Published in final edited form as:
Res Comput Mol Biol. 2009 ; 5541: 31–45. doi:10.1007/978-3-642-02008-7_3.

Boosting Protein Threading Accuracy

Jian Peng and **Jinbo Xu***
Toyota Technological Institute at Chicago, Chicago, IL, USA, 60637

Abstract

Protein threading is one of the most successful protein structure prediction methods. Most protein threading methods use a scoring function linearly combining sequence and structure features to measure the quality of a sequence-template alignment so that a dynamic programming algorithm can be used to optimize the scoring function. However, a linear scoring function cannot fully exploit interdependency among features and thus, limits alignment accuracy.

This paper presents a nonlinear scoring function for protein threading, which not only can model interactions among different protein features, but also can be efficiently optimized using a dynamic programming algorithm. We achieve this by modeling the threading problem using a probabilistic graphical model Conditional Random Fields (CRF) and training the model using the gradient tree boosting algorithm. The resultant model is a nonlinear scoring function consisting of a collection of regression trees. Each regression tree models a type of nonlinear relationship among sequence and structure features. Experimental results indicate that this new threading model can effectively leverage weak biological signals and improve both alignment accuracy and fold recognition rate greatly.

Keywords

protein threading; conditional random fields; gradient tree boosting; regression tree; nonlinear scoring function

1 Introduction

Protein structure prediction based on protein threading has made a significant progress due to both enlargement of the Protein Data Bank (PDB) and improvement of prediction protocols. Protein threading predicts the three-dimensional structure of a new protein (i.e., target) by aligning its primary sequence to a similar experimental structure (i.e., template) in the PDB. The PDB statistics1 shows that in recent years, only a limited number of completely new protein folds appear although several thousand new structures are deposited to the PDB. According to [1,2], almost all the single-domain proteins with up to 200 residues can be aligned to a protein in the PDB with an average RMSD less than 5Å and an average coverage of 70%. These observations imply that in principle, a reasonably good template can be identified for most proteins with unknown structures if a perfect protein threading protocol is available.

*Corresponding author..
pengjian@tti-c.org
j3xu@tti-c.org

[1]http://www.rcsb.org/pdb/static.do?p=general_information/pdb_statistics/index.html

The three-dimensional structure model of a target protein is built upon its alignment to the template. Therefore, the alignment accuracy of a protein threading method is critical to the model quality [3]. The best sequence-template alignment is generated by optimizing a scoring function, usu- ally consisting of several sequence and structure items such as sequence profile, secondary structure, solvent accessibility, contact capacity and pairwise interactions. It is difficult to construct accurate alignment between two proteins with low sequence identity even if they are structurally similar. It has been observed that alignment quality drops rapidly when two proteins share less than 25% sequence identity [4]. There are usually errors in the alignment of two proteins with less than 40% sequence identity [5,6,7].

There are a number of protein threading programs in the literature based on various methods such as the profile-profile alignment [8,9,10,11,12,13,14,15], the structural profile alignment [16,17,18,19], the HMM models [20,21], the optimization approaches [22,23,24,25] and the multiple mapping method [26]. The pure sequence profile-based alignment methods have been very successful in identifying a good template for the target. However, it has been demonstrated by many programs that using structure information can improve both alignment accuracy and fold recognition rate. For example, several leading threading programs such as HHpred [21], SPARKS [14,19] and RAPTOR [22,23] make use of structure information such as secondary structure, solvent accessibility and residue depth. Zhang *et al* have shown that by using five structure features plus sequence profile, their threading program MUSTER [27] outperforms their profile-profile alignment program PPA [28]. Zhou *et al* show that a residue-depth based structure profile can improve both the sensitivity and specificity of the sequence profile alignments [14]. Silva shows that a simplified hydrophobicity matrix can help detect remote homologs [29]. Skolnick *et al* show that contact predictions are helpful for the targets without close homologs in the PDB [30].

Although various structure features have been used by many threading programs, they usu- ally use a very simple method to combine these features. Most threading programs use a linear combination of these features as a scoring function [14,27,31,16,18,17]. A linear scoring function can be easily tuned and also can be efficiently optimized using a dynamic programming algorithm. However, a linear scoring function can not accurately account for the interdependency among features. It has been observed that some sequence and structure features (e.g., secondary structure and solvent accessibility) are highly correlated. To model interactions among features, the SVM-align method in [32] explicitly enumerates hundred-thousands of complex features, which leads to the same number of model parameters to be trained. A complex feature is a combination of some basic features, e.g., secondary structure, solvent accessibility and amino acid type. A model with a large number of parameters is not amenable to training since 1) it needs a large number of training examples to fit these parameters; 2) it needs careful tuning to avoid overfitting; and 3) the training process is also time-consuming. Using such a complicated model, it is also computationally intensive to find the best sequence-template alignment between a protein pair, which makes the model unsuitable for protein structure prediction at genome scale. In addition, not all the model features are equally important and some unimportant features may introduce noise into the model.

This paper presents a nonlinear scoring function for protein threading, which not only can model interactions among various sequence and structure features, but also can be optimized quickly using a dynamic programming algorithm. We fulfill this by modeling the protein threading problem using a probabilistic graphical model Conditional Random Fields (CRFs) and training the model using the gradient tree boosting algorithm proposed in [33]. The resultant threading scoring function consists of only dozens of regression trees, which are automatically constructed during model training process to capture the nonlinear relationship among sequence and structure features. Experimental results indicate that by

modeling feature interactions using regression trees, we can effectively leverage weak biological signals and greatly improve alignment accuracy and fold recognition rate.

2 Methods

2.1 Conditional Random Fields Model for Protein Threading

Conditional random fields are probabilistic graphical models that have been extensively used in modeling sequential data [34,35]. Recently, CRFs have also been used to model various computational biology problems such as protein secondary structure prediction [36,37], protein conformation sampling [38] and protein sequence alignment [39]. Here we describe how to model the protein threading problem using conditional random fields. In this paper, for a given pair of target and template, their sequence and structure features are called observations and their alignment is viewed as a sequence of hidden states or labels.

Let s denote the target protein and its associated features, e.g., PSI-BLAST sequence profile, PSIPRED-predicted secondary structure [40] and predicted solvent accessibility. Let t denote the template and its associated information, e.g., position-specific scoring matrix, solvent accessibility and secondary structure. Let $X = \{M, I_s, I_t\}$ be a set of three possible alignment states. Meanwhile, M indicates that two positions are matched and the two Is indicate insertion/deletion states. I_s and I_t indicate insertion at sequence and template, respectively. We also tried to differentiate gaps at the two ends from gaps in the middle region by using four more states, but the resultant 7-state model is only slightly better than the three-state model. Let $a = \{a_1, a_2, \ldots, a_L\}$ ($a_i \in X$) denote an alignment between s and t where $a_i \in X$ represents the state (or label) at position i. Our CRF model for threading defines the conditional probability of a given s and t as follows.

$$P_\theta(a|s,t) = \exp\left(\sum_{i=1}^{L} F(a,s,t,i)\right)/Z(s,t) \quad (1)$$

where $\theta=(\lambda_1, \lambda_2, \ldots \lambda_p)$ is the model parameter and $Z(s,t) = \Sigma_a \exp\left(\Sigma_{i=1}^{L_a} F(a,s,t,i)\right)$ is a normalization factor summing over all the possible alignments between s and t.[2] $F(a, s, t, i)$ is the sum of the CRF features at alignment position i:

$$F(a,s,t,i) = \sum_k \lambda_k e_k(a_{i-1}, a_i, s, t) + \sum_l \lambda_l v_l(a_i, s, t) \quad (2)$$

Where $e_k(a_{i-1}, a_i, s, t)$ and $v_l(a_i, s, t)$ are called edge and label feature functions, respectively. The edge features model the dependency of the state transition (from i-1 to i) on the target and template information around positions i-1 and i. The label features model the relationship between the state at position i and the target and template information around this position. Once the model parameters are determined, we can find the best sequence-template alignment by maximizing $P_\theta(a|s, t)$, which can be done using a dynamic programming algorithm since $P_\theta(a|s, t)$ only models state transition between two adjacent positions.

Our CRF-based threading model is more expressive than the generative threading models [21,41,42]. First, both the label and edge feature functions can be a nonlinear function, which can be used to capture the complex relationship among alignment state, sequence information and structural information. Second, the alignment state at position i may depend on sequence and structure information at positions around i instead of only at position i. So is the state transition between two adjacent alignment positions. Therefore, the CRF model

[2] We use L_a to denote the length of an alignment a in the case we need to differentiate the length of two alignments.

can accommodate complex feature sets that may be difficult to incorporate within a generative HMM model. The underlying reason is that CRFs only optimize the conditional probability $P_\theta(a|s, t)$ instead of joint probability $P_\theta(a, s, t)$, avoiding calculating the generative probability of the observations.

In Equation 2, the function potential $F(a, s, t, i)$ at position i is a linear combination of some features. This kind of linear parameterization implicitly assumes that all the features are linearly independent. This contradicts with the fact that the sequence and structure features are highly correlated. For example, the target secondary structure is usually predicted from PSI-BLAST sequence profile using PSIPRED [43]. The secondary structure is also correlated with primary sequence and solvent accessibility. To model interactions among features using Equation 2, one way is to explicitly define more complex features, each of which is a combination of some basic features, just like what [39] and [32] have done. However, explicitly defining complex features leads to a combinatorial explosion in the number of features, and hence, in the model complexity. The parameter training procedure for such a model needs careful tuning to avoid overfitting. In addition, explicit enumeration of features may also introduce a large number of unnecessary features, which greatly increases the computational time of finding the best sequence-template alignment.

2.2 Model Training Using Gradient Tree Boosting

Instead of explicitly enumerating thousands of complex features, we implicitly construct only a small number of important features using regression trees. In this method, the left hand side of Equation 2 is represented as a combination of regression trees instead of a linear combination of features. Each regression tree models the complex interactions among the basic features and each path from the tree root to a leaf corresponds to a single complex feature. We can build these regression trees automatically during the training process using the gradient boosting algorithm proposed by Dietterich *et al* [33]. Only those important features emerge as a path (from tree root to a leaf) in the trees. The resulting scoring function has the form of a linear combination of regression trees. One advantage of the gradient tree boosting approach is that it is unnecessary to explicitly enumerate all the complex features. The important features can be automatically learned during the training process. By contrast, explicit enumeration may not generate features as good as those learned by regression trees. Another advantage is that we may avoid overfitting because of the ensemble effect of combining multiple regression trees and much fewer complex features used. Finally, once the regression-tree-based threading model is trained, we can find the best alignment very quickly using dynamic programming since there are only dozens of regression trees in the scoring function.

To use the gradient tree boosting algorithm [33], we use a slightly different representation of Equations 1 and 2. Let $F^{a_i}(a_{i-1}, s, t)$ be a function that calculate the log-likelihood of the alignment state at position i given the alignment state at position $i-1$ and the target and the template information around positions $i-1$ and i. Then the CRF threading model has the form given by

$$P(a|s, t) = \frac{\exp\left(\sum_i F^{a_i}(a_{i-1}, s, t)\right)}{Z(s, t)} \quad (3)$$

In this new representation, there are no concepts of edge and label features. Instead, $\ln P(a|s, t)$ is a linear combination of some functions of form $F^{a_i}(a_{i-1}, s, t)$ where $F^{a_i}(a_{i-1}, s, t)$ is a linear combination of regression trees. To train such a model, we need to calculate the functional gradient of the conditional probability with respect to $F^{a_i}(a_{i-1}, s, t)$. Using a similar technique described in [33], we have the following result.

Lemma 1—Let u and v denote the alignment states at positions i−1 and i, respectively. The functional gradient of *ln* P (a|s, t) with respect to $F^{a_i}(a_{i-1},s,t)$ is given by

$$\frac{\partial \ln P(a|s,t)}{\partial F^v(u,s,t)} = I(a_{i-1}=u, a_i=v) - P(a_{i-1}=u, a_i=v|s,t) \quad (4)$$

Where $I(a_{i-1} = u, a_i = v)$ is a 0-1 function. Its value equals to 1 if and only if in the training alignment the state transition from i-1 to i is u → v. $P(a_{i-1} = u, a_i = v|s, t)$ is the predicted probability of the state transition u → v under current threading model.

The functional gradient in Equation 4 is easy to interpret. Given a training alignment, if the transition $u \rightarrow v$ is observed at position i, then ideally the predicted probability $P(a_{i-1} = u, a_i = v|s, t)$ should be 1 in order to make $\frac{\partial \ln P(a|s,t)}{\partial F^v(u,s,t)}$ be 0 and thus, to maximize $P(a|s, t)$. Similarly, if the transition is not observed, then the predicted probability should be 0 to maximize $P(a|s, t)$. Given an initial $F^v(u, ., .)$, to maximize $P(a|s, t)$, we need to move $F^v(u, ., .)$ along the gradient direction defined by all the $I(a_{i-1} = u, a_i = v) - P(a_{i-1} = u, a_i = v|s, t)$. Since $F^v(u, ., .)$ is a function taking as input the sequence and structure features around each alignment position, the gradient direction is also a function with the same input variables. We can use a function $T^v(u, ., .)$ to fit $I(a_{i-1} = u, a_i = v) - P(a_{i-1} = u, a_i = v|s, t)$ with the corresponding input values being the sequence and template features around positions $i - 1$ and i. Then $F^v(u, ., .)$ is updated by $F^v(u, ., .) + wT^v(u, ., .)$ where w is the step size and $T^v(u, ., .)$ is the gradient direction. The gradient tree boosting algorithm simply involves fitting regression trees to the difference between the observed and the predicted probabilities of each possible state transition. There are many possible functions that can fit a given set of data. Regression trees are chosen because they are easy to interpret and can be quickly trained from a large number of examples. In addition, we can also control the tree depth or the number of leaves to avoid overfitting.

Given a threading model and a training alignment, we can calculate $P(a_{i-1} = u, a_i = v|s, t)$ using a forward-backward method. Let $\alpha(v, i)$ and $\beta(v, i)$ denote the probabilities of reaching state v at position i, starting from the N-terminal and C-terminal of the alignment, respectively. Both $\alpha(v, i)$ and $\beta(v, i)$ can be recursively calculated as follows.

$$\alpha(v,1) = \exp F^v(\phi, s, t) \quad (5)$$

$$\alpha(v,i) = \sum_u [\exp F^v(u,s,t)]\,\alpha(u, i-1) \quad (6)$$

Where φ represents a dummy state.

$$\beta(u,L) = 1 \quad (7)$$

$$\beta(u,i) = \sum_v [\exp F^v(u,s,t)]\,\beta(v, i+1) \quad (8)$$

Then $P(a_{i-1} = u, a_i = v|s, t)$ can be calculated by

$$P(a_{i-1}=u, a_i=v|s,t) = \frac{\alpha(u,i-1)\,[\exp F^v(u,s,t)]\,\beta(v,i)}{Z(s,t)} \quad (9)$$

and the normalizer $Z(s, t)$ can be calculated by

$$Z(s,t) = \sum_u \alpha(u,0)\beta(u,0) \quad (10)$$

Given a set of training alignments, the gradient tree boosting algorithm to train the threading model is shown in Algorithm 1. The main component of the algorithm is to generate a set of examples to train a regression tree $T^v(u, ., .)$ for any feasible state transition $u \rightarrow v$. At any two adjacent positions of a training alignment, we generate an example by calculating $I(a_{i-1} = u, a_i = v) - P(a_{i-1} = u, a_i = v|s, t)$ as the response value and extracting sequence and structure features around these two positions as the input values. Then we fit a regression tree to these examples and update $F^v(u, ., .)$ accordingly.

There are some tricky issues in building the regression trees due to the extremely unbalanced number of positive and negative examples. A training example is positive if its response value is positive, otherwise negative. Given a training alignment with 200 residues in each protein and 150 aligned positions, the ratio between the number of positive examples and that of negative ones is approximately $\frac{200+200-150}{200\times 200\times 3}$. This will result in serious bias in regression tree training. We employed two strategies to resolve this issue. One is to add more weight to the positive examples and the other is that we randomly sample a small subset of negative examples for training [44].

Unlike the traditional CRF using $L2$ norm to regularize the model complexity and avoid overfitting [35], the complexity of our model is regularized by the tree depth. In building each regression tree, we use an internal 5-fold cross-validation procedure to determine the best tree depth. In our case the average tree depth is 4. Using such a regularization, we can avoid overfitting in training the model.

2.3 Sequence-Template Alignment Algorithm

Once a regression-tree-based threading model has been trained, we can find the best alignment a by maximizing $P(a|s, t)$ using a dynamic programming algorithm. This step is similar to all the HMM-based sequence alignment procedure. The best sequence-template alignment can be computed by the well-known Viterbi algorithm [45], which has the advantage that it does not need to compute the normalizer $Z(s, t)$.

2.4 Sequence and Structure Features

We use both evolutionary information and structure information to build regression trees for our threading model. We generate position specific score matrix (PSSM) for a template and position specific frequency matrix (PSFM) for a target using PSI-BLAST with five iterations and E-value 0.001. $PSSM(i, a)$ is the mutation potential for amino acid a at template position i and $PSFM(j, b)$ is the occurring frequency of amino acid b at target position j. The secondary structure and solvent accessibility of a template is calculated by the DSSP program [46]. For a target protein, we use PSIPRED [43] and SSpro [47] to predict its secondary structure and solvent accessibility, respectively.

Features for match state—We use the following features to build regression trees for a state transition to a match state. Suppose that template position i is aligned to target position j.

1. Sequence similarity. The sequence similarity score between two aligned positions is calculated by $\Sigma_a PSSM(i, a) \times PSFM(j, a)$.

2. Contact capacity score. The contact capacity potential describes the hydrophobic contribution of free energy, measured by the capability of a residue make a certain number of contacts with other residues in a protein. The two residues are in physical contact if the spatial distance between their C_β atoms is smaller than 8Å. Let $CC(a, k)$ denote the contact potential of amino acid a having k contacts (see Section 3 in [48]). The contact capacity score is calculated by $\Sigma_a CC(a, c) \times PSFM(j, a)$ where c is the number of contacts at template position i.
3. Environmental fitness score. This score measures how well to align one target residue to a template local environment, which is defined by a combination of three secondary structure types and three solvent accessibility states. Let $F(env, a)$ denote the environment fitness potential for amino acid a being in a local environment env (see Section 3 in [48]). The environment fitness score is given by $\Sigma_a F(env_i, a) \times PSFM(j, a)$
4. Secondary structure match score. Supposing the secondary structure type at template position i is ss, then the predicted likelihood of ss at target position j is used as the secondary structure match score.
5. Solvent accessibility match score. This is a binary feature used to indicate if the template position and the target position are in the same solvent accessibility state.

Features for gap state—The simplest scoring model for gap penalty is an affine function $o + e \times g$ where o is the gap open penalty, e gap extension penalty and g the number of gapped positions. To improve alignment accuracy, some threading programs such as SALIGN [49] also use a context-specific gap penalty function while others such as HHpred [21], SP5 [50] and the new PROSPECT [51] use a position-specific gap penalty model. In our threading model, we use a more sophisticated context-specific gap penalty function. Our future plan is to also incorporate position-specific gap penalty into our threading model. The regression trees for a state transition to an insertion state at the template depend on the following features: secondary structure type, solvent accessibility, amino acid identity and hydropathy count [39]. Similarly, the regression trees for a state transition to an insertion state at the target depend on the following features: predicted secondary structure likelihood scores, predicted solvent accessibility, amino acid identity and hydropathy count.

3 Results

3.1 Model Training

Similar to CONTRAlign [39], our boosting-based threading model does not need a large data set for training. The alignment accuracy on the validation set does not increase with respect to the training set size as long as it is at least 30. We arbitrarily choose 30 protein pairs from the PDB as our training set and 40 pairs as the validation set. The average size of a protein contains 200 residues. In the training set, 20 pairs are in the same fold level but different superfamily level according to the SCOP classification [52]. The other 10 pairs are in the same superfamily but different family level. Any two proteins in the training and validation set have sequence identity less than 30%. Reference alignments are built by the structure alignment program TM-align [53]. We also guarantee that the proteins used for model training have no high sequence identity (30%) with the proteins in the Prosup and SALIGN benchmarks (see Section 3.2).

As shown in Figure 1, the training process runs very fast. It takes approximately two minutes for each training iteration and achieves very good alignment accuracy after only six or seven iterations. The alignment accuracy reaches the best value after 21 iterations. More training iterations do not improve alignment accuracy but result in more regression trees.

The more regression trees are used in the threading model, the more running time will be spent aligning a protein pair. As a result, we choose the model trained after 21 iterations as our final threading model. For each state transition, the model has twenty-one regression trees with an average depth four.

3.2 Alignment Accuracy

To compare our method with other state-of-art threading programs, we evaluate them on two popular benchmarks: Prosup[54] and SALIGN [49]. The Prosup benchmark [54] has 127 protein pairs with structural alignment generated by Prosup. The SALIGN benchmark [49] contains 200 protein pairs. On average, two proteins in a pair share 20% sequence identity and 65% of structurally equivalent C_α atoms superposed with RMSD 3.5Å.

We used TM-align [53] to generate structural alignments for the SALIGN benchmark. The SALIGN benchmark is more difficult than the Prosup benchmark and includes many pairs of proteins with very different sizes. To evaluate the alignment quality, we use the exact match accuracy which is computed as the percentage of one-to-one match positions in the benchmark pairs. We also evaluate the 4-offset match accuracy, which is defined as the percentage of the matches within 4 positions shift from one-to-one match.

Table 1 compares the performance of various alignment methods on the Prosup benchmark. Our method, denoted as BoostThreader, shows a significant improvement over the others. The absolute improvement over SP3/SP5, a leading threading program, is more than 5%. The major difference between our method and SP3/SP5 is that SP3/SP5 linearly combines various sequence and structure features as its scoring function while our method uses a nonlinear scoring function. CONTRAlign [39] is run locally with the default hydropathy model. CONTRAlign mainly aims at sequence alignment, so it is not surprising that its performance is not as competitive as some leading threading methods. The results of other methods are taken from [55,50,41].

Also as shown in the right three columns of Table 1, our method also has the best alignment accuracy on the SALIGN benchmark. This benchmark contains many pairs of proteins with very different sizes, which is the major reason why RAPTOR performs badly on this benchmark.

3.3 Fold Recognition Rate

We also evaluate the fold recognition rate of our new method BoostThreader on the Lindahl's benchmark [56], which contains 976 proteins. Any two proteins in this set share less than 40% sequence identity. All-against-all threading of these proteins can generate 976 × 975 pairs. After generating the alignments of all the pairs using our regression-tree-based threading model, we rank all the templates for each sequence using a method similar to [48] and then evaluate the fold recognition rate of our method. When evaluating the performance in the superfamily level, all the templates similar in the family level are ignored. Similarly, when evaluating the performance in the fold level, all the templates similar in the superfamily or family level are ignored. “Top 1” means that the only the first-ranked templates are evaluated while “Top 5” indicates that the best templates out of the top 5 are evaluated. As shown in Table 2, our method performs well in all three similarity levels. The fold recognition rate of our new method is much better than SP3/SP5, HHpred and RAPTOR, especially in the superfamily and fold levels. These three programs performed very well in recent CASP (Critical Assessment of Structure Prediction) events.

4 Conclusion

This paper has presented a new protein threading model based on conditional random fields and gradient tree boosting. By using regression trees to represent the scoring function, this CRF-based threading model can accommodate as many sequence and structure features as possible and accurately model their interactions. Our model can also capture complex feature interactions without introducing a large number of redundant features. Although complex, such a scoring function still can be efficiently optimized by a dynamic programming algorithm. It takes less than half a second to thread a typical protein pair. Experimental results also demonstrate that by carefully account for the interdependency among features, we can greatly improve alignment accuracy over other leading threading programs. The improved alignment accuracy also leads to the improvement of fold recognition rate.

Currently, our threading model only considers state transition between two adjacent positions. A straight-forward generalization is to model state dependency among three adjacent positions. We cam also model pairwise interaction between two nonadjacent positions. The challenge of modeling non-local interactions is that it is computationally hard to train and test such a model. Some approximation algorithms may be resorted. It is also possible to use tree decomposition [25] or linear programming [22,23] to train such a model efficiently.

Acknowledgments

This work is supported by TTI-C internal research funding and NIH research grant. The authors are grateful to Liefeng Bo and Kristian Kersting for their help with the gradient tree boosting technique. We also thank Chuong Do for his help with the CONTRAlign software.

References

1. Kihara D, Skolnick J. The PDB is a covering set of small protein structures. Journal of Molecular Biology. 2003; 334(4):793–802. [PubMed: 14636603]
2. Zhang Y, Skolnick J. The protein structure prediction problem could be solved using the current PDB library. Proceedings of National Academy Sciences, USA. 2005; 102(4):1029–1034.
3. Jones DT. Progress in protein structure prediction. Current Opinion in Structural Biology. 1997; 7(3):377–387. [PubMed: 9204280]
4. Rost B. Twilight zone of protein sequence alignments. Protein Engineering. 1999; 12:85–94. [PubMed: 10195279]
5. John B, Sali A. Comparative protein structure modeling by iterative alignment model building and model assessment. Nucleic Acids Research. 2003; 31(14):3982–3992. [PubMed: 12853614]
6. Chivian, Dylan; Baker, David. Homology modeling using parametric alignment ensemble generation with consensus and energy-based model selection. Nucleic Acids Research. 2006; 34(17):e112. [PubMed: 16971460]
7. Marko AC, Stafford K, Wymore T. Stochastic Pairwise Alignments and Scoring Methods for Comparative Protein Structure Modeling. Journal of Chemical Information and Modeling. March. 2007
8. Jaroszewski L, Rychlewski L, Li Z, Li W, Godzik A. FFAS03: a server for profile–profile sequence alignments. Nucleic Acids Research. July.2005 33 Web Server issue.
9. Rychlewski L, Jaroszewski L, Li W, Godzik A. Comparison of sequence profiles. Strategies for structural predictions using sequence information. Protein Science. 2000; 9(2):232–241. [PubMed: 10716175]
10. Yona G, Levitt M. Within the twilight zone: a sensitive profile-profile comparison tool based on information theory. Journal of Molecular Biology. 2002; (315):1257–1275. [PubMed: 11827492]

11. Pei J, Sadreyev R, Grishin NV. PCMA: fast and accurate multiple sequence alignment based on profile consistency. Bioinformatics. 2003; 19(3):427–428. [PubMed: 12584134]
12. Marti-Renom MA, Madhusudhan MS, Sali A. Alignment of protein sequences by their profiles. Protein Science. 2004; 13(4):1071–1087. [PubMed: 15044736]
13. Ginalski K, Pas J, Wyrwicz LS, von Grotthuss M, Bujnicki JM, Rychlewski L. ORFeus: Detection of distant homology using sequence profiles and predicted secondary structure. Nucleic Acids Research. 2003; 31(13):3804–3807. [PubMed: 12824423]
14. Zhou H, Zhou Y. Single-body residue-level knowledge-based energy score combined with sequence-profile and secondary structure information for fold recognition. Proteins: Structure, Function, and Bioinformatics. 2004; 55(4):1005–1013.
15. Han S, Lee B-C, Yu ST, Jeong C-S, Lee S, Kim D. Fold recognition by combining profile-profile alignment and support vector machine. Bioinformatics. 2005; 21(11):2667–2673. [PubMed: 15769835]
16. Shi J, Blundell TL, Mizuguchi K. FUGUE: sequence-structure homology recognition using environment-specific substitution tables and structure-dependent gap penalties. Journal of Molecular Biology. 2001; 310(1):243–257. [PubMed: 11419950]
17. Jones DT. GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences. Journal of Molecular Biology. 1999; 287(4):797–815. [PubMed: 10191147]
18. Kelley LA, MacCallum RM, Sternberg MJ. Enhanced genome annotation using structural profiles in the program 3D-PSSM. Journal of Molecular Biology. 2000; 299(2):499–520. [PubMed: 10860755]
19. Zhou H, Zhou Y. Fold recognition by combining sequence profiles derived from evolution and from depth-dependent structural alignment of fragments. Proteins: Structure, Function, and Bioinformatics. 2005; 58(2):321–328.
20. Karplus K, Barrett C, Hughey R. Hidden Markov Models for Detecting Remote Protein Homologies. Bioinformatics. 1998; 14(10):846–856. [PubMed: 9927713]
21. Johannes S. Protein homology detection by HMM-HMM comparison. Bioinformatics. 2005; 21(7): 951–960. [PubMed: 15531603]
22. Xu J, Li M, Lin G, Kim D, Xu Y. Protein threading by linear programming. The Pacific Symposium on Biocomputing. 2003:264–275.
23. Xu J, Li M, Kim D, Xu Y. RAPTOR: optimal protein threading by linear programming. Journal of Bioinformatics and Computational Biology. 2003; 1(1):95–117. [PubMed: 15290783]
24. Xu J, Li M. Assessment of RAPTOR's linear programming approach in CAFASP3. Proteins: Structure, Function and Genetics. 2003
25. Xu J, Jiao F, Berger B. A tree-decomposition approach to protein structure prediction. Proceedings of IEEE Computational Systems Bioinformatics Conference. 2005:247–256.
26. Rai BK, Fiser A. Multiple mapping method: a novel approach to the sequence-to-structure alignment problem in comparative protein structure modeling. Proteins: Structure, Function, and Bioinformatics. 2006; 63(3):644–661.
27. Wu S, Zhang Y. MUSTER: Improving protein sequence profile-profile alignments by using multiple sources of structure information. Proteins: Structure, Function, and Bioinformatics. 2008; 9999(9999) NA+
28. Wu S, Skolnick J, Zhang Y. Ab initio modeling of small proteins by iterative TASSER simulations. BMC Biology. 2007; 5 17+
29. Silva PJ. Assessing the reliability of sequence similarities detected through hydrophobic cluster analysis. Proteins: Structure, Function, and Bioinformatics. 2008; 70(4):1588–1594.
30. Skolnick J, Kihara D. Defrosting the frozen approximation: PROSPECTOR - a new approach to threading. Proteins: Structure, Function, and Genetics. 2001; 42(3):319–331.
31. Kim D, Xu D, Guo J, Ellrott K, Xu Y. PROSPECT II: Protein structure prediction method for genome-scale applications. Protein Engineering. 2002
32. Yu CN, Joachims T, Elber R, Pillardy J. Support vector training of protein alignment models. Journal of Computational Biology. 2008; 15(7):867–880. [PubMed: 18707536]

33. Dietterich, TG.; Ashenfelter, A.; Bulatov, Y. Training Conditional Random Fields via Gradient Tree Boosting; Proceedings of the 21st International Conference on Machine Learning (ICML); 2004. p. 217-224.
34. Lafferty, J.; McCallum, A.; Pereira, F. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data; ICML: Proc. 18th International Conf. on Machine Learning; San Francisco: Morgan Kaufmann; 2001. p. 282-289.
35. Sha F, Pereira F. Shallow parsing with conditional random fields. Proceedings of Human Language Technology NAACL 2003. 2003:134–141.
36. Shen, R. Protein secondary structure prediction using conditional random fields and profiles. Department of Computer Science, Oregon State University; 2006. Master Thesis
37. Lafferty, J.; Zhu, X.; Liu, Y. Kernel Conditional Random Fields: Representation and Clique Selection; ICML 2004: Proceedings of the twenty-first international conference on Machine learning; New York: ACM Press; 2004.
38. Zhao F, Li S, Sterner BW, Xu J. Discriminative learning for protein conformation sampling. Proteins: Structure, Function, and Bioinformatics. 2008; 73(1):228–240.
39. Do C, Gross S, Batzoglou S. CONTRAlign: Discriminative Training for Protein Sequence Alignment. 2006
40. Mcguffin LJ, Bryson K, Jones DT. The PSIPRED protein structure prediction server. Bioinformatics. 2000; 16(4):404–405. [PubMed: 10869041]
41. Qiu J, Elber R. SSALN: An alignment algorithm using structure-dependent substitution matrices and gap penalties learned from structurally aligned protein pairs. Proteins: Structure, Function, and Bioinformatics. 2006; 62(4):881–891.
42. Karplus K, Karchin R, Shackelford G, Hughey R. Calibrating E-values for Hidden Markov Models using Reverse-Sequence Null Models. Bioinformatics. 2005; 21(22):4107–4115. [PubMed: 16123115]
43. Jones DT. Protein secondary structure prediction based on position-specific scoring matrices. Journal of Molecular Biology. 1999; 292(2):195–202. [PubMed: 10493868]
44. Gutmann, B.; Kersting, K. Stratified Gradient Boosting for Fast Training of Conditional Random Fields; Proceedings of the 6th International Workshop on Multi-Relational Data Mining; p. 56-68.
45. Rabiner LR. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE. 1990:267–296.
46. Kabsch W, Sander C. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers. 1983; 22(12):2577–2637. [PubMed: 6667333]
47. Pollastri G, Baldi P, Fariselli P, Casadio R. Prediction of coordination number and relative solvent accessibility in proteins. Proteins: Structure, Function, and Genetics. 2002; 47(2):142–153.
48. Xu J. Fold Recognition by Predicted Alignment Accuracy. IEEE/ACM Transaction of Computational Biology and Bioinformatics. 2005; 2(2):157–165.
49. Marti-Renom MA, Madhusudhan MS, Sali A. Alignment of protein sequences by their profiles. Protein Science. 2004; 13(4):1071–1087. [PubMed: 15044736]
50. Zhang W, Liu S, Zhou Y. SP5: Improving protein fold recognition by using torsion angle profiles and profile-based gap penalty model. PLoS ONE. 2008; 3(6)
51. Ellrott K, Guo JT, Olman V, Xu Y. Improvement in protein sequence-structure alignment using insertion/deletion frequency arrays. Computational systems bioinformatics / Life Sciences Society. Computational Systems Bioinformatics Conference. 2007; 6:335–342. [PubMed: 17951836]
52. Murzin AG, Brenner SE, Hubbard T, Chothia C. SCOP: a structural classification of proteins database for the investigation of sequences and structures. Journal of Molecular Biology. 1995; 247(4):536–540. [PubMed: 7723011]
53. Zhang Y, Skolnick J. TM-align: a protein structure alignment algorithm based on the TM-score. Nucleic Acids Research. 2005; 33(7):2302–2309. [PubMed: 15849316]
54. Lackner P, Koppensteiner WA, Sippl MJ, Domingues FS. ProSup: a refined tool for protein structure alignment. Protein Engneering. 2000; 13(11):745–752.
55. Liu S, Zhang C, Liang S, Zhou Y. Fold Recognition by Concurrent Use of Solvent Accessibility and Residue Depth. Proteins: Structure, Function, and Bioinformatics. 2007; 68(3):636–645.

56. Lindahl E, Elofsson A. Identification of related proteins on family, superfamily and fold level. Journal of Molecular Biology. 2000; 295(3):613–625. [PubMed: 10623551]

57. Cheng J, Baldi P. A machine learning information retrieval approach to protein fold recognition. Bioinformatics. 2006; 22(12):1456–1463. [PubMed: 16547073]

function TreeBoostThreader(Data) // $Data = \{(a^j, s^j, t^j)\}$ where j indicates the j^{th} alignment example. L^j is the length of the j^{th} alignment.
for all state transition $u \rightarrow v$ **do**
initialize $F_0^v(u, ., .) = 0$ //$u \rightarrow v$ is the feasible state transition at two adjacent positions
//the second variable of $F_0^v(u, ., .)$ is the features extracted from the target protein
//the third variable of $F_0^v(u, ., .)$ is the structure features from the template protein
end for
//training at most M iterations
for m from 1 to M **do**
for all state transition $u \rightarrow v$ **do**
S(u,v):=GenerateExamples(u, v, Data)
$T_m(u, v)$:=FitRegressionTree(S(u,v))
$F_m^v(u, ., .) := F_{m-1}^v(u, ., .) + T_m(u, v)$
end for
end for
return $F_M^v(u, s, t)$ as $F^v(u, ., .)$ for all $u \rightarrow v$
end function

function GenerateExamples(u,v,Data)
for all training alignments **do**
for all i from 1 to L^j **do**
Calculate α and β according to Equations 6 and 8
for all state transition $u \rightarrow v$ **do**
Calculate $P(u, v|s, t)$ using Equation 9
$\delta(i, u, v, s, t) = I(a_{i-1} = u, a_i = v) - P(a_{i-1} = u, a_i = v|s, t)$
//δ is the response value to be fitted by a regression tree
//$s(i)$ and $t(i)$ are the sequence and structure features around position i
insert an example data entry ($s(i)$, $t(i)$, δ) into $S(u, v)$
end for
end for
end for
return $S(u, v)$
end function

Algorithm 1.
Gradient Tree Boosting Training Algorithm

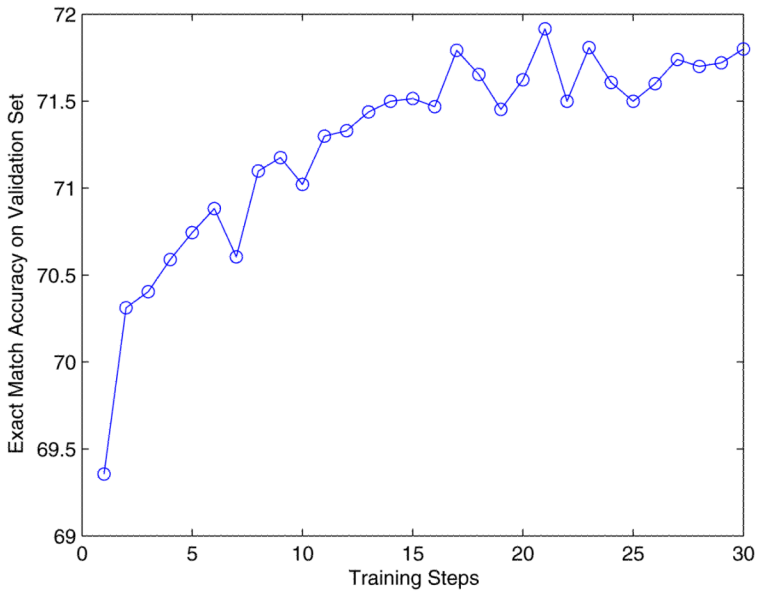

Fig. 1.
The alignment accuracy of the models on the validation data set during the whole training process

Table 1

Alignment accuracy (%) of our method BoostThreader and other alignment methods on the Prosup and SALIGN benchmarks

Prosup			SALIGN		
Methods	**Exact**	**4-offset**	**Methods**	**Exact**	**4-offset**
CONTRAlign	52.79	69.42	CONTRAlign	44.38	57.37
PSIBLAST	35.60		PSIBLAST	26.10	
SPARKS	57.20		SPARKS	53.10	
SSALGN	58.30		SALIGN	56.40	
RAPTOR	61.30	79.32	RAPTOR	40.20	59.80
SP3	65.30	82.20	SP3	56.30	56.60
SP5	68.70		SP5	59.70	
BoostThreader	**74.05**	**88.90**	BoostThreader	**63.60**	**79.01**

Table 2

Fold recognition rate (%) of various threading programs. The PSI-BLAST, SPARKS, SP3, SP5 and HHpred results are taken from [50]. The FOLDpro, HMMER, FUGUE, SAM-98 results are from [57]. The RAPTOR and PROSPECT-II results are from [48].

Methods	Family		Superfamily		Fold	
	Top1	Top5	Top1	Top5	Top1	Top5
PSIBLAST	71.2	72.3	27.4	27.9	4.0	4.7
HMMER	67.7	73.5	20.7	31.3	4.4	14.6
SAM-98	70.1	75.4	28.3	38.9	3.4	18.7
THREADER	49.2	58.9	10.8	24.7	14.6	37.7
FUGUE	82.2	85.8	41.9	53.2	12.5	26.8
PROSPECT-II	84.1	88.2	52.6	64.8	27.7	50.3
SPARKS	81.6	88.1	52.5	69.1	24.3	47.7
SP3	81.6	86.8	55.3	67.7	28.7	47.4
FOLDpro	85.0	89.9	55.5	70.0	26.5	48.3
SP5	81.6	87.0	59.9	70.2	37.4	58.6
HHpred	82.9	87.1	58.8	70.0	25.2	39.4
RAPTOR	**86.6**	89.3	56.3	69.0	38.2	**58.7**
BoostThreader	86.5	**90.5**	**66.1**	**76.4**	**42.6**	57.4