Pcons: A consensus approach to protein fold recognition.

Jesper Lundström *

January 2, 2001

Supervisor: Arne Elofsson Stockholm Bioinformatics Center, Stockholm University

Examiner: Leif Bülow, Pure and Applied Biochemistry, LTH, Lund University.

*Stockholm Bioinformatics Center, Stockholm University, SE-106 91 Stockholm, Sweden E-mail:jesper@sbc.su.se

Abstract

As the genome projects proceed, we are presented with an exponentially increasing number of protein sequences but with only a very limited knowledge of their structure and function. During the last years many different fold recognition methods have been developed. Some of them has been put up on the would wide web open to use by the public. Benchmarking experiments [MHFP99, FBB+99, BEFR00] have been performed to investigate which method is the best for predicting structure from sequence information. From these studies a common conclusion can be made: for different sequences, different methods produce the best model.

We have developed, Pcons, a neural network based consensus protein structure predictor. Instead of utilizing a single prediction method, Pcons distinguishes predictions made by 6 different fold recognition web-servers. By performing structural comparisons between the models and analyzing the score of a particular model, Pcons is trained to predict the quality for each of the top 10 predictions from each server. It makes 10% more correct predictions than the best single method. A web-server has been implemented using this approach as a part of the meta-server at http://bioinfo.pl/meta/

Contents

1 Introduction **4**

2 Background and Theory **5**
2.1 Protein Structure Prediction 5
2.1.1 Sequence alignment 6
2.1.2 Threading techniques 6
2.1.3 Building and evaluating models 7
2.2 Benchmarking of Structure prediction methods 7
2.3 Structure prediction servers used by Pcons 8
2.3.1 PDB-BLAST 8
2.3.2 GenTHREADER 9
2.3.3 Sam-T98 9
2.3.4 FFAS 9
2.3.5 Inbgu 9
2.3.6 3D-PSSM 10
2.4 Structure comparisons 10
2.4.1 LGscore 10
2.4.2 LGscore2 11
2.5 Supervised learning 11
2.5.1 Artificial Neural Network 13
2.5.2 Support vector machines 13

3 Methods **14**
3.1 Pcons 14
3.2 The dual use of LGscore and LGscore2 14
3.3 Similarity between predictions 15
3.4 The different Pcons versions 15
3.5 Neural networks 18
3.5.1 Regression 19
3.6 Test and training data 19
3.6.1 The LiveBench-1 set. 19
3.6.2 Additional test sets 19
3.7 Evaluation of performance of fold recognition methods 19
3.7.1 Correlation coefficient 19
3.7.2 Matthews correlation coefficient, specificity and sensitivity 20

4 Results **20**
4.1 Performance of consensus method on the LiveBench-1 data set. 21
4.2 Analysis of Neural network response 23
4.3 Analysis of origin of models selected by the consensus method. 24
4.4 The performance of Pcons is sustained on additional test sets. 25
4.5 Importance of using regression 26
4.6 Time requirements for running Pcons. 26

5 Conclusions **27**

1 Introduction

As the genome projects proceed, we are presented with an exponentially increasing number of protein sequences. The human genome alone includes 20 000 to 50 000 genes coding for over a million protein sequences [Ven00].

Of this large number of protein sequences only a small fraction is structurally and functionally known. Protein structure and function can sometimes be determined experimentally, but it is a very time consuming process. Protein Data Bank [BWF+00] is a database collecting the proteins structurally solved through NMR or X-ray crystallography. It holds just over 12,500 protein structures (26-12-2000). Annually about 3,500 protein structures is deposited in protein data bank. A complementary approach receiving information about function and structure of an unknown protein sequence is relating it to proteins with known properties. By determining relationships between a sequence and a known protein, we can make predictions of function, structure and evolutionary features. Relationships between proteins span a broad range, from the case of almost identical sequences to apparently unrelated sequences sharing only rough 3d-structure. This poses different challenges to the detection algorithms used – a method excellent at finding sequence similarity might not perform very well in the case of only structural relationship, or vice versa.

In protein fold recognition the aim is to find the structure (or fold) an unknown sequence will adopt. To this aim a database with structurally known proteins, i.e. Protein Data Bank [BWF+00], is searched collecting one or several homologous proteins. The homologs are aligned to the unknown sequence, finally one (or several) three dimensional models are built. The classical method to detect related proteins is through sequence comparison algorithms termed sequence alignment. Sequence alignment can be single sequence based, such as Smith-Waterman and Needleman-Wunch [SW81, NW70] or multiple sequence based [GME87, AMS+97] Since many proteins share the same fold without having a clear sequence similarity other approaches to find homologs have been developed, these include a variety of threading techniques [JTT92]. Protein threading involves evaluating the fit of an unknown sequence in a known fold. Many of the available methods are using sequence alignment modified by adding structural information, for instance using special gap-penalties in loop regions [SS98], evaluating the model through energy function evaluation [Jon99a] or comparing predicted secondary of the sequence with the template protein [KMS00, Fis00]. Today, several methods for protein fold recognition is available on the web as automatic prediction servers.

Lately some studies have in addition to studying the ability of different methods to recognize the correct fold also studied the quality of the alignment [DLAS00, BEFR00, Elo00]. The results from the two later of these studies provide additional information about the performance of different methods. From these two studies it is clear that for different targets the best predictions are made by different methods. Based on these observations the current work examined if it is possible to make better predictions by utilizing several different fold recognition methods and then selecting the the best prediction from any method for each target.

Several groups have used a simple form of consensus predictions, where multiple models are built using a few methods and then for each target the highest scoring model is chosen [Fis00, KMS00]. Two of the fold recognition methods used here are based on this approach. Inbgu performs five alignments using different combination of sequence and profile data, while 3D-PSSM aligns profiles different ways.

The approach used here differs significantly from the earlier fold recognition methods. Firstly, Pcons is based on predictions from a set of publicly available web servers and not from internal predictions. As a result it is easy to update and improve the consensus predictor as new, or improved, servers become available. Secondly, information from structural comparisons between all predicted models is used as a criterion for selecting the best model. If one particular fold frequently is predicted by several servers Pcons will give a higher score to all these models. Thirdly the consensus predictor differs in the way that it selects a prediction from a certain method. A set of artificial neural networks is used to select the best models. The NNs use information about the score from a particular method and the fraction of other similar models. Finally Pcons is trained to predict the quality of a model and not just if a correct fold is recognized or not. This may be advantageous since it is quite difficult to build a good model even when a good homolog is found.

A web-server has been implemented using this approach as a part of the meta-server at http://bioinfo.pl/meta/.

2 Background and Theory

A protein consists of one or several peptide chains. A polypeptide is of variable length and built from 20 distinct amino acids bound by peptide linkages. The structure of a protein is tightly bound to it's function; it has therefore been important to understand the native structure and the folding process.

It is common to divide protein structure into four levels. The primary structure is simply the amino acid sequence. The secondary structure is the spatial arrangement of amino acid residues located near each other on the sequence. The tertiary structure is the name of the spatial arrangement of amino acids located further from each other in the sequence. Finally quaternary structure refers the spatial arrangement of subunits and the nature of their contacts.

Many of the proteins with determined structure have structure similarities. This has been utilized to work out ways to predict structures for unknown protein sequences, as described below.

2.1 Protein Structure Prediction

There are a number of different approaches to protein structure prediction. It would be very nice if it were possible to simulate all the atom to atom interactions in a protein and between the protein and its surrounding, to allow a simulation of the folding process. However, this is not practically possible, today at least. Future improvements in computer technology together with better understanding of protein folding dynamics may open this possibility. At the moment the most successful way to predict protein structure is to rely on information about known protein structures. Sequence alignment and threading are two different approaches to find the structure of a query sequence, they both rely on searching a database of known protein structures [Ste96]. Sequence alignment takes the sequence of the query and compares it to a sequence database (from proteins with known structure). Threading is evaluating the fit between the query sequence and folds in a fold library. There are prediction methods using both sequence alignment and threading techniques [KMS00, Jon99a, Fis00].

This report is presenting Pcons, which is a protein fold recognition method. Thus it is predicting of the tertiary structure of a protein. There are also efforts made to determine the

secondary structure of a protein fragments. The secondary structure prediction is often more reliable, and is sometimes used when predicting the tertiary structure.

2.1.1 Sequence alignment

Relationships between two proteins can be detected by comparing two sequences using sequence alignment techniques. All alignment algorithms rely on some kind of scheme to score the equivalence of two amino acids. The simplest form of alignment is to use an identity scoring system. In this case a identical amino acid scores one and all others score zero. However, the sequence is subject to evolution, amino acids can be substituted, inserted or deleted. These substitutions follows a pattern, for example Leucine and Isoleucine are often interchanged. Therefore more sophisticated substitution matrices for scoring are developed, where amino acids of common character score higher than amino acids with different character [Bar96]. Usually this involves a 20*20 matrix where identical amino acids score highest.

Sequences are not only mutated through substitution, insertions and deletions also occur, therefore dynamic programming is applied to the problem to allow for insertions and deletions [Bar96]. Some kind of gap penalty is given when "skipping" amino acids of one sequence.

Multiple alignment is the process of aligning more than two sequences. Multiple alignment can be useful because it can be useful to find more remotely related protein sequences, i.e. with less than 30% identity. With multiple alignment it may for example be possible detect sequence A's relatedness to sequence C through both showing a good alignment to sequence B.

From a multiple alignment is also possible to create a protein profile (also called position specific substitution matrix). The first step in creating a profile for a given amino acid sequence is to search a general database of protein sequences for homologs. A general database does include structurally unknown protein sequences as well as structurally known ones. After collecting related sequences a substitution matrix can be constructed from these sequences. The result is a L * 20 matrix, L being the query protein length, with the probability to find an amino acid in each position. This approach is more powerful than using a normal (20*20) scoring matrix since it holds information on how conserved one particular residue is.

Different approaches to the generation of the protein profile will yield different results. The main difference is on weighing the impact each sequence should make on the final profile. For instance consider 50 homologous sequences, of which 45 have more than 95% sequence similarity. Equal weighting will result in a profile very similar to 45 of the sequences, but potentially not similar to the other 5 sequences. As a result a protein more similar to one of the 5 odd sequences will receive a worse alignment score and may not be recognized as a related sequence.

2.1.2 Threading techniques

There are proteins sharing the same fold and still having a very low sequence similarity (less than 15% identity). These related proteins can be difficult to find with sequence alignment searches. Protein threading is an alternative approach to sequence alignment, that tries to overcome this problem. The amino acid sequence of the query is threaded into the structure of a template protein. Then the structure-sequence fit is evaluated through different techniques, for example inter-residue potentials or other statistical parameters [JTT92, RSS97].

There are three categories of threading approaches, 1D-3D profiles, pairwise distance methods and prediction based methods. 1D-3D profile [BLE91] are similar to profile sequence alignment, see Section 2.1.1. The profiles are however received a different way, through the 3D structure of the template. The profile is generated in two steps. First the environment for each residue is described through a number of variables, i.e. polar versus non-polar environment and secondary structure environment. The second step in generating the profile is to use chemical information for each of the 20 amino acids to work out the probability of finding an amino acid in a certain position. The query sequence can then be aligned to the profiles in the database using dynamic programming methods, such as normal sequence alignment.

Pairwise methods [JTT92] evaluate pairwise interactions between the amino acids in the structure. Correct alignment between the fold and the query sequence is however a NP-complete problem, Which means that the computational time increases exponentially with number of residues. A variety of heuristic algorithms have been developed to reach "approximate" solutions.

Prediction based methods [RSS97, FE96] are using prediction of secondary structure for the query sequence. The predicted secondary structure of the query is matched against the assigned secondary structure of the template protein. Prediction based methods are always combined with some other method of searching for related proteins.

Some prediction servers [Fis00, Jon99a, KMS00] used in this study use threading techniques in combination with traditional sequence alignment.

2.1.3 Building and evaluating models

The web servers make a model by giving the unknown sequence the same coordinates as the aligned template. Together with each model an alignment score is returned as a measure of the model quality. Some methods use additional evaluation criterias, for example energy functions and secondary structure fit.

2.2 Benchmarking of Structure prediction methods

There have been a number of benchmarking experiments aiming to distinguish between the performance of different methods for protein structure prediction. A benchmarking experiment is made on a number of targets (protein sequences with known structure). The structure is predicted for these targets without consideration of their true structure. The true structure of the target can be compared with the predicted structures, and thus the quality of the prediction can be measured.

The Critical Assessment of Techniques for Protein Structure Prediction (CASP) [MHFP99] is a "competition" held every second year since 1994, where the participating groups get a number of target sequences to predict the structure of. The structure of these queries are determined but not released. In CASP human intervention is allowed to generate predictions. For the biologist it may be more interesting to find out the performance of the variety of web-servers available for protein fold prediction. To this purpose Critical Assessment of Fully Automated Structure Prediction Methods (CAFASP) [FBB+99] started in 1999. Both CASP and CAFASP have been important to get an apprehension of what the current methods of protein structure prediction can deliver, as well as giving researchers hints about which methods work.

Method	Cutoff	Query	Template database	Evaluation
PDB-BLAST	< 0.1	Profile	Sequence	Align. score
FFAS	> 8	Profile	Profile	Align. score
3D-PSSM	< 0.37	Profile&Sec.str.	Profile &Sec-str.	Best of 3 align. scores
GenTHREADER	> 0.7	Sequence	Profile	Thread. &Align. score
Sam-T98	> 20	HMM based profile	Sequence	HMM Probabilty
Inbgu	> 12	Seq.& Profile	Seq.& Profile	5 align. scores & sec.str.

Table 1. Description of the different servers that are the basis for the neural network used in this study and the proposed cutoff used.

One downfall to the CAFASP is that it only involves a limited number of unknown sequences each year; CAFASP-2 involved 26 sequences. To rectify this problem LiveBench [BEFR00] was initiated as a complement to CAFASP. In livebench newly solved structures are continually collected and their sequence is submitted to the participating web-servers. The structures are publicly available at the time of the predictions. In the first round of LiveBench the conclusion was made that each of these servers outperformed all other servers for some queries [BEFR00]. We developed Pcons with the aim to use predictions from all servers participating in livebench and with a higher accuracy predict a protein structure.

2.3 Structure prediction servers used by Pcons

This section will briefly describe each of the six servers utilized by Pcons.

The general approach of these prediction methods is to search a database of known protein structures to find related proteins (templates). A model of the structure can then be built for the query sequence by assigning it to the same structure as the template protein. A score is given to the model, generally this score is based on how well the query aligned with the template. Three of the methods (GenTHREADER, 3D-PSSM and Inbgu) involve a measure of the sequence-structure fit weighted together with the sequence alignment score to give a output score. The methods used by the prediction servers can also be found in Table 1.

2.3.1 PDB-BLAST

PDB-BLAST (http://bioinformatics.ljcrf.edu/pdb_blast/) basically uses the same algorithm as PSI-BLAST [AMS+97] with some changed settings [RJWG00]. PSI-BLAST uses a non-redundant database while PDB-BLAST searches protein data bank [BWF+00] for known protein structures. PDB-BLAST is slower than PSI-BLAST however it seems to perform slightly better [RJWG00].

The first step in the PDB-BLAST method involves creating a profile for the query sequence [AMS+97]. The profile is made through searching a database five times using BLAST and creating a multiple alignment. Sequences which are too alike are discarded, after which a profile is created from the remaining sequences.

The profile is used to search PDB for related proteins. Finally models are built for the best alignments. The alignment score between the query profile and the PDB protein is returned as the only indication of the model accuracy.

2.3.2 GenTHREADER

The GenTHREADER (http://www.psipred.net) algorithm can be divided into three distinct steps. First the query sequence is aligned to a database of protein profiles to find related proteins [Jon99a]. When a possible match is found the query sequence is threaded onto the template protein, the solvation energy and pairwise energy of mean force are evaluated. These energy scores are together with alignment score, query length, alignment length and template length fed into a neural network. The neural network is trained to predict if template and query proteins are related or not.

2.3.3 Sam-T98

SAM-T98 (http://www.cse.ucsc.edu/research/compbio/HMM-apps/T98-query.html) uses Hidden Markov models (HMMs) [KBM+94] to align a query sequence to a template protein [PKB+98]. To initialize the HMM, a multiple alignment of sequences homologous to the query is performed.

2.3.4 FFAS

In congruence with GenTHREADER, FFAS (http://bioinformatics.burnham-inst.org/FFAS/) involves a database with protein profiles [RJWG00]. However, the FFAS algorithm also produces a profile of the query sequence, generated with the PSI-BLAST algorithm [AMS+97]. The query profile is used to search the profile database, where a profile to profile search is performed.

2.3.5 Inbgu

Inbgu (http://www.cs.bgu.ac.il/~bioinbgu/form.html) is a consensus method running five alignment procedures and weighing the results together [Fis00]. These are the alignment procedures:

- GONP - A simple sequence to sequence alignment.
- GONPM - Uses a multiple alignment of sequences to the query and aligns to a fold library.
- PRFSEQ - Query profile is aligned to sequence fold library.
- SEQPPRF - Query sequence is aligned to profile fold library.
- SEQPMPRF - Uses a multiple alignment of sequences to the query and aligns it to a profile fold library.

Threading techniques developed by Fisher and Eisenberg [FE96] are used to calculate the sequence-structure fit for these alignments. The structure evaluation is done through comparing predicted secondary structure for the query with observed secondary structure in the template. A final score on each model is calculated from the above five alignment scores together with a predicted secondary structure match [Fis00].

2.3.6 3D-PSSM

3D-PSSM (three-dimensional position specific scoring matrix) involves a specialized profile fold library which also includes secondary structure and solvent accessibility information incorporated in each profile [KMS00]. Another trick used in the fold library of 3D-PSSM is that for each entry three profiles exists. The first is created analogous to the PSI-blast profile (A). The other two profiles (B & C) are based on a protein sequence in the same super-family [MBHC95] as the database entry. Proteins in the same super-family share the fold, eventhough, sequence identity too low to be detectable. Thus, in the fold library three profiles exist A, B and C.

The secondary structure for the query sequence is predicted using PSI-pred [Jon99b] and a query-profile is created. The fold library is then scanned three times. The first search is performed by aligning the query profile with all A-profiles. The second time around each A-profile (in the database) is aligned to the query sequence (bi-directional scoring). Finally the third search is performed by aligning the query to the 3D-profile received from weighing A,B and C together [KMS00]. The highest scoring is taken as a final result. 3D-PSSM can be accessed at the address http://www.bmm.icnet.uk/servers/3dpssm/.

2.4 Structure comparisons

Different approaches of measuring the quality of a model exists [CZF+00, LKDS99]. The traditional way of measuring structural similarity is to use a global measure, often measuring the root mean square distance (RMSD). It measures the offset at each residue and returns an average for the whole protein. This procedure does not work well for protein fold recognition, because some models only model part of the protein correctly. These half correct models can still be interesting to the biologist. As an alternative methods that find the "most significant" segment in common between the model and the crystal structure have been developed, for example LGscore and LGscore2 [BEFR00, CZF+00, Elo00] used in this study. LGscore and LGscore2 are similar to several other measures used in CASP and CAFASP [LKDS99], including GDT [ZVMF99] and MaxSub [SERF00]. The most significant fragment should be as long and similar (between the two structures) as possible. It is too time consuming to look at all possible fragments. Instead a heuristic algorithm is implemented which searches for the best scoring fragment.

2.4.1 LGscore

LGscore [BEFR00, CZF+00, Elo00] is based on a method, recently introduced by [LG98], to calculate the significance of the similarity between two structures. This measure is based on the following score:

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

where M is equal to 20, d_{ij} is the distance between residues i and j, d_0 is equal to 5 Å and N_{gap} is the number of gaps in the alignment.

To calculate the statistical significance of this score Lewitt and Gerstein [LG98] used a set of structural alignments of unrelated proteins to calculate a distribution of S_{str} dependent

length	10^{-1}	10^{-3}	10^{-5}	10^{-10}
25	5.0	2.6	-	-
50	6.5	4.2	2.5	-
100	8.6	6.2	4.6	1.8
200	11.1	8.6	6.9	4.4

Table 2. LGscore dependency on fragment length (number of residues) and RMSD value (Å) for a given fragment.

on the alignment length, l. From this distribution a P-value dependent on S_{str} and l was calculated. A lower P-value indicates it has less chance the similarity is coincident. LGscore works under the assumption that the most significant segment has the lowest P-value. Thus, the algorithm searches for the fragment with the lowest P-value. Table 2 shows LGscore for an ungapped fragment given the number of residues and RMSD. A short fragment need to be more accurate on per residue basis than a longer fragment to receive the same LGscore. A one hundred residue fragment with RMSD of 6.2 Å receives an LGscore of 10^{-3}, which is the same as a 25 residue fragment with RMSD 2.6 Å.

2.4.2 LGscore2

Often when the fold is correctly predicted the alignment is sub-optimal. An "alignment dependent" measure like LGscore will give a low score in such cases because LGscore it is locked to compare residue #i in the model with residue#i in the crystal structure. To ignore this problem it is possible to use an "alignment independent" algorithm, which superimposes the model with the correct structure before the evaluation. A search is initiated to shift the alignment to find a better fit between the crystal structure and the model. After the superposition the structurally aligned residues are considered to be equivalent. From these equivalences it is possible to detect the most significant subset with the same algorithms as described above (for LGscore). This "alignment independent" measure is termed LGscore2 [CZF+00].

2.5 Supervised learning

When using computers to solve a problem it can often be explicitly described to the computer how the output is required from the input. This approach is however not always convenient because the method to receive output from a given input is unknown or too computationally expensive. An alternative approach for these problems is to teach the computer the functionality by supplying examples of input/output pairs. The computer can then, through a supervised learning algorithm, learn the functionality between input and output [CST00]. Supervised learning algorithms are being used for a variety of tasks today, including hand written text recognition, image recognition and bioinformatics [Bis95]. In the field of bioinformatics it is utilized for many different tasks, for example protein homology detection, gene expression analysis, identification of DNA-regions of interest (e.g.cleavage sites and protein binding sites). Here two algorithms for supervised learning will be discussed. The current work involved an artificial neural network, which will be discussed first. Initial tries were made using a support vector machine but it did not work well for our purpose. SVMs will be briefly explained in a later section.

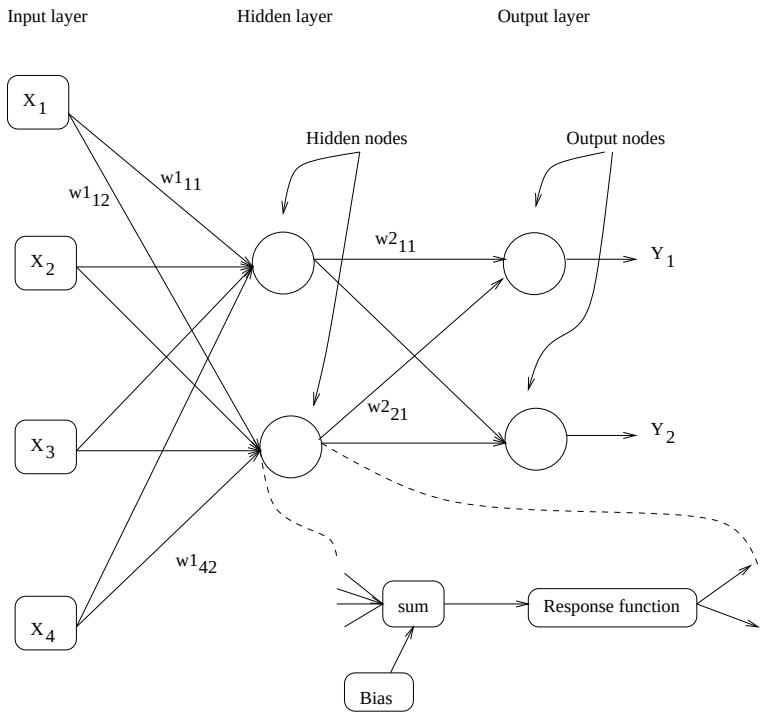

Figure 1. An schematic picture of a two layered neural network. This instance has 4 input variables($X_1, .., X_4$),two hidden nodes and two output variables (Y_1, Y_2). There is one weight for each synapse, which is multiplied with the output from the previous node as it is transfered via the synapse to the next node. The enlargement of one node shows what happens in each node. First the input to the node is added, and a bias is also added to the final sum. $sum = \sum_{i=0}^{4}(x_i \times w_{i2}) + bias$. The sum is then passed through a response function of some kind, i.e. linear or logistic response functions. The output is then sent out through each synapse from the node. Figure idea from Emanuelsson [Ema98].

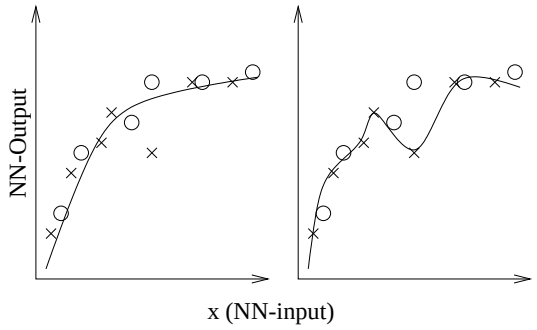

Figure 2. An illustration of what happens in the case of "over-training". The crosses symbolize the training set and circles the test set. The left diagram shows an optimally trained net and the right diagram shows an "over-trained" NN. The "over-trained" NN performs worse on test data (circles) than the optimally trained NN.

2.5.1 Artificial Neural Network

The artificial neural network (hereafter called neural network or NN) is built with the architecture of the brain as a model. An example of a neural network is given in Figure 1. The nodes (sometimes called neurons) are connected with synapses, equipped with a weight to reinforce or repress signals.

In a two layered NN (Figure 1) the input variables are transfered through synapses to the hidden layer of output nodes. In these nodes a response is triggered, and values are sent from the nodes in the hidden layer to the nodes of the output layer. Where a response is triggered, which results in the output variables.

A NN is trained against a set of input/output pairs (called the training set). It involves adjusting the weights and the biases so that the input/output pairs match. A Neural network is trained for a set number of "training cycles" predetermined by the user. For each loop the algorithm runs the net forward for every input in the training set. The offset between the expected output and the actual output is evaluated through an error function (E). For Pcons we used a sum of squares error function.

$$E = \frac{\sum_{n=1}^{N} \sum_{k=1}^{c} [(Y_{n,k} - T_{n,k})^2]}{2}$$

Where c is the number of training examples and N is the number of output variables. Y is the actual NN-output and T is the target outputs. Gradients with respect to weights and biases ($\frac{dE}{dw_{ij}}$) are calculated and biases and weights are updated for next training cycle.

It is important to stop the training in time because too many training cycles will result in an "over trained" NN [Bis95]. Over-training is when the NN learns the noise of the training set. This will decrease the NN's ability to generalize, see also Figure 2. An "over-trained" network will perform worse on unknown data.

For more readings on supervised learning see Cristianini et al. [CST00] and Bishop [Bis95]

2.5.2 Support vector machines

The support vector machine (SVM) is another algorithm for supervised learning, built for classifying the data into two classes. Multiple SVMs can obviously divide a data set into more

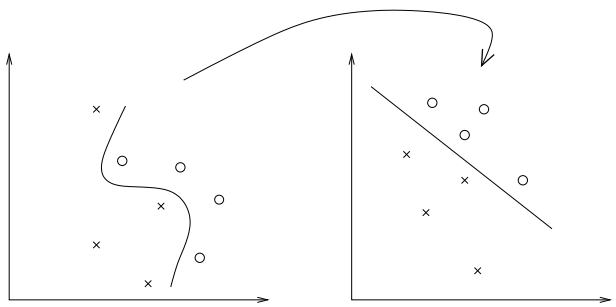

Figure 3. The classification approach used by SVMs. The data points is transfered to a higher dimension using a kernel-function supplied by the user. In the higher dimension (to the right) the data points can be split by a hyper-plane.

than two classes, and specialized SVM algorithms for regression problems exists [CST00]. The approach of the standard SVM algorithm is to transfer the input points of the training set to a higher dimension where they can be divided with a hyper-plane, see Figure 3.The transfer to a higher dimension is accomplished through a user defined kernel function. To avoid "over-training" an optimization algorithm is built into the SVM. Support vector machines often perform better than a neural network on classification tasks, because they are designed to find the optimal separating hyper-plane [CST00].

3 Methods

3.1 Pcons

Pcons does not make any new structure predictions. Instead it selects between predictions made from six publicly available web-servers. A flow chart of how Pcons predicts the structure of a unknown protein sequence (query) is given in Figure 4. Each query sequence is submitted to the servers, the predictions are collected, evaluated and given a Pcons-score. Pcons utilizes two types of information from the servers, the server-score and the fraction of predictions from any server that are similar to the one being evaluated. A neural network gives each prediction a score based on previous experience.

3.2 The dual use of LGscore and LGscore2

LGscore and LGscore2 [BEFR00, CZF$^+$00, Elo00] are two related algorithms for comparing two proteins and determining how structurally alike they are, see Section 2.4. We have used LGscore (and LGscore2) for a dual purpose in this study. First to distinguish between good and bad predictions, the model was in this case compared to the crystal structure using LGscore2. A model that received $LGscore2 < 10^{-3}$ was considered to be correct. LGscore2 was also used in LiveBench-1 to evaluate protein structure similarity but they used a more lenient cutoff $LGscore2 < 10^{-2}$[BEFR00].

The second usage of LGscore/LGscore2 were to identify similar predictions. One complication by using publicly available fold recognition methods is that the different methods use different fold libraries. To overcome this problem we used LGscore or LGscore2 to find structural similarities between templates and models.

Method	Score	Model/Template	All	"RANK1"	"SCORE"	Similarity Measure
NN-score	Yes	none	No	No	No	None
NN-all	Yes	Both	Yes	No	No	LGscore2
NN-combined	Yes	Both	Yes	Yes	Yes	LGscore2
NN-noscore	No	Both	Yes	Yes	Yes	LGscore2
NN-model	Yes	Only model	Yes	Yes	Yes	LGscore

Table 3. Description of different Pcons-versions tested. NN-score is a reference using only the server-score. NN-model is relying on model comparisons using LGscore instead of LGscore2.

3.3 Similarity between predictions

The similarity between two predictions were measured two ways, by structural comparisons of the models and the template proteins. A pair of models, or templates, were assumed to be similar if the structural alignment has a P-value, as measured by LGscore2, less than 10^{-3}.

Pcons groups the predictions collected from the web-servers two ways according to the score it received (from the server). "RANK1" includes the highest scoring prediction from each server, and "SCORE", includes the predictions with a score better than the significant cutoff given in Table 1.

Six fractions were calculated to estimate the frequency of that fold. The three fractions based on model comparisons were calculated the following way:

- Fraction of all predictions having a model similar to current prediction's model.
- Fraction of "RANK1"-predictions having a model similar to current prediction's model.
- Fraction of "SCORE"-predictions having a model similar to current prediction's model.

The same fractions were calculated for the template similarities found. The reason behind using fractions instead of number of similar templates (or models), is to normalize the variables in case a prediction server returns less than 10 predictions for one query. Using different combinations of these similarity measures, five different neural network versions were trained, see Table 3.

3.4 The different Pcons versions

The first network, NN-score, uses only the scores from the servers, while NN-all, in addition uses the fraction of all models and templates that are similar to the evaluated model. In NN-combined we use two additional types of structural comparisons, the fraction of similar first ranked ("RANK1") models/templates and the fraction of similar models/templates that are above the proposed cutoff ("SCORE") for each server. In NN-noscore we use the same information but exclude the scores. Finally we made a network that can be used for fast comparisons, NN-model. This method does only uses information about the model, and not about the template. To measure the similarity between two models NN-model uses the alignment dependent LGscore instead of LGscore2. The alignment dependent method LGscore is much faster than LGscore2 as it does not perform a structural superposition, therefore it can obviously not be used to compare the templates.

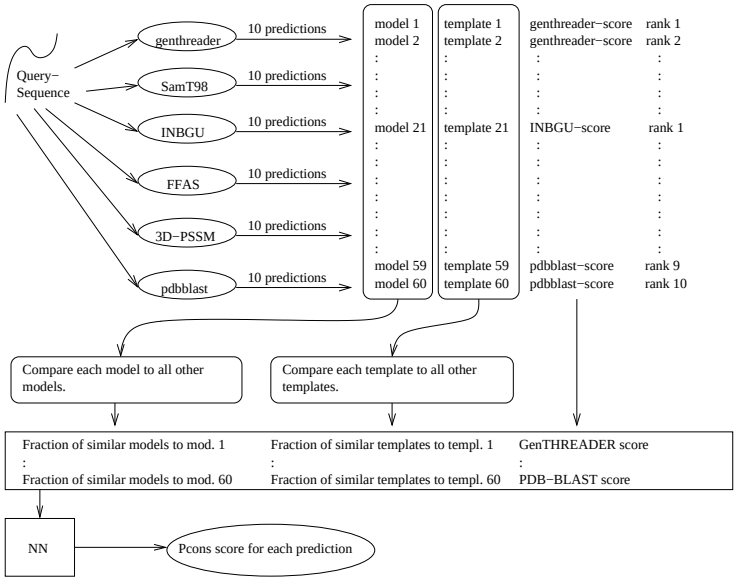

Figure 4. A schematic figure of the consensus predictor, this is the NN-all version. The query sequence is submitted to the 6 servers and up to 60 predictions are collected. Similarities between predictions are investigated through model/template comparisons. Finally, the NN gives each prediction a score.

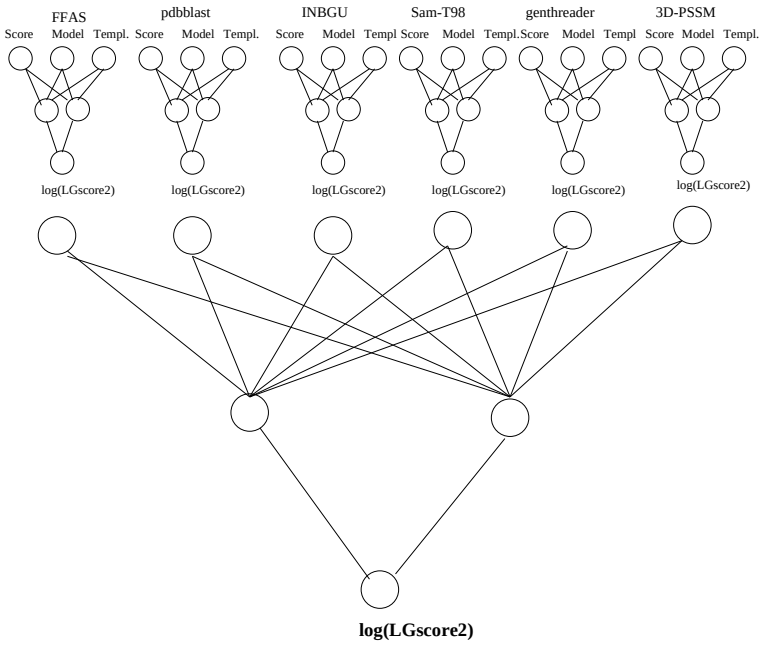

Figure 5. The Neural network architecture for NN-all. Three data-points are fed into the network, the score, the fraction similar models and the fraction similar templates. A separate network is trained for each server For each model obtained from one server the log of LGscore2 is predicted by a first layer neural network. The output from these networks is then fed into the jury network.

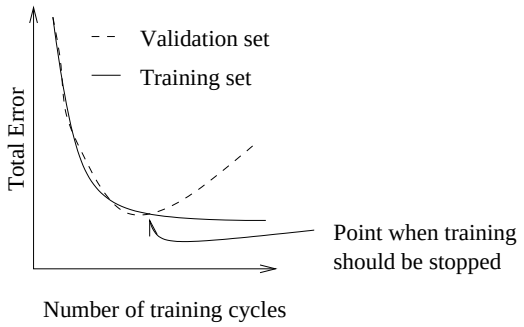

Figure 6. The number of training cycles is determined through monitoring sum of error for both training data and a validation set. The error for the validation set starts to increase as the net becomes over-trained.

3.5 Neural networks

For the neural network implementations we used Netlab [NB91], a neural network package for Matlab. A linear activation function was chosen since it did not carry restrictions on the range of the output. The training was carried out using error back-propagation with a sum of square error function and the scaled conjugate gradient algorithm to figure the step direction and length.

The Neural network architecture is built in two layers, see Figure 5. First layer consists of one network for each method. The output from the method neural network is fed into a final neural network and is thereby normalized. It should be noted that the jury network is not absolutely necessary but we found that including it increased the performance slightly.

The minimization of the error function (training) should be done with optimal number of hidden nodes and training cycles to avoid 'over-training' and minimize the training time. The number of hidden nodes, which is a kind of degree of freedom for the NN, should be large enough to carry out the separation. Too many hidden nodes will lead to a slower training process and also the risk for over-training is increased. The appropriate number of training cycles is decided by the same approach as [ENvH99]. We monitored the error-sum magnitude of both the training and a validation set through each cycle of the training. The ultimate number of cycles is when the error sum for the test set stops decreasing and starts to increase, see Figure 6. It should be noted that both the number of hidden nodes and the number of training cycles are decided at one time before the rest of the experiment is carried out. The first layer of neural networks (one for each method) were trained 150 to 500 cycles with 7 or 9 hidden nodes and the final-NN (second layer) was trained 200 cycles with 5 hidden nodes.

We used a 4 fold cross validation developing Pcons with the LiveBench-1 data set. This implies that the neural network is trained on three the dataset and tested on fourth quarter. This was repeated 4 times to receive data from all quarters of the data set. This is not true for the validation test sets (CAFASP2 and Livebench-2). In those cases the whole Livebench-1 data set was used to train the NN.

3.5.1 Regression

NNs are most commonly used to classify the examples into two classes (for example to predict good (=1) and bad (=0) protein models). In this case a NN-output closer to one is more likely to belong to the "good" class while a NN-output closer to 0 is more likely to belong to the "bad"class. However, we used the NN as a regression device predicting $-\log_{10}(LGscore2)$, thus a NN-output of 3 predicts a $LGscore2 = 10^{-3}$.

In initial studies a support vector machine, SVM-light [Joa99], was used instead of the neural network. However, SVM-light only supported classification and not regression, which made it less suitable for us.

3.6 Test and training data

3.6.1 The LiveBench-1 set.

This study is based on 125 different sequences and 6 different servers producing up to ten different models for each sequence. The input data comes from the the LiveBench program, that resulted in 125 targets submitted in the period between 1999/10/29 and 2000/04/06. The target proteins were divided into 30 easy targets (EASY category) and the 95 remaining difficult targets (HARD category). Per definition, easy targets were correctly predicted by PDB-BLAST [BEFR00] with an E-value $<$ 1.e-5. For this classification MaxSub [SERF00], and not LGscore2 as in this study, was used to evaluate if the model was correct. The performance of the different servers using this set of data has been analyzed carefully in an earlier study [BEFR00].

3.6.2 Additional test sets

To validate the performance of Pcons we have used two additional test-sets. We have used the 25 fold recognition targets that were evaluated in the CAFASP2 [SERF00] sessions of CASP4 [MHFP99]. Secondly we have used a new LiveBench test set created between 2000/04/13 and 2000/09/22.

3.7 Evaluation of performance of fold recognition methods

This section will explain the measures used to evaluate performance of Pcons in this study.

3.7.1 Correlation coefficient

The correlation coefficient can be calculated between two vectors. It is calculated the following way for vector y and x [Olb96]:

$$r = \frac{\sum_1^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_1^n (x_i - \bar{x})^2}\sqrt{\sum_1^n (y_i - \bar{y})^2}}$$

In this application the neural network is trained to predict $-\log_{10}(LGscore2)$. The correlation between the NN-output and $-\log_{10}(LGscore2)$ was therefore calculated as a measure of network performance.

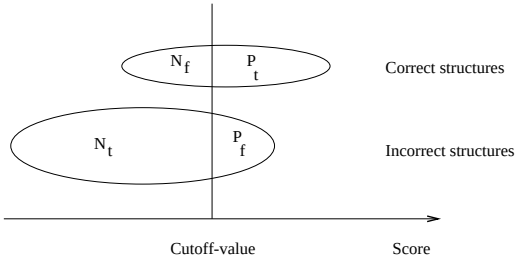

Figure 7. This figure explains the terms P_f, P_t, N_t and N_f. The top data set (in the figure) includes all predictions which are correct, i.e. has $LGscore2 < 10^{-3}$.

3.7.2 Matthews correlation coefficient, specificity and sensitivity

It is vital to have a measure of how well the method is distinguishing between good and bad models. To evaluate this we labeled each Pcons-score according to Figure 7. A positive model is one receiving a score better than the predetermined cutoff value. When evaluating the server performance we used their proposed cutoff (see Table 1). Pcons gives predictions a score from 0 for the worst models up to about 20 for the best ones. Since Pcons aims to foresee $-\log_{10}(LGscore2)$ it was natural to use 3 as the Pcons-score cutoff. However, for some evaluations we used other cutoffs. By adding the number of test-examples in each class Matthews correlation factor was calculated according to:

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

Matthews correlation factor varies between -1 and 1, where 1 equals perfect predictions, 0 is random predictions, and -1 is opposite predictions.

Specificity and sensitivity are two other commonly used evaluation variables. They are defined as:

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

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

The sensitivity is the fraction of all structures in the "correct" class predicted positive. The specificity on the other hand is the fraction of the predicted positive which belongs to the "correct" class.

4 Results

We chose to study five different networks, namely NN-all, NN-combined, NN-noscore, NN-model and NN-score (see Table 3).

Method	Easy (30)	Hard (95)	All (125)
GenTHREADER	23	13	36
Sam-T98	22	16	38
FFAS	28	14	42
Inbgu	23	21	44
3D-PSSM	22	21	43
PDB-BLAST	28	10	38
NN-score	28	20	48
NN-all	27	29	56
NN-combined	28	30	58
NN-noscore	28	30	58
NN-model	28	29	57
Any method	28	65	93
Any method rank 1	28	42	70

Table 4. Number of correct models predicted as the best model. All 125 targets in LiveBench-1 was analyzed keeping the highest scoring model from each method ("RANK1"). All versions of Pcons perform similar except for NN-score which only used method score from the server.

4.1 Performance of consensus method on the LiveBench-1 data set.

In Table 4 the number of correct models from the individual servers and the consensus networks are shown. It is clear that the the consensus networks detect more correct targets than any individual server. For the easy targets all networks perform approximately as well as, but not better than, the best individual server. For the hard targets all the consensus methods that use structural comparisons detect significantly more (29-30) targets than the best individual server (21). For these targets NN-score, that does not use structural comparisons, does not show any improvement over the best server.

In addition to finding a correct structure it is also important to be able to separate incorrect models from correct models. One way to study this is to sort the models according to the score and then plot the cumulative number of correct models versus incorrect models as has been done in several earlier studies [Elo00, PTHC97, PKB$^+$98]. The consensus predictor, NN-model, finds about 20% more correct models for any number of incorrect models than the best individual server, see Figure 8. For clarity we have only shown NN-model in this figure. The other networks perform slightly better. The consensus methods improve the predictions. However, according to Table 4 there exist correct models for 93 targets. This indicates that it might be possible to use additional criteria to further improve the consensus predictors. However we have not been able to do this.

From Table 4 it seems as if these four networks perform similarly. However from the correlations coefficients it can be seen that NN-model does not perform as well as the other networks, see Table 5. The main reason to use NN-model is that it is significantly faster than the others, as it uses the alignment dependent algorithm LGscore instead of the alignment independent LGscore2. In Table 5, the Matthews correlation coefficients (Mc), correlation coefficients and sensitivity/specificity values for all the Pcons methods are shown. It can be seen that at an NN-output of 3 about 80% of the models are correct and up to 75% of all correct models are detected. At 5 the specificity is higher than 90% but the sensitivity has dropped below 50%. Interestingly it seems as if NN-model has a slightly higher specificity but a lower sensitivity than the other networks.

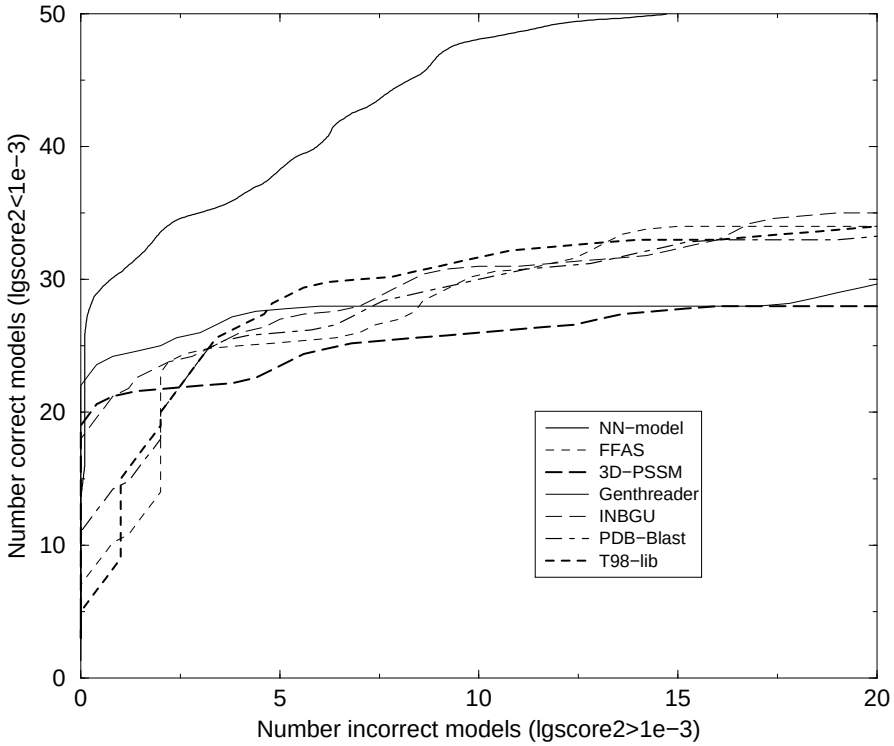

Figure 8. Cumulative plot of correct versus incorrect models. To make the curves easier to analyze they are smoothed by a running average. Correct and incorrect models are defined by using LGscore2 and a cutoff of 10^{-3} The X-axis reports the number of incorrect models according, while the y-axis indicated number of correct models.

Method	Mc($NN_{cut} = 3$)	Spec/Sens ($NN_{cut} = 3$)	Spec/Sens($NN_{cut} = 5$)	Correlation
NN-score	0.54	0.80/0.46	0.91/0.36	0.61
NN-all	0.72	0.81/0.75	0.94/0.46	0.77
NN-combined	0.73	0.81/0.77	0.95/0.46	0.80
NN-noscore	0.74	0.84/0.75	0.95/0.48	0.81
NN-model	0.70	0.87/0.66	0.98/0.41	0.76

Table 5. Matthews Correlation Coefficients, specificity, sensitivity and correlation coefficients (between the true and predicted $-\log_{10} LGscore2$) calculated for the different networks.

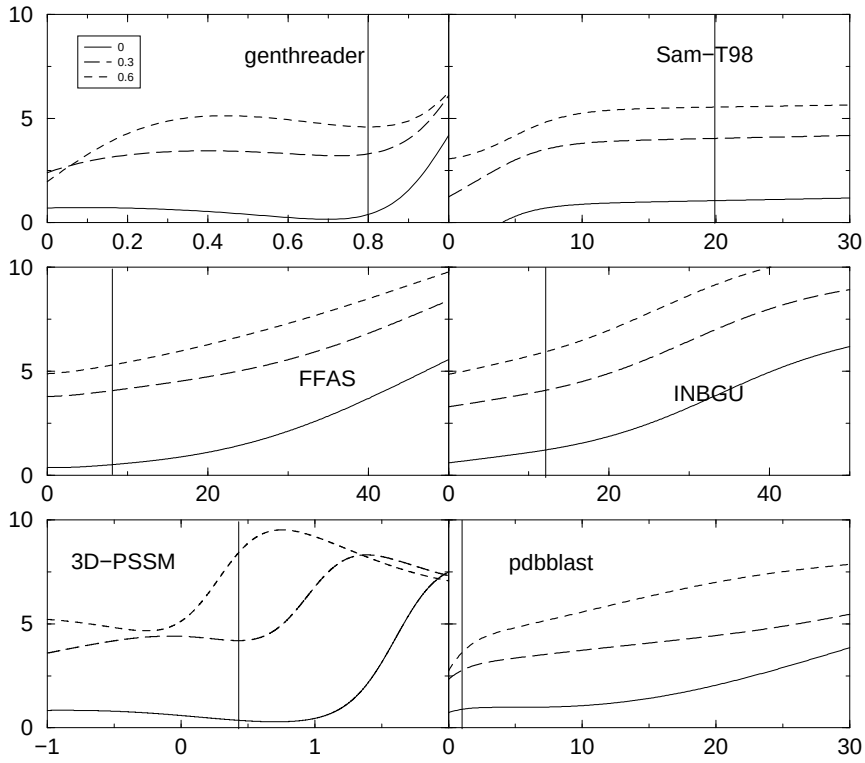

Figure 9. Response plot for different fraction of similar networks. The X axis correspond to the score from a particular method (the negative log of the scores for 3D-PSSM and PDB-BLAST is used for clarity).The Y axis corresponds to the output from the network. In each figure three curves are shown, one corresponding to no other similar models, one to 30% similar models and one to 60% similar models.

The good performance of NN-noscore was a surprise to us. We did not expect it to be possible to disregard the score of the individual methods and still perform quite well. One possible explanation is that it is very unlikely that an incorrect model should be similar to many other models. The correlation coefficient and Mc for NN-score is substantially lower than for the networks that use structural comparisons. In Table 4, it can also be seen that NN-score does not detect more easy or hard targets than the best servers. However the overall performance is slightly better as different methods are best at easy or hard targets. This indicates that the additional information included in the model/template comparisons is the main reason for the improvement achieved by Pcons.

4.2 Analysis of Neural network response

To get a better understanding of the factors that are important for Pcons we trained a set of networks using only the scores and the fraction of similar models, i.e. the same input as in NN-all but excluding the template comparisons. The overall performance of this method is slightly better than the overall performance of the best server. In Figure 9 the response to different inputs of the networks is shown. For each method three different curves are plotted, corresponding to 0, 30% and 60% similar models. In each figure a vertical line is shown at the cutoff proposed for each server.

The curves vary between the different server NNs; however there are some striking similarities. At the proposed cutoff for the different methods all the methods produce an output of less than 1 when no similar protein is found and a score at about 3 when 30% similar models are found, and 5 for 60% similar models. It should be remembered that a network output of 3 correlates to an LGscore2 of 1.e-3, which is the cutoff used to define a correct prediction in this study. For low scores most methods produce an output score of 2-3 with 30% similar models. The maximum output for the different methods varies significantly. The networks trained on FFAS, Inbgu and PDB-BLAST data respond with high scores for good input scores, while GenTHREADER and Sam-T98 do not produce outputs higher than 6. The output for 3D-PSSM for very high scoring pairs behaves strangely. The network predicts that models with intermediate scores and many similar proteins are better than models with high scores. A possible explanation is that 3D-PSSM might give high scores to models based on distantly related templates[Mac00].The profiles for templates could include the sequences for closer templates of the same fold (they are correctly detected but because the templates are very distant the models will be of low quality). Moreover, the number of models receiving high scores are rare; thus random noise may distort the response curve. For trivial predictions the model will always be chosen from Inbgu, FFAS or PDB-BLAST. This could be an explanation for why GenTHREADER is chosen so rarely. The reason for these outputs can be found in the scoring function used by the different servers. The difference between a strong hit and an extremely strong hit is quite low for 3D-PSSM, GenTHREADER and Sam-T98, while for Inbgu, FFAS and PDB-BLAST the difference is much larger. With no structural neighbors the curves for FFAS, Inbgu, 3D-PSSM and GenTHREADER suddenly increase to about 3 when the score is good enough. For GenTHREADER this happens between 0.8 and 1, for FFAS and Inbgu between 20 and 40, and for 3D-PSSM between $10^{-1.5}$ and 10^{2}. For Sam-T98 and PDB-BLAST even very good scores result in quite low outputs.

4.3 Analysis of origin of models selected by the consensus method.

A consensus prediction only sets new scores to predictions from other methods. In Table 6 the origin of the first ranked models by the consensus predictors is shown. The most common origin for predictions by NN-score is Inbgu, while for all other networks 3D-PSSM is the origin for most of these models. A model with an origin in FFAS is also quite common (15 to 24%), while GenTHREADER, Sam-T98 and PDB-BLAST are less frequent. In NN-noscore all methods except GenTHREADER are chosen at almost the same frequency. The surprise in Table 6 is that so few models from GenTHREADER are given the best Pcons-score. As can be seen in Figure 8, GenTHREADER is one of the methods with best ability to distinguish a correct from an incorrect model. It is possible that the trivial models are never chosen from GenTHREADER, see above. Moreover, the model quality of GenTHREADER models is rarely the best [BEFR00].

The consensus method does not necessarily have to select the first ranked from a particular method. In Table 7 it can be seen that a first ranked model is chosen in 26 to 82% of the cases. When structural comparisons are included a significant number of higher ranked models are chosen. NN-score should in theory only pick ranked #1 predictions since they received a better score than all lower ranked prediction. However, due to noise and over-training some lower ranked models are picked by NN-score too. The high ranked model chosen by NN-model selects more first ranked models than the other networks, and NN-combined selects fewer than NN-all. This indicates that as more structural comparisons are included the less important

Method	GenTHREADER	Sam-T98	FFAS	Inbgu	3D-PSSM	PDB-BLAST
NN-score	6	11	15	40	19	9
NN-all	5	8	19	26	34	8
NN-combined	7	9	17	20	33	14
NN-noscore	4	14	23	19	25	16
NN-model	4	8	24	27	27	10

Table 6. This table illustrates which server the highest scoring prediction comes from. The 125 LiveBench-1 target are analyzed. Figures are given in percent of 125 predictions.

Method	Rank 1	Rank 2	Rank 3	Rank 4 to 10.
NN-score	82	10	3	5
NN-all	48	12	7	33
NN-combined	42	11	8	39
NN-noscore	26	15	10	50
NN-model	57	14	5	25

Table 7. The highest scoring model for each of the 125 unknown sequences is analyzed in this table. In column rank 1 the models which were ranked as best by the prediction server, rank 2 second highest ranked etc.Figures are given as a percent of 125 predictions.

the score becomes.

4.4 The performance of Pcons is sustained on additional test sets.

In addition to the jack-knifed tests on the LiveBench-1 data, we have used two additional test sets, CASP4 and the LiveBench-2 set for proteins released during April and Sept 2000. In the CASP4 set as well as for some of the LiveBench targets we had no predictions from Sam-T98, which had been replaced by Sam-T99. To compensate for this we retrained the networks without Sam-T98, using all the LiveBench-1 data. In all these predictions we have used NN-model since this is the method implemented as a part of the meta-server at http://bioinfo.pl/meta/.

The LiveBench-2 data was obtained in a similar way as LiveBench-1 data but during the following time period. On this data the consensus prediction performed better than any individual method predicting the correct structure for 53 out of 121 targets, with the second best performance by 3D-PSSM which predicted 47 targets correctly. It should also be noted that a new server, Fugue, which was not included in LiveBench-1 predicted 37 targets correctly, indicating that there exist new good servers that could be included in future versions of Pcons. The successor of Sam-T98, Sam-T99, also performed quite well on LiveBench-2 and could be included in the next version.

In the automatic evaluation of the CAFASP2 another method, MaxSub [SERF00] was used to evaluate the targets. Using this evaluation Pcons did not perform significantly better than the best individual server. The consensus predictor predicted 6 (out of 26) models correctly, with a total MaxSub score of 11.1. FFAS also predicted 6 models correctly with a total MaxSub score of 11.6. However it should be noted that for 18 out of the 26 targets none of the individual methods made a correct first ranked prediction. There were only eight targets where the consensus predictor possibly could make a correct prediction. In one case, T0110, the consensus prediction made a suboptimal choice by choosing the first model from Inbgu

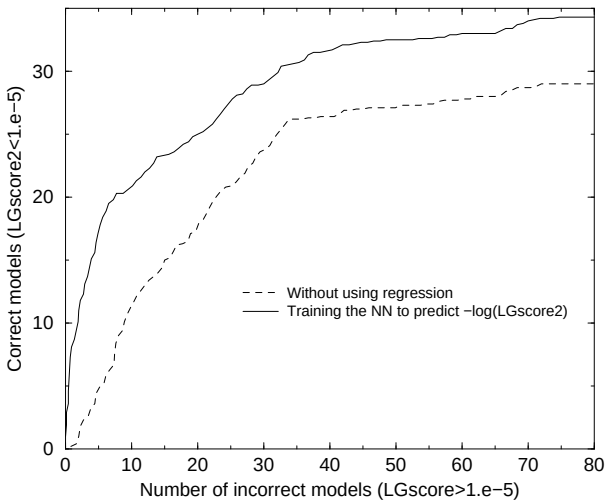

Figure 10. This figure shows the difference between using a regression NN and a classification NN. It is a cumulative plot of correct versus incorrect models for the highest scoring predictions generated same way as Figure 8 but using a more stringent definition of a correct model ($LGscore2 < 10^{-5}$).

instead of the first model from 3D-PSSM. The 3D-PSSM model was the second choice by the consensus predictor.

4.5 Importance of using regression

To investigate the importance of training the NN against $-\log_{10} LGscore2$ and not only using it as a classification device we trained an additional version of NN-model. It was trained to predict only if a model is good or bad , see Section 3.5.1.

The regression NN did not predict more correct models on the LiveBench-1 test set. They both made 57 correct predictions, see Table 4. However, the quality of the chosen models differ,see Figure 10. Only 29 out of the classification NN's 57 correct models received an LGscore2 better than 10^{-5}. The regression NN on the other hand found 34 with an LGscore2 better than 10^{-5}. Thus predicting $-log_{10}LGscore2$ of each model improves the performance of Pcons; it actually gives a value of the model quality instead of a likelihood for the model to be correct or incorrect.

4.6 Time requirements for running Pcons.

The time limiting step in running Pcons is the comparison of models and templates. One comparison is often done under 30 seconds using LGscore2. However, the total time to do all comparisons depend on the number of predictions squared. For 60 predictions all comparisons can take several hours to perform. This can become especially troublesome when extending Pcons to use predictions from new servers. To overcome this problem we decided to use the faster LGscore1 algorithm in NN-model. LGscore1 is much faster than LGscore2, but is alignment dependent and thus is not suitable for comparing templates.

Pcons versions using template similarities seemed to perform slightly better than NN-model. However, it is not possible to search such similarities with an alignment dependent

algorithm like LGscore. However, we believe gain in performance is too small to justify more computationally expensive algorithm comparing the template proteins. Implementation of a database with template proteins can be one way around the problem.

5 Conclusions

In this study we present a novel method, Pcons, which makes a consensus fold recognition prediction. The studied version of Pcons uses models from 6 web-servers and tries to predict the quality of all these models. Pcons uses the score of the model as well as the fraction of other models that are similar as its input. Pcons detects about 10% more correct models than the best individual method. The main reason for the improvement is structural comparisons between the models and templates. In an in-house fold recognition method the structural comparison part could have been replaced by assigning template similarities if two templates belong to the same fold. However as the different servers use vastly different fold libraries we found the structural comparisons to be necessary.

From this study it is evident that a combination of several methods improves the performance of fold recognition methods. Every method is optimized to perform as well as possible and several methods use very similar algorithms, therefore one could expect that it would be difficult to combine them into a consensus prediction. Now the question is: why does it appear to be impossible for a single method to incorporate the information contained in all methods?

We believe that there are three major reasons for the improvements using a consensus method. (1) The structural similarity information used here is not included in most fold recognition methods. (2) The consensus method takes multiple models for each template into account. (3) Pcons normalizes the scores so that they relate better to the quality of the model. From earlier CASP experiments it has been a common practice among the best manual predictors to not only look at the number one prediction, but rather select the most common fold among the top predictions. This is what Pcons tries to reproduce. The second factor is that we are not only studying one alignment for a given pair of fold and sequence, but we are studying several. By using several alignments it is possible that one of them captures the important features of the correct structure. For instance in some cases it might be better to use predicted secondary structures and in some not. It should be possible to develop a single method that takes all this information into account. The final difference is that the consensus predictor tries to predict the quality of the models and not simply if the correct fold is found or not. In preliminary studies we tried a simple correct fold/incorrect fold measure, however this was not successful. As NN-score does not perform significantly better than the best individual servers, it seems as if (1) is most important for the improvement. It is very likely that the results from the individual servers can be improved by using a scoring dependent on the score for all sequence-template pairs of the same fold.

Prediction servers get updated from time to time. This will result in a decreased performance of Pcons. To minimize these problems Pcons should be retrained on new training data as often as possible, perhaps every half year.

There is a potential for improving the Pcons selection further since there still are models which are correct but are not found by the current version of Pcons (see Table 4). Attempts to include other types of in-data to the NN have been made. Lengths of models, template and query sequence did not improve performance. Future developments may include threading techniques for evaluating the sequence to structure. For example a prediction based secondary

structure match or a pairwise energy function [Sip93, BHPL97]. Initial tries with pairwise energy functions have been made and did not result in any improvements. However, further investigations may prove fruitful.

It is fairly easy to add new servers to Pcons. The only thing required to update Pcons is new training data, which can be acquired through continuation of LiveBench. However, there is a risk to involve too many servers since it increases the noise too, (especially if the quality of the servers are not good). Fugue, Sam-T99 and mGenTHREADER are three examples of possible servers to include in future versions Pcons. The two latter servers are improved versions of servers already included in Pcons, and thus will merely be replacing the old versions.

Acknowledgements

First of all I would like to thank my supervisor Arne Elofsson, at Stockholm Bioinformatics center, for his guidance and valuable ideas throughout the project. I would also like to thank all the people at Stockholm bioinformatics center for technical assistance, proofreading and for providing a stimulating atmosphere to work in.

References

[AMS+97] S. F. Altschul, T. L. Madden, A. A. Schaffer, 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 Res.*, 25:3389–3402, 1997.

[Bar96] G. J. Barton. Protein sequence alignment and database scanning. In M. J. E. Sternberg, editor, *Protein Structure Prediction*, chapter 2. IRL Press, 1996.

[BEFR00] J.M. Bujnicki, A. Elofsson, D. Fischer, and L. Rychlewski. Livebench: Continous benchmarking of protein structure prediction servers. *submitted*, 2000.

[BHPL97] E. S. Huang B. H. Park and M. Lewitt. Factors affectin the ability of energy functions to discriminate correct from incorrect folds. *Journal of Molecular Biology*, 266:831–846, 1997.

[Bis95] C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[BLE91] J. U. Bowie, R. Lüthy, and D. Eisenberg. A method to identify protein sequence that fold into a known three-dimensional structure. *Science*, 253:164–170, 1991.

[BWF+00] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[CST00] N. Christianini and J. Shawe-Taylor. *An introduction to support vector machines : and other kernel-based learning methods*. Cambridge University Press, 2000.

[CZF+00] S. Cristobal, A. Zemla, D. Fischer, L. Rychlewski, and A. Elofsson. How can the accuracy of a protein model be measured ? *Manuscript in preparation*, 2000.

[DLAS00] F.S. Domingues, P. Lackner, A. Andreeva, and M. J. Sippl. Structure based evaluation of sequence comparison and fold recognition alignment accuracy. *J. Mol. Biol*, 297(4):1003–1013, 2000.

[Elo00] A. Elofsson. A study on how to best align protein sequences. *submitted*, 2000.

[Ema98] O. Emanuelsson. Prediction of subcellular location of plant proteins using neural networks. Master's thesis, Uppsala university school of egineering, 1998.

[ENvH99] O. Emanuelsson, H. Nielsen, and G. von Heijne. Chlorop, a neural network-based method for predicting chloroplast transit peptides and their cleavage sites. *Protein Science*, 8:978–984, 1999.

[FBB+99] D. Fischer, C. Barret, K. Bryson, A. Elofsson, A. Godzik, D. Jones, K.J. Karplus, L.A. Kelley, R.M. MacCallum, K. Pawowski, B. Rost, L. Rychlewski, and M. Sternberg. Critical assessment of fully automated protein structure prediction methods. *Protein: Structure, Function and Genetics*, Suppl 3:2–6, 1999.

[FE96] D. Fischer and D. Eisenberg. Protein fold recognition using sequence-derived predictions. *Protein Sci.*, 5:947–955, 1996.

[Fis00] D. Fischer. Hybrid fold recognition: Combining sequence derived properties with evolutionary information. In R.B. ALtman, A.K. Dunker, L. Hunter, and T.E. Klien, editors, *Pacific Symposium on Biocomputing*, volume 5, pages 116–.127. World Scientific, 2000.

[GME87] M. Gribskov, A. D. McLachlan, and D. Eisenberg. Profile analysis: detection of distantly related proteins. *Proc Natl Acad Sci U S A*, 84:4355–4358, 1987.

[Joa99] T. Joachims. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999.

[Jon99a] D.T. Jones. Genthreader: an efficient and reliable protein fold recognition method for genomic sequences. *J. Mol. Biol.*, 287(4):797–815, 1999.

[Jon99b] D.T. Jones. Protein secondary structure prediction based on possition specific scoring matrices. *J. Mol. Biol.*, 292:195–202, 1999.

[JTT92] D. T. Jones, W. R. Taylor, and J. M. Thornton. A new appoach to protein fold recognition. *Nature*, 358:86–89, 1992.

[KBM+94] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: applications to protein modeling. *J. Mol. Biol.*, 235:1501–1531, 1994.

[KMS00] L.A. Kelley, R.M. MacCallum, and M.J. Sternberg. Enhanced genome annotation using structural profiles in the program 3d-pssm. *J. Mol. Biol.*, 299(2):523–544, 2000.

[LG98] M Levitt and M Gerstein. A unified statistical framework for sequence comparison and structure comparison. *Proc Natl Acad Sci U S A*, 95(11):5913–20, 1998.

[LKDS99] P. Lackner, W.A. Koppensteiner, F.S. Domingues, and M.J. Sippl. Automated large scale evaluation of protein structure predictions. *Protein: Structure, Function and Genetics*, Suppl 3:7–14, 1999.

[Mac00] R. MacCallum. Personal communication, 2000.

[MBHC95] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *J. Mol. Biol.*, 247:536–540, 1995.

[MHFP99] J. Moult, T. Hubbard, K. Fidelis, and J.T. Pedersen. Critical assessment of methods of protein structure predictions (casp): Round iii. *Protein: Structure, Function and Genetics*, Suppl 3:2–6, 1999.

[NB91] I. Nabney and C. Bishop. *Netlab*. Free Software Foundation, Inc., 1991.

[NW70] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[Olb96] L. Olbjer. *Experimentell och industriell statistik.* Lunds universitet och Lunds tekniska högskola, Institutionen för matematisk statistik, 3rd edition, 1996.

[PKB+98] J. Park, K. Karplus, C. Barrett, R. Hughey, D. Haussler, T. Hubbard, and Chothia C. Sequence comparisons using multiple sequences detect three times as many remote homologues as pairwise methods. *J Mol Biol*, 284:1201–1210, 1998.

[PTHC97] J. Park, S. A. Teichmann, T. Hubbard, and C. Chothia. Intermediate sequences increase the detection of homology between sequences. *J. Mol. Biol.*, 273:249–254, 1997.

[RJWG00] L. Rychlewski, L. Jaroszewski, L. Weizhong, and A. Godzik. Comparison of sequence profiles. strategies for structural predictions using sequence profiles. *Protein Science*, 9:232–241, 2000.

[RSS97] B. Rost, R. Schneider, and C. Sander. Protein fold recognition by prediction-based threading. *J. Mol. Biol.*, 270:471–480, 1997.

[SERF00] N. Siew, A. Elofsson, L. Rychlewski, and D. Fischer. Maxsub: An automated measure to assess the quality of protein structure predictions. *Bionformatics*, in press, 2000.

[Sip93] M. J. Sippl. Recognition of errors in three-dimensional structures of proteins. *Proteins: Structure, function, and genetics*, 17:355–362, 1993.

[SS98] R Sanchez and A. Sali. Large-scale protein structure modeling of the saccharomyces cerevisiae genome. *Proc Natl Acad Sci U S A.*, 95(23):13597–13602, 1998.

[Ste96] M. J. E. Sternberg. Protein structure prediction-priciples and approaches. In M. J. E. Sternberg, editor, *Protein Structure Prediction*, chapter 1. IRL Press, 1996.

[SW81] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *J. Mol. Biol.*, 147:195–197, 1981.

[Ven00] R. Venter. Speech at Stockholm University, 2000.

[ZVMF99] A. Zemla, C. Veclovas, J. Moult, and K. Fidelis. Processing and analysis of casp3 protein structure predictions. *Protein: Structure, Function and Genetics*, Suppl 3:22–29, 1999.