Learning to Align Sequences: A Maximum-Margin Approach

Thorsten Joachims
Department of Computer Science
Cornell University
Ithaca, NY 14853
tj@cs.cornell.edu

August 28, 2003

Abstract

We propose a discriminative method for learning the parameters (e.g. cost of substitutions, deletions, insertions) of linear sequence alignment models from training examples. While the resulting training problem leads to an optimization problem with an exponential number of constraints, we present a simple algorithm that finds an arbitrarily close approximation after considering only a subset of the constraints that is linear in the number of training examples and polynomial in the length of the sequences. We also evaluate empirically that the method effectively learns good parameter values while being computationally feasible.

1 Introduction

Methods for sequence alignment are common tools for analyzing sequence data ranging from biological applications [3] to natural language processing [1]. They can be thought of as measures of similarity between sequences where the similarity score is the result of a discrete optimization problem that is typically solved via dynamic programming. While the dynamic programming algorithm determines the general notion of similarity (e.g. local alignment vs. global alignment), any such similarity measure requires specific parameter values before it is fully specified. Examples of such parameter values are the costs for substituting one sequence elements for another, as well as costs for deletions and insertions. These parameter values greatly determine how well the measure works for a particular task.

In this paper we tackle the problem of inferring the parameter values from training data. Our goal is to find parameter values so that the resulting similarity measure best reflects the desired notion of similarity. We consider training data where we have examples of similar and dissimilar sequences. Instead of assuming a generative model of sequence alignment (e.g. [9]), we take a discriminative approach to training. In particular, we aim to find the set of parameter values that corresponds to the best similarity

s_1:	5	2	9	2	1	3	5	5		2	1	9	2	1		
s_2:			3	2	1	2	5	5	3	2	1	4	7	6	5	2
a:				m	m	s	m	m	i	m	m					

Figure 1: Example of a local sequence alignment.

measure a given alignment algorithm can represent. Taking a large-margin approach, we show that we can solve the resulting training problem efficiently for a large class of alignment algorithms that implement a linear scoring function. While the resulting optimization problems have exponentially many constraints, our algorithm finds an arbitrarily good approximation after considering only a subset of constraints that scales polynomially with the length of the sequences and linearly with the number of training examples. We empirically and theoretically analyze the scaling of the algorithm and show that the learned similarity score performs well on test data.

2 Sequence Alignment

Sequence alignment computes a similarity score for two (or more) sequences s_1 and s_2 from an alphabet $\Sigma = \{1, .., \sigma\}$. An alignment a is a sequence of operations that transforms one sequence into the other. In global alignment, the whole sequence is transformed. In local alignment, only an arbitrarily sized subsequence is aligned. Commonly used alignment operations are "match" (m), "substitution" (s), "deletion" (d) and "insertion" (i). An example of a local alignment is given in Figure 1. In the example, there are 6 matches, 1 substitution, and 1 insertion/deletion. With each operation there is an associated cost/reward. Assuming a reward of 3 for match, a cost of -1 for substitution, and a cost of -2 for insertion/deletion, the total alignment score $D_{\vec{w}}(s_1, s_2, a)$ in the example is 15. The optimal alignment a^* is the one that maximizes the score for a given cost model.

More generally, we consider alignment algorithms that optimize a linear scoring function $D_{\vec{w}}(s_1, s_2, a) = \vec{w}^T \phi(s_1, s_2, a)$ where $\phi(s_1, s_2, a)$ is some function that generates features based on an alignment a (e.g. # of substitutions, # of deletions) and w is a given cost vector. $\phi(s_1, s_2, a)$ depends on the particular alignment algorithm and can be any feature vector. Finding the optimal alignment corresponds to the following optimization problem

$$D_{\vec{w}}(s_1, s_2) = max_a \left[D_{\vec{w}}(s_1, s_2, a)\right] = max_a \left[\vec{w}^T \phi(s_1, s_2, a)\right]. \quad (1)$$

This type of problem is typically solved via dynamic programming. In the following we consider local alignment via the Smith/Waterman algorithm [10]. However, the results can be extended to any alignment algorithm that optimizes a linear scoring function and that solves (1) globally optimally[1].

[1]This also holds for other structures besides sequences.

3 Inverse Sequence Alignment

Inverse sequence alignment is the problem of using training data to learn the parameters $\vec{w}$ of an alignment model and algorithm so that the resulting similarity measure $D_{\vec{w}}(s_1, s_2)$ best represents the desired notion of similarity on new data. While previous approaches to this problem exist [5, 11], they are limited to special cases and very small numbers of parameters. We will present an algorithm that applies to any linear alignment model with no restriction on the function or number of parameters.

We assume the following model, for which the notation is inspired by protein alignment. In protein alignment the goal is a similarity measure so that homologous protein sequences are similar to a native sequence, but also so that non-homologous (i.e. decoy) sequences are not similar to the native sequence. We assume that examples are generated i.i.d. according to a distribution $P(s^N, s^H, S^D)$. s^N is the native sequence, s^H the homologous sequence, and S^D is a set of decoy sequences $s^{D_1}, ..., s^{D_d}$. The goal is a similarity measure so that native sequence s^N and homolog s^H are more similar than the native sequence s^N and any decoy s^{D_j}, i.e.

$$D_{\vec{w}}(s^N, s^H) > D_{\vec{w}}(s^N, s^{D_j}). \tag{2}$$

The goal of learning can be formulated in two reasonable ways based on two different loss functions. In the first setting, the goal is to find the cost parameters $\vec{w}$ that minimize the probability $Err_P^{\Delta}(\vec{w})$ that the similarity with any decoy sequence $D_{\vec{w}}(s^N, s^{D_j})$ is higher than the similarity with the homolog sequence $D_{\vec{w}}(s^N, s^H)$.

$$Err_P^{\Delta}(\vec{w}) = \int \Delta\left(\max_{s^{D_j} \in S^D} D_{\vec{w}}(s^N, s^{D_j}) > D_{\vec{w}}(s^N, s^H)\right) dP(s^N, s^H, S^D) \tag{3}$$

The function $\Delta(.)$ returns 1, if its argument is false, 0 otherwise. In the second setting, the goal is less ambitious. We do not want a homolog that is necessarily more similar than *all* decoys. Instead, we merely want to have the homolog sequence ranked highly, but not necessarily on top of the ranking. In particular, we could optimize the average rank of the homolog. This is equivalent to the following kind of error rate.

$$Err_P^{\tau}(\vec{w}) = \int \sum_{s^{D_j} \in S^D} \Delta\left(D_{\vec{w}}(s^N, s^{D_j}) > D_{\vec{w}}(s^N, s^H)\right) dP(s^N, s^H, S^D) \tag{4}$$

In the following, we consider only the first type of error $Err_P^{\Delta}(\vec{w})$ for conciseness.

4 A Maximum-Margin Approach to Learning the Cost Parameters

The distribution $P(s^D, s^H, S^D)$ is unknown. However, we have a training sample S from $P(s^N, s^H, S^D)$. It consists of native sequences $s_1^N, ..., s_n^N$, homolog sequences $s_1^H, ..., s_n^H$, and a set of decoy sequences $S_1^D, ..., S_n^D$ for each native $s_1^N, ..., s_n^N$. As a simplifying assumption, we assume that between native and homolog sequences the alignment $a_1^{NH}, ..., a_n^{NH}$ of maximum score is known[2]. The goal is to find an optimal

[2]For protein alignment, for example, this could be generated via structural alignment.

$\vec{w}$ so that the error rate $Err_P^{\Delta}(\vec{w})$ is low. First, we consider the case where there exists a $\vec{w}$ such that the error on the training set

$$Err_S^{\Delta}(\vec{w}) = \sum_{i=1}^{n} \Delta \left(\max_{s_i^{D_j} \in S_i^D} D_{\vec{w}}(s_i^N, s_i^{D_j}) < D_{\vec{w}}(s_i^N, s_i^H) \right) \tag{5}$$

is zero. Since we assume a scoring function that is linear in the parameters

$$D_{\vec{w}}(s_1, s_2, a) = \vec{w}^T \phi(s_1, s_2, a), \tag{6}$$

the condition of zero training error can be written as a set of linear inequality constraints. There is one constraint for each combination of native sequence s_i^N, decoy sequence $s_i^{D_j}$, and possible alignment a of s_i^N into $s_i^{D_j}$.

$$\begin{aligned} \forall s_1^{D_j} \in S_1^D \forall a: \quad & D_{\vec{w}}(s_1^N, s_1^{D_j}, a) < D_{\vec{w}}(s_1^N, s_1^H, a_i^{NH}) \\ \cdots & \\ \forall s_n^{D_j} \in S_n^D \forall a: \quad & D_{\vec{w}}(s_n^N, s_n^{D_j}, a) < D_{\vec{w}}(s_n^N, s_n^H, a_i^{NH}) \end{aligned} \tag{7}$$

This approach of writing the training problem as a linear system follows the method in [8] proposed for the special case of global alignment without free insertions/deletions. However, for the general case in Equation (7) the number of constraints is exponential, since the number of alignments a between s_i^N and $s_i^{D_j}$ is exponential in the length of s_i^N and $s_i^{D_j}$. Unlike the restricted case in [8], standard optimization algorithms cannot handle this size problem. To overcome this limitation, in Section 5 we will propose an algorithm that exploits the special structure of Equation (7) so that it needs to examine only a subset that is at most linear in the number of training examples n.

If the set of inequalities in Equation (7) is feasible, there will typically be more than one solution $\vec{w}^*$. To specify a unique solution, we select the $\vec{w}$ for which the similarity of the homolog $D_{\vec{w}}(s_n^N, s_n^H, p_i^{NH})$ is uniformly most different from the similarity $max_{s_i^{D_j} \in S_i^D} D_w(s_i^N, s_i^{D_1})$ of the best alignment with any decoy. This corresponds to the maximum-margin principle employed in Support Vector Machines [12]. Denoting the margin by δ and restricting the L_2 norm of $\vec{w}$ to make the problem well-posed, this leads to the following optimization problem.

$$max_{\vec{w}} \quad \frac{1}{2}\delta^2 \tag{8}$$

$$\forall s_1^{D_j} \in S_1^D \forall a: \quad D_{\vec{w}}(s_1^N, s_1^{D_j}, a) \leq D_{\vec{w}}(s_1^N, s_1^H, a_i^{NH}) - \delta \tag{9}$$

$$\cdots \tag{10}$$

$$\forall s_n^{D_j} \in S_n^D \forall a: \quad D_{\vec{w}}(s_n^N, s_n^{D_j}, a) \leq D_{\vec{w}}(s_n^N, s_n^H, a_i^{NH}) - \delta \tag{11}$$

$$||\vec{w}|| = 1 \tag{12}$$

Due to the linearity of the similarity function (6), the length of $\vec{w}$ is a free variable and we can fix it to $1/\delta$. Substituting for δ and rearranging leads to the equivalent optimization problem

$$min_{\vec{w}} \quad \frac{1}{2}\vec{w}^T\vec{w} \tag{13}$$

$$\forall s_1^{D_j} \in S_1^D \forall a: \qquad \left(\phi(s_1^N, s_1^H, a_i^{NH}) - \phi(s_1^N, s_1^{D_j}, a)\right) \vec{w} \geq 1 \tag{14}$$

$$\ldots \tag{15}$$

$$\forall s_n^{D_j} \in S_n^D \forall a: \qquad \left(\phi(s_n^N, s_n^H, a_i^{NH}) - \phi(s_n^N, s_n^{D_j}, a)\right) \vec{w} \geq 1 \tag{16}$$

Since this quadratic program (QP) has a positive-definite objective function and (feasible) linear constraints, it is strictly convex. This means it has a unique global minimum and no local minima [4]. The constraints are similar to the ordinal regression approach in [6] and it has a structure similar to the Ranking SVM described in [7] for information retrieval. However, the number of constraints is much larger.

To allow errors in the training set, we introduce slack variables ξ_i [2]. Corresponding to the error measure $Err_P^{\Delta}(\vec{w})$ we have one slack variable for each native sequence. This is different from a normal classification or regression SVM, where there is a different slack variable for each constraint. The slacks enter the objective function according to a trade-off parameter C. For simplicity, we consider only the case where the slacks enter the objective function squared.

$$min_{\vec{w},\vec{\xi}} \qquad \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^{n}\xi_i^2 \tag{17}$$

$$\forall s_1^{D_j} \in S_1^D \forall a: \qquad \left(\phi(s_1^N, s_1^H, a_i^{NH}) - \phi(s_1^N, s_1^{D_j}, a)\right) \vec{w} \geq 1 - \xi_1 \tag{18}$$

$$\ldots \tag{19}$$

$$\forall s_n^{D_j} \in S_n^D \forall a: \qquad \left(\phi(s_n^N, s_n^H, a_i^{NH}) - \phi(s_n^N, s_n^{D_j}, a)\right) \vec{w} \geq 1 - \xi_n \tag{20}$$

Note that $\sum_{i=1}^{n}\xi_i^2$ is an upper bound on the training error $Err_S^{\Delta}(\vec{w})$. Therefore, the algorithm minimizes the training error $Err_S^{\Delta}(\vec{w})$ while maximizing margin.

5 Training Algorithm

Due to the exponential number of constraints, standard optimization software cannot handle the number of constraints resulting from problems of interesting size. However, by exploiting the special structure of the problem, we propose the following algorithm that finds the solution of (17)-(20) after examining only a small number of constraints. The algorithm proceeds by greedily adding constraints from (18)-(20) to a working set K. The algorithm stops, when all constraints in (18)-(20) are fulfilled up to a precision of ϵ.

Input: native sequences $s_1^N, ..., s_n^N$, homolog sequences $s_1^H, ..., s_n^H$, alignments $a_1^{NH}, ..., a_n^{NH}$, sets of decoy sequences $S_1^D, ..., S_n^D$, tolerated approximation error $\epsilon \geq 0$.

$K = \emptyset$, $\vec{w} = 0$, $\vec{\xi} = 0$

repeat

- $K_{org} = K$
- for i from 1 to n
  - for j from 1 to $|S_i^D|$
    - find $a^* = argmax_a \left[\vec{w}^T \phi(s_i^N, s_i^{D_j}, a)\right]$ via dynamic programming
    - if $\vec{w}^T(\phi(s_i^N, s_i^H, a_i^{NH}) - \phi(s_i^N, s_i^{D_j}, a^*)) < 1 - \xi_i - \epsilon$
      - $K = K \cup \left\{ \vec{w}^T(\phi(s_i^N, s_i^H, a_i^{NH}) - \phi(s_i^N, s_i^{D_j}, a^*)) \geq 1 - \xi_i \right\}$
      - solve QP $(\vec{w}, \vec{\xi}) = argmin_{\vec{w},\vec{\xi}} \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^n \xi_i^2$ subject to K.

until($K = K_{org}$)

Output: $\vec{w}$

The following two theorems show that the algorithm returns the correct solution and that it stops after $O(n)$ iterations through the repeat-loop.

Theorem 1 (CORRECTNESS)
The algorithm returns an approximation to (17)-(20) that has an objective value not higher than the solution of (17)-(20), and that fulfills all constraints up to a precision of ϵ. For $\epsilon = 0$, the algorithm returns the solution $(\vec{w}^*, \vec{\xi^*})$ of (17)-(20).

Proof *Let $\vec{w}^*$ and ξ_i^* be the solution of (17)-(20). Since the algorithm solves the QP on a subset of the constraints in each iteration, it must return a solution $\vec{w}$ with $\frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^n \xi_i^2 \leq \frac{1}{2}\vec{w}^{*T}\vec{w}^* + C\sum_{i=1}^n \xi_i^{*2}$. This follows from the fact that decreasing the feasible region cannot lead to a lower minimum.*

It is left to show that the algorithm does not terminate before all constraints (18)-(20) are fulfilled up to precision ϵ. In the final iteration, the algorithm finds the highest scoring alignment a^* for each decoy and checks whether the constraint $\vec{w}^T(\phi(s_i^N, s_i^H, a_i^{NH}) - \phi(s_i^N, s_i^{D_j}, a^*)) < 1 - \xi_i - \epsilon$ is fulfilled. Three cases can occur: First, the constraint can be violated and the algorithm will not terminate yet. Second, it is already in K and is fulfilled by construction. Third, the constraint is not in K but fulfilled anyway and it is not added. If the constraint is fulfilled, the constraints for all other alignments into this decoy are fulfilled as well, since we checked the constraint for which the margin was smallest for the given $\vec{w}$. It follows that the algorithm terminates only if all constraints are fulfilled up to precision ϵ. ∎

Similarly, it can be shown that for $\epsilon > 0$ the algorithm finds a solution so that the KKT-conditions of the Wolfe Dual of (17)-(20) are fulfilled up to a precision of ϵ. We omit the proof for brevity.

Theorem 2 (TERMINATION)
The algorithm stops after adding at most

$$\frac{2VR^2}{\epsilon^2} \tag{21}$$

constraints to the set K. V is the minimum of (17)-(20) and R^2 is a constant bounded by the maximum of $(\phi(s_i^N, s_i^H, a_i^{NH}) - \phi(s_i^N, s_i^{D_j}, a))^2 + \frac{1}{2C}$.

Proof *The first part of the proof is to show that the objective value increases by some constant with every constraint that is added to K. Denote with V_k the solution $V_k = P(\vec{w}_k^*, \vec{\xi}_k^*) = min_{\vec{w},\vec{\xi}} \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^n \xi_i^2$ subject to K_k after adding k constraints. This primal optimization problem can be transformed into an equivalent problem of the form $V_k = P(\vec{w}_k'^*) = min_{\vec{w}'} \frac{1}{2}\vec{w}'^T\vec{w}'$ subject to K_k', where each constraint has the form $\vec{w}^T\vec{x} \geq 1$ with $\vec{x} = (\phi(s_i^N, s_i^H, a_i^{NH}) - \phi(s_i^N, s_i^{D_j}, a^*); 0; ...; 0; 1/\sqrt{2C}; 0; ...; 0)$. Its corresponding Wolfe dual is $D(\vec{\alpha}_k^*) = max_{\vec{\alpha}\geq 0} \sum_{i=1}^k \alpha_i - \frac{1}{2}\sum_{i=1}^k\sum_{j=1}^k \alpha_i\alpha_j\vec{x}_i\vec{x}_j$. At the solution $D(\vec{\alpha}_k^*) = P(\vec{w}_k'^*) = P(\vec{w}_k^*, \vec{\xi}_k^*) = V_k$ and for every feasible point $D(\vec{\alpha}) \leq P(\vec{w}, \vec{\xi})$. Primal and dual are connected via $\vec{w}'^* = \sum_{i=1}^k \alpha_i^*\vec{x}_i$. Adding a constraint to the dual with $\vec{w}'^{*T}\vec{x}_{k+1} = \sum_{i=1}^k \alpha_i^*\vec{x}_i\vec{x}_{k+1} \leq 1-\epsilon$ means extending the dual to*

$$\begin{aligned}
D_{k+1}(\vec{\alpha}_{k+1}^*) = & \max_{\vec{\alpha}_{k+1}\geq 0} \sum_{i=1}^k \alpha_i - \frac{1}{2}\sum_{i=1}^k\sum_{j=1}^k \alpha_i\alpha_j\vec{x}_i\vec{x}_j + \alpha_{k+1} - \alpha_{k+1}\sum_{i=1}^k \alpha_i\vec{x}_i\vec{x}_{k+1} - \frac{1}{2}\alpha_{k+1}^2\vec{x}_{k+1}^2 \\
\geq & \quad D_k(\vec{\alpha}_k^*) + \max_{\alpha_{k+1}\geq 0} \alpha_{k+1} - \alpha_{k+1}\sum_{i=1}^k \alpha_i^*\vec{x}_i\vec{x}_{k+1} - \frac{1}{2}\alpha_{k+1}^2\vec{x}_{k+1}^2 \\
\geq & \quad D_k(\vec{\alpha}_k^*) + \max_{\alpha_{k+1}\geq 0} \alpha_{k+1} - \alpha_{k+1}(1-\epsilon) - \frac{1}{2}\alpha_{k+1}^2\vec{x}_{k+1}^2
\end{aligned}$$

Solving the remaining scalar optimization problem over α_{k+1} shows that $\alpha_{k+1}^* \geq 0$ and that $V_{k+1} \geq V_k + \frac{\epsilon^2}{2R^2}$.

Since the algorithm only adds constraints that are violated by the current solution by more than ϵ, after adding $k_{max} = \frac{2VR^2}{\epsilon^2}$ constraints the solution $V_{k_{max}}$ over the subset $K_{k_{max}}$ is at least $V_{k_{max}} \geq V_0 + \frac{2VR^2}{\epsilon^2}\frac{\epsilon^2}{2R^2} = 0 + V$. Any additional constraint that is violated by more than ϵ would lead to a minimum that is larger than V. Since the minimum over a subset of constraints can only be smaller than the minimum over all constraints, there cannot be any more constraints violated by more than ϵ and the algorithm stops. ∎

The theorem directly leads to the conclusion that the maximum number of constraints in K scales linearly with the number of training examples n, since V can be upper bounded as $V \leq C * n$ using the feasible point $\vec{w} = 0$ and $\vec{\xi} = 1$ in (17)-(20). Furthermore, it scales only polynomially with the length of the sequences, since R is polynomial in the length of the sequences.

While the number of constraints can potentially explode for small values of ϵ, experience with Support Vector Machines for classification showed that relatively large

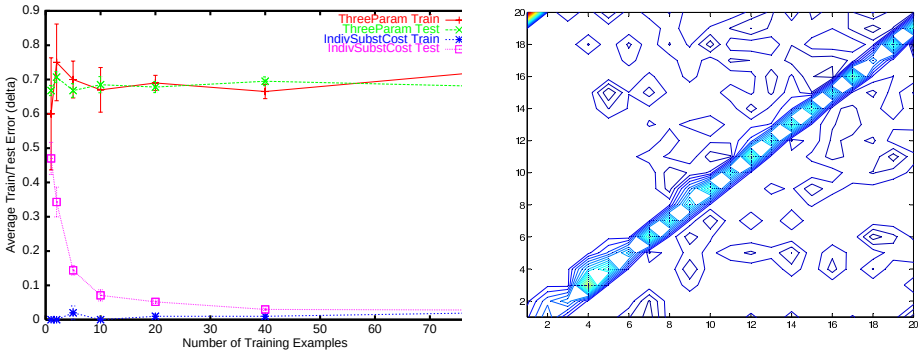

Figure 2: Left: Train and test error rates for the 3 and the 403 parameter model depending on the number of training examples. Right: Typical learned substitution matrix after 40 training examples for the 403-parameter model.

values of ϵ are sufficient without loss of generalization performance. We will verify the efficiency and the prediction performance of the algorithm empirically in the following.

6 Experiments

To analyze the behavior of the algorithm under varying conditions, we constructed a synthetic dataset according to the following sequence and alignment model. The native sequence and the decoys are generated by drawing randomly from a 20 letter alphabet $\Sigma = \{1, .., 20\}$ so that letter $c \in \Sigma$ has probability $c/210$. Each sequence has length 50, and there are 10 decoys per native. To generate the homolog, we generate an alignment string of length 30 consisting of 4 characters "match", "substitute", "insert" , "delete". For simplicity of illustration, substitutions are always $c \to (c \bmod 20) + 1$. While we experiment with several alignment models, we only report typical results here where matches occur with probability 0.2, substitutions with 0.4, insertion with 0.2, deletion with 0.2. The homolog is created by applying the alignment string to a randomly selected substring of the native. The shortening of the sequences through insertions and deletions is padded by additional random characters.

Figure 2 shows training and test error rates for two models depending on the number of training examples averaged over 10 trials. The first model has only 3 parameters ("match", "substitute", "insert/delete") and uses a uniform substitution matrix. The second model also learns the 400 parameters of the substitution matrix, resulting in a total of 403 parameters. We chose $C = 0.01$ and $\epsilon = 0.1$. The left-hand graph of Figure 2 shows that for the 403-parameter model, the generalization error is high for small numbers of training examples, but quickly drops as the number of examples increases. The 3-parameter model cannot fit the data as well. Its training error starts out much higher and training and test error essentially converge after only a few examples. The right-hand graph of Figure 2 shows the learned matrix of substitution costs for the 403-parameter model. As desired, the elements of the matrix are close to zero except for the off-diagonal. This captures the substitution model $c \to (c \bmod 20) + 1$.

Figure 3 analyzes the efficiency of the algorithm via the number of constraints that

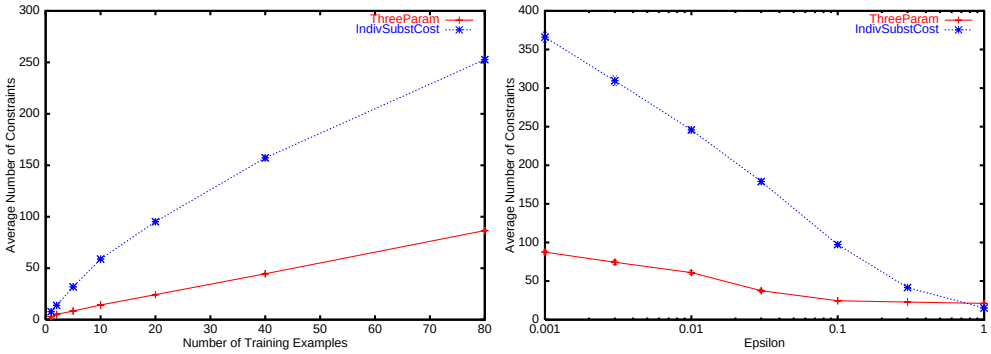

Figure 3: Number of constraints added to K depending on the number of training examples (left) and the value of ϵ (right). If not stated otherwise, $\epsilon = 0.1$, $C = 0.01$, and $n = 20$.

are added to K before convergence. The left-hand graph shows the scaling with the number of training examples. As predicted by Theorem 2, the number of constraints grows (sub-)linearly with the number of examples. Furthermore, the actual number of constraints is small enough so that it can easily be handled by standard quadratic optimization software. The right-hand graph shows how the number of constraints in the final K changes with $\log(\epsilon)$. The observed scaling appears to be better than suggested by the upper bound in Theorem 2. A good value for ϵ is 0.1. We observed that larger values lead to worse prediction accuracy, while smaller values decrease efficiency while not providing further benefit.

7 Conclusions

The paper presented a discriminative learning approach to inferring the cost parameters of a linear sequence alignment model from training data. We proposed an algorithm for solving the resulting training problem and showed its efficiency both theoretically and empirically. Experiments show that the algorithm can effectively learn the alignment parameters on a synthetic task.

We are currently applying the algorithm to learning alignment models for protein homology detection. An open question is whether it is possible to remove the assumption that the alignment between native and homolog is known while maintaining the tractability of the problem.

Acknowledgments

Many thanks to Ron Elber and Tamara Galor for their insightful discussions and their contribution to the research that lead to this paper.

References

[1] R. Barzilay and L. Lee. Bootstrapping lexical choice via multiple-sequence alignment. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2002.

[2] Corinna Cortes and Vladimir N. Vapnik. Support–vector networks. *Machine Learning Journal*, 20:273–297, 1995.

[3] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis*. Cambridge University Press, 1998.

[4] P. E. Gill, W. Murray, and M. H. Wright. *Practical Optimization*. Academic Press, 1981.

[5] D. Gusfield and P. Stelling. Parametric and inverse-parametric sequence alignment with xparal. *Methods in Enzymology*, 266:481–494, 1996.

[6] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. MIT Press, Cambridge, MA, 2000.

[7] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, 2002.

[8] J. Meller and R. Elber. Linear programming optimization and a double statistical filter for protein threading protocols. *Proteins Structure, Function, and Genetics*, 45:241–261, 2001.

[9] S. E Ristad and P. N. Yianilos. Learning string edit distance. *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol. 20(5):522–532, 1998.

[10] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[11] Fangting Sun, D. Fernandez-Baca, and Wei Yu. Inverse parametric sequence alignment. In *International Computing and Combinatorics Conference (COCOON)*, 2002.

[12] V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.