Prediction of the quality of protein models using neural networks

Björn Wallner [*]

Supervisor: Arne Elofsson, Stockholm Bioinformatics Center,Stockholm University
Examiner: Olle Edholm, Theoretical Physics, Royal Institute of Technology

June 19, 2001

[*]Stockholm Bioinformatics Center, Stockholm University, SE-106 91 Stockholm, Sweden
E–mail: bjorn@sbc.su.se

Abstract

Models of proteins are made to help our understanding of how a particular protein functions. However, no good predictor of the quality of the model exist. To address this problem neural networks are trained to predict quality of protein models. Besides, the possibility to measure the quality of a model, this might also be useful to increase the specificity of fold–recognition methods.

Here we generate a large set of models, using alignment methods and the homology model program Modeller [25]. The quality of these models were measured using a modified version of the LGscore [21].

The training was based on accessibility surfaces, the contacts between residues and contacts between 13 different atom types. The training was performed for different cutoffs. For the atom type contacts, networks were trained on ten cutoffs ranging from 3.0 Å to 5.25 Å in 0.25 Å intervals, the contacts with atoms in the same residue were omitted. For the residue contacts six cutoffs in the range 4 Å to 12 Å were used, only contacts between residues more than five residues apart in the sequence were counted, to avoid accumulation of contacts between residues laying close in the sequence. The accessibility surfaces were represented as fraction of <25%, 25%–50%, 50%–75% and >75% relative accessibility for each residue respectively.

A neural network was trained for every single combination of parameter type and a correlation coefficient for an independent test set was calculated as a measure of how good each network performed. For the atom contacts alone the best correlation, 0.70, was obtained with a 4.5 Å cutoff, for the residue contacts cutoff of 6 Å gave the best correlation, 0.63. For the accessibility surfaces high and low relative accessibility gave best correlation with 0.70 for low and 0.52 for high.

The different parameter types probably contain overlapping information, nevertheless if a network is trained on a combination of the best atom and residue contacts together with the accessibility surfaces a correlation coefficient of 0.83 is obtained. However on an additional test set generated from LiveBench–2 [39] data, this high performance is not sustained, the correlation coefficient for this set was only 0.50.

Contents

1 Introduction

The genome projects produce a growing number of protein sequences with unknown structure and function. To experimentally determine the structure and function of a protein is both difficult and time–consuming, thus the need for computer–aided sequences analysis is large.

The prediction of the three–dimensional structure of a protein only from the amino acid sequence has been a problem of major interest for many years. Approaches have ranged from purely *ab–initio* methods that are based entirely on physical chemical principles, to homology methods that are based primarily on the information available in sequence and structural databases. [1]

Homology modelling has recently assumed an increased importance in the era of structural genomics. One long–term goal of high–throughput experimental structure determination is to obtain enough protein structures so that the rest can be reliably predicted using homology methods. Homology modelling usually follows the steps outlined in figure 1.

The first step in homology modelling of an unknown protein is to search databases like, Protein Data Bank[1] [2], SCOP[2] [3], FSSP[3] [4] or CATH[4] [5]

[1]http://www.rcsb.org/pdb/
[2]http://scop.mrc-lmb.cam.ac.uk/scop/
[3]http://www2.ebi.ac.uk/dali/fssp.html
[4]http://www.biochem.ucl.ac.uk/bsm/cath/

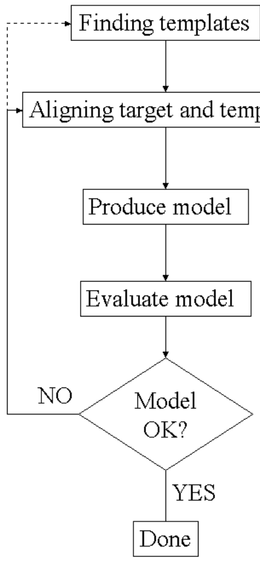


Figure 1: Steps in homology modelling of protein structure. First databases are searched for related structures, which can act as templates for the target sequence. The selected templates are aligned against the target sequence. Models are built based on the information in the template structure and the alignment. The quality of the model is assessed. Process might continue to produce better models.

for homologous proteins with known structure, which can be used as templates. There are three main classes of protein comparison methods that are useful in finding related proteins. The first is by pairwise sequence alignment, which is the classical way to detect related proteins. Frequently used programs in this class are FASTA [6] and BLAST [7]. The second class uses multiple sequence comparisons to improve the sensitivity of the search. A widely used program in this class is PSI–BLAST [8]. The third class are threading methods [9]. These methods involve the identification of a structural template that most closely resembles the structure of a query sequence, by "threading" the query sequence through a known structure and calculating some score based on the fit. Since 3D structure in general is more conserved than sequence, threading methods are useful when there are no proteins clearly related to the target sequence on the sequence level. [10]

When templates are found and alignments are determined there are a number of modelling building programs available, that can be used to construct a 3D model. The accuracies of the various programs seems to be relatively similar when used optimally [11, 12]. The template selection and alignment accuracy is usually of greater importance than the modelling program used for modelling, especially for models based on less than 40% sequence identity to the templates. [10]

Once a 3D model is determined the quality of the model must be determined in some way, in order to get some estimation if the model is good or bad and if the alignments needs to be changed and a new model produced. What is needed is some kind of discriminatory function. These functions can be simple, for example, counting atomic contacts in a given structure [13], or can involve elaborate calculations based on the physics of the system to determine the free energy of a conformation [14].

The physics based discriminatory functions, assumes that the correct native protein fold is at a global free–energy minimum [15]. Even if this is the case, it is not very helpful in devising a discrimination function, since the free energy is not a simple function of atomic coordinates, but also depends on the extent of motion and degree of order. Levitt and co–workers [16, 17] have tested a number of energy functions and found none of them to be completely satisfactory in discriminating incorrect from native structure. What they did found, was that simple functions often have as effective discriminating power as more complex function.

Another type of discriminatory functions are knowledge–based. These functions compile parameters from tendencies observed in a database of experimentally determined protein structures [18]. Generally, knowledge–based discriminatory functions have used a simple one– or two–point–per–residue representation, i.e each residue in protein sequence is represented with one or two positions in the three–dimensional space [19]. Discrimination can be based on each residue's preference to be buried or exposed, its preference to be in contact at a particular distance and sequence separation from other residues and its preference to be in a certain secondary structure conformation [20]. But in order to be able to discriminate between quite similar protein models (i.e 1 to 3 Å root mean square deviation) a more detailed representation is necessary which capture more details of the protein models [19]. A one–point–per–residue discriminatory function may not be able to discriminate as well as an all–atom discriminatory function, which takes into account the environment of all the atoms on the main–chain and the side–chain of each residue.

This study deals with this last step of homology modelling assessing the quality of a protein model. This problem is more complex than just discrimination of good from bad models, since it involves an estimation of how good the model really is.

To address this problem many homology models are produced of structural known proteins. And a set of neural networks are trained to predict the quality, as measured by a modified version of LGscore [21] between the models and the correct structures, which is a measure of how similar the two structure are. The models are represented by a number of different parameters, such as accessibility surfaces, the contacts between residues and contacts between 13 different atom types with different cutoffs. Besides, the possibility to measure the quality of a model, this might also be useful to increase the specificity of fold–recognition methods, whose aim is to identify the conformation a protein will adopt.

2 Background and Theory

2.1 Proteins

Proteins are build up by amino acids forming a peptide chain of variable length. There exists 20 different amino acids, the general structure of the amino acid is shown in figure 2. At one end, there is an alkaline amino group at the other an acidic carboxyl group. The difference between the amino acids are the side chain R, which can vary from just an hydrogen in glycine and methyl group in alanine to long acid or aromatic groups.

Proteins play a central role in many biological processes, since they make up the cell machinery. They are the active components which catalyse various processes, and in that way they control and direct chemical pathways behind all processes of the living cell. They can act as transporters, transporting oxygen in the blood or controlling the flow of ions across the cell membrane.

The three–dimensional structure of a protein is of great importance to its the function. The structure hierarchy is commonly divided in four levels. The primary structure which is simply the amino acid sequence. The secondary structure is the formation of main structural elements like helices and sheets. When these are coupled together one gets the tertiary structure, and finally the quaternary structure which is the term for several peptide chains bound

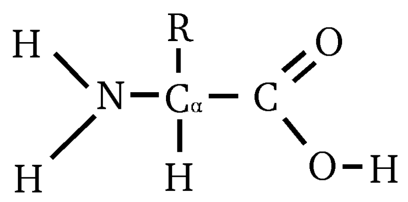


Figure 2: The general structure of an amino acid in non–ionized form. At one end, there is an alkaline amino group at the other an acidic carboxyl group. The R symbolizes the side chain, which differs between the different amino acids.

together, possible also with other elements such as metals.

To understand protein function the three–dimensional structure of the protein might help, therefore intense efforts have been made to understand the folding process.

2.2 Protein folding

Important for the folding of proteins are various interactions between the groups of amino acids and in particular their interaction with water. It has become clear that hydrophobicity plays a very important role in protein folding [22]. The hydrophobicity originates from the fact that water with polar properties interact unfavorable with non–polar amino acid side groups. A protein with both polar and non–polar groups would tend to obtain a structure where polar groups are turned towards the water and with non–polar ones sheltered inside the protein. This create rather compact structures.

Another crucial factor in protein folding, besides the hydrophobicity, is the forming of hydrogen bonds. They are important for stabilizing secondary structures. In a correct structure the fraction of hydrogen bonded atoms is optimized [23, 24], and in an incorrect structure one could expect this fraction to be lower [13].

Then of course electrostatics and van der Waals interactions also contribute to the complexity of the folding process.

2.3 Sequence alignment

Sequence alignment can be used to detect relationships between two proteins. All alignment algorithms uses some method to score relation between to amino acids. The simplest way to score to amino acids is by identity scoring, amino acid pairs are classified into two types: identical and non-identical. Non-identical pairs are scored 0 and identical are scored 1. But since the sequence is subjected to evolution, which means that amino acids sequence can change, due to substitutions, insertions and deletions, it is often more useful to use som other scoring scheme. The substitution follow a pattern, meaning that some amino acids are more likely to by interchanged. A change from a valine to an isoleucine is for example more likely to be found than a valine to a histidine change. Therefore more sophisticated scoring schemes have been developed, which score the equivalencing of each of the 210 possible pairs of amino acids (usually represented by a 20×20 matrix). To deal with the possibility of insertions and deletions, the alignment are allowed to have gaps. Introducing gaps in an alignment results in a gap–penalty on the score for the alignment.

Once the scoring scheme for amino acids and gaps is chosen, what is left is to found the best possible alignment between two sequences, i.e the one with the highest score. The naive approach to finding this alignment is to generate all possible alignments, add up the scores for equivalencing each amino acid pair and all gap penalties in each alignment and then select the highest scoring alignment. But even for two sequences of 100 residues there are $> 10^{75}$ possible alignments, which makes this approach infeasible. Fortunately, there is a group of algorithms using dynamic programming that can calculate the best scoring alignment in the order of mn steps, where m and n are the length of the two sequences to be aligned.

A way to improve alignment quality is to include information about the predicted secondary structure, which can be predicted with an accuracy of about 75% from the amino acid sequence [26]. That is, instead of only relating the amino acids based on residue type, they are also related based on secondary structure.

Another way to improve detection of remotely related proteins, i.e less than 30% sequence identity, is to use multiple sequence alignment [27]. Which makes it possible to detect a relationship between two proteins because they both are related to a third protein. From a multiple alignment it is also possible to produce a protein profile, which holds information on the conservation of different residues in the proteins. This approach does not only improve detection of remotely related proteins, it also gives a significant improvement in alignment quality, even at levels of sequence identity for which pairwise alignment methods are known not to work. [1]

2.4 Protein structure comparisons

There exist a number of methods to measure the similarity between a protein model [21] and the correct structure. The more similar the model and the correct structure are the higher quality of the model. The most common way to measure this similarity between a model and the correct structure is to calculate the root mean square deviation (rmsd) after an optimal superposition. The rmsd gives a measure of the average deviation at each residue. A problem with the rmsd measure is that it is a global measure. This means that the rmsd for model, which is mostly correct, but with one bad region can get very high.

Other methods, like LGscore [21] avoid this problem by finding the "most significant" segment in common between a model and the correct structure. This segment should ideally be as long and similar as possible. In this study a modified version of LGscore is used, the original method is modified in order to get better statistics for short fragments. For a detailed description see Quality measure under Methods.

2.5 SCOP

The Structural Classification of Proteins database [3], SCOP[5], is a hierarchical database, which provides a detailed description of the structural and evolutionary relationships of proteins whose three–dimensional structures have been determined. The current release (1.53) contains all 11 410 PDB entries from 1 Jul 2000. The quality of the database is very high since the classification is constructed manually by visual inspection and comparison of structures, but with the assistance of tools to make the task manageable and help provide generality.

Each protein is classified on four hierarchical levels, *family*, *superfamily*, *fold* and *class*. Proteins within a *family* have a clear common evolutionary relationship. They either have residue identities of 30% and greater, or lower sequence identity but very similar structure and function. That two families share the same *superfamily* means the proteins based on their structure and functional features might have a common evolutionary origin. Proteins with the same *fold* have the same major secondary structures in the same arrangement

[5]http://scop.mrc-lmb.cam.ac.uk/scop/

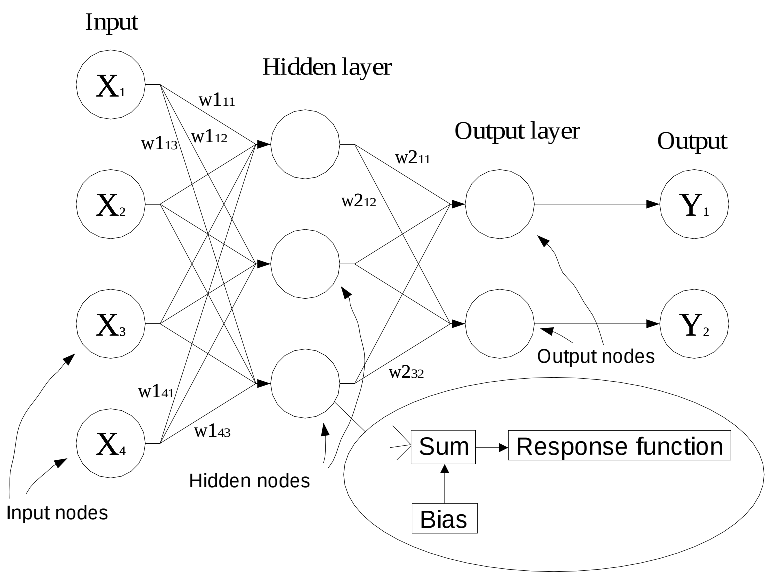


Figure 3: A schematic picture of a two layered neural network, with four input variables ($X_1, \ldots, X_4$), three hidden nodes and two output variables (Y_1, Y_2). For each connection (*synapse*) there is one weight, which is multiplied with the output from the previous node, when it is transfered through the synapse to the next node. The zoom–in shows what happens at each node. First the input to the node is summed and a bias or a threshold is added to this sum ($\sum_{i=0}^{4} X_i w1_{i3}$). This sum is then passed through a response function, which usually is a linear or a logistic function. Figure idea from Lundström [28].

and with the same topological connections. Different *folds* are grouped into *classes* like, all–α–helices, all–β–sheet or α and β.

2.6 Artificial neural networks

The theory of neural networks is based on the actual physiology of the human brain and show a great resemblance to the way our brain works. The building blocks of the neural networks are *neurons* or nodes that are connected with synapses, which has a weight to reinforce or repress signals. The nodes are grouped in *layers*, and the nodes in one layer only take input from the previous layer and give output to the next layer.

An example of a two layered neural network is shown in figure 3. The input variables are transfered through the synapses to the hidden layer of hidden nodes. A response is triggered and the result is sent to the output layer with output nodes, where a new response is triggered resulting in the output variables. At each node the input is summed and a bias or a threshold is added to this sum. This sum is then passed through a response function, which usually is linear or logistic.

A neural network is trained with a set of known input and output variables. From these examples the network tries to generalize and learn the functionality between the them. The learning or training consists of adjusting the weights and the biases in such a way that when the input variables are forwarded through the network, the error between the network output and the desired known output is minimized. Efficient algorithms have been developed to deal with this.

What is needed, is first an efficient way to calculate derivatives of the error function with respect to the weights (it can be shown that the biases can be included in the weights by adding an extra node [29], therefore biases will not be included explicitly in the derivation) secondly a way to go from the derivatives to the adjustments in the weights. These two methods are then repeated in an iterative manner for each training cycle. The method for calculating the derivatives is called error back–propagation, the term refers to the fact that the derivatives are calculated by propagate the errors backwards through the network. There are a number of methods to go from the derivatives to update the weights, the method used in this study is called scaled conjugate gradients.

2.6.1 Error back–propagation

This is the algorithm for evaluating the derivatives in an efficient way. Assume that the response function is chosen in such a way that it can be differentiable with respect to the input variables, the weights and biases Also assume that the chosen error function is differentiable with respect to the output variable. Then the derivatives of the error function with respect to the weights can be evaluated.

In a neural network at each node the following sum is calculated:

$$a_j = \sum_i w_{ji} z_i \tag{1}$$

where z_i is the response of node, which sends a connection to node j and w_{ji} is the weight for that connection. Summation taken over all nodes with connection to node j. This sum is then transformed through a response function $g(\cdot)$ to give the response z_j of node j:

$$z_j = g(a_j) \tag{2}$$

Since there is a special case when z_i is an input or an output node, x_i will denote input nodes and y_k will denote output nodes. The error function E is assumed to be a sum of errors over all patterns in the training set, of an error defined for each pattern separately

$$E = \sum_n E_n$$

E_n is assumed to be expressed as differentiable function of all its output variables

$$E_n = E_n(y_1, \ldots, y_c)$$

With this assumptions the derivatives with respect to the weights can be expressed as sums over the training set patterns of the derivatives for each pattern separately. Therefore only one pattern need to be considered at a time. By applying the chain rule

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}}$$

define

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j}$$

using equation 1

$$\frac{\partial a_j}{\partial w_{ji}} = z_i$$

gives

$$\frac{\partial E_n}{\partial w_{ji}} = \delta_j z_i$$

The derivative of the error with respect to the weight is obtained simply by multiplying the δ value for the node j at the output end of the weight by the value of the response for node i at the input end of the weight see figure 4a. Left to calculate is δ_j for each hidden and output node. For the output node using equation 2 with z_k denoted by y_k (because of the output node).

$$\delta_k \equiv \frac{\partial E_n}{\partial a_k} = \frac{\partial y_k}{\partial a_k}\frac{\partial E_n}{\partial y_k} = g'(a_k)\frac{\partial E_n}{\partial y_k} \tag{3}$$

once the response and error function are chosen this can be easily calculated. For the hidden nodes it is a bit more complicated because there is no explicit error function that can be calculated for each node. By using the chain rule

$$\delta_j \equiv \frac{\partial E_n}{\partial a_j} = \sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j}$$

and equation 1 and 2 gives

$$a_k = \sum_i w_{ki} z_i = \sum_i w_{ki} g(a_i) \Longrightarrow \frac{\partial a_k}{\partial a_j} = \sum_i w_{ki} g'(a_i)$$

which means that δ_j can be written as

$$\delta_j = g'(a_j)\sum_k w_{kj}\delta_k \tag{4}$$

where the sum runs over all nodes k that are connected to node j. The value of δ for a particular hidden node can be obtained by propagating the δ values backwards see figure 4b. Since the δ values for the output nodes are known from equation 3, it follows that values of δ for all hidden nodes can be calculated by recursively applying equation 4.

Error back–propagation is much faster way to calculate the derivatives then by numerically forward propagation. The numerically differentiation scale as $\mathcal{O}(W^2)$, since derivative scale as $\mathcal{O}(W)$ and that has to be evaluated W times, where the error back–propagation scale as $\mathcal{O}(W)$. This makes a huge difference in computational time.

2.6.2 Scaled conjugate gradients

The easiest way to change the weights from the derivatives is by changing the weights with a fixed step length in the direction of the negative gradient of the error function, this algorithm is known as gradient descent. However it turns

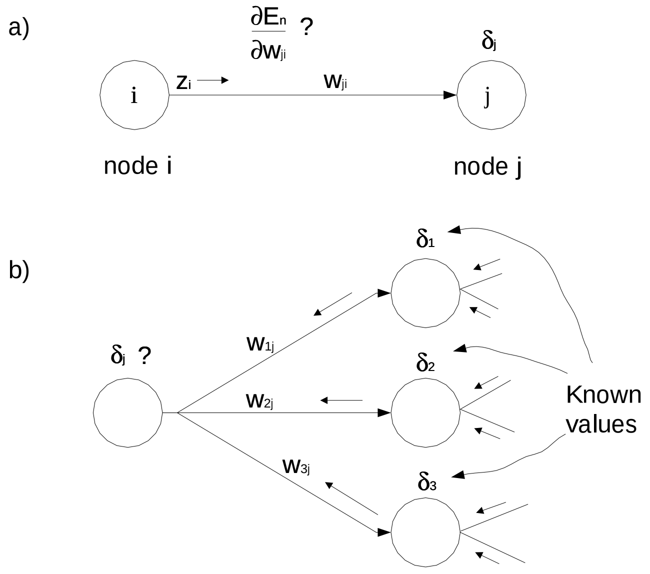


Figure 4: An illustration of how the error back–propagation algorithm works. The top figure shows which parameters are involved in the calculation of $\partial E_n/\partial w_{ji} = \delta_j z_i$. The lower figure illustrates how the δ values are calculated by propagated back through the network, $\delta_j = \sum_{k=1}^{3} w_{kj}\delta_k$.

out that this method converge rather slowly and that a more elaborate method is needed.

Scaled conjugate gradients algorithm is an improvement of gradient descent algorithm, because it choses the direction at each training cycle, as a linear combination of the negative current gradient and the the direction chosen in the previous step. This is more efficient since information from former steps is used, which is not used in the gradient descent algorithm. The step length, which in the gradient descent algorithm is fixed, is also calculated in an efficient way which minimizes the error in the chosen direction. For a full derivation of the scaled conjugate gradients algorithm see [30] or [29].

2.6.3 Over–training

Even tough it is certain that the error for the training data will decrease with the number of training cycles. It is still important to stop the training in time because too many training cycles will result in an "over–trained" neural network [29]. Over-training denotes the phenomenon that the network learns the noise of the training data, when this happens the networks ability to generalize is reduced an example is shown in figure 5. Another factor which increase the risk of over–training is a high number of hidden nodes. Since hidden node introduces more network variables (weights), the number of hidden nodes is a measure of the degree of freedom in the network. A rule of thumb is that these number of weights should not exceed the number of training examples (input/output values) [31]. The number of hidden nodes and training cycles must be optimized in order to avoid over–training and minimize the training time.

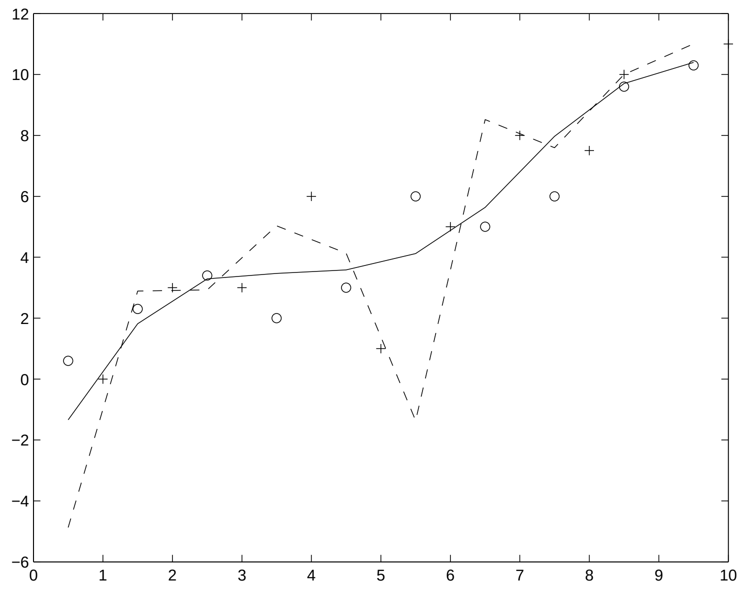


Figure 5: A picture of what happens when a neural network is over–trained. The pluses symbolize the training set and the circles the test set. The solid line represents and optimally trained network and the dashed line shows an over–trained network. The over–trained network performs worse on the test data than the optimally trained network.

3 Methods

3.1 Selection of target–template pairs

To get good results it is important to use a large and broad set of related and unrelated protein domains of high quality. This was achieved by choosing a set of proteins that according to SCOP (version 1.39) shared the same fold [3]. The version 1.39 of SCOP used here contained 12 805 domains. Since some of the structures lack coordinates for some of the residues, or have other problems, all proteins could not be used. Therefore only proteins that consisted of a continuous domain with the same sequence as in the SCOP database were included, leaving 10 306 protein domains. From these a subset of proteins representing all families with sequence identity less than 50% were selected, leaving a total of 1060 proteins.

All proteins sharing the same fold were aligned against each other using ordinary sequence alignment. They were then used as target and template for each other. Giving a total of 19 966 target–template pairs for homology modelling.

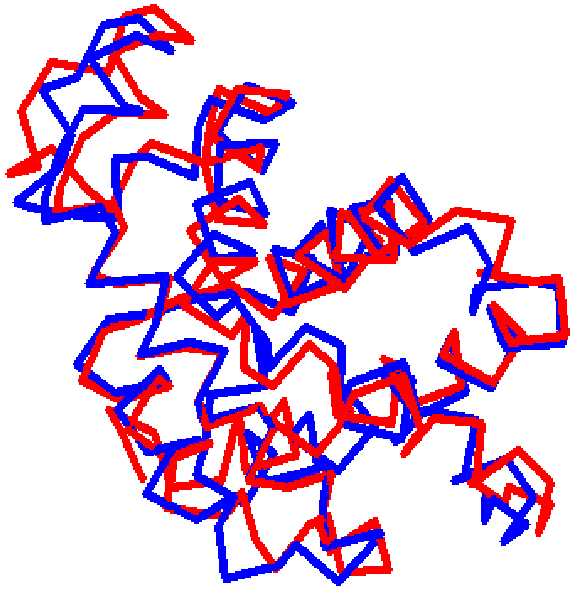


Figure 6: A protein model superimposed on its correct structure. This model had a Q–score of 30.

3.2 Homology modelling program

The homology modelling program Modeller [25] was used for building protein models. Modeller starts building the model by using the distance and dihedral angle restraints on the target sequence derived from its alignment with the template structure. Then, the spatial restraints and the CHARMM22 force field terms [37], which enforce proper stereochemistry, are combined into an objective function. The model is then generated by optimizing the objective function in Cartesian space. It takes about 3 minutes on a standard PC work station for Modeller to create a model from an alignment. An example of a protein model superimposed on its correct structure is shown in figure 6.

3.3 Training and test sets

In order to train and test performance of neural networks, a training and a test set is needed. These should be chosen carefully, it is desirable that similar models are not present both in the training and the test set. A good result on that test set could very well mean that the neural networks has memorized instead of generalized. On the other hand it is also not good to have too similar models in the training set, because this may bias the neural network to recognize only those models. This is avoided by choosing a broad set with examples of all different kinds of models to start with.

To avoid the possibility of a having similar models in the training and test set, the models were split into training and test sets, in such a way that one fold type only was present in either the training or the test set. With the restriction that the total number of models for one fold type should not be present more than 500 times, to avoid over representation of one particular fold in the training or the test set. Really large folds like immunoglobulin fold, were split into training and test based on SCOP superfamily instead of fold, i.e one superfamily from that fold were placed either in the training or the test set.

Test set consisted of 6 100 models representing 79 different folds and the training set consisted of 7 040 models from 80 different folds.

3.3.1 LiveBench–2 test set

An additional independent test set was created from LiveBench–2 data [38, 39]. LiveBench is a benchmarking of protein structure prediction servers. Every week newly solved protein structures, which does not show any significant similarity to any of the proteins previously available in the structural database, are sent to different structure prediction servers. The result from the servers are alignments on a number of template structures. The LiveBench–2 data consisted of 203 targets and a total of 13 035 target–template pairs collected under the period 2000–04–12 to 2000–12–29 from seven prediction servers, PDB-Blast[6], FFAS[7], fugue[8], mgenthreader[9], SAM–T99[10], foldfit[11] and dali[12]. Unfortunately it was only possible to generate models for 8 334 target–template pairs, since some of the alignments form the servers were too short or there was an inconsistency between the alignment sequence and the coordinate file for the template.

3.4 Parameter representation of protein models

The representation of a protein model should include as much information as possible about the structure, in order to be able to distinguish subtle differences between models. On the other hand too much information introduce noise, which of course is undesirable.

This section will describe the three different parameter sets, atom and residue contacts and accessibility surfaces, used to represent a protein model in this study.

3.4.1 Atom contacts

Different atom types are distributed nonrandomly with respect to each other in proteins because of energetic and geometric effects. More random distributions are expected in incorrect than in correct protein models [13]. To capture this information contacts between 13 different heavy atom types were counted for ten cutoffs ranging from 3.0 Å to 5.25 Å in 0.25 Å intervals, omitting the contacts with atoms in the same residue. The different atom types are explained in table 1. They were chosen, among the 167 heavy atoms of the 20 amino acids, on the basis that they should represent chemical differences, for example a backbone carbon and a carbon bound to two oxygens are chemically different.

[6]http://bioinformatics.ljcrf.edu/pdb_blast
[7]http://bioinformatics.burnham-inst.org/FFAS
[8]http://www-cryst.bioc.cam.ac.uk/~fugue/prfsearch.html
[9]http://insulin.brunel.ack.uk/psipred
[10]http://www.cse.ucsc.edu/research/compbio/HMM-apps/T99-query.html
[11]http://www.bmm.icnet.uk/servers/3dpssm
[12]http://www.ebi.ac.uk/dali

Atom type	Description
C	Backbone
N	Backbone
O	Backbone
C_α	Backbone
CH_3	Methyl group
CH/CH_2	Carbon acid group with one or two hydrogens
C(OO-)	Carbon in carbon acid group in Asp and Glu
(C)OO-	Oxygens in carbon acid group in Asp and Glu
NH	Amino group
NH_2	Amino group
=O	Double bonded oxygen, present in the acidic amino acids Asn and Gln
OH	Hydroxyl group present in Ser, Thr and Tyr
S	Sulfur present in Cys and Met

Table 1: Description of the different atom types used.

The contacts are symmetrically, thus 91 ($14 \times 13/2$) different contact types can be identified. Since the number of contacts is depended on the length of the protein, the number of contacts for each type must be normalized. This was done simply by the total number of contacts. That is each contact type were represented as a fraction of contacts.

3.4.2 Residue contacts

If the atom contacts describe short contacts or short–ranged forces, the residue contacts try to describe the proteins in a more global sense. But the border between and the physics behind the residue and atom contacts is not completely clear, except that both atom and residue contacts have information about hydrophobicity, electrostatics and hydrogen bonding. Perhaps the atom contacts has more information on hydrogen bonding and the residue contacts more on hydrophobicity.

Contacts between residues for six cutoffs, 4 Å, 5 Å, 6 Å, 8 Å, 10 Å and 12 Å were counted, with the restriction that the two residues in contact had to be more than five residues apart in the sequence. In order to avoid accumulation of contacts laying close in sequence. The distance between two residues were taken as the closest distance between the two residues all heavy atom included. This means that the backbone of two residues might be far apart, but if their side chains are close, they will still be regarded as being in contact. One could think of other ways to define a distance between two residues, such as C_α–C_α distance or the distance between the center of the two residues.

Since the residue contacts as well as the atom contacts are depending on the length of the protein, they were normalized similarly.

As the atom contacts, the contacts between the 20 amino acids are symmetrical giving a total of 210 ($21 \times 20/2$) different contact types.

3.4.3 Solvent accessibility surfaces

Hydrophopic forces is clearly one of the major forces in protein folding. Atom and residue contacts capture some information on hydrophobicity. But since it is such an important factor, solvent accessibility surfaces were also included in the parameter representation of a protein model.

The accessibility surfaces were calculated using the program Naccess[13], which is an implementation of a method developed by Lee & Richards [41]. Briefly, Naccess calculates the atomic accessible surface defined by rolling a probe of with radius 1.4 Å around a van der Waals surface. This probe is supposed to describe a water molecule. The program also sums the atomic accessible surface areas over each residue, and gives a relative accessibility of each residue calculated as the percentage of accessibility surface compared to a standard accessibility surface of that particular residue, X, in an extended ALA–X–ALA tripeptide [42]. Furthermore the residue accessibility surfaces are divided into side chain and main chain accessibility surfaces, C_α is included in the side chain in this description, which means that the glycine amino acid, which lack a heavy atom side chain actually can have a side chain accessibility surface.

In this study the relative accessibility surface for the side chains were used. It was thought that they would carry more information on hydrophicity than the main chain, because they are more probable to stretch out into the solvent, The accessibility surfaces were represented as fraction of relative accessibility <25%, 25%–50%, 50%–75% and >75% for each residue respectively. Also the fraction between total accessibility surface of non–polar and polar residues were considered.

3.4.4 Quality measure

As a measure of protein quality a modified version of the LGscore [21] was used. LGscore gives a score based on the most significant segment between a model and the correct structure. The statistical significance of a segment is calculated using a method developed by Levitt and Gerstein [34]. After a structural superposition of the two structures a comparison score is calculated.

$$S_{str} = M\left(\sum \frac{1}{1+(d_{ij}/d_0)^2} - \frac{N_{gap}}{2}\right)$$

where M is equal to 20, d_{ij} is the distance between residue i and j, d_0 is equal to 5 Å and N_{gap} is the number of gaps in the alignment. By calculating S_{str} for a set of structural alignments of unrelated proteins a distribution of S_{str} depending on the alignment length, l, can be determined. The distribution for S_{str} follows an extreme–value distribution. From this distribution a P–value dependent on S_{str} and l can be calculated. The P–value is the probability that a better score would occur by chance. LGscore is the negative log of this P–value for the most significant segment (lowest P–value).

One problem with the P–value is that the best possible P–value is depending on the length of the protein. For a very short proteins it is not possible to obtain a significant P–value even for a perfect model. To compensate for this the Q–score was introduced (Fang et al unpublished results). The Q–score was calculated from a number of models made for proteins with more than 50% identity. For each of the models the best P–value was calculated and then this value was fitted against the length of the proteins.

The Q–score is a better measure of protein quality than for instance the rmsd (root mean square diviation), since it consider segments of the model and

[13]http://wolf.bms.umist.ac.uk/naccess/

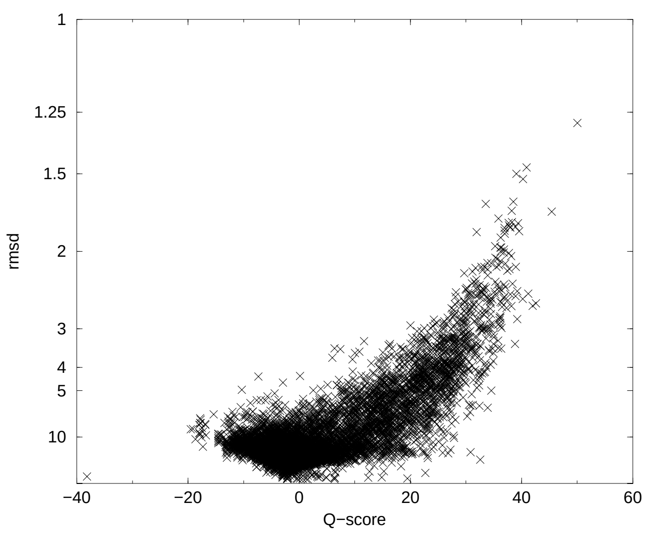


Figure 7: Q-score for all models used in this study plotted against 1/rmsd. For low rmsd (i.e rmsd < 3) the Q–score is generally high, but then there also exists models which has a high Q–score even tough their rmsd is high. These models are probably mostly correct, but with some bad region which results in a high rmsd.

not the whole model. In this way a model which is mostly correct can get a good Q–score, while the rmsd for the same model is very high. This is desirable since mostly correct models can still be of interest. A model with a Q–score above 5 is good, while a Q–score below 0 is not significant at all. In figure 7 the Q–score is compared to rmsd. For low rmsd (rmsd < 2) the Q–score is generally high, but then there also exists models which has a high Q–score even tough their rmsd is high. These models are partly correct, but with some bad region which gives a high rmsd.

3.5 Neural networks

The neural network package Netlab[14] for Matlab was used in this study. It is a two layer network, meaning that there is one hidden layer and one output layer. The response function for the hidden nodes was a *tanh* function. Empirically it has been shown that *tanh* response function converge faster than a logistic function [29] and that is the reason why it is used. The response function for the output node can be specified by the user, since the range of output is between -20 and 40 (see figure 9) a linear response function was used, instead of a logistic which restricts the output to values between 0 and 1. The training was performed using error back–propagation with a sum of square error function:

$$E = \frac{\sum_{i=1}^{N}(y_i - t_i)^2}{2}$$

where N is the number of training examples, y_i the predicted value by the neural network and t_i the correct value. Scaled conjugate gradients was used to minimize the error function.

[14]Available at http://www.ncrg.aston.ac.uk/netlab/index.html

The optimization of the number of hidden nodes and training cycles was done using the same approach as Emanuelsson *et al* [36]. The error function, E, was monitored for both the training and the test set through each cycle of the training for different number of hidden nodes. The optimal number of training cycles is when the error for the test set stops decreasing and start to increase. The optimal number of hidden nodes is chosen by training networks with different number of hidden nodes and choosing the network for which the error on the test set is minimized.

This method has been criticized, since it involves the test set in the optimization and the performance might not reflect a true generalization ability. However practical experience in bioinformatics applications has shown that the performance on a new independent test set to be as good as that found on the test used to stop training [40].

Networks were trained with hidden nodes in the range 0 to 40 for 2000 training cycles, and the optimal number of hidden nodes and training cycles was 5 and 1000 respectively. These parameters were sufficient for almost all networks trained. So in order to avoid optimizition for every single network training. The parameters were set to 5 nodes and 1000 training cycle to begin with, and during each network training the error and the correlation with the test set was monitored, and if the error and correlation showed no sign of overtraining, no further optimization was performed. This opens the possibility that some other network setting might be better, but this improvement is not likely to be that significantly different between the different networks. Which basically means that the default setting can be used to compare how good different networks performs with different input parameters. Once these are found a more solid optimization can be done, which then might improve performance slightly. This is not certain to work for every type of neural network training, but for the case here it did.

3.6 Evaluation of neural network performance

Neural network performance can be evaluated using the correlation coefficient between the correct, x_i, and predicted, y_i, values on an independent test set.

$$C = \frac{\sum_i (x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_i (x_i - \overline{x})^2 (y_i - \overline{y})^2}}$$

where $\overline{x}$ and $\overline{y}$ denotes the mean of x_i and y_i respectively. This is a value between -1 and 1, where 1 indicates a perfect prediction and perfect linear dependence between prediction and correct value, -1 indicates the same but an association in the opposite direction and 0 indicates a random prediction.

Instead of predicting the quality of model one could choose to discriminate good from bad models by choosing a cutoff value for correct prediction, all values above the cutoff are counted as correct and all below as incorrect. Then the normal correlation coefficient can not be used to calculate network performance. Instead Matthews correlation coefficient, M_C, can be used. It is defined as:

$$M_C = \frac{P_t N_t - P_f N_f}{\sqrt{(N_t + N_f)(N_t + P_f)(P_t + N_f)(P_t + P_f)}}$$

where P_t is the number of true positives, P_f the number false positives, N_t the number of true negatives and N_f the number of false negatives, these are

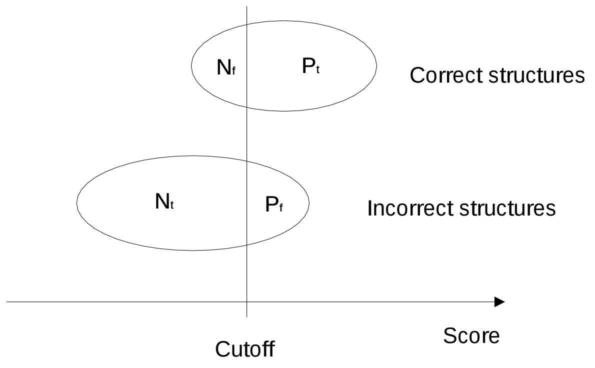


Figure 8: Illustration of the parameters P_t, P_f, N_t and N_f.

illustrated in figure 8. From these some other useful measures for evaluating neural network performance can be calculated. The specificity of a prediction that is the fraction of predictions predicted correct is defined as:

$$Specificity = \frac{P_t}{P_t + P_f}$$

and the sensitivity defined as the fraction of all possible correct structures found:

$$Sensitivity = \frac{P_t}{P_t + N_f}$$

4 Results and Discussion

To predict the quality of a protein model using neural networks, many protein models are needed in order to produce good training data for the neural network. To achieve this target and template structures for homology modelling were selected from the SCOP database (release 1.39). All structures sharing the same fold and with less than 50% sequence identity were selected as target and template for each other. They were aligned using pairwise alignment then the modelling program Modeller [25] was used to produce protein models from the alignments.

The quality of these models were assessed by comparing them to their known structure calculating the Q–score (see Methods). As shown in figure 9 the quality for these models ranged from really bad models with Q–score less than -10, to very good models with a Q–score above 5. This is good, since it is desirable that the neural network learns to recognize features of both good and bad models.

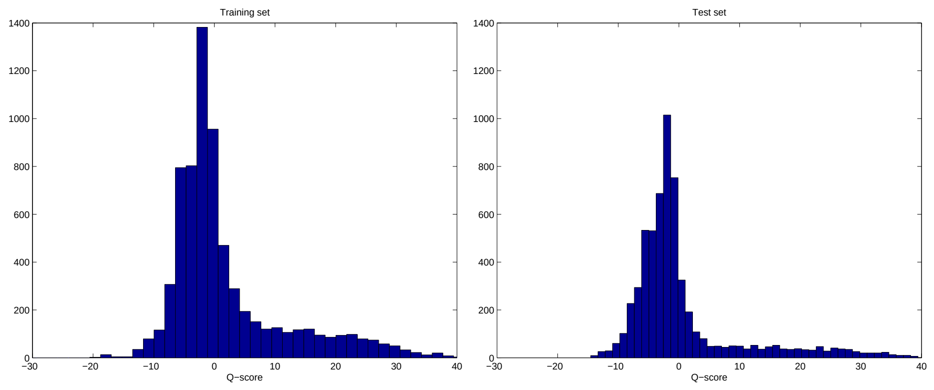


Figure 9: Histograms over the Q–score for the training (left) and the test set (right). Most models lie in the range -8 to 5, but there are some really good models with a Q–score above 10. The training set contained 1 000 more models than the test test, which is the reason for the difference in area.

4.1 Neural network training

The get some idea of which parameters and contact cutoffs that were important for describing the quality of protein models. Neural network were trained to predict protein quality for every single set of parameters, i.e one network for every atom contact cutoff, one for every residue contact cutoff and and one for every relative accessibility surfaces per residue type. The result is shown in table 2, 3 and 4.

Atom cutoff (Å)	C
3.0	0.57
3.25	0.60
3.5	0.60
3.75	0.62
4.0	0.66
4.25	0.67
4.5	0.70
4.75	0.70
5.0	0.69
5.25	0.70

Table 2: Correlation coefficient, C, between the predicted and correct value when a neural network was trained only on atom contacts with different.

It is clearly seen that for some parameters the neural network performs better. For the atom contacts best correlation, 0.70, was obtained for some cutoffs in the range 4.5–5.25 Å. The best correlation for the residue contacts was 0.65 with a 6 Å cutoff. This indicate that the atom contacts contains some what more information on the protein quality than the residue contacts. It could also be that the residue contacts which contain 210 different parameters, as compared to the 91 parameters of the atom contacts, is more noisy.

For the fraction of relative accessibility surfaces per residue, low (<25%) and high (>75%) relative accessibility surface gives the best correlations of 0.69

Residue cutoff (Å)	C
4.0	0.54
5.0	0.63
6.0	0.65
8.0	0.57
10.0	0.51
12.0	0.44

Table 3: Correlation coefficient, C, between the predicted and correct value when a neural network was trained only on residue contacts with different cutoffs.

Rel. accessibility	C
<25%	0.69
25%–50%	0.12
50%–75%	0.38
>75%	0.52

Table 4: Correlation coefficient, C, between the predicted and correct value when a neural network was trained only on the fraction of relative accessibility surface per residue.

and 0.52 respectively. It is more useful to know whether there is low or high relative accessibility, than to know that there is something in between. In other words the information on which residues that are most often buried or exposed, corresponding to low and high accessibility, is more useful than the information that some residues sometimes are buried/exposed and sometimes not. This seems natural, since the accessibility is a measures hydrophicity, the fraction of low (<25%) relative accessibility should ideally be high for non–polar residues and low for polar residues and vice versa for high accessibility. This fact could be used by the neural network to predict the quality of a model.

4.2 Additional analysis of training parameters

A further analysis of the training parameters was performed by varying every parameter individually, and study the neural network prediction. This gave a lot of information, but it was hard to draw any significant conclusions. In table 5 an example of this analysis is shown. For example an increase in the N–O contact type, which could be a measure of hydrogen bonding in the backbone, results in an increase in predicted Q–score for 3.0 Å cutoff, but a decrease in predicted Q–score for 4.5 Å cutoff. The length of a hydrogen bond is in the range 2.8 Å. This could mean that for the 3.0 Å cutoff N–O contacts measures the hydrogen bonding, whereas the N–O contacts for 4.5 Å cutoff includes other N–O contacts which do not constitute a hydrogen bond. Despite this the correlation coefficient for the 4.5 Å cutoff is much better according to table 2.

Contact with different type of carbons appears have a positive effect on the Q–score prediction in most cases for the two cutoffs. This is probably due th hydrophicity. Finally high S–S contacts seems also to have an increasing effect on the Q–score prediction, this might be due to disulfur bridges.

The residue contacts and accessibility surfaces were also analyzed with the same scheme as the atom contacts, but it was difficult to draw conclusions and are therefore omitted here.

4.3 Combining different parameter sets

In the next step neural networks were trained with various combination of the different parameter sets from section 4.1. The result is shown in table 6.

The first obvious combination was to combine the best atom and residue contacts, i.e 4.5 Å atom contact cutoff with correlation 0.70 and 6 Å residue contact cutoff with a correlation of 0.64 and train a network with these two parameter sets. The 4.5 Å atom contact cutoff was chosen instead of the longer cutoff choices with the same correlation, to restrict the atom contacts to short–ranged forces.

The training on the new parameter set improved the performance to a correlation of 0.74. If all relative accessibility surfaces also were added to the training data for the network, a correlation of 0.81 was obtained. This is clearly an improvement from the training only with atom or residue contacts.

Another combination used in the training was to use two pairs of atom and residue contacts, one with short and one with long contact cutoff. This improved the performance, for some cutoff choices, almost as much as the combination of atom and residue contacts in the first step. This indicates that the short and long contact cutoff have some non–overlapping information, which is useful for predicting the quality of a model. If these four parameter sets are combined with the accessibility surfaces a correlation of 0.83 was obtained for the best cutoff choices.

Scatter plots for some parameter sets are shown in figure 10. Ideally all points should lie on the solid line, but even if that is not the case here there is faint tendency in that direction, when more combinations of the different parameter sets is used. By comparing the upper left and lower right pictures, corresponding the least and most parameter sets, a clear difference is seen. The one with the most parameter sets is clearly better, with more points lying close the solid line and not as spread as in the upper left picture.

To get an estimation of which discriminating power the neural network trained with the best parameter sets has, i.e how good it is at discriminating good from bad models. The matthews correlation coefficient, the sensitivity and specificity were calculated for different cutoff choices on the Q–score. The result is shown in figure 11. The matthews correlation coefficient has its maximum value of 0.77 for cutoff 7.5, the sensitivity and specificity for this cutoff are 0.69 and 0.92 respectively. This means that if the cutoff is set to 7.5, the neural network founds 69% of all correct structures and that 92% of all predictions made are correct.

From table 6, 3 and 2 it is evident that the accessibility surfaces has most information on protein quality of the three parameter sets. Alone it gets a correlation of 0.78 and together with some other parameter sets it improves the correlation with about 0.05. This points to the fact that hydrophicity is an important factor, and that the introduction of that information clearly has an effect on how good the neural network predict the quality of a model.

4.5 Å cutoff

contact type	C	N	O	CA	CH3	CH/CH2	C(OO)	NH	NH2	(C)OO	O=	OH	S
C	-	-	+	0	-	-	0	+	0	-	0	0	0
N		+	-	0	-	-	+	0	+	+	0	+	+
O			-	+	+	+	0	+	0	-	-	-	0
CA				+	-	0	0	0	-	+	0	+	+
CH3					+	+	-	0	-	-	-	-	+
CH/CH2						+	-	0	-	-	-	0	+
C(OO)							0	0	+	0	0	+	0
NH								0	0	+	0	0	0
NH2									+	+	0	0	-
(C)OO										0	0	+	-
O=											0	+	0
OH												0	0
S													+

3.0 Å cutoff

contact type	C	N	O	CA	CH3	CH/CH2	C(OO)	NH	NH2	(C)OO	O=	OH	S
C	+	-	+	+	0	+	+	0	-	-	0	-	+
N		0	+	+	-	-	-	0	-	0	+	+	-
O			-	-	-	-	+	+	+	-	-	0	0
CA				-	+	+	+	-	+	-	-	+	0
CH3					+	0	-	+	-	-	-	-	-
CH/CH2						0	-	-	-	-	-	-	-
C(OO)							+	-	0	-	0	-	-
NH								+	-	+	-	-	-
NH2									0	+	+	+	-
(C)OO										-	-	0	-
O=											-	-	+
OH												-	-
S													+

Table 5: An illustration of what happens with the prediction when a atom contact parameter is increased, in the above table the neural network were trained only on atom contacts with a cutoff of 4.5 Å and in the lower table with a cutoff of 3.0 Å, '-' denotes a decrease in predicted quality, '+' denotes an increase in predicted quality and '0' denotes no change in predicted quality.

res_1	res_2	$atom_1$	$atom_2$	accessibility	C
6.0 Å	-	4.5 Å	-	-	0.74
-	-	-	-	*all*	0.73
-	-	4.5 Å	-	*all*	0.78
6.0 Å	-	-	-	*all*	0.78
6.0 Å	-	4.5 Å	-	*low/high*	0.79
6.0 Å	-	4.5 Å	-	*all*	0.80
6.0 Å	8.0 Å	3.25 Å	4.5 Å	-	0.79
6.0 Å	8.0 Å	3.25 Å	4.5 Å	*all*	0.83
6.0 Å	10.0 Å	3.25 Å	4.5 Å	-	0.78
6.0 Å	10.0 Å	3.25 Å	4.5 Å	*all*	0.83
6.0 Å	12.0 Å	3.25 Å	4.5 Å	-	0.77
6.0 Å	12.0 Å	3.25 Å	4.5 Å	*all*	0.82
6.0 Å	10.0 Å	3.0 Å	4.5 Å	-	0.76
6.0 Å	10.0 Å	3.0 Å	4.5 Å	*all*	0.82
6.0 Å	10.0 Å	3.5 Å	4.5 Å	-	0.76
6.0 Å	10.0 Å	3.5 Å	4.5 Å	*all*	0.82
6.0 Å	10.0 Å	3.75 Å	4.5 Å	-	0.76
6.0 Å	10.0 Å	3.75 Å	4.5 Å	*all*	0.82
4.0 Å	8.0 Å	3.0 Å	4.75 Å	-	0.76
4.0 Å	8.0 Å	3.0 Å	4.75 Å	*all*	0.80
4.0 Å	12.0 Å	3.0 Å	4.75 Å	-	0.73
4.0 Å	12.0 Å	3.0 Å	4.75 Å	*all*	0.79
4.0 Å	10.0 Å	3.25 Å	4.5 Å	-	0.76
4.0 Å	10.0 Å	3.25 Å	4.5 Å	*all*	0.80
4.0 Å	12.0 Å	3.25 Å	4.5 Å	-	0.75
4.0 Å	12.0 Å	3.25 Å	4.5 Å	*all*	0.82

Table 6: Correlation coefficient, C, between the predicted and correct value from neural networks trained on various combinations of residue and atom contacts and accessibility surfaces. res_1, res_2, $atom_1$, $atom_2$ is the cutoff used in the training for residue for residue and atom contacts respectively, *low* and *high* accessibility means relative accessibility per residue in the range <25% and >75% respectively, *all* accessibility means that all relative accessibility surfaces were used, i.e <25%, 25%–50%, 50%–75% and >75% and also the fraction of total accessible surface between non–polar and polar residues.

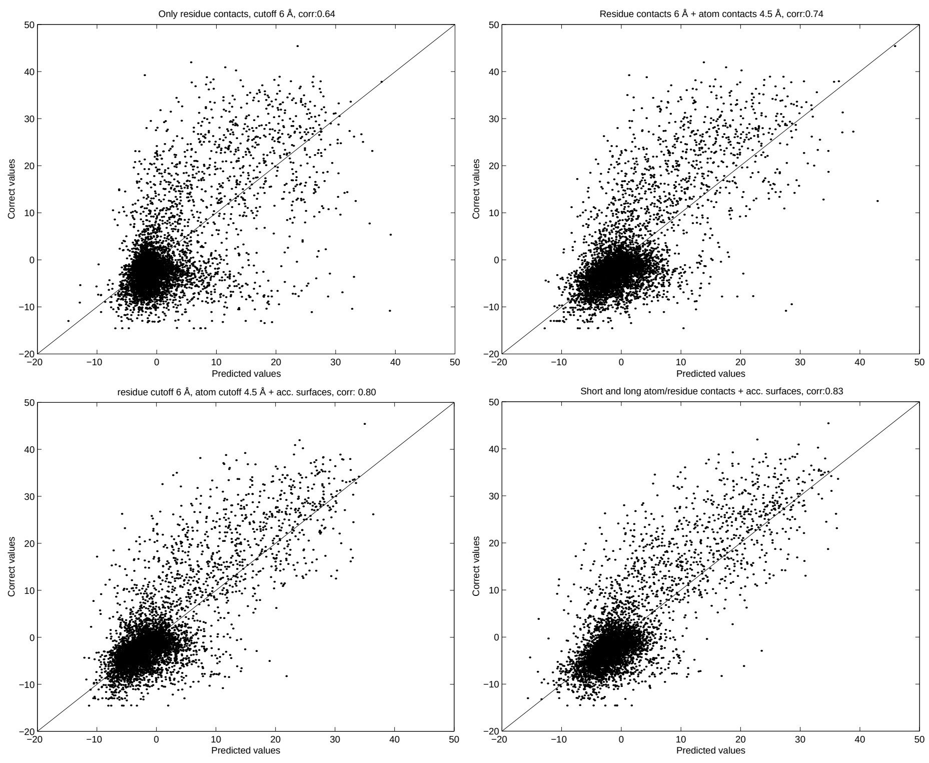


Figure 10: Scatter plots from predictions made by neural networks trained on different combination of input parameters. Above left are the predicted and correct values for a neural network trained only on the best residue contacts with 6 Å cutoff, the correlation coefficient is 0.64. Above right is the same for a network train on residue contact cutoff 6 Å and on the best atom contact, 4.5 Å cutoff. Lower left a network trained on residue contact cutoff 6 Å, atom contact cutoff 4.5 Å and accessibility surfaces. Lower right is the prediction for the neural network train with short and long atom and residue contacts together with accessibility surfaces. For the atom contacts the cutoffs 3.25 Å and 4.5 Å were used and for the residue contacts the cutoffs 6.0 Å and 10.0 Å were used. All points should ideally lie on the solid line, this is obvious not the case here. But by introducing more parameter sets in the training the performance is a clearly improved. This is seen comparing the upper pictures with the lower, corresponding to an increase in number of parameters.

4.4 Prediction of other quality measures

It is not certain that the Q–score is the most appropriate measure for the neural network to predict, some other measure might be more suitable for the neural network. To explore this, networks were trained to predict two other measures, 1/rmsd and LGscore, based on those parameter sets which gave the best results for Q–score. The results is shown in table 7. These parameter settings might not be the best for that particular measure and in a more thorough examination one of course has to retrain on all different kinds of parameters. Because of lack of time this was not done in this study.

The best correlation for 1/rmsd was 0.78 and for LGscore the best correlation was 0.83. Which means that the neural network performs worse when

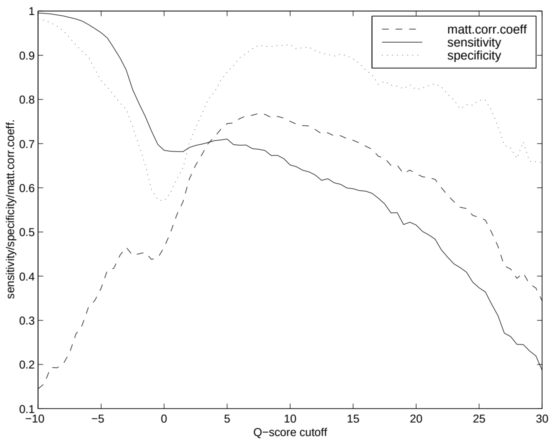


Figure 11: Sensitivity (solid line), specificity (dotted line) and matthews correlation coefficient (dashed line) as a function of cutoff for Q–score for prediction made by the neural network trained on residue contacts cutoffs 6.0 Å and 10.0 Å, atom contacts cutoffs 3.25 Å and 4.5 Å and accessibility surfaces. The matthews correlation coefficient is at maximum for cutoff 7.5, with a value of 0.77, for this cutoff the sensitivity and specificity are 0.69 and 0.92 respectively.

res_1	res_2	$atom_1$	$atom_2$	acc.surface	$C_{1/rmsd}$	$C_{LGscore}$
6.0 Å	8.0 Å	3.25 Å	4.5 Å	*all*	0.74	0.83
6.0 Å	10.0 Å	3.25 Å	4.5 Å	*all*	0.77	0.83
6.0 Å	12.0 Å	3.25 Å	4.5 Å	*all*	0.78	0.83

Table 7: Correlation coefficient, C_{rmsd} and $C_{LGscore}$, for neural network trained to predict 1/rmsd and LGscore instead of Q–score. For the combination of parameters indicated by res_1, res_2, $atom_1$, $atom_2$, acc.surface. For LGscore the performance is comparable to the Q–score, while the performance for 1/rmsd is decreased.

predicting 1/rmsd instead of Q–score, but when predicting LGscore the performance is maintained. This rises the question whether it might have been better to train everything against LGscore instead of Q–score. To investigate this the prediction for the neural network trained with 6.0 Å and 12.0 Å residue contact cutoff in table 7, was analyzed further. This was done by comparing the predictions from networks trained to predict Q–score, 1/rmsd and LGscore respectively. A cutoff for correct prediction was defined, in such a way that the number of true positives and true negatives, i.e the number of models predicted true, were equal for all three cases. To begin the cutoff for Q–score was set to 7.5 from section 4.3. The number of models predicted as correct for this case was 643. Then the cutoffs for 1/rmsd and LGscore was defined in such a way that the total number of models predicted as correct were equal to 643. In this the top 643 predictions from each method were compared. The result from this comparison is shown in table 8. From the table Q–score gives the best prediction and LGscore and 1/rmsd have about the same performance. Still this is something that can be more thoroughly investigated in the future.

Attempts were also made to train two separate neural networks with different parameter set, and then use the the output from these two networks as input to

Pred value	cutoff	matthews corr	specificity	sensitivity
Q–score	7.5 Å	0.77	0.92	0.69
LGscore	4.25 Å	0.70	0.81	0.67
rmsd	7.8 Å	0.71	0.80	0.69

Table 8: Matthews correlation coefficient, specificity and sensitivity for three neural networks trained to predict Q–score, LGscore and rmsd respectively. The cutoff was chosen in such a way that the total number of models with a score above the cutoff was equal in all three cases.

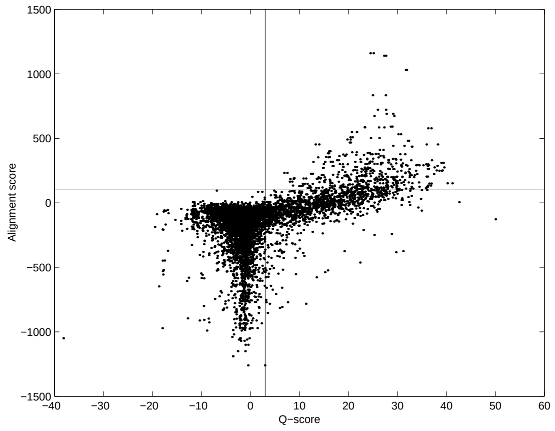


Figure 12: Q–score plotted against the the corresponding alignment score between target sequence and template structure used in homology modelling. A high alignment score usually results in a high quality model, every model with an alignment score above 100 (horisontal line) always gives a Q–score higher than 3 (vertical line).

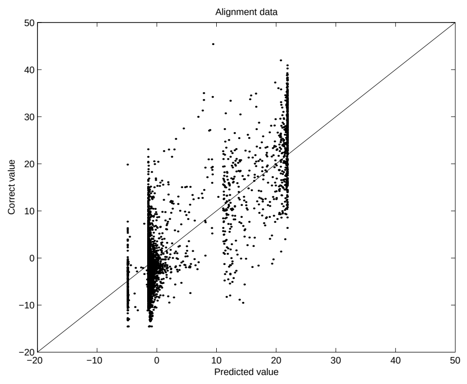
a third network. This prodcedure did not improve performance, and thus was not explored further.

4.5 Training on alignment data

It is well known that a good alignment between target sequence and template structure in general produce a good model. In figure 12 it is clearly seen that a good alignment results in a high quality model. If the alignment score is low the model could still be of high quality, which means that the opposite, i.e that a bad alignment (based on the score), not necessarily implies a bad model.

As a comparison to the other training parameters, a neural network was trained to predict the quality of the model based on the score from the alignment. Long sequences have more possibilities for aligning the sequences, which increases the probability of getting a higher alignment score, this means that alignment score is dependent of the length of the two aligned sequences. To compensate for this the neural network was trained on the alignment score together with the lengths of the two aligned sequences.

The correlation between the predicted and correct quality was 0.85 for the test set, which is better than the best parameter set. But still this neural network does not give better predictions, which is shown in the scatter plots



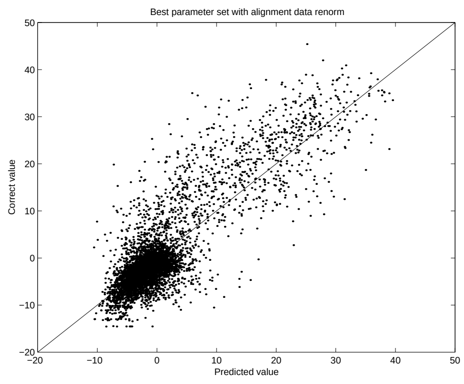


Figure 13: To the left a scatter plot from values predicted by a neural network trained on only the alignment score and length of the two aligned sequences. To the right a scatter plot from values predicted by a neural network trained on the best combination of training parameters together with the alignment score.

in figure 13. It seems as if the network either gives a low or a high Q–score prediction, this is reasonable since the only information it has is essentially the alignment score. However this rises the question, if the correlation coefficient actually is the best measure of neural network performance?

An alternative measure would be to calculate the correlation coefficient for some specific region, which are of interest. If one has to choose it is more interesting to distinguish between high quality models, since low quality models probably are quite easy to filter out anyway. The correlation coefficient for all models with a Q–score higher than 5, was taken as a new measure. This new measure was 0.65 for the neural network prediction trained only on alignment data and 0.69 for the best other combination of parameters. But even if this could be used as a new measure, the ordinary correlation coefficient together with a scatter plot work just as fine.

If a neural network is trained with the best parameters, that is 6.0 Å and 10.0 Å cutoff on the residue contacts, 3.25 Å and 4.5 Å cutoff on the atom contacts, all accessibility surfaces and the alignment score a correlation coefficient of 0.86 is obtained. The scatter plot for that prediction is shown in figure 13. Even though the correlation coefficient is only slightly better than for the training only on alignment data, the scatter plot reveals that this prediction is significantly better.

5 Conclusions and future steps

A method for predicting the quality of a protein model using neural networks has been presented. Best performance was obtained for a neural network trained with short and long atom and residue contacts together with solvent accessibility surfaces. This performance was comparable to the performance obtained for a neural network trained only on alignment data (score and the length of the two aligned sequences). The alignment is regarded to be the single most important factor determining the accuracy of a 3D model and a high alignment score usually gives a model of high quality. The method seems promising and worth

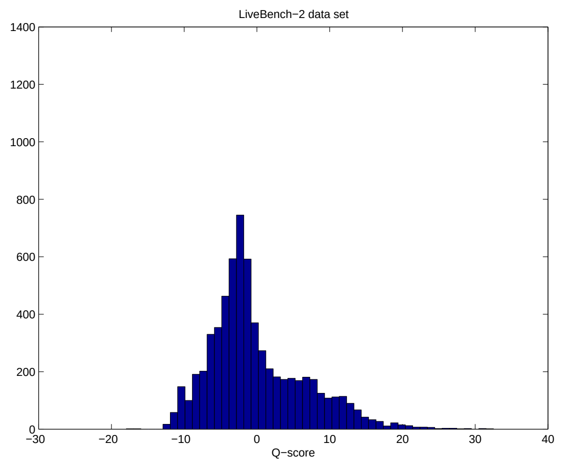


Figure 14: Histogram over the quality for the LiveBench–2 models used as an additional test set. The quality of the LiveBench models were general

developing further.

5.1 Neural network performance on an additional test set

To evaluate the method further, models generated from LiveBench–2 (LB–2) data [39], were used as a additional independent test set. The quality distribution for the LB–2 models were a bit different for the models in the test set earlier in this study compare figure 9 and 14. The LB–2 set had more models with a Q–score between 0 and 15 than the test set, on the other the hand the LB–2 had fewer really good models with a Q–score above 20.

Unfortunately performance was not sustained, the correlation between predicted and correct value was 0.50, which is rather modest. This could be due to some feature of the LB–2 models, or some artifact in the training and test set. But still the prediction is not completely random, and it might contain some information which could be useful.

This problem needs to be looked in to, and it would be the next natural step to take.

It could be something wrong in how the test and training set are split up. This could be done in another way, for instance using cross–validation, were the data is split into five parts and in each neural network training, one part is used as a test set and the remaining four as training set. This is then repeated so that all five parts are used both for testing and training, but not both in the same network training. This might remove some undesirable features in the current test and training set.

Other possibilities to improve performance could be to include more parameters, with the risk of introducing more noise. For instance some parameter with information on secondary structure might increase the performance. One could also think of the possibility of having more atom contact types than the 13 used here. Or to increase the number of atom contacts types and remove all residue contacts, and describe the whole protein on the atom level.

Even if the method described here does not perform as good on the LiveBench–

2 data as on the test set, it could still be used to improve the specificity of fold–recognition methods. Pcons [44] is a fold recognition consensus predictor developed by Elofsson and co–workers. It is, like the method presented in this study, neural network based, and it tries to select the best prediction from several predictions made by different fold recognition servers, by assessing the quality of the models generated by the servers. Right now Pcons makes 10% more correct predictions than the best single server. This might be improved further by including the method developed in this study.

Acknowledgements

I would like to thank my supervisor Arne Elofsson at Stockholm Bioinformatics Center, for his help and valuable ideas throughout the project. I would also like to thank all students and all other people at Stockholm Bioinformatics Center for their help and support during this project.

References

[1] Al-Lazikani B, Jung J, Xiang Z & Honig B. (2001) Protein structure prediction. *Curr. Opin. Chem. Biol.* 5, 51–56.

[2] Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN & Bourne PE. (2000). The Protein Data Bank. *Nucleic Acids Res.* 28, 235–242.

[3] Murzin AG, Brenner SE, Hubbard T & Chothia C. (1995). SCOP: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.* 247,536–540.

[4] Holm L & Sander C. (1999). Protein folds and families: sequence and structure alignments. *Nuleic Acids Res.* 27, 244–247.

[5] Orengo CA, Pearl FMG, Bray JE, Todd AE, Martin AC, Conte L. Lo & Thornton JM. (1999). The CATH database provides insights into protein structure/function relationship. *Nuleic Acids Res.* 27, 275–279.

[6] Pearson WR. (1990). Rapid and Sensitive Sequence Comparison with FASTP and FASTA. *Methods Enzymol.*, 183, 63–98.

[7] Altschul SF, Gish W, Miller W, Myers EW & Lipman DJ. (1990). Basic Local Alignment Search Tool. *J. Mol. Biol.* 215, 430–410.

[8] Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W & Lipman DJ. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucl. Acid Res.* 25, 3389–3402.

[9] Jones DT, Taylor WR & Thornton JM (1992) A new approach to protein fold recognition. *Nature*, 358, 86–89.

[10] Marti-Renom MA, Stuart AC, Fiser A, Sanchez R, Melo F & Sali A. (2000). Comparative protein structure modeling of genes and genomes. *Annu. Rev. Biophys. Biomol. Struct.* 29, 291–325.

[11] Murzin AG. (1999). Structure classification-based assessment of CASP3 predictions for the fold recognition targets. *Proteins*, 37(S3), 88–103.

[12] (1999). CAFASP-1: Critical assessment of fully automated structure prediction methods. *Proteins*, 37(S3), 209–217.

[13] Colovos C & Yeates TO. (1993). Verification of protein structures: Patterns of nonbanded atomic interactions. *Protein Science*, 2, 1511–1519.

[14] Lazaridis T & Karplus M. (1998). Discrimination of the native from misfolded protein models with and energy function including implicit solvation. *J. Mol. Biol.* 228, 477–487.

[15] Anfinsen CD. (1973). Principles that govern the folding of protein chains. *Science*, 181, 233–230.

[16] Park B & Levitt M. (1996). Energy functions that discriminate X–ray and near–naive folds from well–constructed decoys. *J. Mol. Biol.*, 258, 367–392.

[17] Park B, Huang ES & Levitt M. (1997). Factors affecting the ability of energy functions to discriminate correct from incorrect folds. *J. Mol. Biol.*, 266, 831–846.

[18] Sippl M. (1995). Knowledge based potentials for proteins. *Curr. Opin. Struct. Biol.* 5, 229–235.

[19] Samudrala R & Moult J. (1998). An all–atom distance–dependent conditional probability discriminatory function for protein structure prediction. *J. Mol. Biol.* 275, 895–916.

[20] Sippl M. (1990). Calculations of confromational ensembles from potentials of mean force. An approach to the knowledge based prediction of local structures in globular proteins. *J. Mol. Biol.* 213, 859–883.

[21] Cristóbal S, Zemla A, Fischer D, Rychlewski L, & Elofsson A. (2001). How can the accuracy of a protein model be measured?. *submitted.*

[22] Dill KA. (1990) Dominant forces in protein folding *Biochemistry.* 29, 7133–7155.

[23] Baker EN & Hubbard RE. (1984). Hydrogen bonding in globular proteins. *Prog. Biophys. Mol. Biol.* 44,97–179.

[24] Stickle DF, Presta LG, Dill KA & Rose GD. (1992). Hydrogen bonding in globular proteins. *J. Mol. Biol.* 226, 1143–1159.

[25] Sali A & Blundell TL. (1993). Comparative protein modelling by satisfaction of spatial restraints. *J. Mol. Biol.* 234, 779–815.

[26] Petersen TN, Lundegaard C, Nielsen M, Bohr H, Bohr J, Brunak S, Gippert GP & Lund O. (2000). Prediction of protein secondary structure at 80% accuracy. *Proteins*, 41, 17–20.

[27] Park J, Karplus K, Barrett C, Hughey R, Haussler D, Hubbard T & Chothia C. (1998). Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *J. Mol. Biol.*, 284, 1201–1210.

[28] Lundström J. (2001). Pcons: A consensus approach to protein fold recognition. Master thesis. Stockholm University.

[29] Bishop CM. (1995). Neural networks for pattern recognition. Oxford University Press.

[30] Møller MF. (1993). A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning. *Neural Networks*, 6, 525–533.

[31] Lawrence S, Giles CL & Tsoi AC. (1997). Lessons in Neural Network Training: Overfitting may be Harder than Expected. *AAAI–97*, 540–545.

[32] Lo Conte, L, Ailey B, Hubbard TJ, Brenner SE, Murzin AG & Chothia C. (2000) SCOP: a structural classification of proteins database. *Nucl. Acid Res.*, 28, 257–259.

[33] Bhat TN, Bourne P, Feng Z, Gilliland G, Jain S, Ravichandran V, Schneider B, Schneider K, Thanki N, Weissig H, Westbrook J & Berman HM. (2001) The PDB data uniformity project. *Nucleic Acids Res.* 29, 214–218.

[34] Levitt M & Gerstein M. (1998). A unified statistical framework for sequence comparison and structure comparison. *Proc. Natl. Acad. Sci. USA*, 95, 5913–5920.

[35] Nabney I & Bishop C. (1991). Netlab. Free Software Foundation Inc.

[36] Emanuelsson O, Nielsen H & von Heijne G. (1999). ChloroP, a neural network–based method for predicting chloroplast transit peptides and their cleavage sites. *Protein Science*, 8, 978–984.

[37] MacKerell AD, Bashford D Jr, Bellott M, Dunbrack RL Jr, Evanseck JD, Field MJ, Fischer S, Gao J, Guo H, Ha S, Joseph-McCarthy D, Kuchnir L, Kuczera K, Lau TK, Mattos C, Michnick S, Ngo T, Nguyen DT, Prodhom B, Reiher WE, Roux B, Schlenkrich M, Smith JC, Stote R, Straub J, Watanabe M, Wiórkiewicz-Kuczera J, Yin D & Karplus M. (1998). All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J. Phys. Chem. B*, 102, 3586–3816.

[38] Bujnicki JM, Elofsson A, Fischer D & Rychlewski L. (2001). LiveBench-1: continuous benchmarking of protein structure prediction servers. *Protein Science*, 10, 352–361.

[39] Bujnicki JM, Elofsson A, Fischer D & Rychlewski L. (2001). LiveBench-2: large-scale automated evaluation of protein structure prediction servers. *submitted.*

[40] Brunak S, Engelbrecht J & Knudsen S. (1991). Prediction of human mRNA donor and acceptor sites from the DNA sequence. *J. Mol. Biol.*, 220, 49–65.

[41] Lee B & Richards FM. (1971). The interpretation of protein structures: estimation of static accessibility. *J. Mol. Biol.*, 220, 507–530.

[42] Hubbard SJ, Campbell SF & Thornton JM. (1991). Molecular recognition. Conformational analysis of limited proteolytic sites and serine proteinase protein inhibitors. *J. Mol. Biol.*, 220, 507–530.

[43] Sippl M. (1993). Recognition of errors in three–dimensional structures of proteins. *Proteins*, 17, 355–362.

[44] Lundström J, Rychlewski L, Bujnicki JM & Elofsson A. (2000). Pcons: A neural network based consensus predictor that improves fold recognition. *submitted.*