KUNGL TEKNISKA HÖGSKOLAN
Numerisk analys och datalogi

Predicting the structure of protein fragments

An application of neural networks in protein structure predictions

av
Christian Ottosson

TRITA-NA-E00XY

NADA

Nada (Numerisk analys och datalogi)
KTH
100 44 Stockholm

Department of Numerical Analysis
and Computer Science
Royal Institute of Technology
SE-100 44 Stockholm, Sweden

Predicting the structure of protein fragments

An application of neural networks in protein structure predictions

av
Christian Ottosson

TRITA-NA-E00XY

Examensarbete i datalogi om 20 poäng
vid Programmet för Teknisk Fysik,
Kungliga Tekniska Högskolan år 2001
Handledare på Nada var Jens Lagergren
Examinator var Stefan Arnborg

Abstract

This Master's thesis discusses the prediction of the structure of protein fragments, using *artificial neuronal networks* (ANN), on the basis of sequential information. In this study, the prediction of the similarity in structure, between pairs of fragments, is used. The influence of different parameters regulating the training set and the network function has been examined.

Among others, it is found that sequential information from 3–4 adjacent amino acids on each side of the fragment should be considered. With the network configuration used in this study around 7–10 neurons should be used in the hidden layer and the network performs as well served with exact structural values as with discrete classes.

Att förutsäga proteinfragments struktur

En tillämning av neurala nätverk
för att förutsäga proteinstrukturer

Sammanfattning

Den här examensarbetsrapporten behandlar möjligheterna att förutsäga proteinfragments struktur, med *artificella neuronnät* (ANN), utgående från sekvensinformation. I den här studien förutsägs strukturlikheten, mellan par av fragment. Påverkan av olika parametrar, som reglerer träningsmängden och nätverksfunktionen har undersökts.

Det framkom, bland annat, att sekvensiell information från 3–4 närliggande aminosyror on vardera sida av fragmentet bör beaktas. Med den aktuella nätverkskonfigurationen bör runt 7–10 neuroner användas i det gömda lagret och nätverket presterar lika bra med exakta värden för strukturen som diskreta klasser som indata.

Contents

Chapter 1

Introduction

Ongoing sequencing projects, such as the human genome project, have produced enormous amounts of biological sequences. Since the structure of a protein determines the function of the protein, the three dimensional structure is sought once the sequence of a previously undescribed protein is revealed. Unfortunately, due to the fact that solving a protein structure can be a very time consuming effort, the X-ray crystallographers and NMR spectroscopists are unable to keep up with the sequencing projects. Hence, thereis a growing gap between known sequences and known structures. However, during the past decade new methods have made it possible to extract information about the protein structure from the amino acid sequence alone, by using computers and biological databases. Today, it is not possible to correctly solve a protein structure from only its sequence, but if small subsequences (secondary structures) in the protein are known, it may be possible to obtain at least a reasonably good picture of the complete structure.

The aim of this study has been to examine the possibilities to predict the structure, using *artificial neuronal networks* (ANN), on the basis of the sequence information, or more exactly, to predict the similarity in structure, between pairs of protein fragments. The influence of different parameters regulating the training set and the network function has been examined.

This Master's project was conducted at *Stockholm Bioinformatics Center* (SBC) under supervision of Arne Elofsson at SBC and Jens Lagergren at Nada, KTH.

Chapter 2

Background

2.1 Proteins and protein structure prediction

If genes are the blueprints of life, proteins are the machinery. Proteins are present and play key roles in almost all biological processes, i.e. nearly all catalysts in biological systems are proteins (enzymes). But proteins also mediate a wide range of other functions in the cell, such as transport, storage, mechanical support, immune protection and control of growth and differentiation [28]. Proteins are constructed by joining amino acids through peptide bonds into long polypeptide chains. Since the amino acid library consists of 20 different letters and since proteins are of different lengths (ranging from 20 amino acids to more than 40,000), the number of possible combinations is huge. In their natural environment water, the proteins are folded into unique 3D structures (see Fig. 2.1).

The two main driving forces in the folding process are the energetically unfavorable interactions between the protein backbone and the residues, and water and certain residues, called hydrophobic[1] residues. The polypeptide chain is packed to a unique structure, where most of the hydrophobic residues can be found on the inside, and most of the hydrophilic on the outside, which makes the latter more accessible to interactions with the surrounding water.

Our understanding of how proteins perform their numerous functions has been greatly enriched by the 3D structures solved and reported by X-ray crystallography and *nuclear magnetic resonance* (NMR). In X-ray crystallography, electrons scatter X-rays and the way in which the scattered waves recombine depends only on the the atomic arrangement. NMR spectroscopy complements crystallography by revealing the structure and dynamics of proteins in solution. The chemical shift of nuclei, which is used in NMR, depends on the local environment. Unfortunately, these methods can be very time consuming and therefore different structure prediction methods have evolved.

[1] Hydrofobia = fear of water

Figure 2.1. A schematic figure of a hydrolase (serine proteinase) (PDB ID: 1btw) generated by the program RASMOL (`ftp://ftp.dcs.ed.ac.uk/pub/rasmol/`). The two main secondary structures, the alpha helices and the beta strands, can easily be detected in this figure.

It is a fact that protein structure determines function, but the underlying principles that fully determine the structure have not yet been found. The most accepted hypothesis is that, for most protiens, the sequence determines the complete structure of the protein [3]. It is still generally assumed that the structure can be found at the free-energy minimum. Two general directions of predicting the protein structure (and hybrids of these) have evolved [10]. The first, a *molecular mechanics* approach, assumes that a correctly folded protein occupies a minimum energy conformation. Potential energy is obtained by summing the terms due to bond (distance, angle, torsion) and non-bond (contact, electrostatic, hydrogen) components calculated from forcefield parameters that are derived from experimental observations or calculations (ab initio, semi-empirical) of amino acids and small molecules. The potential energy can then be minimized as a function of the atomic coordinates of the protein in order to reach a minimum (global/local). Since this method is depending on the starting conformation, *molecular dynamics* is used to simulate the way the protein would move away from the, usually arbitrary, initial state. This method has more or less failed because of two reasons [12]:

1. Energy differences between native and unfolded proteins are extremely small (order of 1 kcal/mol).

2. The high complexity of protein folding requires several orders of magnitude more computing time than we anticipate to have over the next decades.

The second direction is based on using information of known protein structures and sequences contained in publicly available databases. This approach tries to find common features, patterns, in the available structures, which then can be generalized to provide structural models of other proteins.

2.2 Secondary structure prediction

The prediction of secondary structures such as alpha helices and beta strands is not only of importance in the process of locating important structural and functional motifs, but it is an intermediate step to solve the whole protein structure problem. If one was able to correctly predict the secondary structures, then it might be possible to solve a small number of 3D structures by using knowledge about the ways that secondary structural elements pack. The principle behind these prediction methods is the fact that a sequence of consecutive residues has preference for a certain substructure [4].

2.3 Pattern recognition

In our daily life, we are all exposed to information from the local environment, but surprisingly, we almost instantly classify or recognize the source behind the information. We can for example recognize a person we know on the other end of a telephone line by his or her voice, or the spices in a dish by just smelling or tasting it. However, the ability to recognize patterns is not a unique feature for humans or animals, even machines can do this. Pattern recognition is thus formally defined as the process whereby a received pattern/signal is assigned to one of a prescribed number of classes (categories) [6]. Machine learning techniques, e.g. neural networks, perform pattern recognition by first going through a training session where the training examples (features extracted from an observation space) together with the target class for each example are shown to the network. After the machine has been trained, a new example that belongs to the same feature space as the training examples is shown to the machine. Hopefully, the machine then recognizes the similarities and classifies the example correctly (Fig. 2.2).

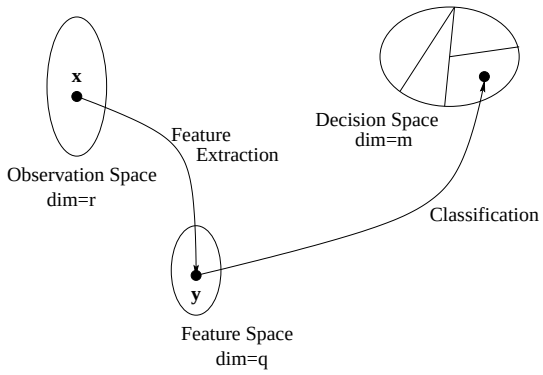

Figure 2.2. The classic approach to pattern recognition.

Chapter 3

Methods

In this study, *artificial neuronal networks* (ANN) have been used to predict the structure similarity of pairs of protein fragments. This chapter describes the background and theory for this machine learning technique. To continue, different ways of encoding biological sequences and comparing sequence structures are explained. A method for measuring prediction performance concludes the chapter.

3.1 Artificial neural networks

3.1.1 The biological background to neural networks

The fundamental task of a neuron is to receive, conduct and transmit signals in the form of electrical potential over the membrane. Neurons, in general, are elongated and consist of numerous dendrites (up to 100,000), which receive signals from other neurons. The incoming signals are then propagated to the cell body (soma), which also contains the nucleus of the neuron. The soma itself can also receive signals and all the signals that reach the cell body over a certain time are summed. If the sum of all signals is higher than a threshold value, the neuron sends a signal along the axon. This is an all or none property, meaning that if the sum of all the incoming signals is lower than the threshold value, the signal will not be propagated. But if the sum is higher than the threshold, a signal with a predefined amplitude will be generated. This is a classical case of a nonlinearity, the amplitude of the generated signal is not proportional to the sum of incoming signals. This traveling signal, actually a wave of electrical potential, is known as *action potential* or *nerve impulse*. The axon commonly divides at its far end into many branches, and by doing so, the signal can be passed on to many other neurons. At the end of each branch there is a specialized site of contact, the synapse. There exist several types of synapses, but in the most common, the chemical synapse, the electrical potential results in the secretion of neurotransmitter substances over a synaptic cleft that separates the two

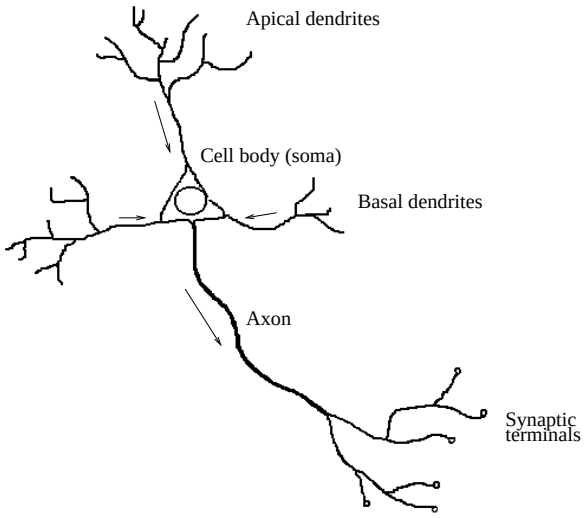

Figure 3.1. A biological scheme of a neuron and the direction of signal propagation.

neurons (presynaptic process). When the substances diffuse over the cleft to dendrites of other neurons, the neurotransmitters indirectly or directly result in the transmission of a signal in the receiving dendrites. Thus, the synapses convert a presynaptic electrical signal into a chemical signal, and then back into a postsynaptic electrical signal.

Events in a silicon chip happen in nanoseconds (10^{-9} s) whereas the neural events in the approximately 10 billion neurons in a human brain happen in milliseconds (10^{-3} s) However, the estimated 60 trillion connections, synapses, make up for the slow propagation rate in each neuron. This feature is what makes the brain such a fantastic computational structure.

3.1.2 Mathematical model of a biological neuron

The nonlinear model is the smallest "constructional part" of an artificial neural network (or just neural network). This model consists of three main features.

1. *The synaptic weights.* Each input signal is multiplied with a synaptic weight before it reaches the summing neuron. This weight gives the actual "strength" of the input. The larger the weight, the more influence, of the output, has that particular input, compared to other inputs with smaller weights.

2. *The summing junction* adds all incoming signals (multiplied with their

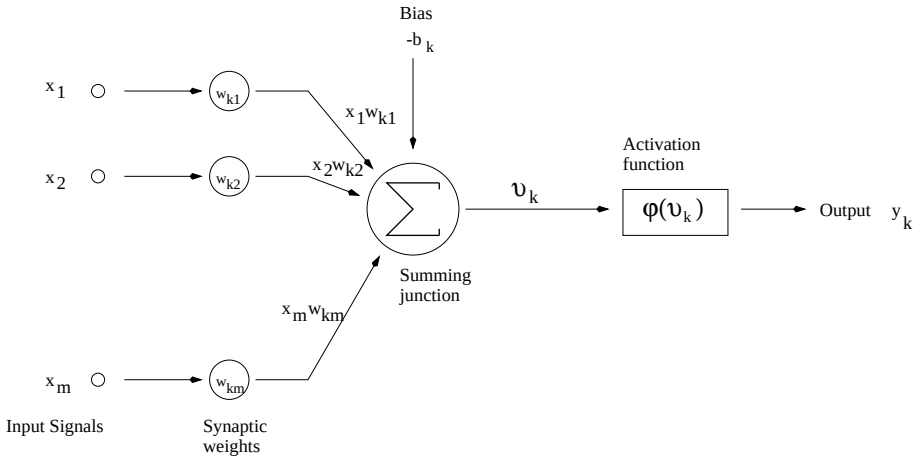

Figure 3.2. A mathematical nonlinear model of a biological neuron.

respective weights). This feature has its counterpart in the soma where all the signals are summed (Fig. 3.2).

3. *An activation function* (nonlinear function) then limits the output to some permissible value. The counterpart in the brain is the threshold value. If the summation is lower than the threshold, a signal is not propagated through the axon.

In the model there is also an external *bias* added. The effect of this bias is to decrease or increase the net input to the activation function. The most common activation function is the sigmoid function, which can be seen as a combination of nonlinear and linear behavior. An example of the sigmoid function is the *logistic function*, defined by

$$\varphi(v) = \frac{1}{1 + \exp(-av)} \tag{3.1}$$

where a is the *slope parameter* of the function. It can easily be seen that if the slope parameter approaches infinity (Fig. 3.3), the logistics function more or less becomes a threshold function (all or nothing—output values: 0 or 1).

Furthermore, the most favorable characteristics of this function and the reason why it is used in neural networks is that it is differentiable whereas a true threshold function is not.

3.1.3 Network architectures

In order to solve parallel computational problems and to mimic the large number of synapses in the brain, the neuron model must be connected to

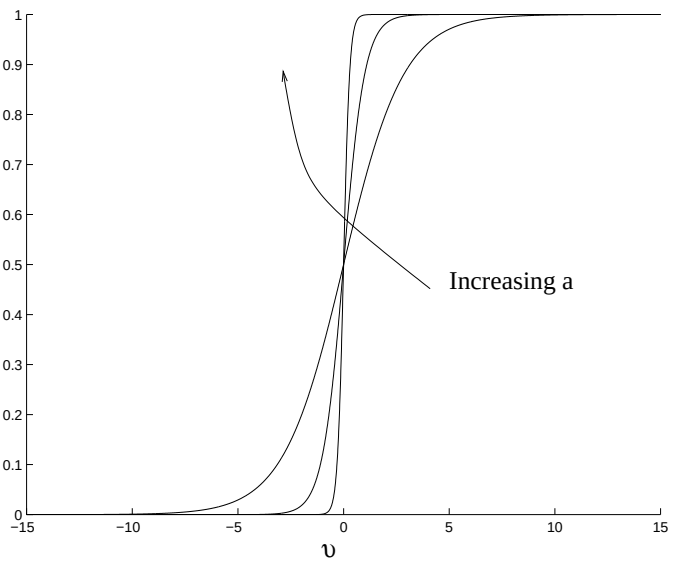

Figure 3.3. Sigmoidial functions with different slope parameters a.

other neurons. In general, one can identify three fundamental classes of network architectures. Two of these are feed-forward networks, i.e the information is passed forward, in the network, from the input source nodes to the output nodes, but not vice versa. The last class of networks is called recurrent networks, i.e. the information from the output layer is passed back to the input layer. Since these networks are not frequently used in biological pattern recognition, they will not be covered further in this report. The two feed-forward networks are the:

Single-layer feed-forward networks The simplest form of a layered network, a network that is organized in forms of layers, is a single layer feed-forward network. The information is only passed forward in this model but the complexity of this structure is enough to solve simple problems.

Multilayer feed-forward networks The multilayer feed-forward networks are commonly referred to as multilayer perceptrons (MLP). This type of network is a generalization of a single layer network, but their key feature is the presence of one or many hidden layers of computational nodes (hidden neurons). These neurons enable the network to learn complex tasks by extracting more information from the input data. In the most common structure the outputs from the first hidden layer are used as inputs to the next hidden layer and so on. The outputs from the last layer are used as inputs to the output layer. A MLP is also

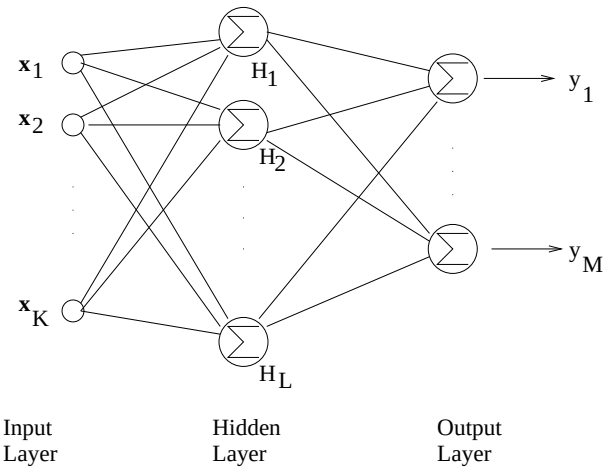

Figure 3.4. A MLP with a input layer, one hidden layer and a output layer. Note that the weights and the inlinearities are not explicitly shown in this figure.

said to be fully connected if every neuron in each layer is connected to every other neuron in the adjacent forward layer. Since there is no way of knowing the optimum number of hidden neurons or if a hidden layer should be used at all, optimization of the architectures must be performed.

3.1.4 The learning algorithm

The hardest part when it comes to learning how a neural network works, is to understand how the synaptic weights are adjusted during the training process. It was not until 1986 a solution to this problem, Rumelhart's, Hinton's and Williams' *error back-propagation algorithm* [24], was published[1]. Basically, the error back propagation algorithm consists of two phases. In

[1] In 1986 the development of the *back-propagation algorithm* was reported by Rumel-Hart, Hinton and Williams. In that same year, the two-volume book, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition* [25], by Rumelhart and McClelland, was published. This latter book has been a major influence in the use of back-propagation learning, which has emerged as the most popular learning algorithm for the training of multilayer perceptrons. In fact, back-propagation learning was discovered independently in two other places about the same time [20, 11]. After the discovery of the back-propagation algorithm in the mid-1980s, it turned out that the algorithm had been described earlier by Werbos in his Ph.D. thesis [29] at Harvard University in August 1974; Werbos' Ph.D. thesis was the first documented description of efficient reverse-mode gradient computation that was applied to general network models with neural network arising as a special case. It is most unfortunate that Werbos' work remained almost unknown in the scientific community for over a decade.

the first phase, *the forward pass*, the information is passed through the net and a set of outputs is produced. During this pass, all the synaptic weights in the network are fixed. In the next phase, *the backward pass*, the synaptic weights are changed according to an error-correction rule, i.e. the outputs from the forward pass are subtracted from a desired outputs and the error is passed back through the network, hence the name error-back propagation. The weights are then adjusted so that the output of the network becomes closer to desired output. The error at the output neuron i at iteration n, i.e training example n, can be described as

$$e_i(n) = d_i(n) - y_i(n) \tag{3.2}$$

where $y_i(n)$ can be calculated according to (denotations as in Fig. 3.2)

$$H_j = \varphi\left(\sum_{k=1}^{K} w_{jk}x_k - b_j\right) \tag{3.3}$$

$$y_i = \varphi\left(\sum_{j=1}^{L} w_{ij}H_j - b_i\right) \tag{3.4}$$

where $w_{i,j}$ is the weight of the connection from unit j to i and H_j is the output value of the hidden unit j. b_j is the bias for the hidden unit j and x_k is the value of the input unit k. The instantaneous value of the error energy for neuron i is then defined by $\frac{1}{2}e_i^2(n)$. The instantaneous value of the total error energy $\xi(n)$ is then obtained by adding the instantaneous value from all neurons in the output layer

$$\xi(n) = \frac{1}{2}\sum_{j\in M} e_i^2(n) \tag{3.5}$$

If we let N denote the total number of training examples, then the *mean squared error* (MSE) energy is defined as

$$MSE = \xi_{av} = \frac{1}{N}\sum_{n=1}^{N} \xi(n) \tag{3.6}$$

where ξ_{av} also can be seen as a function that depends on the weights in the network. The most commonly used learning algorithm, the standard *back-propagation algorithm* (BP), uses gradient decent for the minimization of ξ_{av}. However, this algorithm requires a lot of memory, especially with large-scale problems, and is also dependent of the user dependent parameters such as the learning rate and momentum constant. Therefore, the *scaled conjugate gradient* (SCG) algorithm [16], which is reported to be both memory efficient and better on large-scaled problems [26], was used in this study.

3.1.5 Generalization

When a neural network is to be trained, a very important factor to remember is to train the network to generalize well. The aim of the training is to make the network learn well enough from the training data so that it is able to generalize to other data that have not been shown before. Burghes [5] describes the generalization problem like this:

> A machine with too much capacity, the ability to learn any training set without error, is like a botanist with a photographic memory who, when presented with a new tree, concludes that it is not a tree because it has a different number of leaves from anything she has seen before. A machine with too little capacity is like the botanist's lazy brother, who declares that if it is green, it is a tree. Neither can generalize well.

A standard tool in statistics to check whether a model has been trained to generalize well is called *cross validation* [27]. First, the training set is randomly divided into a larger training part (training set) and a smaller test part. Then, the test part is further partitioned into two equally large sets; a validation set and a test set. The validation set is used for the validation of a specific network, whereas the test set can be used for comparisons of the performance of several networks. The reason for this "double validation" is that there exists a small chance of the model ending up over-fitting the validation set. The validation and test set make it possible to determine the multilayer network with the best number of hidden neurons, and when it is time to stop training the network.

There exist several cross validation techniques, but a very common and a fairly simple technique is the method of *early stopping* (ES). After a period of training, the synaptic weights and bias levels are fixed, and a forward pass is initiated with the validation set as input set. The error of each example of the validation set is then calculated, and the MSE is calculated. The MSE for both the training and the validation set usually decreases during the first epochs (one round of calculation of entire training set), but at some time the weights are adjusted to favorize the training set over the validation set. At that point, the training is stopped and a good generalization is preserved in the network (Fig. 3.5). Another method to avoid over-fitting by reducing the number of connections during training, is called *weight decay* (WD). The synaptic weights in a neural network can be grouped into two categories, those that have large influence on the network (model) and those that have little or no influence on it. The weights in the latter category are referred to as excess weights, and these weights often result in poor generalization, due to their high likelihood of taking on completely arbitrary values. Reducing these weights by the weight decay method may improve the generalization.

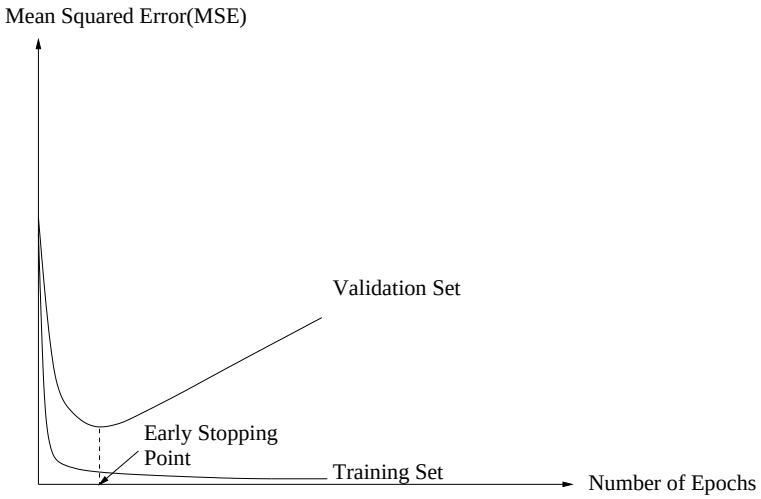

Figure 3.5. A graph describing the method of early stopping.

3.2 Encoding biological sequence data

Only two years after Rumelhart published the back-propagation algorithm in 1986 [24], Qian & Sejnowski used it for secondary structure problems [21]. After this pioneer work, neural networks have been used frequently in biological problems concerning pattern recognition. But how is the sequence of letters in for example a DNA or amino acid sequence coded into digits that a machine learning technique such as a network can use?

There exist several ways of coding a DNA sequence (four bases) or a amino acid sequence (twenty amino acids). Obviously, these letters have different chemical and physical properties and one way would be to collect and incorporate as much information as possible of each letter into a coding scheme. Holley and Karplus [9] explored the possibility to use hydrophobicity as coding scheme in 1989, but with poor results. Today, the most used coding style, for proteins at least, is the *unary format* or *sparse encoding scheme* [23]. According to this format, each residue (amino acid) is encoded by a vector that is 20 elements long if gaps are encoded as 20 zeros, or 21 elements if gaps are encoded as the 21st position in the vector:

```
ALA:  10000000000000000000
CYS:  01000000000000000000
ASP:  00100000000000000000
  ⋮
```

The advantage of the sparse encoding scheme is that each amino acid is given equal weight and thus no bias is introduced. The main disadvantage is that

it results in a large number of network parameters. When introducing the training examples to the model, most machine learning techniques used for secondary structure prediction "attempt" to predict the secondary structure for each residue. However, the information in each vector (see above) for each residue is not enough to give a proper classification. Therefore, almost all predictors incorporate information from adjacent residues when predicting the secondary structure of a residue. The number of adjacent residues (together with the residue to be predicted) is called the *input window size* or just *window size*. For ordinary secondary structure prediction, the window size ranges from 9 to 21 residues, but this value has to be optimized for each experiment.

3.3 PSI-BLAST

Instead of using a single amino acid at a given position in the query sequence, it is better to use a combination of amino acids known to be present at the same position in that protein and related ones. A version of BLAST [1, 19] called *Position-Specific-Iterated* BLAST or *PSI-BLAST* [2, 13] has been designed to provide information on this combination of amino acids, starting with a BLAST search by a single query sequence.

PSI-BLAST will search a protein sequence database with a query sequence motif, a matrix with rows representing sequence positions and columns representing variations in that position. The motif represents the observed variations in the alignment of a set of related proteins.

Once the motif has been found, the frequencies of amino acids in each column are adjusted by weighting the sequences to reduce the influence of the more alike sequences, and by adding more counts (pseudo-counts) representing other amino acid substitutions found among the observed types in to increase the statistical power of the matrix.

Iterated profile search methods have led to biologically important observations but, for many years, were quite slow and generally did not provide precise means for evaluating the significance of their results. This limited their utility for systematic mining of the protein databases. The principal design goals in developing the PSI-BLAST were speed, simplicity and automatic operation. The procedure PSI-BLAST uses can be summarized in five steps:

1. PSI-BLAST takes as an input a single protein sequence and compares it to a protein database, using the gapped BLAST program [2].

2. The program constructs a multiple alignment, and then a profile, from any significant local alignments found. The original query sequence serves as a template for the multiple alignment and profile, whose

lengths are identical to that of the query sequence. Different numbers of sequences can be aligned in different template positions.

3. The profile is compared to the protein database, again seeking local alignments. After a few minor modifications, the BLAST algorithm [2, 1] can be used for this directly.

4. PSI-BLAST estimates the statistical significance of the local alignments found.

5. Finally, PSI-BLAST iterates, by returning to step 2, an arbitrary number of times or until convergence.

Other approaches to encode the sequence data is discussed in [7, 8].

3.4 RMSD

The difference in structure between two fragments can be measured with *root mean squared deviation*, *RMSD*, according to

$$RMSD = \sqrt{\sum_{n=1}^{N} \frac{(q_n - q'_n)^2}{N}} \tag{3.7}$$

q_n is the coordinates of the nth residue in one fragment and q'_n is the coordinates of the nth residue in the other fragment. The more similar the fragments are, the lower is the RMSD value. This is done to give the artificial neuronal network examples to learn from and to calculate the right answers which the network can be validated against.

3.5 Balanced and unbalanced training sets

An important feature when it comes to the classification, is that the similar fragments constitute a very small portion of the total data set. This has to be compensated for, because if the machine learning tool is to be trained with a training set that shows the natural proportions of similar/non similar fragments, i.e. a *fully unbalanced set*, the predictor will most certainly classify all training examples into the most numerous class, i.e. the non similar class. If, on the other hand, the predictor is trained with equal number of examples, a so called *balanced* set, the predictor will probably under-predict the most numerous class, when presented with a test set that shows the natural proportions. The optimum choice is, of course, somewhere in between, i.e. an unbalanced set, but not fully unbalanced. In this study, unbalanced training sets, with 1-5 times as many examples from the overrepresented class were used.

3.6 Measuring the performance of a learning machine

There are numerous ways of measuring the performance of a predicting machine. In a two-class prediction, such as a similar/not similar prediction, the most common measures can be derived from the scalar quantities [26]:

p the number of correctly classified similar fragments.

n the number of correctly classified non similar fragments.

o the number of non similar fragments incorrectly classified as similar fragments (over-prediction).

u the number of similar fragments incorrectly classified as non similar fragments (under-prediction).

t the total number of fragments.

Another way to visualize and arrange the four quantities is to use a *contingency* or *confusion matrix*:

$$C = \begin{pmatrix} p & u \\ o & n \end{pmatrix} \tag{3.8}$$

The most frequently used measure to describe a predictor's overall performance is Q_{total}, the percentage of correctly classified residues:

$$Q_{total} = (p + n) \times 100/t \tag{3.9}$$

This measure gives fairly reliable information if the classes contain approximately the same number of examples. Otherwise, the Q_{total} value can be misleading. A much used alternative measure, that takes into account the under- and over-predictions, is the *Matthews Correlation Coefficient* (MCC) [15]:

$$MCC = \left(\frac{pn - ou}{\sqrt{(p+o)(p+u)(n+o)(n+u)}} \right) \tag{3.10}$$

As all correlation coefficients, MCC is a value between -1 and 1, where a 1 means a fully perfect prediction and -1 means a fully imperfect prediction.

Chapter 4

Experiments and results

The goal was to tell how similar the structure of two protein fragments are. The prediction should be based on the fragments' sequences of the amino acids. The general idea was to give an ANN the sequence information as well as a correct measure of the similarity between pairs of fragments, to train on. Thereafter, only the sequence information for fragment pairs, with fragments taken from other proteins, are given to the ANN and the ANN predicts the structure similarity. Finally the ANN prediction is evaluated against the correct answer. To make the prediction as good as possible, many parameters can be varied. For example, the fragment size and the construction and training method of the network. The experiments was done to examine how these parameters influenced the prediction quality.

Before the sequence could be used, it had to be encoded in a way understandable for the network. Also the correct RMSD values, used to compare the network output with, had to be calculated, based on the structure known for the fragments.

4.1 Implementation

A program, in the Perl language, was implemented to format the input data, run a neural network with the right parameters and data, and to evaluate the results. The requirement to be able to vary many variables made it more complicated. For the neural network, Matlab [14] with the Netlab library [18, 17] was used. In order to optimize the prediction many different variables were altered as described in the following sections.

4.1.1 ANN input data

The input data, with the sequence information, was taken from PSI-BLAST (see Section 3.3) output files already prepared. Depending on the window size, the PSI-BLAST files were splitted in portions representing the windows

and formated so the PSI-BLAST data could be used as input data to the network. A distinction is made between *window size* and *fragment size*. Window size means the number of residues the network uses as input data, i.e. gets it's sequence information from. The fragment size means the number of residues the structure difference is calculated on. This is the fragment the network is trying to predict. The distinction is made because it is possible that the residues close to the observed fragment affects the structure. In all tests the window size is larger or equal to the fragment size. In order to get as many fragments as possible, overlapping fragments could be used. For example the first fragment could be constructed of residue one to nine, the second from residue two to ten and so on. However, with a large supply of proteins, this was not done. Instead the next fragment started where the previous ended. In the cases where the fragment size was smaller than the window size, the next window started where the previous ended and the residues in the middle (in appropriate amount) of the window was devoted to the actual fragment which should be predicted.

In the input data every residue is encoded by 20 elements as described in Section 3.2. With a window size of, for example, 9 residues the input will have 180 elements and thus there have to be 360 input neurons in the network for the two fragments.

4.1.2 Reference output data

For the training, the network needed correct reference output data to learn from and the correct values were also needed for validation (as the right answers to compare the network prediction with). The comparison data were calculated by a RMSD program (as described in Section 3.4) written in the C language. However, it had to be adjusted to fit this task. As input, the RMSD program reads files in the *Protein Data Bank* (PDB) [22] format. The available files represented whole proteins. Therefore the appropriate data for the current fragment had to be extracted and stored in separate files. The fragments were chosen as described in previous section. The RMSD values were calculated on the coordinates for the C_α atoms included in the prepared files.

In some tests the raw RMSD values were given to the network. In other tests the fragment pares were classified depending on the value and the network was given a 0 or 1 representing similar and not similar. When RMSD values were given to a network which used a logistic output function, large values had to be cut off, not to exceed the range of the function. Values larger than the cut off limit were set to the limit value as

$$f = min(RMSD, limit) \tag{4.1}$$

In this study, the value 3 was used as limit. To fit the function range (0–1), all RMSD values, including the limit between similar and not similar, thereafter

were divided by the cut off value.

$$g = \frac{f}{limit} \tag{4.2}$$

The RMSD data were then combined with the input data in a format feasible for the network. In each case one set was made for training and one was made for validation. Training and validation sets have the same format. They just have different purposes and the validation set generally is much smaller then the training set.

4.1.3 Neural network

Two layer feed-forward networks were created with Netlab. The weights were drawn from a zero mean, unit variance isotropic Gaussian, with variance scaled by the fan-in of the hidden or output units as appropriate. The hidden units used the *tanh* activation function. In the output unit layer linear and logistic activation functions were used. The networks were optimized with the training sets using *scaled conjugate gradient* (SCG) optimization. The input part of the validation sets was forward propagated through the network to generate predictions.

4.2 Training and validation sets

Plenty of protein data was available to work with. The size of the training and validation sets was instead limited by the computational power. Matlab which was used in these experiments is easy to program and work with, but requires much memory and processor power. Therefore the sets had to be cut down in some tests and could be extended in some for a broader set. Because of this variation in sets from test to test, all results can not be compared to each other. In this thesis, care has been taken to that. So only comparisons between tests with the same training and validation sets has been done. Other comparisons between tables may not be appropriate. A typical training set consisted of 30,000–150,000 pairs and the validation set of 2,000–70,000 pairs. The proteins were randomly selected from a list with available proteins, with no proteins occuring both in the validation and the training set. With a few exceptions, all fragments, from the proteins chosen for training, were combined pairwise with all others, constituting a training set. The validation sets were constituted in a similar way. The output function tests, presented in Section 4.4, used 32 different validation sets. In each set, one selected fragment was matched with all other fragments in the validation set. The proportion of similar pairs was very small with the limit used. This sometimes made it more complicated to reach good results.

Fragment	Linear			Logistic		
	Share	Ratio	MCC	Share	Ratio	MCC
d1r69-1	**0.59**	0.34	0.273	0.41	**0.43**	**0.289**
d1ctf-4	**0.33**	0.51	**0.310**	0.27	**0.52**	0.278
d2cro-1	**0.41**	**0.46**	**0.321**	0.29	0.45	0.257
d2cro-2	**0.33**	**0.51**	**0.309**	0.17	0.49	0.206
d1r69-1	**0.88**	0.30	0.306	0.83	**0.37**	**0.393**
d1ctf-4	**0.59**	0.51	0.432	0.55	**0.57**	**0.458**
d2cro-1	0.59	**0.51**	**0.433**	**0.74**	0.43	0.431
d2cro-2	0.18	0.56	0.244	**0.25**	**0.64**	**0.328**
d3icb-1	0.11	0.67	0.216	**0.13**	0.67	**0.235**

Table 4.1. Comparison between linear and logistic output function. Evaluated with the share of similar pairs which was found, the ratio of the predicted which were similar (correct) and MCC. The best values are marked with bold text. The tests were carried out in two different rounds with different training set.

4.3 Number of training cycles

As discussed in Section 3.1.5 the network can be over-trained. During the test the optimal number of training cycles, i.e. the number of cycles giving the smallest MSE (mean squared error, Eq. 3.6) for an independent control set with unrelated fragments, varied between 2 and 140. It was often around 20–25. But the performance of the network was generally unstable after a few cycles, in the sense that it altered drasticly from cycle to cycle. The generalization did not dramatically decrease after more cycles. The MSE was calculated after each training cycle and the training then continued with the next cycle. It is impossible to know which cycle is the best before all cycles are evaluated and can be compared. Since the network at that time already is modified by the further training, most of the tests were carried out with 150 training cycles regardless of which cycle had the best value.

4.4 ANN output unit activation function

Two test rounds, with different training and validation sets, were carried out to compare *linear* and *logistic* output unit activation functions. In the second round, with the largest training and validation sets, 5 fragments out of 32 got a reasonable number of similar matches (Table 4.1). In this case reasonable means about 700 out of 3,700. The training set consisted of 62,000 pairs. Most of the other fragments got less than 10 similar matches and many of them no matches at all.

Balance ratio	Initial est	Average of 5 tests
0	0.010	
1	0.068	
2	0.113	
3	**0.125**	0.149
4	0.118	**0.157**
5	0.113	

Table 4.2. Balance between similar and not similar training samples, evaluated with MCC. The best values are bolded

Even Tables 4.4 and 4.6 can be used for evaluating the output functions, while constant training and validation sets as well as other parameters, were used throughout those experiment.

4.5 Balance

The balance is the ratio between similar and not similar samples. Balance 5 means there is five times as many not similar as similar samples. As I considered it incorrect to alter the balance of the validation sets (the network should after all predict realistic sets) only the balance of the training set was altered.As seen in Table 4.2, the best results are made with a balance around 3 or 4, varying from test to test. In most of the further experiments ratio 3 is used.

4.6 Fragment size

All fragment prediction is based on fragments with 9 residue. It will be easier to predict shorter fragment and harder to predict longer as they are more complex. But where to compromise between the quality of prediction and the length of the fragment. It is not an obvious task to value the worth of a longer or shorter fragment and thus to say which prediction is most successful.

4.7 Window size

With the assumption that the neighbor residues on each side of the fragment influence the structure different window sizes were tested. Since the fragment size 9 was used in all tests, a window size of 11 means one residue on each side is taken in concern. Window size 13 means two on each side and so on.

Number of Residues	RMSD logistic	classes linear
9	0.110	0.121
11	0.100	0.149
13	0.112	0.152
15	0.166	0.183
17	**0.168**	**0.194**
19	0.144	0.150
21	0.158	0.137
23	0.125	0.107
25	0.126	0.103

Table 4.3. Window size, evaluated with MCC. Except for the classification (see Section 4.10) and the output activation function, the network parameters are the same.

According to the method to generate fragments described in 4.1.1, windows of size 9 gave rise to more fragments than windows of size 25 with the same set of proteins. The discrepancy in number of fragment pairs (the number of fragments to the power of two) was too big to be useful. Either the sets were too big or too small. Big sets gave rise to more small (size 9) pairs of fragments than the computer could handle. Small sets gave too few big (size 25) pairs to get reliable results. Instead the same fragments had to be used throughout the experiment. Using the vocabulary in 4.1.1, the window size was fixed to 25 but the sequential information was only taken from the appropriate residues (9–25) in the middle.

Table 4.3 clearly shows the advantage of making the window bigger than the fragment. It is harder to decide an exact limit. The MCC values decrease with windows wider than 17 residues, despite that no information is removed. One theory is that a higher noise (not useful data) ratio affects the net negatively.

4.8 Number of neurons in the hidden layer

Due to earlier studies, most of the networks used had seven neurons in the hidden layer. The experiments presented in Table 4.4 gave better results for a few more neurons. But the experiments also show that too many neurons give a lower MCC value, probably due to poor generalization (see Section 3.1.5).

Hidden neurons	Output function	
	linear	logistic
5	0.181	0.189
7	0.182	**0.222**
10	**0.208**	0.203
20	0.203	0.171
50	0.178	0.185
100	0.187	0.204

Table 4.4. Number of neurons in the hidden layer in the network, evaluated with MCC. The results differentiate between linear and logistic output unit function.

Window size	RMSD limit/Å	
	0.9	0.7
9	**0.163**	0.127
11	**0.153**	0.149
13	**0.184**	0.158
15	**0.188**	0.173
17	**0.199**	0.179
19	**0.116**	0.107
21	**0.135**	0.122
23	**0.244**	0.130
25	**0.159**	0.106

Table 4.5. RMSD limit, separating similar and not similar pairs, evaluated with MCC

4.9 RMSD limit

If two fragments are similar or not is decided by the limit defined for the RMSD value. In this project advices from earlier studies were taken and just a few experiments were carried out to examine the best limit. One principle for deciding which limit is best is to maximize the MCC value, thus achieving the most significant results. Table 4.5 shows a bit better MCC values for 0.9 Å than for 0.7 Å. But even if better MCC values are received with higher limits, there is a trade off, in the sense that fragment pairs which are less similar are let through.

RMSD		classes	
linear	logistic	linear	logistic
0.121	**0.222**	0.176	0.205

Table 4.6. Comparison between using raw RMSD values and discrete classes (0 or 1) as network output unit values, in training and validation. Evaluated with MCC.

4.10 Classes or RMSD values

As mentioned before, the network output data in training and validation sets can be encoded in two ways. Either the data can be encoded as the exact, raw RMSD values, calculated by the RMSD program, or as two discrete classes, representing similar or not similar. In this study the values 0 and 1 are used for those classes. A RMSD value contains more information and is the only alternative, if the exact RMSD value is desired. If, on the other hand, a classification is sufficient, the class may be easier, for the network, to predict. Table 4.6 does not show large discrepancies between the encodings, but larger discrepancies between output unit functions than in previous experiments.

Chapter 5

Conclusions

For several altered parameters, clear reslults have been achieved and optimal values has been found. The optimal window size was the primary question. Even if the achieved results are not always very impressive, the distinction between compared valued create a basis for future work and improvments.

5.1 Optimal parameter values

The residues adjacent to the fragment, clearly influenced the structure. With a fragment size of 9 residues, 15–17 residues taken in consideration gave the best results. More than 17 residues negatively affected the results and were more uncertain. Wider windows also mean more data, which requires more resources.

A fairly low number of neurons in the hidden layer, around 7–10, has proven best in this, as well as in earlier, studies. The exact number probably depends on the number of input neurons, which depends on the window size and the sequence encoding.

In this type of prediction where the rate of similar pairs is low, a balanced training set gives better predictions, with an optimum around 3–4 times as many non similar as similar pairs.

The results from the output unit activation function comparisons are varying. In Table 4.1 the logistic function gave slightly better results in the second, larger test. In the first, though, the linear function generally performed better. In Table 4.6 the logistic function is superior. The type of structures and the frequency of similar pairs may have been sources of influence. Since the logistic function rarely performed significantly worse and often significantly better than the linear function, the logistic function is recommended.

Using around 150 training cycles is proposed in this experiment. Although fewer cycles sometimes gave a bit lower MSE, the values were unstable from cycle to cycle. Probably the MSE would vary from set to set of

validation pairs.

The RMSD limit, separating similar from non similar pairs, has not been exhaustively examined. The comparison between 0.7 and 0.9 Å, gives favor to the higher value. The trade off is that the pairs considered as similar are not as similar as with a stricter limit. The limit should therefore be chosen suitable for the classification needed.

Although it may sound easier to train on a class than on an exact RMSD value this was not the case in this study (MCC 0.205 against 0.222 with logistic output function). When the classes 0 and 1 were used, the cut off was set in the middle, i.e. at 0.5. Since the RMSD prediction also supplies more information, embodied in an RMSD value, this method must be preferred in most cases. Notable, the training and validation on classes gave better results when a linear output function was used (0.176 against 0.121).

5.2 Further suggestions

There are also other factors, which further studies are suggested to examine. Over all, the MCC values are not very impressive. A low rate of similar fragments probably affected them. With predictions made on selected classes of fragments the results may be better.

The sequence input data was made with PSI-BLAST in this study. There are other ways to produce "pseudo counts", which could be evaluated for this task. Different methods may be more suitable for different kind of structures.

RMSD was used to calculate the structure similarity, but other distance measurements could be used.

Bibliography

1. S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.
2. S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.
3. C. B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 8:223–230, 1973.
4. C. Bränden and J. Tooze. *Introduction to Protein Structure*. Garland Publishing, New York, 1991.
5. C. J. C. Burges. *A Tutorial on Support Vector Machines for Pattern Recognition*. Kluwer Academic Publishers, Boston, 1998.
6. S. Haykin. *Neural Networks. A Comprehensive Foundation*. Prentice Hall, New Jersey, 1994.
7. J. G. Heinkoff and S. Heinkoff. Using substitution probabilities to improve postition-specific scoring matrices. *Cabios*, 12(2):135–143, 1996.
8. S. Heinkoff. Scores for sequence searches and alignments. *Current Opinion in Structural Biology*, 6:353–360, 1996.
9. H. Holley and M. Karplus. Protein secondary structure predicition with neuronal network. *pnas*, 86:152–156, 1989.
10. L. Hunter. *Artificial Intelligence and Molecular Biology*. AAAI Press, Menlo Park, 1993.
11. Y. LeCun. Une procedure d'apprentissage pour reseau a seuil assymetrique. *Cognitiva*, 85:599–604, 1985.
12. D. Lundh, B. Olsson, and A. Narayanan, editors. *Biocomputing and Emergent Computation*, Skövde, 1997. BCEC97, World Scientific.
13. T. Madden, S. B. Shavirin, A. Schäffer, and A. Egorov. PSI-BLAST search. http://www.ncbi.nlm.nih.gov/blast/psiblast.cgi.
14. MathWorks. Matlab. http://www.mathworks.com/products/matlab/.
15. B. W. Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta*, 405:442–451, 1975.
16. M Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6(4):525–533, 1993.

17. I. Nabney. Netlab software. http://www.ncrg.aston.ac.uk/netlab/.

18. I. Nabney. *Netlab: Algorithms for Pattern Recognition.* Advances in Pattern Recognition. Springer-Verlag, Heidelberg, 2001.

19. NCBI. NCBI BLAST web site. http://www.ncbi.nlm.nih.gov/blast/.

20. D. B. Parker. Learning-logic: Casting the cortex of the human brain in silicon. Technical Report TR-47, Center for Computational Research in Economics and Management Science, MIT, Cambridge, Massachusetts, 1985.

21. N. Qian and M. Karplus. Predicting the secondary structure of globular proteins using neuronal network models. *jmb*, 202:865–884, 1988.

22. Research Collaboratory for Structural Bioinformatics. The RCSB Protein Data Bank. http://www.rcsb.org/pdb/.

23. S. K. Riis and A Krogh. Improved prediction of protein secondary structure using structured neuronal networks and multiple sequence alignments. *Journal of Computational Biology*, 3:163–183, 1996.

24. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations of back-propagation errors. *Nature*, 323:533–536, 1986.

25. D. E. Rumelhart and J. L. McClelland. *Parallel Distibuted Processing: Explorations in the Microstructures of Cognition.* MIT Press, Cambridge, Massachusetts, 1986.

26. A. J. Shepard, D. Gorse, and J. M. Thornton. Prediction of the location and type of β-turns in proteins using neural networks. *Protein Science*, 8:1045–1055, 1999.

27. M. Stone. Cross-validatory choise and assessment of statistical predictions. *Journal of Statistical Society*, B36:111–133, 1974.

28. L. Stryer. *Biochemistry.* Freeman, New York, 4 edition, 1995.

29. P. J. Werbos. *Beyond regression: New tools for prediction and analysis in the behavioral science.* PhD thesis, Harvard University, Cambridge, Massachusetts, 1974.