Protein Contact Prediction with Triplet CRF Factors

Roshan Rao and Edward Williams

December 20, 2016

1 Introduction

Protein structure prediction is a seminal problem in computational biology. The goal of protein structure prediction is to infer the 3D structure of a protein from its amino acid sequence. Previous work in this field generally uses simulated annealing to determine protein structure, however Pacheco et. al. are currently attempting to apply the D-PMP algorithm, which had success in the sub-problem of protein side chain prediction, to full structure prediction. D-PMP is a dynamic programming algorithm that operates on pairwise Markov Random Fields (MRFs). A protein can be represented as a pairwise MRF by assigning a variable to each amino acid to keep track of its position and orientation, and adding an edge between variables if the corresponding amino acids are close enough to interact.

In order to obtain efficient inference for D-PMP it is necessary to provide a sparse graphical model. Currently, D-PMP is being provided with the true graph structure, obtained using knowledge of the correct protein structure. Note that the complete graph is always a valid representation of a protein, but this requires high computational cost. However, although inter-atomic energies never vanish, they do have an inverse square relationship with distance. Therefore it is computationally beneficial to remove edges between amino acids that are beyond some specified interaction distance, and thus have inter-atomic energy near zero. On the other hand, removing edges between amino acids that are within that interaction distance can significantly impact energy evaluation and cause inference to fail. Seen in this manner, the problem of predicting a sparse graph structure is equivalent to determining the probability that two amino acids will be within the interaction distance in the final protein structure, with a goal of limiting false negatives. Here we describe a CRF-based method for estimating the adjacency matrix of a protein given its amino acid sequence, a problem known as protein contact prediction.

2 Related Work

Predicting amino acid contacts is an active area of research in the field of protein structure prediction. An accurate adjacency matrix for a protein's 3D structure is useful for two applications: initializing a protein structure prediction algorithm, and using the amino acid contacts to predict 3D structure directly. An accurate set of protein contacts reduces the number of pairwise interactions that must be calculated when predicting 3D protein configuration with an energy model. This reduces the computational load, speeding up execution time [1]. Efforts to predict a 3D protein structure *de novo* by using amino acid proximity estimates to constrain structure have proven partially successful, with an RMSD error range of 2.7-5.1 Å across 75% of the protein chain for proteins with 50-260 amino acid residues [2].

Rather than attempting to predict amino acid contacts from an amino acid sequence directly, modern contact prediction algorithms have relied on relationships between homologous protein domains, which are subdivisions of proteins that are evolutionarily related. Ekeberg et al. use sequence-aligned protein domains to fit a Potts model to protein domain families [3]. The Potts model, learned via pseudolikelihood maximization, contains a set of coupling strengths between amino acid pairs at particular positions in the amino acid chain, and is used to produce a contact map. In this model, amino acid positions in a sequence are likely to be in contact if their values are correlated across many proteins domains in the sequence's family. Other approaches model protein families as Gaussian Graphical Models with some correlations existing between families [4]. Unfortunately, protein domain family models, such as the Potts model presented by Ekeberg et al., tend to produce

many false negatives in their final predictions. As mentioned above, false negatives are much more detrimental to D-PMP inference than false positives. The CoinFold contact prediction system [5], combines evolutionary correlation models, sequence-level features, and some elements of 3D structure prediction, which may produce a model more uniquely suited to initializing a physical protein structure prediction algorithm. However, the CoinFold model still focuses on minimizing total error and obtains a large number of false negatives. As well, attempting to predict the 3D structure of a protein using a model initialized with a different structure prediction system may produce spurious results.
A paper recently presented by Golkov et al. at NIPS provides a close analogue to our stated goal of initializing a protein, although they use an co-evolutionary Potts model technique. Golkov et al. use the results of the Potts model to train a convolutional neural network to predict contact maps. This method outperforms the contact prediction technique presented by Ekeberg et al., as it the convolutional neural network architecture can use amino acid pairs' neighbors to inform prediction [6].

3 Model

Let $y_1, y_2, \dots y_N$ represent the observed amino acid sequence of a protein. Each variable y_i denotes the type of amino acid at the i-th position in the sequence. Our goal is to predict a symmetric adjacency matrix X:

$$\mathbb{P}\left(\begin{bmatrix} 0 & x_{12} & x_{13} & \dots & x_{1N} \\ x_{12} & 0 & x_{23} & \dots & x_{2N} \\ x_{13} & x_{23} & 0 & & \\ \vdots & \vdots & & \ddots & \vdots \\ x_{1N} & x_{2N} & & \dots & 0 \end{bmatrix} \middle| \, y_1, \dots, y_N \right) \tag{1}$$

where entry $x_{ij} = 1$ if the corresponding amino acids y_i, y_j are within the specified interaction distance. We do this by extracting features from the amino acid sequence and creating an exponential family model, which we represent using a conditional random field (CRF).

3.1 Features

We use three features extracted from the observed sequence y along with a three-factor interaction feature between variables x_{ij}, x_{ik}, and x_{jk}. The sequence features are a distance feature $\phi_{\text{dist}}(y_i, y_j) = |i - j|$, a sequence length feature $\phi_{\text{seqlen}}(y) = N$ (which acts as a prior), and amino acid pair features. The amino acid pair features are defined as follows: let A, B be two types of amino acids. Then $\phi_{A,B}(y_i, y_j) = \mathbb{1}\{y_i = A, y_j = B\}$, with $\phi_{A,B} = \phi_{B,A}$. Since there are 20 total types of amino acids, this creates 210 distinct indicator variables.

Additionally we note that $\mathbb{P}(x_{ij} \mid x_{ik} = 1, x_{jk} = 1) > \mathbb{P}(x_{ij})$. This leads to triplet factors $\phi_2(x_{ij}, x_{ik}, x_{jk}) = \mathbb{1}\{x_{ij} + x_{ik} + x_{jk} = 2\}$ and $\phi_3(x_{ij}, x_{ik}, x_{jk}) = \mathbb{1}\{x_{ij} + x_{ik} + x_{jk} = 3\}$.

3.2 Graphical Model

The full log-likelihood is then given by

$$\begin{aligned} \mathcal{L}(y, \theta) = \exp \Bigg\{ & \sum_{i,j} x_{ij} \Big[\theta_{\text{dist}} |i - j| + \theta_{\text{seqlen}} N + \sum_{A,B} \theta_{A,B} \mathbb{1}\{y_i = A, y_j = B\} \Big] \\ & \sum_{i,j,k} \Big(\theta_2 \mathbb{1}\{x_{ij} + x_{ik} + x_{jk} = 2\} + \theta_3 \mathbb{1}\{x_{ij} + x_{ik} + x_{jk} = 3\} \Big) - \Phi(y, \theta) \Bigg\} \end{aligned} \tag{2}$$

This likelihood then factorizes into the graphical model depicted in Figure 1 below.

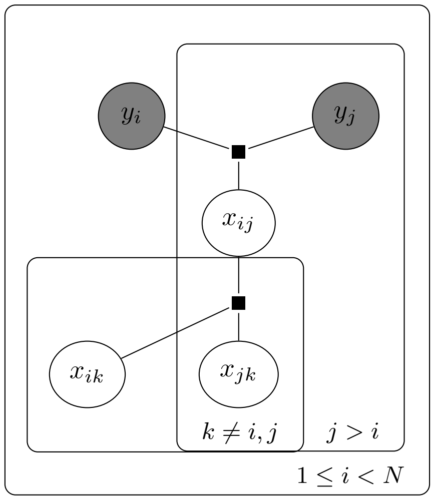


Figure 1: Graphical model of Eq. 2. x_{ij} has factor representing conditional probability given y_i, y_j. x_{ij}, x_{jk}, x_{ik} have factors increasing the probability of the third if the other two equal 1.

4 Methods

4.1 Mean Field

As performing full inference on such a large densely connected graph would be computationally difficult, we employed several approximations in the process of learning and inference for our model. One such method, mean field inference, approximates the graph as fully disconnected, and minimizes a variational objective function:

$$\mathcal{L}(y, \theta) = \exp\left\{ \sum_{i,j} x_{ij} \left[\theta_{\text{dist}} |i - j| + \theta_{\text{seqlen}} N + \sum_{A,B} \theta_{A,B} \mathbb{1}\{y_i = A, y_j = B\} \right] \right.$$
$$\left. \sum_{i,j,k} \Big(\theta_2 \mathbb{1}\{x_{ij} + x_{ik} + x_{jk} = 2\} + \theta_3 \mathbb{1}\{x_{ij} + x_{ik} + x_{jk} = 3\} \Big) - F(\hat{\mu}_y, y, \theta) \right\} \quad (3)$$

Where

$$F(\mu, \theta) = \sum_{i,j} \mu_{ij} \left[\theta_{\text{dist}} |i - j| + \theta_{\text{seqlen}} N + \sum_{A,B} \theta_{A,B} \mathbb{1}\{y_i = A, y_j = B\} \right]$$
$$+ \sum_{i,j,k} \left[\theta_2 \left[\mu_{ij} \mu_{jk} (1 - \mu_{ik}) + \mu_i j (1 - \mu_{jk}) \mu_{ik} + (1 - \mu_{ij}) \mu_{jk} \mu_{ik} \right] + \theta_3 \mu_{ij} \mu_{jk} \mu_{ik} \right]$$
$$- \mu_{ij} \log \mu_{ij} - (1 - \mu_{ij}) \log (1 - \mu_{ij}) \quad (4)$$

and μ_{ij} is the marginal probability of x_{ij} under the mean field approximation. To calculate the mean-field likelihood, first we need $\hat{\mu}_y$, the value of μ that maximizes $F(\mu, \theta)$ for a particular observation set y. $\hat{\mu}_y$ is computed using coordinate ascent over all μ_{ij}. which is a significant contributor to the computational cost of our training algorithm.

Calculating $\hat{\mu}$ is implemented as MFINFERENCE in Algorithm 1, however despite implementing MFINFERENCE in C++, the resulting inference algorithm is extremely slow. Inference requires approximately 45 minutes on a dataset of 11 relatively small proteins ($N < 1000$ for all proteins). On a larger dataset of 80 proteins, we ran inference for several days but it did not converge. The bottleneck is clearly MFINFERENCE. On the 11 protein dataset a single call to MFINFERENCE could

Algorithm 1 Mean Field estimation of parameters θ. Inputs are σ, the sufficient statistics, and $\phi(y)$ a feature vector for each of L proteins in the training set. The MFGETTHETA function is implemented in MATLAB, but the MFINFERENCE function is implemented in C++ with a Mex interface.

```
function MFGETTHETA($\sigma, \phi(y)$)
    while $progress > convTol$ do
        $\hat{\mu} \leftarrow$ MFINFERENCE($\phi(y), \theta$)        ▷ note that this is a $O(20LN^3)$ operation
        $ll \leftarrow \theta^T\sigma - F(\hat{\mu}, \theta)$        ▷ calculating $F$ also $O(LN^3)$
        $\nabla_\theta ll \leftarrow \sigma - \nabla_\theta F(\hat{\mu}, \theta)$
        $\theta \leftarrow \theta - \epsilon \nabla_\theta ll$
    end while
end function

function MFINFERENCE($\phi(y), \theta$)
    for $\ell \leftarrow 1 : L$ do
        for $iter \leftarrow 1 : 20$ do        ▷ until convergence
            for $i \leftarrow 1 : N - 1$ do
                for $j \leftarrow i + 1 : N$ do
                    $\alpha \leftarrow 0$
                    Add value from sequence features to $\alpha$
                    for $k \leftarrow 1 : N, k \neq i, j$ do
                        Add value from three-way factors to $\alpha$
                    end for
                    $\hat{\mu}^{(\ell)}_{i,j} = \exp(\alpha)/(1 + \exp(\alpha))$
                end for
            end for
        end for
    end for
    return $\hat{\mu}$
end function
```

take up to five minutes. On the large dataset, it could take up to five hours. No other function call takes time within that order of magnitude.

One obvious solution to this problem is parallelization. Estimating $\hat{\mu}^{(\ell)}$ is independent of all other proteins given the features for the ℓ-th protein and θ. Therefore, we replace MFINFERENCE with a parallelized version in Algorithm 2.

Algorithm 2 Parallelized Mean Field estimation of parameters θ. Same as Alg. 1, but with MFINFERENCE implemented as MFINFERENCEPARALLEL.

```
function MFINFERENCEPARALLEL($\phi(y), \theta$)
    $nRunningThreads \leftarrow 0$
    for $\ell \leftarrow 1 : L$ do
        Wait until $nRunningThreads < nMaxThreads$
        Launch new thread to calculate $\mu^{(\ell)}$. Decrement $nRunningThreads$ when thread finishes.
        Increment $nRunningThreads$
    end for
    return $\hat{\mu}$
end function
```

4.2 Pseudo Likelihood Maximization

To more effectively learn our model parameters, we used pseudo-likelihood maximization, which maximizes the conditional likelihood of nodes given their neighbors $p(x_{ij}|x_{\Gamma(x_{ij})})$. This allows for the rapid computation of the log-normalizer, without the costly calculation of $\hat{\mu}$ at each timestep.

$$\log p(x_{ij}=1|x_{\Gamma(x_{ij})}) = \left[\theta_{\text{dist}}|i-j| + \theta_{\text{seqlen}}N + \sum_{A,B}\theta_{A,B}\mathbb{1}\{y_i = A, y_j = B\}\right] + \sum_k \theta_3\left[\mathbb{1}\{x_{ik}+x_{jk}=2\} + \theta_2\mathbb{1}\{x_{ij}+x_{jk}=1\}\right]$$

$$\log p(x_{ij}=0|x_{\Gamma(x_{ij})}) = \sum_k \theta_2\mathbb{1}\{x_{ik}+x_{jk}=2\}$$

$$Pll(y,\theta) = \sum_{ij}\log p(x_{ij}|x_{\Gamma(x_{ij})}) - \log\left[\exp(\log p(x_{ij}=1|x_{\Gamma(x_{ij})})) + \exp(\log p(x_{ij}=0|x_{\Gamma(x_{ij})}))\right] \quad (5)$$

We replace the calculation of ll with Pll in Algorithm 1. This removes the need for MFINFERENCE during training (although we still use MFINFERENCE at test time).

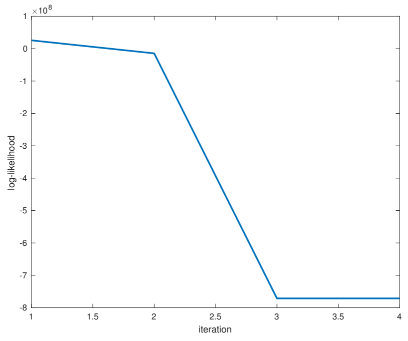
5 Results

5.1 Data

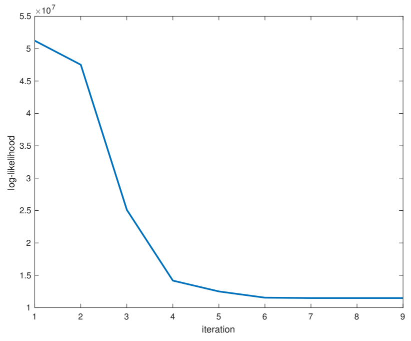
To train and test our model, we used protein data from the Protein Data Bank. PDB files contain x, y, z coordinates for each atom in each amino amino acid of a protein. From these coordinates, distances between atoms can be computed and converted into a graph, where edges exist between amino acids if they lie within a threshold distance d of each other, with $d = 10$ for our initial data processing. We are using a 477-protein set for our initial model training and testing, with 331 allocated for training and 146 for test.

5.2 Training

Below are log-likelihood traces when training the mean-field and pseudo-log-likelihood models on a training set of 331 proteins. Training the mean field model on the Brown CS Grid with 11 threads (1 main thread, 10 threads for calculating $\hat{\mu}$) took approximately 15 hours, while training the pseudolikelihood model with a single thread (using 10 for calculating mean field marginals at test time) took approximately 100 minutes.



(a) Mean field log-likelihood trace



(b) Pseudo log-likelihood trace

Figure 2: Log-likelihood traces during training

5.3 Model Results - ROC Curve

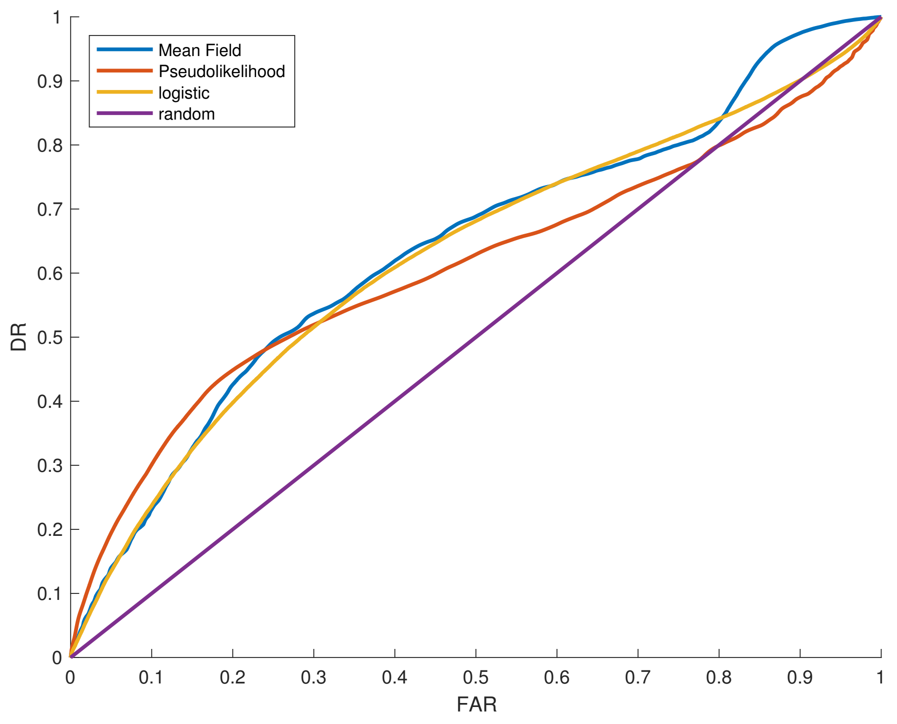


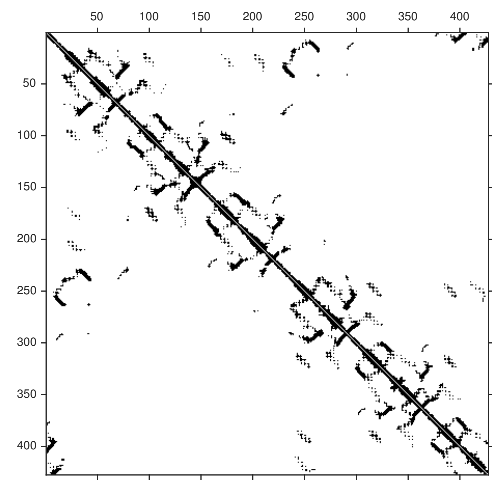
Figure 3: ROC Curve for Mean Field, Pseudolikelihood, Logistic, and Random models.

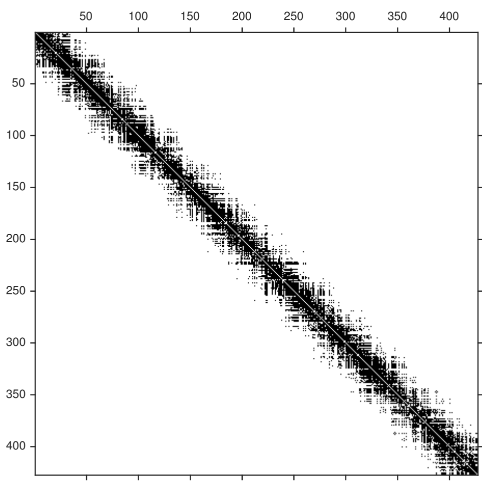
The above figure displays the performance of the two models and a logistic baseline on a data set of 146 test proteins, after being trained on the 331 proteins in the training set. Neither the pseudolikelihood model nor the mean field model greatly outperforms the logistic baseline. This is particularly interesting, given that the pseudolikelihood and mean field models assign weight zero to all of the amino acid presence features, unlike the logistic model. The pseudolikelihood and mean field models are entirely relying on triplet factor weights and amino acid pair distance to perform inference. The logistic model assigns a strongly negative weight to the prior, with magnitude ten times that of the amino acid weights. The logistic model, however, assigns a weight close to zero to the distance features. This indicates that the logistic model is largely relying on the prior to determine if an amino acid pair is an edge, with some input from the distance and amino acid type features. This indicates that the amino acid pair features provide very little information to the triplet factors.

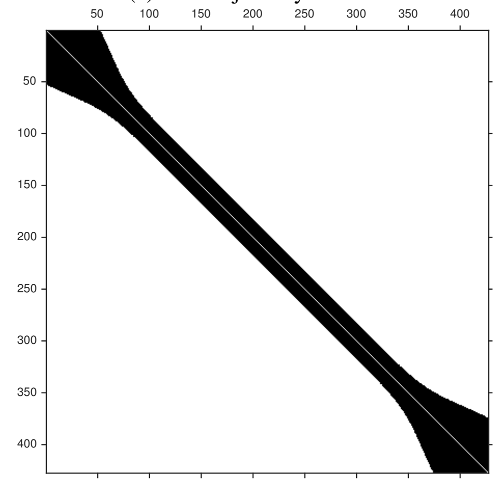
The triplet factors, in the absence of observations that provide a high signal to noise ratio, could encourage a "clumping" behavior that would exacerbate the model's dependence on the distance feature. Edge indicator variables that possess a short protein chain distance, and thus the highest possible signal under our limited observation set, would encourage other adjacent edge indicator variables to appear as positive. A sign of this is present in the ROC curve - the pseudolikelihood model (and the mean field model, to a lesser extent) outperforms the logistic model at lower decision boundaries, as it is likely picking up edges that have a lower distance and are connected via triplet factors.

All models exhibit a *non-convex* ROC curve, which indicates that some some problem exists with the models themselves. One possibility is that there is an as-yet-undiscovered bug in our code that is impeding gradient calculation. Another is that the pseudolikelihood and mean field approximations are not suitable to this graphical model topology, although this is called into question by the behavior of the logistic model, which also has a non-convex ROC curve.

5.4 Model Results - Individual Proteins


(a) True adjacency matrix.


(b) Logistic regression estimate.


(c) Mean field estimate

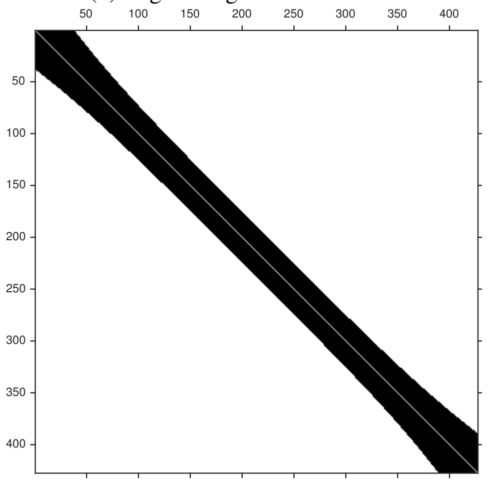

(d) Pseudo log-likelihood estimate

Figure 4: True adjacency matrix and estimated matrices using logistic, mean field, and pseudo-log likelihood learning for protein 1EPS.

Figure 4 shows estimated adjacency matrices using the two methods discussed, as well as a logistic regression baseline. Both methods used, as well as the logistic regression baseline show a heavy reliance on the sequence distance feature; essentially no long range contacts are captured at all. This suggests that the current feature set is not able to provide enough information to predict long range contacts, and that only the distance feature is being used. Therefore we believe it would be necessary to provide better features (such as evolutionary features) before assessing the success of these methods.

Additionally, both the mean field and pseudo log-likelihood estimates seem to ignore amino acid type altogether. We are unsure why this should be the case. Although amino acid type does provide very little information, it should be enough to at least moderately affect the estimate. This could arise from a characteristic of the two approximations, where the amino acid feature is drowned out by other features, it could arise from the three-factor potentials encouraging clustering too strongly, or it could arise from a simple coding error.

Another point of interest is that the true adjacency matrix (Fig. 4a) does show some degree of clumping, which is encouraged by the three-factor potentials. This suggests that if stronger unary factors could be found, the three-factor potentials may work well, however with only the sequence distance feature providing a strong unary indicator, they work very poorly.

6 Conclusion

In this paper, we presented a method for identifying amino acid contacts in a protein chain using a conditional random field with three-edge spatial proximity triplet factors. Our results demonstrated that while the model was able to successfully learn weights for the triplet factors, the observation set that we used was insufficient to perform predictions knowledge of amino acid types and their distance in a protein chain are

7 Future Work

Although the potential of the three-edge factors when applied to amino acid contact prediction has been demonstrated by this research, we have also shown that the amino acid observations used in our model are not particularly useful. A relatively small change to our current model would be conditioning on the presence of edges that are close in the protein chain, and performing CRF training and inference otherwise normally. We investigated this possibility when developing our current method, but given the poor performance of the existing models on edges without that are not close in the protein chain, this may not provide any more information. A better approach would be to use a different source of observations altogether, such as returning to the evolutionary protein domain models mentioned earlier.

An extension of our work would be applying the three-edge factors to the protein domain models learned by Ekeberg et al. and Golkov et al. [3] [6]. The use of evolutionary relationships and spatial relationships between amino acids in protein domain-specific Potts models provide a much stronger signal than simply observing the amino acid distance and amino acid types in a given edge. Combining our triplet factors, which will be particularly relevant when learning domain-specific potentials rather than general parameters across many protein types, with the existing pairwise relationships could capture more of the amino acid contact information. A more computationally expensive but likely useful model would learn triplet potentials for every individual triplet of amino acid pairs in the protein domain, although a large amount of data and time may be required to train such a model.

8 Contributions

Edward Williams wrote data processing code in Python, C++ code for calculating pseudo-log-likelihood, and C++ code for calculating $\hat{\mu}$ when performing mean field inference. Roshan Rao wrote the C++ code for calculating the mean field log-normalizer, parallelizing the calculation of $\hat{\mu}$, and performing logistic regression training and inference. Both members of the group wrote the final research paper.

References

[1] Qiqige Wuyun, Wei Zheng, Zhenling Peng, and Jianyi Yang. November 1, 2016. A large-scale comparative assessment of methods for residue?residue contact prediction. Brief Bioinform (2016) doi:10.1093/bib/bbw106

[2] Debora S. Marks, Lucy J. Colwell, Robert Sheridan, Thomas A. Hopf, Andrea Pagnani, Riccardo Zecchina, Chris Sander. 25 October 2011. 3D Protein Structure Predicted from Sequence. `arXiv:1110.5091 [q-bio.BM]`

[3] Magnus Ekeberg, Cecilia Lövkvist, Yueheng Lan, Martin Weigt, Erik Aurell. 12 January 2013. Improved contact prediction in proteins: Using pseudolikelihoods to infer Potts models. `arXiv:1211.1281 [q-bio.QM]`

[4] Ma J., Wang S., Wang Z., Xu J. August 14, 2015. Protein contact prediction by integrating joint evolutionary coupling analysis and supervised learning. Bioinformatics 2015:btv472

[5] Sheng Wang, Wei Li, Renyu Zhang, Shiwang Liu and Jinbo Xu. April 12, 2016. CoinFold: a web server for protein contact prediction and contact-assisted protein folding. Nucl. Acids Res. (2016) doi: 10.1093/nar/gkw307

[6] Vladimir Golkov, Marcin J. Skwark, Antonij Golkov, Alexey Dosovitskiy, Thomas Brox, Jens Meiler, and Daniel Cremers. Protein contact prediction from amino acid co-evolution using convolutional networks for graph-valued images. In Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.