# Learning to Align Sequences: A Maximum-Margin Approach

Thorsten Joachims Tamara Galor
Ron Elber
Department of Computer Science
Cornell University
Ithaca, NY 14853
{tj,galor,ron}@cs.cornell.edu

June 24, 2005

**Abstract**

We propose a discriminative method for learning the parameters of linear sequence alignment models from training examples. Compared to conventional generative approaches, the discriminative method is straightforward to use when operations (e.g. substitutions, deletions, insertions) and sequence elements are described by vectors of attributes. This admits learning flexible and more complex alignment models. 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 [11][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 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

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $s_1$: | 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.

reflects the desired notion of similarity. Instead of assuming a generative model of sequence alignment (e.g. [11]), we take a discriminative approach to training following the general algorithm described in [14]. A key advantage of discriminative training is that operations can easily be described by features without having to model their dependencies like in generative training. In particular, we aim to find the set of parameter values that corresponds to the best similarity measure a given alignment model 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

$$D_{\vec{w}}(s_1, s_2, a) = \vec{w}^T \Psi(s_1, s_2, a) \tag{1}$$

where $\Psi(s_1, s_2, a)$ is a feature vector describing the alignment $a$ applied to $s_1$ and $s_2$. $\vec{w}$ is a given cost vector. Instead of assuming a finite alphabet $\Sigma$ and a finite set of operations, we only require that the reward/cost of each operation $a_o$ on any two characters $c_1, c_2 \in \Sigma$ can be expressed as a linear function over attributes $\phi(c_1, c_2, a_o)$

$$score(c_1, c_2, a_o) = \vec{w}^T \phi(c_1, c_2, a_o). \tag{2}$$

$\phi(c_1, c_2, a_o)$ can be thought of as a vector of attributes describing the match of $c_1$ and $c_2$ under operation $a_o$. Note that $c_1$ and $c_2$ can take dummy values for insertions, deletions, etc. This representation allows different scores depending on various properties of the characters or the operation. The feature vector for a complete alignment $\Psi(s_1, s_2, a)$ is the sum of the individual feature vectors

$$\Psi(s_1, s_2, a) = \sum_{i=1}^{|a|} \phi(c_1(a_i), c_2(a_i), a_i) \tag{3}$$

where $c_1(a_i)$ and $c_2(a_i)$ indicate the characters that $a_i$ is applied to. Only those operation sequences are valid that transform $s_1$ into $s_2$. Note that a special case of this model is the conventional parameterization using a substitution matrix and fixed scores for deletions and insertions. Finding the optimal alignment corresponds to the following optimization problem

$$\begin{aligned} D_{\vec{w}}(s_1, s_2) &= max_a \left[D_{\vec{w}}(s_1, s_2, a)\right] && (4)\\ &= max_a \left[\vec{w}^T \Psi(s_1, s_2, a)\right] && (5)\\ &= max_a \left[\sum_{i=1}^{|a|} score(c_1(a_i), c_2(a_i), a_i)\right]. && (6) \end{aligned}$$

This type of problem is typically solved via dynamic programming. In the following we consider local alignment via the Smith/Waterman algorithm [12]. However, the results can be extended to any alignment algorithm that optimizes a linear scoring function and that solves (6) globally optimally. This also holds for other structures besides sequences [14].

# 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, 13], they are limited to special cases and 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, instantiating the general algorithm we described in [14]. An interesting related approach is outlined in [9, 10], but it is not clear in how far it leads to practical algorithms.

We assume the following two scenarios, for which the notation is inspired by protein alignment.

## 3.1 Alignment Prediction

In the first scenario, the goal is to predict the optimal sequence of alignment operations $a$ for a given pair of sequences $s^N$ and $s^H$, which we call native and homolog sequence. We assume that examples are generated i.i.d. according to a distribution $P(s^N, s^H, a)$.

We approach this prediction problem using the following linear prediction rule which is parameterized by $\vec{w}$.

$$\hat{a} \quad = \quad argmax_a \left[ D_{\vec{w}}(s^N, s^H, a) \right] \tag{7}$$

This rule predicts the alignment sequence $\hat{a}$ which scores highest according to the linear model. By changing the cost of alignment operations via $\vec{w}$, the behavior of the prediction rule can be modified. The error of a prediction $\hat{a}$ compared to the true alignment $a$ is measured using a loss function $L(a, \hat{a})$. The goal of learning is to find a $\vec{w}$ that minimizes the expected loss (i.e. risk).

$$R_P^L(\vec{w}) = \int L\left(a, argmax_a \left[ D_{\vec{w}}(s^N, s^H, a) \right]\right) dP(s^N, s^H, a) \tag{8}$$

One reasonable loss function $L(.,.)$ to use is the number of alignment operations that are different in $a$ and $\hat{a}$. For simplicity, however, we will only consider the 0/1-loss $L_\Delta(.,.)$ in the following. It return the value 0 if both arguments are equal, and value 1 otherwise.

### 3.2 Homology Prediction

In the second scenario the goal is to predict whether two proteins are homologous. 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 $D_{\vec{w}}(.,.)$ 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.

$$D_{\vec{w}}(s^N, s^H) > D_{\vec{w}}(s^N, s^{D_j}). \tag{9}$$

The goal of learning 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 $D_{\vec{w}}(s^N, s^H)$.

$$Err_P^L(\vec{w}) = \int L_\Delta \left( s^H, \underset{s \in S^D \cup \{s^H\}}{\arg\max} D_{\vec{w}}(s^N, s) \right) dP(s^N, s^H, S^D) \tag{10}$$

Again, we assume a 0/1-loss $L_\Delta(.,.)$.

## 4 A Maximum-Margin Approach to Learning the Cost Parameters

In both scenarios, the data generating distributions $P(s^D, s^H, a)$ and $P(s^D, s^H, S^D)$ are unknown. However, we have a training sample $S$ drawn i.i.d from $P(.)$. This training sample will be used to learn the parameters $\vec{w}$. We will first consider the case of Alignment Prediction, and then extend the algorithm to the problem of Homology Prediction.

## 4.1 Alignment Predictions

Given is a training sample $S = ((s_1^D, s_1^H, a_1), ..., (s_n^D, s_n^H, a_n))$ of $n$ sequence pairs with their desired alignment. In the following, we will design a discriminative training algorithm that finds a parameter vector $\vec{w}$ for rules of type (7) by minimizing the loss on the training data $S$.

$$R_S^{L_\Delta}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n} L_\Delta\left(a_i, argmax_a\left[D_{\vec{w}}(s_i^N, s_i^H, a)\right]\right) \tag{11}$$

First, consider the case where there exists a $\vec{w}$ so that the training loss $R_S^{L_\Delta}(\vec{w})$ is zero. Since we assume a scoring function that is linear in the parameters

$$D_{\vec{w}}(s_1, s_2, a) = \vec{w}^T \Psi(s_1, s_2, a), \tag{12}$$

the condition of zero training error can be written as a set of linear inequality constraints. For each native/homolog pair $s_i^N$ / $s_i^H$, we need to introduce one linear constraint for each possible alignment $a$ of $s_i^N$ into $s_i^H$.

$$\begin{aligned} \forall a \neq a_1: \quad & D_{\vec{w}}(s_1^N, s_1^H, a) < D_{\vec{w}}(s_1^N, s_1^H, a_1) \\ ... & \\ \forall a \neq a_n: \quad & D_{\vec{w}}(s_n^N, s_n^H, a) < D_{\vec{w}}(s_n^N, s_n^H, a_n) \end{aligned} \tag{13}$$

Any parameter vector $\vec{w}^*$ that fulfills this set of constraints has a training loss $R_S^{L_\Delta}(\vec{w}^*)$ of zero. 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 (13) the number of constraints is exponential, since the number of alignments $a$ between $s_i^N$ and $s_i^H$ can be exponential in the length of $s_i^N$ and $s_i^H$. Unlike the restricted case in [8], standard optimization algorithms cannot handle this size of problem. To overcome this limitation, in Section 5 we will propose an algorithm that exploits the special structure of Equation (13) so that it needs to examine only a subset that is polynomial in the length of $s_i^N$ and $s_i^H$.

If the set of inequalities in Equation (13) is feasible, there will typically be more than one solution $\vec{w}^*$. To specify a unique solution, we select the $\vec{w}^*$ for which each score $D_{\vec{w}}(s_i^N, s_i^H, a_i)$ is uniformly most different from $\max_{a \neq a_i} D_{\vec{w}}(s_i^N, s_i^H, a)$ for all $i$. This corresponds to the maximum-margin principle employed in Support Vector Machines (SVMs) [15]. Denoting the margin by $\delta$ and restricting the $L_2$ norm of $\vec{w}$ to make the problem well-posed, this leads to the following optimization problem.

$$max_{\vec{w}} \quad \delta \tag{14}$$

$$\begin{aligned} \forall a \neq a_1: \quad & D_{\vec{w}}(s_1^N, s_1^H, a) \leq D_{\vec{w}}(s_1^N, s_1^H, a_1) - \delta \\ ... & \\ \forall a \neq a_n: \quad & D_{\vec{w}}(s_n^N, s_n^H, a) \leq D_{\vec{w}}(s_n^N, s_n^H, a_n) - \delta \end{aligned} \tag{15}$$

$$||\vec{w}|| = 1 \tag{16}$$

Due to the linearity of the similarity function (12), the length of $\vec{w}$ is a free variable and we can fix it to $1/\delta$. Substituting for $\delta$ and rearranging leads to the equivalent optimization problem

$$min_{\vec{w}} \qquad \frac{1}{2}\vec{w}^T\vec{w} \tag{17}$$

$$\forall a \neq a_1 : \qquad \left(\Psi(s_1^N, s_1^H, a_1) - \Psi(s_1^N, s_1^H, a)\right)\vec{w} \geq 1 \tag{18}$$

$$... \tag{19}$$

$$\forall a \neq a_n : \qquad \left(\Psi(s_n^N, s_n^H, a_n) - \Psi(s_n^N, s_n^H, a)\right)\vec{w} \geq 1 \tag{20}$$

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 $\xi_i$ [2]. Corresponding to the error measure $R_P^{L_\Delta}(\vec{w})$ 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.

$$min_{\vec{w},\vec{\xi}} \qquad \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^{n}\xi_i^2 \tag{21}$$

$$\forall a \neq a_1 : \qquad \left(\Psi(s_1^N, s_1^H, a_1) - \Psi(s_1^N, s_1^H, a)\right)\vec{w} \geq 1 - \xi_1 \tag{22}$$

$$... \tag{23}$$

$$\forall a \neq a_n : \qquad \left(\Psi(s_n^N, s_n^H, a_n) - \Psi(s_n^N, s_n^H, a)\right)\vec{w} \geq 1 - \xi_n \tag{24}$$

Analogous to classification and regression SVMs [15], this formulation minimizes a regularized upper bound on the training loss $R_S^{L_\Delta}(\vec{w})$.

**Proposition 1** *For any feasible point $(\vec{w}, \vec{\xi})$ of (21)-(24), $\frac{1}{n}\sum_{i=1}^{n}\xi_i^2$ is an upper bound on the training loss $R_S^{L_\Delta}(\vec{w})$ for the 0/1-loss $L_\Delta(.,.)$.*

The proof is given in [14]. The quadratic program and the proposition can be extended to any non-negative loss function.

## 4.2 Homology Prediction

For the problem of Homology Prediction, we can derive a similar training problem. Here, the training sample $S$ 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[1]. The goal is to find an optimal $\vec{w}$ so that the error rate $Err_P^{L_\Delta}(\vec{w})$ is low. Again, finding a $\vec{w}$ such that the error

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

on the training set

$$Err_P^{L_\Delta}(\vec{w}) = \frac{1}{n}\sum_{i=1}^{n} L_\Delta\left(s_i^H, \arg\max_{s \in S_i^D \cup \{s_i^H\}} D_{\vec{w}}(s_i^N, s)\right) \tag{25}$$

is zero 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 $s_i^{D_j}$.

$$\begin{aligned} \forall s_1^{D_j} \in S_1^D \forall a: \quad & D_{\vec{w}}(s_1^N, s_1^{D_j}, a) < D_{\vec{w}}(s_1^N, s_1^H, a_1^{NH}) \\ & \ldots \\ \forall s_n^{D_j} \in S_n^D \forall a: \quad & D_{\vec{w}}(s_n^N, s_n^{D_j}, a) < D_{\vec{w}}(s_n^N, s_n^H, a_n^{NH}) \end{aligned} \tag{26}$$

Similar to the case of Alignment Prediction, one can add a margin criterion and slacks and arrives at the following convex quadratic program.

$$min_{\vec{w},\vec{\xi}} \quad \frac{1}{2}\vec{w}^T\vec{w} + C\sum_{i=1}^{n}\xi_i^2 \tag{27}$$

$$\forall s_1^{D_j} \in S_1^D \forall a: \quad \left(\Psi(s_1^N, s_1^H, a_1^{NH}) - \Psi(s_1^N, s_1^{D_j}, a)\right)\vec{w} \geq 1 - \xi_1 \tag{28}$$

$$\ldots \tag{29}$$

$$\forall s_n^{D_j} \in S_n^D \forall a: \quad \left(\Psi(s_n^N, s_n^H, a_n^{NH}) - \Psi(s_n^N, s_n^{D_j}, a)\right)\vec{w} \geq 1 - \xi_n \tag{30}$$

Again, $\frac{1}{n}\sum_{i=1}^{n}\xi_i^2$ is an upper bound on the training error $Err_S^{L_\Delta}(\vec{w})$.

# 5 Training Algorithm

Due to the exponential number of constraints in both the optimization problem for Alignment Prediction and Homology Prediction, naive use of off-the-shelf tools for their solution is computationally intractable for problems of interesting size. However, by exploiting the special structure of the problem, we propose the algorithms shown in Figures 2 and 3 that find the solutions of (21)-(24) and (27)-(30) after examining only a small number of constraints. The algorithms proceeds by greedily adding constraints from (21)-(24) or (28)-(30) to a working set $K$. The algorithms stop, when all constraints in (21)-(24) or (28)-(30) are fulfilled up to a precision of $\epsilon$.

The following two theorems show that the algorithms return a solutions of (21)-(24) and (27)-(30) that are accurate with a precision of some predefined $\epsilon$, and that they stop after a polynomially bounded number of iterations through the repeat-loop.

**Theorem 1** (CORRECTNESS)
*The algorithms return an approximation that has an objective value not higher than the solution of (21)-(24) and (27)-(30), and that fulfills all constraints up to a precision of $\epsilon$. For $\epsilon = 0$, the algorithm returns the exact solution $(\vec{w}^*, \vec{\xi}^*)$.*

Input: native sequences $s_1^N, ..., s_n^N$, homolog sequences $s_1^H, ..., s_n^H$, alignments $a_1, ..., a_n$, tolerated approximation error $\epsilon \geq 0$.

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

repeat

- $K_{org} = K$
- for $i$ from 1 to $n$
  - find $\hat{a} = argmax_a \left[\vec{w}^T \Psi(s_i^N, s_i^H, a)\right]$ via dynamic programming
  - if $\vec{w}^T(\Psi(s_i^N, s_i^H, a_i) - \Psi(s_i^N, s_i^H, \hat{a})) < 1 - \xi_i - \epsilon$
    - $K = K \cup \left\{\vec{w}^T(\Psi(s_i^N, s_i^H, a_i) - \Psi(s_i^N, s_i^H, \hat{a})) \geq 1 - \xi_i\right\}$
    - 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: $\vec{w}$

Figure 2: Sparse Approximation Algorithm for the Alignment Prediction task.

**Proof** *Let $\vec{w}^*$ and $\xi_i^*$ be the solution of (21)-(24) or (27)-(30) respectively. Since the algorithm solves the QP on a subset of the constraints in each iteration, it returns 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 restricting the feasible region cannot lead to a lower minimum.*

*It is left to show that the algorithm does not terminate before all constraints (21)-(24) or (28)-(30) are fulfilled up to precision $\epsilon$. In the final iteration, the algorithms find the most violated constraint. For Alignment Prediction, this is the constraint $\vec{w}^T(\Psi(s_i^N, s_i^H, a_i) - \Psi(s_i^N, s_i^H, \hat{a})) < 1 - \xi_i$ corresponding to the highest scoring alignment $\hat{a}$. For Homology Prediction, it is the constraint $\vec{w}^T(\Psi(s_i^N, s_i^H, a_i^{NH}) - \Psi(s_i^N, s_i^{D_j}, \hat{a})) < 1 - \xi_i$ corresponding to the highest scoring alignment $\hat{a}$ for each decoy. Three cases can occur: First, the constraint can be violated by more than $\epsilon$ 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 $\epsilon$.* ∎

It is left to show that the algorithms terminates after a number of iterations that is smaller than the set of constraints. The following theorem shows that the algorithm stops after a polynomial number of iterations.

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 $\hat{a} = argmax_{a \neq a_i} \left[ \vec{w}^T \Psi(s_i^N, s_i^{D_j}, a) \right]$ via dynamic programming
    - if $\vec{w}^T(\Psi(s_i^N, s_i^H, a_i^{NH}) - \Psi(s_i^N, s_i^{D_j}, \hat{a})) < 1 - \xi_i - \epsilon$
      - $K = K \cup \left\{ \vec{w}^T(\Psi(s_i^N, s_i^H, a_i^{NH}) - \Psi(s_i^N, s_i^{D_j}, \hat{a})) \geq 1 - \xi_i \right\}$
      - 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: $\vec{w}$

Figure 3: Sparse Approximation Algorithm for the Homology Prediction task.

**Theorem 2** (TERMINATION)
*The algorithms stops after adding at most*

$$\frac{2VR^2}{\epsilon^2} \tag{31}$$

*constraints to the set $K$. $V$ is the minimum of (21)-(24) or (27)-(30) respectively. $R^2$ is a constant bounded by the maximum of $(\Psi(s_i^N, s_i^H, a_i) - \Psi(s_i^N, s_i^H, a))^2 + \frac{1}{2C}$ or $(\Psi(s_i^N, s_i^H, a_i^{NH}) - \Psi(s_i^N, s_i^{D_j}, a))^2 + \frac{1}{2C}$ respectively.*

**Proof** *In the following, we focuses on the Homology Prediction task. The proof for the Alignment Prediction task is analogous. 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 $\vec{x} = (\Psi(s_i^N, s_i^H, a_i^{NH}) - \Psi(s_i^N, s_i^{D_j}, \hat{a}); 0; ...; 0; 1/\sqrt{2C}; 0; ...; 0)$. 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*

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

*Solving the remaining scalar optimization problem over* $\alpha_{k+1}$ *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* $\epsilon$, *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* $\epsilon$ *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* $\epsilon$ *and the algorithm stops.* ∎

Since $V$ can be upper bounded as $V \leq C * n$ using the feasible point $\vec{w} = 0$ and $\vec{\xi} = 1$ in (21)-(24) or (27)-(30), the theorem directly leads to the conclusion that the maximum number of constraints in $K$ scales linearly with the number of training examples $n$. 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 $\epsilon$, experience with Support Vector Machines for classification showed that relatively large values of $\epsilon$ 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. While this simple model does not exploit the flexibility of the parameterized linear model $D_{\vec{w}}(s_1, s_2, a) = \vec{w}^T \Psi(s_1, s_2, a)$, it does serve as a feasibility check of the learning algorithm. 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 $c \rightarrow (c \bmod 20) + 1$. While we experiment with several alignment models, we only report typical results here where matches occur with probability $0.2$, substitutions with

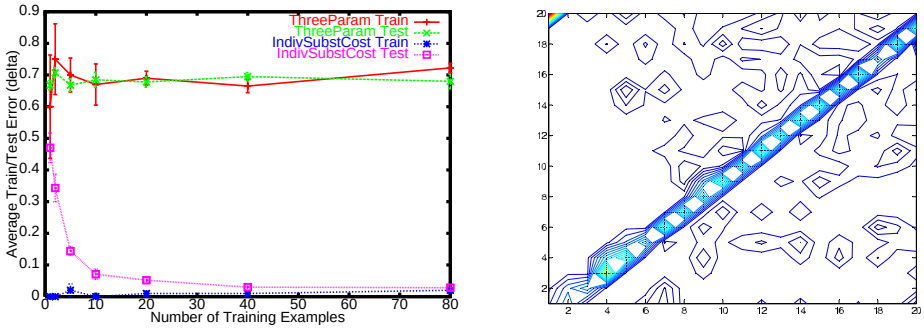


Figure 4: Left: Train and test error rates for the 3 and the 403 parameter model depending on the number of training examples. Right: Typical learned substitution matrix after 40 training examples for the 403-parameter model.

$0.4$, 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.

In the following experiments, we focus on the problem of Homology Prediction. Figure 4 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. This makes the feature vectors $\phi(c_1, c_2, a_i)$ three-dimensional with a 1 indicating the appropriate operation. The second model also learns the 400 parameters of the substitution matrix, resulting in a total of 403 parameters. Here, $\phi(c_1, c_2, a_i)$ indicates the element of the substitution matrix in addition to the operation type. We chose $C = 0.01$ and $\epsilon = 0.1$. The left-hand graph of Figure 4 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 4 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 $c \rightarrow (c \bmod 20) + 1$.

Figure 5 analyzes the efficiency of the algorithm via the number of constraints that 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 encountered during any iteration of the algorithm is small enough to 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 $\epsilon$ is $0.1$. We observed that larger values lead to worse prediction accuracy, while smaller values decrease efficiency while not providing further benefit.

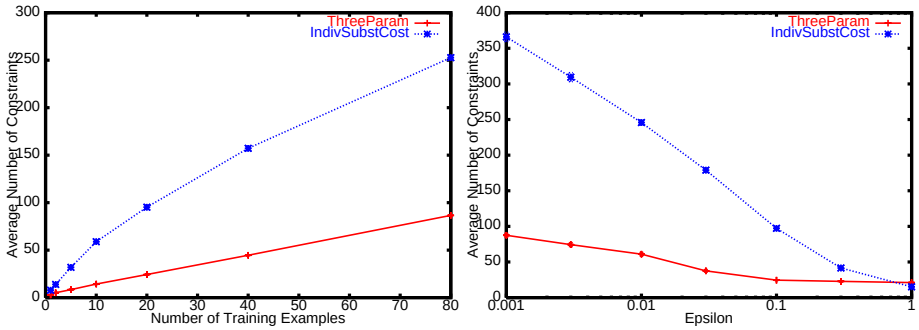


Figure 5: Number of constraints added to $K$ depending on the number of training examples (left) and the value of $\epsilon$ (right). If not stated otherwise, $\epsilon = 0.1$, $C = 0.01$, and $n = 20$.

# 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 that it is computationally efficient. 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 and protein alignment prediction.

# 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] L. Pachter and B. Sturmfelds. Parametric inference for biological sequence analysis. In *Proceedings of the National Academy of Sciences*, volume 101, pages 16138–16143, 2004.

[10] L. Pachter and B. Sturmfelds. Tropical geometry of statistical models. In *Proceedings of the National Academy of Sciences*, volume 101, pages 16132–16137, 2004.

[11] 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.

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

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

[14] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *International Conference on Machine Learning (ICML)*, 2004.

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