Supplemental Materials: Predicting protein inter-residue contacts using composite likelihood maximization and deep learning

1. Calculation of gradients of composite likelihood

To find parameters $h_i$ and $e_{ij}$ such that $\mathcal{CL}_{\text{pairwise}}$ is maximized, we employed the classical BFGS technique with efficient calculation of gradients. The gradients of $\mathcal{CL}_{\text{pairwise}}$ can be calculated in polynomial time through the following equations.

\[
\frac{\partial \mathcal{CL}_{\text{pairwise}}}{\partial h_i(a)} = (L - 1)f_i(a) - \frac{1}{M} \sum_{m=1}^{M} \sum_{k=1 \atop k \neq i}^{L} P_{ik}^m(a) \tag{1}
\]

\[
\frac{\partial \mathcal{CL}_{\text{pairwise}}}{\partial e_{ij}(a,b)} = (2L - 3)f_{ij}(a,b) - \frac{1}{M} \sum_{m=1}^{M} |P_{ij}^m(a,b) + \sum_{k=1 \atop k \neq i, k \neq j}^{L} (I(X_j^m = b)P_{ik}^m(a) + I(X_j^m = a)P_{kj}^m(b))| \tag{2}
\]

Composite likelihood maximization is a tradeoff between model capacity and model complexity. Table 1 shows the time complexity of the calculation of the gradients of actual likelihood, pseudo-likelihood, and pairwise composite likelihood. Here, $q$ denotes the alphabet size of amino acids types, $L$ denotes the length of protein and $M$ denotes the number of sequences in MSA.

<table>
<thead>
<tr>
<th>Likelihood function</th>
<th>Time complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actual likelihood $\mathcal{L}$</td>
<td>NP-hard</td>
</tr>
<tr>
<td>Pseudo-likelihood $\mathcal{PL}$</td>
<td>$O(qL^2M)$</td>
</tr>
<tr>
<td>Pairwise composite likelihood $\mathcal{CL}$</td>
<td>$O(q^2L^2M)$</td>
</tr>
</tbody>
</table>

Table 1: Time complexity for calculating likelihood function.

2. Pre- and post-processing, regularization, and speed-up strategies

It is very common that some homologous proteins of a query protein show considerable sequence similarity, forming redundancy in MSA. This redundancy makes the assumption of independence among homologous proteins false and the likelihood function inaccurate. To reduce the impact caused by redundancy, we pre-processed MSA by weighting all homologous proteins within it as performed by plmDCA [2] and PSICOV [3]. Similar to [1], we also applied the average-product correction (APC) technique on the predicted contacts as post-processing procedure.

The MRF model used in this study contains a total of $O(L^2)$ parameters, usually exceeding the number of homology proteins in MSA. For example, about $2 \times 10^7$ parameters are needed to model a query protein with length $L = 300$;
in contrast, few MSAs have sufficient homologous proteins for estimating these parameters, thus causing potential overfitting. To avoid overfitting, we added a penalty term to the likelihood function as follows:

\[ R_{t_2} = \lambda h \sum_{i=1}^{L} \|h_r\|_2^2 + \sum_{i=1}^{L} \sum_{j=i+1}^{L} \|e_{ij}\|_2^2 \]  

(3)

To speed up likelihood maximization, we began with initial parameters calculated using plmDCA. We also exploited the parallelism in calculating gradient for residue pairs and implemented our method using OpenMP. Overall, it usually takes only a few hours for clmDCA to predict contacts for proteins with typical length of 200 amino acids.

3. Comparison of plmDCA and clmDCA: a case study

![Figure 1: Two residue pairs in protein 1ne2A. (a) Non-contact residue pair ALA183-ILE189 and (b) Contacting residue pair THR75-MSE97.](image-url)
4. Building protein 3D structures using the predicted inter-residue contacts

Figure 2: Comparison of quality of structures built using predicted contacts by (a) clmDCA vs. plmDCA. (b) clmDCA alone version clmDCA together with refinement using deep learning. Here protein quality is measured using TMscore. Data set: PSICOV.

5. Comparison of runtime

The runtime of clmDCA is mainly determined by the number of sequences of MSA and the columns of MSA. For runtime benchmarks, we generated synthetic MSAs with 3000 sequences and 50, 250 and 500 columns. 4 CPU cores are used for clmDCA, plmDCA and PSICOV. As shown in Figure 3, clmDCA is generally 1.5 – 2 times slower than plmDCA. We will explore GPU techniques to further speed up clmDCA in our future work.
Figure 3: Comparison of runtime

6. Performance of 3-order clmDCA

<table>
<thead>
<tr>
<th>Methods</th>
<th>( L/10 )</th>
<th>( L/5 )</th>
<th>( L/2 )</th>
<th>( L )</th>
<th>( L/10 )</th>
<th>( L/5 )</th>
<th>( L/2 )</th>
<th>( L )</th>
</tr>
</thead>
<tbody>
<tr>
<td>clmDCA</td>
<td>0.57</td>
<td>0.53</td>
<td>0.44</td>
<td>0.36</td>
<td>0.53</td>
<td>0.49</td>
<td>0.38</td>
<td>0.29</td>
</tr>
<tr>
<td>3-order clmDCA</td>
<td>0.56</td>
<td>0.53</td>
<td>0.46</td>
<td>0.36</td>
<td>0.54</td>
<td>0.49</td>
<td>0.40</td>
<td>0.28</td>
</tr>
</tbody>
</table>

References

