Technical Report

STP: The Sample-Train-Predict Algorithm and Its Application to Protein Structure Meta-Selection

Hani Z. Girgis and Jason J. Corso

Department of Computer Science and Engineering
University at Buffalo, The State University of New York

September 16, 2008

STP: The Sample-Train-Predict Algorithm and Its Application to Protein Structure Meta-Selection

Hani Z. Girgis, Jason J. Corso
Department of Computer Science and Engineering
University at Buffalo, The State University of New York
Buffalo, NY 14260
{hzgirgis,jcorso}@cse.buffalo.edu

Abstract

The importance and the difficulty of the folding problem have led scientists to develop several computational methods for protein structure prediction. Despite the abundance of protein structure prediction methods, these approaches have two major limitations. First, the top ranked model reported by a server is not necessarily the best predicted model. The correct predicted model may be ranked within the top 10 predictions after some false positives. Second, no single method can give correct predictions for all proteins. To attempt to remedy these limitations, protein structure prediction "meta" approaches have been developed. A meta-server can select a set of candidate models by ranking models obtained from other servers. In this article we present the Sample-Train-Predict algorithm and its application to implement a new model quality assessment program (MQAP) based on a consensus of five MQAP's then we discuss the application of our MQAP as a meta-selector. STP depends on the clustered nature of the training data and it can dynamically handle constantly growing training data. STP selects clusters which are similar to the input data, then trains a model on these clusters, and finally uses the trained model to get predictions for the input data. Our experimental results show that a hierarchical model trained using STP outperforms any tested model quality assessment program by 7%-8%. When selecting from predictions made by humans in a standard benchmark CASP7, our meta-selector achieves about 3% improvement above the best human predictor.

1 Introduction

We live in the post-genomic era in which a large amount of biological data are being accumulated. Such data have two main properties. First, the available training (labeled) data is constantly growing. For example, gene banks and protein structure banks are increasing in size on an annual or even on a monthly or a daily basis. Second, the data is intrinsically clustered, meaning that data can be clustered based on similarity in sequence, structure or function i.e. each cluster has a biological meaning. Once new labeled data which may contain new patterns becomes available, models such as artificial neural network (ANN), support vector machine (SVM), and decision trees trained on the old training data become out of date and need to be retrained on the updated training set. Our goal is to devise a machine learning algorithm that can extend these models to make use of the newly available labeled data dynamically. To that end, we propose two algorithms to realize this goal. The first algorithm trains a model dynamically on related data to the unlabeled query (testing) data, in another words, it trains dynamically a custom-made expert. The second algorithm dynamically mixes local experts which are already trained and cached.

One of the most important problems in structural biology is the protein folding problem. A few principles that govern protein folding are currently known; however, the main folding algorithm is yet to

be discovered [1]. The three-dimensional structure of protein can be determined experimentally by X-ray crystallography, nuclear magnetic resonance and other methods. However, the output of these methods is not as fast as the output of sequencing projects. In addition, these methods are expensive, and not applicable to all proteins. Therefore, protein structure prediction based on computational methods is currently an active research area.

The importance and the difficulty of the folding problem have led scientists to develop several computational methods for protein structure prediction, the majority of which are available as servers accessible through the internet. Despite the abundance of protein structure prediction servers, these approaches have two major limitations [2]. First, the top ranked model reported by a server is not necessarily the best predicted model. The correct predicted model may be ranked within the top 10 predictions after some false positives. Second, no single method can give correct predictions for all target proteins. It has been observed in experiments to assess the accuracy of protein structure prediction methods such as CASP [3] that the correct model is usually predicted by one of the participating servers. To attempt to remedy these limitations, protein structure prediction "meta" approaches have been developed.

Protein structure prediction meta-approaches consider the results obtained from several different methods. A meta-server selects a set of candidate models by ranking models obtained from other servers based on a local model quality assessment program (MQAP). Such servers are known as " s" [4, 5]. Several MQAP's are currently available to assess the quality of a given model of protein of unknown three-dimensional structure. A model quality assessment program can use (i) physics principles, (ii) statistical information derived from the known protein structures, and (iii) machine learning techniques that are trained on both physical and statistical properties of proteins of known structures [7]. Currently many MQAP's [4, 7, 8] [10]-[14] are available and have been applied in protein structure prediction. The majority of these MQAP's are composite scores i.e. they are based on consensus of a few quality scores. $Proq$ [13] and $SIFT$ [9] are ANN-based composite scores. $SVMod$ [7] is a SVM-based composite score. In this article we present The STP: Sample-Train-Predict algorithm and its application *ZicoSelector* which is a meta-MQAP-selector based on a consensus of MQAP's.

2 The STP: Sample-Train-Predict Algorithm

The machine learning scientist often faces the problem of a shortage in labeled training data; however, in fields such as modern computational biology we can acquire new labeled data on daily basis. Thus, one must deal with large quantities of constantly growing training data since the newly acquired labeled data may contain information that is not present in the old training data. Typically one will discard the models trained on the old training data and train new ones. Clearly such an approach is a waste of computation and needs manual human intervention to retrain the learning algorithm. We have devised the Sample-Train-Predict (STP) algorithm to attempt to solve this problem. The STP algorithm can handle a growing training data and learns from the newly added labeled data dynamically. In another words, the STP algorithm grows as the training data set grows.

We propose a dynamic learning algorithm called STP: **S**ample-**T**rain-**P**redict. The STP algorithm can be used when the data have two main properties. First, the available training (labeled) data is constantly growing. For example, gene banks and protein structure banks are increasing in size on an annual or even a monthly basis. Second, the data is intrinsically clustered based on similarity in sequence, structure or function (each cluster has high-level semantic meaning).

We state the problem as follows. The input is a set of labeled data which is continuously growing in size and a set of testing unlabeled data. The algorithm outputs labels for the unlabeled data. Formally, $STP(D, X) = T$ where D is the set of labeled data and the size of the training data set |D| continuously increases. X is the set of testing data and T is the set of labels (classification) or real values (regression) corresponding to elements in set X. The STP algorithm does its prediction in a batch mode i.e. it takes a cluster of data of unknown target values as its input and outputs the results in a batch mode as well. The STP algorithm has three stages: (i) sample (ii) train (iii) predict. We describe two variants of the STP algorithm: STPdata and STPmodel. We consider STPdata as a way to dynamically build a custom-made expert and STPmodel as a method to dynamically mix a set of local experts similar to [15, 17].

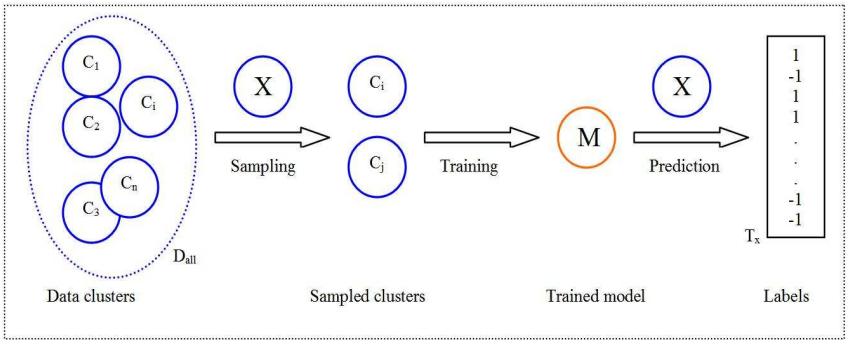

Figure 1: The STPdata Algorithm

2.1 STPdata

Figure 1 outlines the STPdata algorithm. In STPdata, we select a subset of the training data based on the similarity to the unlabeled data. The similarity measure is problem dependent and the implementer should decide on how to sample from the training data. Then the STPdata algorithm trains a model such as ANN, SVM, etc. on the sampled data in the "train" stage. In the "predict" stage, STPdata uses the trained model to assign labels to the unlabeled input set. Since STPdata builds models dynamically, it incorporates the newly added labeled data in the "train" stage.

The STPdata algorithm works as follows: let D_{all} be a set of all the available training data clusters, and X is the set we wish to predict its unknown target values i.e. X is the testing data, then D_x is the union of similar clusters to cluster X in the training data and is the output of the sample function such that $s(D_{all}, X) = D_x$. The sample function is defined as follows. *Input:* D_{all} = { D_i | D_i is a subset of the training data} and X. Let $A, B \in D_{all}$ and

$$\delta(A, B) = \begin{cases} 1 & \text{if set } A \text{ and set } B \text{ are neighbors;} \\ 0 & \text{o.w.} \end{cases}$$

Output: D_x such that $D_x \subseteq D_{all}$, $D_x \neq \emptyset$ and $D_x = \{D_i | D_i \in D_{all}$ and $\delta(D_i, X) = 1\}$.

The train function outputs a model M_x trained on the data sampled in the previous step i.e. the neighbors of X such that $t(D_x) = M_x$. The train function is defined as follows. *Input:* $D_x = \{(b^1, a_1), ..., (b^k, a_k)\}$ where input b^i is m-dimensional vector and a^i is the corresponding target. *Output:* A model M_x trained on D_x. The predict function outputs T_x which is a set of the predicted target values or labels for set X such that $p(M_x, X) = T_x$. The predict function is defined as follows. *Input:* M_x is a model trained on the neighbors of set X. *Output:* T_x is the prediction of set X target values.

The STPdata algorithm has been implemented successfully to train a hierarchy of linear models based on the pseudo inverse solution, details are given in section 3. The STPdata cycle takes a few seconds for a batch of about 200 queries. However, retraining a non-linear model for each batch can be computationally expensive and limits the practical models to computationally simple ones e.g. linear. For example, training a multi-layer neural network requires a long time of training, in addition to manually adjusting several parameters. To remedy this limitation, we propose (i) the STPmodel algorithm discussed in section 2.2 (ii) a modification to the STPdata algorithm to cache the trained models.

The modified STPdata algorithm treats the training data as an array of clusters in which each cluster can be indexed by a unique number. Each trained model is associated with the indices of the clusters used in its training. The sampling step returns the indices of the clusters which are similar to the testing cluster. There is an extra query step before the training step. In the query step it searches the cached trained models for a model whose indices significantly overlap with the sampled clusters indices. If such a model exists, then STPdata uses it in the prediction and the training step is escaped, otherwise it trains a new model. The modified STPdata algorithm grows as it is being used and as the training data grow.

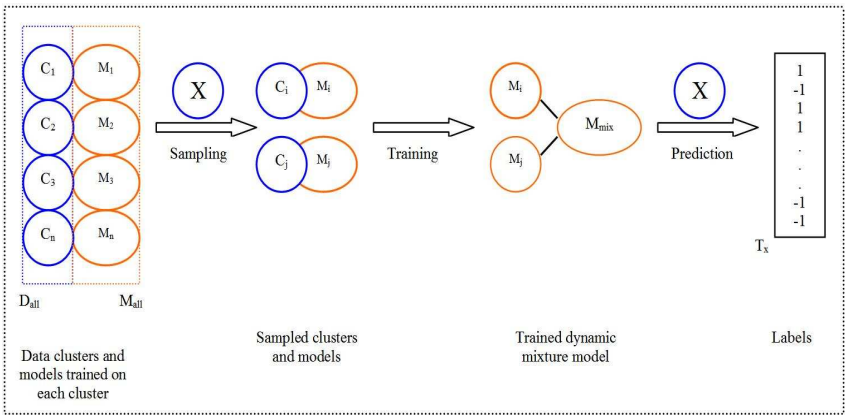

Figure 2: The STPmodel Algorithm

2.2 STPmodel

Figure 2 outlines the STPmodel algorithm. In the STPmodel algorithm we start with a set of trained models each is trained on a cluster of the training data; therefore, in the "sample" stage, we sample from trained models by selecting models which are trained on data similar to the unlabeled data. In the "train" stage we combine these models linearly and use that mixed model in predicting the labels of the unlabeled input set. When we have new labeled data set, we train a new model on these data, then add that trained model to the trained models set. Since STPmodel builds mixed models dynamically, it incorporates the newly added labeled data in the training stage.

The STPmodel algorithm works as follows: let D_{all} and X be defined as before, M_{all} is a set of models each trained on a cluster of the training data. Then the sampling stage in STPmodel collects two sets. The first set is M_x which is a set of models M_i's, each M_i is trained on a cluster that is similar to set X. The second set is D_x which is the union of similar clusters to cluster X in the training data such that $s(D_{all}, M_{all}, X) = \{M_x, D_x\}$. The sample function is defined as follows. *Input:* D_{all} = $\{$ D_i | D_i is a subset of the training data$\}$, M_{all} = $\{$ M_i | M_i is a model trained on subset D_i of the training data$\}$ is a set of models M_i's, each M_i is trained on a cluster $D_i \in D_{all}$, and the set X. Let $A, B \in D_{all}$ and $\delta(A, B)$ defined as before. *Output:* M_x such that $M_x \subseteq M_{all}$, $M_x \neq \emptyset$ and M_x = $\{M_i | M_i \in M_{all}$ and $D_i \in D_{all}$ and $\delta(D_i, X) = 1\}$. D_x such that $D_x \subseteq D_{all}$, $D_x \neq \emptyset$.

The train function outputs a linear model M_{mix} trained to linearly combine the outputs of the set of local experts sampled in the sampling step such that $t(M_x, D_x) = M_{mix}$. The train function is defined as follows. *Input:* M_x =$\{$ M_i | M_i is a model trained on subset D_i of the training data$\}$ is a set of local experts each trained on cluster D_i, $D_x = \{(b^1, a_1), ..., (b^k, a_k)\}$ where input b^i is m-dimensional vector and a^i is the corresponding target, D_x is the union of clusters D_i's. *Output:* a linear model M_{mix} trained to linearly combine the outputs of models M_i's. Each M_i is used to predict the target values of set D_x. In other words, the train function assigns weights to these local experts.

In the prediction step we use the set of local experts M_x and the mixing model M_{mix} which we have trained in the training stage to predict T_x which is a set of the predicted target values or labels for set X such that $p(M_x, M_{mix}, X) = T_x$. The predict function is defined as follows. *Input:* M_x is a set of local experts. M_{mix} is a linear mixing model trained on the neighbors of the testing set X. *Output:* T_x is the prediction of set X target values.

The STPmodel algorithm is related to the mixture of local experts [17, 15]. One applies a mixture of local experts if the data set can be partitioned into smaller subsets which have a higher level semantics. Then we train a system of a set of local experts and a gating model. Each local expert is trained on a subset. The gating model allocates the experts to be used on a given input data and decides the strategy to combine the experts' predictions. Hampshire and Waibel [17] use a gating model that linearly combines the outputs made by the local experts while Jacobs and Hinton [15]

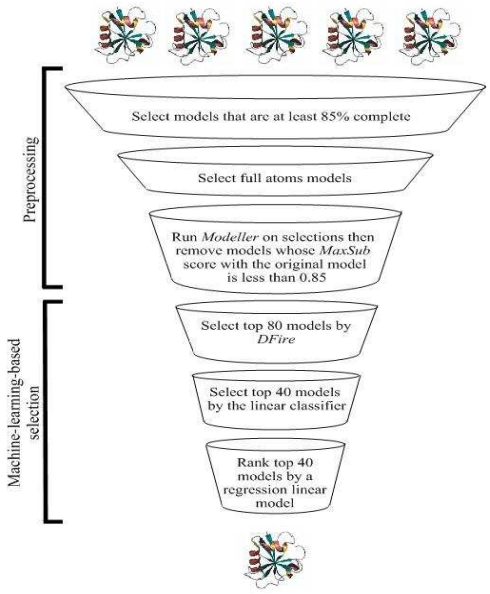

Figure 3: ZicoSelector: a hierarchy of general linear models

adapt a competitive strategy in the gating model to select the local experts to be used on the input data. The weights of the local experts and the gating model are adjusted together in the training process. Each local expert is a multi-layer network. We regard the STPmodel as a way to build dynamically a mixture of local experts in which the weights of the local experts and the gating model are adjusted separately. The STPmodel algorithm can mix ANN, SVM, decision trees and other models and is not limited to ANN as the mixtures described by Hampshire, Waibel, Jacobs and Hinton.

3 An Example Application: Protein Structure Meta-Selection

In this section we consider a concrete implementation of the STP algorithm. The data used in this experiment is the protein structure predictions submitted to CASP6 and CASP7 by servers and human predictors. The data is intrinsically clustered i.e. all three-dimensional structures predicted for the same a.a. (amino acids) sequence form a cluster. For CASP6 there are about 60 clusters and for CASP7 there are about 100 clusters. New clusters are available once new protein structures are deposited in the protein structures bank. In this section we first state the meta-selection problem. Second, we describe a hierarchical model called *ZicoSelector*. Then we illustrate how the STPdata and the STPmodel algorithms train *ZicoSelector* dynamically.

The protein structure meta-selection problem can be stated formally as follows. Input: A $n \times m$ scores matrix S, such that s_{ij} is the score of 3d-structure i assigned by MQAP j. The scores matrix S contains the MQAPs' scores of the 3d-structures predicted for the same a.a. sequence.

$$S = \begin{pmatrix} S_{11} & S_{12} & . & . & . & S_{1m} \\ . & . & . & & & . \\ . & . & & . & & . \\ . & . & & & . & . \\ S_{n1} & S_{n2} & . & . & . & S_{nm} \end{pmatrix}$$

Output: The predicted quality score of each 3d-structure.

ZicoSelector is a hierarchy of general linear models (GLM). Figure 3 gives an overall view of the hierarchical system. The input to *ZicoSelector* is a set of 3d-structures predicted for the same amino acids sequence. The selection process is composed of three levels. At the first level, the top n_1 models ranked by $DFire$ are chosen to pass to the second level where a linear classifier is used to

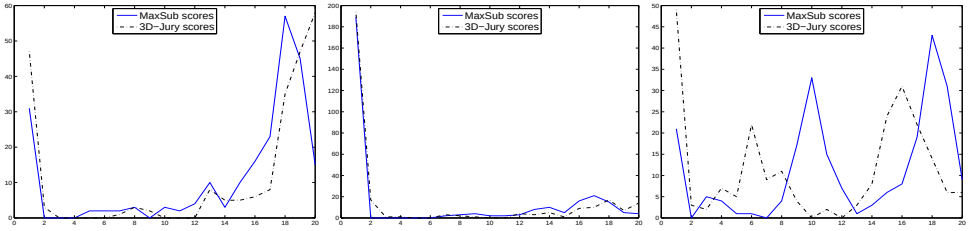

Figure 4: The distributions of MaxSub scores and 3D-Jury scores of the 3d-strucutres generated from three different a.a. sequences

select models whose ranks are less than n_2. Models selected at the second level are ranked by a regression linear model at the final stage. The GLM for n observations can be expressed as

$$A = Sv + e$$

Where A is a matrix of target values, S is the $n \times m$ features matrix as defined earlier, v is the m-dimensional regression coefficients vector, and e is the m-dimensional errors vector. Let w be a $m + 1$ dimensional vector of 1 and m regression coefficients, then

$$Sw = A$$
$$S^T Sw = S^T A$$
$$w = (S^T S)^{-1} S^T A$$

Next we describe how we train the *ZicoSelector* using the STPdata algorithm.

Pre-processing. This stage is designated to the following data preprocessing (i) select models that are at least 85% complete (ii) select full atoms models (iii) run the $Modeller$ program on selections, Modeller is a homology or comparative modeling program, then remove models whose $MaxSub$ score with the original model is less than 0.85. MaxSub is a similarity measure between two 3d-strucutres.

STP Sampling. STPdata considers the MQPAs' scores of the 3d-strucutre generated for the same a.a. sequence as one cluster. We represent each cluster by two centers of the bimodal distribution of the 3D-Jury score and the percentages of the 3d-structures that belong to each mode. We obtain the two centers by applying the k-means clustering algorithms with initial centers 0.0 and 1.0. We specifically choose the 3D-Jury score for the following three reasons (i) the 3D-Jury score is highly correlated with the MaxSub (the target) score (ii) the distributions of MaxSub and 3D-Jury scores are bimodal as shown in figure 4 (iii) the distribution of the 3D-Jury score has one of the least KL divergence from the distribution of the MaxSub score indicating the similarity between these two distributions.

STP Training. Once the sampling step is done, we train *ZicoSelector*, which is a hierarchy of general linear models, on the sampled data. Training a linear model based on the pseudo inverse solution is very fast which makes the STPdata algorithm suitable to the problem under consideration. Next we describe the three levels of the hierarchical model.

ZicoSelector: first level. Models whose $DFire$'s rank is less than n_1 are chosen to pass to the next stage. We assume that models whose ranks are greater than n_1 are noisy models and hence will affect the training process negatively.

ZicoSelector: second level. In this stage a linear classifier (GLM) is trained to select the top n_2 models. The classifier is trained to separate two classes: the first class is composed of models whose ranks are less than n_2 and the second class is composed of models whose ranks are greater than n_2. We choose $n_2 = 0.5 \times n_1$ to make sure that the data is balanced and the classifier is not biased to any of the two classes. The linear classifier is trained on set $D_1 = \{(x^1, a_1), ..., (x^q, a_q)\}$ Where $q = n_1 \times |T|$, such that $|T|$ is the number of the clusters in the sampled data, inputs $x^1, ..., x^q$ are m-dimensional vectors and m is the number of MQAP's. $x^i = \{x_1, ..., x_m\}$, where

$$x_j = \begin{cases} 1 & \text{if the mqap } j \text{ ranks the model below } n_2 \text{ ;} \\ -1 & \text{o.w.;} \end{cases}$$

Methods	Easy	Medium	Hard	Total
Best Human	16.399	24.53	6.116	47.761
Best MQAP	15.834	23.872	5.164	45.62
ZicoSelectorSTP	16.175	25.329	6.297	49.263
Improvement over Best Human	-1.366%	3.257%	2.959%	3.145%
Improvement over Best MQAP	2.154%	6.103%	21.940%	7.986%

Table 1: The performance of *ZicoSelector* trained by STPdata on the set of human predictions

ZicoSelector: third level. Models selected by the linear classifier in the previous stage are passed to this stage. A linear model for regression is trained to predict the rank of the 3d-structures. The training set $D_2 = \{(x^1, a_1), ..., (x^k, a_k)\}$. Where $k = n_2 \times |T|$, inputs $x^1, ..., x^k$ are m-dimensional vectors. $x^i = \{x_1, ..., x_m\}$,where x_j is the model's rank according to the j^{th} MQAP. The output a_j is the model rank according to $MaxSub$. In other words, let $R = \{1, 2, .., n_2 - 1, n_2\}$, then $x_j \in R$ and $a_j \in R$ i.e. the model is trained on the 3d-structures' ranks according to the MQAP's to predict the 3d-structure's rank according to $MaxSub$.

STP Prediction. Once the hierarchical model *ZicoSelector* is trained, STPdata uses it to predict the quality of the query 3d-structures.

To evaluate our methods we have divided targets into four categories according to the difficulty level: easy, medium, hard, and impossible. The impossible category is ignored in our evaluation. The results presented in this section are obtained by setting k to 22 in the k-nearest neighbors algorithm and the threshold at the third stage to 80 and the threshold at the fourth stage to 40.

Two experiments have been conducted to evaluate the performance of *ZicoSelector* trained by STPdata. The meta-MQAP-selector used in both experiments is trained on 3d-structures sampled from predictions made by servers and human predictors in CASP6. In the first experiment, models predicted by human predictors in CASP7 are used in testing. *ZicoSelector* outperforms all tested MQAP's in the three categories and in the total score by about 7%-8%, and outperforms the best human predictor in the medium and the hard categories and in the total score by about 3%. The performance of *ZicoSelector* is shown in table 1, the total score in the table includes scores from the impossible category. In the second experiment, models predicted by servers in CASP7 are used in testing. Results are not shown. *ZicoSelector* outperforms all tested MQAP's and its performance is very similar to the best server performance in all of the three categories and in the overall score.

We also applied the STPmodel algorithm to train *ZicoSelector*. We obtained 56 clusters by applying the k-nearest neighbors algorithm with $k = 20$ to each protein in the CASP6 servers and humans sets. Each cluster is indexed by the target protein. Next, we trained 56 GLM's. Each GLM is trained on one cluster of the 56 clusters. In the sampling stage, we select models based on the similarity between the test protein and the index proteins in the same way used in the STPdata algorithm. STPmodel trains another GLM to learn to combine the outputs of the set of the local experts sampled in the previous stage. Figure 5 shows the performance of the meta-MQAP-selector trained by STPmodel as a function of the number of experts on the servers' set and on the humans' set respectively. The x-axis represents the number of experts collected in the sampling stage and the y-axis represents the total MaxSub score of rank one models selected by the meta-MQAP-selector. The performance of the meta-MQAP-selector deteriorates as the number of experts increases due to the curse of dimensionality. The performance of *ZicoSelector* trained by STPmodel with two or three local experts is very similar to the performance of the one trained by STPdata.

4 Conclusions

Our findings from the current research can be summarized as follows: (i) the invention of the STPdata and STPmodel algorithms which are suitable to large, intrinsically clustered, constantly growing training data sets (ii) the dynamic training used in the STP algorithm proved effective when applied to the protein structure meta-selection problem (iii) the STP algorithm can be generalized and applied to other problems in the computational biology field and problems in other fields. The experimental results show that *ZicoSelector*, our hierarchical model trained using the STPdata algorithm, outperforms any tested MQAP by 7%-8%. When selecting from predictions made by humans in CASP7, our meta-MQAP-selector achieves about 3% improvement over the best human predic-

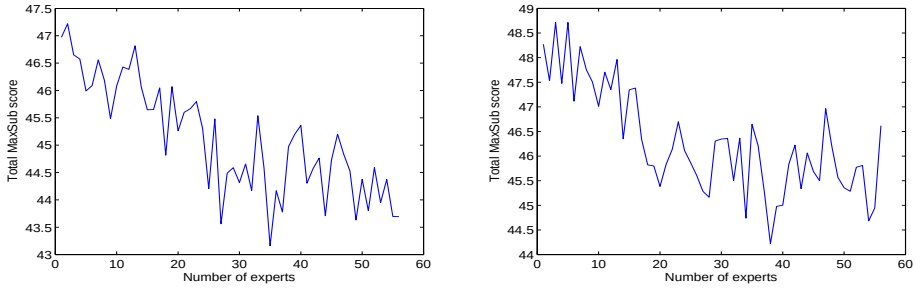

Figure 5: The performance of the meta-MQAP-selector trained by STPmodel as a function of the number of experts on the servers' predictions and on the humans' predictions respectively

tors. The performance of the meta-MQAP-selector on servers' predictions in CASP7 is very similar to the performance of the best server.

Acknowledgments

The first author was supported by NIH grant 1053330/1/38835.

References

[1] Lesk A: *Introduction to Protein Architecture*. New York: Oxford University Press 2001.

[2] Bujnicki J, Fischer D. Meta Approaches to Protein Structure Prediction. In *Practical Bioinformatics, Volume 15*. Edited by Bujnicki JM, Berlin Heidelberg: Springer-Verlag 2004:23–34.

[3] Moult J, Pedersen J, Judson R, Fidelis K. A large-scale experiment to assess protein structure prediction methods. *Proteins* 1995, 23:ii–iv.

[4] Ginalski K, Elofsson A, Fischer D, Rychlewski L. 3D-Jury: a simple approach to improve protein structure predictions. *Bioinformatics* 2003, 19:1015–1018.

[5] Lundstrom J, Rychlewski L, Bujnicki J, Elofsson A. Pcons: A neural-network based consensus predictor that improves fold recognition. *Protein Science* 2001, 10:2354–2362.

[6] Cristobal S, Zemla A, Fischer D, Rychlewski L, Elofsson A. A study of quality measures for protein threading models. *BMC Bioinformatics* 2001, 2:5.

[7] Eramian D, Shen M, Devos D, Melo F, Sali A, Marti-Renom M. A composite score for predicting errors in protein structure models. *Protein Science* 2006, 15:1653–1666.

[8] Eisenberg D, Luthy R, Bowie J. VERIFY3D: Assessment of protein models with three-dimensional profiles. *Methods in Enzymology* 1997, 277(396-404).

[9] Adamczak R, Porollo A, J M. Accurate prediction of solvent accessibility using neural networks-based regression. *Proteins* 2004, 56:753–767.

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

[11] Sippl MJ. Boltzmann's principle, knowledge-based mean fields and protein folding. An approach to the computational determination of protein structures. *J. Comput. Aided Mol. Des.* 1993, 7:473–501.

[12] Tosatto SCE. The victor/FRST function for model quality estimation. *Journal of Molecular Biology* 2005, 12:1316–1327.

[13] Wallner B, Elofsson A. Can correct protein models be identified? *Protein Science* 2003, 12:1073–1086.

[14] Zhou H, Zhou Y. Distance-scaled, finite ideal-gas reference state improves structure-derived potentials of mean force for structure selection and stability prediction. *Protein Science* 2002, 11:2714–2726.

[15] Jacobs RA, Jordon MI, J NS, E HG. Adaptive mixtures of local experts. *Neural Computation* 1991, 3(1):79–87.

[16] Siew N, Elofsson A, Rychlewski L, Fischer D. MaxSub: an automated measure for the assessment of protein structure prediction quality. *Bioinformatics* 2000, 16:776–785.

[17] Hampshire J, Waibel A. The Meta-Pi network: Building distributed knowledge representations for robust pattern recognition. Technical Report CMU-CS-89-166, Carnegie Mellon University, Pittsburgh, PA. 1989.