A max-margin model for efficient simultaneous alignment and folding of RNA sequences

Chuong B. Do*, Chuan-Sheng Foo and Serafim Batzoglou
Computer Science Department, Stanford University, Stanford, CA 94305, USA

ABSTRACT
Motivation: The need for accurate and efficient tools for computational RNA structure analysis has become increasingly apparent over the last several years: RNA folding algorithms underlie numerous applications in bioinformatics, ranging from microarray probe selection to de novo non-coding RNA gene prediction.

In this work, we present RAF (RNA Alignment and Folding), an efficient algorithm for simultaneous alignment and consensus folding of unaligned RNA sequences. Algorithmically, RAF exploits sparsity in the set of likely pairing and alignment candidates for each nucleotide (as identified by the CONTRAfold or CONTRAlign programs) to achieve an effectively quadratic running time for simultaneous pairwise alignment and folding. RAF's fast sparse dynamic programming, in turn, serves as the inference engine within a discriminative machine learning algorithm for parameter estimation.

Results: In cross-validated benchmark tests, RAF achieves accuracies equaling or surpassing the current best approaches for RNA multiple sequence secondary structure prediction. However, RAF requires nearly an order of magnitude less time than other simultaneous folding and alignment methods, thus making it especially appropriate for high-throughput studies.

Availability: Source code for RAF is available at: http://contra.stanford.edu/contrafold.
Contact: chuongdo@cs.stanford.edu

1 INTRODUCTION
The secondary structure adopted by an RNA molecule in vivo is a vital consideration in many bioinformatics analyses. In PCR primer design, stable secondary structures can obstruct proper binding of the primer to DNA (Diefenbach et al., 1993); in RNA folding pathway studies, secondary structure forms the basic scaffold on which more complicated 3D structures organize (Broun and Westhof, 1997); and in computational non-coding RNA gene prediction, RNA secondary structural stability provides the characteristic signal for distinguishing real RNA sequence from non-functional genes, as secondary structural constraints often exist even when primary sequence conservation is lacking (Torarinsson et al., 2006).

In this article, we describe RNA alignment and folding (RAF), a new algorithm for predicting RNA secondary structure from a collection of unaligned RNA sequences. Algorithmically, RAF belongs to a category of RNA secondary structure prediction methods which simultaneously align and fold RNA sequences. By optimizing a pair of unaligned RNA sequences for both sequence homology and structural conservation concurrently, simultaneous alignment and folding approaches sidestep the usual problem of needing accurate sequence alignments before the folding is done. By exploiting sparsity in the set of likely base pairings and aligned nucleotides, RAF achieves $O(L^4)$ running time for sequences of length $L$, improving significantly upon the $O(L^2)$ running times of typical simultaneous folding and alignment approaches.

The main contribution of RAF, however, is its application of discriminative machine learning techniques for parameter estimation to the problem of simultaneous alignment and folding. Unlike previous methods, RAF's scoring model does not rely on ad hoc combinations of thermodynamic free energies for structural features (Mathews et al., 1999) with arbitrary alignment match and gap penalties (Hofacker et al., 2002), nor does RAF attempt the ambitious task of simultaneously modeling the evolutionary history of both sequences and structure (Knudsen and Hein, 2003).

Instead, RAF defines a fixed set of basis features describing aspects of the alignment, RNA secondary structure, or both. RAF then poses the task of learning weights for these features as a convex optimization problem, giving rise to efficient algorithms with guaranteed convergence to optimality.

The concept of using discriminative methods for parameter estimation rather than relying solely on parameters compiled from experimental measurements originated with the CONTRAfold (Do et al., 2006b) program, and later also became the basis of the CG (Andronescu et al., 2007) method. In a manner analogous to these two previous methods for single sequence secondary structure prediction, RAF demonstrates that automatic learning of parameters can also confer benefits to multiple sequence structure prediction accuracy.

2 METHODS
The RAF algorithm consists of four components: (1) a simple yet flexible objective function for pairwise alignment and folding of unaligned RNA sequences; (2) a fast Sankoff-style inference engine for maximizing this objective function via sparse dynamic programming; (3) a simple progressive strategy for extending the pairwise algorithm to handle multiple unaligned sequence inputs; and (4) a max-margin framework for automatically learning model parameters from training data. We describe each of these in turn.
where \( w_B \) is a linear combination of arbitrary basis features (Appendix A.1): of all conserved base pairings. In turn, RAF models each scoring term as a consensus secondary structure of unaligned input RNA sequences. We refer to a candidate alignment and these sequences (such as generated by a probabilistic HMM). In each panel, the darkness of each square represents the posterior confidence in the corresponding base pairing or alignment match. While the single sequence folder or the pairwise sequence aligner may not be able to identify the single correct folding or alignment, respectively, the set of likely candidate base pairings and matched positions, nonetheless, is extremely sparse.

2.1 The RAF scoring model

We begin our description of the algorithm by describing a scoring scheme for alignments and consensus foldings of two sequences. Let \( a \) and \( b \) be a pair of unaligned input RNA sequences. We refer to a candidate alignment and consensus secondary structure of \( a \) and \( b \) collectively as a parse. Informally, a parse \( y \) for a pair of sequences \( a \) and \( b \) is a set whose elements consist of base pairings \((a_i, a_j)\) belonging to sequence \( a \), base pairings \((b_k, b_l)\) belonging to sequence \( b \), and aligned positions \((a_i, b_k)\) between \( a \) and \( b \).

For a given parse \( y \) from the space of all valid \(^2\) parses \( \mathcal{Y} \), RAF uses a simple scoring scheme which takes into account aligned positions and conserved base pairings. Specifically, RAF defines the score, \( \text{Score}(y, \mathbf{w}) \), of such a parse \( y \) to be:

\[
\sum_{(i, j, k) \in \mathcal{Y}} \psi_{\text{align}}^y(i, j, k) + \sum_{(a_i, b_k) \in \mathcal{R}(y)} \psi_{\text{paired}}^y(a_i, b_k),
\]

where \( \psi_{\text{align}}^y(i, j, k) \) and \( \psi_{\text{paired}}^y(a_i, b_k) \) are scoring terms for aligned positions and conserved base pairs, respectively, and where \( \mathcal{R}(y) \) is the set of all conserved base pairings. In turn, RAF models each scoring term as a linear combination of arbitrary basis features (Appendix A.1):

\[
\psi_{\text{align}}^y(i, j, k) = \sum_{q=1}^{n_q} w_q \psi_{\text{align}}^y(i, j, k),
\]

\[
\psi_{\text{paired}}^y(a_i, b_k) = \sum_{l=1}^{n_l} w_l \phi_{\text{paired}}^y(a_i, b_k),
\]

where \( \mathbf{w} \in \mathbb{R}^{n_q \times 1 + n_l \times 1} \) is a vector of scoring parameters.

2.2 Fast pairwise alignment and folding

Given the scoring scheme described in the previous section, the problem of simultaneous alignment and folding reduces to the optimization problem:

\[
y^* = \arg \max_{y \in \mathcal{Y}} \text{Score}(y, \mathbf{w}),
\]

We say that a parse \( y \) of inputs \( a \) and \( b \) is valid provided that (1) each nucleotide of \( a \) and \( b \) base pairs with at most one other nucleotide in the same sequence; (2) each nucleotide aligns with at most one nucleotide in the opposite sequence; (3) neither sequence contains pseudo-knotted base pairings; (4) the alignment of the two sequences does not contain rearrangements or repeats; and (5) all base pairings are conserved.

In principle, the solution to (1) follows immediately from the original dynamic programming algorithm for simultaneous alignment and folding presented by Sankoff (1985). Sankoff’s algorithm, however, has an \( O(L^2) \) time complexity and \( O(L^3) \) space complexity for \( K \) sequences of length \( L \), rendering it impractical for all but the smallest multiple folding problems. Therefore, most programs for RNA simultaneous alignment and folding use heuristics to reduce time and memory requirements while minimally compromising alignment and structure-prediction quality. Some heuristics used in previous programs have included incorporating structural information into a single alignment scoring matrix (Dalli et al., 2006), disallowing multi-branched loops (Gorodkin et al., 1997), and precomputing potential conserved helices prior to alignment (Tabib et al., 2006; Tueret and Perriquet, 2004).

The most popular heuristics, however, involve reduction of the portion of the dynamic programming matrices (which we call the DP region) that must be computed. For example, some methods restrict the DP region to a strip of fixed width about the diagonal (Hofacker et al., 2004; Mathews and Turner, 2002) or about an initial alignment path (Kiryu et al., 2007). Other methods rely on external single-sequence folding and probabilistic alignment programs to generate base pairing probability matrices (Toutanpassion et al., 2003; Will et al., 2007) or alignment match posterior probability matrices (Kiryu et al., 2007), and then exploit the sparsity of these matrices in order to reduce the amount of computation required.

The RAF algorithm adopts the last of these strategies. Namely, RAF uses a single-sequence RNA secondary structure prediction program (CONTRAfold; Do et al., 2006a) and a pairwise RNA sequence alignment program (CONTRAAlign; Do et al., 2006a)\(^3\) respectively, to construct a constraint set \( C \) of allowed base pairs and aligned positions in \( a \) and \( b \). Given a constraint set \( C \), RAF then replaces (1) with the reduced inference problem,

\[
y^* = \arg \max_{y \in \mathcal{Y}} \text{Score}(y, \mathbf{w}),
\]

where \( \mathcal{Y}_C = \{ y \in \mathcal{Y} : y \subseteq C \} \) is the space of valid parses, restricted to those which contain only base pairings and alignment matches from the constraint set \( C \) (Fig. 1).

To obtain the set of allowed base pairings, RAF uses the implementation of McCaskill’s algorithm (McCaskill, 1990) from CONTRAfold in order to compute the posterior probability of each possible base pairing in sequence \( a \).

\(^3\)The original CONTRAAlign program was designed for protein sequences. We adapted this for RNAs by removing all protein-specific features (e.g. hydrophobicity), modifying the underlying alphabet (A, C, G and U) and simply retraining on the appropriate training set.

![Fig. 1. Sparsity patterns in posterior probability matrices. Panels (a) and (b) illustrate the pairwise pairing posterior probabilities for two different sequences (such as generated by a single-sequence probabilistic or partition function-based RNA folding program). Panel (c) shows the alignment match probabilities for these sequences (such as generated by a probabilistic HMM). In each panel, the darkness of each square represents the posterior confidence in the corresponding base pairing or alignment match. While the single sequence folder or the pairwise sequence aligner may not be able to identify the single correct folding or alignment, respectively, the set of likely candidate base pairings and matched positions, nonetheless, is extremely sparse.](image-url)
and similarly for sequence b. All base pairs with posterior probability at least \( \epsilon_{\text{passed}} \) are then retained. Similarly, to determine the set of allowed alignment matches per position and \( s \) is the minimum allowed distance between adjacent helices.

Table 1. Comparison of computational complexity of RNA simultaneous folding and alignment algorithms

| Algorithm   | Time complexity | Space complexity |
|-------------|-----------------|------------------|
| Sankoff     | \( O(L^4) \)    | \( O(L^3) \)     |
| FOLDALIGN   | \( O(L^4) \)    | \( O(L^3) \)     |
| LocARNAX    | \( O(c_L^2) \)   | \( O(c_L^2) \)   |
| Mallet      | \( O(\eta_{\text{L}}^2 + \eta_{\text{L}} \eta_{\text{S}} + \eta_{\text{S}}^2) \) | \( O(\eta_{\text{L}}^2) \) |
| RAF         | \( O(\min(d, d^2)) \) | \( O(\min(d, d^2)) \) |

Here, \( L \) denotes the sequence length, \( s \) is the number of candidate base pairs per position, \( d \) is the number of candidate alignment matches per position and \( s \) is the minimum allowed distance between adjacent helices.

Using this new subroutine for aligning alignments, we can then perform multiple alignment in RAF using a standard progressive strategy (Feng and Doolittle, 1987). Specifically, we cluster the sequences with a UPGMA (Sneath and Sokal, 1962) tree-building procedure, using the expected accuracy similarity measure (Do et al., 2005). Finally, we perform progressive alignment by aligning subgroups of sequences according to the tree.

2.4 A max-margin framework

Given a set of \( n \) training examples, \( S = \{ (a^{(i)}, b^{(i)}, y^{(i)}) \}_{i=1}^{n} \), the parameter estimation problem is the task of identifying a vector of weights \( w = (w_1, w_2, \ldots, w_n) \in R^n \) for which the RAF inference algorithm, as described in the previous section, will yield accurate alignments and consensus structures. In this section, we present a max-margin framework for parameter estimation in RAF.

2.4.1 Formulation

In the max-margin framework, our goal is to obtain a parameter vector \( w \) for which the RAF inference algorithm will generate accurate alignments and consensus structures. Clearly, this goal is met if for each training example \( (a^{(i)}, b^{(i)}, y^{(i)}) \) from our training set \( S \),

\[
\text{SCORE}(y^{(i)}; w) > \text{SCORE}(y^{(i)}; w') + \xi_{y^{(i)}} \in Y^{(i)}, \forall y^{(i)} \in Y^{(i)} \setminus \{y^{(i)}\},
\]

In such a case, we would be guaranteed that the maximum of (2) is attained for \( y^{(i)} = y^{(i)} \) (provided the true parse \( y^{(i)} \) belongs to \( Y^{(i)} \)), and hence our inference procedure would necessarily return the correct alignment and consensus folding. This intuition is captured in the following convex optimization problem:

\[
\begin{align*}
\minimize_{w \in \mathbb{R}^{d \times d}} & \quad \frac{1}{2} \sum_{i=1}^{n} \xi_i \\
\text{subject to} & \quad \text{SCORE}(y^{(i)}; w) - \text{SCORE}(y^{(i)}; w') \\
& \quad \geq \Delta(\alpha^{(i)}; w') - \Delta(\alpha^{(i)}; w), \quad i = 1, \ldots, n, \quad y^{(i)} \in Y^{(i)}.
\end{align*}
\]

Here, \( C \) is a regularization constant, and \( \Delta(\alpha^{(i)}; y) \) is a non-negative distance measure between pair of parses, conventionally referred to as the loss function, which takes value 0 if and only if its two arguments are equal (Section 2.4.2).

The inequality constraints play the role of (3)—they try to ensure that the training output \( y^{(i)} \) scores higher than any alternative incorrect parse \( y' \) by some positive amount \( \Delta(\alpha^{(i)}; y') \). In cases where this condition is not achieved, the objective function incurs a penalty of \( \xi_i \). Finally, the regularization term \( \| w \|^2 \) is a penalty used to prevent overfitting.

2.4.2 The loss function

The loss function \( \Delta(\alpha^{(i)}; y') \) in (4) plays two significant roles. Technically, the loss function establishes an appropriate scale for the parameters of the problem and prevents the trivial solution, \( w = 0 \). Intuitively, however, the loss function also helps to make the max-margin optimization robust. By choosing a loss function that takes large positive values for incorrect candidate outputs \( y' \) that differ from the true margin output \( y^{(i)} \) in a very critical way, but that takes small positive values for incorrect candidate outputs \( y' \) whose errors are more forgivable, the loss function allows the user to implement a notion of ‘cost’ for different types of mistakes in the max-margin model.

For RAF, we defined the loss function by restricting our attention to four types of parsing errors: (1) \( \text{false positive base-pairings} \ (a_{i}a_{j}) \ not \ in \ y^{(i)}, \) or

또한 비슷하게 서식 A. 모든 반대편에 고유한 가능성을 포함하여 다음을 건설: \( \epsilon_{\text{passed}} \)라는 값은 둘 다 정해진 후에 재정의됩니다. 비슷하게 서식 B. 모든 가능성을 유지하고 있습니다. 각 서식의 기지 포인트는 단일 서열 분할에 있다. 이로 인해 모든 관련된 분할 포인트 값이 명백한 전기점에 대해 사용됩니다. 이전 알고리즘의 경우, 가능성을 제공합니다 예외적으로 정의된 제한 조건 \( C \).

지나친 제한 조건이 없는 경우 \( \| \alpha(\alpha^{(i)}; y') \|^2 \)는 치명적인 오류를 예방하기 위해 사용됩니다.

2.4.2 The loss function

두 가지 중요한 역할을 합니다. 실질적으로 수치 증가를 위해 문제의 파라미터를 결정하고 균형을 방지하는 것이 됩니다. \( w = 0 \). 직관적으로, 그러나, 손실 함수는 최대 고려를 만드는 갱신 효과를 지속적으로 도출합니다. 선택한 손실 함수는 큰 양의 값을 데가치의 사망 오류 \( y' \)와 \( y^{(i)} \) 사이의 유의한 차이를 가지고 있으며, 이는 임의의 사망 오류 \( y^{(i)} \)의 매우 극한 경우를 만듭니다. 그러나, 작은 양의 값을 드는 사망 오류 \( y' \)의 오류는 더 용이한 것으로, 이 손실 함수는 사용자에 의해 정의되는 '비용'의 개념에 대한 다른 유형의 실수를 최대 고려 모델에 적용할 수 있습니다.

위에서 설명한 것과 다음에 설명한 것과 동일합니다. \( \text{false positive base-pairings} \ (a_{i}a_{j}) \ not \ in \ y^{(i)}, \) or

또한 비슷하게 서식 A. 모든 반대편에 고유한 가능성을 포함하여 다음을 건설: \( \epsilon_{\text{passed}} \)라는 값은 둘 다 정해진 후에 재정의됩니다. 비슷하게 서식 B. 모든 가능성을 유지하고 있습니다. 각 서식의 기지 포인트는 단일 서열 분할에 있다. 이로 인해 모든 관련된 분할 포인트 값이 명백한 전기점에 대해 사용됩니다. 이전 알고리즘의 경우, 가능성을 제공합니다 예외적으로 정의된 제한 조건 \( C \).

지나친 제한 조건이 없는 경우 \( \| \alpha(\alpha^{(i)}; y') \|^2 \)는 치명적인 오류를 예방하기 위해 사용됩니다.
RNA alignment and folding

simultaneously in sequence b), (2) false negative base-pairings \((a_i, a_j)\in\gamma^{(i)}\backslash\gamma^{(i)},\) or similarly in sequence b), (3) false positive base-pairings \((a_i, b_j)\in\gamma^{(i)}\backslash\gamma^{(i)},\) and (4) false negative matched bases \((a_i, b_j)\in\gamma^{(i)},\gamma^{(i)}\). Then, we set

\[ \Delta^{(i)}(i, j) = y^{TP\mbox{-}paired}. \]

No. of false positive base pairings

\[ +y^{TP\mbox{-}aligned} \cdot \#a_i \]

false positive matched bases

\[ +y^{FN\mbox{-}paired} \cdot \#a_i \]

false negative base-pairings

\[ +y^{FN\mbox{-}aligned} \cdot \#a_i \]

false negative matched bases.

The numbers \(y^{TP\mbox{-}paired}\), \(y^{TP\mbox{-}aligned}\), \(y^{FN\mbox{-}paired}\), \(y^{FN\mbox{-}aligned}\) are hyperparameters, chosen by the user prior to training the RAf algorithm, to allow the user to express her preference for models with either high sensitivity or high specificity for base-pairing positions and aligned nucleotides.6

2.4.3 Optimization algorithm At first glance, the constrained optimization problem stated in (4) appears to be a standard convex quadratic program and hence solvable using off-the-shelf packages for convex programming. In reality, for each training example, the optimization problem has an exponential number of inequalities, one corresponding to each possible candidate parse \(y^{(i)}\) of the input sequence! Despite our use of constraints sets to reduce the number of allowed candidate outputs, in most cases, this space is still too large to enumerate.

One approach to deal with this problem is an iterative algorithm known as constraint generation (or column generation), as used in the program CG (Andreou et al., 2007). In this approach, the parameter vector \(w_t\) at each time \(t\) is the solution to a reduced version of (4) in which only a small subset of the constraints are retained. Next, one checks if \(w_t\) violates any of the constraints of the original full optimization problem by more than an prescribed tolerance \(\epsilon\). If so, the worst violated constraint is added to the current set of constraints to form a new reduced optimization problem, whose solution, in turn, gives the next iterate \(w_{t+1}\). If not, the optimization algorithm terminates. Each of the optimization problems in the sequence requires a quadratic programming solver.

Here, we take a simpler approach based on the recent SVM training algorithm of (Shalev-Shwartz and Singer, 2007) and Shalev-Shwartz et al. (2007). Omitting details, we begin by converting (4) into an equivalent unconstrained problem: namely, minimize (with respect to \(w\in\mathbb{R}^n\)),

\[
J(w) = \frac{1}{m} \sum_{i=1}^{m} \max \left( \text{Score}(y^{i}, w) + \Delta^{(i)}(i, j) \right) - \frac{1}{2} \|w\|^2 + \frac{1}{2} \|C\|\|w\|^2. \tag{5}
\]

Next, we use strong duality from optimization theory in order to derive an upper bound \(B\) on the norm of the optimal solution of our unconstrained problem (Appendix C.1). Finally, we actually run the optimization procedure by applying the simple update rule,

\[
w_{t+1} = \Pi_B \left( w_t - \frac{1}{C} \nabla J(w_t) \right), \tag{6}
\]

starting from \(w_0 = \mathbf{0}\). Here, \(g_i \in \partial J(w)\) is any subgradient of the objective function \(J(w)\) evaluated at \(w=w_t\), and the operator \(\Pi_B\) projects a vector onto an origin-centered ball of radius \(B\) (i.e. \(\Pi_B(v) = (B/\|v\|) v\) if \(\|v\| > B\) and \(\Pi_B(v) = v\) otherwise). Intuitively, the algorithm works much like a standard gradient descent procedure adapted for non-differentiable objective functions, but with the added twist that the projection operation ensures that the weight vector iterates stay with a region of the parameter space where the optimum is known to exist.

Given an existing routine for computing subgradients of the unconstrained objective, this algorithm can be implemented in a few lines of code with no complicated numerical optimization software. As shown by Singer and Shalev-Shwartz, the algorithm is also quite efficient, requiring only \(O(n/\epsilon)\) iterations to achieve \(\epsilon\) accuracy on a training set of \(m\) examples. An online variant of the algorithm, in which the subgradients \(g_i\) in each step are computed based only on a randomly sampled subset of the training data (e.g. a single example), achieves an \(O(1/\epsilon)\) expected running time, independent of \(m\), the size of the training set.

2.4.4 Subgradient computation Finally, we show how to compute a subgradient \(g_i \in \partial J(w_i)\). In order to simplify notation, define an \(n\)-dimensional vector \(\Phi_i(y)\) whose \(p\)th component is

\[
\Phi_p(y) = \sum_{\substack{i, j, k \text{ s.t. } (i, j) \in (\gamma^{(i)} \setminus \gamma^{(i)}) \cap (\gamma^{(i)} \setminus \gamma^{(i)}, \gamma^{(i)}) \cap (\gamma^{(i)}, \gamma^{(i)} \setminus \gamma^{(i)})}} \phi^{(i)}(y_{i, j, k}), \tag{7}
\]

where \(\phi^{(i)}(\cdot)\) is simply any \(y^{(i)}\) which attains the maximum in the \(r\)th term of the summation in (5), for \(w \in \mathbb{R}^n\). Each ‘loss-augmented’ maximization, in turn, is easily performed by modifying the original RAf inference procedure to incorporate an approximately defined additional scoring matrix, \(\phi_0(y_{i, j, k})\), with fixed weight \(w_0 = 1\).

3 RESULTS

To evaluate the performance of RAf on real data, we collected training and testing data from a variety of sources. In particular, for training, we obtained Rfam 8.1 (Griffiths-Jones et al., 2005), a database of alignments and covariance models for RNA families along with annotated secondary structures where available. For testing, we obtained BRAIBASE II (Gardner et al., 2005), a benchmark set for RNA alignment programs. We also obtained a testing set of RNA families used by the authors of the recent program, MASTR (Lindgreen et al., 2007).

An important concern in the validation of RNA alignment programs is the confounding factor that unless cross-validation is properly performed, the performance that one sees on any given validation set is not likely to be a reliable judge of the program’s performance on future data. Even in cases where the training and evaluation tests are disjoint but still contain sequences from the same RNA family, evaluation can still give misleading results, because the weights learned for loop lengths and composition will be biased toward specific properties of that RNA family.

To be absolutely sure of no contamination between training and testing data, we preprocessed our Rfam training set of alignments and consensus structures (October 2007 version, 607 families) by excluding all families for which neither of the two testing databases contained an example from that family. We then also removed all families for which only automatically predicted consensus structures were known, leaving a total of 154 families. Finally, we generated a training set \(T_T\) of up to 10 randomly sampled pairwise alignments with consensus structures from each remaining family (1361 pairwise alignments in total), a training set \(T_C\), of up to 10 randomly sampled sequences with structures from each family (1179 sequences in total), and a training set \(T_S\), containing one
To evaluate the quality of the resulting alignments, we used five
values of \( \epsilon \) to observe the effects of different cutoffs.
matches can be retained when each sparsity factor is roughly 10.7
Figure, nearly complete coverage of base pairings and alignment
averaged over the entire training set) for each cutoff. A plot of these
number of pairing partners or matching partners for any nucleotide,
reference aligned matches recovered for varying cutoff constraints.
RAF at a significant disadvantage in the comparisons shown here.
Nonetheless, as shown in the following sections, RAF performs well,
sets used for CONTRAlign and CONTRAfold, and thus places
significantly reduces both the size and coverage of the training
each multiple alignment of
randomly sampled five-way multiple alignment from each family (118 multiple alignments in total).
RAF uses two external programs, CONTRAlign (Do et al., 2006a)
and CONTRAfold (Do et al., 2006b), to compute alignment match
and base-pairing posterior probabilities, respectively. To ensure
proper cross-validation, CONTRAlign was retrained from scratch
using CONTRAfold was retrained using \( T_2 \). Finally, the
RAF algorithm itself was trained using all pairwise projections of
each multiple alignment of \( T_3 \). Our strict cross-validation procedure
significantly reduces both the size and coverage of the training
sets used for CONTRAlign and CONTRAfold, and thus places
RAF at a significant disadvantage in the comparisons shown here.
Nonetheless, as shown in the following sections, RAF performs well,
indicating its ability to generalize for sequences not present in the
training set.

3.1 Alignment and base-pairing constraints
To observe the effects of different cutoffs \( \epsilon_{aligned} \) and \( \epsilon_{spaced} \),
we computed the proportions of reference base pairings and reference aligned matches recovered for varying cutoff constraints.
In addition, we also computed the sparsity ratio (i.e. the maximum
number of pairing partners or matching partners for any nucleotide,
averaged over the entire training set) for each cutoff. A plot of these
two values for training set \( T_3 \) is shown in Figure 2. As seen in
the figure, nearly complete coverage of base pairings and alignment
matches can be retained when each sparsity factor is roughly 10.7

3.2 Evaluation metrics
To evaluate the quality of the resulting alignments, we used five
different scoring measures:

(1) the standard sum-of-pairs (SP) score (Thompson et al., 1999),
which computes the proportion of matches in a reference alignment
which are present in the predicted alignment,

\[
\text{Proportion of reference recovered} = \frac{\text{aligned matches}}{\text{base pairs}}
\]

(2) sensitivity (Sens), the proportion of base pairings in a
reference parse which are recovered in the predicted parse,
(3) specificity or positive-predictive value (PPV), the proportion
of base pairings in a predicted parse which are also present
in the reference parse, and
(4) the Matthews correlation coefficient (MCC) (Matthews,
1975), which we approximate as \( \sqrt{\text{Sens} \cdot \text{PPV}} \), following
Gorodkin et al. (2001).

3.3 Comparison of accuracy
In our first accuracy assessment, we evaluated RAF as well as
a number of other current RNA secondary structure prediction
programs using the BRAliBASE II dataset. In particular, the first
dataset from BRAliBASE II contains collections of 100 five-
sequence subalignments, sampled from five specific Rfam families
(5S rRNA, group II intron, SRP, tRNA and U5). For each of
these alignments, we ran a number of current RNA secondary structure prediction
programs using the BRAliBASE II dataset. In particular, the first

\[
\begin{array}{cccccc}
\text{Dataset} & \text{Program} & \text{Time (s)} & \text{SP} & \text{Sens} & \text{PPV} & \text{MCC} \\
55\text{ rRNA} & \text{Murlet} & 687 & 0.84 & 0.70 & 0.70 & 0.70 \\
& \text{LocARN} & 812 & 0.93 & 0.65 & 0.60 & 0.57 \\
& \text{RNA Sampler} & 2361 & 0.90 & 0.55 & 0.64 & 0.59 \\
& \text{RAF} & 87 & 0.95 & 0.66 & 0.66 & 0.66 \\
\text{group II intron} & \text{Murlet} & 962 & 0.78 & 0.76 & 0.76 & 0.75 \\
& \text{LocARN} & 250 & 0.74 & 0.79 & 0.65 & 0.72 \\
& \text{RNA Sampler} & 1626 & 0.72 & 0.77 & 0.65 & 0.71 \\
& \text{RAF} & 48 & 0.78 & 0.83 & 0.65 & 0.73 \\
\text{SRP} & \text{Murlet} & 20548 & 0.88 & 0.75 & 0.78 & 0.76 \\
& \text{LocARN} & 22467 & 0.85 & 0.66 & 0.70 & 0.68 \\
& \text{RAF} & 1290 & 0.87 & 0.72 & 0.71 & 0.70 \\
\text{tRNA} & \text{Murlet} & 525 & 0.93 & 0.86 & 0.90 & 0.88 \\
& \text{LocARN} & 246 & 0.93 & 0.86 & 0.90 & 0.88 \\
& \text{RNA Sampler} & 763 & 0.92 & 0.93 & 0.91 & 0.92 \\
& \text{RAF} & 52 & 0.94 & 0.81 & 0.85 & 0.83 \\
\text{US} & \text{Murlet} & 1772 & 0.84 & 0.69 & 0.75 & 0.72 \\
& \text{LocARN} & 549 & 0.80 & 0.56 & 0.61 & 0.58 \\
& \text{RNA Sampler} & 4084 & 0.77 & 0.75 & 0.70 & 0.72 \\
& \text{RAF} & 99 & 0.82 & 0.83 & 0.79 & 0.81 \\
\end{array}
\]

As seen from the table, on the BRAliBASE II benchmark, RAF
attains comparable accuracy to the other methods, achieving either
the best or second-best overall accuracy according to MCC on four
out of the five datasets. The running time of the method, however, is
dramatically faster than the other algorithms, often taking an order
of magnitude less time than many of the other programs.

We also obtained the dataset used in the benchmarking of the
MASTR RNA secondary structure prediction program. For a number
of different programs, pre-generated predictions for each input file

\[
\text{Proportion of reference recovered} = \frac{\text{aligned matches}}{\text{base pairs}}
\]

\[
\text{Proportion of reference recovered} = \frac{\text{aligned matches}}{\text{base pairs}}
\]
Table 3. Performance comparison on MASTR benchmarking sets. The best number in each column is marked in bold.

| Program         | SP  | Sens | PPV  | MCC  |
|-----------------|-----|------|------|------|
| CLUSTAL W + Alifold | 0.81| 0.57| 0.73| 0.65|
| FoldalignM       | 0.78| 0.38| 0.81| 0.55|
| LocARNa         | 0.75| 0.41| 0.77| 0.56|
| MASTR           | 0.84| 0.64| 0.73| 0.68|
| Mfold           | 0.89| 0.62| 0.78| 0.70|
| RNAforester      | 0.53| 0.55| 0.55| 0.55|
| RNA Sampler      | 0.82| 0.65| 0.70| 0.67|
| RAF             | 0.88| 0.68| 0.77| 0.72|

are available for download on the MASTR website. In addition to scoring these pre-generated predictions, we also generated and scored predictions using Murlet and RAF. The results are shown in Table 3. In this benchmark set, RAF obtains the highest overall MCC.

We emphasize, however, that benchmarking results such as these should be taken with a grain of salt; both the BRAliBASE II and MASTR benchmarking sets are extremely restricted in their coverage of the space of RNA families, choosing to focus on a few individual RNA families only. As a result, methods carefully tuned to the benchmarks may perform less well on diverse RNA families not found in either of these benchmarks. By using cross-validation, we improve the chances that RAF’s validation results really do indicate reliable out-of-sample performance.

We also note that the performance of RAF on particular RNA families is often closely related to the accuracy of the underlying alignment and single-sequence models used to derive folding and alignment constraints. Because the tools involved in the RAF pipeline all rely on automatic parameter learning, RAF allows the possibility of learning custom parameter sets well-suited for predictions on particular RNA families.

4 DISCUSSION

We presented RAF, a new tool for simultaneous folding and alignment of RNA sequences which exploits sparsity in base pairing and alignment probability matrices and max-margin training in order to achieve faster running times and higher accuracy than previous tools.

Besides its speed, one principal advantage of the RAF methodology is its use of a flexible scoring function for combining an arbitrary set of functions into a coherent objective function for alignment scoring. The ability to introduce new basis scoring functions into the RAF scoring model means that there remains a rich space of possible features to explore.

In addition, the use of the max-margin framework to identify relevant linear combinations of scoring functions has other promising potential applications. For example, Wallace et al. (2006) recently introduced M-Coffee, a meta-algorithm for protein sequence alignment, which combines the results of several different protein sequence alignment programs using the T-Coffee framework.

The difficulty of identifying appropriate weights for the various programs used in the M-Coffee scoring scheme (i.e. some heuristically derived tree-based weights the authors tried did not give a significant improvement in accuracy over flat weights), led the authors to rely on a uniform weight model, treating programs known to be more accurate on equal footing with less accurate aligners. The max-margin framework developed in this paper obviates the need for heuristically-derived weights altogether.

ACKNOWLEDGEMENTS

C.B.D. was supported by an NSF Graduate Research Fellowship. C.S.F. was supported by an A*STAR National Science Scholarship. This material is based in part upon work supported by the NSF under grant number EF-0312459.

Conflict of Interest: none declared.

REFERENCES

Andronescu,M. et al. (2007) Efficient parameter estimation for RNA secondary structure prediction. Bioinformatics, 23, 19–28.

Böttcher,M. et al. (2003) Conser analysis and optimization. Athena Scientific.

Bromley,J. and Hecht,E. (1997) Hierarchy and dynamics of RNA folding. Ann. Rev. Biophys. Biomol. Struct., 26, 113–137.

Dai,J. et al. (2006) STRAL: progressive alignment of non-coding RNA using base pairing probability vectors in quadratic time. Bioinformatics, 22, 1593–1599.

Diefenbach,C.R. et al. (1993) General concepts for PCR primer design. PCR Methods Appl., 3, 30–37.

Do,C.B. et al. (2005) PRORCONS: probabilistic consistency-based multiple alignment. Genome Res., 15, 330–340.

Do,C.B. et al. (2006a) CONTRAfold: RNA secondary structure prediction without physics-based models. Bioinformatics, 22, 490–498.

Dowell,M.D. and Eddy,S.R. et al. (2006) Efficient pairwise RNA structure prediction and alignment using sequence alignment constraints. BMC Bioinformatics, 7, 400.

Eddy,S.R. et al. (2002) Computational genomics of noncoding RNA genes. Cell, 109, 137–140.

Feng,D.F. and Doolittle,R.F. (1987) Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J. Mol. Evol., 25, 351–360.

Gardner,P.P. et al. (2005) A benchmark of multiple alignment programs upon structural RNAs. Nucl. Acids Res., 33, 2433–2439.

Gorodkin,J. et al. (1997) Finding the most significant common sequence and structure motifs in a set of RNA sequences. Nucl. Acids Res., 25, 3724–3732.

Gorodkin,J. et al. (2001) Discovering common stem-loop motifs in unaligned RNA sequences. Nucl. Acids Res., 29, 2135–2144.

Griffiths-Jones,S. et al. (2005) Rfam: annotating non-coding RNAs in complete genomes. Nucl. Acids Res., 33, D123–D124.

Hamvas,A.O. et al. (2007) Efficient pairwise RNA structure prediction using probabilistic alignment constraints in Dynalign. BMC Bioinformatics, 8, 189.

Hofacker,L.L. et al. (1994) Fast folding and comparison of RNA secondary structures (The Vienna RNA Package). Monatsh. Chem., 125, 187–188.

Hofacker,L.L. et al. (2002) Secondary structure prediction for aligned RNA sequences. J. Mol. Biol., 319, 1059–1066.

Hofacker,L.L. et al. (2004) Alignment of RNA base pairing probability matrices. Bioinformatics, 20, 2222–2227.

Holmes,J.L. et al. (2005) Accelerated probabilistic inference of RNA structure evolution. BMC Bioinformatics, 6.

Kuipers,B. et al. (2007) Murlet: a practical multiple alignment tool for structural RNA sequences. Bioinformatics, 23, 1588–1598.

Knudsen,B. and Heits,J. et al. (2003) Fold: RNA secondary structure prediction using stochastic context-free grammars. Nucl. Acids Res., 31, 3423–3428.

Lindgren,S. et al. (2007) MASTR: multiple alignment and structure prediction of non-coding RNAs using simulated annealing. Bioinformatics, 23, 3304–3311.

Matthews,B.D. and Turner,D.H. (2002) Dynalign: an algorithm for finding the secondary structure common to two RNA sequences. J. Mol. Biol., 317, 191–203.

Matthews,B.D. et al. (1999) Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. J. Mol. Biol., 288, 911–940.

Matthews,B.W. (1975) Comparison of predicted and observed secondary structure of T4 phage lysozyme. Biochim. Biophys. Acta, 405, 442–451.
The features used by the RAF program, as evaluated in this article, consist of alignment features, \(\{\varphi_{\text{paired}}(i, k)\}\) and pairing features, \(\{\varphi_{\text{paired}}(i, j, k, l)\}\). Specifically, the alignment features, \(\varphi_{\text{aligned}}(i, k)\) ∈ \(\mathbb{R}^4\) for a candidate alignment match \((a_i, b_k)\) are

\[
\begin{align*}
\varphi_{\text{aligned}}(i, k) &= \frac{P(a_i \text{ aligns with } b_k)}{\log P(a_i \text{ aligns with } b_k)} \\
&= \log P(a_i \text{ aligns with } b_k) - \log P(a_i \text{ unaligned} \land b_k \text{ unaligned})
\end{align*}
\] (A1)

The pairing features, \(\varphi_{\text{paired}}(i, j, k, l)\) ∈ \(\mathbb{R}^4\) for a preserved base pairing \((a_i, a_j)\) and \((b_k, b_l)\) are given by \(\varphi_{\text{paired}}(i, j, k, l) = \varphi_{\text{paired}}(a_i, a_j) + \varphi_{\text{paired}}(b_k, b_l)\). In turn, \(\varphi_{\text{paired}}(a_i, a_j)\) ∈ \(\mathbb{R}^4\) is given by

\[
\begin{align*}
\varphi_{\text{paired}}(a_i, a_j) &= \frac{P(a_i \text{ pairs with } a_j)}{\log P(a_i \text{ pairs with } a_j)} \\
&= \log P(a_i \text{ pairs with } a_j) - \log P(a_i \text{ unpaired} \land P(a_j \text{ unpaired})
\end{align*}
\] (A2)

and similarly for \(\varphi_{\text{paired}}(b_k, b_l)\). Thus, the model contains a total of eight features whose weights need to be learned. Here, the posterior probabilities for aligned positions and base-pairing positions are computed using the CONTRAAlign (Do et al., 2006a) and CONTRAfold (Do et al., 2006b) programs, respectively.

### A.1 THE RAF INFECTION ENGINE

In the section, we describe the RAF inference engine for fast approximate simultaneous alignment and consensus folding of pairs of sequences. In particular, we first present some exact recurrences for alignment and folding, and then use restrictions on the set of allowed base pairings and aligned positions to achieve an improvement in computational complexity.

#### B.1 Recurrences

First, we describe a straightforward \(O(L^6)\) dynamic programming recurrence for computing the optimal simultaneous alignment and consensus fold for a pair of sequences \(a\) and \(b\).

To compute the optimal parse of \(a\) and \(b\), we construct 2 four-dimensional matrices, \(S\) and \(D\). Here, \(S_{i, j, k, l}\) denotes the optimal score for aligning and folding \(a_i + a_{i+1} + \ldots + a_j\) with \(b_k + b_{k+1} + \ldots + b_l\). Furthermore, \(D_{i, j, k, l}\) denotes the optimal score for aligning and folding the same substrings, subject to the additional constraint that the outermost positions \((a_i, a_j)\) and \((b_k, b_l)\) form conserved base pairs.

For \(0 \leq i < j \leq |a|\) and \(0 \leq k < l \leq |b|\), we have

\[
S_{i, j, k, l} = \max_{0 \leq i < j \leq |a|, 0 \leq k < l \leq |b|} \left\{ \begin{array}{ll}
0 & \text{if } i = j \text{ and } k = l \\
S_{i, j-1, k, l} + \psi_{\text{aligned}}(j, l) & \text{if } j > 1 \text{ and } l > k \\
S_{i, j-1, k, l-1} + \psi_{\text{aligned}}(j, l) & \text{if } j > 1 \text{ and } l = k \\
S_{i, j, k-1, l} + \psi_{\text{aligned}}(i, j) & \text{if } i > 1 \text{ and } j = k \\
S_{i, j, k, l-1} & \text{otherwise}
\end{array} \right.
\] (B1)

and for \(0 \leq i < j \leq |a|\) and \(0 \leq k < l + 2 \leq |b|\), we have

\[
S_{i, j, k, l} = \max_{0 \leq i < j \leq |a|, 0 \leq k < l \leq |b|} \left\{ \begin{array}{ll}
0 & \text{if } i = j \text{ and } k = l \\
S_{i, j-1, k-1, l-1} + \psi_{\text{aligned}}(i, j) + \psi_{\text{aligned}}(i+1, j+1, k+1, l+1) & \text{if } i > 1 \text{ and } j > 1 \text{ and } k > 1 \text{ and } l > 1 \\
S_{i, j-1, k-1, l-1} & \text{otherwise}
\end{array} \right.
\] (B2)

Here, recurrence (B1) takes the form of a standard Needleman-Wunsch procedure for aligning the substrings \(a_i + a_{i+1} + \ldots + a_j\) with \(b_k + b_{k+1} + \ldots + b_l\), with an extra case to handle bifurcations in the base-pairing structure of the RNAs. At the end of the recurrence, \(S_{i, j, k, l}\) gives the score of the optimal alignment and consensus fold of the input sequences \(a\) and \(b\). By using traceback pointers in the standard way, the optimal parse can be recovered easily once the recurrence has been evaluated.

In the next section, we explore how these recurrences may be sped up considerably if a constraint set \(C\) of allowed base pairings and aligned positions is known ahead of time. For complexity analysis, we assume \(O(c)\) and \(O(d)\) bounds on the number of candidate base pairing and alignment partners per sequence position, respectively.

#### B.1.2 Exploiting base-pairing sparsity

LocARNAs (Will et al., 2007) was the first program for simultaneous alignment and folding of RNA to take advantage of base pairing sparsity in a manner that significantly improved in both running time and memory usage. In this section, we recount the innovations of LocARNAs as they are applied in RAF. In the next section, we extend these ideas to also account for alignment sparsity.

First, observe that since all parses in \(Y^4_i\) contain only conserved base pairings, the evaluation of (B2) may be restricted to only
those $D_{j,k+1}$ cells for which both $(a_{j+1}, a_j) \in C$ and $(b_{k+1}, b_k) \in C$. Similarly, the inner loop for considering bicausations in (B1) may also be restricted to only those $f'$ for which both $(a_{j+1}, a_j) \in C$ and $(b_{k+1}, b_k) \in C$. Since the bottleneck in the dynamic programming complexity is the number of executions of the innermost loop in (B1), it follows that restricting the considered bicausations in the manner described above yields an $O(c^2L^3)$ running time; in particular, for each $i$ and $k$, computing all values of $S_{i,i,k}$ takes $O(c^2L^2)$ time as each entry of the $D$ matrix is touched at most once. This optimization was originally implemented as part of the LocARNA (Will et al., 2007) and FoldAlignM (Torarinsson et al., 2007) algorithms.

Second, consider the task of computing all entries in the $D$ matrix. From (B2), we see that the values $D_{i,i,k}$ depend only on $S_{i+1,i,k+1}$. Similarly, from (B1), the values $S_{i+1,i,k+1}$ depend only on $D_{j,j,f'}$, for $f' \geq i+1$ and $f' \geq k+1$. Thus, ordering computations in the following way allows the recurrences to be evaluated in a single pass:

For $i \leftarrow |a| - 2$ downto 0
For $k \leftarrow |b| - 2$ downto 0
Compute $S_{i+1,i,k+1}$
Compute $D_{i,i,k}$

Furthermore, since $S_{i+1,i,k+1}$ is only needed while computing $D_{i,i,k}$ (but not for any later values of $i$ and $k$), we need only to retain one $S_{i+1,i,k+1}$ matrix in memory at any given time while computing the $D$ matrix. This observation was originally incorporated in the LocARNA program of Will et al. (2007).

Finally, observe that once the $D$ matrix has been computed, the score $S_{0,|a|,|b|}$ of the optimal parse is easily obtainable in $O(c^2L^2)$ time by recomputing $S_{0,|a|,|b|}$. Likewise, computing the full traceback requires at most $O(c^2L^3)$ time, negligible relative to the cost of computing the $D$ matrix itself. Thus, we obtain an overall $O(c^2L^2)$ time complexity with $O(c^2L^3)$ space complexity (for storing the $D$ matrix).

### B.1.3 Exploiting alignment sparsity

To exploit sparsity in the set of allowed aligned positions in $C$, we again use the strategy of limiting the $D$-region. We accomplish this by first considering the simpler problem of computing the reduced $D$ region, $A$ (known as the alignment envelope) for pairwise sequence alignment without scoring rules. Using $A$, we then define a reduced $D$ region for our original alignment and folding task.

For the first step, consider the following restatement of recurrence (B1) using the notation $S_{j,l} = S_{0,|a|,|b|}$ where we have omitted the case involving bicausations/base-pairing:

\[
S_{j,l} = \max
\begin{cases}
0 & \text{if } j = 0 \text{ and } l = 0 \\
S_{j-1,l} & \text{if } j > 0 \text{ and } l > 0 \\
S_{j-1,l-1} + \psi^{\text{aligned}}(a_j,b_l) & \text{if } j > 0 \text{ and } l > 0 .
\end{cases}
\]

As before, $S_{j,l}$ represents the optimal score of aligning $a_1 \ldots a_j$ to $b_1 \ldots b_l$. Here, our goal is to find $A$, the minimal set of cells containing no holes, such that for every parse $\gamma \in C$, there exists some DP path through $A$ corresponding to an alignment with the same set of aligned positions. Under the assumption that $A$ contains no holes, we can represent $A$ by keeping track of its boundaries:

for each $j \in \{0, 1, \ldots, |a|\}$, let $[A_{\text{First}}(j), A_{\text{Last}}(j)]$ denote the first and last positions $l \in \{0, 1, \ldots, |b|\}$ such that $S_{l,j} \in A$.

We compute these boundaries in linear time using the following procedure. First, we adjust the boundaries to include $S_{0,|a|,|b|} \in A$ and $S_{j+1,|a|,|b|} \in A$ for each candidate aligning pair $(a_j,b_l) \in C$. In addition, we also include the corners $S_{0,0}$ and $S_{|a|,|b|} \in A$. Finally, we force the boundaries of $A$ to satisfy the monotonicity conditions

\[
A_{\text{First}}[0] \leq A_{\text{First}}[1] \leq \cdots \leq A_{\text{First}}[|a|],
\]

\[
A_{\text{Last}}[0] \leq A_{\text{Last}}[1] \leq \cdots \leq A_{\text{Last}}[|b|],
\]

in such a way that guarantees all DP cells $S_{j,l} \in A$ are accessible via some DP path from $S_{0,0}$ to $S_{|a|,|b|}$.

For the second step, we define the reduced $D$ region for our original simultaneous alignment and folding recurrences as the set $\mathcal{R}$ of all positions $S_{j+k,l+k}$ such that $S_{l,j+k} \in A$ and $S_{j+l,k} \in A$. To use this reduced $D$ region $\mathcal{R}$, then, we simply force $S_{l,j+k} = -\infty$ for all $S_{l,j+k} \not\in \mathcal{R}$. Under this restriction, we can reduce the amount of computation performed in the recurrence (B1) by iterating only over cells $S_{l,j+k} \in \mathcal{R}$ and, similarly, restricting the evaluation of the $D$ matrix in (B2) to only those cells $D_{j,k}$ for which $S_{j,l+k} \in \mathcal{R}$. To ensure that each allowed parse belongs to $\mathcal{Y}$, we could penalize any base pairing or aligned position not in $C$ by $-\infty$. In practice, we instead augment $C$ to include all aligned matches allowed by $\mathcal{R}$, since this can be done at no increase in computational complexity.

To analyze the new computational complexity of the algorithm, we begin by bounding the size of $D$ matrix in two different ways. First, for each of the $O(cL)$ base pairs $(a_j,b_k) \in C$, there are $O(d)$ aligning partners for $a_j$ and $O(d)$ aligning partners for $b_k$, giving a total size of $O(c^2dL)$. Alternatively, for each of the $O(dL)$ aligning pairs $(a_j,b_k) \in C$, there are $O(c)$ base-pairing partners for $a_j$ and $O(c)$ base-pairing partners for $b_k$, giving a total size of $O(c^2dL)$. Thus, the size of the $D$ matrix is $O(\min(c,d) cdL)$.

As in Section B.1.2, the space complexity of the algorithm is dominated by cost of storing the $D$ matrix, and hence, is $O(\min(c,d) cdL)$. Similarly, the time complexity can be estimated as the number of evaluations of the innermost loop in the bifurcation case of (B1). Since the innermost loop touches each entry of the $D$ matrix at most once for each $i$ and $k$, and since there are $O(cL)$ choices of $(a_j,b_k) \in \mathcal{A}$, it follows that the time complexity of the algorithm is $O(\min(c,d) c^2dL^2)^{\text{4}}$.

### C.1 NORM BOUND

In this section, we derive a bound on the maximum norm of the optimal parameter vector $w^*$ for (4). From standard arguments (see, $^4$Note that in these bounds, we assume an $O(cL)$ bound on the number of base-pairing partners per position, and an $O(dL)$ bound on the number of aligning partners per position. A weaker condition would be to assume an $O(cL)$ bound on the total number of candidate base-pairing partners for sequences $a$ and $b$ and similarly, an $O(dL)$ bound on the total number of candidate aligned positions; under these conditions, we obtain a worst-case space complexity of $O(\min(c,d) c^2dL^2)$ and a worst-case time complexity of $O(\min(c,d) c^2dL^2)$.}
e.g. Taskar et al., 2003), the dual optimization problem is
\[
\max_{\alpha \in \Lambda} \sum_{i=1}^{m} \sum_{y \in \bar{Y}(\alpha)} a_{i,y} \Delta(y(i), y') - \frac{1}{2} C||w(\alpha)||^2
\]
where
\[
\Lambda = \left\{ (\alpha_{i,y}): \alpha_{i,y} \geq 0, \sum_{y' \in \bar{Y}(\alpha)} \frac{1}{m} \right\}
\]
\[
w(\alpha) = \frac{1}{m} \sum_{i=1}^{m} \sum_{y \in \bar{Y}(\alpha)} a_{i,y} (F(x,y(i)) - F(x,y')).
\]
By strong duality, for any solutions \((w^*, \xi^*)\) and \(\alpha^*\) of the primal and dual optimization problems, respectively, the values of the primal and dual objectives must be equal, i.e.,
\[
\frac{1}{2} C||w^*||^2 + \frac{1}{m} \sum_{i=1}^{m} \xi_i^* = \sum_{i=1}^{m} \sum_{y \in \bar{Y}(\alpha)} a_{i,y} \Delta(y(i), y') - \frac{1}{2} C||w(\alpha^*)||^2.
\]
Now, suppose that \(D_i \in \mathbb{R}\) for \(i = 1, \ldots, m\) satisfy
\[
D_i \geq \max_{y \in \bar{Y}(\alpha)} \Delta(y(i), y').
\]
In the case of the RAF loss function, for example, we can use
\[
D_i = (|a| + |b|)(\gamma_{FP \text{ paired}} + \gamma_{FN \text{ paired}} + \gamma_{FP \text{ aligned}} + \gamma_{FN \text{ aligned}}).
\]
Then the KKT optimality condition \(w^* = w(\alpha^*)\), the primal constraint that \(\xi_i^* \geq 0\) for \(i = 1, \ldots, m\), and (C1) imply that
\[
C||w^*||^2 \leq \sum_{i=1}^{m} \sum_{y \in \bar{Y}(\alpha)} a_{i,y} \Delta(y(i), y') - \frac{1}{2} m \sum_{i=1}^{m} \xi_i^* \\
\leq \sum_{i=1}^{m} \sum_{y \in \bar{Y}(\alpha)} a_{i,y} D_i = \frac{1}{m} \sum_{i=1}^{m} D_i.
\]
Therefore, \(||w^*|| \leq \sqrt{\frac{1}{m} \sum_{i=1}^{m} D_i / C}.\)