Inverse Sequence Alignment from Partial Examples

Eagu Kim and John Kececioglu

Department of Computer Science
The University of Arizona, Tucson AZ 85721, USA
{egkim,kece}@cs.arizona.edu

Abstract. When aligning biological sequences, the choice of parameter values for the alignment scoring function is critical. Small changes in gap penalties, for example, can yield radically different alignments. A rigorous way to compute parameter values that are appropriate for biological sequences is *inverse parametric sequence alignment*. Given a collection of examples of biologically correct alignments, this is the problem of finding parameter values that make the example alignments score close to optimal. We extend prior work on inverse alignment to *partial examples* and to an improved model based on minimizing the *average error* of the examples. Experiments on benchmark biological alignments show we can find parameters that generalize across protein families and that boost the recovery rate for multiple sequence alignment by up to 25%.

1 Introduction

A fundamental issue in molecular sequence analysis is deciding what parameter values to use when aligning biological sequences. For example, the standard scoring function for protein sequence alignment requires determining values for 210 substitution scores and two gap penalties. An interesting approach to determining these values is *inverse parameteric sequence alignment* [6,10], where parameters are set using examples of correct alignments. Informally, inverse alignment tries to find parameter values that make the examples be optimal-scoring alignments of their strings. In practice, parameter values rarely exist that make a collection of biological examples optimal, so the problem becomes finding values that make the examples score *close* to optimal. An important issue is determining what measure of *error* between the scores of the examples and the scores of optimal alignments should be optimized.

Recently, Kececioglu and Kim [8] discovered a new method for inverse alignment based on linear programming that for the first time could quickly find values for all 212 parameters in the standard protein sequence alignment model from hundreds of examples of complete alignments. Their approach minimized the maximum relative error across the examples. In this paper we extend this work in three directions: (1) to examples consisting of *partial alignments*, which are the type of examples currently available in the standard suites of benchmark protein alignments, and which consist of incomplete sequence alignments; (2) to

R. Giancarlo and S. Hannenhalli (Eds.): WABI 2007, LNBI 4645, pp. 359–370, 2007.

an improved error model involving minimization of *average error* across the examples; and (3) to experimentally study the performance of parameters learned by inverse alignment in terms of their *recovery rate* on benchmark alignments.

Related Work. Inverse parametric alignment was introduced in the seminal paper of Gusfield and Stelling [6]. They considered the problem for two parameters and one example, and gave an indirect approach to inverse alignment that attempted to avoid computing a parametric decomposition of the parameter space. Sun, Fernández-Baca and Yu [10] gave the first direct algorithm for inverse alignment for the case of three parameters and one example; given two strings of length n, their algorithm finds parameters that make the example optimal in $O(n^2 \log n)$ time. Kececioglu and Kim [8] gave the first polynomial-time algorithm for arbitrarily-many parameters and examples; their algorithm finds parameters that make the examples score as close to optimal as possible in terms of relative error. As they demonstrated, it is also fast in practice.

The authors recently learned that Eppstein [5] independently discovered a general approach to inverse parametric optimization that is similar to [8]. Eppstein applied it in the context of minimum spanning trees, and considered finding parameters that make an example tree the unique optimal solution; in the context of biological sequence alignment, however, this rarely has a solution.

Alternate approaches for determining alignment parameters have been recently proposed based on machine learning. Do, Gross and Batzoglou [4] use discriminative training on conditional random fields to find parameter values for a hidden Markov model of sequence alignment. Their approach requires solving a convex numerical optimization problem (which becomes nonconvex in the presence of partial alignments), and does not provide a polynomial-time guarantee on running time. Yu, Joachims, Elber and Pillardy [12] describe a support-vector-machine approach for learning parameters to align a protein sequence to a protein structure. Their approach involves solving a quadratic numerical optimization problem with linear constraints, and for the first time incorporates a measure of alignment recovery directly into the problem formulation.

In contrast to these machine learning approaches, our method for inverse alignment uses linear programming (which can be solved quickly even for very large instances) and for the first time we rigorously address partial examples.

Overview. The next section presents several variations of inverse alignment, with relative or absolute error, and with complete or partial examples. Section 3 reduces these variations to linear programmming, and develops an iterative approach to partial examples. Finally Section 4 presents results from experiments on recovering benchmark protein alignments when using learned parameters for both pairwise and multiple sequence alignment.

2 Inverse Alignment and Its Variations

The conventional sequence alignment problem is, given a pair of strings and a scoring function f on alignments, find an alignment $\mathcal{A}$ of the strings that has optimal score under f. The *inverse* alignment problem turns this around: given

an alignment $\mathcal{A}$ of a pair of strings, find parameter values for scoring function f that makes $\mathcal{A}$ be an optimal alignment of its strings. To learn parameter values that are useful in practice this basic form of inverse alignment, which was originally studied in [6,10], must be generalized in several directions.

When function f has many parameters, many input alignments $\mathcal{A}$ are needed to determine reliable values for the parameters. Accordingly we consider inverse alignment where the input is a large *collection* of example alignments. In practice there are usually no parameter values that make the example alignments have optimal score. Consequently we consider finding parameters that make the examples score near-optimal, and we examine two criteria for measuring the *error* between the example scores and the optimal alignment scores: minimizing relative error or absolute error. Finally, the type of benchmark alignments that are available in practice for learning parameters actually consist of regions where the alignment is specified, interspersed with stretches where no alignment is specified. We call such an input alignment a *partial* example, since it is only a partial alignment of its strings. When an example specifies a complete alignment of its strings, we call it a complete example.

Our approach to inverse alignment from partial examples builds upon a solution to the problem with complete examples, which we discuss first.

Complete Examples. Inverse alignment from complete examples with arbitrarily many parameters was first considered by Kececioglu and Kim [8]. They examined the relative-error criterion, which we review below.

Let f be the alignment *scoring function*, which gives score $f(\mathcal{A})$ to alignment $\mathcal{A}$. Typically f is a function of several *parameters* $p_1, p_2, \ldots, p_t$, which assign scores or penalites to various alignment features such as substitutions and gaps. (For example, the standard scoring model for aligning protein sequences has 210 substitution scores for all unordered pairs of amino acids, plus two gap penalties for opening and extending a gap, for a total of $t = 212$ parameters.) We view the entire set of parameters as a vector $p = (p_1, \ldots, p_t)$. When we want to emphasize the dependence of f on its parameters p, we write f_p.

The input consists of many example alignments $\mathcal{A}_i$, where each example aligns a corresponding set of strings $\mathcal{S}_i$. (Typically the examples $\mathcal{A}_i$ are induced pairwise alignments that come from a structural multiple alignment; in this case, each $\mathcal{S}_i$ contains two strings.) For scoring function f and parameters p, we write $f_p^*(\mathcal{S}_i)$ for the score of an *optimal* alignment of strings $\mathcal{S}_i$ under f_p. The following definitions assume that an optimal alignment *maximizes* scoring function f. (The original formulation [8] was in terms of minimizing f.)

Definition 1 (Complete examples under relative error). *Inverse Alignment* from complete examples under the relative error criterion is the following problem. Let the alignment scoring function be f_p with parameter vector $p = (p_1, \ldots, p_t)$ drawn from domain $\mathcal{D}$. The *input* is a collection of complete alignments $\mathcal{A}_1, \ldots, \mathcal{A}_k$ that respectively align the sets of strings $\mathcal{S}_1, \ldots, \mathcal{S}_k$. The *output* is parameter vector $x^* := \operatorname{argmin}_{x \in \mathcal{D}} E_{\text{rel}}(x)$, where

$$E_{\text{rel}}(x) \;:=\; \max_{1 \le i \le k} \frac{f_x^*(\mathcal{S}_i) \;-\; f_x(\mathcal{A}_i)}{f_x^*(\mathcal{S}_i)}.$$

In other words, the output vector x^* minimizes the maximum relative error of the alignment scores of the examples. □

While $E_{\text{rel}}(x)$ is not well-defined for $f_x^*(\mathcal{S}_i) \leq 0$, we will avoid this in Section 3.

When scoring function f_p is linear in its parameters p, inverse alignment under relative error can be solved in polynomial time [8] as long as an optimal alignment can be computed in polynomial time for any *fixed* parameter choice. We review the solution in Section 3, which uses a reduction to linear programming. The above formulation considers the *maximum* error of the examples, because as we will see later, minimizing the average relative error would lead to an optimization problem with nonlinear constraints.

We also consider here a new model: minimizing absolute error. This has the advantage that we can minimize the *average* error of the examples and still have a formulation that is efficiently solvable by linear programming.

Definition 2 (Complete examples under absolute error). *Inverse Alignment* from complete examples under the absolute error criterion is the following problem. The input is a collection of complete alignments $\mathcal{A}_i$ of strings $\mathcal{S}_i$ for $1 \leq i \leq k$. The output is parameter vector $x^* := \operatorname{argmin}_{x \in \mathcal{D}} E_{\text{abs}}(x)$ where

$$E_{\text{abs}}(x) \;:=\; \frac{1}{k} \sum_{1 \leq i \leq k} \Big(f_x^*(\mathcal{S}_i) \;-\; f_x(\mathcal{A}_i) \Big).$$

Output vector x^* minimizes the average absolute error of the example scores. □

A key issue in the above formulations of inverse alignment is that the problem is *degenerate*. In both formulations, the trivial parameter choice $x = (0, 0, \ldots, 0)$ is an optimal solution (as it makes every alignment, including the example, be an optimal alignment). This trivial solution must be ruled out in an application-specific manner that depends on the particular form of the alignment problem being considered. Section 3 presents a new approach for avoiding degeneracy that applies to both global and local alignment of protein sequences.

Partial Examples. For inverse alignment of protein sequences, the best example alignments that are available come from multiple alignments of protein families that are determined by aligning the three-dimensional structures of family members. Several suites of such benchmark alignments are now available [1] and are widely used for evaluating the accuracy of software for multiple alignment of protein sequences. Most all these benchmark alignments, however, are partial alignments. The benchmark alignment has regions that are reliable and where the alignment is specified, but between these regions the alignment of the strings is effectively left unspecified. These reliable regions are usually the *core blocks* of the multiple alignment, which are gapless sections of the alignment where structure is conserved across the family.

For our purposes a *partial example* is an alignment $\mathcal{A}$ of strings $\mathcal{S}$ where each column of $\mathcal{A}$ is labeled as being either reliable or unreliable. A *complete example* is a partial example whose columns are all labeled reliable.

When learning parameters by inverse alignment from partial examples, we treat the unreliable columns as missing information: such columns do not specify the alignment of the strings. Given a partial example $\mathcal{A}$ for strings $\mathcal{S}$, a *completion* $\overline{\mathcal{A}}$ of $\mathcal{A}$ is a complete example for $\mathcal{S}$ that agrees with the reliable columns of $\mathcal{A}$. In other words, a completion $\overline{\mathcal{A}}$ can change $\mathcal{A}$ on the substrings that are in unreliable columns, but must not alter $\mathcal{A}$ in reliable columns.

We define inverse alignment from partial examples as the problem of finding the optimal parameter choice over all possible completions of the examples.

Definition 3 (Partial examples). *Inverse Alignment* from partial examples is the following problem. The input is a collection of partial alignments $\mathcal{A}_i$ of strings $\mathcal{S}_i$ for $1 \leq i \leq k$. The output is parameter vector

$$x^* := \operatorname*{argmin}_{x \in \mathcal{D}} \min_{\overline{\mathcal{A}}_1, \ldots, \overline{\mathcal{A}}_k} E(x),$$

where error function E is either E_{abs} or E_{rel}. In other words, vector x^* minimizes the error of the example scores over all completions of the partial examples. □

In the next section we reduce inverse alignment from complete examples to linear programming, and approach the problem with partial examples by solving a series of problems on complete examples.

3 Solution by Linear Programming

When the alignment scoring function f_p is linear in its parameters p, inverse alignment from complete examples under relative error can be reduced to linear programming [8], and a similar reduction applies to absolute error. We define a linear scoring function as follows. Suppose f scores an alignment $\mathcal{A}$ by measuring $t+1$ features of $\mathcal{A}$ through functions $g_0, g_1, \ldots, g_t$ and combines these measures into one score through a weighted sum involving parameter vector $p = (p_1, \ldots, p_t)$ by $f_p(\mathcal{A}) := g_0(\mathcal{A}) + \sum_{1 \leq i \leq t} p_i\, g_i(\mathcal{A})$. Then we say f is *linear* in parameters $p_1, \ldots, p_t$.

For example, in the standard scoring model for alignment of protein sequences, for every unordered pair a, b of amino acids there is a substitution score σ_{ab}, plus gap penalty γ for opening a gap and penalty λ for extending a gap. This gives a scoring function with 212 parameters $\sigma_{ab}, \gamma, \lambda$ for the alphabet of 20 amino acids. The functions g_{ab} count the number of substitutions of each type a, b in $\mathcal{A}$, and functions g_γ and g_λ count the number of gaps and the total length of all gaps in $\mathcal{A}$.

Complete Examples. As described in [8], for inverse alignment from complete examples with relative error, we first consider the problem assuming a fixed upper bound ϵ on the relative error. For a given bound ϵ, we test whether there is a feasible solution x with relative error at most ϵ by solving a linear program. We then find the smallest ϵ^*, to a given accuracy, for which there is a feasible solution using binary search on ϵ. The feasible solution x^* found at bound ϵ^* is an optimal solution to inverse alignment under relative error.

We briefly summarize this linear programming approach for the standard model of protein sequence alignment. The parameters of the scoring function are the variables of the linear program. The domain $\mathcal{D}$ of the parameters is described by the inequalities $(-1, 0, 0) \leq (\sigma_{ab}, \gamma, \lambda) \leq (1, 1, 1)$. When the alignment problem is to maximize the score f of an alignment, substitution scores σ_{ab} are usually allowed to be both positive and negative, and parameter values can always be rescaled so the largest magnitude hits 1 without changing the alignment problem. We also add the inequalities $\sigma_{ab} \leq \sigma_{aa}$ for all $a \neq b$, since an identity should score better than any substitution involving that letter. To ensure relative error $E_{\text{rel}}(x)$ is well-defined, we constrain $f_x(\mathcal{A}_i) \geq 0$ for all examples.

For the *relative error* criterion, the remaining inequalities in the linear program enforce that the relative error of all examples is at most ϵ. For each example $\mathcal{A}_i$, and *every* alignment $\mathcal{B}_i$ of strings $\mathcal{S}_i$, the linear program has an inequality

$$f_x(\mathcal{A}_i) \;\geq\; (1 - \epsilon)\, f_x(\mathcal{B}_i).$$

Notice that for a fixed value of ϵ, this is a linear inequality in parameters x, since function f_x is linear in x. Example $\mathcal{A}_i$ satisfies all these inequalities iff the inequality with $\mathcal{B}_i = \mathcal{B}^*$ is satisfied, where $\mathcal{B}^*$ is an optimal-scoring alignment of $\mathcal{S}_i$ under parameters x. In other words, the inequalities are all satisfied iff the score of $\mathcal{A}_i$ has relative error at most ϵ under f_x. Finding the minimum ϵ for which this system of inequalities has a feasible solution x corresponds to minimizing the maximum relative error of the example scores.

This linear program has an exponential number of inequalities, since for an example $\mathcal{A}_i$ there are exponentially-many alignments $\mathcal{B}_i$ of $\mathcal{S}_i$ (in terms of the lengths of the strings). Nevertheless, this program can be solved in *polynomial time* using a far-reaching result from linear programming theory. This result, known as the equivalence of optimization and separation [2], states that one can solve a linear program in polynomial time iff one can solve the *separation problem* for the linear program in polynomial time. The separation problem is, given a possibly infeasible vector $\widetilde{x}$ of parameter values, to report an inequality from the linear program that is violated by $\widetilde{x}$, or to report that $\widetilde{x}$ satisfies the linear program if there is no violated inequality.

We can solve the separation problem in polynomial time for the above linear program by the following algorithm. Given a vector $\widetilde{x}$ of parameter values, for each example $\mathcal{A}_i$ we compute an optimal-scoring alignment $\mathcal{B}^*$ of $\mathcal{S}_i$ under $f_{\tilde{x}}$. If the above inequality is satisfied when $\mathcal{B}_i = \mathcal{B}^*$, the inequalties are satisfied for all $\mathcal{B}_i$, and if the above inequality is not satisfied for $\mathcal{B}^*$, this gives the requisite violated inequality. For a problem with k examples, solving the separation problem involves computing at most k optimal alignments.

In practice, this leads to the following *cutting plane* algorithm [2] for solving a linear program consisting of inequalities $\mathcal{L}$.

(1) Start with a small subset $\mathcal{P}$ of the inequalities in $\mathcal{L}$.
(2) Compute an optimal solution $\widetilde{x}$ to the linear program given by subset $\mathcal{P}$. If no such solution exists, halt and report that $\mathcal{L}$ is infeasible.

(3) Call the separation algorithm for $\mathcal{L}$ on $\widetilde{x}$. If the algorithm reports that $\widetilde{x}$ satisfies $\mathcal{L}$, output $\widetilde{x}$ and halt: $\widetilde{x}$ is an optimal solution for $\mathcal{L}$.
(4) Otherwise, add the violated inequality returned by the separation algorithm in Step (3) to $\mathcal{P}$, and loop back to Step (2).

While such cutting plane algorithms are not guaranteed to terminate in polynomial time, they can be fast in practice [8]. For inverse alignment, we start with subset $\mathcal{P}$ containing just the trivial inequalities that specify parameter domain $\mathcal{D}$.

For the *absolute error* criterion, we modify the linear program as follows. For each example $\mathcal{A}_i$ we have an additional error variable δ_i. The inequalities for each example $\mathcal{A}_i$ are replaced by

$$f_x(\mathcal{A}_i) \;\geq\; f_x(\mathcal{B}_i) \;-\; \delta_i.$$

Finally, the objective function for the linear program is to minimize $\sum_i \delta_i$. An optimal solution x^* to this linear program gives a parameter vector that minimizes the average absolute error of the example scores. Again the program has exponentially-many inequalities, but the same separation algorithm that computes an optimal alignment $\mathcal{B}^*$ solves the separation problem in polynomial time, so in principle the linear program can be solved in polynomial time. In practice we use a cutting plane algorithm as described above.

Partial Examples. Inverse alignment from partial examples involves optimizing over all possible completions of the examples. While for partial examples we do not know how to efficiently find an optimal solution, we present a practical iterative approach which as demonstrated in Section 4 finds a good solution.

Start with an initial completion $\overline{\mathcal{A}}_i^{(0)}$ for each partial example $\mathcal{A}_i$. These initial completions may be formed by computing alignments of the unreliable regions that are optimal with respect to a default parameter choice $x^{(0)}$. (In practice for $x^{(0)}$ we use a standard substitution matrix [7] with appropriate gap penalties.) Alternately, an initial completion may be trivially obtained by taking the alignment of the unreliable regions in the partial example as the completion.

We then iterate the following for $j = 0, 1, \ldots$. Compute an optimal parameter choice $x^{(j+1)}$ by solving the inverse alignment problem on the complete examples $\overline{\mathcal{A}}_i^{(j)}$. Given $x^{(j+1)}$, form a new completion $\overline{\mathcal{A}}_i^{(j+1)}$ of $\mathcal{A}_i$ by (1) computing alignments of the unreliable regions that are optimal with respect to parameters $x^{(j+1)}$, and (2) concatenating them to form a complete example. Such a completion optimally stitches together the reliable regions of the partial example, using the current estimate for parameter values. This *iterative scheme* repeatedly solves inverse alignment using improved complete examples. As the following result shows, each iteration yields a better parameter estimate.

Theorem 1 (Error convergence for partial examples). *For the iterative scheme for inverse alignment from partial examples, denote the error in score for iteration $j \geq 1$ by $e_j := E\big(x^{(j)}\big)$, where E is error criterion E_{abs} or E_{rel} measured on completions $\overline{\mathcal{A}}_i^{(j-1)}$. Then*

$$e_1 \;\geq\; e_2 \;\geq\; \cdots \;\geq\; e^*,$$

where e^* is the optimum error for inverse alignment from partial examples $\mathcal{A}_i$.

Proof sketch. Since $\overline{\mathcal{A}}_i^{(j)}$ is an optimal-scoring completion of $\mathcal{A}_i$ with respect to parameters $x^{(j)}$,

$$f_{x^{(j)}}\left(\overline{\mathcal{A}}_i^{(j)}\right) \;\geq\; f_{x^{(j)}}\left(\overline{\mathcal{A}}_i^{(j-1)}\right).$$

This implies that with respect to the new complete examples $\overline{\mathcal{A}}_i^{(j)}$, the old parameters $x^{(j)}$ are still feasible at error e_j. So for the new examples, error e_j is achievable. Since the optimum error e_{j+1} for the new examples cannot be worse, $e_{j+1} \leq e_j$. Furthermore e^* lower bounds the error for all completions. □

By the above result, the error of the iterative scheme converges, though it may converge to a value larger than the optimum error e^*. As shown in Section 4, choosing a good initial completion can reduce the error. In practice we iterate this scheme until the improvement in error becomes too small or a bound on the number of iterations is reached. Moreover as the error improves across iterations, recovery of the examples generally improves as well.

Eliminating Degeneracy. To eliminate the degenerate solution $x = (0, \ldots, 0)$ we use the following approach. When the alignment problem is to maximize scoring function f, substitution scores σ_{ab} are typically both positive and negative, where a positive score indicates letters a, b are similar, and a negative score indicates they are dissimilar. For the σ_{ab} to be appropriate for local alignment, the expected score of a substitution in an alignment of two random strings should be strictly negative. (Otherwise, extending a local alignment by concatenating columns tends to increase its score, so an optimal local alignment degenerates into a trivial global alignment that substitutes as much as possible.) Similarly for global alignment, this expected score should be negative so random substitutions are considered dissimilar.

Let threshold τ be the expected score of a random substitution for a default substitution scoring matrix. Values of τ for commonly-used `BLOSUM` [7] and `PAM` [3] substitution matrices at standard amino acid frequencies are shown below, where each matrix has been scaled so its scores lie in interval $[-1, 1]$.

	BLOSUM45	BLOSUM62	BLOSUM80	PAM250	PAM160	PAM120
τ	-0.056	-0.091	-0.136	-0.050	-0.056	-0.138

Note that as the percent identity value for the matrix increases (corresponding to increasing `BLOSUM` or decreasing `PAM` numbers), threshold τ gets more negative.

To eliminate degeneracy, we add to the linear program the inequality

$$\sum_a q_a^2\, \sigma_{aa} \;+\; \sum_{a,b\,:\,a \neq b} 2\, q_a q_b\, \sigma_{ab} \;\leq\; \tau,$$

Table 1. Dataset characteristics. For sets U, P, Q, and S of PALI benchmarks, the table reports the number of benchmarks in each set, and averaged across its benchmarks, their number of strings, their string length, and the percent identity of their induced pairwise alignments. Also shown averaged for the *core blocks*, or reliable regions of the benchmarks, are their percent coverage of the strings and their percent identity.

					core blocks	
Datasets	benchmarks	strings	length	identity	coverage	identity
U	102	14	239	29.1	40.4	35.8
P	51	15	239	29.7	41.0	37.1
Q	51	12	239	28.1	39.9	33.8
S	25	20	245	27.4	33.2	33.5

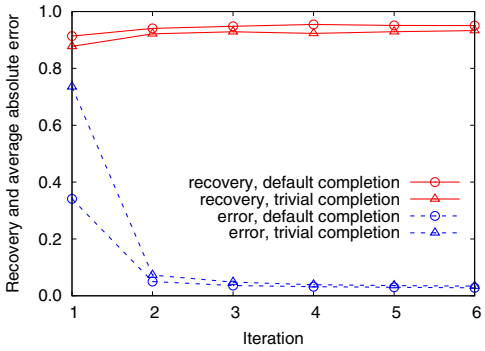

Fig. 1. Improvement in recovery and error for the iterative approach to partial examples. Each curve shows either the recovery or error across the iterations starting from a given initial completion. Recovery is the percentage of columns from reliable regions that are present in an optimal alignment computed using the estimated parameters. Results are plotted for two initial *completions*: the *default*, which aligns the unreliable regions using default parameters, and the *trivial*, which takes all columns of the partial alignment including unreliable ones. The set of examples for the curves is all induced pairwise alignments of the PALI benchmark with SCOP identifier b.1.8.1.

where q_a is the probability of amino acid a appearing in a random protein sequence. This forces the optimal solution x^* of the linear program to be as nondegenerate as the default substitution matrix from which τ was measured. (In our experiments we use the τ value of BLOSUM62.) When τ is negative, which holds for standard scoring schemes, this inequality cuts off the trivial solution $(0, \ldots, 0)$.

4 Experimental Results

To evaluate the performance of this approach to inverse alignment, we ran several types of experiments on biological data. For the examples, we used benchmark alignments from the PALI [1] suite of structural multiple alignments of proteins. For each family from the SCOP [9] classification of protein families, PALI contains

Table 2. Recovery rates for variations of inverse alignment. For PALI benchmarks in set S, the table reports the average recovery rate across the examples, which are all induced pairwise alignments of the benchmark. Recovery is measured using the learned parameters to either compute optimal *pairwise* alignments of the example strings, or to compute a *multiple* alignment of the benchmark strings using the tool Opal [11]. Parameters are learned under the *absolute* or *relative* error criteria. Recovery is also shown for pairwise alignments computed using the BLOSUM62 substitution matrix with learned gap penalties, and for multiple alignments computed with Opal using its *default* parameters, which are BLOSUM62 with carefully-chosen gap penalties. To save space we do not list every benchmark, but the *average* row is across all benchmarks in S. The relative-error binary search is to a precision of 0.01%. When parameters are used for alignment they are rounded to an integer scale of 100.

SCOP	Pairwise alignment			Multiple alignment	
identifier	absolute	relative	BLOSUM62	default	absolute
c.95.1.1	**47.8**	34.3	39.0	45.2	**70.1**
e.3.1.1	**50.6**	33.1	46.2	66.6	**77.4**
c.95.1.2	**69.4**	32.9	46.8	64.9	**82.9**
d.32.1.3	**56.6**	38.9	45.4	64.3	**84.7**
a.127.1.1	**73.1**	47.5	71.1	82.9	**89.8**
a.104.1.1	**80.0**	69.0	80.7	89.6	**90.7**
d.54.1.1	**67.4**	54.3	51.9	70.4	**91.7**
b.43.3.1	**76.0**	57.2	58.6	82.2	**93.4**
d.81.1.1	**85.6**	67.8	69.2	85.2	**98.4**
b.1.8.1	**91.0**	85.9	79.2	89.6	**98.6**
average	**78.6**	64.9	68.3	82.3	**91.5**

a multiple alignment of the sequences of the family members, computed by aligning their three-dimensional structures. In total, PALI has 1655 benchmark alignments, from which we selected a subset U of 102 benchmarks consisting of all alignments with at least 7 sequences that have nontrivial gap structure. We also perform a detailed study of a smaller subset $S \subset U$ containing the 25 benchmarks with the most sequences. Set U was also partitioned into two equal-size subsets P, Q for the purpose of conducting training-set/test-set cross-validation experiments. Table 1 summarizes the characteristics of these datasets. Each of these PALI benchmarks consists of partial (not complete) examples.

Figure 1 illustrates the improvement in error for the iterative approach to partial examples discussed in Section 3. As the error in alignment scores improves across the iterations, the recovery of the example alignments tends to improve as well. Generally, smaller error correlates with higher recovery.

Table 2 shows a detailed comparison of recovery rates from different scenarios for inverse alignment. Parameters are learned from all induced pairwise alignments in a given PALI benchmark, and are applied to the strings in the same benchmark, either to compute pairwise alignments or a multiple alignment of the strings. A key conclusion from this comparison is that the *absolute error* criterion substantially outperforms the relative error criterion with respect to

Table 3. Recovery rates for cross validation experiments on training and test sets. Parameters learned on training sets $\widetilde{U}, \widetilde{P}, \widetilde{Q}$ using the absolute error criterion are applied to test sets U, P, Q. For a generic set X of `PALI` benchmarks, the examples in training set $\widetilde{X}$ are a subset of the induced pairwise alignments of the benchmarks in X. Set $\widetilde{X}$ contains pairwise alignments selected by their recovery rate in a multiple alignment of the benchmark computed with `Opal` using default parameters. Set $\widetilde{\widetilde{X}}$ selects one alignment of median rank from each benchmark, together with a sample of alignments that occur at equally-spaced ranks in the union of the benchmarks in X. Parameters from a given training set were used in `Opal` to compute multiple alignments of the benchmarks in the test set, and the table reports the average benchmark recovery.

Training set characteristics			Test set recovery		
dataset	examples	identity	U	P	Q
$\widetilde{U}$	204	33.1	83.4	84.8	82.0
$\widetilde{P}$	153	34.5	82.9	86.1	82.2
$\widetilde{Q}$	153	31.9	82.8	84.4	81.2

recovery of example alignments. When used for pairwise alignment, the parameters learned using absolute error outperform the standard `BLOSUM62` [7] matrix in recovery by up to 20%; when used for multiple alignment in the tool `Opal` [11], which scores alignments under the sum-of-pairs objective, they outperform the default parameters of `Opal` in recovery by up to 25%. Also note that the recovery rates of parameters when used for multiple alignment are generally much higher than when used for pairwise alignment. In short by performing inverse alignment from partial examples one can learn parameters for multiple sequence alignment that are tailored to a given protein family and that yield very high recovery.

Finally, Table 3 presents recovery results from cross validation experiments. Parameters learned on sparse training sets using the absolute error criterion are applied to full test sets. Their recovery is measured when computing multiple sequence alignments of the benchmarks in the test sets using the learned parameters within `Opal`. Note there is only a small difference in recovery when parameters are applied for multiple sequence alignment to disjoint test sets, compared to their recovery on their training set. This suggests that the absolute error method is not overfitting the parameters to the training data.

To give a sense of running time, performing inverse alignment on a given training set involved around 6 iterations for completing partial examples and took about 4 hours total on a 3.2 GHz Pentium 4 with 1 GB of RAM. An iteration took roughly 40 minutes and required around 4,000 cutting planes.

5 Conclusion

We have explored a new approach to inverse parametric sequence alignment that for the first time carefully treats partial examples. The approach minimizes the average absolute error of alignment scores, and iterates over completions of partial examples. We also studied for the first time the performance of learned

parameters when used for multiple sequence alignment, and showed that a substantial improvement in alignment accuracy can be achieved on individual protein families. Furthermore our results suggest that parameters learned across a sampling of protein families generalize well to other families.

Further Research. Inverse alignment can be extended in several directions: to more general models of protein sequence alignment that use an ensemble of *hydrophobic gap penalties* [4], to formulations that directly incorporate *example recovery* [12], and to formulations that use *regularization* to improve parameter generalization [4,12].

Acknowledgements. We wish to thank Chuong Do for helpful discussions, and Travis Wheeler for assistance with using `Opal` [11]. This research was supported by the US National Science Foundation through grant DBI-0317498.

References

1. Balaji, S., Sujatha, S., Kumar, S.S.C., Srinivasan, N.: PALI: a database of alignments and phylogeny of homologous protein structures. Nucleic Acids Research 29(1), 61–65 (2001)
2. Cook, W., Cunningham, W., Pulleyblank, W., Schrijver, A.: Combinatorial Optimization. John Wiley and Sons, New York (1998)
3. Dayhoff, M.O., Schwartz, R.M., Orcutt, B.C.: A model of evolutionary change in proteins. In: Dayhoff, M.O. (ed.) Atlas of Protein Sequence and Structure, Washington DC. National Biomedical Research Foundation, vol. 5(3), pp. 345–352 (1978)
4. Do, C., Gross, S., Batzoglou, S.: CONTRAlign: discriminative training for protein sequence alignment. In: Proceedings of the 10th ACM Conference on Research in Computational Molecular Biology, pp. 160–174. ACM Press, New York (2006)
5. Eppstein, D.: Setting parameters by example. SIAM Journal on Computing 32(3), 643–653 (2003)
6. Gusfield, D., Stelling, P.: Parametric and inverse-parametric sequence alignment with XPARAL. Methods in Enzymology 266, 481–494 (1996)
7. Henikoff, S., Henikoff, J.G.: Amino acid substitution matrices from protein blocks. Proc. National Academy of Sciences USA 89, 10915–10919 (1992)
8. Kececioglu, J., Kim, E.: Simple and fast inverse alignment. In: Proc. 10th ACM Conference on Research in Computational Molecular Biology, pp. 441–455. ACM Press, New York (2006)
9. Murzin, A.G., Brenner, S.E., Hubbard, T., Chothia, C.: SCOP: a structural classification of proteins database for the investigation of sequences and structures. Journal of Molecular Biology 247, 536–540 (1995)
10. Sun, F., Fernández-Baca, D., Yu, W.: Inverse parametric sequence alignment. Journal of Algorithms 53, 36–54 (2004)
11. Wheeler, T., Kececioglu, J.: Multiple alignment by aligning alignments. In: Proc. 15th Conference on Intelligent Systems for Molecular Biology (2007)
12. Yu, C.-N., Joachims, T., Elber, R., Pillardy, J.: Support vector training of protein alignment models. In: Proceedings of the 11th ACM Conference on Research in Computational Molecular Biology, pp. 253–267. ACM Press, New York (2007)