Training Protein Threading Models using Structural SVMs

Chun-Nam John Yu CNYU@CS.CORNELL.EDU
Thorsten Joachims TJ@CS.CORNELL.EDU
Ron Elber RON@CS.CORNELL.EDU
Dept. of Computer Science, Cornell University, Ithaca, NY, USA

Abstract

Protein threading is the problem of inferring the structure of a protein from its sequence by matching the sequence against a set of known structures. Unlike conventional sequence to sequence alignment tasks, alignment models for threading can exploit a rich set of features derived from the geometry of the known structure. To make use of these complex and interdependent features, we explore the use of discriminative training with structural Support Vector Machines. We present empirical results for the CASP5 dataset and compare against conventional generative training.

1. Introduction

Protein Structure is essential for understanding the mechanism of many biological processes. With the increased availability of experimentally determined protein structures, comparative modelling techniques for inferring the structure of new proteins is gaining in attractiveness. An important step in comparative modelling is the alignment of an unknown target protein sequence to one or more templates of known structures. This alignment problem is challenging for structurally similar proteins that nevertheless have low sequence similarity. Aligning protein sequences in this low sequence similarity region (below 25%, usually referred to as the "twilight zone") is difficult for traditional sequence alignment algorithm that uses subsitution matrices like BLOSUM or PAM.

Numerous studies show that alignment accuracy within the "twilight zone" can be impoved by including extra information (structure geometry, profile, secondary structures, etc.). However, this leads to an increase in the number of parameters in the alignment model. Parameter tuning by hand becomes very difficult, and traditional generative estimation techniques are challenged by the statistical dependencies between the features. In this paper we propose the use of a discriminative method based on structural SVM (Tsochantaridis et al., 2005) for learning the parameters of protein alignment models from a set of training examples. Unlike approaches based on CRF (Lafferty et al., 2001, McCallum et al., 2005, Do et al., 2006), our method allows us to explicitly optimize application specific loss functions. The method provides a well-founded way of including large number of features, and it allows us to build flexible and highly complex alignment model without having to assume conditional independence between features. Experimental results on the CASP5 data shows that our method is competitive with some of the best existing algorithms.

2. Basic Proteomics and Terminology

Proteins are sequences of amino acids, typically several hundreds long. There are 20 common amino acids, each with different physio-chemical properties. Proteins fold into a stable shape under their usual chemical environments, and their structure determines their function. Understanding how proteins fold is one of the central problems in biology. While the genome projects provide us with sequence information for a large number of proteins (DNA codes amino acids sequence), current methods for determining the structure of proteins experimentally are expensive and time-consuming. Therefore, it is desirable to predict the structure of proteins from the amino acid sequence alone.

Comparative modelling of proteins is based on the idea that similar amino acid sequences fold into similar shapes. Suppose a sequence of amino acids of unknown structure is given, which from here on we refer to as the target sequence. A search is performed on this new target sequence in a database of protein sequences with known structures, and proteins that we believe to be structurally similar to the target sequence are selected. Then the target sequence is aligned against all the templates, and structural models for the target sequence are produced from these alignments.

Appearing in *Proceedings of the ICML Workshop on Learning in Structured Output Spaces*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

To produce good structural models, not only do the correct templates need to be found, but the alignments between the target sequence and templates also need to be as accurate as possible.

Alignment errors between target and template is a major source of errors for target-template pairs with less than 30% sequence similarity. In this study we attempt to improve alignment accuracy by building more complex models of alignments using extra information. The three features that we are going to use in this study are residue (amino acid), secondary structure, and relative exposed surface area (solvent accessibility). For the target sequence with no structural annotation, secondary structure and solvent accessibilty are predicted using the program SABLE (Adamczak et al., 2004).

3. Sequence Alignment

Following discussions from the last section, we begin with introducing notations to formuate the alignment problem. Let $x = (x^{ta}, x^{te})$ be a pair of target and template sequences (we will use the abbreviations ta and te for targets and templates extensively in this paper). For an alignment y of the pair (x^{ta}, x^{te}), we write y as a sequence of alignment operations $(y^1, y^2, ..., y^k)$. Each y^j is an alignment operation of the form (a, b), where a, b are one of the 20 amino acids or the special gap character '$-$'.

We consider alignment algorithms that optimizes a linear scoring function $D_{\vec{w}}(y) = \vec{w} \cdot \Psi(y)$where Ψ is a function that maps the alignment y to a feature vector, and $\vec{w}$ is a given cost vector that parameterizes the scoring function D. Furthermore, we require that $\Psi(y)$ be linear in the individual alignment operations y^j in y. To be precise,

$$\Psi(y) = \sum_{j=1}^{length(y)} \phi(y^j) \tag{1}$$

where ϕ is a function that maps each individual alignment operation onto the feature space. To compute the highest scoring alignment between $x = (x^{ta}, x^{te})$, we compute

$$\underset{y \in Align(x)}{\operatorname{argmax}} [\vec{w} \cdot \Psi(y)] = \underset{y \in Align(x)}{\operatorname{argmax}} \left[\vec{w} \cdot \sum_{j=1}^{length(y)} \phi(y^j) \right] \tag{2}$$

where $Align(x)$ is the set of all possible local alignments between x^{ta} and x^{te}. This is typically computed using the Smith-Waterman dynamic programming algorithm. Note that our setting includes the common scenarios of alignment with substitution matrices such as BLOSUM, where the function ϕ maps the alignment operation $y^j = (a, b)$ to one of the 400 substitution costs (or the gap cost). In this study, however, we consider richer feature mappings that include structural information.

4. Learning the Alignment Model

In the above section, the cost vector $\vec{w}$ parameterizes the scoring scheme and has great influence over the quality of alignments between target and template sequences. The commonly used substitution matrices are estimated using log-likelihoods of aligned protein blocks, while the gap parameters are usually hand-tuned. We aim to learn all parameters automatically from a training set of "gold standard" — or ideal — alignments (these alignments are either produced manually or computed by programs using 3D coordinates). This is known as the inverse alignment problem. Below we present an algorithm that learns a cost vector $\vec{w}$ from a set of ideal training alignments. The approach is discriminative and tries to maximize the score difference (margin) between the ideal alignment and alternative alignments.

4.1. Formulation of the Learning Problem

We denote the training set of ideal alignments as $Z = ((x_1, y_1), (x_2, y_2), ..., (x_n, y_n))$. Each training pattern x_i is a pair of target and template sequences, and we write $x_i = (x_i^{ta}, x_i^{te})$. Each label y_i(the ideal alignment) is a sequence of pairs $y_i = (y_i^1, y_i^2, ..., y_i^k)$, where each alignment operation y_i^j is a pair of aligned residues or a gap aligned with a residue.

In the framework of structural SVMs (Tsochantaridis et al., 2005), training the parameters can be formulated as the following optimization problem (Joachims et al., 2005).

$$\begin{aligned} \min_{\vec{w}, \vec{\xi}} \quad & \frac{1}{2}\|\vec{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ s.t. \quad & \forall y \in Y_i \backslash \{y_i\} : \vec{w} \cdot (\Psi(y_i) - \Psi(y)) \geq \Delta(y_i, y) - \xi_i \end{aligned} \tag{3}$$

The objective is the conventional regularized risk used in SVMs. The constraints state that the score $\vec{w} \cdot \Psi(y_i)$ of the ideal alignment y_i must be greater than the score $\vec{w} \cdot \Psi(y)$ of all alternative alignments y by a difference of $\Delta(y_i, y)$. Δ is a loss function that measures how different the two alignments y_i and y are. Intuitively, the larger the loss, the further should the score be away from that of the ideal alignment. ξ_i is a slack variable shared among constraints from the same example, since in general the problem is not separable. Note that $\sum \xi_i$ is an upper bound on the training loss. We will discuss the design of suitable feature vectors Ψ and loss function Δ in Section 4.3, a task in which both biological knowledge and algorithmic considerations play a role.

4.2. Learning Algorithm

Given any two sequences of length n and m, the number of possible alignments between them is exponential in n, m. The number of constraints in optimization problem (3) is

```
Input: pairs of target and template sequences $(x_1^{ta}, x_1^{te}), ..., (x_n^{ta}, x_n^{te})$, ideal alignments $y_1, ..., y_n$,
    tolerated error $\epsilon \geq 0$.

$K = \emptyset, \vec{w} = 0, \vec{\xi} = 0$
repeat
    • $K_{org} = K$
    • for $i$ from 1 to $n$
        – $\hat{y} = argmax_{y \in Y_i \setminus y_i} [\Delta(y_i, y) + \vec{w} \cdot (\Psi(y) - \Psi(y_i))]$
          via dynamic programming
        – if $\vec{w} \cdot (\Psi(y_i) - \Psi(y)) < \Delta(y_i, y) - \xi_i - \epsilon$
            * $K = K \cup \{\vec{w} \cdot (\Psi(y_i) - \Psi(y)) \geq \Delta(y_i, y) - \xi_i - \epsilon\}$
            * $(\vec{w}, \vec{\xi}) = argmin_{\vec{w}, \vec{\xi}} \frac{1}{2}\|\vec{w}\|^2 + \frac{C}{n}\sum_{i=1}^{n} \xi_i$
              subject to $K$.
until $(K = K_{org})$
Output: $\vec{w}$
```

Figure 1. Sparse Approximation Algorithm for the Alignment Prediction task.

huge and it is infeasible to solve the quadratic program directly. However, it has been shown that the cutting plane algorithm in Figure 1 can be used to efficiently approximate the optimal solution of this type of optimization problem (Tsochantaridis et al., 2005, Joachims et al., 2005). The algorithm starts with an empty set of constraints, adds the most violated constraint among the exponentially many during each iteration, and repeats until the desired precision $\epsilon > 0$ is reached. It can be proved that only a polynomial number of constraints will be added before convergence (Tsochantaridis et al., 2005, Joachims et al., 2005). One crucial aspect of the algorithm, however, is the use of an oracle which can pick out the most violated constraint among the exponentially many in polynomial time. That is, we need to compute

$$\underset{y \in Y_i \setminus y_i}{\operatorname{argmax}} [\Delta(y_i, y) + \vec{w} \cdot (\Psi(y) - \Psi(y_i))]. \qquad (4)$$

For our problem of sequence alignment we have already assumed that the feature mapping Ψ is linear in the individual alignment operations. If the loss function Δ is also linear, then we can use a variant of the Smith-Waterman algorithm to find the most violated constraint efficiently. The running time of the overall learning algorithm is then polynomial in the number of training examples, the length of the sequences, and ϵ (Tsochantaridis et al., 2005, Joachims et al., 2005).

4.3. Alignment Model

We now turn to the important issue of designing the feature vectors and loss functions. Since the loss function has less variety than the feature vector, we would discuss the loss function first.

4.3.1. Loss Function

A natural measure of loss is the number of incorrect alignment operations. However, since we are dealing with local alignments, we are more interested in obtaining more correctly aligned residue pairs than avoiding extraneous pairs. Moreover, we are more interested in getting the 'match' operations rather than the 'gap' operations correct since the matches tell us something biologically meaningful and are useful for structural modelling. These properties are reflected in the commonly used Q score, which is the number of correct 'match' operations S in y (i.e., matches in both y and y_i), over the number of 'match' operations S_i in the reference alignment y_i.

$$\Delta_Q(y, y_i) = 1 - \frac{|S_i \cap S|}{|S_i|} \qquad (5)$$

Note that this loss function based on the Q score is linear in the alignment operations, making it accessible to dynamic programming. We also consider a less stringent version of this loss function (called the Q4 loss), which counts a correct match for two residues if they align within a window of size 4 (which is acceptable for structural modelling).

4.3.2. Substitution Model

We want to define a linear model to determine the cost of aligning target residue R_{ta} with template residue R_{te}, with the following extra information:

1. S_{ta}, predicted secondary structure of target
2. S_{te}, true secondary structure of template
3. A_{ta}, predicted solvent accessibility of target
4. A_{te}, true solvent accessibility of template

The usual alignment operations are of the form (a, b), where a is a target residue and b is a template residue. Now we consider tuples of features, for $a = (R_{ta}, S_{ta}, A_{ta})$, $b = (R_{te}, S_{te}, A_{te})$ All the predicted secondary structures and solvent accessibility for the targets are generated by the SABLE program (Adamczak et al., 2004), while the true secondary structures and relative exposed surface area for the templates are computed using DSSP. In the following discussion on feature vectors, we use the notation $s(X; Y)$ to denote a score measuring how compatible the structures X from the target and Y from the template are, where X and Y can be any of the basic attributes mentioned above, or any combination of them.

(I) SIMPLE Commonly used substitution matrices like BLOSUM consider only the identity of the residue, and the substitution score just consist of a single constant $s(R_{ta}; R_{te})$. Instead, we consider scores of the form $s(R_{ta}; R_{te}) + s(S_{ta}; S_{te}) + s(A_{ta}; A_{te})$ which take into account the compatibility of secondary structure, exposed surface area at the two sites as well. It is also meaningful to consider the alignment cost between a residue and a surface area type $s(R_{ta};A_{te})$, since we know that some residues are hydrophobic and tend to stay buried in the core of the protein while some other residues are polar and prefer to be exposed to water. This leads us to the following substitution cost function:

$$\begin{aligned} &s(R_{ta};R_{te}) + s(R_{ta};S_{te}) + s(R_{ta};A_{te}) \\ +&s(S_{ta};R_{te}) + s(S_{ta};S_{te}) + s(S_{ta};A_{te}) \\ +&s(A_{ta};R_{te}) + s(A_{ta};S_{te}) + s(A_{ta};A_{te}) \end{aligned} \tag{6}$$

Alternatively, this alignment cost can be written nicely in algebraic notations. If we overload the notation and consider $\vec{R_{ta}}$ as a binary indicator feature vector of dimension 20 for the residue R_{ta}, $\vec{S_{ta}}$ of dimension 3 for the predicted secondary structure, $\vec{A_{ta}}$ of dimension 5 for the predicted surface area(we bin the relative exposed surface area(0-100%) into 5 bins), and likewise for the corresponding features in the template. Then the feature vector can be written out as:

$$(\vec{R_{ta}} \oplus \vec{S_{ta}} \oplus \vec{A_{ta}}) \otimes (\vec{R_{te}} \oplus \vec{S_{te}} \oplus \vec{A_{te}}) \tag{7}$$

where $\oplus$ is the direct sum operation and $\otimes$ is the tensor product operation over vector spaces.

(II) ANOVA2 Our next proposal is to also consider the pairwise combination of features. For example, we can have a cost of $s(R_{ta} \wedge S_{ta}; R_{te} \wedge S_{te})$, which determines the score of aligning a site with residue type R_{ta} and predicted secondary structure S_{ta} of the target sequence with another site with residue type R_{te} and true secondary structure S_{te} of the template sequence. We add all possible alignment costs for pairs of basic features. In particular, we consider

$$\begin{aligned} &((\vec{R_{ta}} \otimes \vec{S_{ta}}) \oplus (\vec{S_{ta}} \otimes \vec{A_{ta}}) \oplus (\vec{A_{ta}} \otimes \vec{R_{ta}})) \\ \otimes&((\vec{R_{te}} \otimes \vec{S_{te}}) \oplus (\vec{S_{te}} \otimes \vec{A_{te}}) \oplus (\vec{A_{te}} \otimes \vec{R_{te}})) \end{aligned} \tag{8}$$

(III) SIMPLE+ANOVA2 We consider the direct sum of the two feature vectors in (7) and (8), since it could be advantageous to learn from a simpler feature vector before learning from more complicated features. In terms of algebraic notations it is:

$$\begin{aligned} &((\vec{R_{ta}} \oplus \vec{S_{ta}} \oplus \vec{A_{ta}}) \oplus (\vec{R_{ta}} \otimes \vec{S_{ta}}) \oplus (\vec{S_{ta}} \otimes \vec{A_{ta}}) \oplus (\vec{A_{ta}} \otimes \vec{R_{ta}})) \\ \otimes&((\vec{R_{te}} \oplus \vec{S_{te}} \oplus \vec{A_{te}}) \oplus (\vec{R_{te}} \otimes \vec{S_{te}}) \oplus (\vec{S_{te}} \otimes \vec{A_{te}}) \oplus (\vec{A_{te}} \otimes \vec{R_{te}})) \end{aligned} \tag{9}$$

(IV) SIMPLE+ANOVA2+WINDOW3 Finally, it might be informative to include information about sites nearby when aligning R_{ta}^{i} with R_{te}^{j}; for example, R_{ta}^{i-1}, R_{ta}^{i+1} and R_{te}^{j-1}, R_{te}^{j+1}, if we consider a window of size 3. Specifically, we add three extra score terms to the feature vector in (9).

$$\begin{aligned} &s(R_{ta}^{i-1} \wedge R_{ta}^{i} \wedge R_{ta}^{i+1}; R_{te}^{j-1} \wedge R_{te}^{j} \wedge R_{te}^{j+1}) \\ +&s(S_{ta}^{i-1} \wedge S_{ta}^{i} \wedge S_{ta}^{i+1}; S_{te}^{j-1} \wedge S_{te}^{j} \wedge S_{te}^{j+1}) \\ +&s(A_{ta}^{i-1} \wedge A_{ta}^{i} \wedge A_{ta}^{i+1}; A_{te}^{j-1} \wedge A_{te}^{j} \wedge A_{te}^{j+1}) \end{aligned} \tag{10}$$

To reduce dimensionality, we group the set of amino acids into 7 equivalence classes according to their physiochemical properties ($\{K,R,H\}$, $\{D,E\}$, $\{I,V,M,L\}$, $\{C\}$, $\{P,S,A,G,T\}$, $\{F,W,Y\}$, $\{N,Q\}$) using protein alphabet compression (Taylor, 1986, Wang & Wang, 1999) when considering the window feature for residues.

4.3.3. Gap Model

In addition to using gap opening and gap extenstion costs, we make the gap costs dependent on the specific environment. Our gap model follows very closely from the one used in (Qiu & Elber, 2006). First, consider the situation of gaps in the target sequence. Suppose when aligning R_{ta} against R_{te}, there is a gap between R_{ta}^{n} and R_{ta}^{n+1}, and the gap character is aligned to R_{te}^{k} of the template. Denoting the position in the sequence with super-script, the cost of opening a gap between R_{ta}^{n} and R_{ta}^{n+1}, with the gap aligned to R_{te}^{k}, is:

$$\begin{aligned} &g(R_{ta}^{n};R_{ta}^{n+1}) + g(S_{ta}^{n};S_{ta}^{n+1}) + g(A_{ta}^{n};A_{ta}^{n+1}) \\ +&g(R_{te}^{k}) + g(S_{te}^{k}) + g(A_{te}^{k}) \end{aligned} \tag{11}$$

where the dyadic terms $g(X_{ta}^{n};X_{ta}^{n+1})$ measures how easy it is to open a gap between the structure type X_{ta}^{n} and X_{ta}^{n+1} in the target. The intuition behind such a comples gap-model is that, for example, when X is the secondary structure, and X_{ta}^{n} and X_{ta}^{n+1} are both α-helix, then it would be unfavorable to open a gap between the two sites. However if X_{ta}^{n} and X_{ta}^{n+1} are of different secondary structure types or they are in the loop region, then opening a gap is more permissible. The monadic term $g(Y)$ measures how compatible a particular template structure is with a gap at that site. The cost of gap openning at the template is similarly defined, with the role of target and template reversed in the above definition.

5. Experiments

We used data from a representative set developed by the computational biology group at Cornell (Qiu & Elber, 2006). After cleaning up missing values and removing examples which the LOOPP server cannot produce structural annotations, we have 3169 ideal alignments in total, as produced by the CE program based on the structures of the target and template proteins. Each target-template pair in the training set shared high structural similarities with CE Z score greater than 4.5.

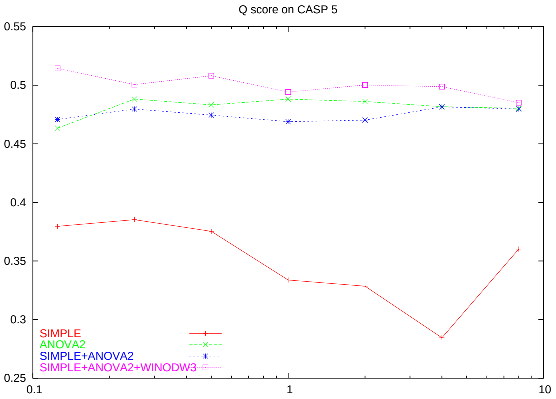

Figure 2. Q score on CASP5 test set.

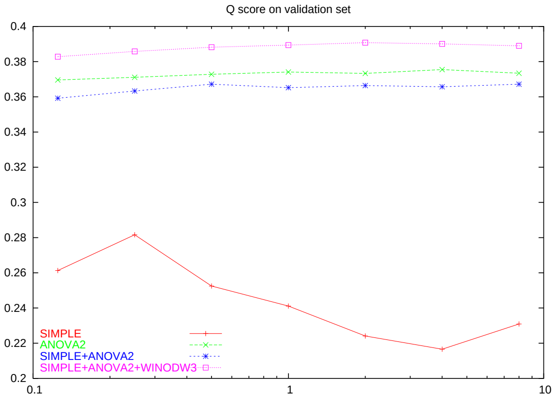

Figure 3. Q score on validation set.

We test our models using sequences from the CASP5 competition as targets. A set of suitable templates are identified using the LOOPP server. In the CASP5 set we have 101 aligned pairs from 30 target sequences. Most of the alignments in this test set fall within the twilight zone, with all but 18 pairs having sequence similarity below 25%. We trained models for the 4 feature vectors with a different values of C and precision $\epsilon = 0.01$. Our preliminary findings are as follows.

Figure 2 show the Q score of our method on CASP5 with the 4 different feature vectors. The horizontal axis is the regularization parameter C, which we train from 2^{-3} to 2^{3}, in powers of 2. The general trend seems to be that the more complex models perform better than the less complex feature vectors. In particular, including neighborhood information from the size 3 window seems to be beneficial.

Figure 3 shows the Q score of our method on another set of examples which we call the 'validation set'. The set contains 3882 examples and was used in the study (Qiu & Elber, 2006) for parameter tunning. It is constructed in a similar manner as the training set, and is independent of the training set in the sense that no target sequence appears in both training and validation set. The relative performance of the different feature vectors are similar to those in the CASP5 set, but the curve is slightly smoother due to the larger test set size. There are differences in the absolute performances on the CASP5 and validation sets due to differences in their distribution of protein sequences in terms of sequence similarities and protein families.

Table 1. Q scores on CASP5. The numbers of the first three rows are computed from the alignments in study in (Qiu & Elber, 2006). The number in brackets are the best performances mentioned in that paper, using a slightly larger CASP5 set with 117 pairs.

Method	Q-Score
BLOSUM50 (GAP Open/Ext=10/1)	27.13 (27.1)
PSI-BLAST	35.97 (34.7)
SSALN	47.86 (51.3)
SIMPLE ($C = 0.25$)	38.53
ANOVA2 ($C = 4$)	48.17
SIMPLE+ANOVA2 ($C = 8$)	47.97
SIMPLE+ANOVA2+WINDOW3 ($C = 2$)	50.02

Table 1 compares the performance of our method with existing approaches. The performance on CASP5 test set reported in table 1 is trained with the value of C that gives the best performance over a validation set of size 3882 mentioned above. SSALN (Qiu & Elber, 2006) is a generative method that incorporates structural information into the subsitution matrices, and are trained using the same feature set as our algorithm does. SSALN was trained on about 5000 examples, while our method was trained on only the subset of 3169 examples for which we had structural annotations in our database. As baselines, we included the performances of BLOSUM and PSI-BLAST. We observe that the incorporation of structural information (i.e. our method and SSALN) boosts the alignment accuracy by a substantial amount. Our method is competitive with SSALN, showing similar performance on the CASP5 set. However, we achieve this performance without any need for manual tuning of parameters.

So far we have assumed that our examples of proteins alignments are independent and identically distributed. In reality the alignment examples are not independent, since each target sequence usually has more than one structurally similar templates, which implies that the templates themselves are also structurally similar. In addition, our training and test sets are mixtures of classes of proteins which vary in average length and composition of secondary structures. We performed a set of simple experiments to investigate the effect of separating these classes in training our alignment models. The SCOP classification (Murzin et al., 1995) is a hierarchical classification of protein structures into classes, folds, superfamilies and families. Proteins within the same family share more structural similarities than proteins within the same superfamily, which share more structural similarities than proteins within the same fold, which in turn share more structural similarities than proteins within the same class. We used the SCOP classification to divide the training set (3169 examples) into 4 major classes at the top of the SCOP hierarchy according to the class of the target sequence: all alpha proteins (SCOP A, 648 examples), all beta proteins (SCOP B, 1161 examples), alpha and beta proteins (SCOP C, 1032 examples), alpha plus beta proteins (SCOP D, 129 examples). Proteins within the same class have similar secondary structure composition. Examples outside these 4 classes are discarded. We repeat the process for the validation set (3882 examples) to obtain 4 test sets of different classes (505 examples for SCOP A, 1786 examples for SCOP B, 959 examples for SCOP C, 213 examples for SCOP D). In this way we have 4 different train sets divided by SCOP class, 4 different test sets also divided by SCOP class, and the relative size of train sets and test sets by class are similar.

We trained alignments models on the 4 train sets individually and tested them on the 4 test sets. We used the feature vector with best performance "SIMPLE+ANOVA2+WINDOW3" in training the models, for values of C in the range 2^{-1} to 2^{3}. We found that the performance of models using different values of C to be rather close (within Q score of 1), and so we only report the performance for $C = 2$ in Table 2. The rows represent the training sets while the columns represent the test sets. We observe that the table is diagonally dominant, suggesting that training and testing on the same SCOP class gives better performance. The effect is particularly marked in SCOP B for all-beta-proteins, since no other class could achieve a performance close to it on the SCOP B test set. The last row is the performance of a model trained with the whole set of 3169 examples using the same feature vector and same value of C. The performance of that model on each test set is close but slightly worse than the best in the same column, i.e., those models which are trained and tested using the training and test sets from the same SCOP class. The difference in performance is consistently maintained across the range of C (2^{-1} to 2^{3}) that we explored.

It seems that information on the SCOP classes of the proteins gives a small advantage in training alignment models. The next natural problem to investigate is to go down the SCOP hierarchy and look at whether knowing the SCOP fold of a protein would help us train better alignment models. These experiments could help us understand how the non-identitically-distributed nature of the data affects the performance of our alignment models, perhaps suggesting ways to utilize these information to increase alignment accuracy in future works.

Table 2. Q score on validation set by SCOP class split, C=2

			test	on	
		SCOP A	SCOP B	SCOP C	SCOP D
	SCOP A	34.59	26.64	23.04	45.65
train	SCOP B	31.69	47.9	29.35	53.59
on	SCOP C	24.27	22.44	31.4	43.42
	SCOP D	31.55	30.68	29.06	52.26
	All	30.92	46.51	30.31	52.69

6. Conclusions and Future Work

We have explored the use of large-margin training for building alignment models for protein threading. The potential benefits of such an approach are the ability to learn complex models in a well-founded way, the ability to optimize to application specific loss functions, and the ability to set the gap parameters without need for hand-tuning. Our initial experiments show that our method is competitive with a state-of-the-art generative learning method, even without yet realizing its full potential.

(Do et al., 2006) considered training alignment models with conditional random field, and obtained excellent results compared to traditional generative models. Although the focus of their study is different from ours, it will be interesting to compare our SVM approach to their CRF approach and understand the differences between these two discriminative methods.

In this work we focused on improving the quality of alignment for comparative modelling of proteins. An interesting problem closely related to the current study is homology detection by alignment. The goal of homology detection is to find structurally similar proteins of a target sequence from a database of known protein structures, and the score used are usually the dynamic programming score or some normalized versions of it. The quality of alignment itself in this problem is not important as long as the score could give us good discriminating power over whether or not two proteins are structurally related. Our large-margin approach could be also adapted for this problem by optimizing over a suitable metric instead of the Q score, and we believe the ability to incorporate many structural features would be helpful in this problem. This would be an interesting direction for further research.

We are also investigating how to relax the i.i.d. assumption over the 'golden standard alignments' used in training. Since each target sequence usually has more than one structurally similar template, it is wasteful to ignore this relation because the templates themselves are going to be structurally similar as well. One way to tackle this problem would be to train models to do multiple alignments directly, but we need to change the dynamic programming algorithm to approximate alignment algorithms and perform the corresponding parameter estimation. Our experiments on splitting training and test sets by SCOP classes show the non-identically-distributed nature of the data, and more work is needed to investigate its full effect on the performance of alignment models. It would be a major challenge to model the dependencies between of the different proteins sequences and their distribution in the fold space due to low sequence similarity, but a proper modelling of these dependencies could increase the alignment accuracy further.

Acknowledgements

We thank Dr. Jaroslaw Pillardy for his help with the data and his many suggestions on the project. We also thank the anonymous reviewers for their comments. This work is supported by NIH Grants IS10RR020889, GM67823 and by the NSF Award IIS-0412894.

References

Adamczak, R., Porollo, A., & Meller, J. (2004). Accurate prediction of solvent accessibility using neural networks-based regression. *Proteins*, *56*, 753–67.

Do, C. B., Gross, S. S., & Batzoglou, S. (2006). CONTRAlign: Discriminative training for protein sequence alignment. *RECOMB 2006*.

Joachims, T., Galor, T., & Elber, R. (2005). Learning to align sequences: A maximum-margin approach. In B. Leimkuhler et al. (Ed.), *New algorithms for macromolecular simulation*, vol. 49 of *LNCS*, 57–68. Springer.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: probabilistic modeling for segmenting and labeling sequence data. *ICML 2001*.

McCallum, A., Bellare, K., & Pereira, F. (2005). A conditional random field for discriminatively-trained finite-state string edit distance. *UAI 2005*.

Murzin, A. G., Brenner, S. E., Hubbard, T., & Chothia, C. (1995). SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology (JMB)*, *247*, 536–540.

Qiu, J., & Elber, R. (2006). SSALN: an alignment algorithm using structure-dependent substitution matrices and gap penalties learned from structurally aligned protein pairs. *Proteins*, *62*, 881–91.

Taylor, W. R. (1986). Classification of amino acid conservation. *Journal of Theoretical Biology*, *119*, 205–258.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, *6*, 1453 – 1484.

Wang, J., & Wang, W. (1999). A computational approach to simplifying the protein folding alphabet. *Nature Structural Biology*, *6(11)*, 1033–1038.