PALM: Probabilistic Area Loss Minimization for Protein Sequence Alignment

Fan Ding[*1] **Nan Jiang**[*1] **Jianzhu Ma**[2] **Jian Peng**[3] **Jinbo Xu**[4] **Yexiang Xue**[1]

[1]Department of Computer Science, Purdue University, West Lafayette, Indiana, USA
[2]Institute for Artificial Intelligence, Peking University, Beijing, China
[3]Department of Computer Science, University of Illinois at Urbana-Champaign, Illinois, USA
[4]Toyota Technological Institute at Chicago, Illinois, USA

Abstract

Protein sequence alignment is a fundamental problem in computational structure biology and popular for protein 3D structural prediction and protein homology detection. Most of the developed programs for detecting protein sequence alignments are based upon the likelihood information of amino acids and are sensitive to alignment noises. We present a robust method PALM for modeling pairwise protein structure alignments, using the area distance to reduce the biological measurement noise. PALM generatively learn the alignment of two protein sequences with probabilistic area distance objective, which can denoise the meansurement errors and offsets from different biologists. During learning, we show that the optimization is computationally efficient by estimating the gradients via dynamically sampling alignments. Empirically, we show that PALM can generate sequence alignments with higher precision and recall, as well as smaller area distance than the competing methods especially for long protein sequences and remote homologies. This study implies for learning over large-scale protein sequence alignment problems, one could potentially give PALM a try.

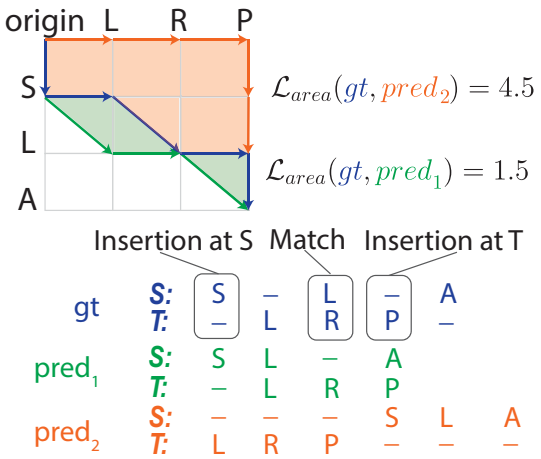

Figure 1: Illustration of protein sequence alignment and the area distance. **(Bottom)** The task is to align two amino acids sequences S and T, where one amino acid from one sequence can be aligned to either one amino acid from the other sequence (match), or to a gap (insertion, marked by -). **(Top)** Such an alignment become a path in the alignment matrix, where a diagonal transition represents a match, a horizontal or a vertical transition represents an insertion. The area between one predicted alignment and ground-truth is viewed as the area distance between them. Both of the two predicted alignments correctly predicted one edge on the ground truth alignment, yet the one with smaller area loss (i.e., "pred$_1$") is much closer to the ground-truth.

1 INTRODUCTION

Protein sequence alignment is a fundamental problem in computational structure biology and has been widely applied to protein sequence, structure and functional study [Do et al., 2006], including protein 3D structure prediction [Marks et al., 2011] and protein homology detection [Söding et al., 2005]. In the past two decades many computer programs have been developed for automatic pairwise sequence alignment [Altschul et al., 1990, Kumar et al., 2004, Hou et al., 2016] and multiple sequence alignment [Armougom et al., 2006, Katoh and Toh, 2008]. Here we only focus on pairwise sequence alignment, where we let S and T be two sequences of amino acids. Our goal is to align the two sequences. As shown in Figure 1(bottom), each amino acid in one sequence can be aligned to either an amino acid in the homology sequence (called a match) or a gap (called an insertion). Our tasks are that: (1) given a dataset of aligned pairs of amino acid sequences, learn the

[*]These authors contribute equally.

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

likelihoods of different alignments of the two sequences; (2) given a new amino acid sequence pair, determine the most likely alignment.

Most probabilistic alignment approaches maximize likelihood functions, which lead to the minimization of the pointwise differences of the two alignments. Nevertheless, this pointwise difference loss function is unrealistic for the biology application. The alignments contain noises and offsets as the measurements are done by different biologists Altschul and Lipman [1990], resulting in mismatches using pointwise loss. For example, Figure 1 presents two predicted alignments against the ground-truth. Both predicted alignments are different from the ground-truth in four locations. Therefore, they have identical pointwise differences against the ground-truth. Nonetheless, the green alignment is considerably better than the orange alignment, because the corresponding amino acids of each pair in the ground-truth are much closer to the green alignment. The missed predictions of the green alignment can be the result of biological measurement noises, while the orange alignment completely misses the ground-truth. Area loss can capture this since small offsets have small area distance.

In order to introduce the distance between two alignments into the probabilistic model, we propose **P**robabilistic **A**rea **L**oss **M**inimization (PALM) for pairwise protein sequence alignment, which is a two-step approach to model the whole alignment matrix. The key idea of PALM is (i) using a generative model $Pr(a|S,T)$ to model the ground-truth alignment a where the matching of each pair of amino acids in the two protein alignment leads to a change in the total likelihood; (ii) the observed alignment a^* is a noisy observation of the ground-truth alignment, the likelihood of observing which $Pr(a^*|a,S,T)$ negatively depends on the area difference of the two alignments. This area difference is capable of greatly penalizing those alignments a that are far from the observed alignment a^*. The learning goal of PALM is to maximize the marginal likelihood of the observed alignment $Pr(a^*|S,T)$, which sums over all the different ground-truth alignment a. However, the closed-form of maximal likelihood estimation of PALM is computationally intractable. We, therefore, propose a novel computing scheme that maximizes a lower bound of $\log Pr(a^*|S,T)$. To optimize the lower bounded objective, we formulate the gradient computation using contrastive divergence framework [Hinton, 2002], where dynamic sampling is used to sample alignments to estimate the gradient unbiasedly. We show theoretically that PALM can converge to the global optimum of the objective in a linear number of iterations.

Traditionally, dynamic programming with a deterministic cost function [Tompa, 2000, Durbin et al., 1998] were used to obtain the alignment, including NeedlemanWunsch method for global alignment [Needleman and Wunsch, 1970] and Smith-Waterman algorithm for local alignment [Smith and Waterman, 1981]. However, they are heavily dependent upon the proper design of the cost function between two amino acids. Recently probabilistic learning have been widely used to model sequences [Ma et al., 2014, Jeong and Kim, 2016, Daniels et al., 2015, Balakrishnan et al., 2011]. Söding [2005] employs HMM-HMM comparison for learning the alignment probability which models a protein family using HMM (Hidden Markov Model). MRFalign [Ma et al., 2014] further uses an MRF-MRF comparison which is more expressive than HMM model. However, they cannot model long-range residue interaction patterns and thus are not sensitive enough to the distant-related homologies. The idea of using area distance to model the gap between different sequences is also found in audio recognition [Lajugie et al., 2014, Gupta et al., 2019], which is under a different setting of ours.

Empirically, we show on the large dataset Protein Data Bank (PDB) [Wu and Xu, 2021] that our method has higher precision and recall value, as well as much smaller area distance than the competing methods, especially the lengths of the two sequences are far from each other. We find that the exact/4-offset/10-offset recall of PALM is twice more than that of dynamic programming (DP) when $|S| \in [1,100], |T| \in [400,+\infty)$ and the exact/4-offset/10-offset precision is also twice more than of DP when $|S| \in [400,+\infty), |T| \in [1,100]$. In addition, in terms of time efficiency between learning our method using dynamic sampling and the automatic differentiation method, PALM takes only one-fourth time to compute the gradient than the automatic differentiation over six different testing sets.

To summarize, our contributions are as follows:

- We propose PALM, a novel two-step approach for protein sequence alignment, where we first model the ground-truth alignment using a generative model, and then leverage a conditional distribution formed by the area difference of two alignments to denoise the observed alignment from noisy observation.
- We propose a novel computing scheme to maximize a tight lower bound of the computationally intractable log-likelihood, and efficiently compute the gradients by estimating it unbiasedly via dynamically sampling alignments. We show theoretically that PALM is able to converge to the global optimum of the objective in linear number of iterations.
- Experimentally we test our method on a large PDB dataset [Wu and Xu, 2021], and find that PALM outperforms the competing methods in either precision, recall, or area distance, especially on long protein sequences and remote homologies[1].

[1] Implementation: github.com/jiangnanhugo/PALM

2 PRELIMINARY

In this section, we briefly introduce the Markov Random Field (MRF), which is used as our probabilistic model. We also introduce the problem of pairwise sequence alignment.

2.1 MARKOV RANDOM FIELD

MRF is a generative model for the joint distribution of multiple correlated random variables. In an MRF, the probability $Pr(x)$ is defined as:

$$w(x) = \prod_{\alpha \in \mathcal{I}} \phi_\alpha(\{x\}_\alpha), \quad Pr(x) = w(x)/Z. \tag{1}$$

where $\{x\}_\alpha$ is a subset of variables in x that the function ϕ depends on. $\phi_\alpha : \{x\}_\alpha \rightarrow \mathbb{R}^+$ is a potential function, or commonly referred to as a clique. ϕ_α maps every assignment of variables in $\{x\}_\alpha$ to a non-negative real value. $\mathcal{I}$ is an index set and Z is a normalization constant, which ensures that the probability adds up to one: $Z = \sum_{x \in \{0,1\}^m} w(x)$. A potential function $\phi_\alpha(\{x\}_\alpha)$ defines the correlation between all variables in the subset $\{x\}_\alpha$. The structure of the MRF or the set $\mathcal{I}$ can be built from domain knowledge and potential functions can be learned from real-world data.

2.2 PAIRWISE SEQUENCE ALIGNMENT

Given a pair of sequence (S, T), we can formulate an alignment matrix of shape $(|S|, |T|)$ as shown in Figure 1. In the matrix, each row represents an amino acid in S and each column represents an amino acid in T. Each alignment for sequences S and T forms a path from the upper-left node to the bottom-right node, where each edge in the path is either horizontal, representing an insertion in T, vertical, representing an insertion in S, or diagonal, representing a match. We use symbols M, I_S and I_T to represent a match, an insertion in S, and an insertion in S, respectively. Thus an alignment a is a sequential combination of symbols M, I_S and I_T. Let $Pr_\theta(a|S, T)$ be the probability of alignment a with parameter θ. Our goal is to align the two given sequences. Our task can be divided into the following two parts.

Learning. Given a training set of $\{(S^{(k)}, T^{(k)}, a^{*(k)})\}_{k=1}^N$, where $S^{(k)}, T^{(k)}$ is a pair of sequences, and $a^{*(k)}$ is the ground-truth alignment between the two sequences. We want to learn the model via maximizing the likelihood, which translates to the following problem:

$$\max_\theta \prod_{k=1}^{N} Pr_\theta(a^{*(k)}|S^{(k)}, T^{(k)}). \tag{2}$$

where θ is the parameter of the model.

Inference. After learning $Pr_\theta(a|S, T)$, we can use the model to find the best alignment between two new sequences by solving the following problem:

$$\hat{a} = \arg\max_{a \in \mathcal{A}} Pr(a|S, T) \tag{3}$$

where the alignment $\hat{a}$ needs to form a consecutive path in the alignment matrix.

Nevertheless, most existing approaches that maximize likelihood functions will lead to the minimization of the pointwise differences of the two alignments, which is unrealistic for biology application. For instance, Figure 1 presents two predicted alignments against the ground-truth. Both alignments are different from the ground-truth in four locations. Therefore, they have equal pointwise differences against the ground-truth. However, the green alignment is considerably better than the orange alignment, because the corresponding amino acids of each pair in the ground-truth are much closer in this alignment. The missed predictions of the green alignment can be the result of biological measurement noises, while the orange alignment completely misses the ground-truth. Therefore, the green alignment should have a larger likelihood compared to the orange one given the ground-truth alignment.

3 PROBABILISTIC AREA LOSS MINIMIZATION

In this section, we first introduce our two-step model and then explain how it can solve the distant-related gap problem. Followed by illustration on how to efficiently learn this model via dynamic sampling. Finally, we detail the process of generating alignments in testing.

3.1 TWO-STEP MODEL

Define function π be the mapping from the index on the alignment to the indexes on both sequences, i.e., $\pi_S(a, k)$ is the index on sequence S for the k-th term of alignment a, and $\pi_T(a, k)$ is the index on sequence T for that term. $(\pi_S(a, k), \pi_T(a, k))$ are the coordinates of the k-th term of a in the alignment matrix. Using the ground-truth in Figure 1 as an example, $\pi_S(a, 1) = 1, \pi_T(a, 2) = 0$ meaning the top-down edge from origin. $\pi_S(a, 1) = 1, \pi_T(a, 2) = 1$ represents a vertical edge from the protein "S" in the alignment matrix.

Prior Prediction. Let $\mathcal{A} = \{a \mid a \text{ is a valid path}\}$ be the set of all valid paths. The validness ensures that alignment a has to start from the upper-left node and end at bottom-right node in the matrix. We model the probability of having the alignment a over sequences S and T as:

$$Pr_\theta(a|S, T) = \frac{e^{\sum_{k=1}^{|a|} \phi_\theta(\pi_S(a,k), \pi_T(a,k), a_k)}}{Z_\phi} \tag{4}$$

Algorithm 1: Probabilistic Area Loss Minimization for Protein Sequence Alignment.

Input: Model parameter θ, Maximum epochs Epo, Learning rate η, Weight λ, Training data $\mathcal{D}$.
Output: The converged model with parameter θ_{Epo}

1 **for** $t \leftarrow 1$ ***to*** Epo **do**
2 Randomly sample data $\{S, T, a^*\}$ from $\mathcal{D}$.
3 Inference $\hat{a} \leftarrow \arg\max_a \{\sum_{k=1}^{|a|} \phi(\pi_S(a,k), \pi_T(a,k), a_k) - \lambda \mathcal{L}_{area}(a^*, a)\}$. ▷ Solved by DP with matrix A
4 **for** $i \leftarrow |S|$ ***to*** 1, $j \leftarrow |T|$ ***to*** 1 **do**
5 Compute $Z(i,j)$. ▷ See Equation 11 and is used in Step 10
6 **end**
7 **for** $m \leftarrow 1$ ***to*** M **do**
8 $i = |S|, j = |T|$
9 **while** $i \geq 1$ ***and*** $j \geq 1$ **do**
10 Sample an edge a_k^m *w.r.t* $Pr_\theta(a_k | S_i, T_j)$ and update i, j. ▷ See Equation 12
11 Compute gradient of the neural network $\nabla \phi_\theta(\pi_S(a^m, k), \pi_T(a^m, k), a_k^m)$.
12 **end**
13 **end**
14 Estimate $\nabla \log Z_\phi \leftarrow \frac{1}{M} \sum_{a^m \sim Pr_\theta(a|S,T))} \left[\sum_{k'=1}^{|a^m|} \nabla \phi_\theta(\pi_S(a,k'), \pi_T(a,k'), a_{k'}) \right]$.
15 Estimate $\nabla \mathcal{L}_{LB} \leftarrow \sum_{k=1}^{|\hat{a}|} \nabla \phi_\theta(\pi_S(\hat{a},k), \pi_T(\hat{a},k), \hat{a}_k) - \nabla \log Z_\phi$.
16 $\theta_{t+1} \leftarrow \theta_t + \eta \nabla \mathcal{L}_{LB}$. ▷ Gradient update
17 **end**
18 **return** θ_{Epo}

where $Z_\phi = \sum_{a \in \mathcal{A}} e^{\sum_{k=1}^{|a|} \phi_\theta(\pi_S(a,k), \pi_T(a,k), a_k)}$ is the normalization factor, $\phi_\theta(\pi_S(a,k), \pi_T(a,k), a_k)$ is the feature function which takes as input the features extracted from every two amino acids of S and T, and θ represent the model parameters. Notice that function ϕ_θ can be either a linear function, or a neural network of arbitrary architecture.

Denoising via Area Distance. Due to the measurement error of biological tools, the observed alignments a^* are usually noisy [Altschul and Lipman, 1990]. In view of this observation, we introduce a conditional probability distribution over a^* conditioned on the latent variable a to diminish the effect of noise. We leverage the area distance as a probabilistic measure where the predicted alignment a that is similar to the observed one a^* has higher probability value:

$$Pr_{area}(a^* | a, S, T) = \frac{e^{-\lambda \mathcal{L}_{area}(a^*, a)}}{Z_{area}} \tag{5}$$

where $Z_{area} = \sum_{a' \in \mathcal{A}} e^{-\lambda \mathcal{L}_{area}(a^*, a')}$ is the normalization term and λ is the weight. We compute the area distance as the gap of area between the two alignments in the alignment matrix, which penalizes those predicted alignment that is far away from the observed alignment. For instance, as in Figure 1, the area distance between the ground-truth and the first predicted alignment (green line) is 1.5, and is 4.5 between the ground-truth and the second one (orange line). Then, we maximize the likelihood of the observed alignment, and the whole model becomes

$$Pr(a^* | S, T) = \sum_a Pr_{area}(a^* | a, S, T) Pr_\theta(a | S, T). \tag{6}$$

which sums over the latent variable a and is the marginal distribution of a^*.

To conclude, when the ground-truth become noisy, the point-wise MLE learning objective base on Equation (4) would oscillate and become harder to converge. The denoised objective in Equation (6) would make training be more steady.

3.2 MODEL LEARNING

To learn the model $Pr(a^* | S, T)$, we would like to maximize the log-likelihood of the observed alignment given sequence S and T:

$$\max_\theta \mathcal{L} = \max_\theta \log Pr(a^* | S, T) \tag{7}$$

Combining with our model definition in Equation 4 and 5, the log likelihood $\mathcal{L}$ can be rewritten as:

$$\begin{aligned} \mathcal{L} &= \log \sum_a \frac{e^{-\lambda \mathcal{L}_{area}(a^*, a)}}{Z_{area}} \frac{e^{\sum_{k=1}^{|a|} \phi_\theta(\pi_S(a,k), \pi_T(a,k), a_k)}}{Z_\phi} \\ &= \log \sum_a e^{\sum_{k=1}^{|a|} \phi_\theta(\pi_S(a,k), \pi_T(a,k), a_k) - \lambda \mathcal{L}_{area}(a^*, a)} \\ &\quad - \log Z_\phi Z_{area} \end{aligned}$$

The evaluation of $\mathcal{L}$ needs to sum over all possible alignments, which is computationally intractable. Since the sum is usually dominated by one alignment that has the maximum likelihood, we optimize the lower bound of $\mathcal{L}$ instead of directly optimize $\mathcal{L}$. Denote the lower bound as $\mathcal{L}_{LB}$, which is

$$\mathcal{L}_{LB} = \max_{a}\{\sum_{k=1}^{|a|}\phi_\theta(\pi_S(a,k),\pi_T(a,k),a_k) - \lambda\mathcal{L}_{area}(a^*,a)\} - \log Z_{area} - \log Z_\phi \quad (8)$$

It is obvious that $\mathcal{L}_{LB} \leq \mathcal{L}$ because of the principle of log-sum-exp function: $\max_x \phi(x) \leq \log\sum_x e^{\phi(x)} \leq \max_x \phi(x) + \log N$, where N represents the number of all possible x. Then, the learning procedure is separated into two steps, where the first step is to inference $\hat{a}$ that has the maximum likelihood:

$$\hat{a} = \max_{a}\{\sum_{k=1}^{|a|}\phi_\theta(\pi_S(a,k),\pi_T(a,k),a_k) - \lambda\mathcal{L}_{area}(a^*,a)\}$$

One can identify that $\hat{a}$ represent those alignments that are close to the observed alignment a^*. Since the observed alignments are generated by biological tools and contains some noisy observation, our method use a bunch of alignments that are close to the observed alignments to reduce the impact of noise. The second step is to optimize parameter θ in order to maximize $\mathcal{L}_{LB}$ using both a^* and $\hat{a}$. By optimizing the parameter θ, we can keep increasing the likelihood of $\hat{a}$, making the lower bound to approach the true likelihood $\mathcal{L}$.

Algorithm 1 shows the training procedure of PALM. In the following parts, we will show how to inference $\hat{a}$ via dynamic programming, how to optimize θ via dynamic sampling, and finally give a convergence analysis of our learning algorithm.

3.2.1 Inference via Dynamic Programming

Based on the observation that our area loss $\mathcal{L}_{area}$ is decomposable, we propose a dynamic programming approach to inference $\hat{a}$. Specifically, we decompose the area loss into the sum of area-unit distance $\mathcal{L}_{area}(a^*,\hat{a}) = \sum_{k'=1}^{|\hat{a}|}\mathcal{L}^{k'}_{area}(a^*,\hat{a})$ where k' is the index of the predicted alignment $\hat{a}$. Given the observed alignment a^*, we compute $\mathcal{L}^{k'}_{area}$ as follows: we first find it's corresponding coordinates on sequences S and T is $\pi_S(\hat{a},k'), \pi_T(\hat{a},k')$ and then find a index k in a such that $\pi_T(a,k) = \pi_T(\hat{a},k')$. Then $\mathcal{L}^{k'}_{area}$ is defined as:

$$\mathcal{L}^{k'}_{area}(a^*,\hat{a}) = \begin{cases} |\pi_S(a^*,k)\text{-}\pi_S(\hat{a},k')+\frac{1}{2}| & a^*_k{=}M, \hat{a}_{k'}{=}I_T \\ |\pi_S(a^*,k)\text{-}\pi_S(\hat{a},k')| & a^*_k = M, \hat{a}_{k'}{=}M \\ |\pi_S(a^*,k)\text{-}\pi_S(\hat{a},k')\text{-}\frac{1}{2}| & a^*_k{=}I_T, \hat{a}_{k'}{=}M \\ |\pi_S(a^*,k)\text{-}\pi_S(\hat{a},k')| & a^*_k{=}I_T, \hat{a}_{k'}{=}I_T \\ 0 & \text{otherwise} \end{cases}$$

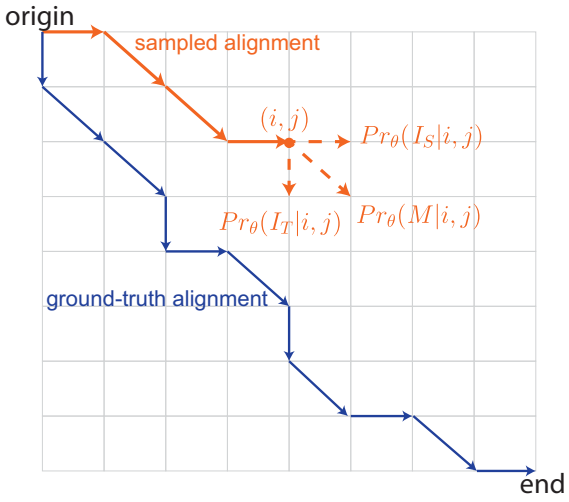

Figure 2: Sampling a path from the original until reaching the bottom-right corner in the alignment matrix. At point (i,j), the sampling approach first calculates the probability of taking the options M, I_S, I_T at this point, and then sample one option according to the probability value.

Since we can decompose $\mathcal{L}_{area}$, $\hat{a}$ can be obtained by the following dynamic programming approach. Let $A(i,j)$ represent the maximum likelihood of the path from node (i,j) to the bottom right corner, in the alignment matrix of sequence S and sequence T. Then we have:

$$A(i,j) = \max\begin{cases} A(i{+}1,j{+}1) + \phi_\theta(S_i,T_j,M) + \lambda\mathcal{L}^{k'}_{area}(a^*,\hat{a}) \\ A(i+1,j) + \phi_\theta(S_i,T_j,I_S) + \lambda\mathcal{L}^{k'}_{area}(a^*,\hat{a}) \\ A(i,j+1) + \phi_\theta(S_i,T_j,I_T) + \lambda\mathcal{L}^{k'}_{area}(a^*,\hat{a}) \end{cases}$$

where $j = \pi_T(\hat{a},k')$ and we initialize $A(|S|,|T|) = 0$. The computation is line by line from the bottom right corner to the up left corner in the alignment matrix. The corresponding alignment $\hat{a}$ can be extracted from the matrix A by following the path that gives the largest likelihood.

3.2.2 Optimization via Dynamic Sampling

Once $\hat{a}$ is obtained according to the area distance, we optimize the lower bound $\mathcal{L}_{LB}$ using stochastic gradient descent. The gradient of $\mathcal{L}_{LB}$ can be written as:

$$\nabla\mathcal{L}_{LB} = \sum_{k=1}^{|\hat{a}|}\nabla\phi_\theta(\pi_S(\hat{a},k),\pi_T(\hat{a},k),\hat{a}_k) - \nabla\log Z_\phi \quad (9)$$

The term $\nabla\phi_\theta(\pi_S(\hat{a},k),\pi_T(\hat{a},k),\hat{a}_k)$ is the gradient of function ϕ_θ, which can be directly computed. Overall, there are $|\hat{a}|$ number of gradient terms with respect to this function. $\log Z_{area}$ term does not contain parameter to optimize, so it does not has the corresponding gradient term in $\nabla\mathcal{L}_{LB}$. For computing $\nabla\log Z_\phi$, we follow the idea of contrastive

divergence [Hinton, 2002] that formulate it as an expectation over probability $P(a|S,T)$:

$$\begin{aligned}\nabla \log Z_\phi &= \frac{1}{Z_\phi}\sum_{a\in\mathcal{A}} \nabla e^{\sum_{k'=1}^{|a|}\phi_\theta(\pi_S(a,k'),\pi_T(a,k'),a_{k'})} \\ &= \mathbf{E}_{a\sim Pr_\theta(a|S,T)}\left[\sum_{k'=1}^{|a|}\nabla\phi_\theta(\pi_S(a,k'),\pi_T(a,k'),a_{k'})\right] \\ &\approx \frac{1}{M}\sum_{a^m\sim Pr_\theta(a|S,T)}\left[\sum_{k'=1}^{|a^m|}\nabla\phi_\theta(\pi_S(a^m,k'),\pi_T(a^m,k'),a^m_{k'})\right]\end{aligned}$$

Therefore, we can approximate the exact gradient $\nabla \log Z_\phi$ unbiasedly by first sampling M paths from distribution $Pr_\theta(a|S,T)$ and then sum the gradients $\nabla\phi_\theta$ of all sampled path $\{a^m\}_{m=1}^M$. Note that the sum-product algorithm offers a general computing paradigm for the factor graph and $\log Z_\phi$ [Pebarz, 2015]. Our computing steps can be a particular case for the general sum-product algorithm.

Below we propose a backward-forward approach to sample one alignment under the probability distribution $Pr_\theta(a|S,T)$.

Backward Computing $Z(i,j)$. We denote $Z(i,j)$ as the sum of all the unnormalized energy values of every path starting from point (i,j) to the end $(|S|,|T|)$.

$$Z_\phi(i,j) = \sum_{a\in\mathcal{A}(i,j)} e^{\sum_{k=1}^{|a|}\phi_\theta(\pi_S(a,k),\pi_T(a,k),a_k)} \tag{10}$$

where $\mathcal{A}(i,j)$ is the set of all possible valid paths starting from node (i,j) to the end $(|S|,|T|)$. Then, we can backward compute $Z(i,j)$ via dynamic programming:

$$\begin{aligned}Z_\phi(i,j) =& Z_\phi(i+1,j+1)e^{\phi_\theta(i,j,M)} \\ &+ Z_\phi(i+1,j)e^{\phi_\theta(i,j,I_S)} \\ &+ Z_\phi(i,j+1)e^{\phi_\theta(i,j,I_T)}\end{aligned} \tag{11}$$

Forward Sampling Alignment. To sample a path a^m starting from the original point in the alignment matrix to the bottom right corner, we recursively sample the k-th edge of the alignment from $\{M, I_S, I_T\}$ at point (i,j) based on the probability distribution $Pr_\theta(a^m_k|S_i,T_j)$ correspondingly.

$$Pr_\theta(a^m_k|S_i,T_j)=\begin{cases}\frac{Z(i+1,j+1)}{Z(i,j)}e^{\phi_\theta(i,j,M)} & a^m_k{=}M \\ \frac{Z(i+1,j)}{Z(i,j)}e^{\phi_\theta(i,j,I_S)} & a^m_k{=}I_S \\ \frac{Z(i,j+1)}{Z(i,j)}e^{\phi_\theta(i,j,I_T)} & a^m_k{=}I_T\end{cases} \tag{12}$$

where $1\le i\le|S|$ and $1\le j\le|T|$. See Figure 2 as an example. We start form the top-left corner and iteratively compute and sample with respect to the probability until we arrives at the bottom-right corner. After we have obtained all the samples $\{a^m\}_{m=1}^M$, we estimate the gradient $\nabla\mathcal{L}_{LB}$ and update parameter θ.

Remark 1. *The time complexity of computing gradient via dynamic sampling is $\mathcal{O}(|S||T|+(|S|+|T|)M)$. The complexity of computing Z is $\mathcal{O}(|S||T|)$. The complexity of computing the probability distribution for sampling is $\mathcal{O}((|S|+|T|)M)$. Thus, the whole optimization process of each iteration is $\mathcal{O}(|S||T|+(|S|+|T|)M)$.*

3.2.3 Convergence Analysis

In view of the convexity of $\mathcal{L}_{LB}$, we analyze our algorithm in terms of convergence rate towards the global optimal. We show in Theorem 1 that PALM is guaranteed to converge in linear number of iterations to the global optimum, based on the linear assumption of function ϕ_θ.

Theorem 1. *Let $\mathcal{L}_{LB}$ be the lower bound of log-likelihood function in Equation 8. Denote $\theta^* = \arg\max_\theta \mathcal{L}_{LB}$ and the total variation $Var_{Pr_\theta(a)}(\phi_\theta(a)) \le L$. In iteration t of PALM in Algorithm 1, $\theta_{t+1} = \theta_t + \eta g_t$ where g_t is an unbiased estimation of the exact gradient $\nabla\mathcal{L}_{LB}(\theta_t)$. $Var(g_t) \le \frac{\sigma^2}{M}$ where M is sample size. Then, for any number of epochs $T > 1$, step size $\eta \le \frac{2}{L}$, and $\overline{\theta_T} = \frac{1}{T}\sum_{t=1}^T \theta_t$,we have:*

$$\mathbb{E}[\mathcal{L}_{LB}(\overline{\theta_T})] - \mathcal{L}_{LB}(\theta^*) \le \frac{||\theta_0-\theta^*||_2^2}{2\eta T} + \frac{\eta\sigma^2}{M}. \tag{13}$$

Theorem 1 states that PALM converges to the global optimal of $\mathcal{L}_{LB}$ with $\mathcal{O}(T)$ iterations. The objective function $\mathcal{L}_{LB}$ is convex w.r.t parameter θ [Jiang et al., 2018], as the first term of $\mathcal{L}_{LB}$ is always linear and the second term is convex. In addition, since the total variation is bounded $Var_{Pr_\theta(a)}(\phi_\theta(a)) \le L$, we can prove that $\mathcal{L}_{LB}$ is also $L-$smooth. Therefore, by unbiasedly estimate the gradient, we can prove the convergence rate based on classic results in the literature of stochastic gradient descent. The complete proof of Theorem 1 is left to supplementary materials. In practice, we can increase either the number of epochs T or the sample size M to approach better result.

3.3 INFERENCE IN TESTING

Inference in testing is to predict an alignment that attains the most likelihood. It is different from Section 3.2.1, which we use inference to generate alignment to estimate lower bounded loss function $\mathcal{L}_{LB}$.. Given sequence S,T and the learned model, the inference case can be computed as finding an alignment that have the highest likelihood without area distance:

$$\hat{a} = \arg\max_{a\in\mathcal{A}} e^{\sum_{k=1}^{|a|}\phi_\theta(\pi_S(a,k),\pi_T(a,k),a_k)} \tag{14}$$

Instead of enumerating all the possible valid paths in the matrix, we can still use dynamic programming to inference an alignment with $O(|S||T|)$ time complexity. Let $A'(i,j)$

represent the path from node (i, j) to the right corner with the maximum energy value in the alignment matrix of sequence S and sequence T. Then we have

$$A'(i,j)=\begin{cases} A'(i+1,j)+\phi_\theta(S_i,T_j,I_S),\ 1\le i\le |S|, j{=}|T|{+}1; \\ A'(i,j+1)+\phi_\theta(S_i,T_j,I_T),\ i{=}|S|{+}1, 1\le j\le |T|; \\ \max\begin{cases} A'(i+1,j+1)+\phi_\theta(S_i,T_j,M) \\ A'(i+1,j)+\phi_\theta(S_i,T_j,I_S) \\ A'(i,j+1)+\phi_\theta(S_i,T_j,I_T) \end{cases} \text{otherwise} \end{cases}$$

where the initial condition is $A'(|S|, |T|) = 0$ and the computation is line by line from the bottom right corner to the up left corner in the alignment matrix. The corresponding alignment $\hat{a}$ can be extracted from the matrix A' by following the edge that gives the largest energy value.

4 EXPERIMENTS

In this section, we first illustrate the experiment setups and then compare PALM with competing methods in terms of testing performance and learning efficiency, followed by an ablation study on the hyper-parameter of the area distance.

Dataset. The full dataset for protein alignment task is from Ma et al. [2014], which contains 10567 distinct sequences and 210477 pairwise aligned sequences. The ground-truth alignments are generated from DeepAlign tool [Wang et al., 2013]. The feature for every amino acid describes the geometric similarity, evolutionary relationship, and hydrogen-bonding similarity of proteins. The feature dimension is 41. We use the full dataset for the training step. For testing, we first partition the full dataset by the length of two sequences and then randomly pick 200 pairwise aligned sequences from each group. The length of sequences are divided into 5 groups: $[1, 100]$, $[100, 200]$, $[200, 400]$, $[1, 200]$, $[400, \infty)$.

Baselines. We consider a dynamic programming algorithm (DP) with a deterministic cost function [Needleman and Wunsch, 1970] to find the global alignment given two sequences. We use the cost for matching (i.e., M) as the summation over the feature vectors of two amino acids. The cost for insertion on sequence S (i.e., I_S) as the summation over the feature vector for the corresponding amino acid on sequence S. The cost for insertion on sequence T (i.e., I_T) is defined similarly. For algorithms that use a deep neural network with a richer set of features for the protein alignment problem, such as CNF [Ma et al., 2012] and DRNF [Wu and Xu, 2021], we do not compare with them for the sake of fairness. Because the parameter size of implemented PALM model is much smaller and the feature used is also limited. However, PALM can be easily extended to a deep neural network by changing the feature function ϕ_θ. We leave the comparison with these deep neural methods as future work.

Evaluation Metrics. We use Precision and Recall to measure the quality of the predicted alignments. In the "exact" scenario, only an exactly matched result is used for computing the true positive rate. The "4-offset" scenario is a relaxed measure that 4-position off the exact match is allowed. "10-offset" case is defined similarly. These two relaxed metrics are applied for depicting the model's performance for longer sequences. We also include the averaged computing time for estimating $\nabla \log Z_\theta$ over 100 epochs to reveal the time efficiency of our sampling approaches.

Implementation. The feature function ϕ_θ is a neural network with one single layer and the dimension of parameter θ is 82. ϕ_θ is adaptable to deeper neural networks with more parameters. For "match" case, we use the concatenation of feature vectors. For insertion on sequence S case, we use the concatenation of feature vector for S_i and a zero vector. For insertion on sequence T, we use the concatenation of a zero vector and feature vector for T_j. For the hyper-parameters, we set the number of sampled paths M to be 100 and the relative weight of area loss $\lambda = 50$. The learning rate η is initialized as 1 and decay by factor 0.9 for every 50 epochs. The maximum training epochs is set as 10^6. It takes a day to converge on the training set.

4.1 LEARNING EFFECTIVENESS FOR PALM

We compare with DP over 8 different testing sets for sequence alignment task. Results are collected in Table 1, where we observe that PALM has a higher Recall over exact/4-offset/10-offset settings when sequence T is much longer than sequence S and PALM has higher Precision over the exact/4-offset/10-offset settings when sequence S is longer. For $|S| \in [1, 100], |T| \in [400, +\infty)$, where the length difference of two sequences are large, Recall of PALM is twice as high as DP. Similarly, for $|S| \in [400, +\infty), |T| \in [1, 100]$, Precision of PALM is roughly twice as high as DP. When the difference of lengths of the two sequences becomes close, e.g., for $|S| \in [1, 100], |T| \in [100, 200]$, PALM has a relatively $3 - 6\%$ higher Recall value than DP. Similarly, for $|S| \in [100, 200], |T| \in [1, 100]$, PALM has a relatively $4 - 6\%$ higher Precision than DP.

4.2 ABLATION STUDY ON WEIGHT λ

Here we analyze how the hyper-parameter λ balances the area distance and the score function for inference during training. When λ approaches infinity, area distance becomes more important in the inference of $\hat{a}$ during training, which leads to $\hat{a}$ more similar to the ground-truth alignment a^*. It can be seen from Table 2 that when we select a suitable λ that strikes a balance between the area distance and the score function, we can learn a better model than pure maximum likelihood learning (when $\lambda \to +\infty$).

	$\|S\| \in [1, 100], \|T\| \in [100, 200]$		$\|S\| \in [100, 200], \|T\| \in [1, 100]$	
	Precision (%)	Recall (%)	Precision (%)	Recall (%)
	exact/ 4-offset/10-offset	exact/ 4-offset /10-offset	exact/4-offset/10-offset	exact/4-offset/10-offset
DP	7.8/**31.3/51.2**	20.4/39.0/56.3	20.2/40.4/59.4	6.1/26.3/**45.1**
PALM	**9.9**/29.8/48.7	**23.5/43.1/62.3**	**26.8/44.6/63.2**	**6.4/26.6**/43.1
	$\|S\| \in [1, 100], \|T\| \in [200, 400]$		$\|S\| \in [200, 400], \|T\| \in [1, 100]$	
	exact/ 4-offset/10-offset	exact/ 4-offset/10-offset	exact/4-offset/10-offset	exact/4-offset/10-offset
DP	5.2/**27.6/46.1**	32.0/39.8/46.7	30.0/37.5/44.7	**3.8/19.8/34.4**
PALM	**6.5**/26.9/43.3	**51.4/62.5/73.3**	**52.7/63.5/73.8**	3.3/18.7/31.0
	$\|S\| \in [1, 100], \|T\| \in [400, +\infty)$		$\|S\| \in [400, \infty), \|T\| \in [1, 100]$	
	exact/ 4-offset/10-offset	exact/ 4-offset /10-offset	exact/4-offset/10-offset	exact/4-offset /10-offset
DP	4.9/**26.4/45.2**	31.5/34.0/36.2	32.5/34.5/36.4	2.5/15.6/27.0
PALM	**5.1**/21.4/35.3	**75.3/81.1/86.3**	**76.0/81.1/86.1**	**3.1/18.1/29.1**
	$\|S\| \in [100, 200], \|T\| \in [200, 400]$		$\|S\| \in [200, 400], \|T\| \in [100, 200]$	
	exact/ 4-offset/10-offset	exact/ 4-offset /10-offset	exact/4-offset/10-offset	exact/ 4-offset /10-offset
DP	6.5/27.0/45.2	26.1/38.6/50.5	25.4/38.9/51.2	**5.9/22.2/37.0**
PALM	**10.4/30.0/47.0**	**34.8/49.4/62.9**	**36.2/50.4/63.4**	4.6/18.5/31.0
	$\|S\| \in [100, 200], \|T\| \in [400, +\infty)$		$\|S\| \in [400, +\infty), \|T\| \in [100, 200]$	
	exact/ 4-offset/10-offset	exact/4-offset/10-offset	exact/ 4-offset/10-offset	exact/4-offset/10-offset
DP	4.9/**24.1/41.0**	33.4/38.1/42.6	34.9/39.9/44.6	2.8/**14.4/24.8**
PALM	**6.1**/23.4/38.3	**61.1/69.0/76.5**	**62.5/71.0/78.8**	**3.2**/14.1/23.6

Table 1: Comparison of precision and recall between our method and dynamic programming (DP) over different lengths of protein sequences on PDB [Wu and Xu, 2021] dataset. 4-offset/10-offset are the relaxed measures. PALM gets better results especially on longer sequences and remote homologies than the competing approach.

	$\|S\| \in [1, 100], \|T\| \in [400, +\infty)$		$\|S\| \in [400, +\infty), \|T\| \in [1, 100]$	
	Precision (%)	Recall (%)	Precision (%)	Recall (%)
	exact/4-offset/10-offset	exact/4-offset/10-offset	exact/4-offset/10-offset	exact/4-offset/10-offset
$\lambda = 50$	**5.1/22.6/36.4**	75.3/81.1/86.3	75.9/**81.1**/86.0	2.6/17.0/27.2
$\lambda = 100$	4.6/21.3/35.2	75.3/81.1/86.3	**76.0/81.1/86.1**	3.1/**18.1/29.1**
$\lambda = 500$	4.5/20.9/34.0	**75.4/81.2/86.4**	75.9/81.0/85.9	3.1/17.4/28.3
$\lambda \to +\infty$	4.2/20.8/35.7	75.1/80.9/85.0	75.0/80.7/85.0	**3.5**/16.8/27.8

Table 2: Ablation study on hyper-parameter λ. When λ approaches infinity, area distance becomes more important in the inference of $\hat{a}$ during training, which leads to $\hat{a}$ more similar to the ground-truth alignment a^*. It can be seen that when we select a suitable λ that strikes a balance between the area distance and the score function, we can learn a better model than pure maximum likelihood learning. Note that the objective becomes MLE when $\lambda \to +\infty$.

4.3 TIME EFFICIENCY FOR GRADIENT COMPUTATION

We compare the time efficiency of computing the gradient with Automatic Differentiation (Autograd) [Paszke et al., 2017], which computes the exact gradient by automatically back-propagation, and our approach in Table 3. We implement both of the methods using PyTorch framework and select 100 sequence pairs in every testing set at random to measure the average time. Since the computational time is approximately proportional to the sequence length, the standard deviation of computation time is included for showing the impact of sequence length. We find that PALM is much more time efficient than Autograd among all the sequence length settings, where PALM only needs one-fourth of the time than the competing approach to approximate the gradient. In the extreme case where $|S| \in [400, +\infty)$ and $|T| \in [400, +\infty)$, Autograd method takes roughly 5 minutes to compute the gradient for just one sequence pairs while PALM only takes 1 minutes. Additionally, the standard deviation of the computing time for the competing approach is much higher than PALM, which means PALM is more stable to the variation of sequence lengths.

Sequence Length		Run Time (sec)	
$\|S\|$	$\|T\|$	PALM	Autograd
$[1, 100]$	$[1, 100]$	$\mathbf{0.7 \pm 0.2}$	2.5 ± 0.8
$[100, 200]$	$[100, 200]$	$\mathbf{2.7 \pm 0.9}$	9.4 ± 3.3
$[100, 200]$	$[200, 400]$	$\mathbf{6.6 \pm 2.3}$	25.4 ± 9.4
$[200, 400]$	$[100, 200]$	$\mathbf{6.2 \pm 2.0}$	23.2 ± 8.1
$[200, 400]$	$[200, 400]$	$\mathbf{12.5 \pm 2.3}$	51.7 ± 11.2
$[400, +\infty)$	$[400, +\infty)$	$\mathbf{63.4 \pm 32.0}$	297.6 ± 282
Averaged		$\mathbf{18.6 \pm 29.4}$	83.3 ± 182

Table 3: Time efficiency of computing the gradient among different testing sets. PALM is much time efficient than the competing method Autograd, which computes the exact gradient by automatically back-propagation, among all length intervals of two protein sequences.

5 CONCLUSION

In this paper we present a robust method for generative pairwise protein sequence alignment under biological errors and noises, which is a two-step generative model based on the area distance between two alignments in the alignment matrix. We propose a novel lower bound of the log-likelihood as the objective and efficiently estimate the gradient during optimization by dynamically sampling the alignments. We showed theoretically that PALM converges to the global optimum of the lower bound in linear number of iterations. In experiments, PALM can generate sequence alignments with higher precision and recall than competing methods especially when proteins under consideration are remote homologies. We also show that the optimization of PALM is much more computationally efficient by dynamically sampling alignments than the automatic gradient differentiable algorithm. For future work, we plan to model the feature function with deep neural nets, while feeding more informative features. We are also active to find more meaningful distance functions to model the difference of two alignments.

Acknowledgements

This research was supported by NSF grants IIS-1850243, CCF-1918327. We thank anonymous reviewers for their comments and suggestions.

References

Stephen F Altschul and David J Lipman. Protein database searches for multiple alignments. *Proceedings of the National Academy of Sciences*, 87(14):5509–5513, 1990.

Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

Fabrice Armougom, Sébastien Moretti, Olivier Poirot, Stéphane Audic, Pierre Dumas, Basile Schaeli, Vladimir Keduas, and Cédric Notredame. Expresso: automatic incorporation of structural information in multiple sequence alignments using 3d-coffee. *Nucleic Acids Res.*, 34(Web-Server-Issue):604–608, 2006.

Sivaraman Balakrishnan, Hetunandan Kamisetty, Jaime G Carbonell, Su-In Lee, and Christopher James Langmead. Learning generative models for protein fold families. *Proteins: Structure, Function, and Bioinformatics*, 79(4): 1061–1078, 2011.

Noah M. Daniels, Andrew Gallant, Norman Ramsey, and Lenore J. Cowen. Mrfy: Remote homology detection for beta-structural proteins using markov random fields and stochastic search. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 12(1):4–16, 2015.

Chuong B. Do, Samuel S. Gross, and Serafim Batzoglou. Contralign: Discriminative training for protein sequence alignment. In *Research in Computational Molecular Biology*, volume 3909, pages 160–174. Springer, 2006.

Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

Chitralekha Gupta, Emre Yilmaz, and Haizhou Li. Acoustic modeling for automatic lyrics-to-audio alignment. In *20th Annual Conference of the International Speech Communication Association*, pages 2040–2044. ISCA, 2019.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14 (8):1771–1800, 2002.

Kaixi Hou, Hao Wang, and Wu-chun Feng. Aalign: A simd framework for pairwise sequence alignment on x86-based multi-and many-core processors. In *2016 IEEE International Parallel and Distributed Processing Symposium*, pages 780–789. IEEE, 2016.

Chan-Seok Jeong and Dongsup Kim. Structure-based markov random field model for representing evolutionary constraints on functional sites. *BMC bioinformatics*, 17(1):1–11, 2016.

Bai Jiang, Tung-Yu Wu, Yifan Jin, Wing H Wong, et al. Convergence of contrastive divergence algorithm in exponential family. *Annals of Statistics*, 46(6A):3067–3098, 2018.

Kazutaka Katoh and Hiroyuki Toh. Recent developments in the MAFFT multiple sequence alignment program. *Briefings in bioinformatics*, 9(4):286–298, 2008.

Sudhir Kumar, Koichiro Tamura, and Masatoshi Nei. Mega3: integrated software for molecular evolutionary genetics analysis and sequence alignment. *Briefings in bioinformatics*, 5(2):150–163, 2004.

Rémi Lajugie, Damien Garreau, Francis R. Bach, and Sylvain Arlot. Metric learning for temporal sequence alignment. In *Advances in Neural Information Processing Systems*, volume 27, pages 1817–1825, 2014.

Jianzhu Ma, Jian Peng, Sheng Wang, and Jinbo Xu. A conditional neural fields model for protein threading. *Bioinformatics*, 28(12):59–66, 2012.

Jianzhu Ma, Sheng Wang, Zhiyong Wang, and Jinbo Xu. Mrfalign: Protein homology detection through alignment of markov random fields. *PLoS computational biology*, 10(3), 2014.

Debora S. Marks, Lucy J. Colwell, Robert Sheridan, Thomas A. Hopf, Andrea Pagnani, Riccardo Zecchina, and Chris Sander. Protein 3d structure computed from evolutionary sequence variation. *PLOS ONE*, 6(12):1–20, 12 2011.

Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443 – 453, 1970. ISSN 0022-2836.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS 2017 Workshop Autodiff*, 2017.

Robert Peharz. *Foundations of sum-product networks for probabilistic modeling*. PhD thesis, PhD thesis, Medical University of Graz, 2015.

T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195 – 197, 1981. ISSN 0022-2836.

Johannes Söding. Protein homology detection by hmm–hmm comparison. *Bioinformatics*, 21(7):951–960, 2005.

Johannes Söding, Andreas Biegert, and Andrei N Lupas. The hhpred interactive server for protein homology detection and structure prediction. *Nucleic acids research*, 33(suppl_2):W244–W248, 2005.

Martin Tompa. Lecture notes on biological sequence analysis. *University of Washington, Seattle, Technical report*, 2000.

Sheng Wang, Jianzhu Ma, Jian Peng, and Jinbo Xu. Protein structure alignment beyond spatial proximity. *Scientific reports*, 3:1448, 2013.

Fandi Wu and Jinbo Xu. Deep template-based protein structure prediction. *PLOS Computational Biology*, 17(5):1–18, 2021.