Consensus Fold Recognition by Predicted Model Quality

by

Libo Yu

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Mathematics

in

Computer Science

Waterloo, Ontario, Canada, 2004

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revision, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Protein structure prediction has been a fundamental challenge in the biological field. In this post-genomic era, the need for automated protein structure prediction has never been more evident and researchers are now focusing on developing computational techniques to predict three-dimensional structures with high throughput.

Consensus-based protein structure prediction methods are state-of-the-art in automatic protein structure prediction. A consensus-based server combines the outputs of several individual servers and tends to generate better predictions than any individual server. Consensus-based methods have proved to be successful in recent CASP (Critical Assessment of Structure Prediction).

In this thesis, a Support Vector Machine (SVM) regression-based consensus method is proposed for protein fold recognition, a key component for high throughput protein structure prediction and protein function annotation. The SVM first extracts the features of a structural model by comparing the model to the other models produced by all the individual servers. Then, the SVM predicts the quality of each model. The experimental results from several LiveBench data sets confirm that our proposed consensus method, SVM regression, consistently performs better than any individual server. Based on this method, we developed a meta server, the Alignment by Consensus Estimation (ACE).

Acknowledgments

I wish to thank my supervisor, Dr. Ming Li, for his guidance, encouragement, patience, and financial support, which has been a tremendous help for me over the past two years. I also want to thank my cosupervisor, Dr. Jinbo Xu, for the many hours of discussions we had in which he showed his enthusiasm, and knowledge of science. I would also like to thank the readers of my thesis, Dr. Forbes Burkowski and Dr. Brendan J. McConkey, for their time and constructive advice. Finally, special thanks go to my parents and brother for their long-distance support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Proteins form our bodies. They make up cells and organs. Proteins also play an important role in biological processes. They perform a big variety of tasks: from breaking down food to fighting off diseases. Biologists are exploring protein function and how they govern the activities of body cells. Based on this, researchers can synthesize useful proteins such as enzymes and new drugs.

It is known that the significant feature of a protein is its ability to fold into the right shape for carrying out a particular function. In this sense, identifying a protein's shape, or structure, is pivotal for the understanding of the protein's biological function and its role in health and diseases. Since the 1950s, determining a protein's structure has been a fundamental challenge in the biological field [1]. Experimental methods have been developed to solve protein structures, such as X-ray crystallography or nuclear magnetic resonance spectroscopy (NMR). Both methods have drawbacks. In some cases, it is impossible to crystallize a protein,

and NMR can be applied to small and medium-sized molecules only. In addition, both methods are costly and time-consuming, since it often takes months to experimentally determine a single structure. Consequently, the number of available protein sequences has increased much faster than the number of solved structures due to advances in the molecular biology field in the past few years. In addition, the demands from biologists to solve protein structures with high throughput have increased.

People will fully utilize structural genomics only if automated, reliable tools are available to model proteins' structures from known structures. Although researchers have been working on protein structure prediction for decades, limited progress has been made in this area. Clearly, the human predictors' expertise must be reproduced and automated. Therefore, researchers have begun to utilize computers to help predict protein structures. The challenge lies in understanding the complex relationship between structures and sequences [2].

Recently, with the development of computational science and availability of high-performance computing facilities, various computational methods have been proposed to predict protein structures, based on known protein structures. Compared with experimental techniques, computational approaches are relatively cheap and efficient.

In recent years, many automatic protein structure prediction servers that use different methodologies have been set up [3, 4]. These automatic servers can be divided into four categories according to their prediction methods: ab initio, homology modeling, fold recognition, and consensus. Within each category, implementations differ from server to server. For instance, for fold recognition servers, a scoring function is usually required to measure the the alignment accuracy between a target sequence and a template. There are different ways to design the scor-

ing function, leading to significant differences in performance. Even for the same scoring function, different parameter values often result in different templates and alignments[5]. Each method has pros and cons. In experiments, no single server can generate the best predictions for all the targets, and the best predictions are often made by different servers [4]. So, a practical question is whether we can build a server by combining individual servers to generate better outputs? In other words, given a set of individual servers, if we can always somehow produce a better prediction from the outputs of the individual servers, we can build a more powerful server and the individual servers serve as component servers. This is the idea behind consensus servers, which are also called meta servers.

1.2 Challenge

The outputs of component servers are produced by using different methodologies or scoring functions. Therefore, it is not practical to develop a more effective scoring function to identify the model with the best possible quality, which can otherwise be implemented in an individual server. Meta servers should make use of the fact that their inputs are the outputs of their component servers. Improvements are anticipated by using different consensus algorithms. The simplest consensus algorithm is majority voting which is based on the assumption that if most of the servers make the same prediction for a target it is likely that the consensus prediction has the best structural quality. Our objective is to extract more information from input models and then apply advanced consensus algorithms for a better prediction. One choice is to use machine learning techniques which have been successfully applied in many bioinformatics applications. For example, Neural Networks (NN)

and Support Vector Machines (SVMs) have been used in fold recognition to select templates. To achieve the best performance with machine learning methods, we must extract effective features or patterns, a challenging task.

In addition, we can surmise that if we use more component servers, possibly we may have better candidate models in the collected inputs. However, it becomes challenging to identify the best model from the larger candidate set. Consequently, the number of component servers is critical for a meta server. Also, if all the individual servers have poor performance, we cannot expect any significant improvement from the consensus output. The performance of meta servers is highly dependent on the number of component servers and the performance of each component server. Some meta servers have been developed and assessed by the LiveBench tests [4, 6] and CAFASPs [3, 7]. It turns out that meta servers perform exceptionally well, surpassing the best individual servers, especially in specificity. In spite of that, the consensus algorithms used by these servers are quite simple and straightforward. Also, more advanced machine learning techniques can be applied to further improve meta servers' performance.

1.3 Goals

In light of the previous discussion, our goal is to extract effective features from input models, and apply advanced machine learning techniques to predict model quality, from which consensus outputs are selected. The consensus outputs should be superior to the outputs of any component server, in terms of sensitivity and specificity. In addition, the performance of a meta server should be more robust than that of its component servers.

1.4 Organization

The rest of this thesis is organized as follows. Chapter 2 is a brief introduction to protein structure and protein structure prediction methods, including ab initio, comparative modeling, and fold recognition. The basis of the proposed consensus algorithm, Support Vector Machine regression, is also introduced in detail in Chapter 2. In chapter 3, various consensus algorithms and different types of meta servers are investigated and compared. Chapter 4 presents our new meta server, the Alignment by Consensus Estimation (ACE). The components of the proposed meta server are addressed in detail, including feature extraction and implementation details. The experimental results obtained with the LiveBench data and some conclusions are reported in Chapter 5.

Chapter 2

Background

A protein is a chain of amino acids, but it is not the linear molecule that the amino acid string suggests. Each protein folds into a unique three-dimensional structure. The folded structure can serve as modules for building up large assemblies such as virus particles or muscle fibres. In addition, protein structure determines protein function, because protein molecules with closely matched shapes are more likely to associate with each other. For example, an enzyme is a protein that catalyzes biochemical reactions. The function of an enzyme relies on the structure of its active site, whose shape allows the enzyme to associate with other molecules. The active site also has some chemical properties which help form bonds with other molecules. Many diseases, including Alzheimer's and Bovine Spongiform Encephalopathy (madcow disease), are now known to result from the proteins that have folded into an incorrect shape [8].

Protein folding involves the formation of local structural motifs such as helices and sheets (secondary structures), and the arrangement of these individual

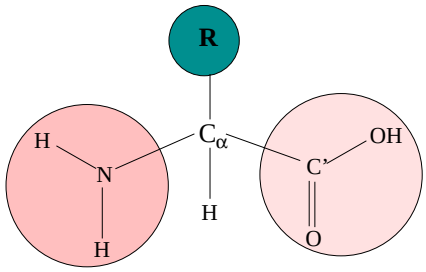

Figure 2.1: Schematic diagram of an amino acid

structures into an overall three-dimensional configuration (tertiary structure). It is believed that genomes encode the precise, three-dimensional shapes of thousands of proteins by using linear sequences. The underlying principle is still not fully understood. In this chapter, the basic units of protein structures will be examined.

2.1 Introduction to Protein Structure

An amino acid is the foundation on which a protein is built. An amino acid includes one central carbon atom (C_α), to which are bounded one amino group (NH_2), a carboxyl group ($COOH$), a hydrogen atom (H), and a side chain (R), as shown in Figure 2.1. What distinguishes one amino acid from another is the side chain. There are 20 different side chains that are commonly seen in proteins, resulting in 20 standard amino acids. Two amino acids can be joined by the formation of a peptide bond when the carboxyl group ($COOH$) of one amino acid condenses with the amino group of another amino acid. One water molecule is eliminated during the process. The formation of successive peptide bonds creates a main chain or backbone, to which various side chains are attached. Peptide bonds are depicted

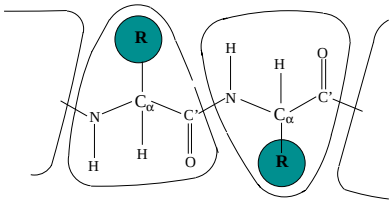

Figure 2.2: Peptide bond in a polypeptide chain

in Figure 2.2.

Protein structure can be broken down into four levels: primary structure, secondary structure, tertiary structure, and quaternary structure. An amino acid sequence is called the primary structure. The different regions of the amino acid sequence form secondary structures such as alpha helices or beta sheets. A tertiary structure is formed by folding these structural units into a compact globular unit. For a multi-chain protein, the chains are arranged in a quaternary structure.

One of the most important discoveries found about protein structure has been that the interior of a protein is hydrophobic. By packing hydrophobic side chains into the inside of a protein, it has a hydrophobic core and a hydrophilic surface. However, there is a problem when a protein chain folds into a hydrophobic core. To pack all the side chains into the core, burying the main chain under the surface cannot be avoided. The main chain is hydrophilic and has one hydrogen bond doner NH and one hydrogen bond acceptor $C' = O$ for each peptide. In a hydrophobic environment, the main chain peptide polarity must be neutralized by forming hydrogen bonds. This is achieved by arranging the main chain into regular secondary structures under the surface of the protein molecule. There are two primary types of secondary structures: alpha helices and beta sheets which are illustrated in Figure 2.3 and 2.4. Both have hydrogen bonds, connecting the NH

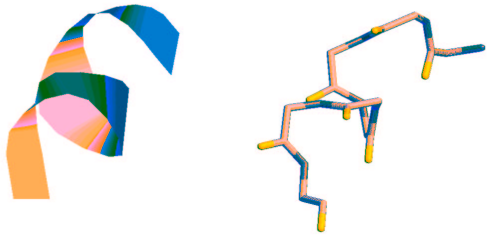

Figure 2.3: Backbone of one α helix in the $\alpha_2\beta_2$-hemoglobin, with two different representations, drawn by RasMol

and $COOH$ groups on the main chain. Secondary structures formed in this way are stable and comparatively inflexible.

An α helix is a classic element of a protein structure, and the most abundant type of secondary structures. It was first discovered in 1951 by Linus Pauling at the California Institute of Technology [9]. An α helix is characterized by the spiral conformation of a polypeptide chain. Within an α helix, the main-chain N and O atoms are connected by hydrogen bonds. An α helix has 3.6 residues per turn, which corresponds to $5.4\dot{A}$ with hydrogen bonds between the $C' = O$ group of residue n and the NH group of $n + 4$. The most common place to find an α helix is along the surface of a protein molecule, with one side of the helix contacting the solution and the other side contacting the hydrophobic core of the protein.

In contrast to an α helix, which is formed by one continuous segment of a polypeptide chain, a β sheet is composed of several nonadjacent regions of a polypeptide chain as shown in Figure 2.4 and Figure 2.5. These regions are usually called β strands, which are aligned adjacent to each other. It is the hydrogen bonds that

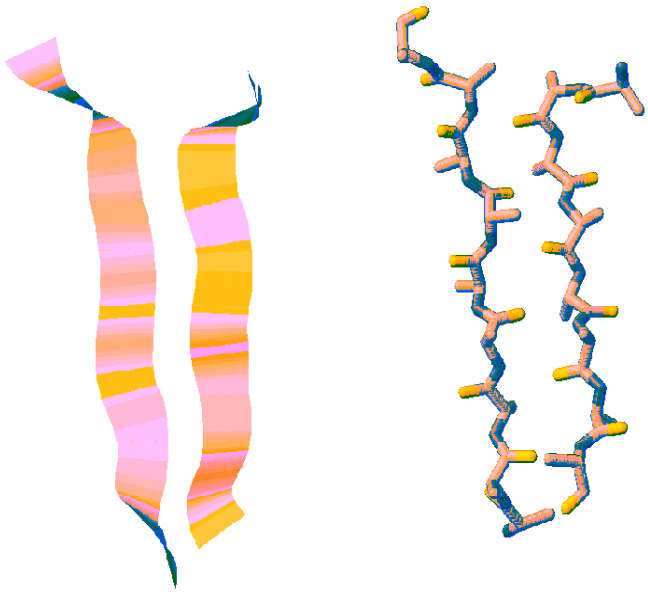

Figure 2.4: Backbone of antiparallel β sheet of ribonuclease with two different representations, drawn by RasMol

connect the $C' = O$ groups from one β strand and the NH groups from another β strand. Depending on the relative directions of two interacting beta strands, beta sheets are found in two forms: Antiparallel or Parallel.

The secondary structures in a protein structure compose the hydrophobic core of the molecule and are connected by loop regions of various lengths and shapes. The loop regions are at the surface of the protein molecule and exposed to the solvent. They can form hydrogen bonds with water molecules.

A tertiary structure is the overall three-dimensional structure of a polypeptide chain, including the overall arrangement of secondary structures and loops in a protein structure. A major driving force in determining the tertiary structures of

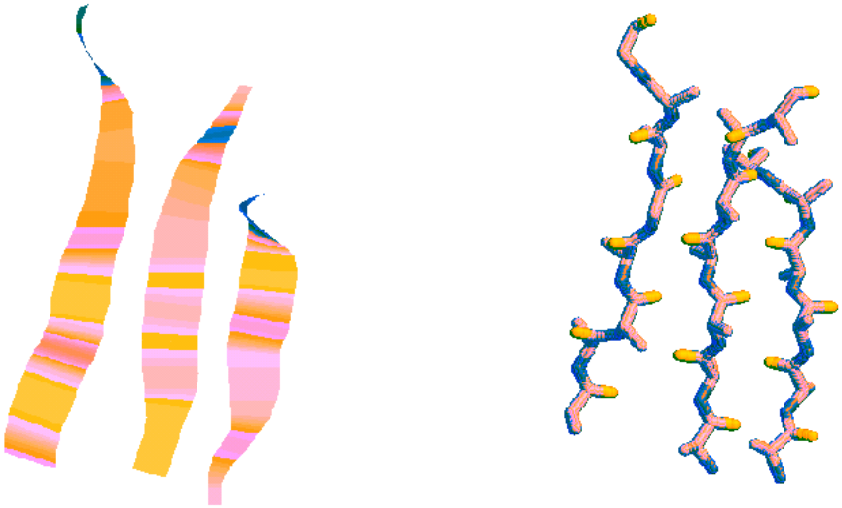

Figure 2.5: Backbone of three-strand parallel β sheet of flavodoxin with two different representations, drawn by RasMol

globular proteins is the hydrophobic effect. A polypeptide chain folds so that the side chains of the nonpolar amino acids are buried within the structure and the side chains of the polar residues are exposed on the outer surface. An example of a protein tertiary structure is represented in Figure 2.6.

The fundamental unit of a tertiary structure is a domain which is defined to be a polypeptide chain or part of a polypeptide chain that folds independently into a tertiary structure. Domains are also units of function. The different domains of a protein have different functions and one protein can have one or several domains.

Some proteins have multiple chains and form a quaternary structure, which consists of several identical polypeptide chains which function independently or cooperatively. An example of a quaternary structure is given in Figure 2.7.

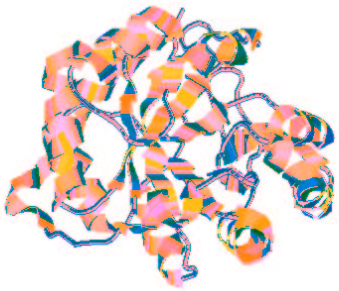

Figure 2.6: 3D structure of Indole-3-Glycerol Phosphate Synthase, drawn by RasMol.

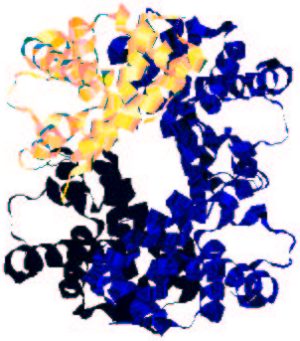

Figure 2.7: Quaternary structure of Hemoglobin Rothschild, drawn by RasMol.

2.2 Protein Structure Prediction

One of the challenges of the post-genomic era is to computationally predict the three-dimensional structures of proteins encoded in genome sequences. As a result of various sequencing and structural genomic projects, the full genomes for more than 100 organisms are known; for more than 60 of these, the data are publicly available and contribute about 250,000 protein sequences. The number of entirely sequenced genomes is expected to continue growing exponentially for several years. This explosion of sequence information has widened the gap between the number of the protein sequences deposited in public databases, e.g. Protein Data Bank [10], and the experimental characterization of the corresponding proteins.

For protein structure prediction, computational techniques and biological knowledge are applied to predict the tertiary structure of a protein, given its amino acid sequence. To achieve this, secondary structures and residue-residue contacts must be predicted first. Structural prediction is expected to be fast and accurate and to provide some insight into protein function and also facilitate the annotation of open reading frames (ORFs) or genes with unknown functions.

In recent years, automatic structure prediction has significantly progressed. A large number of fully automated structural prediction servers have been developed, and are available to the research community. The servers cover various prospectives of structural prediction. To promote the research in this area, some community-wide experiments have been carried out. The biennial Critical Assessment of Structure Prediction (CASP) provides a great opportunity for establishing the current state of the art in protein structure prediction, reflecting the progress made during the intervening two years and indicating where future efforts should be made. From 1994 to 2004, six CASP experiments have been completed [11, 12, 13]. In the Crit-

ical Assessment of Fully Automated Structure Prediction (CAFASP) experiments, the performance of fully automated servers for structure prediction is assessed by blind tests. Since the first CAFASP at CASP3, more and more fully automatic web servers have been developed, and great progress has been made in automatic protein structure prediction. In CAFASP3 (2002), the number of participants almost doubled compared to the number in CAFASP2 [3, 7].

The methodologies used in automatic servers can be divided into three general strategies: ab initio, comparative modeling, and fold recognition. In the following sections, we will go though the pros and cons of these three categories.

2.3 Ab Initio Folding

Ab initio folding methods build the 3D structure of a protein from its sequence without using any templates. Ab initio protein folding has traditionally been an area of purely academic interest with relatively slow progress.

Generally, it is assumed that a protein folds to a global minimum-energy conformation. In order to find such a conformation, researchers simulate protein folding by doing standard molecular dynamic simulation with a physically reasonable potential function. This approach has a long history and is still popular but one obvious problem is that it is computationally expensive. In addition, due to the inadequacies of current potential functions, the likehood that a native state will be found at a global minimum-energy conformation is significantly reduced [14].

Another approach to do ab initio folding is direct conformation space search. For this approach, the successful prediction of the native structure of a protein requires both an efficient sampling of the conformational space, and an energy

function that recognizes the native conformation as the lowest in energy. However, exhaustive conformation space search is still formidable due to current computing speeds. To deal with that, researchers have attempted to reduce the search space by simplifying models or restraining the conformation space.

It has been observed that ab initio folding cannot perform consistently for all classes of proteins. In fact, ab initio folding totally fails for proteins longer than 150 residues [14]. In spite of that, for short proteins that do not have structural templates and significant homology, ab initio folding is a valid solution.

2.4 Comparative Modeling

To date, the most successful methods for protein structure prediction are homology-based comparative modeling and fold recognition. The former exploits the evolutionary relationships between proteins and produces structural models of unknown proteins by using the known structures of their homologues as templates. The underlying premise for comparative modeling is that if a set of proteins are homologous, then their three-dimensional structures are more conserved than their primary sequences.

Based on this, given a target sequence, first, its homologous proteins (templates) are found from a structure database by doing pair-wise sequence alignments. Some computer tools such as PSI-BLAST can be used. Then, a multiple sequence alignment is built from the target sequence and the templates. The most conserved segments in the multiple sequence alignment are identified. After the conserved regions of the target are modeled, the structurally divergent regions are modeled. Next, the loop conformations are assigned and then, the structural model is refined.

The quality of the generated models depends on the extent of the structural conservation between the target and the templates, the servers' ability to identify homologous templates, the quality of the sequence-structure alignments and the ability to predict the conformation of loop regions and nonconserved regions. Comparative modeling is the most reliable structure prediction method on the basis of known three-dimensional structures. The CASP evaluation results show that the models, obtained with comparative modeling, are usually sufficiently accurate for designing experiments, because the biologically important regions of a structure are typically more accurately modeled than the rest of the structure. In spite of that, when there is no significant homology or only distant homology found for a given protein sequence, comparative modeling fails or gives partially accurate 3D structural models [15]. FFAS03 [16], SUPERF_PP [17], 3D-JIGSAW [18], and PDB-BLAST [19] are good examples of comparative modeling servers.

2.5 Fold Recognition

Fold recognition methods are for those targets which do not have significant homology with any known structure, but have the same fold as some known structures. Unlike sequence-only comparisons, these methods take advantage of the extra information that is available from 3D structure information. Protein threading is based on two observations. One is that the number of different folds in nature is fairly small, and the other is that, according to the statistics of the PDB, 90% of the new structures, submitted to the PDB from 2001 to 2003, have structural folds that are similar to the ones in the PDB [10]. Thus, given a protein sequence, it is likely that a fold can be found from which to build its three-dimensional structure.

By using the statistical knowledge of the relationship between structures and sequences, fold recognition methods predict the 3D structure of a given protein sequence. For that, a structure template database must be constructed first. Also, a scoring function is required to measure the fitness of a sequence and a structure template in an alignment, based on the known relationships between sequences and structures. After the target sequence is aligned to each template structure in the structure template database by optimizing the scoring function, the best-fit template is identified. The structural model of the target sequence is then built from the best alignment of the target sequence with the selected template.

The completeness of the template database used influences the performance of fold recognition greatly. In addition, when a target belongs to a new fold, fold recognition methods cannot produce a reliable prediction [6]. Some well developed fold recognition servers are: RAPTOR [20, 21], FFAS04, 3D-PSSM [18], FUGUE3 [22], PROSPECT [23], Sam-T02 [24], among others.

2.6 Consensus

From the analysis of the proceeding methods, it can be seen that each of three categories of methods has its advantages and disadvantages. No single method is consistently effective for all the classes of targets, as has also been observed in experiments [7]. It is noteworthy that these methods are complementary to each other, since different methods are effective for different types of targets. If different methodologies are combined by using consensus algorithms, we can build a meta server with a more reliable prediction and more stable performance.

2.7 Support Vector Machine Regression (SVMR)

Approximating a real-valued function from a finite set of samples is the linchpin in many areas. Commonly used techniques for such tasks are linear regression, or logistic regression, but they are often not sufficient to approximate complex functions with high nonlinearity. In such cases, nonlinear regression methods should be adopted to improve approximation accuracy. For our application, we use Support Vector Machine Regression (SVMR) to approximate the functional relationship between the features of a model and its structural quality.

Developed by Vapnik et al. in the 1970s, the SVM is a set of supervised learning methods, applicable to both classification and regression [25]. First, let us examine SVM classifiers. For a training task and a set of training data, the machine learner attempts to achieve the best generalization performance. A machine learner that is over-trained with a training set remembers too much detail from the training set and a machine that is not well-trained cannot capture enough features of the training set. Neither generalizes well. A machine learner must seek a balance between the accuracy attained on that particular set of training data and the capacity of the machine learner, that is, the ability of the machine to learn a training set without errors. The concept has been mathematically formulated into SVM models.

Let us look at a simple case of SVM classifiers: a linear SVM is trained on separable data. Given training data $\{x_i, y_i\}, i = 1, \cdots, n$, $y_i \in \{-1, 1\}$, $x_i \in R^m$, suppose there is a hyperplane that separates the positive from negative examples. The points x on the hyperplane satisfies $w \cdot x + b = 0$, where w is normal to the hyperplane. $|b|/\|w\|$ is the perpendicular distance from the hyperplane to the origin, and $\|w\|$ is the Euclidean norm of w. Let $d_+(d_-)$ be the shortest distance from the separating hyperplane to the closest positive (negative) example. Define

the "margin" of the hyperplane to be $d_+ + d_-$. For the linear separable case, the SVM algorithm simply looks for a separating hyperplane with the maximum margin. The goal of maximizing the margin is motivated by attempts to bound generalization error. The analysis of the boundaries of the constraints shows that when $\|w\|$ is minimized, the margin is maximized. This can be formulated into an optimization problem as follows.

$$\begin{array}{ll} \text{Minimize} & \frac{1}{2}\|w\|^2 \\ \text{Subject to} & \begin{cases} w \cdot x_i + b \geq +1 & \text{for} \quad y_i = +1 \\ w \cdot x_i + b \leq -1 & \text{for} \quad y_i = -1 \end{cases} \end{array} \tag{2.1}$$

The above formulae can be extended to accommodate nonseparable cases and they can also be transformed into a nonlinear form. SVM classifiers turn out to have excellent generalization performance and are successfully applied in many areas such as pattern recognition and information retrieval [26]. For fold recognition, SVM classifiers are first used by RAPTOR, a state-of-the-art fold recognition server developed by Jinbo Xu et al. [20, 21]. After a target sequence has been threaded to each structure template in RAPTOR's template database, features are extracted from each sequence-structure alignment and an SVM classifier is trained to select the best template. It is shown that the SVM method has great advantage over the conventional Z-score method. RAPTOR participated in CAFASP3 and was ranked 1st among the individual servers [7].

When the SVM is applied in regression and time series prediction applications, it also exhibits excellent performance [27]. The model, produced by the Support Vector Machine classification, depends on only a subset of training data, because the cost function for building the model ignores training points that lie within the margin. Similarly, a model produced by the Support Vector Machine Regression (depends on only a subset of training data, because the cost function for building

the model ignores any training data that is close (within a threshold ϵ) to the model prediction. We will start with the linear SVM regression which is simple and straightforward.

Linear SVM Regression For the training data $\{x_i, y_i\}$, $i = 1, \cdots, n$, in our application, $x_i \in R^m$ is a model feature vector and $y_i \in R^1$ is a model quality score. In ϵ-SV regression, our goal is to look for a function $f(x) = w \cdot x + b$ that has, at most, ϵ deviation from the actual obtained y_i for all the training data points. Here, w is a vector in R^m and $(\cdot)$ represents inner product. For the SVM regression, w plays a role similar to that in the SVM classification. By minimizing $\|w\|$, the width of the tube around the output curve of the approximating function is maximized. Mathematically, this is formulated as a convex optimization problem [26]:

$$\begin{array}{ll} \text{Minimize} & \frac{1}{2}\|w\|^2 \\ \text{Subject to} & \begin{cases} y_i - w \cdot x_i - b \leq \epsilon \\ w \cdot x_i + b - y_i \leq \epsilon \end{cases} \end{array} \tag{2.2}$$

The assumption in (2.2) is that there exists such a function f that approximates all the pairs (x_i, y_i) with ϵ precision; that is, the convex optimization problem is feasible. Sometimes, however, this can not be guaranteed. Therefore, slack variables ξ_i, ξ_i^* are introduced to cope with, otherwise, infeasible solutions of the optimization problem (2.2). Thus,

$$\begin{array}{ll} \text{Minimize} & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ \text{Subject to} & \begin{cases} y_i - w \cdot x_i - b \leq \epsilon + \xi_i \\ w \cdot x_i + b - y_i \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \end{array} \tag{2.3}$$

where C penalizes the amount to which deviations that are larger than ϵ are tolerated.

By introducing Lagrange multipliers λ_i and λ_i^* $(i = 1, \cdots, l)$, we can construct a Lagrange function from the objective function and the corresponding constraints,

$$\begin{aligned} L &= \|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) - \sum_{i=1}^{l}\lambda_i(\epsilon + \xi_i - y_i + w \cdot x_i + b) \\ &\quad - \sum_{i=1}^{l}\lambda_i^*(\epsilon + \xi_i^* + y_i - w \cdot x_i - b) - \sum_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\xi_i^*) \end{aligned} \tag{2.4}$$

where $\lambda_i, \lambda_i^*, \eta_i, \eta_i^* \geq 0$.

According to the Karush-Kuhn-Tucker (KKT) condition, a dual formulation (2.5) of the original optimization problem, (2.2), can be obtained from the Lagrange function,

$$\begin{aligned} \text{Maximize} \quad & L_D \equiv -\tfrac{1}{2}\textstyle\sum_{i,j=1}^{l}(\lambda_i - \lambda_i^*)(\lambda_j - \lambda_j^*)(x_i \cdot x_j) \\ & \qquad -\epsilon\textstyle\sum_{i=1}^{l}(\lambda_i + \lambda_i^*) + \sum_{i=1}^{l} y_i(\lambda_i - \lambda_i^*) \\ \text{Subject to} \quad & \begin{cases} \sum_{i=1}^{l}(\lambda_i - \lambda_i^*) = 0 \\ \lambda_i, \lambda_i^* \in [0, C] \end{cases} \end{aligned} \tag{2.5}$$

where C penalizes the amount to which deviations that are larger than ϵ are tolerated.

By solving (2.5), we have:

$$\begin{aligned} w &= \sum_{i=1}^{l}(\lambda_i - \lambda_i^*)x_i \\ f(x) &= \sum_{i=1}^{l}(\lambda_i - \lambda_i^*)(x_i \cdot x) + b \end{aligned} \tag{2.6}$$

In an optimal solution, the training data for which $\lambda_i > 0$ are called support vectors (SVs). Note that w is a linear combination of the SVs x_i. In a sense, the complexity of a function's representation by SVs is independent of the dimensionality of the input space R^m, and depends only on the number of the SVs. Also, the

SVs appear only in the form of a dot product with x in the trained SVM machine. When calculating $f(x)$, we need not compute w explicitly. The dual formulation provides the key for extending the SVMR to nonlinear functions.

Nonlinear SVM Regression To make an SVMR nonlinear, a straightforward way is to map x_i to a higher dimension space and then apply the standard SVMR algorithm. Now, the SVMR model has the following form:

$$\begin{array}{ll} \text{Maximize} & \begin{array}{rl} L_D \equiv & -\frac{1}{2}\sum_{i,j=1}^{l}(\lambda_i-\lambda_i^*)(\lambda_j-\lambda_j^*)(\phi(x_i)\cdot\phi(x_j)) \\ & -\epsilon\sum_{i=1}^{l}(\lambda_i+\lambda_i^*)+\sum_{i=1}^{l}y_i(\lambda_i-\lambda_i^*) \end{array} \\ \text{Subject to} & \begin{cases} \sum_{i=1}^{l}(\lambda_i-\lambda_i^*)=0 \\ \lambda_i,\lambda_i^*\in[0,C] \end{cases} \end{array} \tag{2.7}$$

The drawback is that the SVM in this form can easily become computationally prohibitive when the dimension is high. A less expensive way to achieve this is to make an implicit mapping via a kernel function. Instead of defining $\phi(\cdot)$ explicitly, an implicit mapping is determined by defining $\kappa(x, x_i) = \phi(x) \cdot \phi(x_i)$, directly, without knowing $\phi(\cdot)$ [26]. $\kappa(x, x_i)$ is called the kernel function, and allows us to rewrite the SVMR model as follows:

$$\begin{array}{ll} \text{Maximize} & \begin{array}{rl} L_D \equiv & -\frac{1}{2}\sum_{i,j=1}^{l}(\lambda_i-\lambda_i^*)(\lambda_j-\lambda_j^*)\kappa(x_i,x_j) \\ & -\epsilon\sum_{i=1}^{l}(\lambda_i+\lambda_i^*)+\sum_{i=1}^{l}y_i(\lambda_i-\lambda_i^*) \end{array} \\ \text{Subject to} & \begin{cases} \sum_{i=1}^{l}(\lambda_i-\lambda_i^*)=0 \\ \lambda_i,\lambda_i^*\in[0,C] \end{cases} \end{array} \tag{2.8}$$

The expansion of the function $f(\cdot)$ in this case is:

$$f(x) = \sum_{i=1}^{l}(\lambda_i-\lambda_i^*)\kappa(x,x_i)+b \tag{2.9}$$

There is no universally-accepted rule to select κ. It is problem-specific. Typically, a kernel function must satisfy the Mercer's conditions to guarantee that the quadratic

program is convex [26]:

(1) $\kappa(x, x) = 0$

(2) $\kappa(x, y) = \kappa(y, x) = 0$

(3) $\kappa(x, y) + \kappa(y, z) \geq \kappa(x, z)$

Some commonly seen kernel functions are linear kernels, polynomial kernels, Radial Basis Function (RBF) kernels and multi-layer perception kernels [26]:

$$\begin{aligned} \kappa(x_1, x_2) &= cx_1x_2 + c_0 \\ \kappa(x_1, x_2) &= (x_1x_2 + 1)^d \\ \kappa(x_1, x_2) &= exp(-\|x_1 - x_2\|^2/2\sigma^2) \\ \kappa(x_1, x_2) &= tanh(ax_1x_2 + c) \end{aligned} \tag{2.10}$$

Chapter 3

Survey of Consensus Prediction

Before consensus was applied to protein tertiary structure prediction, it had already been used in other areas, including multiple classifier aggregation, the optimization of database queries, and data fusion. Not surprisingly, consensus has also been used extensively in bioinformatics applications, such as genome assembly and protein secondary structure prodiction.

3.1 Introduction

In the CAFASP experiments, it has been observed that for different targets, the best predictions are often made by different servers. No single server can reliably generate the best models for all the targets [3, 7]. In CAFASP, each server submits its top ten models and only the top model is taken into consideration for the performance evaluation. Often, the best quality model is not the top one in the submitted

ranking list. Even for the same fold recognition server, setting different values for parameters in the scoring function can lead to different templates and models [5]. Consequently a practical question is whether different methods or scoring functions can be combined for better predictions.

Consensus fold recognition was first applied in some individual servers rather than meta servers. INBGU [28] utilized five different scoring functions to scan its fold library, respectively, which in fact was composed of five component servers. By combining the scores and ranks, obtained with different scoring functions for the same fold in the library, a more sensitive score was obtained for each fold and the best possible fold was identified. 3D-PSSM [18] threaded a query sequence to each entry in its template database three times by using a different position specific matrix each time. Only the maximum of three obtained scores was used as the final score. When INBGU and 3D-PSSM were evaluated in CAFASP3, they performed very well [7]. However, only the simplest consensus methods such as averaging or selecting the maximum were employed by these servers.

The idea of combining the outputs of individual servers was first successfully applied in CASP4 by a human group, the CAFASP-CONSENSUS. It can be viewed as a human meta server. The four human experts derived predictions by inspecting and analyzing the outputs from the automated fold recognition servers running in the parallel CAFASP2. It turned out that the CAFASP-CONSENSUS outperformed all the individual servers in the CAFASP2 and ranked seventh among the human predictors in CASP4 [12]. This, however, requires a great deal of human intervention: the human predictors must gather the input models from the individual servers, and use software to evaluate the models, select the best model from the collected inputs, improve the model manually, or determine whether or not correct predictions can be obtained. One strategy is to apply a number of inde-

pendent servers to arrive at a prediction from the top ranking predictions. It is desirable to automate these procedures and relieve human predictors from the tedious tasks. This led to the development of the first automated consensus server, Pcon [29], which turned out to perform better than any individual server, especially in specificity [6].

Meta servers can be divided into two categories, based on the strategies to obtain a prediction: the selector and the assembler. A selector meta server selects one model from the collected input models by some algorithm and reports the selected model as the output. As mentioned, the first model in the ranking list, submitted by a server, may not be the best model in the list. Selector meta servers attempt to identify the best model from the ranking list. One possible approach is to predict the quality of each of the input models, and based on this quality an output is selected, as we will see in the following examples. Assembler meta servers go beyond the selection of models. A hybrid model is assembled by combining the structural fragments of collected input models. This approach is believed to be more sensitive and produce more accurate models [30]. In the following section, we will review some meta servers of both types.

3.2 Selector-Type Meta Servers

Following the success of the CAFASP-CONSENSUS group in CASP4, the first automatic consensus server, named Pcon, was constructed by Lundstrom et al. in 2001 [29]. Pcon attempts to reproduce the consensus procedures that were followed by the human predictors, and uses neural networks to predict the quality of input models. Pcon receives, as inputs, the top ten models from each of its

six component servers (GenTHREADER, SamT98, INBGU, FFAS, 3D-PSSM and pdbBLAST). Each input model has a confidence score, which is reported by one of the component servers and is used as one feature to be fed to the neural networks. To obtain other features, each input model is compared with other models by using a structure superimposition algorithm. Based on the confidence scores and the structural similarities of the input models, the quality of each model is predicted by the neural networks. Because the confidence scores are specific to each component server and do not have same scale and distribution, an individual neural network is required for each component server. There are six neural networks in Pcon. The final prediction is based on the outputs of the six networks. If several servers predict a particular fold, Pcon will assign a high score to it. It is easy to add a new server to Pcon with this configuration. Pcon turns out to perform better than any of its component servers, especially in specificity [6]. A newer version of the meta server, derived from Pcon, is Pmodeler, which predicts the quality of a model by combining the output scores of Pcon and ProQ. ProQ is a neural network-based tool that predicts the quality of a protein model from the intrinsic structural parameters computed from the model. With ProQ, a small but significant improvement is observed [31].

Since the development of the first meta server, several new meta servers have been proposed. One of them is the 3D-Jury system, which was developed by Rychlewski et al. [32]. 3D-Jury selects its output from the input models by using different scoring schemes. Unlike Pcon, 3D-Jury does not use machine learning techniques; thus, no training procedure is required, which makes 3D-Jury simple and flexible. The 3D-Jury compares the input models with each other by using MaxSub, and a similarity score is obtained for each pair of models. Then, a score for each model is calculated as the prediction of the model quality, based on the pair-wise Max-

Sub scores. The 3D-Jury system can be operated in two modes whose scores are calculated in different ways. In the best-model-mode (3D-Jury-Single), only one model from each server is used to calculate a prediction. In the all-model-mode (3D-Jury-All), all the models from each server are used. The scores are calculated by using the follow formulae:

$$3D - Jury - all(M_{a,b}) = \frac{1}{1+Nn} \sum_{i=1, i\neq a}^{N} \sum_{j=1}^{n} sim(M_{a,b}, M_{i,j}) \tag{3.1}$$

$$3D - Jury - single(M_{a,b}) = \frac{1}{1+N} \sum_{i=1, i\neq a}^{N} \max_{j=1}^{n} \{sim(M_{a,b}, M_{i,j})\} \tag{3.2}$$

where

$sim(M_{a,b}, M_{i,j})$: the similarity score between model $M_{a,b}$ and $M_{i,j}$,

$M_{a,b}$: the bth model from server a,

$M_{i,j}$: the jth model from server i,

N: the total number of servers,

n: the number of top ranking models reported by server i (maximum ten).

3D-Jury was evaluated in the LiveBench 6 program. It is shown that 3D-Jury has a high sensitivity and specificity for both easy and hard targets [6].

3.3 Assembler-Type Meta Servers

Some meta servers can assemble a new model from the structural fragments of input models. 3D-SHOTGUN was the first meta server that is capable of assembling a new hybrid model from input models [33]. It incorporates some strategies

that human predictors have successfully applied. By comparing the input structure models, the recurrent structural similarities are identified, from which hybrid models are assembled. Scores are assigned to the hybrid models by combining the original model scores and structural similarities among them. It is believed that these two steps render the new server more sensitive and specific than any of its component servers. Two new meta servers (3DS3 and 3DS5) have been developed by using the 3D-SHOTGUN method. One has three component servers and the other has five component servers. In CAFASP3, both servers were ranked among the three most sensitive and specific meta servers [7]. In spite of the success with this approach, it has been observed that the hybrid models, assembled by the 3D-SHOTGUN method, contain a number of nonnative-like structural fragments due to the assembly procedure. This is due to the fact that the assembly is residue-based. Therefore, an automatic refinement method is being developed.

Robetta is a meta server that is unlike either of the previous two types. Robetta was originally an ab initio server that built models by using a fragment insertion protocol without utilizing any template or any homology information. Later, Robetta was upgraded so that if a template is found for a given target, the template-based approach is used to build a structural model; if there is no template, the fragment insertion method is employed to build the model. The fragment insertion method is also used in the context of the template-based approach for loop regions. Each input protein sequence is classified as a de novo or template-based target by using PSI-BLAST and Pcon2. After the classification, an appropriate method is applied to build a model [34]. In this sense, Robetta is a meta server that is different from the previous two types of meta servers. Rosetta was evaluated in CASP3, CASP4 and CASP5, and turned out to be very successful [11, 12, 13]. In CAFASP3, it was ranked first among meta servers [7].

3.4 Difference between Consensus Server and Individual Servers

The inputs of meta servers are the outputs of their component servers. Therefore, the superior performance of meta servers is dependent on the performance of the component servers. The quality of input models significantly influences the outputs of meta servers. There are three cases, depending on the inputs: The first exists if most of the component servers make good predictions; For the second, none of the component servers attains any significant prediction; The third case occurs when only a few component servers make good predictions. In the first case, it is easy for meta servers to make an equivalently good prediction or an even better prediction. In the second case, it is almost impossible for meta servers to come up with an improved prediction. The third case exists where consensus can make a difference. It is possible for meta servers to differentiate between good predictions and poor predictions in this case by using consensus algorithms. Thus, when most or part of the component servers make good predictions, meta servers can make significantly improved predictions.

To achieve the best performance for a meta server, it is necessary to include individual servers with the best performance. However, it is not enough to include as many good-performance individual servers as possible. The combinations of component servers are also important, and the included component servers should be complementary to each other to guarantee stable performance for all the classes of targets. This topic has not been researched previously.

Chapter 4

New Meta Server : Alignment by Consensus Estimation (ACE)

Selector-type meta servers have a high sensitivity and specificity as proven by Pcon and 3D-Jury. Here, we want to develop a selector meta server to combine multiple individual servers. Input models of the meta server are the outputs generated by the component servers with various scoring functions. Therefore, it is not practical to attempt to design a more effective scoring function as a direct prediction of model quality, which, otherwise, can be used in an individual server. Some researchers attempt to compute intrinsic structure features (stereochemical parameters) from a model and predict model quality by using machine learning-based approaches. One example is ProQ which computes structural parameters of a model from its 3D coordinates, and then use neural networks to predict the MaxSub score of the model from the computed structural parameters [35]. However, the performance

of ProQ is not satisfactory. Another way to predict model quality is to compare a model with other models by using some structure comparison tool to obtain a score as the quality prediction of the model. This is how 3D-Jury predicts model quality. Pcon has gone one step further and uses neural networks to predict model quality. After some simple features are extracted from a model, the confidence score of the model and the features are fed to the neural networks to predict the model quality. Our goal is to extract effective features through structural comparisons as 3D-Jury did, and use more advanced machine learning techniques to predict model quality by which to select an output.

In this chapter, we will introduce a new meta server, Alignment by Consensus Estimation (ACE), a selector-type meta server that extracts the features from a model through structural comparisons. The Support Vector Machine Regression (SVMR) is adopted to train a model and predict the quality of a structure model. We will explore the techniques used by ACE in detail in this chapter.

4.1 Feature Extraction

Features are crucial for machine learning-based meta predictors, and influence the performance of meta servers significantly. First, let us look at what features Pcon has used. Pcon uses both the reported confidence score and the number of similar models. Specifically, it compares a model with all other models and counts the number of models whose similarity scores are over a specific cutoff. Pcon also extracts the same types of features from the templates of the reported models by doing structure comparisons. The structure comparison tool used in Pcon is LGscore which is alignment independent. For the 3D-Jury system, the 3D-Jury-All

and 3D-Jury-Single scores are very efficient and good candidates for our features.

For our meta server, all the features are extracted from the structural comparisons of input models, similar to 3D-Jury. The confidence scores reported by component servers are not used as features. This simplifies the complexity of the meta server and training process because there is no need to train an SVMR model for each component server and set up a jury mechanism. We also do not use the templates of input models to extract features because experimental results indicate that features obtained from the templates by structural comparisons are not effective. Using ineffective features can introduce noise to a meta server and degrade the performance. Next the features used in ACE will be presented.

After the top ten models, reported by each component server, are collected, all the models are compared with each other, and a similarity score is obtained for each pair, representing the structural similarity between the two models. MaxSub has been used to structurally compare two models. The quality score of a model is calculated by MaxSub as well, and serves as the objective function of the SVM regression. MaxSub is a program originally developed to measure the quality of a single model by structurally comparing the model with the corresponding real structure obtained experimentally. MaxSub is a sequence-dependent quality assessment tool that identifies the maximum superimposable (within $3.5\dot{A}$) subset of C_α atoms of a model and an experimental structure. As a result, a normalized score respresenting the quality of the model is produced. MaxSub can also be used to measure the structural similarity between two models [36]. It is the official tool used in the LiveBench and CAFASP tests [4, 7]. The features used in ACE are defined as follows:

$sim(M_{a,b}, M_{i,j})$: the MaxSub similarity score between model $M_{a,b}$ and $M_{i,j}$
$M_{a,b}$: the bth model reported by server a

$M_{i,j}$: the jth model reported by server i

N: the total number of servers

n: the total number of top models reported by each server

Feature 1 is specific for each model. First, compare model $M_{a,b}$ with all the models, reported by the other servers and then calculate the average score as follows:

$$\frac{1}{(N-1)n} \sum_{i=1, i\neq a}^{N} \sum_{j=1}^{n} sim(M_{a,b}, M_{i,j}) \tag{4.1}$$

Feature 2 is also specific for each model. First, compare model $M_{a,b}$ with all the models from one of the other servers respectively, and then select the maximum score. Repeat this procedure for each of the other servers. Finally, calculate the average of the selected maximum scores. Mathematically,

$$\frac{1}{N-1} \sum_{i=1, i\neq a}^{N} \max_{j=1}^{n} \{sim(M_{a,b}, M_{i,j})\} \tag{4.2}$$

Feature 3 is different from feature 1 and 2 and is composed of a set of subfeatures rather than one feature. In addition, feature 3 is not model-specific, but target-specific, that is, for each target protein, feature 3 is different and all the models predicted for the same target protein have the same feature 3. To obtain feature 3, the similarity between the predictions made by every two servers must be calculated. Thus, for N servers, there are C_N^2 subfeatures. The disadvantage is that if N is very large, C_N^2 grows quickly. Later, we will demonstrate that a large N may not be a good choice for the SVMR model. For server a and i, the similarity between the predictions made by them is represented by:

$$\frac{1}{n^2} \sum_{b=1}^{n} \sum_{j=1}^{n} sim(M_{a,b}, M_{i,j}) \tag{4.3}$$

Feature 1 is viewed as model-level voting and a rough measure of model quality. Also, we assume that the most accurate model has more similarities with the

other models than a less accurate model does. Therefore, one possible approach to estimate the quality of a model is to compare it with all the models from the other servers, which is the basis for feature 1. Note that feature 1 is very similar to 3D-Jury-Single (3.1). The difference between (4.1) and (3.1) is that in (4.1), the models from the same server as $M_{a,b}$ are ignored. The reason for this is that it has been observed in experiments that the models, reported by the same server, are more likely to be similar to each other even if all of them are poorly predicted. So, including the models from the same server in the sum introduces bias and degrades the effectiveness of the proposed feature.

Feature 2 is server-level voting and a different way to estimate the quality of a model though structural comparisons. Unlike feature 1, feature 2 uses only one model from each server. Feature 2 is very similar to (3.2). The difference between (3.2) and (4.2) is that in (4.2) the maximum score obtained from the same server is ignored. The reason is similar to that for feature 1. Because feature 2 uses only one model from each server for calculating the sum, it is not as stable as feature 1 but more sensitive.

Feature 1 and 2 are principal and feature 3 is an auxiliary feature. Feature 3 represents the similarity between the two sets of structural predictions made by any two servers for a particular target protein. This can help in some cases to estimate the performance of different servers with respect to the same target protein, which can indirectly help to differentiate between models in some cases. For instance, suppose there are three servers, a, b and c. Assume at any time, the majority of the servers make correct predictions. If we know that for a particular target, servers a and b have similar predictions, but servers b and c, and servers c and a do not, then it is possible that server c makes poor predictions for this target. Thus, feature 3 does facilitate the estimation of model quality in some cases. Note that all these

features are calculated by averaging many similarity scores; that is, these features are obtained from raw scores in a statistical way. In this sense, if more models are involved in structural comparisons, the variations of the proposed features can be reduced and the performance can be improved. That is why all the top ten models from each server are utilized.

Because structural comparisons can be conducted in different ways, different features can be extracted accordingly. For example, we can compare one model with the top model from each of the other servers, or compare each model with all the models from each of the other servers. In addition, some other features can be derived from sequence alignments such as gap numbers, gap length, target length, and template length. It is also possible to generate some features by using software such as PROCHECK and WHATIF to measure the stereochemical quality of structural models. The software can calculate some structural parameters from the coordinates of a protein structure, such as torsion angles, hydrogen bond energies, all of which have been shown to correlate with model quality. The distributions of these parameters are calculated from the solved protein structures stored in the PDB so that some staticstics about a model's stereochemical quality can be calculated to provide a simple guide to the reliability of the structure [37]. A similar idea has also been implemented in ProQ by using machine learning techniques to predict the quality of a model. ProQ uses neural networks to predict the MaxSub score or LGscore of a protein model according to the computed intrinsic parameters of the model such as atom-atom contact, solvent accessibility surfaces [35].

In spite of the abundance of available features, not all the features are equally effective. When ineffective features are used to train the SVM, they can introduce noise and degrade the performance significantly. The proposed features are tested and refined through a trial-and-error process. We explored different combinations

of the features mentioned above and eventually arrived at the features that are currently used in ACE.

4.2 ACE Implementation

ACE is particularly designed to participate in large-scale evaluations. The assumption is that the results of individual servers are available for download from some website. In CAFASP3 [7], individual servers had 48 hours to make a prediction. and meta servers were allowed 96 hours to make a prediction. And all the submissions were publicly available on the CAFASP website. Consequently meta servers downloaded all the required results of individual servers from the website and carried out their consensus algorithms.

After a target is given, the first thing ACE needs to do is to download all the structure models submitted by its component servers from some website, together with alignment files. GNU *wget* is used by ACE to download data from the internet. *wget* is a free software package for retrieving data using HTTP, HTTPS and FTP, the most widely-used internet protocols. It is a non-interactive commandline tool, so it may easily be launched from the commandline. For training, the PDB files of all the targets must be downloaded from the PDB website by using *wget* as well.

The second step is to extract features through structure comparisons. MaxSub is used in ACE to do structure comparisons. Because MaxSub is not very stable, to avoid the program from being suspended, MaxSub is launched as a child process of the main process and timed. If MaxSub is still not terminated after some time threshold, it is killed by the main process.

For the CAFASP and LiveBench data sets, all the models fed to MaxSub are

Figure 4.1: Top model reported by RAPTOR for target T0196 in CASP6

preprocessed ones. That is, only the modeled regions of a structure are compared by MaxSub. The nonmodeled regions are removed from the structure. The advantage is that nonmodeled parts can have different shapes, which will influence the superimpositions obtained with MaxSub. For instance, if *modeler* is used to build a 3D model from an alignment, nonmodeled regions will be modeled as sticks by default and they will stick out from the molecular surface as shown in Figure 4.1. If the nonmodeled regions are not removed, they will cause large RMSD.

In the training stage, the quality of a model (its MaxSub score) is obtained

by comparing the model to the real structure with MaxSub. The real structure is generally one chain of a protein. So the coordinates of the chain must be extracted from the downloaded PDB file which generally stores the coordinates of multiple chains of a protein. Otherwise, the MaxSub score obtained is not correct.

After the features of all the models are obtained, the next step is to preprocess the data. The purpose is to let the SVMR model achieve the best performance. In the training stage, the mean and standard error of each feature are calculated and all the features are normalized. These means and standard errors are then stored. In the testing stage, the features in the testing data are normalized with respect to the means and standard errors of the training data.

For training, the preprocessed features and the MaxSub scores of models are fed to the SVMR. Then, after training, the trained model is stored in a file. For testing, the preprocessed features are fed to the trained SVMR model and an output is generated for each model.

The last step is to rank the SVMR outputs for all the models and select the model with the maximum SVMR output as the prediction made by ACE.

The operation of ACE can be represented by the flowchart in Figure 4.2

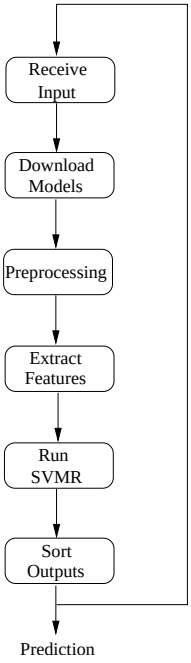

Figure 4.2: How ACE makes a prediction.

Chapter 5

Experimental Results and Conclusions

In the proceeding chapter, we have introduced the SVMR and the features that are extracted by ACE. In this chapter, we will investigate some experimental results on the performance of the proposed consensus algorithm in terms of sensitivity and specificity. For this purpose, we will use the publicly available LiveBench data sets. Besides comparing the performance of ACE with that of its component servers, we will also compare ACE with 3D-Jury and Pcon.

5.1 LiveBench Tests

The LiveBench project [38] is a continuous benchmarking program. Each week selected new PDB proteins are submitted to participating fold recognition servers,

and the results are collected and evaluated by using automated model assessment programs. The LiveBench program has two principal goals:

1. The program provides a simple evaluation of structure prediction servers from the point of view of a potential user. The evaluation of the sensitivity and specificity of available servers can help a user to develop sequence analysis strategies and to assess the confidence of obtained predictions.

2. The program offers a simple weekly procedure for the prediction service providers, which can help to locate possible problems and tune the methods for the best performance.

In the LiveBench tests, each server submits its top ten models as the predictions for one target. For N servers, we will collect $10N$ models for each target. For each submitted model, both its structure file and alignment file are available. The main advantage of the LiveBench is the fast evaluation cycle and easy access of data. All the data are publicly available on the website (http://bioinfo.pl/LiveBench/).

The disadvantage of the LiveBench tests is that these tests are not completely blind since some servers update their template databases synchronously with the PDB.

5.2 Experiment Setup

To test the performance of our new consensus algorithm, the LiveBench 5∼8 data were downloaded to train the SVMR model and test it. At the time of the experiments, LiveBench 8 was incomplete, with only 148 targets. As mentioned, the

LiveBench tests are not completely blind. Some servers submitted the experimental structure of a target as their predictions, or built models from the experimental structure. To reduce biases and errors, we removed such trivial predictions by preprocessing downloaded data.

After preprocessing, the proposed features were extracted from each model by comparing the model with other models by MaxSub. Thus, four sets of data were obtained, corresponding to LiveBench 5∼8, with a different number of targets in each data set. In the following context, the four data sets will be referred to as LiveBench5, LiveBench6, LiveBench7, and LiveBench8, respectively. To test the performance of our meta server, a four-fold cross validation was used in our experiments. Specifically, one data set was used to train an SVMR model and the trained model was tested by the other three data sets. This was repeated four times, once for each data set for training.

The software that was used to train our SVMR model is *SVMlight* which was developed by Thorsten Joachim [27]. To obtain the best performance, a proper kernel function must be selected. After the comparisons are considered by experiments, a RBF kernel was used as the option for the SVM software. The RBF kernel has the following form:

$$\kappa(x_1, x_2) = exp(-\|x_1 - x_2\|^2/2\sigma^2) \tag{5.1}$$

There are two parameters that are tunable for the kernel function: σ and a cost-factor. Different combinations of the two parameters were explored to best tune the SVMR model. To obtain the best performance, all the features were normalized before being used to train the SVMR model. The testing data were also normalized with respect to the means and standard errors of the training data. The performance of a server is normally measured in terms of sensitivity and specificity.

5.3 Sensitivity

Sensitivity is defined as the sum of the MaxSub scores of the top models for all the targets. First, we collected the results of three individual servers: FFAS03[16], 3D-PSSM [18], and FUGUE2 [22]. To evaluate our method more objectively, it is desirable to include individual servers that are comparable to each other. We did not use our own RAPTOR server which significantly outperformed others in CAFASP3 [7]. The top ten models from each server were collected so there were 30 input models for each target. By comparing the models with each other, all the features were generated and the RBF kernel was used in the SVMR. There are two tunable parameters in the SVM regression model. The parameters with the best performance were used in our experiments.

In Table 5.1, it is evident that no matter which data set is used for training, our SVMR method consistently exhibits stable sensitivity on all the test sets and the variation of the data in each column is very small. The sum of the MaxSub scores on the four LiveBench data sets is 498.27. A comparison with the sensitivity of 3D-Jury and the component servers is presented in Table 5.2. It is evident that 3D-Jury-All has poor performance, which is not even as good as some component servers. Both ACE and 3D-Jury-Single perform better than any component server. For ACE, its sensitivity is superior to that of any component server by approximately 8%. For the same three component servers, the sensitivity of ACE is higher than that of 3D-Jury-Single by 6%. Also, the sensitivity of ACE is approximately 10% better than that of Pcon in LiveBench 5$\sim$7. Moreover, the performance of any component server is not as stable as that of our meta server. For example, FFAS03 performs well in LiveBench 5$\sim$7, but poorly in LiveBench 8. In contrast, 3D-PSSM does not perform as well as FFAS03 in LiveBench 5$\sim$7, but much better than FFAS03

Table 5.1: MaxSub scores of ACE with three component servers used. The number of targets is shown under the name of each data set. One data set is used for training and the other three for testing. The average testing result for each data set is calculated and summed.

Training Data Set	LiveBench5 78	LiveBench6 98	LiveBench7 115	LiveBench8 148	Sum of Average
LiveBench 5	—	77.31	93.18	257.53	—
LiveBench 6	73.59	—	91.73	256.57	—
LiveBench 7	73.15	75.35	—	256.87	—
LiveBench 8	73.15	74.47	91.89	—	—
Average	73.30	75.71	92.27	256.99	498.27

in LiveBench 8. With the proposed consensus algorithm, the performance of our meta server is consistently stable.

Based on the previous three servers, FFAS03, 3D-PSSM, and FUGUE2, we conducted another experiment with three additional servers: INBGU [28], SUPERF_PP [17], and mGenTHREADER [39]. The results are provided in Table 5.3. We expected to achieve better performance by using more servers because more candidates were available. The experimental results from six servers are surprisingly worse than those of three servers for our meta server, ACE. In spite of this, ACE is still superior to any component server. When more servers are included, a meta server is more likely to collect even better models in its inputs. When

Table 5.2: Sensitivity (MaxSub score) comparison with three component servers and other meta servers. The results of 3D-Jury are derived from the same three component servers: FFAS03, 3D-PSSM and FUGUE2. The results of all the other servers are taken from LiveBench. Pcon's results are only available for LiveBench 5-7.

Training Data Set	LiveBench5 78	LiveBench6 98	LiveBench7 115	LiveBench8 148	Sum Score
FFAS03	69.68	66.30	88.97	234.52	459.97
3D-PSSM	58.97	61.62	81.47	252.17	454.23
FUGUE2	59.53	63.54	79.71	233.59	436.37
Pcon	62.79	68.65	83.77	—	—
3D-Jury-all	44.24	57.80	64.52	191.38	357.94
3D-Jury-single	64.09	65.36	88.26	250.08	467.79
ACE	73.30	75.71	92.27	256.99	498.27

more models are collected, this brings two problems. First, if some poor quality models are included, they will contaminate the extracted features, which will result in performance degradation. Secondly, the capability of the learning machine is not unlimited. When more candidates are to be considered, it becomes more challenging for the learning machine to select the best one from the candidates. For meta servers that use statistical methods, the performance is enhanced and

Table 5.3: MaxSub scores of ACE obtained with six component servers.

Training data set	LiveBench5 78	LiveBench6 98	LiveBench7 115	LiveBench8 148	sum of average
LiveBench 5	—	63.15	92.57	253.44	—
LiveBench 6	69.22	—	90.37	253.39	—
LiveBench 7	68.47	68.52	—	255.81	—
LiveBench 8	67.30	69.11	90.94	—	—
Average	68.33	66.93	91.29	254.21	480.76

becomes more stable, as more servers are included. When the number of servers exceeds some threshold, the performance of such meta servers will no longer vary significantly with the number of component servers. 3D-Jury is such an example. When six servers are used, its performance is improved significantly as observed in Table 5.4. This is the principal difference between machine learning-based meta servers and statistical method-based meta servers.

Table 5.4 summarizes the comparison with 3D-Jury and the six individual servers. In this case, ACE and 3D-Jury-Single have very similar performances and have higher sensitivities than any component server, whereas ACE does not have an obvious advantage over 3D-Jury-Single. Note that for 3D-Jury-All and 3D-Jury-Single, when the number of servers increases, their performance is improved significantly.

Table 5.4: Sensitivity comparison of ACE, six component servers, and the meta servers: Pcon and 3D-Jury. The results of 3D-Jury are derived from the same six component servers. The results of all the other servers are taken from LiveBench.

data set	LiveBench5 78	LiveBench6 98	LiveBench7 115	LiveBench8 148	sum score
FFAS)3	69.68	66.30	88.97	234.52	459.97
3D-PSSM	58.97	61.62	81.47	252.17	454.23
FUGUE2	59.53	63.54	79.71	233.59	436.37
INBGU	61.56	46.63	79.25	219.22	406.66
SUPERF_PP	40.34	50.02	58.67	186.72	335.75
MGenTHREADER	58.52	68.04	70.98	237.40	434.94
Pcon	62.79	68.65	83.77	—	—
3D-Jury-all	64.09	65.36	88.26	250.08	467.79
3D-Jury-single	68.56	65.59	91.52	256.41	482.08
ACE	68.33	66.93	91.29	254.17	480.76

5.4 Specificity

In addition to the sensitivity of servers, specificity is also important for high-throughput automated structure prediction servers. High sensitivity and specificity

are desirable goals, but are difficult to achieve simultaneously. In this study, specificity was calculated by the method applied by CAFASP3 [7] as follows: (1) Rank models by their confidence scores (SVM outputs). Note that only the top one model for each target is considered here; (2) Count the number of the correct predictions before the first K false positives $TP(K)$; (3) Calculate the average of $TP(K)$, K=1, 2, $\cdots$, 5 as the specificity of the server. Here, a correct model is defined as a model which has at least 40 C_α atoms that can be superimposed on the native structure within 3.0 $\dot{A}$ by using the MaxSub program [6].

Table 5.5: Specificity of ACE, obtained with three component servers.

Training	LiveBench5	LiveBench6	LiveBench7	LiveBench 8
data set	78	98	115	148
LiveBench 5	—	18.20	24.00	59.80
LiveBench 6	18.00	—	24.00	59.80
LiveBench 7	18.00	19.00	—	59.80
LiveBench 8	18.00	19.00	24.00	—
Average	18.00	18.73	24.00	59.80

The specificity of ACE was calculated with the same three component servers by using four fold cross validation. The results are listed in Table 5.5 and a comparison with 3D-Jury, Pcon and the component servers is shown in Table 5.6. We can see that the specificity of ACE is significantly higher than that of 3D-Jury-All and 3D-Jury-Single. Also, the specificity of ACE is higher than that of any component server and Pcon. The specificity of ACE was also calculated when six individual

Table 5.6: Specificity comparison between ACE and its three component servers and other meta servers.

Training	LiveBench5	LiveBench6	LiveBench7	LiveBench 8
data set	78	98	115	148
FFAS03	18.00	17.00	23.00	56.60
3D-PSSM	15.80	16.80	20.40	57.00
FUGUE2	18.00	16.60	17.60	57.79
Pcon	16.00	19.00	22.00	—
3D-Jury-All	12.00	15.20	16.00	54.00
3D-Jury-Single	15.40	15.60	17.80	59.60
ACE	18.00	18.73	24.00	59.80

servers were used. The results in Table 5.7 and Table 5.8 indicate that when six servers are used, the specificity of ACE drops as its sensitivity does. Even though the specificity is still better than that of 3D-Jury-All, it is no longer higher than that of any individual server.

5.5 CASP6 Evaluation

In this section, we will present ACE's performance in the CASP6 evaluation. Unlike the LiveBench tests, the CASP experiments are totally blind. Participants of

Table 5.7: Specificity of the ACE, obtained with six component servers.

Training	LiveBench5	LiveBench6	LiveBench7	LiveBench8
data set	78	98	115	148
LiveBench 5	—	14.80	20.60	59.60
LiveBench 6	15.00	—	21.20	59.40
LiveBench 7	15.00	15.60	—	59.60
LiveBench 8	15.00	15.60	21.20	—
Average	15.00	15.33	21.00	59.53

CASPs are either human groups or automatic servers. It is the largest scale blind test of structure prediction servers in the community. The structures of testing sequences are unknown until the end of the test. In CASP6, both individual servers and meta servers must submit the prediction for each target sequence within 48 hours after the sequence was released. In CASP6, ACE mainly used RAPTOR, 3D-PSSM, FUGUE3, FFAS04, SamT02, and mGenTHREADER.

Each server can submit up to five models for each target. Global Distance Test (GDT) scores are used to measure model quality. The GDT score of a model represents the percentage of residues in the model that are close to those in the template. "Closeness" means within $0.5\dot{A}$, or $1.0\dot{A}$, or $1.5\dot{A}$, ... , or $10.0\dot{A}$. Human groups and automatic servers were ranked together in the final ranking list. The top 16 groups turn out to be human groups. BAKER-ROBETTA is ranked 17th and ACE is ranked 18th. So ACE is the second in the ranking list of meta servers.

Table 5.8: Specificity comparison between the ACE and its six component servers and other meta servers.

Training data set	LiveBench5 78	LiveBench6 98	LiveBench7 115	LiveBench 8 148
FFAS03	18.00	17.00	23.00	56.60
3D-PSSM	15.80	16.70	20.40	57.00
FUGUE2	18.00	16.60	17.60	56.20
INBGU	18.00	19.00	24.00	55.79
SUPERF_PP	18.00	18.73	24.00	59.80
MGenTHREADER	18.00	18.73	24.00	59.80
3D-Jury-All	16.40	14.8	18.20	58.60
3D-Jury-Single	15.00	15.20	21.20	58.80
ACE	15.00	15.33	21.00	59.53

5.6 Conclusion and Future Work

In this study, we introduce an SVM regression-based approach to build the protein fold recognition meta server, Alignment by Consensus Estimator (ACE). ACE extracts features from each protein structure model by structural comparisons, and predicts model quality by applying SVM regression. All the structure models, generated by individual servers, are ranked according to the predicted model qual-

ity, from which an output is selected. Test experiments were conducted on the LiveBench data and experimental results show that our meta server is more sensitive and specific than the component servers, and slightly better than 3D-Jury and Pcon, when not many individual servers are available for consensus. This feature is very desirable, since collecting the prediction results of many servers is not a trivial task. Also, there are not many structure prediction servers that provide unlimited and consistent service to the community. There is still the problem of finding the best combination of individual servers to produce the best prediction. This topic has not been studied before in the community and is our future research topic. The work in this study will be published for the APBC Conference (Singapore, Jan. 2005).

5.7 Acknowledgement

The author wishes to thank the developers of all the servers, FFAS03, SUPERF_PP, 3D-PSSM, FUGUE2, INBGU, MGenTHREADER, Pcon and 3D-Jury, and the developers of the LiveBench experiments, which provide an excellent platform for benchmarking a new server. Our research is supported by the National Science and Engineering Research Council of Canada, CITO's Champion of Innovation Program, the Killam Fellowship, and the Canada Research Chair Program.

Bibliography

[1] Carl Branden and John Tooze. *Introduction to protein structure.* GarLand Publishing Inc., 1995.

[2] Molecular Modeling: A Method for Unraveling Protein Structure and Function. http://www.ncbi.nlm.nih.gov/about/primer/molecularmod.html, 2004.

[3] Daniel Fischer, Arne Elofsson, and Leszek Rychlewski. The 2000 Olympic Games of protein structure prediction; fully automated programs are being evaluated vis-a-vis human teams in the protein structure prediction experiment CAFASP3. *Protein Engineering*, 13(10):667–670, October 2000.

[4] Janusz M. Bujnicki, Arne Elofsson, Daniel Fischer, and Leszek Rychlewski. LiveBench-2: Large-scale automated evaluation of protein structure prediction servers. *Proteins: Structure, Function and Genetics Suppl*, 5:184–191, 2001.

[5] Jinbo Xu. *Protein Structure Prediction by Linear Programming.* PhD thesis, University of Waterloo, 2003.

[6] Leszek Rychlewski, Daniel Fischer, and Arne Elofsson. LiveBench-6: Large-scale evaluation of protein structure prediction servers. *Proteins: Structure, Function and Genetics*, 53:542–547, 2003.

[7] Daniel Fischer, Leszek Rychlewski, Roland L. Dunbrack, Angel R. Ortiz, and Arne Elofsson. CAFASP3: The third critical assessment of fully automated structure prediction methods suppl. *Proteins: Structure, Function and Genetics*, 6(53):503–516, October 2003.

[8] The three-dimension structure of a protein structure is crucial to its function. http://www.wellcome.ac.uk/en/genome/thegenome/hg03b001.html, 2003.

[9] Linus Pauling, RoBert Corey, and Herman Brandson. The structure of proteins; two hydrogen-bonded helical configurations of the polypeptide chain. *Proc Natl Acad Sci U S A*, 37(4):205–11, 1951.

[10] Protein Data Bank. http://www.rcsb.org/pdb/holdings.html, 2003.

[11] John Moult, Tim Hubbard, Krzysztof Fidelis, and Jan T. Pedersen. Critical assessment of methods on protein structure prediction (CASP)-round III. *Proteins: Structure, Function and Genetics Suppl*, 37(3):2–6, December 1999.

[12] John Moult, Krzysztof Fidelis, Adam Zemla, and Tim Hubbard. Critical assessment of methods on protein structure prediction (CASP)-round IV. *Proteins: Structure, Function and Genetics Suppl*, 45(5):2–7, December 2001.

[13] John Moult, Krzysztof Fidelis, Adam Zemla, and Tim Hubbard. Critical assessment of methods on protein structure prediction (CASP)-round V. *Proteins: Structure, Function and Genetics Suppl*, 53(6):334–339, October 2003.

[14] Richard Bonneau and David Baker. Ab initio protein structure prediction: progress and prospects. *Annual Review Biophysics Biomolecular Structure*, 30:173–89, 2001.

[15] Anna Tramontano and Veronica Morea. Assessment of homology-based predictions in CASP5. *Proteins: Structure, Function and Genetics*, 53:352–368, 2003.

[16] Leszek Rychlewski, Lukasz Jaroszewski, Weizhong Li, and Adam Godzik. Comparison of sequence profiles. strategies for structural predictions using sequence information. *Protein Science*, 9:232–241, 2000.

[17] Julian Gough and Cyrus Chothia. SUPERFAMILY: HMMs representing all proteins of known structure. SCOP sequence searches, alignments, and genome assignments. *Nuclear Acids Research*, 30(1):268–272, 2002.

[18] Lawrence A. Kelley, Robert M. MacCallum, and Michael J.E. Sternberg. Enhanced genome annotation using structural profiles in the program 3D-PSSM. *Journal of Molecular Biology*, 299(2):499–520, 2000.

[19] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[20] Jinbo Xu, Ying Xu, Dongsup Kim, and Ming Li. RAPTOR: Optimal protein threading by linear programming. *Journal of Bioinformatics and Computational Biology*, 2003.

[21] Jinbo Xu and Ming Li. Assessment of RAPTOR's linear programming approach in CAFASP3. *Proteins: Structure, Function and Genetics Suppl.*, 53(6):579–84, 2003.

[22] Jiye Shi, Tom L. Blundell, and Kenji Mizuguchi. FUGUE: Sequence-structure homology recognition using environment-specific substitution tables

and structure-dependent gap penalties. *Journal of Molecular Biology*, 310:243–257, 2001.

[23] Ying Xu, Dong Xu, and Edward C. Uberbacher. An efficient computational method for globally optimal threadings. *Journal of Computational Biology*, 5(3):597–614, 1998.

[24] Kevin Karplus, Christian Barrett, and Richard Hughey. Hidden Markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, 1998.

[25] Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[26] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. Technical report, 1998.

[27] Thorstem Joachims. *Making large-scale support vector machine learning practical*. MIT Press, Cambridge, MA, 1998.

[28] Daniel Fischer. Hybrid fold recognition: Combining sequence derived properties with evolutionary information. pages 119–130, Hawaii, 2000. Biocomputing: Proceedings of the 2000 Pacific Symposium, World Scientific Publishing Co.

[29] Jesper Lundström, Leszek Rychlewski, Janusz Bunnicki, and Arn Elofsson. Pcons: A neural-netwrok-based consensus predictor that improves fold recognition. *Protein Science*, 10:2354–2362, 2001.

[30] Janusz M. Bujnicki and Daniel Fisher. 'meta' approaches to protein structure prediction. *Nucleic Acids and Molecular Biology*, 14:23–33, 2004.

[31] Björn Wallner, Huisheng Fang, and Arne Elofsson. Automatic consensus-based fold recognition using Pcons, ProQ, and Pmodeller. *Proteins: Structure, Function and Genetics Suppl*, 6:534–541, 2003.

[32] Krzysztof Ginalski, Arne Elofsson, Daniel Fischer, and Leszek Rychlewski. 3D-Jury: a simple approach to improve protein structure predictions. *Bioinformatics*, 19(8):1015–1018, 2003.

[33] Iris Sasson and Daniel Fischer. Modeling three-dimensional protein structures for CASP5 using the 3D-SHOTGUN meta-predictors. *Proteins: Structure, Function and Genetics*, 53:389–394, 2003.

[34] Kim T. Simons, Charles Kooperberg, Enoch Huang, and David Baker. Assembly of protein tertiary structures from fragments with similar local sequences using simulated annealing and Bayesian scoring functions. *Molecular Biology*, 268(1):209–225, 1997.

[35] Björn Wallner and Arne Elofsson. Can correct protein models be identified? *Protein Science*, 12(5):1073–1086, 2003.

[36] Naomi Siew, Arne Elofsson, Leszek Rychlewski, and Daniel Fischer. MaxSub: An automated measure for the assessment of protein structure prediction quality. *Bioinformatics*, 16(9):776–785, 2000.

[37] Anne Louise Morris, Malcolm W. MacArthur, E. Gail Hutchinson, and Janet M. Thornton. Stereochemical quality of protein structure coordinates. *Proteins: Structure, Function, and Genetics*, 12:345–364, 1992.

[38] LiveBench Tests. http://bioinfo.pl/livebench.

[39] David T. Jones. GenTHREADER: An efficient and reliable protein fold recognition method for genomic sequences. *Journal of Molecular Biology*, 287:797–815, 1999.