Template-based protein modeling project

Olha Kholod
Carlos Martinez Villar
Farhan Quadir
Sriganesh Pera
Md. Bodruzzaman Sarker

March 6, 2019

Data preparation and sharing

Data:

- T0951 sequence as a target
- 5z82.pdb as a target native structure

The project will be shared on GitHub: template-based-modeling

Alignment

- perform blast search for identifying template candidate (less than 90% similarity);
- select protein 4i1a, which share 60.59% of sequence similarity;
- perform alignment of template protein sequence to target protein sequence (using Modeller);

Chain A, Crystal Structure Of Karrikin Insensitive 2 (kai2) From Arabidopsis Thaliana	352	352	97%	8e-123	60.59%	4IH1_A (scored below threshold on previous iteration)

Libraries and software tools

- numpy
- math
- Bio.PDB
- matplotlib
- Modeller

Algorithm development: initialization

- define Ca-Ca distances as a square root of the sum of the squares of the differences between corresponding coordinates;
- define probability density functions (pdfs) as

$$f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

- initialized arrays for the following data structures: number of target points, number of template points, target Ca-Ca distances, template Ca-Ca distances and pdfs.

Algorithm development: restrains extraction and representation

- extract Ca-Ca distances between each pair of amino acid residues and store these distances in target and template distance arrays.
- compute probability density function (pdf) using the following function and store obtained values in an array:

```
pdf_array[i,j] = pdf[target_dist_array[i,j], template_dist_array[i,j], σ)
```

- multiply these values (excluding values on the diagonal) and maximize its value with logarithmic function.

```
product_pdf = prod(pdf_array_excuding_diag[pdf_array_excuding_diag > 0])
log_product_pdf = -log(product_pdf)
```

Algorithm development: restrains extraction and representation

The matrix representation for Ca-Ca distances:

0	3.80354861	5.25631982	5.21633712	6.61498367	8.50955557	10.04463	10.8140367	12.6723028	14.6265519	15.1495024	16.7478824
3.80354861	0	3.80195397	5.43371779	5.2973953	5.86864363	8.511825	9.77057383	10.5642437	12.4376328	13.8290149	15.2995608
5.25631982	3.80195397	0	3.79538918	5.59456048	4.88216817	5.950608	8.32283191	9.76050987	10.6080428	11.778059	14.0727708
5.21633712	5.43371779	3.79538918	0	3.80499934	5.3240602	5.288231	6.02269408	8.61113918	10.0894529	10.1065781	12.1163933
6.61498367	5.2973953	5.59456048	3.80499934	0	3.81313375	5.646052	4.99302954	6.22689762	8.8815904	9.45678725	10.3952904
8.50955557	5.86864363	4.88216817	5.3240602	3.81313375	0	3.800763	5.34636783	5.19498171	6.59188357	8.48540948	10.0672326
10.0446275	8.51182501	5.95060812	5.28823052	5.64605225	3.80076269	0	3.81987775	5.37425809	5.16325769	5.90832794	8.66614978
10.8140367	9.77057383	8.32283191	6.02269408	4.99302954	5.34636783	3.819878	0	3.81569941	5.49047685	4.63012926	6.09547882
12.6723028	10.5642437	9.76050987	8.61113918	6.22689762	5.19498171	5.374258	3.81569941	0	3.7908102	5.13792069	5.18848668
14.6265519	12.4376328	10.6080428	10.0894529	8.8815904	6.59188357	5.163258	5.49047685	3.7908102	0	3.78854867	5.58407826
15.1495024	13.8290149	11.778059	10.1065781	9.45678725	8.48540948	5.908328	4.63012926	5.13792069	3.78854867	0	3.80158428
16.7478824	15.2995608	14.0727708	12.1163933	10.3952904	10.0672326	8.66615	6.09547882	5.18848668	5.58407826	3.80158428	0

Algorithm development: optimization

- define gradient in order to construct a vector that helps to pinpoint direction to the local minimum;
- calculate gradient for each target coordinate and target distances:

```
grad_x[i] = grad_dist[i,i+1]*(target_points[i,0] - target_points[i+1,0])/target_dist_array[i,i+1]

grad_y[i] = grad_dist[i,i+1]*(target_points[i,0] - target_points[i+1,0])/target_dist_array[i,i+1]

grad_z[i] = grad_dist[i,i+1]*(target_points[i,0] - target_points[i+1,0])/target_dist_array[i,i+1]

grad_points[i] = grad_dist[i,i+1]*(target_points[i,0] - target_points[i+1,0])/target_dist_array[i,i+1]
```

Algorithm development: optimization

- build a gradient descent function to adjust target coordinates in a way that minimize the error of the predicted protein structure;

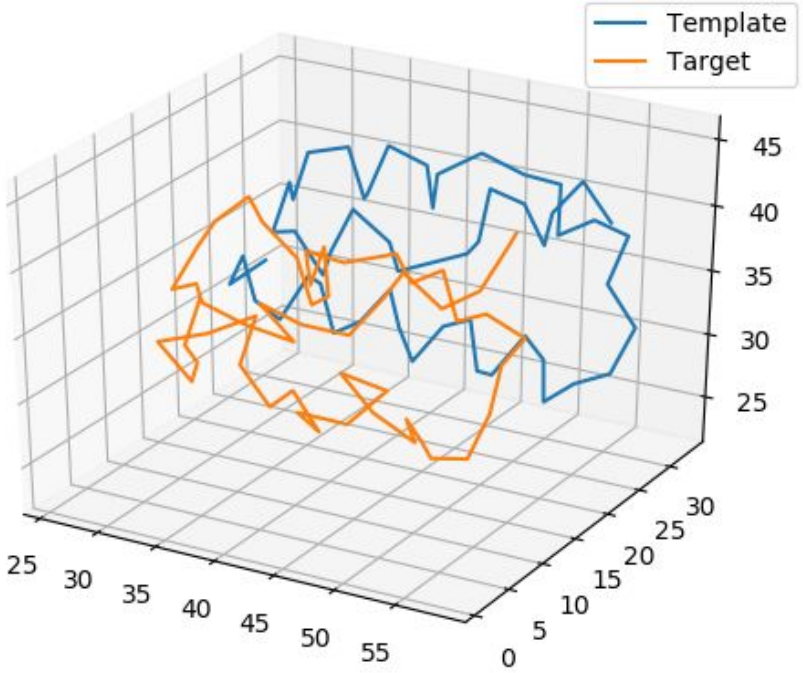
```
target_points = - alpha*gradient_points

error = abs(log_pdf - new_log_pdf)
```

- obtain three-dimensional array, where each point corresponds to target coordinates;

Training process

- perform 703 iterations
- logpdf = 26309
- error = 1.04e-05.

Results


Template
Target
45
40
35
30
25
25
30
35
40
45
50
55
0
5
10
15
20
25
30

Evaluation

$$\text{TM-score} = \max \left[\frac{1}{L_{\text{target}}} \sum_{i}^{L_{\text{aligned}}} \frac{1}{1 + \left(\frac{d_i}{d_0(L_{\text{target}})} \right)^2} \right]$$

, where L_{target} and $L_{aligned}$ are the lengths of the target protein and the aligned region respectively. d_i is the distance between the ith pair of residues and

$$d_0(L_{\text{target}}) = 1.24 \sqrt[3]{L_{\text{target}} - 15} - 1.8$$

Questions