Scalable Convex Multiple Sequence Alignment via Entropy-Regularized Dual Decomposition

Jiong Zhang[†] **Ian E.H. Yen**[‡] **Pradeep Ravikumar**[‡] **Inderjit S. Dhillon**[†]

† University of Texas at Austin ‡ Carnegie Mellon University

Abstract

Multiple Sequence Alignment (MSA) is one of the fundamental tasks in biological sequence analysis that underlies applications such as phylogenetic trees, profiles, and structure prediction. The task, however, is NP-hard, and the current practice resorts to heuristic and local-search methods. Recently, a convex optimization approach for MSA was proposed based on the concept of atomic norm [23], which demonstrates significant improvement over existing methods in the quality of alignments. However, the convex program is challenging to solve due to the constraint given by the intersection of two atomic-norm balls, for which the existing algorithm can only handle sequences of length up to 50, with an iteration complexity subject to constants of unknown relation to the natural parameters of MSA. In this work, we propose an *accelerated dual decomposition* algorithm that exploits *entropy regularization* to induce closed-form solutions for each atomic-norm-constrained subproblem, giving a single-loop algorithm of iteration complexity linear to the problem size (total length of all sequences). The proposed algorithm gives significantly better alignments than existing methods on sequences of length up to hundreds, where the existing convex programming method fails to converge in one day.

1 Introduction

The *Multiple Sequence Alignment (MSA)* problem considers finding the alignments between a collection of sequences, which serves a fundamental tools for many applications in Bioinformatics such as Phylogenetic analysis and Protein Structure prediction [14]. There are several formulation for the problem, including the *Sum-of-Pair* objective, *Star-Alignment* objective and some other Phylogency-based score. However, all of these problems are known to be NP-hard [4].

Despite the NP-hardness of MSA, there have been numerous attempts for a tractable MSA algorithm. In particular, a direct extension of dynamic-programming-based pairwise alignment algorithm to MSA suffers from a time complexity exponential to the number of sequences. And traditional global optimization techniques for MSA only works on small problems [5, 7, 15, 16]. The mostly used algorithms in current practice are based on heuristics. For example, the *progressive algorithm* [8] greedily aligns two sequences at a time according to a guided tree, and constructs the final alignment sequentially in an order defined by the tree. Some other approaches, such as Expectation Maxmimization (EM) algorithm [2], fits probabilistic models (i.e. Profile-Hidden Markov Model) to the sequences through a local search. However, it has been observed that the EM approach could barely outperform progressive algorithms [6, 13].

Following a recent thread of convex relaxation approach to exemplar-based unsupervised learning [10, 24, 25], [23] proposed a convex optimization method for MSA via atomic norm constraints [1]. The convex optimization approach shows a significantly better quality of alignments than traditional approaches. However, since they expose the domain of the convex program as the interaction of two atomic-norm constraints with exponentially large number of atoms, it can hardly be tackled by traditional optimization methods. In [23], a method called *Greedy Direction Method of Multiplier (GDMM)* was proposed to solve the atomic-norm-constrained program. Although the algorithm has guaranteed convergence to optimal solution, their experiments only demonstrate results on sequences of length and number no more than 50, and in our experiment we found it can hardly scale to longer

Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54. Copyright 2017 by the author(s).

sequences. Another issue of the algorithm is that its iteration complexity contains several constants that can be hardly estimated from the natural parameters of an MSA problem, so it is unclear whether in practice the algorithm can produce a reasonable solution in finite time.

In this paper, we propose a new algorithm for solving the convex atomic-norm formulation of MSA — *Entropy-Regularized Dual Decomposition (ERDD)* . We show that an entropy regularized problem with single atomic-norm constraint yields a closed-form solution that can be efficiently computed via a dynamic programming algorithm, which results in a single-loop algorithm of $O(NL\sqrt{|\Sigma|\log|\Sigma|/\epsilon})$ iteration complexity to find ϵ-suboptimal alignments for N sequences of target length L and an alphabet size $|\Sigma|$. In our experiments, the proposed algorithm gives significantly better alignments than existing methods on both real and synthetic sequences of length up to hundreds, where the existing convex programming method fails to converge in one day.

2 Multiple Sequence Alignment

A sequence is defined as a string of characters from an alphabet Σ, augmented with start symbol $*$ and end symbol $\#$. Let $\hat{\Sigma} = \Sigma \cup \{*, \#\}$. Given two sequences $x_1 \in \hat{\Sigma}^{l_1}, x_2 \in \hat{\Sigma}^{l_2}$ of length l_1 and l_2, we define an alignment from x_2 to x_1 to be a path of state transitions. At each state $(i,j) \in S = [l_1] \times [l_2]$, we can take one of the three types of transition $T : S \mapsto S$

$$\begin{aligned} &T_M(i,j) = (i+1, j+1),\ i \le l_1 - 1, j \le l_2 - 1 \\ &T_I(i,j) = (i+1, j),\ i \le l_1 - 1 \\ &T_D(i,j) = (i, j+1),\ j \le l_2 - 1 \end{aligned}$$

known as matching, insertion and deletion. An alignment is a series of transitions that starts with initial state $(1,1)$ and ends with state (l_1, l_2), where where $x_1[1] = x_2[1] = *$ and $x_1[l_1] = x_2[l_2] = \#$. To evaluate an alignment, we assign each alignment $a = \{t_i\}_{i=1}^{l'}$ a score by:

$$D(a; x_1, x_2) = \sum_{t \in a} d(t; x_1, x_2).$$

where $d(t; x_1, x_2)$'s value is determined by whether T is a match, insert or delete transition:

$$d(t; x_1, x_2) = \begin{cases} d_I & , t \in \mathcal{T}_I \\ d_D & , t \in \mathcal{T}_D \\ d_M & , t \in \mathcal{T}_M, x_1[i+1] \ne x_2[j+1] \\ 0 & , otherwise. \end{cases} \tag{1}$$

Here d_I, d_D and d_M are penalties associated with insertion, deletion and mismatch respectively. The pairwise sequence alignment problem aims to find an alignment of minimal loss: $a^* = arg\min D(a; x_1, x_2)$. This can be solved via a dynamic programming algorithms, such as *Smith-Waterman* or *Needleman-Wunsch*, in $O(l_1 l_2)$ time.

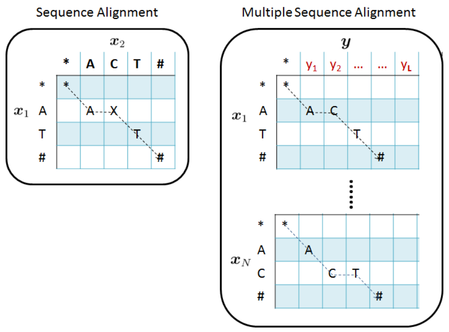


Figure 1: State transition paths for Pairwise and Multiple Sequence Alignment.

For the problem of aligning multiple sequences, things get complicated. There are a number of ways to define an MSA problem. In this work, we consider the formulation known as *MSA with Consensus* or *Star Alignment*. Given a set of N sequences $\mathcal{D} = \{x_n\}_{n=1}^N$, each sequence is of length l_n and define $l = \sum_{n=1}^N l_n$ to be the total length. In our setting, we aim to find a single consensus sequence z^*, together with its alignments to the data sequences $a^* = \{a_n^*\}_{n=1}^N$ such that the sum of alignment losses is minimized.

$$(z^*, a^*) = arg\min_{z,a} \sum_{n=1}^N D(a_n; x_n, z) \tag{2}$$

This objective function is known as *Star Alignment* score, indicating that we are looking for the hidden centroid of a star, whose tips are our observations. Note another popular setting is the *Sum of Pairs (SP)* objective, which looks at the sum of pairwise alignment costs between data sequences. Both the Star and SP alignment are proved to be NP-hard [4].

3 Convex Relaxation

In this section we introduce the atomic-norm based convex relaxation of (2) proposed in [23]. Recall that we let every sequence starts and ends with special symbols $*$ and $\#$ respectively. Thus we define the set of all sequences with length limit L as:

$$\mathcal{Z}_L = \left\{ z \in \hat{\Sigma}^l \;\middle|\; \begin{array}{c} l \in [L] \\ z[1] = *, z[l] = \# \\ z[m] \in \Sigma, 2 \le m \le l-1 \end{array} \right\}$$

which specifies our search space of the consensus sequence z^*, as illustrated as an path from left to right on the top surface of the cube in Figure 2. Then we define the ambient space that can be used to characterize any alignment between two sequences. Suppose we are dealing with a data of N sequences and we want to find the consensus sequence with length bounded by L. For a data sequence x of length T, we define the ambient space through constructing a directed acyclic graph (DAG) $G = (V, E)$. We denote a vertex v by a tuple: $v = (t, l, d) \in [T] \times [L] \times \hat{\Sigma}$ where t, l indicate the reading position on t-th character of x and l-th character of z with $z[l] = d$. The set of vertexes V is then defined as:

$$V = \left\{ (t, l, d) \middle| t \in [T], l \in [L], d \in \hat{\Sigma} \right\}$$

as illustrated as a cube in Figure 2. Then we define the set of possible transitions $E = E_I \cup E_D \cup E_M \subset V \times V$, representing insertion (E_I), deletion (E_D) and matching (E_M) respectively. Figure 2 (left) gives the idea of the transition space and detailed definition can be found in appendix. We connect all the transitions in E with an weight of $\{0, 1\}$ and define the ambient space associated with sequence x_n as $\mathcal{M}_T = \{0, 1\}^{|E|}$. Recall that an alignment is a series of insertion, deletion and matching transitions that takes initial state $(1, 1)$ and ending state (T, l) with $l \leq L$. Then we can observe that if we have a path from$(1, 1, *)$ to $(T, l^*, \#)$:

$$(1, 1, *) \to (t_1, l_1, d_1) ... \to (t_i, l_i, d_i) \to ... \to (T, l^*, \#)$$

then this path can determine uniquely a consensus sequence with $z[l_i] = d_i$ as well as an alignment a between z and x. Let $\mathcal{S}_T$ be the set of all such paths, we define our alignment space $\mathcal{A}_T \subset \mathcal{M}_T$ by assigning 1 on transitions included in the path:

$$\mathcal{A}_T = \left\{ m \in \mathcal{M}_T \middle| \begin{array}{c} a \in \mathcal{S}_T, \\ m[e] = \mathbb{I}(e \in a), \forall e \in E \end{array} \right\}$$

as illustrated in Figure 2 as a path inside the cube. Now consider all data sequences $\{x_n\}_{n=1}^N$ with lengths $\{T_n\}_{n=1}^N$. The whole ambient space, and alignment space, is the span of all such spaces associated with $\{T_n\}_{n=1}^N$:

$$\mathcal{M} = \otimes_{n=1}^N \mathcal{M}_{T_n}; \qquad \mathcal{A} = \otimes_{n=1}^N \mathcal{A}_{T_n}$$

Now if given $W \in \mathcal{A} \subset \mathcal{M}$, we actually have N alignments respectively between x_n and z_n^* for all the x_n in data. Since problem (2) asks us to find a *single* consensus $z^* = z_1^* = z_2^* = ... = z_N^*$, we need to impose another constraint on W. To do this, we define the consensus space $\mathcal{P}$. Given any sequence $z \in \mathcal{Z}_L$, we define the subspace $P(z) \subset \bigcup_{i=1}^N E^{(i)}$ associated with z to be all the transitions consistent with sequence z:

$$P(z) = \left\{ (t_1, l_1, d_1) \to (t_2, l_2, d_2) \middle| \begin{array}{c} z[l_1] = d_1 \\ z[l_2] = d_2 \end{array} \right\}$$

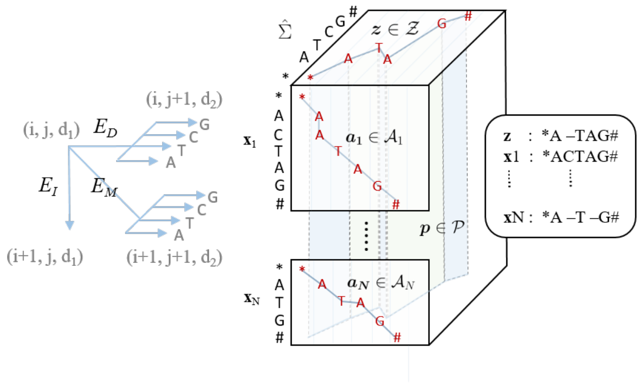


Figure 2: Transition space (left) and consensus sequence $z \in \mathcal{Z}_L$ (right) with atomic sets $\mathcal{P}$ and $\mathcal{A}$.

then the consensus space is defined as the union of subset of $\mathcal{M}$ supported on such subspaces:

$$\mathcal{P} = \bigcup_{z \in \mathcal{Z}_L} \{W \in \mathcal{M} | Supp(W) \subseteq P(z)\} \tag{3}$$

Now we can define the MSA problem as a minimization problem in the ambient space, with both alignment and atom constraints:

$$\begin{array}{ll} \min\limits_{W \in \mathcal{M}} & \langle D, W \rangle \\ s.t. & W \in \mathcal{A} \\ & W \in \mathcal{P}. \end{array} \tag{4}$$

where the penalty object $D \in \mathcal{M}$ is constructed by associating each transition with a penalty parameter, refer to appendix for detailed fefinition. It is easy to observe that given such two constrains, W can uniquely determine a consensus sequence z^* and its alignment with all data sequences from $\{x_n\}_{n=1}^N$, which is all we asked for in *Star-alignment* problem. Using this one can easily construct a convex relaxation of the MSA problem as:

$$\begin{array}{ll} \min\limits_{W \in Conv(\mathcal{M})} & \langle D, W \rangle \\ s.t. & W \in Conv(\mathcal{A}) \\ & W \in Conv(\mathcal{P}). \end{array} \tag{5}$$

where the notation $Conv(\mathcal{S})$ means the convex hull of atoms in set $\mathcal{S}$. Note the domain $Conv(\mathcal{A}) \cap Conv(\mathcal{P})$ in (5) gives a looser relaxation than the direction convex relaxation $Conv(\mathcal{A} \cap \mathcal{P})$. However, the relaxation given by the latter is as difficult as the NP-hard problem (4) since one can always find an integer solution at a corner of $Conv(\mathcal{A} \cap \mathcal{P})$, giving an integer solution $W \in \mathcal{A} \cap \mathcal{P}$ that contradicts the NP hardness of (4). On the other hand, (5) is tractable and permits quite efficient algorithm as we show in section 4.

4 Entropy Regularization and Accelerated Dual Decomposition

In this section, we propose an algorithm that efficiently solves (5) via a variant of Dual Decomposition method. In particular, we show how the technique of entropy-based dual smoothing [12] can elegantly simplifies an atomic-norm constrained problem into the problem of computing marginal probability of a transition under a distribution of atoms.

4.1 Dual Decomposition

Since handling two atomic-norm constraints at the same time is hard, one can use Dual Decomposition technique to decouple (5) into two subproblems, each involving only one constraint. By strong duality, we obtain the following equivalent formulation of (5):

$$\begin{aligned} \max_{Y \in \mathcal{M}^{\mathbb{R}}} \min_{W_1, W_2 \in \mathcal{M}^{\mathbb{R}}} \quad & \langle D, W_1 \rangle + \langle W_1 - W_2, Y \rangle \\ s.t. \quad & W_1 \in Conv(\mathcal{A}) \\ & W_2 \in Conv(\mathcal{P}). \end{aligned} \tag{6}$$

here $\mathcal{M}^{\mathbb{R}} := Conv(\mathcal{M})$. Let $I_{\mathcal{S}}(W)$ be the indicator function of set S that takes value 0 if $W \in \mathcal{S}$ and ∞ otherwise. Denoting

$$\mathcal{L}_1(W_1, Y) = \langle D + Y, W_1 \rangle + \sum_{n=1}^{N} I_{Conv(\mathcal{A}_{T_n})}(W_1^{(n)})$$

$$\mathcal{L}_2(W_2, Y) = -\langle Y, W_2 \rangle + I_{Conv(\mathcal{P})}(W_2),$$

the dual objective function

$$G(Y) := \min_{W_1, W_2} \mathcal{L}_1(W_1, Y) + \mathcal{L}_2(W_2, Y)$$

is non-smooth and hard to optimize for a solution of moderate precision. In [23], a standard *Augmented Lagrangian* technique is used to obtain a smooth dual objective. However, it results in *quadratic subproblems* that require iterative methods to solve. [23] proposed a single-loop algorithm—*Greedy Direction Method of Multiplier (GDMM)*—that alternates between one *non-drop step* of a Frank-Wolfe method in the primal and a gradient ascent step in the dual. They showed an $O(1/\epsilon)$-type iteration complexity for the algorithm, but only subject to constants (i.e. Pyramidal Widths of $Conv(\mathcal{A})$, $Conv(\mathcal{P})$ and Hoffman constant) of unknown relation to the natural parameters N, L and $|\Sigma|$.

4.2 Entropy Regularization & Acceleration

This section describes a more efficient variant of dual smoothing method via *entropy regularization*, which yields closed-form solutions for each primal subproblem of single atomic-norm constraint.

Algorithm 1 $W_1^{(n)*} = arg\min_{W_1^{(n)}} \mathcal{L}_1^{\mu}(W_1, Y)$.

```
Input: $Y^{(n)} \in \mathcal{M}_{T_n}$, Output: $W_1^{(n)} \in \mathcal{M}_{T_n}$
Initialize by ones $W^+, W^- \in \mathcal{M}_{T_n}, \Xi = 0$
for $e = v_1 \to v_2 \in E^{(n)}$ in BFS order do
    $W^+[e] = FlowIn(v_1, W^+) \times \exp(\frac{-D[e] - Y^{(n)}[e]}{\mu})$
end for
$\Xi = \sum_{e: e \text{ transits to an END state}} W^+[e]$.
Reverse all edges in graph.
for $e = v_2 \to v_1 \in E^{(n)}$ in BFS order do
    $W^-[e] = FlowIn(v_2, W^-) \times \exp(\frac{-D[e] - Y^{(n)}[e]}{\mu})$
end for
for $e = v_1 \to v_2 \in E^{(n)}$ in any order do
    $W_1^{(n)}$[e]= $FlowIn(v_2, W^-) \times W^+[e])/\Xi$.
end for
( $FlowIn(v, W) := \sum_{e: v' \to v \in E} W[e]$ )
```

Subproblem 1. The minimization of $\mathcal{L}_1(W_1, Y)$ w.r.t. W_1 can be separated into n independent subproblems:

$$\begin{aligned} \min_{W_1^{(n)}} \quad & \langle D^{(n)}, W_1^{(n)} \rangle \\ s.t. \quad & W_1^{(n)} \in Conv(\mathcal{A}_n) \end{aligned} \tag{7}$$

for $n \in [N]$, where $\mathcal{A}_n := \mathcal{A}_{T_n}$. Since $W_1^{(n)} \in Conv(\mathcal{A}_n)$, we can express $W_1^{(n)}$ as

$$W_1^{(n)} = \sum_{a_j \in \mathcal{A}_n} c_j a_j$$

where $c_j \geq 0$, $\sum_j c_j = 1$. Let $d_j^{(n)} = \langle D^{(n)}, a_j \rangle$ and $y_j^{(n)} = \langle Y^{(n)}, a_j \rangle$. We can augment (7) with a negative entropy term $\mu \sum_{a_j \in \mathcal{A}_n} c_j \log c_j$ to obtain the following augmented version of (7):

$$\begin{aligned} \min_{c_j : a_j \in \mathcal{A}_n} \quad & \sum_{j : a_j \in \mathcal{A}_n} (d_j^{(n)} + y_j^{(n)}) c_j + \mu \, c_j \log(c_j) \\ s.t. \quad & c_j \geq 0, \sum_{j : a_j \in \mathcal{A}_n} c_j = 1 \end{aligned} \tag{8}$$

Since the negative entropy is strongly convex with parameter μ, the augmented objective (8) ensures the smoothness of its convex conjugate with parameter $1/\mu$. Intuitively, the entropy regularization enforces the solution of each subproblem (7) to be smoothed as a distribution over all possible alignments $a_j \in \mathcal{A}_n$ instead of a single best alignment $a^* \in \mathcal{A}_n$. This speeds up the communication among sequences in order to reaching a consensus. By the KKT condition of (8), one can derive the following closed-form solution

Algorithm 2 $W_2^* := arg\min_{W_2}\ \mathcal{L}_2^{\mu}(W_2, Y)$.

```
**Input**: $Y \in \mathcal{M}$, **Output**: $W_2 \in \mathcal{M}$
Initialize $p^+, p^- \in [L] \times \hat{\Sigma} \times \hat{\Sigma}$ by zeros.
Compute edge weights $y \in [L] \times \hat{\Sigma} \times \hat{\Sigma}$ through (17)
Compute $p^+$, $p^-$ from $y$ via (19), (20) respectively.
Compute $\Xi$ using (21) or (22)
**for** $i \in [N]$, $e : (t_1, l_1, d_1) \to (t_2, l_2, d_2) \in E^{(i)}$ **do**
    $W_2^{(i)}[e] = \begin{cases} 0 & \text{, if } Y^{(i)}[e] \le 0 \\ \frac{p^+(l_2,d_1,d_2) \times p^-(l_2,d_1,d_2)}{y(l_2,d_1,d_2) \times \Xi} & \text{,o.w.} \end{cases}$
**end for**
```

$$c_j^* = \frac{\exp(\frac{-d_j - y_j}{\mu})}{\sum\limits_{a_j \in \mathcal{A}} \exp(\frac{-d_j - y_j}{\mu})}. \tag{9}$$

to be a distribution over $a_j \in \mathcal{A}_n$. Then for each transition $e \in E$, one can compute $W_1^{(n)}[e]$ as:

$$W_1^{(n)}[e] = \sum_{a_j \in \mathcal{A}_n : a_j[e]=1} c_j^*, \tag{10}$$

that is, the marginal probability that the alignment passes transition e. While the summation in (9), (10) involve exponentially large number of terms, the marginal probabilities can be computed efficiently using dynamic programming technique similar to *Forward-Backward algorithm* [18] in $O(|\hat{\Sigma}|LT_n)$ time, as we detail in Algorithm 1.

Subproblem 2. Similarly, for $W_2 \in Conv(\mathcal{P})$, it can be expressed as

$$W_2 = \sum_{j : p_j \in \mathcal{P}} \delta_j p_j, \tag{11}$$

and since for any $z_k \in \mathcal{Z}_L$ there is a $p_j^*(z_k)$ with $Supp(p_j) \subseteq P(z_k)$ that minimizes $\langle -Y, p_j \rangle$ by setting

$$p_j(z_k)[e]^* = \mathbb{I}(Y[e] > 0)\mathbb{I}(e \in P(z_k)) \tag{12}$$

where $\mathbb{I}(\cdot) \in \{0, 1\}$ is the indicator function. The minimization w.r.t. W_2 can be expressed as

$$\begin{aligned} \min_{b_k \ge 0} \quad & \sum_{k : z_k \in \mathcal{Z}_L} -\hat{y}_k b_k \\ s.t. \quad & \sum_{k : z_k \in \mathcal{Z}_L} b_k = 1 \end{aligned} \tag{13}$$

where $\hat{y}_k = \langle Y, p_j^*(z_k) \rangle$. The augmented version of (13):

$$\begin{aligned} \min_{b_k \ge 0} \quad & \sum_{k : z_k \in \mathcal{Z}_L} -\hat{y}_k b_k + \mu b_k \log b_k \\ s.t. \quad & \sum_{k : z_k \in \mathcal{Z}_L} b_k = 1 \end{aligned} \tag{14}$$

Algorithm 3 ConvexMSA-ERDD

```
**Input**: Data $\{x_n\}_{n=1}^N$, $L$ (upper bound on sequence length), $\mu$ and constant $M := 3NL|\Sigma|/(2\mu)$.
**Output**: $W_1, W_2 \in \mathcal{M}$
Initialize $\theta_0 = 1$ and $Y$, $Z \in \mathcal{M}$ with zeros.
Construct penalty tensor $D \in \mathcal{M}$
**for** $t = 1, 2, .....$ **do**
    $Y := (1 - \theta_t)Z + \theta_t Y$
    $W_1^{(i)} := arg\min_{W_1^{(i)}} \mathcal{L}_1^{\mu}(W_1^{(i)}, Y^{(i)})$, for $i \in [N]$.
    $W_2 := arg\min_{W_2} \mathcal{L}_2^{\mu}(W_2, Y)$
    $Y := Y - (W_1 - W_2)/(M\theta_t)$
    $Z := (1 - \theta_t)Z + \theta_t Y$
    $\theta_{t+1} = \frac{\sqrt{\theta_t^4 + 4\theta_t^2} - \theta_t^2}{2}$
**end for**
```

has the closed-form solution

$$b_k^* = \frac{\exp(\frac{\hat{y}_k}{\mu})}{\sum\limits_{k : z_k \in \mathcal{Z}_L} \exp(\frac{\hat{y}_k}{\mu})}. \tag{15}$$

by examining the KKT condition. Combining (15) with (12), we obtain

$$W_2^*[e] = \mathbb{I}(Y[e] > 0) \sum_{k : e \in P(z_k)} b_k^*. \tag{16}$$

Although it involves a summation over exponentially many terms, we can use a variant of the forward-backward algorithm to compute the marginal (probability) weight in $O(|\Sigma|^2 L + |\Sigma|LT)$ time, where $T = \sum_{n=1}^N T_n$. In particular, define the score on each transition $y \in [L] \times \hat{\Sigma} \times \hat{\Sigma}$ as

$$y(l, d_1, d_2) = \sum_{e \in H(l, d_1, d_2)} \frac{\mathbb{I}(Y[e] > 0)}{\mu} \tag{17}$$

where $H(l, d_1, d_2) = \{(\cdot, l, d_1) \to (\cdot, \cdot, d_2) \in E\}$. Then we can use this score in the forward-backward algorithm to evaluate (16) for each transition $e = (t_1, l_1, d_1) \to (t_2, l_2, d_2)$ as

$$W_2[e] = \frac{p^+(l_2, d_1, d_2) \times p^-(l_2, d_1, d_2)}{y(l_2, d_1, d_2) \times \Xi} \mathbb{I}(Y[e] > 0) \tag{18}$$

where $p^+, p^- \in [L] \times \hat{\Sigma} \times \hat{\Sigma}$ are recursively defined as:

$$p^+(l, d_1, d_2) = \begin{cases} e^{y(l,d_1,d_2)} \sum\limits_{r \in \Sigma} p(l-1, r, d_1) & , l > 1 \\ e^{y(l,d_1,d_2)} \mathbb{I}(d_1 = *) & , l = 1 \end{cases} \tag{19}$$

$$p^-(l, d_1, d_2) = \begin{cases} e^{y(l,d_1,d_2)} \sum\limits_{r \in \Sigma} p(l+1, d_2, r) & , l < L \\ e^{y(l,d_1,d_2)} \mathbb{I}(d_2 = \#) & , l = L \end{cases} \tag{20}$$

and Ξ is the normalizer, which can be computed by either of the followings:

$$\Xi = \sum_{l\in[L-1]} \sum_{d_1\in\hat{\Sigma}} p^{+}(l, d_1, \#) \tag{21}$$

$$\Xi = \sum_{d_2\in\hat{\Sigma}} p^{-}(1, *, d_2). \tag{22}$$

Algorithm 2 sketches the overall procedure.

Accelerated Gradient Method. Let $\mathcal{L}_1^{\mu}(c, Y)$ and $\mathcal{L}_2^{\mu}(b, Y)$ denote the entropy-regularized objectives (8) and (14) respectively. The dual objective can be derived explicitly as

$$\begin{aligned} G_\mu(Y) &:= \min_{W_1} \mathcal{L}_1^{\mu}(W_1, Y) + \min_{W_2} \mathcal{L}_2^{\mu}(W_2, Y) \\ &= \mu \log\Big(\sum_{j:a_j\in\mathcal{A}} \exp\big(\frac{-d_j - y_j}{\mu}\big)\Big) + \mu\log\Big(\sum_{k:z_k\in\mathcal{Z}_L} \exp\big(\frac{\hat{y}_k}{\mu}\big)\Big) \end{aligned}$$

where $d_j = \langle D, a_j\rangle$, $y_j = \langle Y, a_j\rangle$ and $\hat{y}_k = \langle Y, p_j^*(z_k)\rangle$. By Danskin's theorem, we have

$$\nabla_Y G_\mu(Y) = W_1^*(Y) - W_2^*(Y). \tag{23}$$

Notice that $G_\mu(Y)$ is the sum of two partition functions, so its second derivative takes the form of [22, 26]

$$\nabla_Y^2 G_\mu(Y) = \frac{1}{\mu}\left(Cov_{\mathcal{A}}[a_j] + Cov_{\mathcal{Z}_L}[p_j^*(z_k)]\right) \tag{24}$$

where $Cov_{\mathcal{A}}[.]$ and $Cov_{\mathcal{Z}_L}[.]$ denote the covariance matrices w.r.t. distributions defined by the potential functions $\exp(-(d_j + y_j)/\mu)$, $a_j \in \mathcal{A}$ and $\exp(\hat{y}_k/\mu)$, $z_k \in \mathcal{Z}_k$ respectively.

To apply the *Accelerated Gradient Method* [11, 21] to maximize the dual objective, we need the Lipschitz-continuous constant M of $\nabla_Y G_\mu(Y)$ (i.e. the maximum eigenvalue of $\nabla_Y^2 G_\mu(Y)$), which can be computed as the product of number of variables and an upper bound M_{diag} on its diagonal element:

$$\|\nabla^2 G_\mu(Y)\| \le \frac{NL(2|\Sigma|+1)}{\mu} M_{diag} \le \frac{3NL|\Sigma|}{2\mu} = M$$

where $M_{diag} = 1/2$ since each diagonal element is the sum of variances of two independent Bernoulli Random variables. Algorithm 3 summarizes the overall procedure. In the next section, we show that, by a careful selection of μ, Algorithm 3 enjoys an $O(1/\epsilon)$ iteration complexity with explicit constants to achieve ϵ suboptimality.

5 Convergence Analysis

The analysis has three steps. First, we bound the difference between the origianl objective $G(Y)$ and its augmented version $G_\mu(Y)$ as a function of μ. Then we show the iteration complexity of Algorithm 3 depends on the smoothness constant parametrized by μ. Finally, we show there is a choice of μ that gives the desired result.

Theorem 1 (Smooth Approximation)**.** *For any $Y \in \mathbb{R}^{|\mathcal{M}|}$, we have*

$$G_\mu(Y) \le G(Y) \le G_\mu(Y) + \mu\left(\log|\mathcal{Z}_L| + \sum_{n=1}^{N}\log|\mathcal{A}_n|\right)$$

where $\sum_{n=1}^{N}\log|\mathcal{A}_n| \le 2NL\log|\Sigma|$ and $\log|\mathcal{Z}_L| \le L\log|\Sigma|$.

Proof. The second inequality is true because

$$\begin{aligned} G_\mu(Y) &:= \min_{b,c} \mathcal{L}_1^{\mu}(c, Y) + \mathcal{L}_2^{\mu}(b, Y) \\ &\le \min_{b,c} \mathcal{L}_1(c, Y) + \mathcal{L}_2(b, Y) = G(Y), \end{aligned}$$

where the inequality is due the non-positivity of augmented term. The first inequality is true because

$$\begin{aligned} &G_\mu(Y) := \min_{b,c} \mathcal{L}_1^{\mu}(c, Y) + \mathcal{L}_2^{\mu}(b, Y) \\ &\ge \min_{b,c} \mathcal{L}_1(c, Y) + \mathcal{L}_2(b, Y) \\ &+ \min_{b,c} -\mu\sum_{n=1}^{N}\sum_{j:a_j\in\mathcal{A}} c_j \log\frac{1}{c_j} - \mu\sum_{k:z_k\in\mathcal{Z}_L} b_k\log\frac{1}{b_k} \\ &\ge G(Y) - \mu\Big(\sum_{n=1}^{N}\log|\mathcal{A}_n|\Big) - \mu\log|\mathcal{Z}_L|, \end{aligned}$$

where the last step is by Jensen's inequality. □

Theorem 2 (Convergence of Accelerated Gradient)**.** *Let $\bar{\mathcal{Y}}$ be the set of optimal solutions of*

$$G_\mu^* := \max_{Y\in\mathbb{R}^{|\mathcal{M}|}} G_\mu(Y).$$

Algorithm 3 has

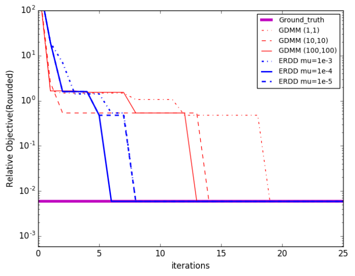
$$G_\mu^* - G_\mu(Y^t) \le \epsilon\ ,\ \textit{for}\ t \ge \sqrt{\frac{MR^2}{\epsilon}}, \tag{25}$$

where $R = \min_{\bar{Y}\in\bar{\mathcal{Y}}}\|\bar{Y}\|$.

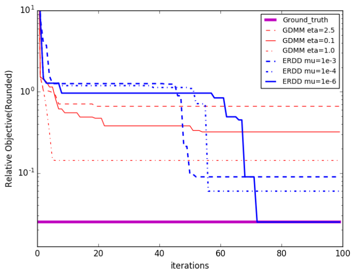
Proof. The proof of this result can be found in, for example, Corollery 1 of [21]. □

Corollary 1 (Iteration Complexity)**.** *Let G^* be the optimal objective value of problem* (5)*. By setting $\mu = \frac{\epsilon}{4NL\log|\Sigma|}$, Algorithm 3 has*

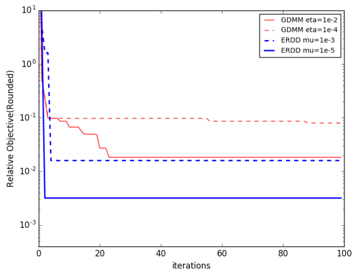
$$G^* - G(Y^{(t)}) \le \epsilon.\ ,\ \textit{for}\ t \ge \frac{NLR\sqrt{12|\Sigma|\log|\Sigma|}}{\epsilon}.$$



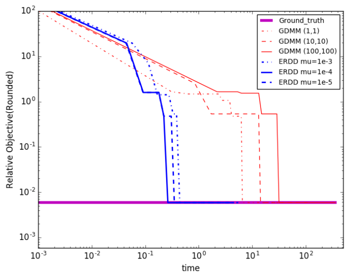
(a)



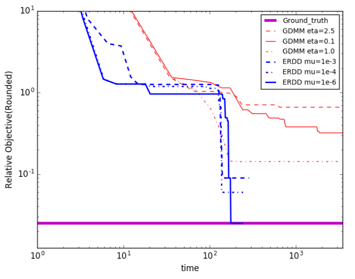
(b)



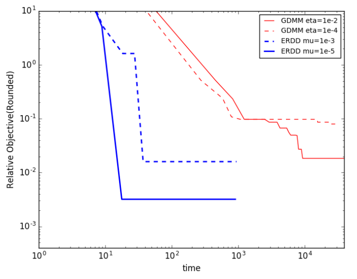
(c)



(d)



(e)



(f)

Figure 3: Iteration history of ConvexMSA-GDMM and ConvexMSA-ERDD on data sets of increasing scale (from left to right). y-axis is the relative objective $(f - f^*)/f^*$ achieved by the best (rounded) solution obtained from beginning up to the current iterate, where f^* is the best objective achieved among all methods. (a)(d) are iteration and time plots on Syn00 dataset ($N = 10$, $\bar{T} = 30$), (b)(e) are results on Syn04 dataset ($N = 50$, $\bar{T} = 100$), and (c)(f) are results on bicoid-3 dataset ($N = 15$, $\bar{T} = 551$).

Proof. With the choice of μ, we have

$$M = 3N^2L^2|\Sigma|\log|\Sigma|/(\epsilon/2).$$

By Theorem 2, for $t \geq \frac{NLR\sqrt{12|\Sigma|\log|\Sigma|}}{\epsilon}$, Algorithm 3 has

$$G_\mu(Y^{(t)}) \geq G^*_\mu - \epsilon/2 \geq G_\mu(Y^*) - \epsilon/2$$

for some dual optimal solution Y^* of (5). Then

$$G(Y^{(t)}) \geq G_\mu(Y^{(t)}) \geq G_\mu(Y^*) - \epsilon/2 \geq G(Y^*) - \epsilon$$

by Theorem 1 and the choice of μ. $\square$

Note the iteration complexity in Corollary 1 is linear to NL because the MSA objective (5) (alignment cost) is growing with NL. To achieve a $\hat{\epsilon}$ suboptimality in *average cost per character*, one would require the scaled objective $\frac{1}{NL}\langle D, W\rangle$ to reach $\hat{\epsilon}$ tolerance, giving an iteration complexity of $\frac{R\sqrt{12|\Sigma|\log|\Sigma|}}{\hat{\epsilon}}$.

6 Experiments

In the experiments we evaluate the compared algorithms through two measures of the alignment quality: *Sum-of-Pair*(SP) score and *Star Alignment*(Star) score. Given the consensus ancestor sequence z^* and each data sequence's alignment a^*_i with z^*, the Star score is computed exactly as our objective (2). The SP score is the sum of all pairwise penalties among data sequences over all aligned characters. In the evaluation, the three penalty parameters (d_I,d_D,d_M) are set to 1.

To produce synthetic datasets of evolutionary structure, a standard TKF1 evolutionary models are used [19, 20]. Given an ancestor sequence, the TKF1 model has each link between two characters independently evolve with insertion, deletion and substitution modeled by Poisson Process of rates α, β and γ respectively. Five synthetic datasets are generated and used in the experiments, in which number of observations N varies from 10 to 80, average sequence length $\bar{L}$ varies from 30 to 150 and mutation rates α, β, γ lie between 0.005 to 0.05. We also evaluate different methods on 5 real datasets with N up to 853 (HIV) and average length up to 550 (bicoid-3).

We compare our algorithm with the GDMM algorithm proposed in [23] that solves exactly the same objective function (5) as our method, as well as five widely used solvers in Bioinformatics community. Their short names are listed as follows:

Table 1: [SP score/Star score] achieved by MSA solvers on synthetic datasets. Information of data includes: number of sequences N, ancestor sequence length L and number of insertion, deletion and substitutions in total (I, D, M). The penalty connected with all three kinds of mutation is set to be 1. Numbers marked with * indicates that it failed to converge in 6 hours and the best-performing intermediate rounding result is reported.

	Syn00	**Syn01**	**Syn02**	**Syn03**	**Syn04**
	N=10, L=30	N=30, L=50	N=30, L=50	N=80, L=150	N=50, L=100
	(I=3, D=2, M=4)	(12, 11, 7)	(19, 18, 9)	(116, 131, 84)	(240, 255, 170)
ClustalOmega	311/47	3295 / 126	6671/274	115446/1725	62491/1749
Kalign	88/10	1440 / 51	2003/71	37484/486	57703/1389
T-coffee	99/12	1031 / 36	1492/53	29059/374	44784/1102
MAFFET	87/10	1196 / 42	1856/66	38062/497	41798/971
MUSCLE	87/10	1060 / 37	1649/59	32565/422	37441/856
ConvexMSA-GDMM	79/**9**	**863/30**	**1285/45**	29069/378*	28254/614*
ConvexMSA-ERDD	**78/9**	**863/30**	**1285/45**	**25702/330**	**27932/602**
Ground Truth	**78/9**	**863/30**	**1285/45**	**25702/330**	**27932/602**

Table 2: MSA algorithms tested on real datasets. Datasets are characterized by number of sequences N and average length $\bar{L}$. Parameter setting is same as experiments on synthetic datasets. Numbers marked with * indicates that it failed to converge in 24 hours and the best-performing intermediate rounding result is reported.

	copA	**antizyme**	**mir92**	**bicoid-3**	**HIV-FE**
	$(N, \bar{L})=(17, 90)$	$(13, 57)$	$(33, 79)$	$(15, 550)$	$(853, 51)$
ClustalOmega	2886/241	1041/123	17461/830	19667/1741	2339340/3662
Kalign	2611/208	834/94	16356/740	17025/1490	2332265/3654
T-coffee	2384/199	814/90	15520/713	16248/1394	N/A
MAFFET	2358/198	838/95	16020/736	17848/1582	N/A
MUSCLE	2363/199	854/98	15611/725	16448/1423	N/A
ConvexMSA-GDMM	2424/185*	838/95*	15684/652*	15566/1261*	2412452/**3625***
ConvexMSA-ERDD	**2340/180**	**782/85**	**15120/648**	**15536/1250**	**2322991/3625**

- CLUSTAL-OMEGA [17]: a popular progressive alignment algorithm.
- Kalign [9]: A MSA method based on Wu-Manber string-matching algorithm.
- MAFFT [7]: A progressive algorithm that uses heuristics based on FFT.
- MUSCLE [3]: A method based on iterative local refinements.
- T-COFFEE [15]: an algorithm generating pairwise alignment libraries to guide MSA.
- Convex-GDMM: the GDMM algorithm [23].
- Convex-ERDD: the proposed Algorithm 3.

For ConvexMSA-GDMM[23] inner iteration for sub-problem 1 is set between 1 and 20 and for sub-problem 2 is set to be 1. The initial stepsize is set to 0.15 for ConvexMSA-ERDD, and the coefficient of the entropy regularizer μ is chosen from the range $[0.0001, 0.001]$.

Figure 3 shows the iteration history of ConvexMSA-ERDD and GDMM, from which we can make several observations. First, despite the sensitivity that ERDD shows w.r.t. the choice of μ, it actually achieves overall better result than that of GDMM. Second, large μ makes convergence of ERDD faster but with less precision. GDMM algorithm generally costs several times more iterations for getting similar performance of ERDD.

From table 1 and 2, it can be observed that ConvexMSA-ERDD succeeded to discover the ground truth alignment solution in all synthetic datasets. Besides this, ConvexMSA-ERDD also has very good scalability, with respect to both N and L. Note that even though ConvexMSA-GDMM achieves similar precision in some of the small datasets, it fails to converge in reasonable time(24 hours)on large datasets such as bicoid-3 and HIV-FE.

Acknowledgements I.D. acknowledges the support of NSF via CCF-1320746, IIS-1546452, and CCF-1564000. P.R. acknowledges the support of ARO via W911NF-12-1-0390 and NSF via IIS-1149803, IIS-1320894, IIS-1447574, and DMS-1264033, and NIH via R01 GM117594-01 as part of the Joint DMS/NIGMS Initiative to Support Research at the Interface of the Biological and Mathematical Sciences.

References

[1] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky. The convex geometry of linear inverse problems. *Foundations of Computational mathematics*, 12(6):805–849, 2012.

[2] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[3] R. C. Edgar. Muscle: multiple sequence alignment with high accuracy and high throughput. *Nucleic acids research*, 32(5):1792–1797, 2004.

[4] I. Elias. *Settling the intractability of multiple alignment*. Springer, 2003.

[5] C. Gondro and B. Kinghorn. A simple genetic algorithm for multiple sequence alignment. *Genetics and Molecular Research*, 6(4):964–982, 2007.

[6] K. Karplus and B. Hu. Evaluation of protein multiple alignments by sam-t99 using the balibase multiple alignment test set. *Bioinformatics*, 17(8):713–720, 2001.

[7] K. Katoh, K.-i. Kuma, H. Toh, and T. Miyata. Mafft version 5: improvement in accuracy of multiple sequence alignment. *Nucleic acids research*, 33(2):511–518, 2005.

[8] M. A. Larkin, G. Blackshields, N. Brown, R. Chenna, P. A. McGettigan, H. McWilliam, F. Valentin, I. M. Wallace, A. Wilm, R. Lopez, et al. Clustal w and clustal x version 2.0. *Bioinformatics*, 23(21):2947–2948, 2007.

[9] T. Lassmann and E. L. Sonnhammer. Kalign–an accurate and fast multiple sequence alignment algorithm. *BMC bioinformatics*, 6(1):1, 2005.

[10] D. Malioutov, A. Kumar, and I. E. Yen. Large-scale submodular greedy exemplar selection with structured similarity matrices. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 507–516. AUAI Press, 2016.

[11] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet. Math. Dokl.*, 1983.

[12] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[13] C. Notredame. Recent progress in multiple sequence alignment: a survey. *Pharmacogenomics*, 3(1):131–144, 2002.

[14] C. Notredame. Recent evolutions of multiple sequence alignment algorithms. *PLoS Comput Biol*, 3(8):e123, 2007.

[15] C. Notredame, D. G. Higgins, and J. Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of molecular biology*, 302(1):205–217, 2000.

[16] M. F. Omar, R. A. Salam, N. A. Rashid, and R. Abdullah. Multiple sequence alignment using genetic algorithm and simulated annealing. In *Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on*, pages 455–456. IEEE, 2004.

[17] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, et al. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular systems biology*, 7(1):539, 2011.

[18] C. Sutton and A. McCallum. An introduction to conditional random fields. *arXiv preprint arXiv:1011.4088*, 2010.

[19] J. L. Thorne, H. Kishino, and J. Felsenstein. An evolutionary model for maximum likelihood alignment of dna sequences. *Journal of Molecular Evolution*, 33(2):114–124, 1991.

[20] J. L. Thorne, H. Kishino, and J. Felsenstein. Inching toward reality: an improved likelihood model of sequence evolution. *Journal of molecular evolution*, 34(1):3–16, 1992.

[21] P. Tseng. On accelerated proximal gradient methods for convex-concave optimization. *gradient methods for convex-concave optimization. SIAM Journal on Optimization*, 2008.

[22] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

[23] I. E. Yen, X. Lin, J. Zhang, P. Ravikumar, and I. S. Dhillon. A convex atomic-norm approach to multiple sequence alignment and motif discovery. In *ICML.*, 2016.

[24] I. E. Yen, D. Malioutov, and A. Kumar. Scalable exemplar clustering and facility location via augmented block coordinate descent with column generation. In *Proc. AISTATS*, 2016.

[25] I. E.-H. Yen, X. Lin, K. Zhong, P. Ravikumar, and I. S. Dhillon. A convex exemplar-based approach to mad-bayes dirichlet process mixture models. In *ICML*, pages 2418–2426, 2015.

[26] K. Zhong, I. E.-H. Yen, I. S. Dhillon, and P. K. Ravikumar. Proximal quasi-newton for computationally intensive l1-regularized m-estimators. In *Advances in Neural Information Processing Systems*, pages 2375–2383, 2014.

7 Appendix A: supplement definitions

In this section we give some detailed definitions of variables used in section 3. Recall that in the ambient space to characterize the alignment between any two sequences is defined through the graph $G = (V, E)$ where $V = [T] \times [L] \times \hat{\Sigma}$. The edges $E = E_I \cup E_D \cup E_M$ are defined through V. In particular:

$$E_I = \left\{ (t,l,d) \to (t+1,l,d) \,\middle|\, \begin{matrix} t \in [T-1], \\ l \in [L], \\ d \in \hat{\Sigma} \end{matrix} \right\}$$
$$E_D = \left\{ (t,l,d_1) \to (t,l+1,d_2) \,\middle|\, \begin{matrix} t \in [T], \\ l \in [L-1], \\ d_1, d_2 \in \hat{\Sigma} \end{matrix} \right\}$$
$$E_M = \left\{ (t,l,d_1) \to (t+1,l+1,d_2) \,\middle|\, \begin{matrix} t \in [T-1], \\ l \in [L-1], \\ d_1, d_2 \in \hat{\Sigma} \end{matrix} \right\}$$

These edges represent the action of insertion, deletion and matching while we are aligning two sequences. By assigning $\{0,1\}$ to each edge we are actually saying if we are taking this action in alignment process or not. Similarly, by assigning penalty weights to each edge we can add our preference to each action. Given a sequence $x_n \in \hat{\Sigma}_n^T$, the penalty associated with edge e is defined as:

$$d(e;x) = \begin{cases} d_I & , e \in E_I \\ d_D & , e \in E_D \\ d_M & , e \in E_M, x_n[t+1] \neq d_2 \\ 0 & , otherwise \end{cases} \tag{26}$$

Then we can define the penalty variable associated with sequence x_n $D_{x_n} \in \{0, d_I, d_D, d_M\}^{|E|}$, where $D_{x_n}[e] = d(e;x_n)$. Now consider all data sequences $\{x_n\}_{n=1}^N$, then the whole penalty variable is defined by simply stacking all D_{x_n} together in a new dimension.