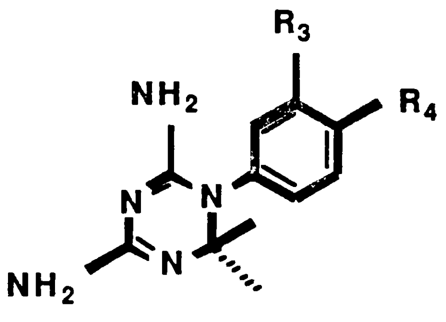


Figure 5.1 Model of a triazine bound to DHFR, displayed using the graphics program PREPI by Dr. Suhail Islam.

5.3 Methods

5.3.1 Data

The data used in the study were 186 2,4-diamino-6,6-dimethyl-5-phenyldihydrotriazines (I), listed in Table 5.1. These



I

data are a subset of those used to derive a QSAR by multilinear regression (Silipo and Hansch, 1975), and by neural networks (Andrea and Kalayeh, 1991). Baker and co-workers in the 1960s synthesised these triazines and studied their inhibition of DHFR isolated from L1210 mouse leukaemia cells and Walker 256 rat tumours. The 186 triazines have two or three rings, with the phenyl ring (the second ring) substituted at either the 3- or 4-positions; ortho-substituents were not considered as there were only 11 examples, and this is too small a set for generalisations (Hansch and Fukunga, 1977). This dataset was chosen because of its size, and it has been the subject of two previous QSAR analyses (Andrea and Kalayeh, 1991; Silipo and Hansch, 1975). Silipo and Hansch (1975) concluded that the data were "an

Drug no.[a]	Activity log (1/*C*)	Substituents
9	4.68	4-$CONHC_6H_4$-4'-SO_2F
10	4.68	4-$CONHC_6H_4$-3'-SO_2F
11	4.70	4-C_6H_5
13	4.85	3-$OCH_2CON(CH_2CH_2)_2O$
14	5.14	4-CN
15	5.19	4-$CHCHCONHC_6H_4$-4'-SO_2F
16	5.44	3-OCH_2CONMe_2
17	5.74	4-$CH(Ph)NHCOCH_2C_6H_4$-4'-SO_2F
18	5.82	4-Cl-3-$(CH_2)_2C_6H_4$-4'-SO_2F
19	5.89	4-$CHCHCONHC_6H_4$-3'-SO_2F
20	5.96	3-$CONHC_6H_4$-4'-SO_2F
21	6.11	3-$NHCOCH_2Br$-4-$O(CH_2)_3C_6H_5$
22	6.11	3-$CH_2NHCONEt_2$
23	6.17	3-OCH_3
24	6.17	4-$OCH_2CON(CH_3)C_6H_5$
25	6.20	4-$CH_2CH(CH_2CH_2Ph)CONHC_6H_4$-4'-$SO_2F$
26	6.21	3-$COCH_2Cl$
27	6.24	4-$CH_2CH(\alpha\text{-}C_{10}H_7)CONHC_6H_4$-4'-$SO_2F$
28	6.26	4-OCH_2CONMe_2
29	6.33	4-$CH_2CH(Ph\text{-}2''\text{-}OCH_3)CONHC_6H_4$-4'-$SO_2F$
30	6.37	3-Cl-4-$OCH_2C_6H_{10}CH_2OC_6H_4$-4'-$SO_2F$
31	6.37	3-$CH(CH_2NHCOCH_2Br)(CH_2)_3C_6H_5$

Table 5.1. Triazines used in this study

32	6.43	3-$CH_2NHCON(CH_2CH_2)_2O$
33	6.45	4-$COCH_2Cl$
34	6.46	4-CH_2CH(Ph-3"-OCH_3)$CONHC_6H_4$-4'-SO_2F
35	6.52	4-$CH(CH_2NHCOCH_2Br)(CH_2)_3C_6H_5$
39	6.58	3-$CH_2NHCOCH_2Br$
40	6.60	3-$CONHC_6H_4$-3'-SO_2F
41	6.63	4-CH_2CONMe_2
42	6.66	4-$OCH_2CON(CH_2)_4$
43	6.68	3-$OCH_2CONMePh$
44	6.72	4-OCH_2CONEt_2
45	6.72	3-$CH_2CH(CH_2NHCOCH_2Br)C_6H_5$
46	6.72	4-Cl-3-$O(CH_2)_5OC_6H_4$-4'-SO_2F
47	6.77	4-CH_2CONEt_2
48	6.77	4-Cl-3-$(CH_2)_4C_6H_4$-4'-SO_2F
50	6.85	3-$CH_2OCONHC_6H_5$
51	6.85	3-C_6H_5
52	6.89	4-$CH_2CH(Ph)CONHC_6H_4$-3'-SO_2F
55	6.92	3-$OCH_2CONHC_6H_4$-4'-SO_2F
56	6.92	4-CH_2CN
57	6.92	H
58	6.92	3-$OCH_2C_6H_4$-3'-$NHCOCH_2Br$
59	7.00	4-$CH_2CON(Me)C_6H_5$
60	7.05	4-$(CH_2)_2CONMe_2$
62	7.07	3-Cl-4-$O(CH_2)_3OC_6H_4$-4'-SO_2F
63	7.07	3-NO_2

Table 5.1 (cont.)

64	7.10	3-$(CH_2)_2COCH_2Cl$
65	7.10	3-$(CH_2)_4COCH_2Cl$
66	7.12	4-$OCH_2CON(CH_2)_5$
67	7.12	4-$CH_2CON(CH_2CH_2)_2O$
68	7.12	4-$(CH_2)_6C_6H_4$-4'-SO_2F
69	7.13	3-Cl-4-$OCH(CH_3)CONHC_6H_4$-4'-SO_2F
70	7.13	4-$CH_2CH(Ph)CONHC_6H_4$-4'-SO_2F
71	7.14	3-Cl-4-$O(CH_2)_2O(CH_2)_2OC_6H_4$-4'-$SO_2F$
72	7.15	3-Cl-4-$O(CH_2)_3CONHC_6H_4$-4'-SO_2F
73	7.16	3-Cl-4-OCH_2CONMe_2
74	7.17	3-Cl-4-$O(CH_2)_3CONHC_6H_4$-3'-SO_2F
75	7.17	4-Cl-3-$O(CH_2)_4OC_6H_4$-4'-SO_2F
76	7.17	4-CH_2CH(Ph-3"-CH_3)$CONHC_6H_4$-4'-SO_2F
77	7.19	3-$(CH_2)_2CONHC_6H_4$-4'-SO_2F
78	7.24	4-CH_2CH(Ph-4"-CH_3)$CONHC_6H_4$-4'-SO_2F
79	7.24	4-CH_2CH(Ph-2"-CH_3)$CONHC_6H_4$-4'-SO_2F
82	7.24	3-Cl-4-$O(CH_2)_4CONHC_6H_4$-4'-SO_2F
84	7.27	4-Cl-3-$O(CH_2)_2OC_6H_4$-4'-SO_2F
85	7.27	3-SO_2F
86	7.28	3-Cl-4-$O(CH_2)_3NHCONHC_6H_4$-3'-SO_2F
87	7.28	4-$(CH_2)_2CONEt_2$
88	7.29	3-Cl-4-$OCH_2CON(CH_2)_4$
89	7.29	4-$OCH_2CON(CH_2CH_2)_2O$
90	7.29	4-$CH(CH_3)CH_2CONHC_6H_4$-4'-SO_2F
91	7.30	4-$CH_2CON(Me)CH_2C_6H_5$

Table 5.1 (cont.)

92	7.31	4-$(CH_2)_2CON(Me)CH_2C_6H_5$
93	7.32	4-$(CH_2)_2CON(CH_2CH_2)_2O$
94	7.32	4-$O(CH_2)_3NHCONHC_6H_4$-3'-SO_2F
95	7.34	3-Cl-4-$O(CH_2)_3NHCOC_6H_4$-4'-SO_2F
96	7.34	3-$CH_2CONHC_6H_4$-4'-SO_2F
97	7.35	4-$CH_2NHCONHC_6H_4$-4'-SO_2F
98	7.35	4-$(CH_2)_2CON(C_3H_7)_2$
101	7.39	3-Cl-4-$S(CH_2)_2CONHC_6H_4$-4'-SO_2F
102	7.41	4-$(CH_2)_2C_6H_4$-4'-SO_2F
104	7.41	4-$(CH_2)_2NHSO_2C_6H_4$-4'-SO_2F
105	7.42	3-Cl-4-$SCH_2CONHC_6H_4$-4'-SO_2F
107	7.43	3-Cl-4-$OCH_2CONHC_6H_4$-4'-SO_2F
109	7.43	3-Cl-4-$OCH_2C_6H_3$-3'-Cl-4'-SO_2F
111	7.44	3-Cl-4-$O(CH_2)_2OC_6H_4$-4'-SO_2F
113	7.46	3-Cl-4-$O(CH_2)_6OC_6H_4$-4'-SO_2F
114	7.46	4-$(CH_2)_2CONHC_6H_3$-3'-OCH_3-4'-SO_2F
115	7.47	3-Cl-4-$OCH_2CON(CH_3)C_6H_4$-4'-SO_2F
116	7.47	3-Cl-4-$OCH_2CON(CH_2)_5$
117	7.48	3-Cl-4-$CH_2OC_6H_4$-4'-SO_2NMe_2
119	7.49	3-$O(CH_2)_4OC_6H_4$-4'-SO_2F
120	7.51	4-Cl-3-$O(CH_2)_3OC_6H_4$-4'-SO_2F
121	7.51	3-Cl-4-$OCH_2C_6H_4$-3'-CN
122	7.52	3-Cl-4-$OCH_2C_6H_5$
123	7.52	4-$SCH_2CONHC_6H_4$-4'-SO_2F
125	7.52	3-$CH_2NHCONHC_6H_5$

Table 5.1 (cont.)

126	7.55	4-$CH_2CH(Me)CONHC_6H_4$-4'-SO_2F
127	7.55	3-$O(CH_2)_3OC_6H_4$-4'-$NHCOCH_2Br$
128	7.56	4-$(CH_2)_2CON(Me)C_6H_5$
129	7.57	3-Cl-4-$O(CH_2)_4OC_6H_4$-4'-SO_2F
130	7.57	3-Cl-4-$O(CH_2)_5OC_6H_4$-4'-SO_2F
131	7.58	3-Cl-4-$OCH_2C_6H_4$-4'-SO_2F
132	7.60	4-$(CH_2)_2CONHC_6H_4$-4'-SO_2F
133	7.62	3-Cl-4-$(CH_2)_2CONHC_6H_4$-4'-SO_2F
134	7.62	3-$CH_2NHCONHC_6H_4$-3'-SO_2F
135	7.64	4-$(CH_2)_2NHSO_2C_6H_4$-3'-SO_2F
136	7.64	3-Cl-4-OCH_2CONEt_2
137	7.64	3-$O(CH_2)3OC_6H_4$-3'-$NHCOCH_2Br$
139	7.66	3-$O(CH_2)_2OC_6H_4$-3'-$NHCOCH_2Br$
140	7.66	3-Cl-4-$SCH_2CONHC_6H_4$-3'-SO_2F
141	7.66	3-Cl-4-$O(CH_2)_4NHCOC_6H_4$-4'-SO_2F
142	7.66	4-$(CH_2)_3CONHC_6H_4$-4'-SO_2F
143	7.68	3-Cl-4-$O(CH_2)_3NHCONHC_6H_4$-4'-SO_2F
144	7.70	3-Cl-4-$O(CH_2)_4NHCONHC_6H_4$-3'-SO_2F
145	7.70	3-$(CH_2)_4C_6H_3$-3'-Cl-4'-SO_2F
146	7.70	3-Cl-4-$(CH_2)_4C_6H_3$-3'-Cl-4'-SO_2F
147	7.70	4-$(CH_2)_4C_6H_4$-4'-SO_2F
148	7.70	4-$CH_2CONHC_6H_4$-4'-SO_2F
149	7.70	4-$O(CH_2)_2OC_6H_4$-4'-$NHCOCH_2Br$
150	7.72	3-Cl-4-$OCH_2C_6H_4$-3'-$CONMe_2$
152	7.72	4-$OCH_2CONHC_6H_4$-4'-SO_2F

Table 5.1 (cont.)

153	7.72	3-Cl-4-$OCH_2CONHC_6H_4$-3'-SO_2F
154	7.72	3-Cl-4-$OCH_2C_6H_4$-3'-SO_2F
156	7.72	4-$CH_2NHCONHC_6H_3$-3'-CH_3-4'-SO_2F
157	7.74	4-$(CH_2)_2CONHC_6H_4$-3'-SO_2F
159	7.76	3-Cl
160	7.76	3-CF_3
162	7.77	3-$CH_2NHCOC_6H_4$-3'-$CONMe_2$
165	7.80	3-Cl-4-$CH_2NHCONHC_6H_3$-3'-CH_3-4'-SO_2F
166	7.80	4-$O(CH_2)_2OC_6H_4$-4'-SO_2F
168	7.82	3-Cl-4-$O(CH_2)_2NHCONHC_6H_3$-3'-CH_3-4'-SO_2F
169	7.82	3-$O(CH_2)_2OC_6H_4$-4'-SO_2F
171	7.85	3-Cl-4-$(CH_2)_2C_6H_4$-4'-SO_2F
173	7.85	3-Cl-4-$(CH_2)_2C_6H_3$-3'-Cl-4'-SO_2F
174	7.85	3-Cl-4-$OCH_2CON(CH_2CH_2)_2O$
175	7.85	3-Cl-4-$OCH_2C_6H_4$-3'-$CON(CH_2CH_2)_2O$
176	7.85	3-Cl-4-$OCH_2C_6H_4$-3'-$CON(CH_2)_4$
177	7.89	3-Cl-4-$OCH_2CON(CH_3)C_6H_5$
178	7.89	4-$OCH_2CONHC_6H_5$
179	7.89	4-$(CH_2)_2C_6H_5$
180	7.89	4-$(CH_2)_2CONHC_6H_3$-3'-CH_3-4'-SO_2F
181	7.92	3-Cl-4-$CH_2NHCONHC_6H_4$-4'-SO_2F
182	7.92	3-Cl-4-$O(CH_2)_2NHCONHC_6H_4$-4'-SO_2F
183	7.92	4-$(CH_2)_3CONHC_6H_4$-3'-SO_2F
184	7.92	4-$(CH_2)_2COCH_2Cl$
185	7.92	3-OC_6H_4-4'-$NHCOCH_2Br$

Table 5.1 (cont.)

186	7.92	3-Cl-4-$(CH_2)_4C_6H_5$
189	7.96	3-$O(CH_2)_3OC_6H_4$-4'-SO_2F
192	8.00	3-Cl-4-$OCH_2C_6H_3$-3'-SO_2F-4'-Cl
194	8.00	4-$OCH_2CONHC_6H_4$-3'-SO_2F
196	8.00	3-$CH_2C_6H_5$
197	8.00	4-$(CH_2)_4C_6H_5$
198	8.02	3-Cl-4-$OCH_2C_6H_4$-3'-$CON(CH_2)_5$
199	8.02	3-$CH_2NHCONHC_6H_4$-3'-OCH_3
200	8.02	4-$(CH_2)_2CONHC_6H_3$-3'-CH_3-4'-SO_2F
201	8.03	3-Cl-4-$(CH_2)_4C_6H_4$-3'-SO_2F
203	8.04	4-$CH_2NHCONHC_6H_4$-3'-SO_2F
204	8.04	4-$(CH_2)_2CON(Me)C_6H_4$-4'-SO_2F
206	8.05	4-$CH_2C_6H_5$
207	8.05	3-$CH_2NHCONHC_6H_4$-3'-Cl
208	8.06	3-Cl-4-$O(CH_2)_3NHCONHC_6H_3$-3'-SO_2F-4'-CH_3
209	8.06	4-$CH_2CONHC_6H_4$-3'-SO_2F
212	8.09	3-$CH_2NHCONHC_6H_4$-3'-NO_2
213	8.10	3-$(CH_2)_4C_6H_4$-4'-SO_2F
214	8.10	3-$(CH_2)_4C_6H_4$-3'-SO_2F
215	8.10	3-$(CH_2)_2C_6H_4$-4'-SO_2F
216	8.11	4-$(CH_2)_2NHCOC_6H_4$-4'-SO_2F
217	8.11	3-Cl-4-$(CH_2)_4C_6H_3$-3'-SO_2F-4'-Cl
219	8.13	3-$O(CH_2)_2OC_6H_4$-4'-$NHCOCH_2Br$
220	8.14	3-Cl-4-$OCH_2C_6H_4$-3'-$CONEt_2$
221	8.14	3-Cl-4-$(CH_2)_4C_6H_4$-4'-SO_2F

Table 5.1 (cont.)

222	8.14	3-Br-4-$OCH_2CONHC_6H_4$-4'-SO_2F
223	8.14	4-$(CH_2)_4OC_6H_4$-4'-SO_2F
224	8.19	3-$(CH_2)_2C_6H_5$
225	8.19	3-$CH_2NHCONHC_6H_4$-3'-CN
229	8.24	4-$(CH_2)_2CONHC_6H_3$-3'-SO_2F-4'-OCH_3
231	8.24	4-$O(CH_2)_4C_6H_5$
233	8.26	3-$(CH_2)_2C_6H_4$-4'-$NHCOCH_2Br$
234	8.27	3-Cl-4-$(CH_2)_2C_6H_3$-3'-SO_2F-4'-Cl
238	8.35	3-$(CH_2)_4OC_6H_5$
239	8.35	3-$(CH_2)_4C_6H_5$
240	8.37	3-$(CH_2)_4C_6H_3$-3'-SO_2F-4'-Cl
241	8.38	3-$(CH_2)_4C_6H_4$-4'-$NHCOCH_2Br$
246	8.41	3-$(CH_2)_4C_6H_4$-3'-$NHCOCH_2Br$
250	8.54	3,4-Cl_2

[a]The numbers in this column correspond to those in column 1 of Table I of Andrea and Kalayeh (1991), and also to those in column 1 of Table I of Silipo and Hansch (1975).

Table 5.1 (cont.)

excellent testing ground for further new approaches to structure-activity analysis".

The data were divided into six different sets, each of 31 drugs; each of the 186 drugs appears once only in only one of the sets (Table 5.2). Each split of the data was used as a testing set, and for each testing set the other five sets were combined to give a training set of 155 drugs. Each of the 186 triazines appears once only in only one of the test sets, so all the drugs are tested.

5.3.2 Hansch parameters

The approach of Silipo and Hansch (1975), now often used, was to correlate the activity of the triazines with the chemical properties of the 3- and 4-substituents of the phenyl ring. The activity was measured by log $1/C$, where C is the molar concentration that produces 50% reversible inhibition of dihydrofolate reductase obtained from L1210 mouse leukaemia cells and Walker 256 rat tumours when assayed with 6μmol dihydrofolate at pH 7. The chemical properties of the substituents were represented using the hydrophobic parameter, π and the molar refractivity, MR. A multilinear regression was then performed using π_3, π_4, MR_3, MR_4, and also $\pi_3{}^2$, $\pi_4{}^2$, $MR_3{}^2$, and $MR_4{}^2$, to allow some non-linear dependence. Six discrete variables (indicator variables: I-1, I-2, I-3, I-4, I-5, I-6) were used to improve the regression. These variables take the value of 1 or 0 for structural features that could not be parameterised by the hydrophobic constants and the molar refractivities. Using the

Split number	data points[a]
1	184, 234, 24, 48, 107, 22, 85, 140, 171, 11, 31, 73, 63, 68, 241, 206, 55, 21, 30, 168, 40, 27, 199, 88, 84, 219, 117, 246, 239, 166, 240
2	9, 33, 23, 177, 114, 132, 50, 135, 123, 69, 120, 71, 58, 102, 156, 196, 15, 119, 45, 129, 98, 130, 143, 213, 225, 96, 165, 224, 176, 43, 233
3	62, 74, 220, 26, 186, 115, 150, 153, 72, 173, 18, 222, 194, 111, 44, 185, 207, 82, 169, 94, 104, 137, 97, 113, 41, 93, 147, 159, 67, 39, 178
4	105, 201, 86, 229, 35, 231, 152, 70, 76, 221, 78, 13, 125, 28, 180, 174, 16, 59, 20, 19, 214, 91, 212, 65, 17, 160, 126, 101, 148, 122, 34
5	146, 57, 157, 189, 116, 183, 87, 223, 154, 200, 162, 25, 95, 144, 136, 145, 46, 51, 181, 90, 89, 10, 121, 75, 42, 197, 109, 77, 192, 204, 216
6	215, 64, 29, 142, 128, 175, 250, 56, 182, 208, 131, 149, 60, 238, 179, 198, 52, 66, 141, 139, 209, 92, 14, 127, 47, 134, 217, 133, 32, 203, 79

[a]The numbers in this column correspond to those in column 1 of Table 5.1.

Table 5.2. Splits of the data used for comparative study

indicator variable *I*-1, with a value of 1 for Walker enzyme data, and 0 for L1210 enzyme data, allowed the merging of the data from the two test systems. Ortho-substitution was flagged using *I*-2. Rigid groups directly attached to the 3- or 4- positions were marked by *I*-3. *I*-4 was 1 for all compounds containing 4-$OCH_2C_6H_4SO_2OC_6H_4$-X. *I*-5 was 1 for flexible bridges between the N-phenyl moiety and a second phenyl ring. *I*-6 was used for some other bridging groups. Other indicator variables that did not appear in the final regression equation were also examined. The hydrophobic constants, molar refractivities, and indicator variables for all the drugs in this study are listed in previous studies (Andrea and Kalayeh, 1991; Silipo and Hansch, 1975), and are not reproduced here.

5.3.3 PCA representation

The substituents of the drugs were described by ten PCAs assigned heuristically: the polarity (the amount of residual charge on the α and β atoms of the substituent), size, flexibility, the number of hydrogen-bond donors and the number of hydrogen-bond acceptors, the presence and strength of π-acceptors and π-donors, the polarisability of the molecular orbitals, the σ-effect and branching. There were six regions where there might be a substituent: the 3- and 4-positions of the phenyl ring (regions 1 and 4 respectively); if the substituent at the 3-position contained a ring itself, then the 3- and 4-positions of this third ring (regions 2 and 3 respectively: the attributes for these regions were set to zero if there was no third ring there); if the substituent at the 4-position of the phenyl ring contained a ring itself, then the 3- and 4-positions of this third ring (regions 5 and 6 respectively: the attributes for these regions were set to zero if there was no third ring there). Thus, with six possible regions of chemical variability and ten properties for each region, the PCA representation describes each drug by 60 parameters. The attributes assigned to the different fragments, some of which link two rings, are given in the previous chapter, in Table 4.4.

5.3.4 Linear regression

The six training sets were assigned the Hansch parameters π_3, π_4, MR_3, MR_4 and $\Sigma\sigma_{34}$ (the sum of the σ values of substituents at the 3- and 4-positions of the phenyl ring (Andrea and Kalayeh, 1991; Hansch and Silipo, 1974)) and the indicator variables I-1, I-3, I-4, and I-5 (I-2 indicates ortho-substitution and I-6 characterises a fourth ring; neither feature was present in these data). A stepwise linear regression was performed on these variables and the squares of the Hansch parameters, in a procedure similar to Silipo and Hansch (1975), but with the advantage that the regression equation was derived fully automatically, using the STEP command in Minitab release 7.2 VAX/VMS version (C) copyright (1989) Minitab, Inc. To assess the contribution of the indicator variables to the regression, a stepwise linear regression was performed using just the Hansch parameters and their squares, and another stepwise regression was performed using just the indicator variables. Indicator variables should be employed as variables of last resort; they can be used when the limit has been reached for one's ability to define important structural features with known physicochemical constants (Hansch and Fukunga, 1977). Stepwise linear regression analyses were also done using the PCA representation of the drugs; one regression used the sixty attributes and the other used the sixty attributes and their squares, analogous to the approach using the Hansch parameters.

5.3.5 Nearest neighbour

A nearest neighbour algorithm was implemented as described in Chapter 4. An n-dimensional vector represents each drug in the data set, where, in this case, n is either 6, when the drugs are described by the Hansch parameters π_3, π_4, MR_3, MR_4, $\Sigma\sigma_{34}$ and the indicator variable I-1, or n is 60, for the PCA representation.

5.3.6 Neural networks

The neural network, as implemented in Chapter 4, was trained using both representations of the data. In an approach similar to Andrea and Kalayeh (1991), a neural network with six input units, five hidden layer units and one output unit was trained to predict the activity of drugs given π_3, π_4, MR_3, MR_4, $\Sigma\sigma_{34}$ and the indicator variable I-1. The only differences were that a monitor set was used to determine when to stop training, and different splits of the data were used. The data were split as for the other methods, with six different sets, each of 31 drugs; each of the 186 drugs appears once only in only one of the sets. Each split of the data was used as a testing set, and for each testing set the other five sets were combined to give a training set of 155 drugs. For each of the 31 drugs in a test set, 1 was used as the unseen test set, and 30 were used to monitor the neural network. This was repeated for each drug in the test set in turn, so that the neural network was only trained once per test set of 31, rather than being trained 31 times. The same monitoring procedure was used for a neural network with 60 input units, a varying number of hidden units and one output unit (Figure 5.2) trained on the same data. Lack of data precluded the use of more than three hidden units, because of the problem of overfitting.

5.3.7 Inductive logic programming

The work discussed in this section and the following section (5.3.7) was done by Dr. Ross King. GOLEM, an inductive logic program, was implemented using the PCA representation. As before (Chapter 4), the positive facts were the paired examples of greater activity and the negative examples were the paired examples of lower activity. The background facts were the chemical structures of the drugs and the properties of the substituents. Chemical structure was represented in the form:

struc3(d217, Cl, absent).
struc4(d217, $(CH_2)_4$, substl4).
subst(substl4, SO_2F, Cl).

This is the Prolog representation of the drug 217 - 3-Cl, 4-$(CH_2)C_6H_3$-4'-Cl, 3'-SO_2F. The first clause represents

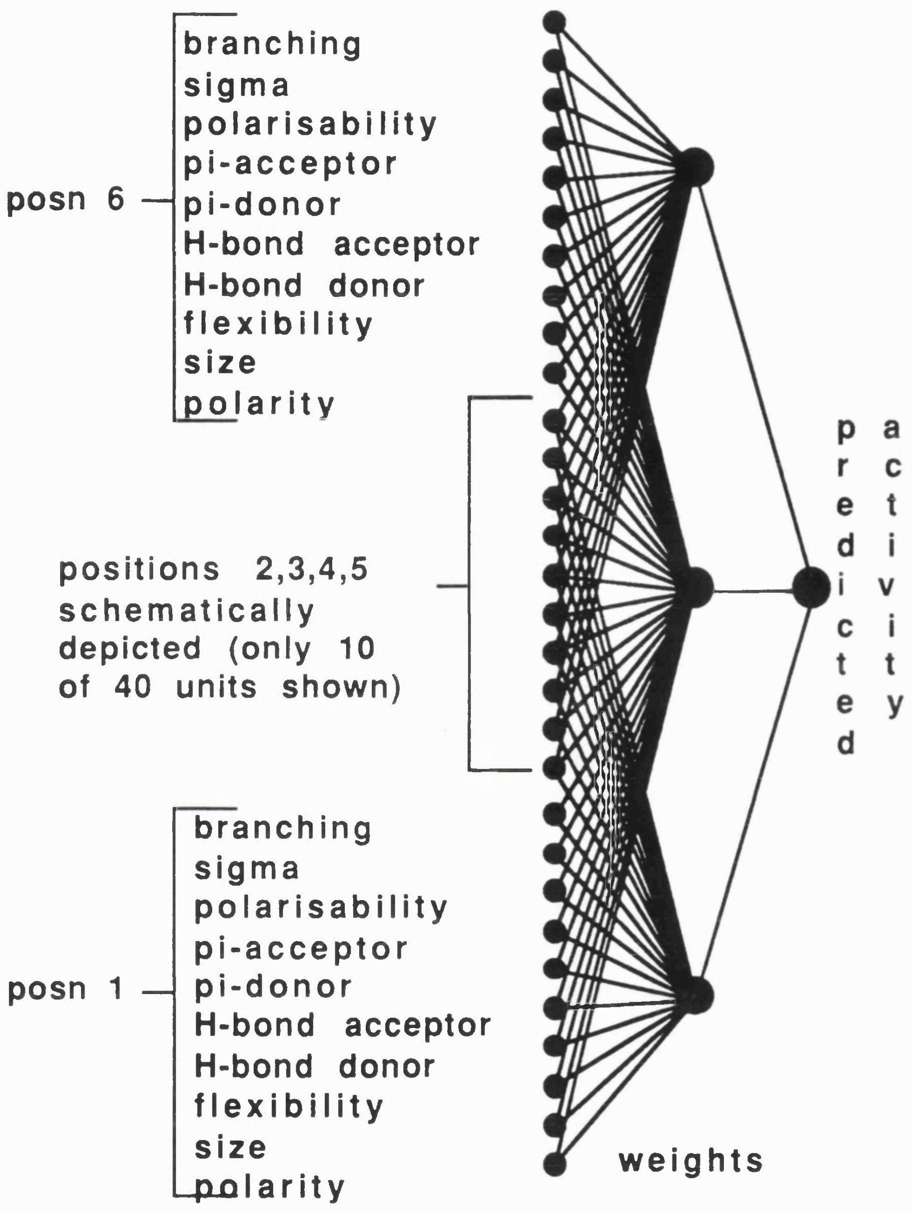


Figure 5.2 Schematic representation of the neural network trained using the PCA representation. Only 30 of the 60 input units actually used are shown. All the weights are not shown, but each input unit was connected to three hidden units by a weight.

substitutions at position 3 on the N-phenyl moiety: a Cl is present and there is an absence of a further phenyl ring. The second clause represents substitutions at position 4 on the N-phenyl moiety: there is a $(CH_2)_4$ bridge to a second phenyl ring (implicit in the representation). This second phenyl ring has a $S0_2F$ group substituted at position 3 and a Cl group substituted at position 4. This is represented using the linker constant subst14 to the third clause. This structural representation could be easily extended to include more substitution positions and more rings, *e.g.*, the drugs included in previous studies (Andrea and Kalayeh, 1991; Silipo and Hansch, 1975) but not in this study.

For each of the six splits the input to GOLEM was 2933 facts in the background information and 1000 positive and negative facts. The positive and negative facts were the equivalent random sample of all possible pairs; all the pairs could not be used because of computational complexity. One hundred rules were found for each split, with the following parameter settings for GOLEM: depth, $i = 5$; clause parameter, $j = 3$; error level, *noise* = 50; sample size, *rlggsample* = 20 (as defined in the original GOLEM work (Muggleton and Feng, 1990)); no examples were covered.

The method for selecting the best rules from the starting set of 100 rules was improved, based on the insight that all correct classifications do not have the same utility. For example, consider three drugs A, B and C, where B is slightly more active that C, and A is far more active that B. It is better to predict that A is more active that C, than to predict that B is more active that

C. The measure used by GOLEM, utility, was the squared difference in rank, with correct predictions having a positive utility, and incorrect predictions having a negative utility. The method used to select the best rules by cost was as follows:

```
repeat
    select the rule with the greatest utility that covers more
    that 50 examples,
    remove examples covered by the most accurate rule,
until the utility of the rule with greatest utility is less than
    1000000.
```

5.3.8 Decision tree

INDCART (a free version of Classification And Regression Trees, CART (Breiman, 1984), supplied by Wray Buntine of NASA's Ames Laboratories) was trained on the PCA representation of the data, with each drug described by the 60 attributes. Paired comparisons were then created, as described for GOLEM, to produce training examples with 120 attributes, which belonged to class 1 if the first drug was more active than the second, and to class 0 if the second drug was more active than the first. The same samples of 1000 examples as used by GOLEM were used to learn the classification trees. As before, the following were used: the Gini splitting criteria, ten fold cross validation, and the 0-SE rule for tree pruning.

5.4 Results

Spearman rank correlation coefficients for the performances of the methods on the training data and testing data are given in Tables 5.3 and 5.4 respectively. There are six sets of test data, each with 31 drugs. Each of the 186 triazines appears once only in only one of the test sets, so all the drugs are tested. The respective training sets comprise the other five sets of data.

5.4.1 Linear regression

Five different regression analyses were performed on six splits of the data, using: the Hansch parameters and their squares; the indicator variables; the indicator variables and the Hansch parameters and squares; the PCAs; the PCAs and their squares. For each regression analysis, the resulting six regression equations have been summarised by a representive correlation equation, in which contains the mean values of the coefficients of any variable that appears in more than one of the six regression equations. For each representive correlation equation, *n* is the number of training set examples, r^2 is the Pearson correlation coefficient and σ is the standard deviation from the regression (r^2 and σ are mean values, and do not characterise the representative equation). The stepwise linear regression on the Hansch parameters, their squares and the indicator variables yields the following representative correlation equation:

Method	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Mean[a] (σ)
LR on Hansch + squares	0.271	0.379	0.295	0.321	0.296	0.238	0.300 (0.048)
LR on 60 PCAs + squares[b]	0.488	0.565	0.553	0.495	0.610	0.530	0.540 (0.046)
neural network on Hansch + I1	0.482	0.693	0.618	0.730	0.686	0.688	0.650 (0.090)
neural network on 60 PCAs	0.862	0.936	0.803	0.781	0.641	0.773	0.799 (0.099)
nearest neighbour on Hansch + I1	1.000	1.000	1.000	1.000	1.000	1.000	1.000 (0.000)
nearest neighbour on 60 PCAs	1.000	1.000	1.000	1.000	1.000	1.000	1.000 (0.000)
CART on 60 PCAs	0.809	0.775	0.802	0.858	0.840	0.816	0.817 (0.029)
GOLEM on 60 PCAs	0.723	0.707	0.672	0.650	0.682	0.666	0.683 (0.027)

[a]Each method was trained on six cross-validation training sets. The mean and standard deviation (σ) of the six performances are given.

[b]Linear regression (LR) on the Hansch parameters and their squares.

Table 5.3 Summary of all methods - Spearman rank correlation coefficient on training sets.

Method	Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Mean[a] (σ)
LR on Hansch + squares[b]	0.463	-0.124	0.314	0.197	0.480	0.304	0.272 (0.220)
LR on 60 PCAs + squares	0.425	0.216	0.724	0.291	0.491	0.529	0.446 (0.181)
neural network on Hansch + I1	0.588	0.103	0.534	0.400	0.438	0.197	0.377 (0.190)
neural network on 60 PCAs	0.329	0.301	0.676	0.399	0.559	0.620	0.481 (0.145)
nearest neighbour on Hansch + I1	0.766	0.535	0.632	0.185	0.461	0.437	0.503 (0.197)
nearest neighbour on 60 PCAs	0.477	0.573	0.438	0.569	0.446	0.606	0.518 (0.073)
CART on 60 PCAs	0.606	0.397	0.708	0.556	0.528	0.396	0.532 (0.121)
GOLEM on 60 PCAs	0.583	0.536	0.321	0.610	0.212	0.325	0.431 (0.166)

[a]Each method was trained on six cross-validation training sets. The mean and standard deviation (σ) of the six performances are given.

[b]Linear regression (LR) on the Hansch parameters and their squares.

Table 5.4 Summary of all methods - Spearman rank correlation coeffecient on testing sets.

$$\log(1/C) = 7.09 + 0.64\pi_3 - 0.87MR_3 - 0.14\pi_3^2 - 0.02\pi_4 - 0.02MR_4^2 - 1.68I\text{-}3 + 0.60I\text{-}5 + 0.59I\text{-}6. \quad \text{[eqn. 5.1]}$$

$n = 155$; $r^2 = 0.57$; $\sigma = 0.51$

The stepwise linear regression on only the Hansch parameters and their squares yields:

$$\log(1/C) = 7.12 + 0.72\pi_3 - 0.15\pi_3^2 - 0.94\Sigma\sigma_{34}. \quad \text{[eqn. 5.2]}$$

$n = 155$; $r^2 = 0.21$; $\sigma = 0.68$

The stepwise linear regression on only the indicator variables gives:

$$\log(1/C) = 7.24 - 1.65I\text{-}3 + 0.67I\text{-}5 + 0.57I\text{-}6. \quad \text{[eqn. 5.3]}$$

$n = 155$; $r^2 = 0.41$; $\sigma = 0.59$

The stepwise linear regression on the 60 PCAs yields:

$$\log(1/C)_N = 0.63 + 0.21PO_1 + 0.30BR_1 + 0.28PL_2 + 0.30\pi A_2 - 0.21SZ_4 + 0.23FL_4 - 0.16\pi D_4 - 0.29\pi A_4 + 0.11\pi D_5 \quad \text{[eqn. 5.4]}$$

$n = 155$; $r^2 = 0.31$; $\sigma = 0.13$.

$\log(1/C)_N$ is $\log(1/C)$ rescaled to lie between 0.1 and 0.9; the PCAs are represented by their two letter abbreviations, with the subscripts denoting the regions. Including the squares of the PCAs in the regression analysis gives:

$$\log(1/C)_N = 0.52 - 0.17HA_1 + 0.19PO_1 + 0.36PL_2 - 0.30PL_2^2 + 0.31SZ_2^2 + 1.31FL_4 - 0.26FL_4^2 - 1.0HD_4^2 - 0.17\pi A_4^2 - 0.81PO_4^2 \quad \text{[eqn. 5.5]}$$

$n = 155$; $r^2 = 0.39$; $\sigma = 0.13$.

5.4.2 Nearest neighbour

The simple nearest neighbour algorithm here is essentially a look-up table. This guarantees 100% accuracy on the training data, by the definition of the method, but, consequently, provides little insight on the relative importance of the features of the drugs.

5.4.3 Neural networks

A neural network similar to one used in the previous neural network study on these data represented by the Hansch parameters (Andrea and Kalayeh, 1991) was implemented to provide a benchmark. Preliminary studies indicated that the neural network exhibited the expected behaviour. In particular, the neural network gave better predictions when hidden units were used, indicating that the activity of the drugs is nonlinearly dependent on the Hansch parameters. The capacity of neural networks to model nonlinear features and cross-terms, as indicated by the improved performance with hidden units, is the major motivation for their application to QSAR analysis. In contrast to a regression analysis, which generates a correlation equation, it is more difficult to interpret the relationship between the activity of a drug and its properties modelled by a neural network with hidden units, because cross-terms are not considered explicitly term by term, but implicitly as a multi-dimensional function. Although the activity of a drug can be plotted as a function of one or two input parameters, with the others held constant, such a multi-dimensional function can not be clearly depicted (Andrea and Kalayeh, 1991; So and Richards, 1992). After the training phase, the predicted activity for any combination of input values can be evaluated, by propagation through the weights of the neural network, but the functional form is not so easily visualised.

One method, developed here, of interpreting a neural network model of a QSAR is to test different combinations of inputs and analyse the common features of the combinations that the neural network predicts to have high activity. By allowing the input values to vary in discrete steps of 0.1, from 0.1 to 0.9, and as the number of input units is small, it was possible to evaluate every possible combination of input values, and thus to find the combination of properties that gives the maximum predicted activity. In this case, the five inputs, π_3, π_4, MR_3, MR_4 and $\Sigma\sigma_{34}$, were allowed to vary between 0.1 and 0.9 (the rescaled minimum and maximum) in steps 0.1, and I-1 was allowed to be either 0.1 or 0.9 (corresponding to the L1210 enzyme data and the Walker enzyme data, respectively). This generated about 3.9 million (2 X 5^9) combinations of input values that were evaluated by propagation through the weights of the neural network. For each cross-validation trial, the weights that gave the best test set performance were used, so that the generalised features would be optimised. These weights were generated in a separate trial; optimisation on the test set performance would not be legitimate for evaluation of the predictive ability, but is useful for investigating the features that are generalised.

The final weights from each cross-validation trial were different. Exhaustive evaluation of the possible combinations of inputs using the five different sets of weights found different combinations of inputs to be highly active. The range of predicted activity also varied, with the weights from the third and sixth trials predicting no combinations to more active than 0.99, whereas the fourth trial predicted more than 6000 combinations to have an activity greater than 0.99. To extract the general features of the combinations that were highly predicted, a small number of the most active combinations were clustered by eye (Table 5.5). In general, for each set of weights, the combinations predicted to be most active are similar to each other, *e.g.*, the weights optimised using third cross-validation set of data predicted 10 combinations to have an activity greater than 0.98, and all 10 combinations have a high π_3 value low π_4, low MR_3, low MR_4 and low $\Sigma\sigma_{34}$. Each trial generated one class of combination of predicted high activity, except for the first trial, which generated two classes. All but one of these classes have a high π_3 value and a low MR_3 value; there is more variation for π_4, MR_4 and $\Sigma\sigma_{34}$.

Hansch parameter	π_3	π_4	MR_3	MR_4	$\Sigma\sigma_{34}$	activity threshold[a]	No. above threshold[b]
set 1a	-	+	+	-	+	0.99	18
set 1b[c]	+	-	-	+	+		
set 2	+	+	-	+	-	0.99	66
set 3	+	-	-	-	-	0.98	10
set 4	+	+	-	+/-	-	0.9994	50
set 5	+	+/-	-	+	-	0.985	60
set 6	+	+/-	-	-	+	0.96	84

[a]The limit of predicted activity is unity. The threshold value was arbitrarily chosen so that the number of combinations with a higher predicted activity was reasonably small, for inspection by eye.

[b]A combination is a set of the five input values, ranging from 0.1 to 0.9 in discrete steps of 0.1. The number of such combinations with a predicted activity higher than the threshold is given in this column.

[c]Two classes of combination were found for set 1, denoted 1a and 1b.

Table 5.5 Classification of combinations of Hansch parameters predicted by a neural network to be more active than a given threshold. - indicates that the value was consistently less than 0.5. + indicates that the value was consistently greater than 0.5. +/- indicates that the value was either greater or less than 0.5

The neural network trained on the PCA representation gave the most accurate test set predictions without hidden units. This would seem to imply that activity is linearly dependent on the PCAs, but this contradicts the linear regression analysis, which indicated that there was some nonlinear dependence on the PCAs. If there is some nonlinear dependence, then the impairment of performance on addition of hidden units is probably due to over-fitting, because of the increase in the number of weights. An analysis of the neural network weights, given in Figure 5.3, suggests the following relationship:

$$\begin{aligned}\log(1/C)_{\mathrm{N}} = {} & - 0.28\pi D_1 - 0.58\pi A_1 + 0.37\sigma_1 + \\ & 0.37BR_1 - 0.60PL_2 + 0.64SZ_2 + 0.37PO_2 + \\ & 0.24BR_2 + 0.16\pi A_3 - 0.25PL_4 + 0.73HD_4 - \\ & 0.46\pi A_4 - 0.55PO_4 + 0.21\pi A_5 + 0.37\sigma_5. \qquad \text{[eqn. 5.6]}\end{aligned}$$

Many of the weights are zero, with no PCA in region 6 having a large weight, but of the ten types of PCA, only flexibility does not appear at all.

5.4.4 Inductive logic programming

GOLEM was only applied using the PCA representation of the data. The rules generated by GOLEM take the form of Prolog clauses, and are readily translated into English. For the six cross-validation runs, 45 rules were found. From these rules, seven consensus rules were generated manually (Table 5.6), from the most common features, assuming that substitutions at each position were independent. The consensus rules are simpler than

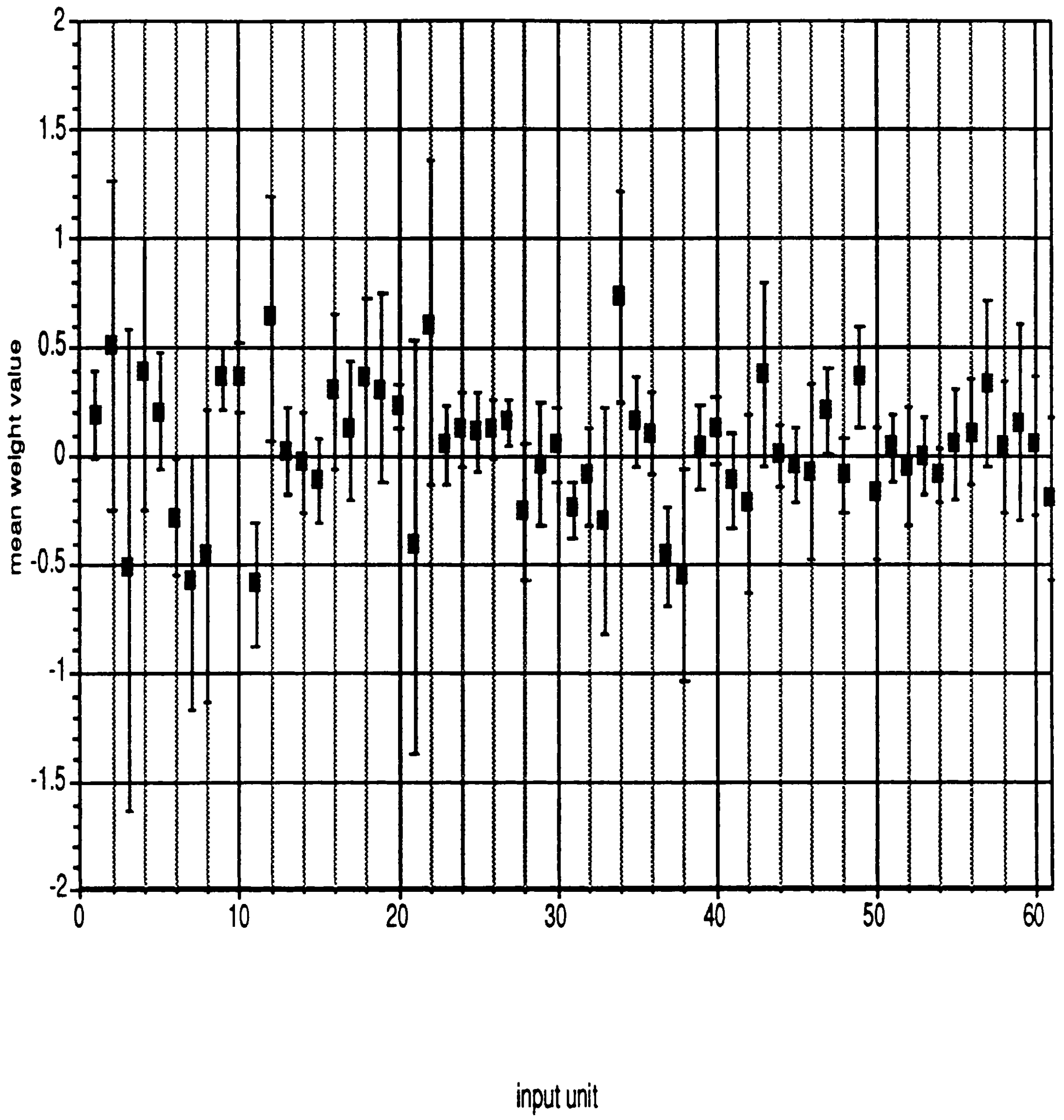


Figure 5.3 The mean and standard deviation (error-bars) of the neural network weights giving the optimal test set performances for the six cross-validation trials.

Rule 1: utility 24615000, accuracy 0.638, coverage[a] 8384

```
great(A,B) :-
        struc3(A,A3,_), polarisable(A3,polarisable1),
        struc3(B,B3,_), not polarisable(B3,polarisable1).
```

English translation - Drug A is better than drug B if:- drug A has a substituent on the first phenyl ring at position 3 with polarisability = 1 and drug B does not.

Rule 2: utility 18764000, accuracy 0.610, coverage 8496

```
great(A,B) :-
        struc3(A,A3,_), pi_donor(A3,pi_donor1),
        struc3(B,B3,_), not pi_donor(A3,pi_donor1).
```

English translation - Drug A is better than drug B if:- drug A has a substituent on the first phenyl ring at position 3 with π-donor = 1 and drug B does not.

Rule 3: utility 17011000, accuracy 0.688, coverage 4404

```
great(A,B) :-
        struc3(A,A3,_), branch(A3,branch0),
        struc3(B,B3,_), not branch(B3,branch0).
```

English translation - Drug A is better than drug B if:- drug A has a substituent on the first phenyl ring at position 3 with branching = 0 and drug B does not.

Table 5.6 Consensus rules

Rule 4: utility 15735000, accuracy 0.630, coverage 5908

```
great(A,B) :-
      struc3(A,A3,X),  h_acceptor(A3,h_acceptor0),
      struc3(B,B3,_),  not h_acceptor(B3,h_acceptor0).
```

English translation - Drug A is better than drug B if:- drug A has a substituent on the first phenyl ring at position 3 with hydrogen-bond acceptor = 0 and drug B does not.

Rule 5: utility 11259000, accuracy 0.564, coverage 8187

```
great(A,B) :-
      struc4(A,A4,_),  polar(A4,polar1),
      struc4(B,B4,_),  not polar(B4,polar1).
```

English translation - Drug A is better than drug B if:- drug A has a substituent on the first phenyl ring at position 4 with polarity = 1 and drug B does not.

Rule 6: utility 15476000, accuracy 0.613, coverage 6110

```
great(A,B) :-
      struc3(A,A3,S1),  not struc3(A,A3,absent),
      struc3(B,B3,absent).
```

English translation - Drug A is better than drug B if:- drug A has a substituent that includes a phenyl ring at position 3 on the first phenyl ring and drug B does not.

Table 5.6 (cont.)

Rule 7: utility 12230000, accuracy 0.580, coverage 8431

```
great(A,B) :-
      struc4(A,A4,S1), not struc4(A,A4,absent),
      struc4(B,B3,absent).
```

English translation - Drug A is better than drug B if:- drug A has a substituent that includes a phenyl ring at position 4 on the first phenyl ring and drug B does not.

[a]The coverage is the number of pair comparisons that the rule covers, and the accuracy is the number of cases for which the rule is correct divided by the coverage. The utility is the sum of the squared differences of ranks, with a positive sign for correctly predicted examples, and a negative sign for incorrect predictions.

Table 5.6 (cont.)

the automatically generated GOLEM rules, and are easier to understand. The consensus rules were tested on the six cross-validation data sets, giving an average Spearman rank correlation coefficient on the training set of 0.498, and an average Spearman rank correlation coefficient on the test set of 0.457. The rank correlations are similar for the training and test sets, indicating that the rules are not over-fitting the data.

The consensus rules can be interpreted to generate the best predicted drug, or drugs. At position 4, rule 5 (Table 5.6) states that there should be a substituent with polarity = 1; and rule 7 states that this group should be connected to a phenyl ring. The rules leave open what type of substituent should be on the second phenyl ring. Although, the best drug in the present study only has a Cl at position 4 (polarity = 3), most of the highly active drugs in the data (drug numbers 246, 241, 240, 239, 234, 233) comply with rules 4 and 7. Position 3 is much more constrained than position 4. Four PCAs are specified: polarisability = 1 (rule 1), π-donor = 1 (rule 2), hydrogen-bond acceptor = 0 (rule 4), and branching = 0 (rule 3); the only substituent that specifies all four attributes is Cl (see Figure 5.4). This is the substituent found in the best drugs in this study and other work (Andrea and Kalayeh, 1991; Silipo and Hansch, 1976). However, rule 6 conflicts with this conclusion and suggests that a second phenyl ring be added. This indicates that it may be possible to do better than Cl by relaxing the condition π-donor = 1 and using a substituent from group A (Figure 5.4) and a second phenyl ring, or relaxing the condition hydrogen-bond acceptor = 1 and using a substituent from group B and a second phenyl ring.

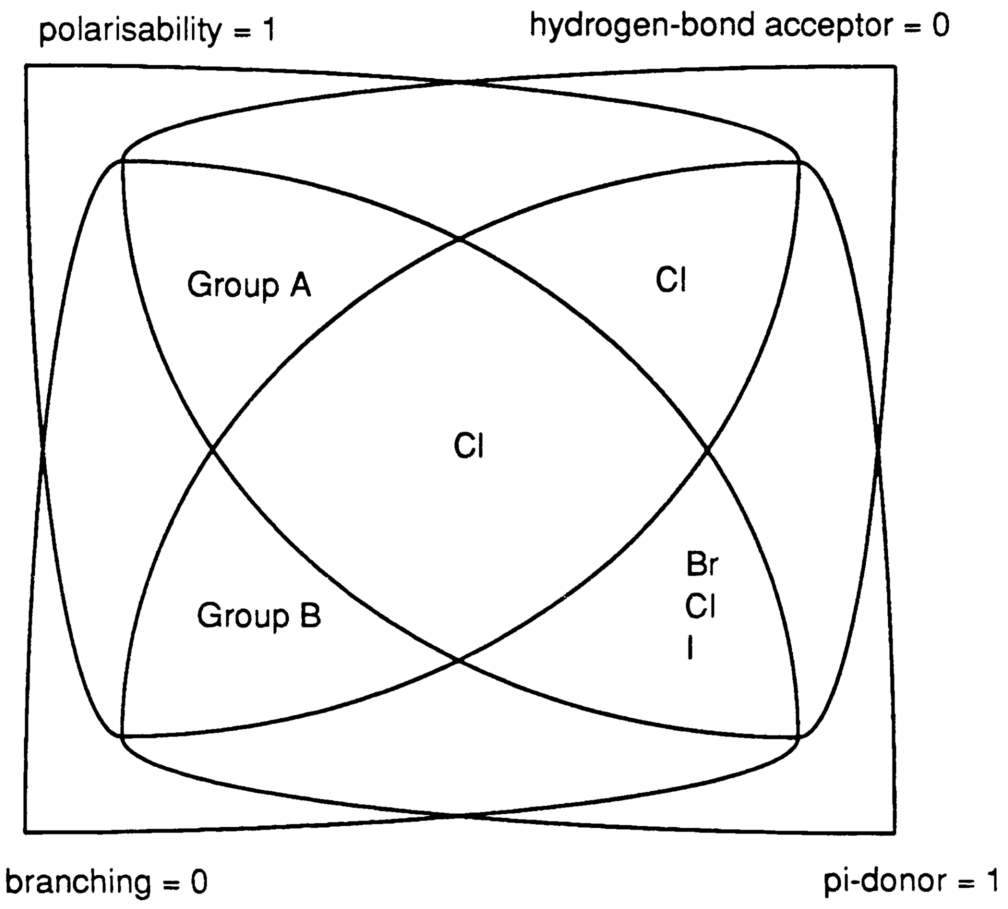


Figure 5.4 Venn diagram of favoured properties for the first substitution at position 3. Cl is the only substituent that has: polarisability = 1, and π-donor = 1, and hydrogen-bond acceptor = 0, and branch = 0. Group A contains the fragments: Cl, $(CH_2)_2$, $(CH_2)_4$, $(CH_2)_6$, CH_2 ,CH_3. Group B: Cl, $NHCOCH_3$, $O(CH_2)_2$, $O(CH_2)_2O$, $O(CH_2)_2O(CH_2)_2O$, $O(CH_2)_3CH_3$, $O(CH_2)_3O$, $O(CH_2)_4$, $O(CH_2)_4O$, $O(CH_2)_5CH_3$, $O(CH_2)_5O$, $O(CH_2)_6O$, $O(CH_2)_6CH_3$, $O(CH_2)_7CH_3$, $OCH_2OCH_2OCH_3$, OCH_3.

5.4.5 Decision tree

The mean number of leaves of the six decision trees (one for each of the six cross-validation trials) was 78. This high number makes the trees quite difficult to interpret and to form a consensus tree. However, examination the most important leaves (the ones with the most examples) suggested that the trees were broadly consistent with the consensus GOLEM rules, with perhaps an additional constraint on the size of substituents at position 3. Figure 5.5 shows a subsection of the tree from run 2.

5.5 **Discussion**

In this study a comparison of methods for deriving QSARs has been presented. Only some QSAR methods have been studied. There are several other established QSAR methods, such as comparative molecular field analysis (CoMFA) (Cramer, *et al.*, 1980), and also other statistical algorithms which have not been examined here. Despite these limitations, with the continuing application of traditional QSAR methods (Debnath, *et al.*, 1993) , and the development of new methods, this study is a stringent assessment of both the more established and newer methods. The linear regression approach of Hansch has had a major impact on the field of QSAR and this type of analysis is still popular. The Hansch parameters are well established and have strong foundations in physical organic chemistry. In this chapter, the new PCA representation has been developed for modelling

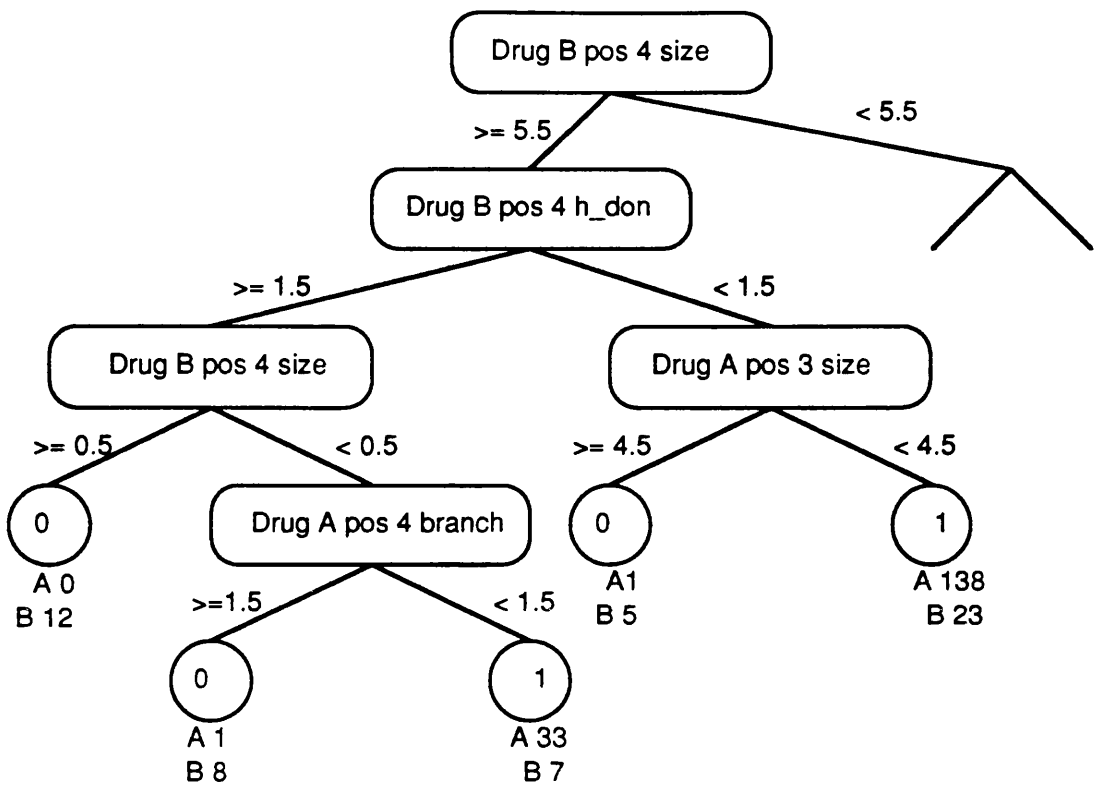


Figure 5.5 Subsection of the binary tree generated on cross-validation run 2. The boxes represent decision nodes and the branches from the nodes alternative possible decisions. The circles represent the leaves (the final decision classes): 1 = drug A is more active than drug B; 0 = drug A is less active than drug B. Underneath the leaves are: A, the number of times drug A is more active; B, the number of times drug A is less active. To classify a drug pair, start at the first decision node (the root) and work down to the leaves, *e.g.*, to test if drug A (250) 3,4-Cl_2 is more active than drug B (93) 4-$(CH_2)_2CON(CH_2CH_2)_2O$: starting at the root, drug B has a 4-substituent with size = 6, so the left branch is chosen; at the next node drug B has a 4-substituent with h_donor = 0, so the right branch is chosen; drug A has a substituent with size = 1, so the right branch is chosen and the decision class is 1, *i.e.*, drug A is predicted to be more active than drug B.

QSARs. Linear regression, neural networks and a nearest neighbour algorithm have been compared using the Hansch parameters and the PCA representation. In each case, the PCA representation gave superior results to the Hansch parameters, although a statistically significant difference was only obtained for the linear regression analysis (test set performance given by the Spearman rank correlation coefficient of 0.446 compared to 0.272). Due to the large variations in performance and the low correlations found in some cases, a detailed analysis of outliers was impractical, but the worst outliers are indicated in Figure 5.6.

The PCA representation may be better than the Hansch parameters, because each of the drugs has a unique PCA representation, whereas there are over fifty pairs of drugs whose Hansch parameters have identical values. In this data set, which is exceptionally large, the Hansch parameters do not distinguish uniquely each drug. This is less likely to be a problem in a smaller data set, but it creates some possible pitfalls in the analysis of these particular data.

One of the surprises of this study was the relatively poor performance of all of the methods compared to what might have expected from the literature (Silipo and Hansch, 1975; Andrea and Kalayeh, 1991) . To reconcile this study with other work, this discrepancy is discussed. The linear regression studies (Silipo and Hansch, 1975), have suggested training set performances of n = 83, $r^2 = 0.819$ and σ = 0.328 for the Hansch parameters and their squares; and n = 244; $r^2 = 0.852$; σ = 0.377, with the added use of

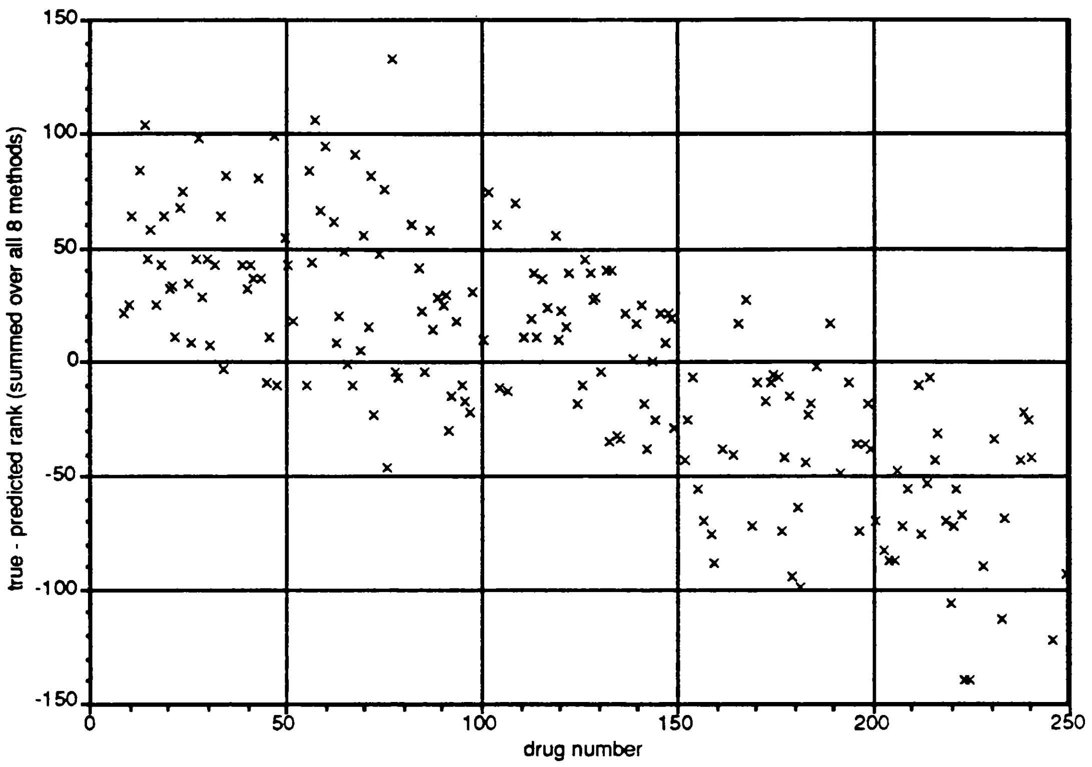


Figure 5.6 Outlier analysis. The difference between the true and predicted rank (summed over all 8 methods: CART, GOLEM, regression on Hansch parameters, regression on PCAs, nearest neighbour on Hansch parameters, nearest neighbour on PCAs, neural network on Hansch parameters, neural network on PCAs) for each drug.

indicator variables. On a different split of the data (Andrea and Kalayeh, 1991), lower training set performances were obtained for a linear regression (used as a benchmark): n = 100, r^2 = 0.380, σ = 0.705, for the Hansch parameters and their squares; and n = 100; r^2 = 0.700, σ = 0.498, with the inclusion of indicator variables. For the six random splits of the data in this study, the average training set performance of the linear regression on the Hansch parameters, their squares and the indicator variables was r^2 = 0.568 (n = 155). The lowest performance was for split number 3, which gave a correlation of r^2 = 0.484, and the highest performance was r^2 = 0.664 for split number 4. In a previous comparative study (Andrea and Kalayeh, 1991), three selected splits of the data gave training set performances for similar regressions of r^2 = 0.700 (n = 100), r^2 = 0.744 (n = 66) and r^2 = 0.591 (n = 57). The original study (Silipo and Hansch, 1975), gave a higher performance of r^2 = 0.852 (n = 244). These differences indicate that estimates of performances can be dependent on the selection of data, and in benchmarking new QSAR methods it is therefore important to avoid any bias towards one particular selection of the data.

Estimates of test set performances are even more susceptible to biases in the data. This is particularly relevant in this study, where many pairs of the drugs are identically described by the Hansch parameters, and also, in many cases, have very similar activities. This is not generally the case for data used in QSAR studies. For this data set, a leave-one-out cross-validation procedure will over-estimate the true test set performance on data that are less similar to the training data, especially of methods which fit the training data well, because much of the test data is essentially equivalent to training data. In a previous study comparing neural networks and linear regression on this data set (Andrea and Kalayeh, 1991), four estimates of the test set performance of neural networks were: $r^2 = 0.804$ (100 drugs in the training set and 32 in the test set); $r^2 = 0.672$ (66 drugs in the training set and 66 in the test set); $r^2 = 0.511$ (57 drugs in the training set and 56 in the test set); $r^2 = 0.787$ (leave-one-out cross-validation on 132 drugs). These particular splits (100/32, 66/66, and 57/56) were deliberately selected using a cluster analysis of the data prior to the assessment of the neural network, to ensure that every point in the test set had points in the training set in its vicinity. While this may ensure that the training set is well distributed in the subspace of independent variables, the estimates of the test set performances, on truly unseen data, may be too optimistic, because of the large number of identical pairs.

The variability of the performance with different splits of the data has made it difficult to discern statistically significant differences between the QSAR methods. It is clear that a linear regression only on the Hansch parameters has a poor predictive capacity, and that the indicator variables give a great improvement. These indicator variables can be valuable in QSAR work, and the identification of gross structural features such as flexible or rigid bridges is of obvious use in drug design. However, for the development of general QSAR methods for predictive purposes, as has been acknowledged (Hansch and Fukunga, 1977), indicator variables are not ideal, because they are not general and they are not amenable to automatic assignment, or even identification. The PCA representation tries to identify features that are of general importance in QSARs, and is amenable to automatic assignment, although this is not currently implemented. While the more problem-specific indicator variables are more successful in modelling the data in this study, the PCA representation may also give an improvement over the Hansch parameters by themselves, and is generally applicable.

This study has shown that the newer methods of neural networks and GOLEM perform comparably with other more established methods such as linear regression, nearest neighbour algorithms and decision trees. Even the largest data sets available for QSAR analysis are probably not of sufficient size to allow significant discrimination between current QSAR methods or between the Hansch parameter and PCA representations. Both neural networks and GOLEM have the capacity to model complex relationships, although this power may not have fully exploited in this study. The generality of PCAs has been demonstrated with a wide extension of the representation to cover a range of substituents and the representation is useful in providing understandable rules about drug-receptor interactions.

Chapter 6

Conclusions

This thesis has used neural networks to model molecular structure and function. Two important areas of biomolecular modelling have been studied: sequence analysis and QSAR. Particular care has been taken to provide rigourous comparisons of neural networks with traditional statistical methods and with more modern approaches from the field of artificial intelligence.

In Chapter 3, a neural network trained to recognise an ATP-/GTP-binding motif, performed with an accuracy of 78% on test data. This was more accurate than a simple search algorithm, PROMOT (Sternberg, 1991b), which correctly identified 197 segments, but also identified 152 false positives. However, a statistical method was developed, which performed with an accuracy of 84%. This study confirms theoretical considerations which indicate that, despite many applications to sequence analysis, neural networks with no hidden units are not expected to perform better than traditional statistical techniques.

In Chapter 4, the QSAR for the inhibition of *E. coli* DHFR by pyrimidines was derived using: neural networks, linear regression, a nearest neighbour algorithm, CART (a decision tree), and GOLEM (a modern machine learning method using ILP). In Chapter 5, the inhibition of rodent DHFR by triazines provided the data for a second study. Two different representations of the data were used. The top results, in cross validation trials, as measured by the Spearman rank correlation coefficient on test data were:

For the pyrimidine data:

nearest neighbour using the Hansch parameters	0.762
neural network using the PCA representation	0.717
linear regression using the Hansch parameters	0.693
GOLEM using the PCA representation	0.692

For the triazine data:

CART using the PCA representation	0.532
nearest neighbour using the PCA representation	0.519
nearest neighbour using the Hansch parameters	0.503
neural network using the PCA representation	0.481

Statistically significant differences were not apparent, given the size of the data sets. However, the level of insight provided varies for the different representations and methods. Although the nearest neighbour algorithm performs well, it does not indicate which parameters are of importance. The PCA representation uses parameters that are widely understood, such as size, flexibility and hydrogen-bonding capacities; the Hansch parameters, molar refactivity and hydrophobicity, are less widely understood. In Chapter 5, a method was developed that used the weights of the neural network to hypothesise highly active drugs (as described by their PCAs). In commercial QSAR packages, such as Tsar^{TM} produced by Oxford Molecular Ltd., a variety of data analysis techniques are provided. It is only a matter of time before neural networks are added to such computational toolboxes.

This thesis presents several comparative studies, demonstrating that neural networks can provide useful empirical models, but that other statistical models can also be of equal predictive value. The major advantage of neural networks is the automatic modelling of nonlinear and cross-terms. This power is probably best exploited in the analysis of large data sets, where there are complex relationships. Future work in this area should concentrate on the interpretation of the weights of the neural network. One strategy, explored in Chapter 5, was to evaluate many different possible inputs using the converged weights. For larger neural networks, this strategy could be extended with the use of an efficient search algorithm.

Understanding the models created by neural networks also depends on the representation of the data. The PCA representation of drug molecules used in Chapters 4 and 5 provided useful information, complementing the Hansch parameters. Further work in this direction would include the investigation of other properties, and automating the calculation of the PCAs.

Advances in neural networks will come from increases in computing power, hardware implementations of neural computers, and from a greater understanding of learning processes. With these advances in prospect, and the continual requirement for data analysis, research into neural networks and their applications will remain an exciting area of science.

Appendix 1

Publications connected with this thesis

Prediction of ATP-binding motifs: a comparision of a perceptron-type neural network and a consensus sequence method. Hirst, J.D. and Sternberg, M.J.E. (1991) *Protein Engineering*, **4**, 615-623.

Prediction of Structural and Functional Features of Protein and Nucleic Acid Sequences by Artificial Neural Networks. Hirst, J.D. and Sternberg, M.J.E. (1992) *Biochemistry*, **31**, 7211-7218.

New Approaches to QSAR: Neural Networks and Machine Learning. King, R.D., Hirst, J.D. and Sternberg, M.J.E. (1993) *Perspectives in Drug Discovery and Design*, in press.

Quantitative Structure-Activity Relationships: Neural Networks and Inductive Logic Programming Compared Against Statistical Methods I. The Inhibition of Dihydrofolate Reductase by Pyrimidines. Hirst, J.D., King, R.D. and Sternberg, M.J.E. (1993), *J. Med. Chem.*, submitted.

Quantitative Structure-Activity Relationships: Neural Networks and Inductive Logic Programming Compared Against Statistical Methods II. The Inhibition of Dihydrofolate Reductase by Triazines. Hirst, J.D., King, R.D. and Sternberg, M.J.E. (1993), *J. Med. Chem.*, submitted.

Drug Design by Machine Learning: A Comparative Study. King, R.D., Hirst, J.D. and Sternberg, M.J.E. (1993) *Machine Learning*, submitted.

Abstracts:

Quantitative Structure Activity Relationships of Dihydrofolate Reducatase Inhibitors by Neural Networks. Hirst, J.D. and Sternberg, M.J.E. (1993) *Protein Engineering*, **6**, 107.

Appendix 2

FORTRAN code for a general backpropagating neural network

```
C*******************************************************
C FORTRAN version of a general backpropagating neural network
C
C written by Jonathan D. Hirst
C
C Biomolecular Modelling Laboratory
C Imperial Cancer Research Fund
C 44 Lincoln's Inn Fields
C London WC2A 3PX
C United Kingdom
C
C Copyright 1993
C
C Use of this program must be acknowledged
C
C*******************************************************
C*******************************************************
C Subroutines:
C     main program          - set parameters for neural network and
C                    data
C     SUBROUTINE INITWT     - initialise weights
C     SUBROUTINE INITACT    - initialise activations
C     SUBROUTINE SHACT      - display activations (diagnostic)
C     SUBROUTINE PROP       - propagate activations through the
C                    weights of the neural network
C     SUBROUTINE BACK       - calculate errors and backpropagate
C                    them
C     SUBROUTINE UPDATE     - changes weights by calculated amounts
C     SUBROUTINE TRAINPER   - evaluate performance on training
C                    (and test) data
C     SUBROUTINE CORREL     - calculate Pearson r-squared
C                    correlation coefficients
C*******************************************************
C*******************************************************
C Program scheme:
C     main program    - set parameters describing the neural
C               network and the data
C     INITWT          - initialise the weights
C     training cycle over the number of learning steps:
C          BACK:
C               for each example in the training set:-
C               INITACT - initialise activations
C               read in training example
C               PROP - propagate through neural network
C               calculate errors
C               calculate weight changes
C          UPDATE - change the weights
C          TRAINPER - calculate performance on training data
C               for each example in the training set:-
C                    INITACT - initialise activations
C                    read in training example
C                    PROP - propagate example through
C                         neural network
C                    calculate squared difference between
C                    desired and actual output
C               CORREL - calculate correlation between
C                    desired and actual outputs
C               MONITOR - calculate performance on monitor set
```

```
C                   to  determine  convergence  criterion
C          TRAINPER - performance on test data
C               (same procedure as for training data)
C*****************************************************
C*****************************************************
C Input to program:
C     File 'train.dat' containing the training set:
C          real numbers separated by white space (tabs or spaces)
C          For each example in the training set -
C          Desired output (true classification) followed by
C          Input (as many numbers as input units)
C     File 'test.dat' containing the test set:
C          same format as for 'train.dat'
C
C     User prompted for:-
C
C     Integer number of learning steps
C     Integer number of examples in the training set
C     Integer number of examples in the test set
C     Integer number of layers
C     Integer number of units in each layer
C Output:
C     File 'outtrain.dat'
C          Pearson r-squared correlation coefficient between
C          desired and predicted outputs on the training set
C          at every 500 cycles
C     File 'outtest.dat'
C          Pearson r-squared correlation coefficient between
C          desired and predicted outputs on the test set
C          at every 500 cycles
C*****************************************************
C*****************************************************
C Variables:
C
C A(I)           - 1-dimensional array of desired outputs DOUBLE
C          PRECISION
C          I indexes examples in test set. Used in MONITOR
C ACT(K,I)  - 2-dimensional array of activations - DOUBLE PRECISION
C          K indexes the layer: I indexes the units within a
C          layer. Maximum number of layers of units is 4. Maximum
C          number of units within a layer is 60. The 0th unit
C          refers to the bias. These maximum levels are set to
C          reduce memory requirements, but can be increased by
C          changing the declaration statements at the start of
C          each routine.
C ACTUAL(I) - 1-dimensional array of desired outputs DOUBLE
C          PRECISION
C          I indexes the examples in the training/test set
C AVA            - real number average (mean) of data set A - DOUBLE
C          PRECISION
C AVB            - real number average (mean) of data set B - DOUBLE
C          PRECISION
CC          - Pearson r-squared correlation coefficient on the
C          monitor DOUBLE PRECISION
CCOR        - Pearson r-squared correlation coefficient DOUBLE
C          PRECISION
C D(K,I)    - 2-dimensional array of errors DOUBLE PRECISION
C          K indexes the layer. I indexes the unit within the
```

```
C          layer.
C ETA      - real number learning rate DOUBLE PRECISION
C I        - dummy variable INTEGER
C ICOUNT   - dummy variable INTEGER
C IP       - dummy variable INTEGER
C ISEED         - seed for random number generator INTEGER
C J        - dummy variable INTEGER
C JP       - dummy variable INTEGER
C K        - dummy variable INTEGER
C KK       - dummy variable INTEGER
C KP       - dummy variable INTEGER
C M        - dummy variable INTEGER
C MAXI(I)  - 1-dimensional array containing the number
C          iterations at current optimal performance of the
C          monitor set for each example in the test set INTEGER
C          I indexes the examples in the test set.
C N        - dummy variable INTEGER
C NAME          - string containing filename CHARACTER*80
C NAMETAG  - Integer: flag: 1 for training set: 0 for test set
C NGEAR         - Integer: flag: 1 to use Gear algorithm: 0 to use
C          standard backpropagation algorithm
C NJACK         - Integer: number of examples in test set for jack-
C          knife analysis in MONITOR
C NJM           -Integer: NJACK - 1
C NTRAIN   - Integer: number of examples in the training set.
C          Used in MONITOR
C NUM_LAYERS    - Integer: number of layers INTEGER*4
C NUM_LEARN_STEPS     - Integer: number of learning steps in the
C          training cycle of backpropagation INTEGER*4
C NUM_SETS - Integer: number of examples in test/training set
C NUM_TE_SETS   - Integer: number of examples in test set
C NUM_TR_SETS   - Integer: number of examples in the training set
C          - INTEGER*4
C NUM_UNITS(I)  - 1-dimensional array of the number of units in each
C               layer INTEGER*4
C P(I)     - 1-dimensional array of predicted outputs DOUBLE
C          PRECISION
C          I indexes examples in test set. Used in MONITOR
C PRED(I)  - 1-dimensional array of predicted outputs DOUBLE
C          PRECISION
C          I indexes examples in training/test set
C          I indexes the layer.
C SUM           - real number product of activations and weights -
C          DOUBLE PRECISION
C TEACT         - desired output for test data DOUBLE PRECISION
C          Used in MONITOR
C TEPRED   - predicted output for test data DOUBLE PRECISION
C          Used in MONITOR
C TMPTRACT(I)     - 1-dimensional temporary store DOUBLE PRECISION
C TMPTRPRED(I)    - 1-dimensional temporary store DOUBLE PRECISION
C TOPACT(I) - 1-dimensional array containing the desired output
C          for each test set example for use in the MONITOR
C          routine DOUBLE PRECISION
C          I indexes the examples in the test set.
C TOPCOR(I) - 1-dimensional array containing the current highest
C          correlation coefficient on the monitor set -
C          DOUBLE PRECISION
C          I indexes the examples in the test set. For each
```

```
C           example in the test set, the performance on the
C           remainder is calculated in the MONITOR routine.
C TOPPRED(I)       - 1-dimensional array containing the predicted output
C           for each test set example at the highest performance
C           on the monitor set DOUBLE PRECISION
C           I indexes the examples in the test set.
C TOPTRACT(I,J)    - 2-dimensional array containing the desired output
C           for each of the training set for use in the MONITOR
C           routine DOUBLE PRECISION
C           I indexes the examples in the test set. J indexes the
C           examples in the training set
C  TOPTRPRED(I,J)- 2-dimensional array containing the predicted output
C           for each of the training set for use in the MONITOR
C           routine DOUBLE PRECISION
C           I indexes the examples in the test set. J indexes the
C           examples in the training set.
C TOT_ERR   - real number sum of the squares of the errors -
C           DOUBLE PRECISION
C TR_CL(I)  - 1-dimensional array of desired ouputs - DOUBLE
C           PRECISION
C           I indexes the number of output units. Maximum number is
C           500 (may be increased).
C UP        - real number numerator of Pearson r-squared correlation
C           coefficient DOUBLE PRECISION
C UPWT(K,J,I)      - 3-dimensional array of changes to be made to weights
C           - DOUBLE PRECISION
C           Indexes, maximum levels as for WT(K,J,I)
C VA        - real number variance of data set A - DOUBLE PRECISION
C VB        - real number variance of data set B - DOUBLE PRECISION
C WT(K,J,I) - 3-dimensional array of weights - DOUBLE PRECISION
C           K indexes the layer, so WT(0,J,I) refers to the weights
C           connecting the input layer to the 1st hidden layer.
C           Maximum number of layers of weights is 3. J indexes the
C           number of units in the 2nd layer, I indexes the number
C           of units in the 1st layer. The maximum number of units
C           per layer is 60. These maximum levels can be
C           increased by changing the declaration statements
C           at start of each routine. They are set to to reduce
C           memory requirements.
C*******************************************************
C*******************************************************
C Common blocks:
C
C VAR            - used for: main program
C                            INITWT
C                            INITACT
C                            SHACT
C                            PROP
C                            BACK
C                            UPDATE
C                            TRAINPER
C                            CORREL
C VARB           - used for: main program
C                            UPDATE
C VARMON    - used for:      TRAINPER
C                            MONITOR
C VCOR           - used for: TRAINPER
C                            CORREL
```

```
C                        MONITOR
CVT          - used for: main  program
C                        TRAINPER
C                        CORREL
C*****************************************************
C*****************************************************
C  Channels:
C            8  - writing  performance  on  training  set
C            9  - writing  performance  on  test  set
C                  Performance  =  Pearson  r-squared  correlation
C                  coefficient
C            13 - reading  old  weights
C            14 - reading  either  training  or  test  set
C*****************************************************
C*****************************************************
C Notes:
C
C     The 0th unit in each layer is the bias of that layer.
C
C     Double  precision  numbers  are  required  for  the  Gear  algorithm.
C     Tests  indicated  that  calculations  with  double  precision
C     numbers  were  not  much  slower  than  with  single  precision.
C
C     Documentation  for  the  Gear  algorithm  is  not  included  here,
C     but  in  a  separate  suite  of  routines.
C*****************************************************
C*****************************************************
C  References:
C     Rumelhart, D.E. and  McClelland, J.L. (1986) Parallel
C           Distributed  Processing.  MIT  Press,  Cambridge,  MA.
C     Gear, C.W. (1971) Numerical  Initial  Value  Problems  in
C           Ordinary  Differential  Equations.  Prentice-Hall,
C           Englewood  Cliffs,  NJ.
C*****************************************************
C*****************************************************
C  main  program
C*****************************************************
C*****************************************************
C  Declarations  of  variables
C*****************************************************
       REAL*8 WT(0:2,0:60,0:60), ACT(0:3,0:60), UPWT(0:2,0:60,0:60)
      REAL*8 TR_CL(500)
      INTEGER*4 NUM_UNITS(0:4)
      INTEGER*4 NUM_LAYERS
      INTEGER*4 K
      INTEGER*4 NUM_LEARN_STEPS, NUM_TR_SETS, NUM_TE_SETS
      INTEGER NAMETAG,NGEAR
C*****************************************************
C  Common  blocks
C*****************************************************
      COMMON /VAR/ WT,ACT,TR_CL,NUM_LAYERS,NUM_UNITS,NUM_TR_SETS
      COMMON /VARB/ UPWT
      COMMON /VT/ NUM_TE_SETS
C*****************************************************
C  Open  new  files  for  writing  output
C*****************************************************
       OPEN(8,FILE='outtrain.dat',STATUS='NEW',FORM='FORMATTED')
      OPEN(9,FILE='outtest.dat',STATUS='NEW',FORM='FORMATTED')
```

```
C*******************************************************
C Flag for method to use: 1 = Gear; 0 = standard backpropagation
C*******************************************************
   NGEAR = 1
C*******************************************************
C Set up neural network architecture
C*******************************************************
    WRITE(6,*) 'HOW MANY LAYERS ?'
    READ(5,*) NUM_LAYERS

   DO 10 K = 0, NUM_LAYERS - 1
        WRITE(6,*)
        WRITE(6,*) 'HOW MANY UNITS IN LAYER ', K, ' ?'
        READ(5,*) NUM_UNITS(K)
10  CONTINUE
C*******************************************************
C Initialise weights to random numbers between -0.33 and +0.33
C*******************************************************
   CALL INITWT
C*******************************************************
C Define parameters describing data
C*******************************************************
      WRITE(6,*) 'How many learning steps ?'
    READ(5,*) NUM_LEARN_STEPS

      WRITE(6,*) 'How many training sets ?'
    READ(5,*) NUM_TR_SETS

     WRITE(6,*) 'How many testing sets ?'
    READ(5,*) NUM_TE_SETS
C*******************************************************
C Training cycle
C*******************************************************
C*******************************************************
C Select method: Gear algorithm or standard backpropagation
C*******************************************************
   IF (NGEAR.EQ.1) THEN
     CALL MAIN
   ELSE
     DO 400 K = 1, NUM_LEARN_STEPS
        CALL BACK
        CALL UPDATE
C*******************************************************
C Calculate performances of training and test sets
C*******************************************************
          IF (MOD(K,500).EQ.0) THEN
           NAMETAG = 0
            CALL TRAINPER(NAMETAG, K)
           NAMETAG = 1
            CALL TRAINPER(NAMETAG, K)
          ENDIF
400  CONTINUE
   ENDIF
   CLOSE(8)
   CLOSE(9)
   STOP
   END
C*******************************************************
```

```
C  Subroutine  to  initialise  weights
C*********************************************************
    SUBROUTINE INITWT
C*********************************************************
C  Declaration  of  global  variables
C*********************************************************
      REAL*8 WT(0:2,0:60,0:60), ACT(0:3,0:60)
     REAL*8 TR_CL(500)
     INTEGER*4 NUM_UNITS(0:4)
    INTEGER*4 NUM_LAYERS, NUM_TR_SETS
C*********************************************************
C  Common  block
C*********************************************************
    COMMON /VAR/ WT,ACT,TR_CL,NUM_LAYERS,NUM_UNITS,NUM_TR_SETS
C*********************************************************
C  Declaration  of  routine  specific  variables
C*********************************************************
    INTEGER ISEED,K,J,I
C*********************************************************
C  Option  to  read  weights  from  a  file
C       commented  out
C*********************************************************
c           open(13,file='oldweights.dat',status='old',form='formatted')
C*********************************************************
C  Seed  randon  number  generator
C       For  same  random  starting  weights  set  ISEED  to  the  same  large
C  integer  rather  than  the  time  in  seconds
C*********************************************************
    ISEED = SECNDS(0.0)
    DO 50 K = 0, NUM_LAYERS - 1
        DO 30 J = 1, NUM_UNITS(K + 1)
            DO 20 I = 0, NUM_UNITS(K)
c                   read(13,*) wt(k,j,i)
                WT(K,J,I) = ((2.0 * RAN(ISEED)) - 1.0)/3.0
20          CONTINUE
30      CONTINUE
C*********************************************************
C  No  connection  between  biases  [WT(K,0,I)]  and  previous  layers
C*********************************************************
        DO 40 I = 0, NUM_UNITS(K)
            WT(K,0,I) = 0.0
40      CONTINUE
50  CONTINUE

c       close(13)

    RETURN
    END
C*********************************************************
C  Subroutine  to  initialise  activations
C*********************************************************
    SUBROUTINE INITACT
C*********************************************************
C  Declaration  of  global  variables
C*********************************************************
      REAL*8 WT(0:2,0:60,0:60), ACT(0:3,0:60)
     REAL*8 TR_CL(500)
     INTEGER*4 NUM_UNITS(0:4)
```

```
      INTEGER*4 NUM_LAYERS, NUM_TR_SETS
C*************************************************************
C Common block
C*************************************************************
      COMMON /VAR/ WT,ACT,TR_CL,NUM_LAYERS,NUM_UNITS,NUM_TR_SETS
C*************************************************************
C Declaration of routine specific variables
C*************************************************************
      INTEGER*4 M, N

      DO 70 N = 0, NUM_LAYERS - 1
        DO 60 M = 1, NUM_UNITS(N)
          ACT(N,M) = 0.0
60      CONTINUE
C*************************************************************
C Biases have activation of 1.0
C*************************************************************
        ACT(N,0) = 1.0
70    CONTINUE

      RETURN
      END
C*************************************************************
C Subroutine to show activations - for diagnostic purposes
C      Not called in this version of the code
C*************************************************************
      SUBROUTINE SHACT
C*************************************************************
C Declaration of global variables
C*************************************************************
        REAL*8 WT(0:2,0:60,0:60), ACT(0:3,0:60)
        REAL*8 TR_CL(500)
        INTEGER*4 NUM_UNITS(0:4)
      INTEGER*4 NUM_LAYERS, NUM_TR_SETS
C*************************************************************
C Common block
C*************************************************************
      COMMON /VAR/ WT,ACT,TR_CL,NUM_LAYERS,NUM_UNITS,NUM_TR_SETS
C*************************************************************
C Declaration of routine specific variables
C*************************************************************
      INTEGER*4 M, N

      DO 70 N = 0, NUM_LAYERS - 1
         WRITE(6,*) (ACT(N,M),M = 0, NUM_UNITS(N))
70    CONTINUE

      RETURN
      END
C*************************************************************
C Subroutine to propagate input activations through the neural network
C*************************************************************
      SUBROUTINE PROP
C*************************************************************
C Declaration of global variables
C*************************************************************
        REAL*8 WT(0:2,0:60,0:60), ACT(0:3,0:60)
        REAL*8 TR_CL(500)
      INTEGER*4 NUM_UNITS(0:4)
     INTEGER*4 NUM_LAYERS, NUM_TR_SETS
C*******************************************************
C Common block
C*******************************************************
     COMMON /VAR/ WT,ACT,TR_CL,NUM_LAYERS,NUM_UNITS,NUM_TR_SETS
C*******************************************************
C Declaration of routine specific variables
C*******************************************************
      INTEGER*4 KP, JP, IP
      REAL*8 SUM
C*******************************************************
C KP runs over the number of layers
C The first layer is the 0th layer
C The loop is executed with        KP = 0 for a  2  layer  network
C                                  KP = 1 for a  3  layer  network
C JP is the number of units in the (KP+1)th layer
C      JP starts at 1 to exclude the bias, which is not connected
C      to the previous layer.
C IP is the number of units in the (KP)th layer.
C*******************************************************
     DO 100 KP = 0, NUM_LAYERS - 2
       DO 90 JP = 1, NUM_UNITS(KP+1)
        SUM = 0.0
C*******************************************************
C The weighted sum (including bias)
C*******************************************************
         DO 80 IP = 0, NUM_UNITS(KP)
              SUM = SUM + (ACT(KP,IP)*WT(KP,JP,IP))
80       CONTINUE
C*******************************************************
C The logistic function
C*******************************************************
          ACT(KP+1,JP) = 1.0 / (1.0 + DEXP(-SUM))
90     CONTINUE
100  CONTINUE

      RETURN
     END
C*******************************************************
C Subroutine to calculate errors and weight changes
C*******************************************************
     SUBROUTINE BACK
C*******************************************************
C Declaration of global variables
C*******************************************************
       REAL*8 WT(0:2,0:60,0:60), ACT(0:3,0:60), UPWT(0:2,0:60,0:60)
      REAL*8 TR_CL(500)
      INTEGER*4 NUM_UNITS(0:4)
     INTEGER*4 NUM_LAYERS, NUM_TR_SETS
C*******************************************************
C Common blocks
C*******************************************************
     COMMON /VAR/ WT,ACT,TR_CL,NUM_LAYERS,NUM_UNITS,NUM_TR_SETS
     COMMON /VARB/ UPWT
C*******************************************************
C Declaration of routine specific variables
C*******************************************************
```

```
      REAL*8 D(0:6,6)
      INTEGER*4 KP, JP, IP, J, ICOUNT, K
      REAL*8 SUM
      CHARACTER*80 NAME
C***************************************************************
C Output to screen to reassure user that program is running
C***************************************************************
      ICOUNT = ICOUNT + 1
      IF (MOD(ICOUNT,500).EQ.0) THEN
        WRITE(6,*) ICOUNT
      ENDIF
C***************************************************************
C Initialise weight increments to be zero
C***************************************************************
      DO 80 KP = 0, NUM_LAYERS - 1
        DO 70 JP = 1, NUM_UNITS(KP+1)
          DO 60 IP = 0, NUM_UNITS(KP)
            UPWT(KP,JP,IP) = 0.0D0
 60       CONTINUE
 70     CONTINUE
 80   CONTINUE
C***************************************************************
C Calculate weight increments over all the training set
C****************************************************************
       NAME = 'train.dat'
      OPEN(14,FILE=NAME,STATUS='OLD',FORM='FORMATTED')
      DO 190 J = 1, NUM_TR_SETS
C***************************************************************
C Initialise activations
C***************************************************************
        CALL INITACT
C***************************************************************
C Read in next example from training set
C***************************************************************
         READ(14,*) TR_CL(1), (ACT(0,K), K = 1, NUM_UNITS(0))
C***************************************************************
C Propagate forward
C***************************************************************
        CALL PROP
C***************************************************************
C Calculate deltas (errors) for the units in the output layer
C      KP - number of the output layer
C      JP - number of units in the output layer
C      D(KP-1,JP) - the errors associated with unit JP
C              Indexed by KP-1, because units in the input layer
C              do not have an error associated with them.
C***************************************************************
        KP = NUM_LAYERS - 1
        DO 110 JP = 1, NUM_UNITS(KP)
         D(KP-1,JP) = ACT(KP,JP)*(1.0-ACT(KP,JP))*(TR_CL(JP)-ACT(KP,JP))
 110    CONTINUE
C***************************************************************
C Calculate deltas (errors) for the previous layers
C      KP - number of the layer - not executed for layer 0
C              (the input layer)
C      JP - number of units in (KP)th layer
C      IP - number of units in the (KP+1)th layer
C      SUM - the weighted sum of the errors at the next layer
```

```
C***********************************************************
      DO 150 KP = NUM_LAYERS - 2, 1, -1
        DO 140 JP = 1, NUM_UNITS(KP)
         SUM = 0.0
          DO 130 IP = 1, NUM_UNITS(KP+1)
             SUM = SUM + (WT(KP,IP,JP) * D(KP,IP))
 130      CONTINUE
           D(KP-1,JP) = ACT(KP,JP) * (1.0 - ACT(KP,JP)) * SUM
 140    CONTINUE
 150  CONTINUE
C***********************************************************
C Add to weight increments
************************************************************
      DO 180 KP = 0, NUM_LAYERS - 1
        DO 170 JP = 1, NUM_UNITS(KP+1)
          DO 160 IP = 0, NUM_UNITS(KP)
               UPWT(KP,JP,IP)=UPWT(KP,JP,IP)+(D(KP,JP)*ACT(KP,IP))
 160      CONTINUE
 170    CONTINUE
 180  CONTINUE

 190 CONTINUE
     CLOSE(14)

     RETURN
     END
C***********************************************************
C Subroutine to add the weight changes to the weights (and biases)
C***********************************************************
     SUBROUTINE UPDATE
C***********************************************************
C Declaration of global variables
C***********************************************************
      REAL*8 WT(0:2,0:60,0:60), ACT(0:3,0:60), UPWT(0:2,0:60,0:60)
      REAL*8 TR_CL(500)
      INTEGER*4 NUM_UNITS(0:4)
     INTEGER*4 NUM_LAYERS, NUM_TR_SETS
C***********************************************************
C Common blocks
C***********************************************************
     COMMON /VAR/ WT,ACT,TR_CL,NUM_LAYERS,NUM_UNITS,NUM_TR_SETS
     COMMON /VARB/ UPWT
C***********************************************************
C Declaration of routine specific variables
C***********************************************************
     REAL*8 ETA
      INTEGER*4 KP,JP,IP
C***********************************************************
C Set the learning rate
C***********************************************************
     ETA = 0.25
C***********************************************************
C Change weights and biases
************************************************************
     DO 280 KP = 0, NUM_LAYERS - 1
       DO 270 JP = 1, NUM_UNITS(KP+1)
         DO 260 IP = 0, NUM_UNITS(KP)
            WT(KP,JP,IP) = WT(KP,JP,IP)+(ETA*UPWT(KP,JP,IP))
```

```
260     CONTINUE
270   CONTINUE
280 CONTINUE

    RETURN
    END
C*****************************************************************
C Subroutine to calculate performance of neural network on training
C      and test data
C      NAMETAG = 0 is for training data
C      NAMETAG = 1 is for test data
C      K is the number of learning steps
C*****************************************************************
    SUBROUTINE TRAINPER(NAMETAG, K)
C*****************************************************************
C Declaration of global variables
C*****************************************************************
     REAL*8 WT(0:2,0:60,0:60), ACT(0:3,0:60)
     REAL*8 TR_CL(500)
     INTEGER*4 NUM_UNITS(0:4)
    INTEGER*4 NUM_LAYERS, NUM_TR_SETS
C*****************************************************************
C Declaration of variables used for MONITOR
C*****************************************************************
     REAL*8 TOPCOR(31),TOPACT(31),TOPPRED(31),TOPTRACT(31,155)
     REAL*8 TOPTRPRED(31,155),MAXI(31)
     REAL*8 TMPTRACT(155),TMPTRPRED(155)
C*****************************************************************
C Common blocks
C*****************************************************************
    COMMON /VARMON/ TOPCOR,TOPACT,TOPPRED,TOPTRACT,TOPTRPRED,
   &            MAXI,TMPTRACT,TMPTRPRED,ICOUNT
     COMMON /VAR/ WT,ACT,TR_CL,NUM_LAYERS,NUM_UNITS,NUM_TR_SETS
    COMMON /VCOR/ ACTUAL,PRED,COR
     COMMON /VT/ NUM_TE_SETS
C*****************************************************************
C Declaration of routine specific variables
C*****************************************************************
    REAL*8 TOT_ERR
    CHARACTER*80 NAME
     INTEGER*4 KK,J,I,NUM_SETS
     REAL*8 ACTUAL(186),PRED(186),COR
C*****************************************************************
C Select training or test set from value of flag NAMETAG
C*****************************************************************
    IF (NAMETAG.EQ.0) THEN
       NAME = 'train.dat'
     NUM_SETS = NUM_TR_SETS
    ELSE
      NAME = 'test.dat'
     NUM_SETS = NUM_TE_SETS
    ENDIF
C*****************************************************************
C Calculate errors
C*****************************************************************
    OPEN(14,FILE=NAME,STATUS='OLD',FORM='FORMATTED')
    TOT_ERR = 0.0
    DO 380 J = 1, NUM_SETS
```

```
C*****************************************************
C Initialise activations
C*****************************************************
      CALL INITACT
C*****************************************************
C Read data
C*****************************************************
       READ(14,*) TR_CL(1), (ACT(0,KK), KK = 1, NUM_UNITS(0))
C*****************************************************
C Propagate forward
C*****************************************************
      CALL PROP
       DO 270 I = 1, NUM_UNITS(NUM_LAYERS - 1)
         TOT_ERR = TOT_ERR + ((TR_CL(I)-ACT(NUM_LAYERS-1,I))
   &                 * (TR_CL(I)-ACT(NUM_LAYERS-1,I)))
270    CONTINUE
C*****************************************************
C Assign desired and predicted outputs to arrays for calculation of
C correlation coefficients
C*****************************************************
       ACTUAL(J) = TR_CL(1)
       PRED(J) = ACT(NUM_LAYERS-1,1)
C*****************************************************
C Store training performance for MONITOR
C*****************************************************
      IF (NAMETAG.EQ.0) THEN
        TMPTRACT(J) = ACTUAL(J)
        TMPTRPRED(J) = PRED(J)
      ENDIF
380  CONTINUE
C*****************************************************
C Calculate Pearson r-squared correlation coefficients
C*****************************************************
    CALL CORREL(NAMETAG)
C*****************************************************
C Write to output files
C*****************************************************
    IF (NAMETAG.EQ.0) THEN
      WRITE(8,*) K,' ', COR
    ELSE
      WRITE(9,*) K,' ',COR
    ENDIF

    CLOSE(14)
C*****************************************************
C Call MONITOR
C*****************************************************
    IF (NAMETAG.EQ.1) THEN
     CALL MONITOR
    ENDIF

    RETURN
    END
C*****************************************************
C Subroutine to calculate Pearson r-squared correlation coefficients
C*****************************************************
    SUBROUTINE CORREL(NAMETAG)
C*****************************************************
```

```
C Declaration of global variables
C*************************************************************
       REAL*8 WT(0:2,0:60,0:60), ACT(0:3,0:60)
      REAL*8 TR_CL(500)
      INTEGER*4 NUM_UNITS(0:4)
     INTEGER*4 NUM_LAYERS, NUM_TR_SETS
C*************************************************************
C Common blocks
C*************************************************************
     COMMON /VAR/ WT,ACT,TR_CL,NUM_LAYERS,NUM_UNITS,NUM_TR_SETS
    COMMON /VCOR/ ACTUAL,PRED,COR
    COMMON /VT/ NUM_TE_SETS
C*************************************************************
C Declaration of routine specific variables
C*************************************************************
     INTEGER*4 I, NUM_SETS
      REAL*8 VA,VB,AVA,AVB,UP,COR,ACTUAL(186),PRED(186)
C*************************************************************
C Select training or test set depending on flag NAMETAG
C*************************************************************
    IF (NAMETAG.EQ.0) THEN
     NUM_SETS = NUM_TR_SETS
    ELSE
     NUM_SETS = NUM_TE_SETS
    ENDIF
C*************************************************************
C Calculate average (mean) of desired outputs and average (mean) of
C predicted outputs
C*************************************************************
    AVA = 0.0
    AVB = 0.0
    DO 10 I = 1, NUM_SETS
      AVA = AVA + ACTUAL(I)
      AVB = AVB + PRED(I)
 10  CONTINUE
    AVA = AVA/FLOAT(NUM_SETS)
    AVB = AVB/FLOAT(NUM_SETS)
C*************************************************************
C Calculate variances of desired and predicted outputs
C*************************************************************
    UP = 0.0
    VA = 0.0
    VB = 0.0
    DO 20 I = 1, NUM_SETS
       UP = UP + ((ACTUAL(I) - AVA) * (PRED(I)   - AVB))
       VA = VA + ((ACTUAL(I) - AVA) * (ACTUAL(I) - AVA))
       VB = VB + ((PRED(I)   - AVB) * (PRED(I)   - AVB))
 20  CONTINUE
C*************************************************************
C Calculate Pearson r-squared correlation coefficient
C*************************************************************
     COR = UP * UP /(VA * VB)

    RETURN
    END
C*************************************************************
C Subroutine to define convergence criterion
C*************************************************************
```

```
   SUBROUTINE MONITOR
C*************************************************************
C Declaration of variables
C*************************************************************
     REAL*8 TOPCOR(31),TOPACT(31),TOPPRED(31),TOPTRACT(31,155)
      REAL*8 TOPTRPRED(31,155),MAXI(31)
      REAL*8 TMPTRACT(155),TMPTRPRED(155)
      REAL*8 ACTUAL(186),PRED(186)
       REAL*8 A(31),P(31)
   REAL*8 TEACT, TEPRED
    REAL*8 AVA,AVB,UP,VA,VB,C
     INTEGER*4 NJACK,NTRAIN,NJM,JJ,J,I
C*************************************************************
C Common blocks
C*************************************************************
   COMMON /VCOR/ ACTUAL,PRED,COR
    COMMON /VARMON/
TOPCOR,TOPACT,TOPPRED,TOPPRED,TOPTRACT,TOPTRPRED
   &          MAXI,TMPTRACT,TMPTRPRED,ICOUNT
C*************************************************************
C Assign variables
C*************************************************************
   NJACK = 31
    NTRAIN = 155
    NJM = NJACK - 1

   ICOUNT = ICOUNT + 1
C*************************************************************
C Create monitor set, excluding test example to be used as true test set
C     Repeat for each example in the test set
C*************************************************************
   DO 200 JACK = 1,NJACK
     JJ = 0
     DO 90 J = 1,NJACK
       IF (J.NE.JACK) THEN
        JJ = JJ + 1
         A(JJ) = ACTUAL(J)
         P(JJ) = PRED(J)
       ELSE
        TEACT = ACTUAL(J)
         TEPRED = PRED(J)
       ENDIF
 90   CONTINUE
C*************************************************************
C Pearson r-squared correlation coefficients are calculated for the
C monitor sets. The routine CORREL is incorporated in this routine,
C rather than further generalising it.
C*************************************************************
C*************************************************************
C Calculate average (mean) of desired outputs and average (mean) of
C predicted outputs for monitor set
C*************************************************************
     AVA = 0.0
     AVB = 0.0
     DO 110 I = 1,NJM
       AVA = AVA + A(I)
       AVB = AVB + P(I)
 110   CONTINUE
```

```
      AVA = AVA/NJM
      AVB = AVB/NJM
C***********************************************************
C Calculate variances of desired and predicted outputs of the monitor
C set
C***********************************************************
      UP = 0.0
      VA = 0.0
      VB = 0.0
      C  = 0.0

      DO 120 I = 1,NJM
         UP = UP + ((A(I) - AVA) * (P(I) - AVB))
         VA = VA + ((A(I) - AVA) * (A(I) - AVA))
         VB = VB + ((P(I) - AVB) * (P(I) - AVB))
 120  CONTINUE
C***********************************************************
C Calculate Pearson r-squared correlation coefficient for the monitor
C set
C***********************************************************
       C = UP * UP/(VA * VB)
C***********************************************************
C Transfer values to temporary stores, if correlation on the monitor
C sets is the best seen so far
C***********************************************************
      IF (C.GT.TOPCOR(JACK)) THEN
       TOPCOR(JACK)  = C
       MAXI(JACK)    = ICOUNT
       TOPACT(JACK)  = TEACT
       TOPPRED(JACK) = TEPRED
       DO 150 K = 1,NTRAIN
         TOPTRACT(JACK,K)  = TMPTRACT(K)
          TOPTRPRED(JACK,K) = TMPTRPRED(K)
 150    CONTINUE
      ENDIF

      RETURN
      END
```

References

Amoore, J.E., Palmieri, G. and Wanke, E. (1967). Molecular Shape and Odour: Pattern Analysis by PAPA *Nature,* **216**, 1084-1087.

Andrea, T.A. and Kalayeh, H. (1991). Applications of Neural Networks in Quantitative Structure-Activity Relationships of Dihydrofolate Reductase Inhibitors *J. Med. Chem.,* **34**, 2824-2836.

Aoyama, T. and Ichikawa, H. (1992). Neural Networks as Nonlinear Structure-Activity Relationship Analysers. Useful Functions of the Partial Derivative Method in Multilayer Neural Networks *J. Chem. Inf. Comput. Sci.,* **32**, 492-500.

Aoyama, T., Suzuki, Y. and Ichikawa, H. (1990a). Neural Networks Applied to Quantitative Structure-Activity Relationship Analysis *J. Med. Chem.,* **33**, 2583-2590.

Aoyama, T., Suzuki, Y. and Ichikawa, H. (1990b). Neural Networks Applied to Structure-Activity Relationships *J. Med. Chem.,* **33**, 905-908.

Argos, P., Rao, J.K.M. and Hargrave, P.A. (1982). Structural Prediction of Membrane-Bound Proteins *Eur. J. Biochem.,* **128**, 565-575.

Arrigo, P., Giuliano, F., Scalia, F., Rapallo, A. and Damiani, G. (1991). Identification of a new motif on nucleic acid sequence data using Kohonen's self organizing map *Comput. Applic. Biosci.,* **7**, 353-357.

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

Barakat, M.T. and Dean, P.M. (1990a). Molecular structure matching by simulated annealing. I. A comparison between different cooling schedules *J. Comput.-Aided Mol. Des.,* **1990**, 295-316.

Barakat, M.T. and Dean, P.M. (1990b). Molecular structure matching by simulated annealing. II. An exploration of the evolution of configuration landscape problems *J. Comput.-Aided Mol. Des.,.* **4**, 317-330.

Barakat, M.T. and Dean, P.M. (1991). Molecular structure matching by simulated annealing. III. The incorporation of null correspondences into the matching problem *J. Comput.-Aided Mol. Des.,* **5**, 107-117.

Barton, G.J. and Sternberg, M.J.E. (1990). Flexible Protein Sequence Patterns *J. Mol. Biol.,* **212**, 389-402.

Bashford, D., Chothia, C. and Lesk, A.M. (1987). Unique Features of the Globin Amino Acid Sequences *J. Mol. Biol.,* **196**, 199-216.

Bengio, Y. and Pouliot, Y. (1990). Efficient recognition of immunoglobulin domains from amino acid sequences using a neural network *Comput. Applic. Biosci.,* **6**, 319-324.

Blaney, J.M., Hansch, C., Silipo, C. and Vittoria, A. (1984). Structure-Activity Relationships of Dihyrofolate Reductase Inhibitors *Chem. Rev.,* **84**, 333-407.

Bohr, H., Bohr, J., Brunak, S., Cotterill, R.M.J., Fredholm, H., Lautrup, B. and Petersen, S.B. (1990). A novel approach to prediction of the 3-dimensional structure of protein backbones by neural networks *FEBS Letts,* **261**, 43-46.

Bohr, H., Bohr, J., Brunak, S., Cotterill, R.M.J., Lautrup, B., Norskov, L., Olsen, O.H. and Petersen, S.B. (1988). Protein secondary structure and homology by neural networks *FEBS Letts,* **241**, 223-228.

Bohr, J., Bohr, H., Brunak, S., Cotterill, R.M.J., Fredholm, H., Lautrup, B. and Petersen, S.B. (1993). Protein Structures from Distance Inequalities *J. Mol. Biol.,* **231**, 861-869.

Bolis, G., Pace, L.D. and Fabrocini, F. (1991). A machine learning approach to computer-aided molecular design *J. Comput.-Aided. Mol. Des.,* **5**, 617-628.

Bowen-Jenkins, P.E., Cooper, D.L. and Richards, W.G. (1985). Ab Initio Computation of Molecular Similarity *J. Phys. Chem.,* **89**, 2195-2197.

Breiman, L., Friedman, J.H., Olshen, R.A. and C.J., S. (1984) Classification and Regression Trees. Wadsworth, Belmont.

Brunak, S., Engelbrecht, J. and Knudsen, S. (1990a). Cleaning up gene databases *Nature,* **343**, 123.

Brunak, S., Engelbrecht, J. and Knudsen, S. (1990b). Neural network detects errors in the assignment of mRNA splice sites *Nucleic Acids Res.,* **18**, 4797-4801.

Brunak, S., Engelbrecht, J. and Knudsen, S. (1991). Prediction of mRNA Donor and Acceptor Sites from the DNA Sequence *J. Mol. Biol.,* **220**, 49-65.

Burt, C. and Richards, W.G. (1990). Molecular similarity: The introduction of flexible fitting *J. Comput.-Aided Mol. Des.,* **4**, 231-238.

Cardozo, M.G., Kawai, T., Iimura, Y., Sugimoto, H. and Yamanishi, Y. (1992). Conformational Analyses and Molecular-Shape Comparisons of a Series of Indanone-Benzylpiperidine Inhibitors of Acetylcholinesterase *J. Med. Chem.,* **35**, 590-601.

Champness, J.N., Stammers, D.K. and Beddell, C.R. (1986). Crystallographic investigation of the cooperative interaction between trimethoprim, reduced cofactor and dihydrofolate reductase *FEBS Letts.,* **199**, 61-67.

Chau, P.-L. and Dean, P.M. (1992a). Automated site-directed drug design: An assessment of the transferability of atomic residual charges (CNDO) for molecular fragments *J. Comput.-Aided Mol. Des.,* **6**, 407-426.

Chau, P.-L. and Dean, P.M. (1992b). Automated site-directed drug design: Searches of the Cambridge Structural Database for bond lengths in molecular fragments to be used for automated structure assembly *J. Comput.-Aided Mol. Des.,* **6**, 397-406.

Chau, P.-L. and Dean, P.M. (1992c). Automated site-directed drug design: The generation of a basic set of fragments to be used for automated structure assembly *J. Comput.-Aided Mol. Des.,* **6**, 385-396.

Chou, P.Y. and Fasman, G.D. (1974). Prediction of Protein Conformation *Biochemistry,* **13**, 222-245.

Clarke, T.J.W., Prager, R.W. and Fallside, F. (1991). The modified Kanerva model: theory and results for real-time word recognition *IEEE Proc.-F,* **138**, 25-31.

Cramer, R.D., Patterson, D.E. and Bunce, J.D. (1988). Comparative Molecular Field Analysis (CoMFA). 1. Effect of Shape on Binding of Steroids to Carrier Proteins *J. Am. Chem. Soc.,* **110**, 5959-5967.

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

David, H.A. (1987). Ranking from unbalanced paired-comparison data *Biometrika,* **74**, 432-436.

Dayhoff, M.O. (1978) *Atlas of Protein Sequence and Structure.* National Biomedical Research Foundation, Washington.

Demeler, B. and Zhou, G. (1991). Neural network optimization for E. coli promoter prediction *Nucleic Acids Res.,* **19**, 1593-1599.

Dickerson, R.E. (1983). Base sequence and helix variation in B and A DNA *J. Mol. Biol.,* **166**, 419-441.

Dietrich, S.W., Blaney, J.M., Reynolds, M.A., Jow, P.Y.C. and Hansch, C. (1980). Quantitative Structure-Selectivity Relationships. Comparison of the Inhibition of Escherichia coli and Bovine Liver Dihydrofolate Reductase by 5-(Substituted-benzyl)-2,4-diaminopyrimidines *J. Med. Chem.,* **23**, 1205-1212.

Dolata, D.P., Leach, A.R. and Prout, K. (1987). WIZARD: AI in conformational analysis *J. Comput.-Aided Mol. Des.,* **1**, 73-85.

Dubchak, I., Holbrook, S.R. and Kim, S.-H. (1993). Prediction of Protein Folding Class From Amino Acid Composition *Proteins,* **16**, 79-91.

Eisenberg, D., Weiss, R.M., Terwilliger, T.C. and Wilcox, W. (1982). *Faraday Symp. Chem. Soc.,* **17**, 109-120.

El Tayar, N., Testa, B. and Carrupt, P.-A. (1992). Polar Intermolecular Interactions Encoded in partition Coefficients: An Indirect Estimation of Hydrogen-Bond Parameters of Polyfunctional Solutes *J. Phys. Chem.,* **96**, 1455-1459.

Elrod, D.W., Maggiora, G.M. and Trenary, R.G. (1990). Applications of Neural Networks in Chemistry. 1. Prediction of

Electrophilic Aromatic Substitution Reactions *J. Chem. Inf. Comput. Sci.,* **30**, 477-484.

Farber, R., Lapedes, A. and Sirotkin, K. (1992). Determination of Eukaryotic Protein Coding Regions using Neural Networks and Information Theory *J. Mol. Biol.,* **226**, 471-479.

Fariselli, P., Compiani, M. and Casadio, R. (1993). Predicting secondary structures of membrane proteins with neural networks *Eur. Biophys. J.,* **22**, 41-51.

Fickett, J.W. (1982). Recognition of protein coding regions in DNA sequences *Nucleic Acids Res.,* **10**, 5303-5318.

Fourney, G.D. (1973). The Viterbi algorithm *Proc. IEEE,* **61**, 268-278.

Free, S.M. and Wilson, J.W. (1964). A Mathematical Contribution to Structure-Activity Studies *J. Med. Chem.,* **7**, 395-399.

Frishman, D. and Argos, P. (1992). Recognition of Distantly Related Protein Sequences Using Conserved Motifs and Neural Networks *J. Mol. Biol.,* **228**, 951-962.

Fujita, T. (1990). In "*Quantitative Drug Design*" pp 497-560. (ed. Ramsden, C.A.) Pergamon Press, Oxford.

Fujita, T. and Ban, T. (1971). Structure-Activity Study of Phenethylamines as Substrates of Biosynthetic Enzymes of Sympathetic Transmitters *J. Med. Chem.,* **14**, 148.

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

Garnier, J. and Levin, J.M. (1991). The protein structure code: what is its present status *Comput. Applic. Biosci.,* **7**, 133-142.

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

Gear, C.W. (1971) *Numerical Initial Value Problems in Ordinary Differential Equations.* Prentice Hall, Englewood Cliffs, NJ.

Gelfand, M.S. (1989). Statistical analysis of mammalian pre-mRNA splicing sites *Nucleic Acids Res.,* **17**, 6369-6382.

Ghose, A.K. and Crippen, G.M. (1983). Combined Distance Geometry Analysis of Dihydrofolate Reductase Inhibitors by Quinazolines and Triazines *J. Med. Chem.,* **26**, 996-1010.

Ghose, A.K. and Crippen, G.M. (1990). Modeling the Benzodiazepine Receptor Binding Site by the General Three-Dimensional Structure-Directed Quantitative Structure-Activity Relationship Method REMOTEDISC *Mol. Pharm.,* **37**, 725-734.

Gibrat, J.F., Garnier, J. and Robson, B. (1987). Further developments of protein secondary structure prediction using information theory. New parameters and consideration of residue pairs *J. Mol. Biol.,* **198**, 425-443.

Good, A.C., So, S.-S. and Richards, W.G. (1993). Structure-Activity Relationships from Molecular Similarity Matrices *J. Med. Chem.,* **36**, 433-438.

Goodford, P.J. (1984). Design by the Method of Receptor Fit *J. Med. Chem.,* **27**, 557-570.

Goodford, P.J. (1985). A Computational Procedure for Determining Energetically Favourable Binding Sites on Biologically Important Macromolecules *J. Med. Chem.,* **28**, 849-857.

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

Gribskov, M., Devereux, J. and Burgess, R.R. (1984). The codon preference plot: graphic analysis of protein coding sequences and prediction of gene expression *Nucleic Acids Res.,* **12**, 539-549.

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

Gund, P., Andose, J.D., Rhodes, J.B. and Smith, G.M. (1980). Three-Dimensional Molecular Modeling and Drug Design *Science,* **208**, 1425-1431.

Hammett, L.P. (1940). *Physical Organic Chemistry, Reaction Rates, Equilibria and Mechanisms.* McGraw-Hill, New York.

Hansch, C. (1969). A Quantitative Approach to Biochemical Structure-Activity Relationships *Acc. Chem. Res.,* **2**, 232-239.

Hansch, C. and Fujita, T. (1964). ρ-σ-π Analysis. A Method for the Correlation of Biological Activity and Chemical Structure *J. Am. Chem. Soc.,* **86**, 1616-1626.

Hansch, C. and Fukunga, J. (1977). Designing biologically active materials *CHEMTECH,* **FEBRUARY**, 120-129.

Hansch, C. and Leo, A.J. (1979). *Substituent Constants for Correlation Analysis in Chemistry and Biology.* Wiley-Interscience, New York.

Hansch, C., Li, R.-I., Blaney, J.M. and Langridge, R. (1982). Comparison of the Inhibition of Escherichia coli and Lactobacillus casei Dihydrofolate Reductase by 2,4-Diamino-5-(Substituted-benzyl)pyrimidines: Quantitative Structure-Activity Relationships, X-ray Crystallography, and Computer Graphics in Structure-Activity Analysis *J. Med. Chem.,* **25**, 777-784.

Hansch, C., Maloney, P.P., Fujita, T. and Muir, R.M. (1962). Correlation of Biological Activity of Phenoxyacetic Acids with Hammett Substitution Constants and Partition Coefficients *Nature,* **194**, 178-180.

Hawley, D.K. and McClure, W.R. (1983). Compilation and analysis of Escherichia coli promoter DNA sequences *Nucleic Acids Res.,* **11**, 2237-2255.

Hayes, D.M. and Kollman, P.A. (1976). Electrostatic Potentials of Proteins. 1. Carboxypeptidase A *J. Am. Chem. Soc.,* **98**, 3335-3345.

Hayward, S. and Collins, J.F. (1992). Limits on α-Helix Prediction With Neural Network Models *Proteins,* **14**, 372-381.

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

Hirst, J.D. and Sternberg, M.J.E. (1991). Prediction of ATP-binding motifs: a comparison of a Perceptron-type neural network and a consensus sequence method *Prot. Engng.,* **4**, 615-623.

Hodgkin, E.E. and Richards, W.G. (1986). A Semi-empirical Method for Calculating Molecular Similarity *J. Chem. Soc., Chem. Commun.,* 1342-1345.

Hodgman, T.C. (1989). The elucidation of protein function by sequence motif analysis *Comput. Applic. Biosci.,* **5**, 1-13.

Holbrook, S.R., Muskal, S.M. and Kim, S.-H. (1990). Predicting surface exposure of amino acids from protein sequence *Prot. Engng.,* **3**, 659-665.

Holley, L.H. and Karplus, M. (1989). Protein secondary structure prediction with a neural network *Proc. Natl. Acad. Sci. USA,* **86**, 152-156.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities *Proc. Natl. Acad. Sci. USA,* **79**, 2554-2558.

Hopfield, J.J. (1984). Neurons with graded response have collective computational properties like those of two-state neurons *Proc. Natl. Acad. Sci. USA,* **81**, 3088-3092.

Hopfield, J.J. and Tank, D.W. (1986). Computing with Neural Circuits: A Model *Science,* **233**, 625-633.

Hopfinger, A.J. (1980). A QSAR Investigation of Dihydrofolate Reductase Inhibition by Baker Triazines Based upon Molecular Shape Analysis *J. Am. Chem. Soc.,* **102**, 7196-7206.

Hopfinger, A.J. (1981). A General QSAR for Dihydrofolate Reductase Inhibition by 2,4-Diaminotriazines Based upon Molecular Shape Analysis *Arch. Bio. Biop.,* **206**, 153-163.

Horton, P.B. and Kanehisa, M. (1992). An assessment of neural network and statistical approaches for prediction of E. coli promoter sites *Nucleic Acids Res.,* **20**, 4311-4338.

Hsu, K.J. and Hsu, A.J. (1990). Fractal geometry of music *Proc. Natl. Acad. Sci. USA,* **87**, 938-941.

Hubbard, T.J.P. and Blundell, T.L. (1987). Comparison of solvent-inaccessible cores of homologous proteins: definitions useful for modelling *Prot. Engng.,* **1**, 159-171.

Hubel, D.H. (1979). The Brain *Sci. Am.,* **241**, 39-46.

Iida, Y. (1987). DNA sequences and multivariate statistical analysis. Categorical discrimination approach to 5' splice site signals of mRNA precursors in higher eukaryotes' genes *Comput. Applic. Biosci.,* **3**, 93-98.

Jurs, P.C. (1986). Pattern Recognition Used to Investigate Multivariate Data in Analytical Chemistry *Science,* **232**, 1219-1224.

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

Kabsch, W. and Sander, C. (1983b). How good are predictions of protein secondary structure? *FEBS Letts,* **155**, 179-182.

Kendall, M. and Stuart, A. (1977). *The Advanced Theory of Statistics.* Griffen, London.

King, R.D., Muggleton, S., Lewis, R.A., Srinivasan, A., Feng, C. and Sternberg, M.J.E. (1993). In "*Proceedings of the 26th Annual Hawaii International Conference on System Sciences*" pp 646-655. (ed. Mudge, T.N., Milutinovic, V. and Hunter, L.) IEEE Computer Society Press, Los Alamitos, California.

King, R.D., Muggleton, S., Lewis, R.A. and Sternberg, M.J.E. (1992). Drug design by machine learning: the use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase *Proc. Natl. Acad. Sci. USA,* **89**, 11322-11326.

King, R.D. and Sternberg, M.J.E. (1990). Machine Learning Approach for the Prediction of Protein Secondary Structure *J. Mol. Biol.,* **216**, 441-457.

Kirkpatrick, S., Gelatt, C. and Vecchi, M. (1983). Optimization by simulated annealing *Science,* **220**, 671-680.

Klopman, G. (1984). Artificial Intelligence Approach to Structure-Activity Studies. Computer Automated Structure Evaluation of Biological Activity of Organic Molecules *J. Am. Chem. Soc.*, **106**, 7315-7321.

Klopman, G. and Ptchelintsev, D. (1993). Antifungal triazole alcohols: A comparison analysis of structure-activity, structure-teratogenicity and structure-therapeutic index relationships using the Multiple Computer-Automated Structure Evaluation (Multi-CASE) methodology *J. Comput.-Aided Mol. Des.*, **7**, 349-362.

Kneller, D.G., Cohen, F.E. and Langridge, R. (1990). Improvements in Protein Secondary Structure Prediction by An Enhanced Neural Network *J. Mol. Biol.,* **214**, 171 - 182.

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

Kowalski, B.R. and Bender, C.F. (1972). Pattern Recognition. A Powerful Approach to Interpreting Chemical Data *J. Am. Chem. Soc.,* **94**, 5632-5639.

Kowalski, B.R. and Bender, C.F. (1974). The Application of Pattern Recognition to Screening Prospective Anticancer Drugs. Adenocarcinoma 755 Biological Activity Test *J. Am. Chem. Soc.,* **96**, 916-918.

Kubinyi, H. (1990). In "*Quantitative Drug Design*" pp 589-644. (ed. Ramsden, C.A.) Pergamon Press, Oxford.

Kuntz, I.D. (1992). Structure-Based Strategies for Drug Design and Discovery *Science,* **257**, 1078-1082.

Kuntz, I.D., Blaney, J.M., Oatley, S.J., Langridge, R. and Ferrin, T.E. (1982). A Geometirc Approach to Macromolecule-Ligand Interactions *J. Mol. Biol.,* **161**, 269-288.

Kvasnicka, V.K., Sklenak, S. and Pospichal, J. (1993). Neural Network Classification of Inductive and Resonance Effects of Substituents *J. Am. Chem. Soc.,* **115**, 1495-1500.

Ladunga, I., Czako, F., Csabai, I. and Geszti, T. (1991). Improving signal peptide prediction accuracy by simulated neural network *Comput. Applic. Biosci.,* **7**, 485-487.

Lapedes, A., Barnes, C., Burks, C., Farber, R. and Sirotkin, K. (1990). In "*Computers and DNA, SFI Studies in the Sciences of Complexity*" pp 157-182. (ed. Bell, G. and Marr, T.) Addison-Wesley, Reading, MA.

Leach, A.R., Prout, K. and Dolata, D.P. (1988). An investigation into the construction of molecular models by the template joining method *J. Comput.-Aided Mol. Des.,* **2**, 107-123.

Lehky, S.R. and Sejnowski, T.E. (1988). Network model of shape-from-shading: neural function arises from both receptive and projective fields *Nature,* **333**, 452-454.

Leo, A., Hansch, C. and Elkins, D. (1971). Partition Coefficients and their uses *Chem. Rev.,* **71**, 525-616.

Levin, J.M., Robson, B. and Garnier, J. (1986). An algorithm for secondary structure determination in proteins based on sequence similarity *FEBS Letts,* **205**, 303-308.

Lewis, R.A. (1989). Determination of clefts in receptor structures *J. Comput.-Aided Mol. Des.,* **3**, 133-147.

Lewis, R.A. (1990). Automated site-directed drug design: Approaches to the formatino of 3D molecular graphs *J. Comput.-Aided Mol. Des.,* **4**, 205-210.

Lewis, R.A. and Dean, P.M. (1989). Automated site-directed drug design: the formation of molecular templates in primary structure generation *Proc. R. Soc. Lond. B,* **236**, 141-162.

Li, R.-L. and Poe, M. (1988). Quantitative Structure-Activity Relationships for the Inhibition of Escherichia coli Dihydrofolate Reductase by 5-(Substituted benzyl)-2,4-diaminopyrimidines *J. Med. Chem.,* **31**, 366-370.

Li, R.L., Hansch, C. and Kaufman, B.T. (1982). A Comparison of the Inhibitory Action of 5-(Substituted-benzyl)-2,4-diaminopyrimidines on Dihydrofolate Reductase from Chicken Liver with That from Bovine Liver *J. Med. Chem.,* **25**, 435-440.

Liebman, M.N., Venanzi, C.A. and Weinstein, H. (1985). Structural Analysis of Carboxypeptidase A and its Complexes with Inhibitors as a Basis for Modeling Enzyme Recognition and Specificity *Biopolymers,* **24**, 1721-1758.

Lim, V.I. (1974). Algorithms for Prediction of α-Helical and β-Structural Regions in Globular Proteins *J. Mol. Biol.,* **88**, 873-894.

Lippmann, R.P. (1987). An Introduction to Computing with Neural Networks *IEEE ASSP Mag,* **4**, 4-22.

Livingstone, D.J. (1991). Novel method for the display of multivariate data using neural networks *J. Mol. Graphics,* **9**, 115-118.

Livingstone, D.J. and Mallanack, D.T. (1993). Statistics Using Neural Networks: Chance Effects *J. Med. Chem.,* **36**, 1295-1297.

Livingstone, D.J. and Salt, D.W. (1992). Regression analysis for QSAR using neural networks *Bioorg. Med. Chem. Letts.,* **2**, 213-218.

Loew, G.H. and Burt, S.K. (1990). In "*Quantitative Drug Design*" pp 105-124. (ed. Ramsden, C.A.) Pergamon Press, Oxford.

Luce, H.H. and Govind, R. (1990). Neural Network Applications in Synthetic Organic Chemistry: I. A Hybrid System Which Performs Retrosynthetic Analysis *Tetra. Comp. Meth.,* **3**, 143-161.

Lukashin, A.V., Anshelevich, V.V., Amirikyan, B.R., Gragerov, A.I. and Frank-Kamenetskii, M.D. (1989). Neural Network models for Promoter Recognition *J. Biomol. Struct. Dyn.,* **6**, 1123-1133.

Martin, Y.C. (1978) *Quantitative Drug Design.* Marcel Dekker, Inc., NY.

Marshall, G.R. and Cramer, R.D. (1988). Three-dimensional structure-activity relationships *Trends Pharmacol. Sci.,* **9**, 285-289.

Mathews, R.J. (1975). A Comment on Structure-Activity Correlations Obtained Using Pattern Recognition Methods *J. Am. Chem. Soc.,* **97**, 935-936.

Matthews, D.A., Bolin, J.T., Burridge, J.M., Filman, D.J., Volz, K.W., Kaufman, B.T., Beddell, C.R., Champness, J.N., Stammers, D.K. and Kraut, J. (1985). Refined Crystal Structures of Escherichia coli and Chicken Liver Dihydrofolate Reductase Containing Bound Trimethoprim *J. Biol. Chem.,* **260**, 381-391.

McCammon, J.A. (1987). Computer-Aided Molecular Design *Science,* **238**, 486-491.

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity *Bull. Math. Biophys.,* **5**, 115-133.

McFarland, J.W. (1992). Comparative Molecular Analysis of Anticoccidal Triazines *J. Med. Chem.,* **35**, 2543-2550.

McGregor, M.J., Flores, T.P. and Sternberg, M.J.E. (1989). Prediction of beta-turns in proteins using neural networks *Prot. Engng.,* **2**, 521-526.

McGregor, M.J., Flores, T.P. and Sternberg, M.J.E. (1990). Corrigendum *Prot. Engng.,* **3**, 459-460.

Metfessel, B.A., Saurugger, P.N. and Connelly, D.P. (1993). Cross-validation of protein structural class prediction using statistical clustering and neural networks *Prot. Sci.,* **2**, 1171-1182.

Meyer, A.M. and Richards, W.G. (1991). Similarity of molecular shape *J. Comput.-Aided Mol. Des.,* **5**, 427-439.

Mezard, M. and Nadal, J.P. (1989). Learning in feedforward layered networks: the tiling algorithm *J. Phys. A.,* **22**, 2193-2203.

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

Muggleton, S. and Feng, C. (1990). In "*Proceedings of the First Conference on Algorithmic Learning Theory*" pp 368-381. (ed. Arikawa, S., Goto, S., Ohsuga, S. and Yokomori, T.) Jpn. Soc. Artifical Intelligence, Tokyo.

Muggleton, S., Srinivasan, A. and Bain, M. (1992). In "*Proceedings of 9th International Conference on Machine Learning*" pp 338-347. (ed. Sleeman, D. and Edwards, P.) Morgan-Kaufman, San Mateo.

Mulligan, M.E., Hawley, D.K., Entriken, R. and McClure, W.R. (1984). Escherichia coli promoter sequences predict in vitro RNA polymerase selectivity *Nucleic Acids Res.,* **12**, 789-800.

Muskal, S.M., Holbrook, S.R. and Kim, S.-H. (1990). Prediction of the disulfide-bonding state of cysteine in proteins *Prot. Engng.*, **3**, 667-672.

Muskal, S.M. and Kim, S.H. (1992). Predicting Protein Secondary Structure Content. A Tandem Neural network Approach *J. Mol. Biol.*, **225**, 713-727.

Nakata, K., Kanehisa, M. and DeLisi, C. (1985). Prediction of splice junctions in mRNA sequences *Nucleic Acids Res.*, **13**, 5327-5340.

Nakata, K., Kanehisa, M. and Maizel, J.V. (1988). Discriminant analysis of promoter regions in Escherichia coli sequences *Comput. Applic. Biosci.*, **4**, 367-371.

O'Neill, M.C. (1989). Escherichia coli Promoter I. Consensus as it relates to spacing class, specificity, repeat substructure, and three-dimensional organization *J. Biol. Chem.*, **264**, 5522-5530.

O'Neill, M.C. (1991). Training back-propagation neural networks to define and detect DNA-binding sites *Nucleic Acids Res.*, **19**, 313-318.

O'Neill, M.C. (1992). Escherichia coli promoters: neural networks develop distinct descriptions in learning to search for promoters of different spacing classes *Nucleic Acids Res.*, **20**, 3471-3477.

Ohshima, Y. and Gotoh, Y. (1987). Signals for the Selection of a Splice Site in Pre-mRNA Computer Analysis of Splice Junction Sequences and Like Sequences *J. Mol. Biol.*, **195**, 247-259.

Owens, A.J. and Filkin, D.L. (1989). In "*Joint IEEE/INNS International Joint Conference of Neural Networks*" pp 381-386. Washington D.C.

Pancoska, P., Blazek, M. and Keiderling, T.A. (1992). Relationships between Secondary Structure Fractions for Globular Proteins. Neural Network Analyses of Crystallographic Data Sets *Biochemistry,* **31**, 10250-10257.

Papadopoulos, M.C. and Dean, P.M. (1991). Molecular structure matching by simulated annealing. IV. Classification of atom correspondences in sets of dissimilar molecules *J. Comput.-Aided Mol. Des.,* **5**, 119-133.

Perrin, C.L. (1974). Testing of Computer-Assisted Methods for Classification of Pharmacological Activity *Science,* **181**, 551-552.

Petersen, S.B., Bohr, H., Bohr, J., Brunak, S., Cotterill, R.M.J., Fredholm, H. and Lautrup, B. (1990). Training neural networks to analyse biological sequences *Trends Biotechnol.,* **8**, 304-308.

Press, W.H., Teukolsky, S.A., Vetterling, W.T. and Flannery, B.P. (1992). *Numerical Recipes.* Cambridge University Press, Cambridge.

Ptitsyn, O.B. and Finkelstein, A.V. (1989). Prediction of protein secondary structure based on physical theory *Prot. Engng.,* **2**, 443-447.

Qian, N. and Sejnowski, T.J. (1988). Predicting the Secondary Structure of Globular Proteins Using Neural Network Models *J. Mol. Biol.,* **202**, 865-884.

Quinlan, J. (1986). Induction of Decision Trees *Machine Learning,* **1**, 81-106.

Richards, W.G. (1989a). *Computer-aided Molecular Design.* IBC Technical Services, London.

Richards, W.G. (1989b). In "*Computer-aided Molecular Design*" pp 43-50. (ed. Richards, W.G.) IBC Technical Services, London.

Richardson, J.S. (1981). The Anatomy and Taxonomy of Protein Structure *Adv. Prot. Chem.,* **34**, 167-339.

Rooman, M.J. and Wodak, S.J. (1988). Identification of predictive sequence motifs limited by protein structure data base size *Nature,* **335**, 45-49.

Rose, G.D., Geselowitz, A.R., Lesser, G.J., Lee, R.H. and Zehfus, M.H. (1985). Hydrophobicity of amino acid residues in globular proteins *Science,* **229**, 834-838.

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

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

Rost, B. and Sander, C. (1993a). Prediction of Protein Secondary Structure at Better than 70% Accuracy *J. Mol. Biol.,* **232**, 584-599.

Rost, B. and Sander, C. (1993b). Improved prediction of protein secondary structure by use of sequence profiles and neural networks *Proc. Natl. Acad. Sci. USA*, **90**, 7558-7562.

Rost, B., Schneider, R. and Sander, C. (1993). Progress in protein structure prediction? *Trends in Bio. Sci.,* **18**, 120-123.

Roth, B., Aig, E., Rauckman, B.S., Srelitz, J.Z., Phillips, A.P., Ferone, R., Bushby, S.R.M. and Siegel, C.W. (1981). 2,4-Diamino-5-benzylpyrimidines and Analogues as Antibacterial Agents.

5. 3',5'-Dimethoxy-4'-substituted-benzyl Analogues of Trimethoprim *J. Med. Chem.,* **24**, 933-941.

Roth, B., Rauckman, B.S., Ferone, R., Baccanari, D.P., Champness, J.N. and Hyde, R.M. (1987). 2,4-Diamino-5-benzylpyrimidines as Antibacterial Agents. 7. Analysis of the Effect of 3,5-Dialkyl Substituent Size and Shape on Binding to Four Different Dihydrofolate Reductase Enzymes *J. Med. Chem.,* **30**, 348-356.

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

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986b). *Parallel Distributed Processing.* MIT Press, Cambridge, MA.

Saraste, M., Sibbald, P.R. and Wittinghofer, A. (1990). The P-loop - a common motif in ATP- and GTP- binding proteins *Trends in Bio. Sci.,* **15**, 430-434.

Sasagawa, F. and Tajima, K. (1993). Prediction of protein secondary structures by a neural network *Comput. Applic. Bio. Sci.,* **9**, 147-152.

Schillen, T.B. (1991). Designing a neural network simulator - the MENS modelling environment for network systems: I *Comput. Applic. Biosci.,* **7**, 417-430.

Schneider, G. and Wrede, P. (1992). Modular Feature Extraction in Protein Sequences with Artificial Neural Networks: Analog Model for Symbiogenenous Constraints *Endocytobiosis and Cell Res.,* **9**, 1-12.

Sejnowski, T.E. and Rosenberg, C.R. (1987). Parallel Networks that Learn to Pronounce English Text *Compl. Syst.,* **1**, 145-168.

Selassie, C.D., Li, R.-L., Poe, M. and Hansch, C. (1991). On the Optimization of Hydrophobic and Hydrophilic Substituent Interactions of 2,4-Diamino-5-(substituted benzyl)pyrimidines with Dihydrofolate Reductase *J. Med. Chem.,* **34**, 46-54.

Shapiro, M.B. and Senepathy, P. (1987). RNA splice junctions of different classes of eukaryotes: sequence statistics and functional implications in gene expression *Nucleic Acids Res.,* **15**, 7155-7173.

Shepherd, J.C.W. (1981). Method to determine the reading frame of a protein from the purine/pyrimidine genome sequence and its possible evolutionary justification *Proc. Natl. Acad. Sci. USA,* **78**, 1596-1600.

Shoichet, B.K., Stroud, R.M., Santi, D.V., Kuntz, I.D. and Perry, K.M. (1993). Structure-Based Discovery of Inhibitors of Thymidylate Synthase *Science,* **259**, 1445-1450.

Silipo, C. and Hansch, C. (1975). Correlation Analysis. Its application to the Structure-Activity Relationship of Triazines Inhibiting Dihydrofolate Reductase *J. Am. Chem. Soc.,* **97**, 6849-6861.

Silipo, C. and Hansch, C. (1976). Correlation Analysis of Baker's Studies on Enzyme Inhibition. 1. Guanine Deaminase, Xanthine Oxidase, Dihydrofolate Reductase, and Complement *J. Med. Chem.,* **19**, 62-71.

Simon, Z. (1974). Specific Interactions. Intermolecular Forces, Steric Requirements, and Molecular Size *Angew. Chem. Int. Ed. Engl.,* **13**, 719-727.

Simon, Z., Badilescu, I. and Racovitan, T. (1977). Mapping of Dihydrofolate-reductase Receptor Site by Correlation with

Minimal Topological (Steric) Differences *J. theor. Biol.,* **66**, 485-495.

Simpson, P.F. (1990). *Artificial Neural Systems.* Pergammon Press, Oxford.

So, S.-S. and Richards, W.G. (1992). Application of Neural Networks: Quantitative Structure-Activity Relationships of the Derivatives of 2,4-Diamino-5-(substituted-benzyl)pyrimidines as DHFR Inhibitors *J. Med. Chem.,* **35**, 3201-3207.

Staden, R. (1984a). Computer nethods to locate signals in nucleic acid sequences *Nucleic Acids Res.,* **12**, 505-519.

Staden, R. (1984b). Measurements of the effects that coding for a protein has on a DNA sequence and their use for finding genes *Nucleic Acids Res.,* **12**, 551-567.

Staden, R. and McLachlan, A.D. (1982). Codon preference and its use in identifying protein coding regions in long DNA sequences *Nucleic Acids Res.,* **10**, 141-156.

Sternberg, M.J.E. (1991a). Library of common protein motifs *Nature,* **349**, 111.

Sternberg, M.J.E. (1991b). PROMOT: A FORTRAN program to scan protein sequences against a library of known motifs *Comput. Applic. Biosci.,* **7**, 257-260.

Sternberg, M.J.E. (1992). Secondary Structure Prediction *Curr. Opinion in Str. Biol.,* **2**, 237-241.

Stolorz, P., Lapedes, A. and Xia, Y. (1992). Predicting Protein Secondary Structure Using Neural Net and Statistical Methods *J. Mol. Biol.,* **225**, 363-377.

Stormo, G.D., Schneider, T.D., Gold, L. and Ehrenfeucht, A. (1982a). Use of the 'Perceptron' algorithm to distinguish translational initiation sites in E. coli *Nucleic Acids Res.,* **10**, 2997-3011.

Stormo, G.D., Schneider, T.D. and Gold, L.M. (1982b). Characterization of translational initiation sites in E. coli *Nucleic Acids Res.,* **10**, 2971-2996.

Stuper, A.J., Brugger, W.E. and Jurs, P.C. (1979). *Computer Assisted Studies of Chemical Structure and Biological Function.* John Wiley & Sons, New York.

Stuper, A.J. and Jurs, P.C. (1975). Classification of Psychotropic Drugs as Sedatives or Tranquilizers Using Pattern Recognition Techniques *J. Am. Chem. Soc.,* **97**, 182-187.

Synder, E.E. and Stormo, G.D. (1993). Identification of coding regions in genomic DNA sequences: an application of dynamic programming and neural networks *Nucleic Acids Res.,* **21**, 607-613.

Taft, R.W. (1952). Polar and Steric Substituent Constants for Aliphatic and *o*-Benzoate Groups from Rates of Esterification and Hydrolosis of Esters *J. Am Chem. Soc.,* **74**, 3120-3128.

Taft, R.W. (1953). Linear Steric Energy Relationships *J. Am. Chem. Soc.,* **75**, 4538-4539.

Taylor, W.R. (1986). Identification of protein sequence homology by consensus template alignment *J. Mol. Biol.,* **188**, 233-258.

Tetko, I.V., Luik, A.I. and Poda, G.I. (1993). Applications of Neural Networks in Structure-Activity Relationships of a Small Number of Molecules *J. Med. Chem.,* **36**, 811-814.

Ting, K.-L.H., Lee, R.C., Milne, G.W.A., Shapiro, M. and Guarino, A.M. (1973). Applications of Artificial Intelligence:

Relationships between Mass Spectra and Pharmacological Activity of Drugs *Science,* **180**, 417-420.

Tong, L., Milburn, M.V., de Vos, A.M. and Kims, S.-H. (1990). Structure of ras Protein *Science*, **245**, 244.

Tramontano, A. and Macchiato, M.F. (1986). Probability of coding a DNA sequence: an algorithm to predict translated reading frames from their thermodynamic characteristics *Nucleic Acids Res.,* **14**, 127-135.

Trifonov, E.N. (1987). Translation Framing Code and Frame-monitoring Mechanism as Suggested by the Analysis of mRNA and 16S rRNA Nucleotide Sequences *J. Mol. Biol.,* **194**, 643-652.

Uberbacher, E.C. and Mural, R.J. (1991). Locating protein-coding regions in human DNA sequences by a multiple sensor-neural network approach *Proc. Natl. Acad. Sci. USA,* **88**, 11261-11265.

Ugi, I., Bauer, J., Brandt, J., Friedrich, J., Gasteiger, J., Jochum, C. and Schubert, W. (1979). New Applications of Computers in Chemistry *Angew. Chem. Int. Ed. Engl.,* **18**, 111-123.

Vieth, M. and Kolinski, A. (1991). Prediction of Protein Secondary Structure by an Enhanced Neural Network *Act. Biochim. Pol.,* **38**, 335-351.

Vieth, M., Kolinski, A., Skolnick, J. and Sikorski, A. (1992). Prediction of Protein Secondary Structure by Neural Networks: Encoding Short and Long Range Patterns of Amino Acid Packing *Acta Biochim. Pol.,* **39**, 369-392.

von Heijne, G. (1986). A new method for predicting signal sequence cleavage sites *Nucleic Acids Res.,* **14**, 4683-4690.

von Heijne, G. (1991). Computer analysis of DNA and protein sequences *Eur. J. Biochem.,* **199**, 253-256.

von Itzstein, M., Wu, W.-Y., Kok, G.B., Pegg, M.S., Dyason, J.C., Jin, B., Phan, T.V., Smythe, M.L., White, H.F., Oliver, S.W., Colman, P.M., Varghese, J.N., Ryan, M., Woods, J.M., Bethell, R.C., Hotham, V.J., Cameron, J.M. and Penn, C.R. (1993). Rational design of potent sialidase-based inhibitors of influenza virus replication *Nature,* **363**, 418-423.

Wade, R.C., Bohr, H. and Wolynes, P.G. (1992). Prediction of Water Binding Sites on Proteins by Neural Networks *J. Am. Chem. Soc.,* **114**, 8284-8285.

Walker, J.E., Saraste, M., Runswick, M.J. and Gay, N.J. (1982). Distantly related sequences in the α- and β-subunits of ATP synthase, myosin, kinases and other ATP-requiring enzymes and a common nucleotide binding fold *EMBO J.,* **1**, 945-951.

Widrow, B. (1960). *An adaptive "adaline" neuron using chemical "memistors"* . Stanford Electronics Laboratory Technical Report 1553-2.

Wikel, J.H. and Dow, E.R. (1993). The Use of Neural Networks for Variable Selection in QSAR *Bioorg. & Med. Chem. Lett.,* **3**, 645-651.

Wilcox, G.L., Poliac, M. and Liebman, M.N. (1990). Neural Network Analysis of Protein Tertiary Structure *Tetra. Comp. Meth.,* **3**, 191-211.

Wilmot, C.M. and Thornton, J.M. (1988). Analysis and Prediction of the Different Types of beta-Turn in Proteins *J. Mol. Biol.,* **203**, 221-232.

Wu, C., Whitson, G., McLarty, J., Ermongkonchai, A. and Chang, T.-C. (1992). Protein classification artificial neural system *Prot. Sci.,* **1**, 667-677.

Zvelebil, M.J.J.M., Barton, G.J., Taylor, W.R. and Sternberg, M.J.E. (1987). Prediction of Protein Secondary Structure and Active Sites Using the Alignment of Homologous Sequences *J. Mol. Biol.,* **195**, 957-961.