Estimating the Accuracy of Protein Multiple Alignments Without a Reference

Dan DeBlasio[1,†], Travis Wheeler[2], Vadimir Filkov[3], and John Kececioglu[1]

THE UNIVERSITY OF ARIZONA®

Overview

Estimating the accuracy of a computed multiple sequence alignment without knowing the correct alignment is an important problem. A good accuracy estimator has broad utility, from building a meta-aligner that selects the best output of a collection of aligners, to boosting the accuracy of a single aligner by choosing the best values for alignment parameters. The accuracy of a computed alignment is typically determined with respect to a *reference alignment,* by measuring the fraction of substitutions in the *core columns* of the reference alignment that are present in the computed alignment. We estimate accuracy without knowing the reference by learning a function that combines several easily-computable features of an alignment into a single value.

For protein alignments, we consider 12 independent *features* that contribute to a quality alignment. An accuracy *estimator* is learned that is a polynomial function of these features; its coefficients are determined by minimizing its error with respect to true accuracy using mathematical programming. We evaluate this approach by applying it to the task of *parameter advising*, the problem of choosing alignment scoring parameters from a collection of parameter values in order to maximize the accuracy of a computed alignment. Accuracy is evaluated by comparing to a reference alignment from the BENCH (Edgar, 2009) and PALI (Balaji, 2001) benchmark suites of structurally aligned proteins . Compared to prior methods for selecting good alignments, our estimator outperforms both MOS (Lasmann and Sonnhammer, 2002) and NorMD (Thompson, et al., 2001) for parameter advising, and gives a 15.4% boost in accuracy for the multiple alignment tool `Opal` (Wheeler and Kececioglu, 2007) on the hardest benchmarks.

Accuracy Estimator

We derive an accuracy *estimator* from easily-computable features of an alignment. The estimator is a polynomial function of alignment features. Below we express the estimator E for an alignment A as a quadratic function of the features f_i.

$$E(A) = c_0 + \sum_{\text{features } f_i} c_i\ f_i(A) + \sum_{\text{features } f_i, f_j} c_{ij}\ f_i(A)\ f_j(A) \qquad (1)$$

We determine the coefficients for the constant term c_0, the linear terms c_i and the quadratic terms c_{ij} by mathematical programming. For the resulting estimator E to apply to any alignment A, each of the features f_i must be bounded by a constant. If the value $f_i(A)$ can grow arbitrarily large, then no constant coefficient c_i or c_{ij} can give appropriate relative weights to the features, since alignment accuracy must remain in the bounded range [0,1].

Learning the Estimator

Training by Mathematical Programming

We learn optimal coefficients for the estimator, based on example alignments from suites of benchmarks, by solving a linear or quadratic program that minimizes the error of the estimator. The mathematical program is solved using MATLAB, considering two error criteria for fitting the estimator.

Fitting to Accuracy Values The most straightforward approach is to minimize the squared error between the estimator E(A) and the true accuracy F(A) over a collection of example alignments A. For an estimator E that is a polynomial function of the features, this results in a quadratic programming problem, no matter what the degree is of the estimator polynomial.

$$\text{minimize} \sum_{\text{alignments } A} w_A \Big(E(A) - F(A) \Big)^2 \qquad (2)$$

Minimizing the squared error is sensitive to outliers. For many applications, such as parameter advising, the estimator only needs to be a monotonic function of accuracy, and does not need to fit the actual accuracy values.

Fitting to Accuracy Differences For the less constrained problem of finding an estimator that is monotonic in true accuracy, we consider the differences in accuracy between pairs of alignments A and B. Let P be a collection of ordered pairs (A,B) of example alignments that are monotonic increasing in true accuracy: F(A) < F(B). For each pair (A,B) the mathematical program has a variable d_{AB} that captures the error between the increase in the estimator E and the true accuracy F. The constraints in the program guarantee that d_{AB} is only measures when the difference in estimator under-shoots the difference in accuracy. If the estimator difference exceeds the accuracy difference the estimator is monotonic for this pair and there is no error. The objective function for the program is the sum of the errors to the power p, where p = 1 results in a linear program, and p = 2 gives a quadratic program.

$$\begin{aligned} &\text{minimize} && \sum_{(A,B)\in P} w_{AB} \big(d_{AB}\big)^p \\ &\text{subject to} \\ & && d_{AB} \geq \max \left\{ \Big(F(B) - F(A) \Big) - \Big(E(B) - E(A) \Big), 0 \right\} \end{aligned} \qquad (3)$$

The number of variables and constraints in the program is essentially the number of pairs of example alignments in P. The example alignments are alternate alignments of the sequences in a reference alignment from a benchmark suite. Set P is made up of two types of alignment pairs (A,B): when A and B are from the same, or different references. To select pairs on the same reference, we examine the difference in true accuracy between all pairs and add to P, those above a threshold. To select pairs that come from different references we sample alternate alignments from each reference, form a list of all samples sorted by accuracy and for each alignment in the list we choose the next highest alignment whose accuracy difference is above a threshold.

Example Alignments for Training

Benchmark Suites The suites of reference alignments used to train the coefficients of the estimator were PALI (Balaji *et al.*, 2001) and BENCH (Edgar, 2009), which is a selection of alignments from BAliBASE (Thomson *et al.*, 1999), OXBENCH (Russell *et al.*, 1992) and SABRE (from SABmark, Van Walle, *et al.,* 2005). Summary statistics on these suites are shown below.

Benchmark suite	Number of alignments	Average number of sequences	Average sequence length
SABRE	303	5.5	169.9
OXBench	336	6.4	130.1
BAliBASE	120	5.3	298.8
PALI	102	13.5	238.7
Overall	**861**	**6.8**	**185.4**

Example Alignments The alternate alignments for each reference were computed using 13 choices for parameter values, selected from a set of 556 choices. Each parameter choice consists of internal and external gap-open and gap-extension penalties for the `Opal` multiple alignment tool (Wheeler and Kececioglu, 2007). `Opal` uses a single default parameter choice that achieves good accuracy on average for the benchmarks in these suites; parameter choices other than the default, however, can yield much more accurate alignments for specific input sequences.

Alignment Bins The alignment instances for the benchmarks from these suites cover a wide range of difficulty, as measured by the accuracy achieved by standard alignment tools such as `Opal`. The reference alignments were placed into bins according to their difficulty under `Opal`'s default parameter choice. Easy instances are highly over-represented compared to hard instances in these benchmarks, as shown in the table below.

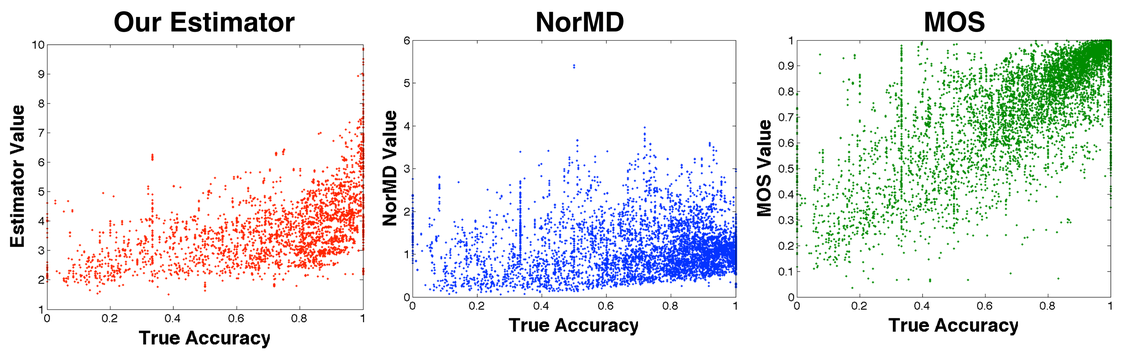
Accuracy bins	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
Reference alignments	13	9	24	40	30	44	66	72	129	434

To correct for this bias in oversampling of easy instances, the objective function for the mathematical program has a weight on each example used in training. The weight w_A on the error term for example alignment A is the reciprocal of the number of reference alignments in the bin for A. The weight w_{AB} for the alignment pair (A,B) is w_A if A and B are from the same reference, and if they are from different references it is the reciprocal of the total number of pairs that involve two references.

Training and Test Sets We evenly divided the reference alignments in each bin into training and testing sets, and generated example alignments from the references in a set by running `Opal` on each of the 13 parameter choices. This produced a training and testing set of 5616 and 5577 total example alignments, respectively.

Experimental Results

Comparing Estimators to True Accuracy



An ideal estimator should be monotonic in true accuracy. For our estimator, MOS, and NorMD, the scatter plots to the left show the trend in the estimator's value and true accuracy for all examples in the test set. Note that this test set is disjoint from our estimator's training set.

Comparing the scatter plots, our estimator has a monotonic trend and low variance. NorMD shows less monotonicity and higher variance, while MOS has better monotonicity and high variance. (To obtain the MOS estimate for an example alignment, MOS took as input all alternate alignments for the example's reference, which is more information than was provided to the other estimators.) The stronger trend for our estimator across all accuracies leads to improved performance for parameter advising.

Application to Parameter Advising

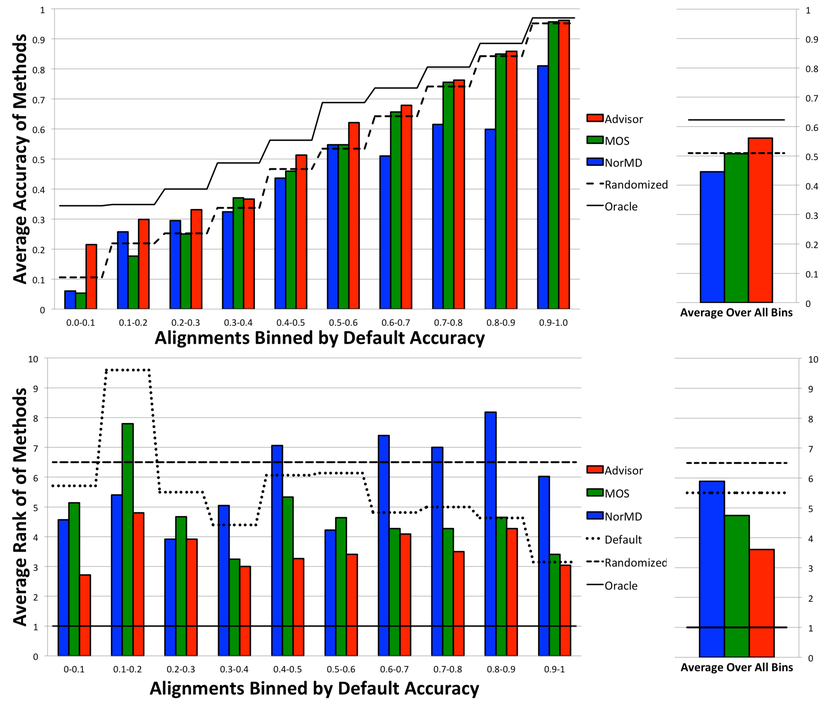
The *parameter advising* task is, given a set of sequences to align and a set of choices of alignment scoring parameters for an alignment tool, to select a parameter choice that maximizes the accuracy of the alignment computed by the tool. For the 13 example alignments for each reference, we selected the example that the estimator assigned the highest value. When we use our estimator for this task we call it the advisor. In addition to MOS and NorMD, we also consider an oracle that always chooses the most accurate example, as well as a randomized estimator that chooses an example uniformly at random and for which we report its expected performance. For comparison, we show the performance using `Opal`'s default parameter choice.

Average Accuracy of Advisor The top figure to the right shows the true accuracy of the alignments chosen by each of the estimators, averaged over each bin.

Our advisor achieves a higher average accuracy than MOS and NorMD in all but one bin. In contrast to MOS and NorMD, our advisor is above random across all accuracies. Compared to the best of MOS and NorMD our estimator is 5% more accurate averaged over all bins, and 15% more accurate on the lowest bin. Compared to a single default parameter choice, our advisor gives a corresponding accuracy increase of 5% and 17%, respectively.

Average Rank of Advisor The bottom figure to the right shows how the alternate alignment for a reference chosen by the estimator ranks in the list of these alignments sorted by accuracy. Note that the rank of the oracle is always 1, and the randomized advisor is always 6.5 (out of 13).

Our advisor outperforms MOS and NorMD in all but one bin (where it ties NorMD). Between MOS and NorMD, MOS is better for higher accuracy alignments, while NorMD is better at lower accuracies.

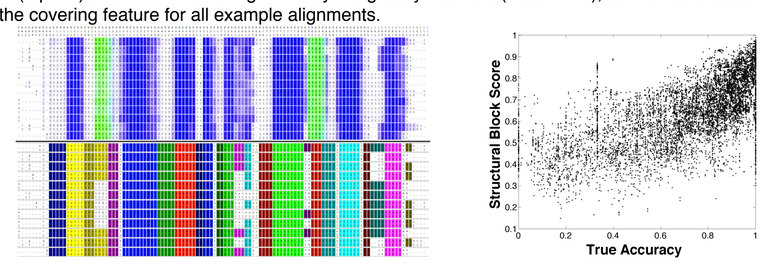


Estimator Features

Our estimator uses 12 feature functions $f_i(A)$ on an alignment A. Key aspects of a good feature function are: (1) it must be efficiently computable, (2) it must be normalized so f_i is bounded by a constant, and finally (3) it should measure some attribute that differentiates high accuracy alignments from others. A few of these features are standard measures, but more than half are novel.

Secondary Structure Block Coverage

For each protein sequence its secondary structure is predicted using PSIPRED (v3.2, Jones 1999). A *block* in an alignment is a subset of the rows and a consecutive run of columns such that all residues in these columns have the same predicted secondary structure. A *covering* of an alignment by blocks is a set of disjoint blocks with at most one block in any column. The *value* of a block is the number of residue pairs in it, and the value of a covering is the sum of the values of its blocks. The *covering feature* is the maximum value of any covering of the alignment, divided by the total number of residue pairs in the alignment. Computing the covering feature exactly appears to be NP-complete; we use an efficient greedy heuristic for approximating it. The heuristic starts with a set of seed rows for each column that have the same structural type, and extends to the left and right, removing rows that do not conform to this type, until adding new columns does not increase the block's value. Each resulting candidate block is scored and a covering set of disjoint blocks is greedily chosen. The figures below show an alignment colored by its residue's predicted structure type (top left) and a block covering found by the greedy heuristic (bottom left), as well as a scatter plot of the covering feature for all example alignments.



Local Secondary Structure Agreement

We use the predicted secondary structure confidences output by PSIPRED to estimate the probability that a pair of residues in an alignment column are both of type alpha-helix or beta-strand. The feature value for a pair of residues is a weighted average of these probabilities in a window of neighboring residues in the alignment, where the central pair has the most weight. The feature value for an alignment is the average value for its substitutions.

Gap Open Density

We count the number of null characters in alignment that start a gap, normalized by the total number of dashes in the alignment. This relates to affine gap penalties (Gotoh, 1993), which are often used to score an alignment.

Gap Extension Density

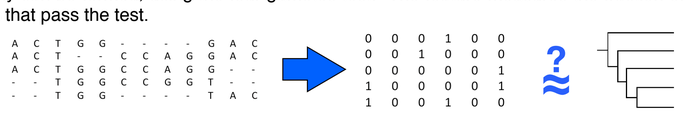
We count the number of null characters in the alignment, normalized by the total number of alignment entries, which also relates to affine gap penalties.

Gap Loop Density

For each pairwise alignment, we measure the fraction of residues in gaps that are predicted by PSIPRED to be of secondary structure type loop (or coil). The feature averages this measure over all pairwise alignments.

Gap Compatibility

The gapping pattern of an alignment is encoded as in cladistics by two binary states: residue or null character. For an alignment in this encoding we collapse adjacent columns that have the same pattern. We test the remaining set of columns for consistency by determining whether a perfect phylogeny can be formed, using the four gametes test. The feature measures the fraction of pairs of columns that pass the test.



Substitution Compatibility

Similar to the gap compatibility feature, we encode the alignment's predicted core columns, where the most prevalent amino acid equivalency class is mapped to 1 and all others to 0. The feature measures the fraction of encoded column pairs that pass the four gametes test.

Information Content

Inspired by Herz and Stormo (1999), this feature measures the average entropy of the alignment, by summing over the columns the log of the ratio of the abundance of a specific amino acid in the column over the background distribution for that amino acid, normalized by the number of columns in the alignment.

Amino Acid Percent Identity

For each column of an alignment, we look at each pair of residues and count the number of pairs that are in the same amino acid equivalency class. We considered standard equivalency classes of 6, 10, 15, and 20 classes.

Secondary Structure Percent Identity

We use the secondary structure sequence for each protein predicted by PSIPRED, and for every induced pairwise alignment we measure the percentage of substitutions that are identities on the secondary structure sequences.

Average Substitution Score

This feature computes the average score of all substitutions in the alignment using the BLOSUM62 (Henikoff and Henikoff, 1992) substitution-scoring matrix.

Predicted Core Column Density

This feature predicts core columns as those without null characters whose substitutions that are identities are above a threshold, and normalizes the count of predicted core columns by the total number of columns in the alignment.

Summary

We provide an efficiently-computable estimator for the accuracy of a protein alignment, without knowing the reference alignment. The estimator is a polynomial function of alignment features, where the coefficients of the polynomial are learned from example alignments using linear and quadratic programming.

We apply our new accuracy estimator to parameter advising, and on average achieve a 5% improvement in accuracy over other methods, with a 15% improvement on the hardest benchmarks.

Further Research There are several directions for further research. An estimator that is a *cubic polynomial*, which provides an inflection point, may better fit the trend observed on the test data. More sophisticated prediction of *core columns* will aid several of our features, as the definition of core column is crucial to the calculation of accuracy. While our current estimator is tailored to protein alignment, expanding the set of features will allow it to be applied to *DNA* and *RNA* alignment. Applying our estimator to develop a *meta-aligner*, which chooses the best output of a collection of aligners, is a potentially fruitful line of research.

References

Balaji, S., Sujatha, S., Kumar, S.S.C. and Srinivasan, N. (2001). PALI-a database of alignments and phylogeny of homologous protein structures. *Nucleic Acids Res.* 2001 29: 61-65.

Edgar, R.C. (2009), http://www.drive5.com/bench

Gotoh, O (1993). Optimal alignment between groups of sequences and its application to multiple sequence alignment. *Comput. Appl. Biosci.* 9, 3:361-70.

Henikoff, S, Henikoff, JG (1992). Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. U.S.A.*, 89, 22:10915-9.

Hertz, GZ, Stormo, GD (1999). Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics* 15, 7-8:563-77

Jones, DT (1999). Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.* 292, 2:195-202.

Lassmann, T, Sonnhammer, EL (2002). Quality assessment of multiple alignment programs. *FEBS Lett.*, 529, 1:126-30.

Russell RB, Barton GJ (1992) Multiple protein sequence alignment from tertiary structure comparison: assignment of global and residue confidence levels. *Proteins* 14:309-323.

Thompson, JD, Plewniak, F, Ripp, R, Thierry, JC, Poch, O (2001). Towards a reliable objective function for multiple sequence alignments. *J. Mol. Biol.* 314, 4:937-51.

Thompson, JD, Plewniak, F, Poch, O (1999). BAliBASE: A benchmark alignments database for the evaluation of multiple sequence alignment programs. *Bioinformatics* 15, 87088.

Van Walle, I, Lasters, I, Wyns, L (2005). SABmark--a benchmark for sequence alignment that covers the entire known fold space. *Bioinformatics*, 21, 7:1267-8.

Wheeler, TJ, Kececioglu, JD (2007a). Multiple alignment by aligning alignments. ISMB'07, *Bioinformatics* 23, 13:i559-68.

Wheeler, TJ, Kececioglu, JD (2007b). `Opal` 0.3.7: Software for multiple sequence alignment by aligning alignments. http://opal.cs.arizona.edu

Affiliations

[1]**University of Arizona**, Department of Computer Science, Tucson AZ
[2]**Janelia Farms Research Campus**, Howard Hughes Medical Institute, Ashburn VA
[3]**University of California at Davis**, Department of Computer Science, Davis CA
[†]Presenting author
Research supported by the NSF IGERT Grant in Comparative Genomics DGE-0654435