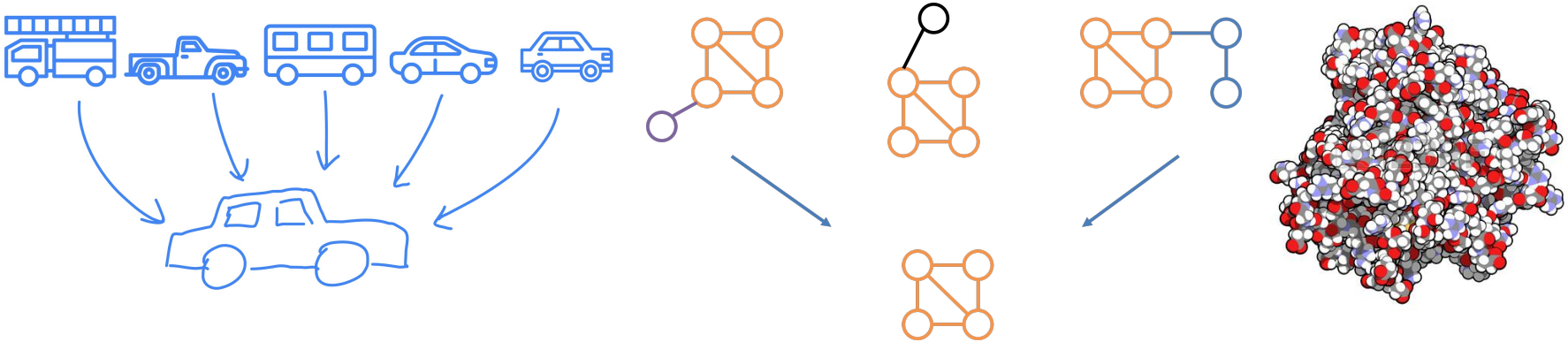
Graph Matching, Pattern Learning, and Protein Modeling

Wesley Wei Qian
Supervised by Prof. Pengyu Hong

Abstract

One of the most amazing capabilities of human beings is to extract common spatial patterns from observations and use these patterns to make inferences.

Here we want to build a similar algorithm/model where we can learn the common pattern from a special kind of spatial representation called attributed relational graph (ARG) and apply this model to protein crystallography data in order to mine functional units from proteins with similar function or discover novel structure motif.

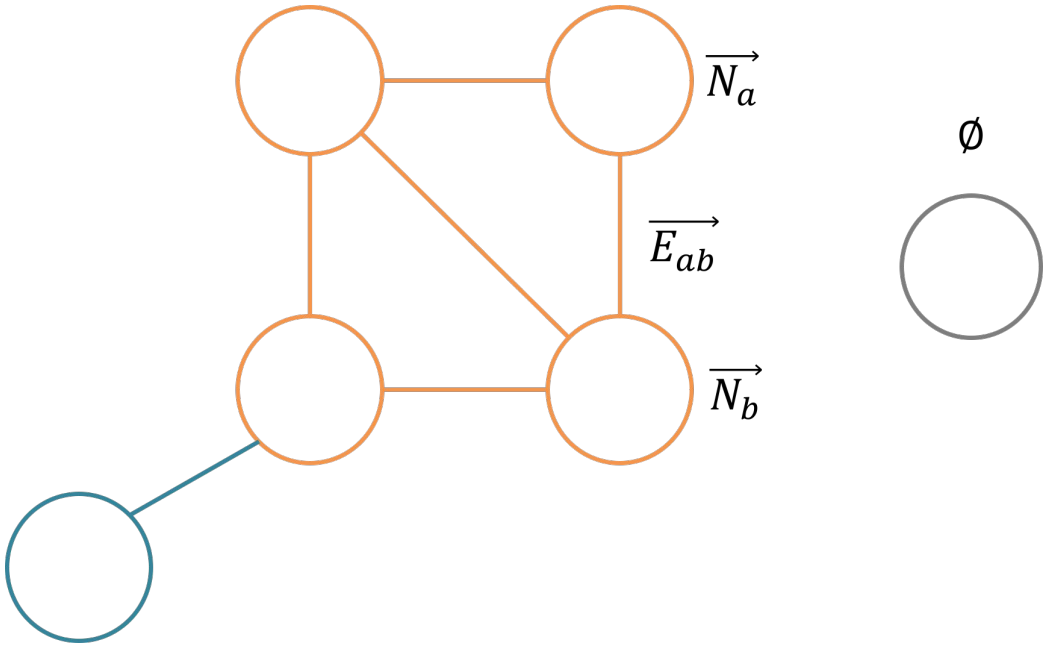


ARG

An ARG, G, is a directional graph with labels on its nodes and edge.

For instance, node here can be individual component of car, and the edge can be the distance between them.

In addition, we also include a null node for matching purpose.



Graph Matching

To summarize pattern from ARGs, we will first learn to compare two ARGs

- Find the largest common sub ARG (NP-Complete!)
- Align the two common sub ARG

Match criteria:

- If two nodes are very similar, we should match them PICS
- If two directed edges are very similar, we should match both ends

Nodes out of the common sub ARG (aka background node) should be matched to the null node of the other ARG.

Graph Matching - Problem Definition

We defined the problem of ARG matching in the following manner. Given two ARG, G and G', we want to find the match matrix M such that the following objective function is minimized:

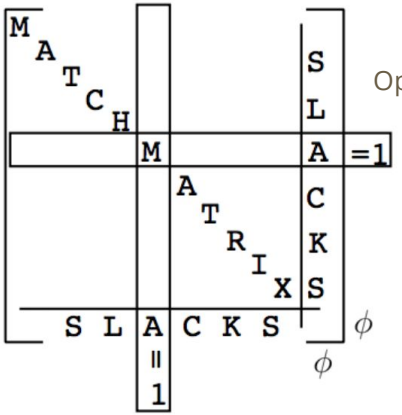
$$E(M) = -\frac{1}{2}\sum_{a=1}^{G}\sum_{i=1}^{G'}\sum_{b=1}^{G}\sum_{j=1}^{G'} M_{ai}M_{bj}C_{abij} - \alpha\sum_{a=1}^{G}\sum_{i=1}^{G'} M_{ai}C_{ai} \tag{1.1}$$

subject to a two way constraints: $\forall a \neq \phi$, $\sum_{i=1}^{G'} M_{ai} = 1$ and $\forall i \neq \phi$, $\sum_{i=1}^{G} M_{ai} = 1$:

Optimize by edge compatibility

Optimize by node compatibility

Compatibility is 0 for null node and any edge that does not exist or has null node on either end.



The probability of one node match to other node should sum to one. This is a two-way constraints.

Graph Matching - Gold and Rangarajan 1996

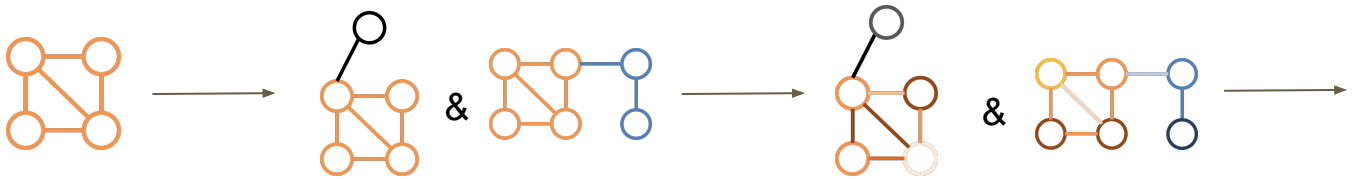
Gradient Search Algorithm:

Assign matches based on gradient:

$$Q_{ai} = -\frac{\partial E}{\partial M_{ai}}\bigg|_{M=M_0} = +\sum_{b=1}^{G}\sum_{j=1}^{G'} M_{bj}^0 C_{abij} + \alpha C_{ai} \qquad (1.5)$$

Initialize β to β_0, M_{ai} to a random sample from $U(0,1)$
Begin A**:** (Do A until $\beta \geq \beta_f$)
  Begin B**:** (Do B until M converges or # of iterations $> I_0$)
  $Q_{ai} \leftarrow \sum_{b=1}^{G}\sum_{j=1}^{G'} M_{bj}^0 C_{abij} + \alpha C_{ai}$
  $M_{ai}^0 \leftarrow exp(\beta Q_{qi})$
    Begin C**:** (Do C until M converges or # of iterations $> I_1$)
    Update M by normalizing across all rows:
    $M_{ai}^1 = \frac{M_{ai}^0}{\sum_{i=1}^{G'} M_{ai}^0}$ for all $a \neq \phi$
    Update M by normalizing across all columns:
    $M_{ai}^0 = \frac{M_{ai}^1}{\sum_{a=1}^{G} M_{ai}^1}$ for all $i \neq \phi$
    End C
  End B
$\beta \leftarrow \beta_r \beta$
End A
Perform Clean-up Heuristic

So how do we test this?



m = # correct match
l = # total match
n = # nodes in pattern

$$precision = \frac{m}{l} \tag{1.12}$$

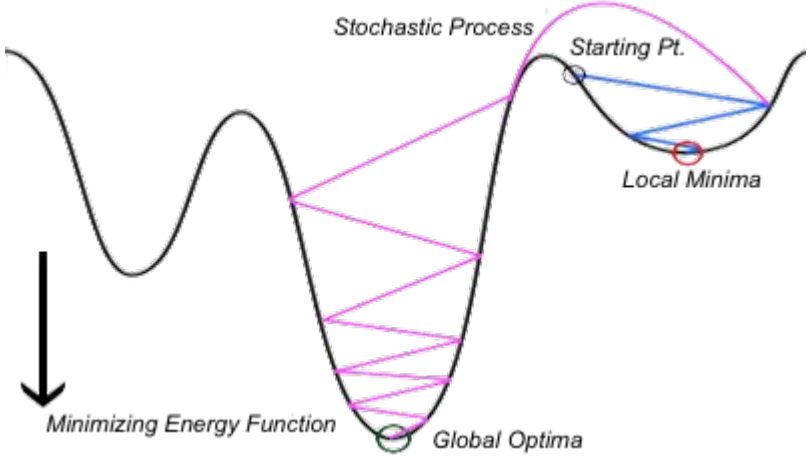
$$recall = \frac{m}{n} \tag{1.13}$$

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{1.14}$$

Graph Matching - Local Minima

During test, the Gold and Rangarajan algorithm will generate correct and even perfect match for majority of the test cases. However, for some of the test case, the match result is entirely wrong, which indicates that the graduated assignment process stuck in some local minima.

Our modification:



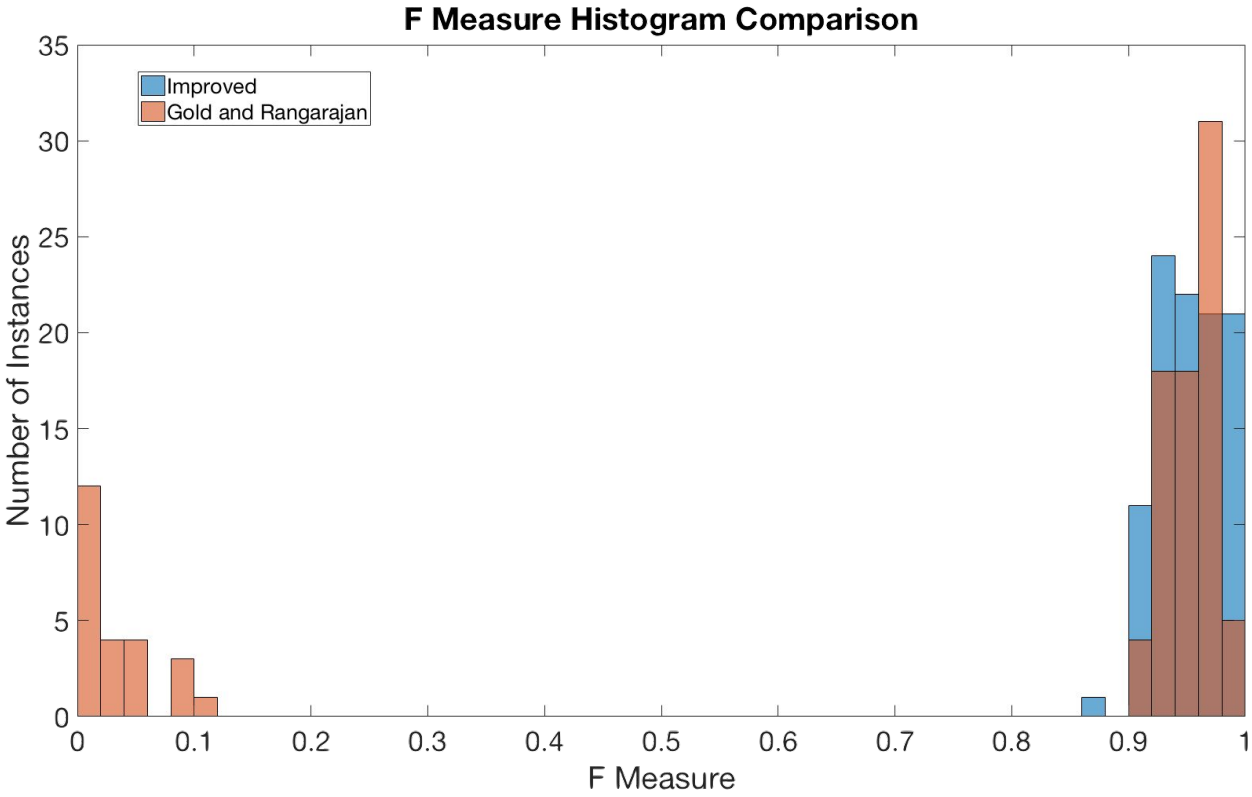
Graph Matching - Local Minima

Introducing stochastic process to the algorithm:

Add an uniform noise to the match result with a level depending on the graph size

Initialize β to β_0, M_{ai} to a random sample from $U(0,1)$
Begin A**:** (Do A until $\beta \geq \beta_f$)
  Begin B**:** (Do B until M converges or # of iterations $> I_0$)
  $M^0_{ai} \leftarrow M^0_{ai} + \frac{\tau * U(-1,1)}{|G|}$
  $Q_{ai} \leftarrow \sum_{b=1}^{G} \sum_{j=1}^{G'} M^0_{bj} C_{abij} + \alpha C_{ai}$
  $M^0_{ai} \leftarrow exp(\beta Q_{qi})$
    Begin C**:** (Do C until M converges or # of iterations $> I_1$)
    Update M by normalizing across all rows:
    $M^1_{ai} = \frac{M^0_{ai}}{\sum_{i=1}^{G'} M^0_{ai}}$ for all $a \neq \phi$
    Update M by normalizing across all columns:
    $M^0_{ai} = \frac{M^1_{ai}}{\sum_{a=1}^{G} M^1_{ai}}$ for all $i \neq \phi$
    End C
  End B
$\beta \leftarrow \beta_r \beta$
End A
Perform Clean-up Heuristic

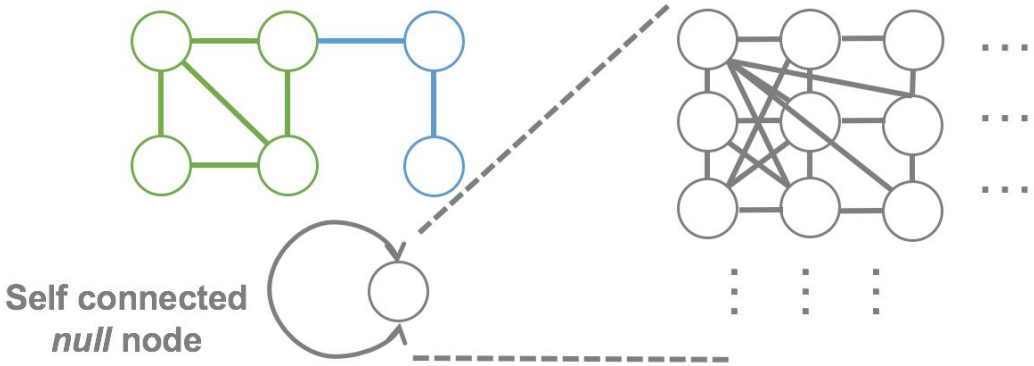
Graph Matching - Local Minima



Graph Matching - High Recall + Low Precision

Even though we get high recall rate in a test, we also get low precision rate, which means the algorithm can generate correct match but also force many of the background nodes matching to each other.

Null node network that can match to any pattern:



Graph Matching - High Recall + Low Precision

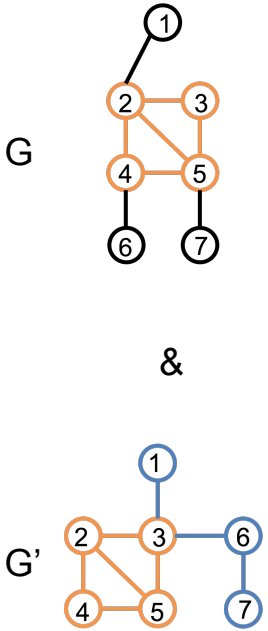
In practice, since we don't normalize null node matches, there is no need to build an actual network. Instead, we just create a self connected edge to the null node (sort of like recursion), and give definition for node and edges involving null node:

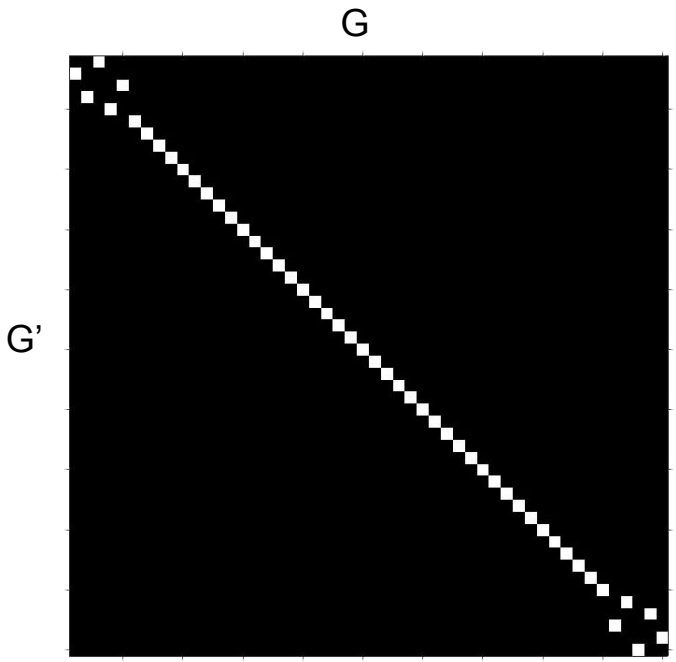
$$C_{ai} = \begin{cases} 0 & a, i = \phi \\ c_N(\overrightarrow{N_a}, \overrightarrow{N_i}) & a, i \neq \phi \\ p \text{ percentile of } [C_{1i}, C_{2i}, ...C_{|G|-1i}] & a = \phi \\ p \text{ percentile of } [C_{a1}, C_{a2}, ...C_{a|G'|-1}] & i = \phi \end{cases} \tag{1.16}$$

$$C_{abij} = \begin{cases} c_E(\overrightarrow{E_{ab}}, \overrightarrow{E_{ij}}) & a, b, i, j \neq \phi \\ p \text{ percentile of } \{C_{abij} | a, b, i, j \neq \phi\} & a, b \neq \phi \bigcap i, j = \phi \\ p \text{ percentile of } \{C_{abij} | a, b, i, j \neq \phi\} & a, b = \phi \bigcap i, j \neq \phi \\ 0 & otherwise \end{cases} \tag{1.17}$$

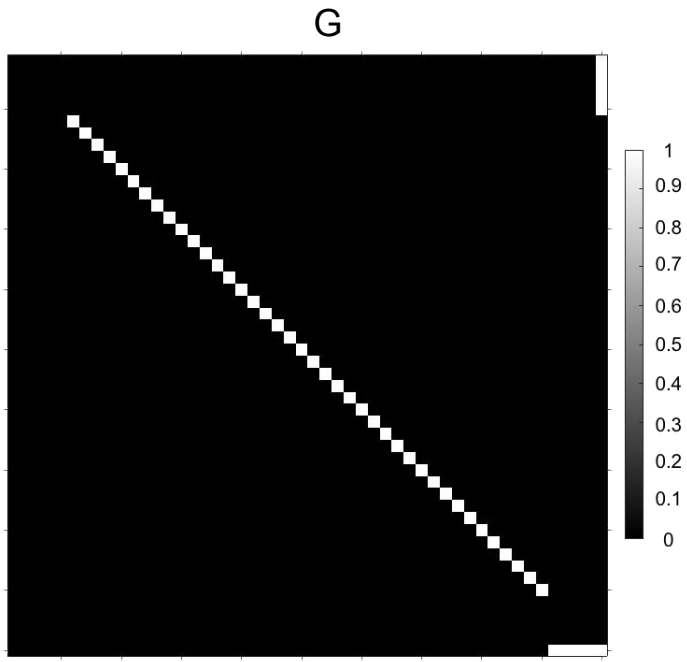
Percentile of the general population instead of 0

Graph Matching - High Recall + Low Precision





Before

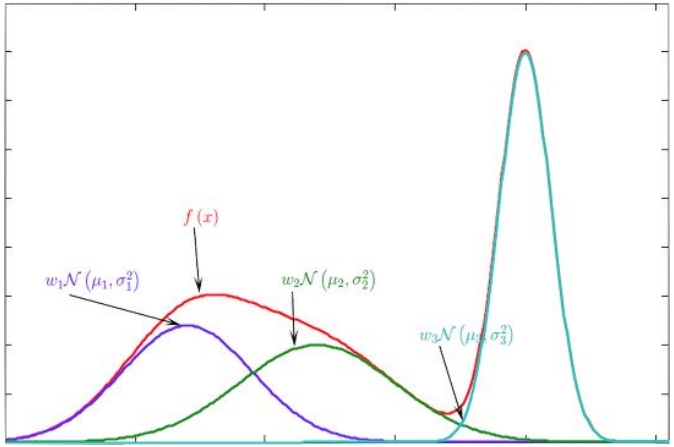


After

Pattern Learning

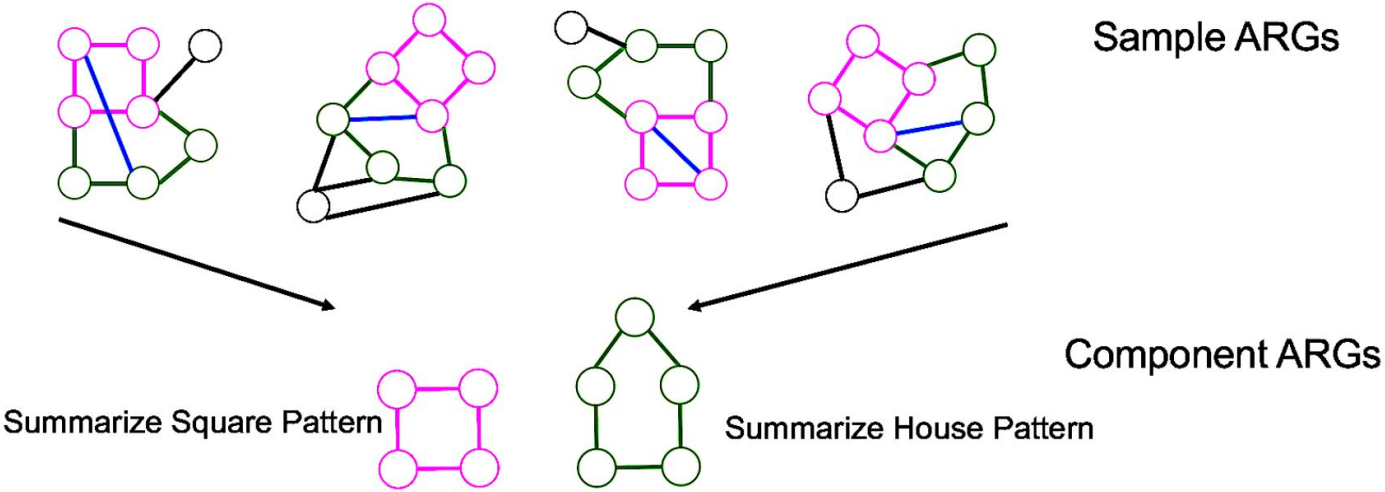
We extract the common pattern from a set of ARGs, and the extracted information can then be used to summarize the given ARGs and predict if a new ARG contains the common pattern we learned.

Here, we utilize a probabilistic parametric model using a set of components to represent the share/common pattern among ARGs. Similar to how the three normal distribution make up the data distribution generated by f(x):

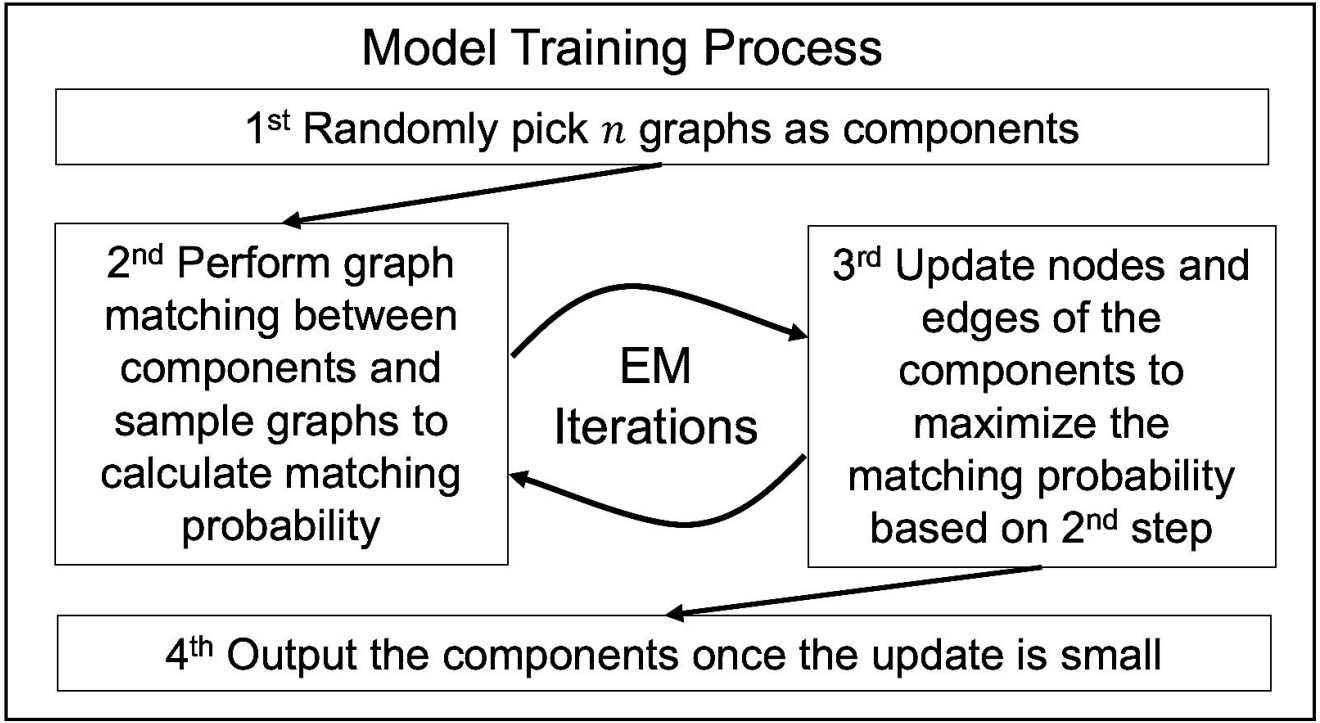


Pattern Learning - Problem Definition

Given a set of sample ARG $\{G_s\}_{s=1}^{S}$, our problem will be inferring the parameters in Z including the number of components W, weight for each component α, mean for each node/edge $\overrightarrow{N^w}/\overrightarrow{E^w}$, and the covariance associated with them Σ.



Pattern Learning - Expectation Maximization Algorithm



Pattern Learning - Estimation Step

In this step, we do graph matching between component ARGs and sample ARGs and get result M.

Based on the matching result, we calculate the likelihood of one particular component matching to a particular sample:

Score contribution by node

$$\xi(G_s|\Phi_w) = \sum_{a=1}^{\widehat{G_s}} \sum_{i=1}^{\Phi_w} M_{ai}^{sw} C_{ai}^{sw} + \sum_{a=1}^{\widehat{G_s}} \sum_{b=1}^{\widehat{G_s}} \sum_{i=1}^{\Phi_w} \sum_{j=1}^{\Phi_w} M_{ai}^{sw} M_{bj}^{sw} C_{abij}^{sw} \tag{2.7}$$

Score contribution by edge

$$P(G_s = \Phi_w) = \frac{\xi(G_s|\Phi_w)}{\sum_{t=1}^{W} \xi(G_s|\Phi_t)} \tag{2.8}$$

Probability of sample matching to one component is normalized by the score matching to all components

We can also calculate a score of one particular sample matching to our model Z:

$$f(G|Z) = \sum_{w=1}^{W} \alpha_w \xi(G|\Phi_w) \tag{2.9}$$

Weighted sum of component-sample matching score.

Pattern Learning - Maximization Step

1.Update Component Weight:

$$\alpha_w = \frac{\sum_{s=1}^{S} P(G_s = \Phi_w)}{S}$$

2.Update Component Node Weight:

$$\beta_a^w = \frac{\sum_{s=1}^{S} \sum_{i=1}^{\widehat{G_s}} M_{ia}^{sw} P(G_s = \Phi_w)}{\sum_{s=1}^{S} |\widehat{G_s}| P(G_s = \Phi_w)}$$

3.Update Mean for Component Node:

$$\overrightarrow{N_a^w} = \frac{\sum_{s=1}^{S} \sum_{i=1}^{\widehat{G_s}} \overrightarrow{N_i^s} M_{ia}^{sw} P(G_s = \Phi_w)}{\sum_{s=1}^{S} \sum_{i=1}^{\widehat{G_s}} M_{ia}^{sw} P(G_s = \Phi_w)}$$

4.Update Covariance for Component Node:

$$\Sigma_a^w = \frac{\sum_{s=1}^{S} \sum_{i=1}^{\widehat{G_s}} \overrightarrow{x_i^s} \overrightarrow{x_i^s}^T M_{ia}^{sw} P(G_s = \Phi_w)}{\sum_{s=1}^{S} \sum_{i=1}^{\widehat{G_s}} M_{ia}^{sw} P(G_s = \Phi_w)}$$

$$\overrightarrow{x_i^s} = \overrightarrow{N_i^s} - \overrightarrow{N_a^w}$$

5.Update Mean for Component Edge:

$$\overrightarrow{E_{ab}^w} = \frac{\sum_{s=1}^{S} \sum_{i=1}^{\widehat{G_s}} \sum_{j=1}^{\widehat{G_s}} \overrightarrow{E_{ij}^s} M_{ia}^{sw} M_{bj}^{sw} P(G_s = \Phi_w)}{\sum_{s=1}^{S} \sum_{i=1}^{\widehat{G_s}} \sum_{j=1}^{\widehat{G_s}} M_{ia}^{sw} M_{bj}^{sw} P(G_s = \Phi_w)}$$

6.Update Covariance for Component Edge:

$$\Sigma_{ab}^w = \frac{\sum_{s=1}^{S} \sum_{i=1}^{\widehat{G_s}} \sum_{j=1}^{\widehat{G_s}} \overrightarrow{z_{ij}^s} \overrightarrow{z_{ij}^s}^T M_{ia}^{sw} M_{bj}^{sw} P(G_s = \Phi_w)}{\sum_{s=1}^{S} \sum_{i=1}^{\widehat{G_s}} \sum_{j=1}^{\widehat{G_s}} M_{ia}^{sw} M_{bj}^{sw} P(G_s = \Phi_w)}$$

$$\overrightarrow{z_{ij}^s} = \overrightarrow{E_{ij}^s} - \overrightarrow{E_{ab}^w}$$

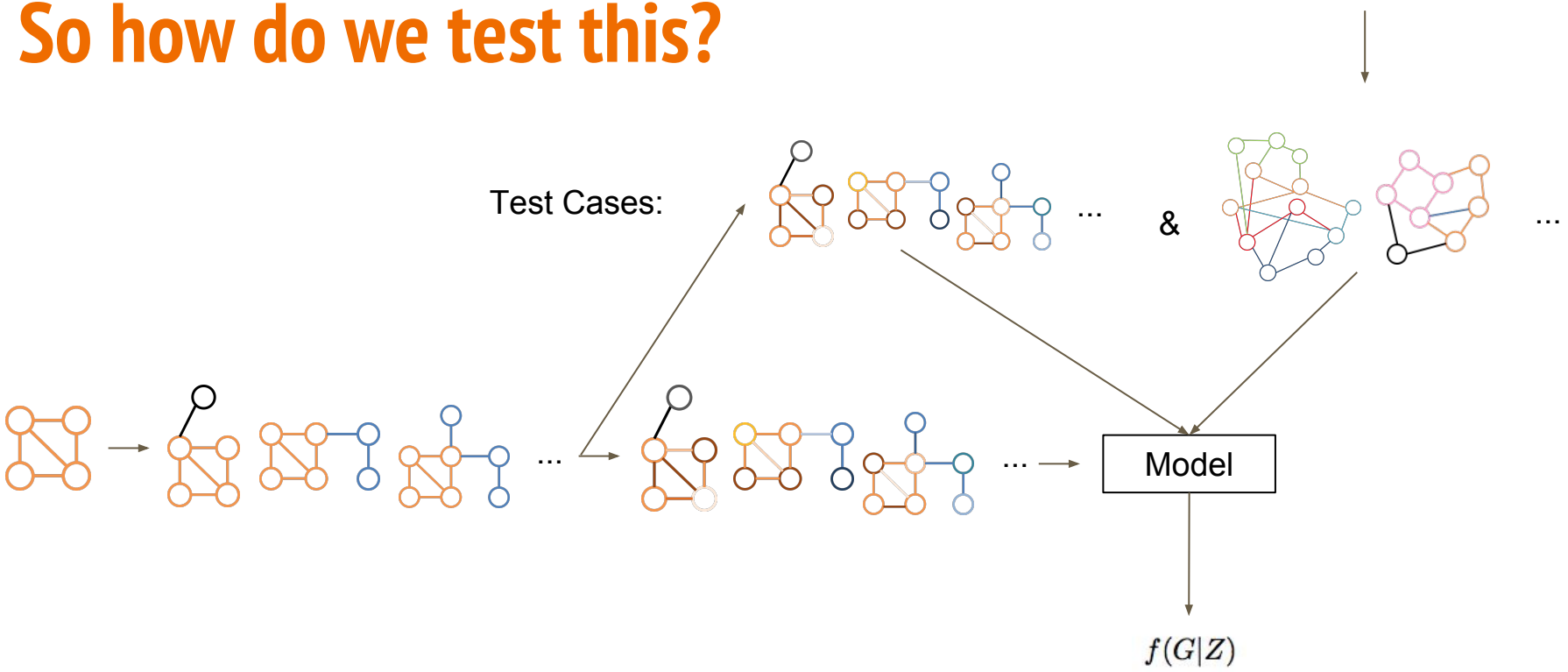
7.Delete Redundant Node:

$$\beta_a^w < 1 - 0.85^n$$

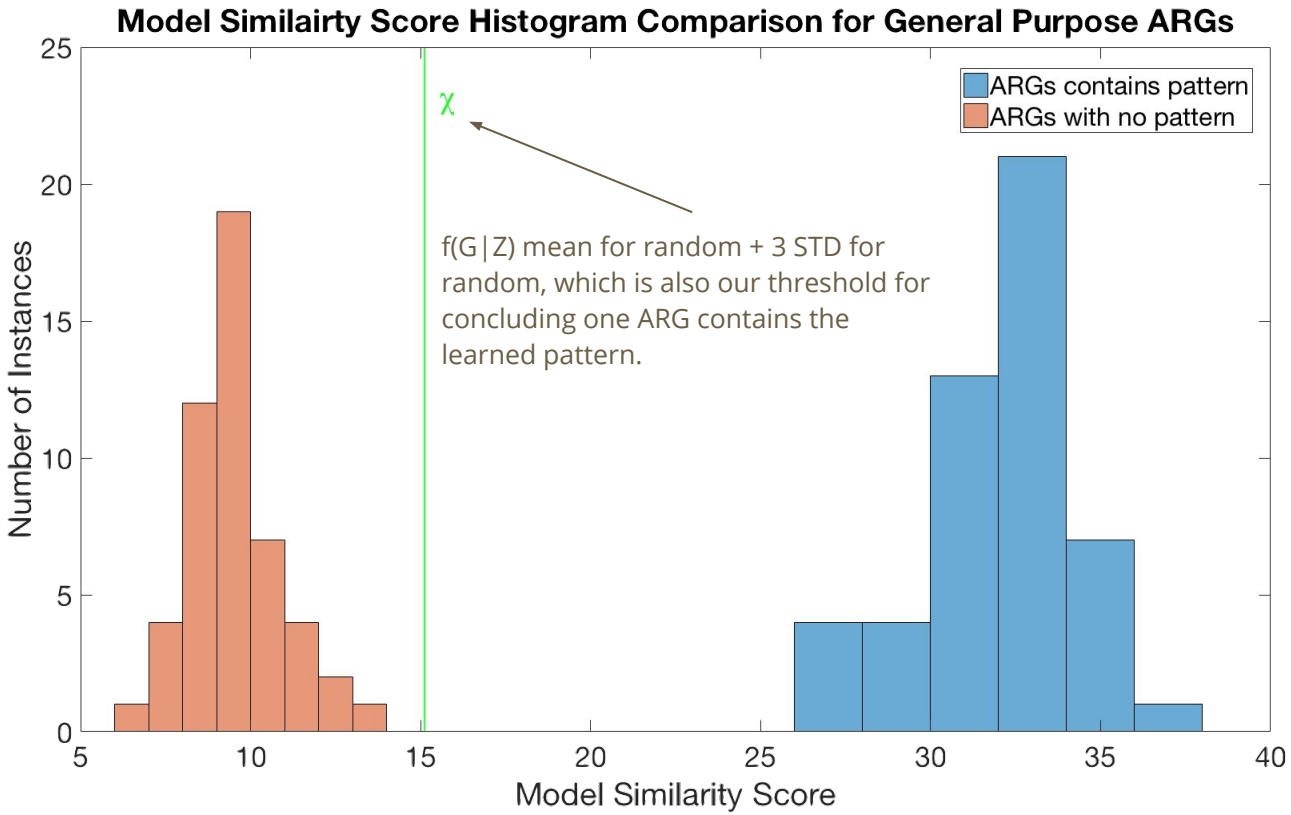
8.Decide if the model converge:

$$\sum_{w=1}^{W} \alpha_w^{(0)} < \iota$$

So how do we test this?



Pattern Learning - Result

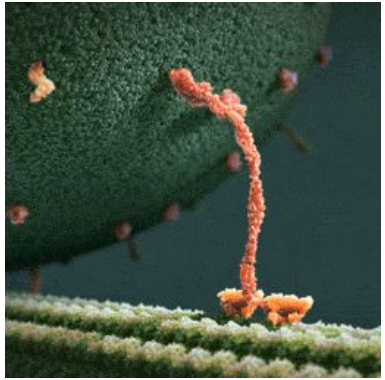


Protein Modeling

Proteins are macromolecules responsible for nearly every task of cellular life.

They are 3D structures consisting of amino acid sequences translated from genes and interact with each other to carry out essential functions, such as catalyzing metabolic reaction, transport molecules, respond to stimuli, and so on.

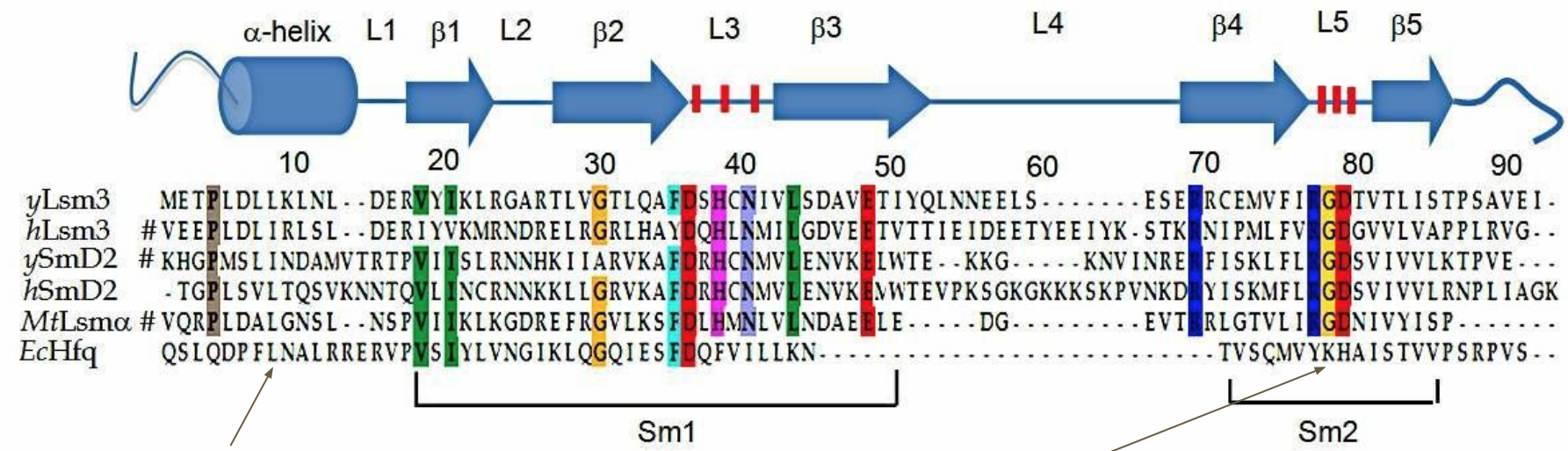
Such interactions often happen at sites (e.g. protein domains) that are conserved between proteins across species. By recombining and rearranging these domains as proteins' basic building block, molecular evolution is able to create proteins with different functions.



kinesin motor protein

Protein Modeling - Conventional Approach

Conventional approach relies on sequential data:

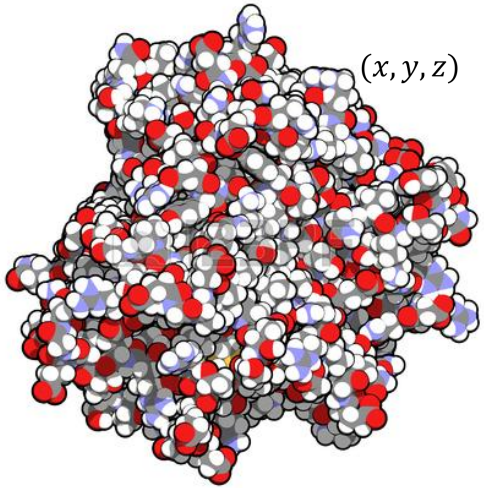


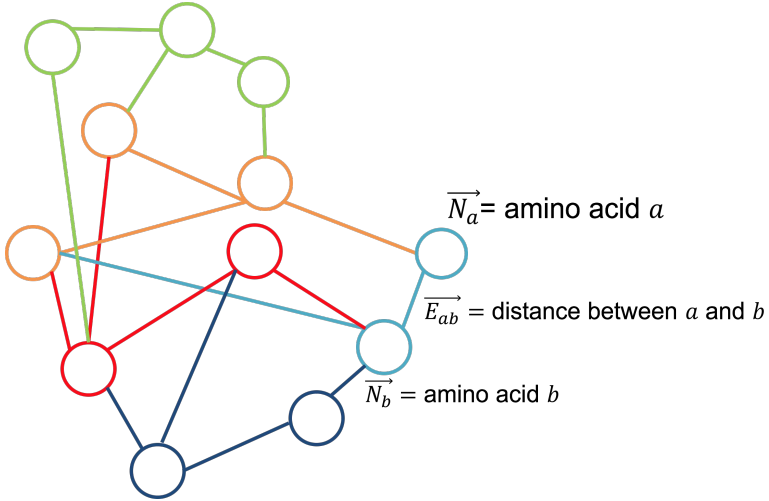
One letter identification for amino acid

Not every sequence is the same

Protein Modeling - Our Approach

We turn 3D crystallographic data to graph, and apply our learning model



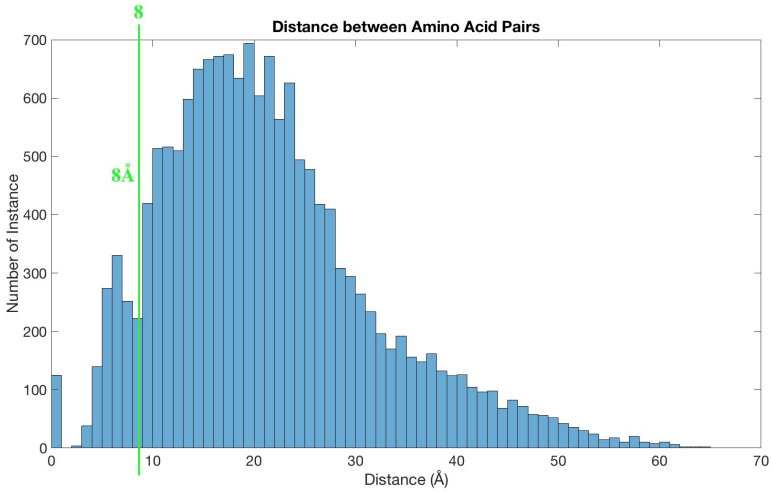


Protein Modeling - Edge for Protein ARG

For the edge, we assign it a single scalar which is the euclidian distance of two amino acid:

$$\overrightarrow{E_{ab}} = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + (z_a - z_b)^2} \tag{3.1}$$

We also have a distance cutoff since not all amino acids are relevant to each other:



Protein Modeling - Node for Protein ARG

BLOSUM Matrix

- Log odd-ratio of how likely two amino acids are interchangeable
- We give a single index for each node to represent
- Use BLOSUM matrix to measure the compatibility
- How do we update the index?
- Consider local context?

	A	B	C	D	E	F	G	H	I	K	L	M	N	P	Q	R	S	T	V	W	X	Y	Z
A	4	-2	0	-2	-1	-2	0	-2	-1	-1	-1	-1	-2	-1	-1	-1	1	0	0	-3	-1	-2	-1
B	-2	6	-3	6	2	-3	-1	-1	-3	-1	-4	-3	1	-1	0	-2	0	-1	-3	-4	-1	-3	2
C	0	-3	9	-3	-4	-2	-3	-3	-1	-3	-1	-1	-3	-3	-3	-3	-1	-1	-1	-2	-1	-2	-4
D	-2	6	-3	6	2	-3	-1	-1	-3	-1	-4	-3	1	-1	0	-2	0	-1	-3	-4	-1	-3	2
E	-1	2	-4	2	5	-3	-2	0	-3	1	-3	-2	0	-1	2	0	0	-1	-2	-3	-1	-2	5
F	-2	-3	-2	-3	-3	6	-3	-1	0	-3	0	0	-3	-4	-3	-3	-2	-2	-1	1	-1	3	-3
G	0	-1	-3	-1	-2	-3	6	-2	-4	-2	-4	-3	0	-2	-2	-2	0	-2	-3	-2	-1	-3	-2
H	-2	-1	-3	-1	0	-1	-2	8	-3	-1	-3	-2	1	-2	0	0	-1	-2	-3	-2	-1	2	0
I	-1	-3	-1	-3	-3	0	-4	-3	4	-3	2	1	-3	-3	-3	-3	-2	-1	3	-3	-1	-1	-3
K	-1	-1	-3	-1	1	-3	-2	-1	-3	5	-2	-1	0	-1	1	2	0	-1	-2	-3	-1	-2	1
L	-1	-4	-1	-4	-3	0	-4	-3	2	-2	4	2	-3	-3	-2	-2	-2	-1	1	-2	-1	-1	-3
M	-1	-3	-1	-3	-2	0	-3	-2	1	-1	2	5	-2	-2	0	-1	-1	-1	1	-1	-1	-1	-2
N	-2	1	-3	1	0	-3	0	1	-3	0	-3	-2	6	-2	0	0	1	0	-3	-4	-1	-2	0
P	-1	-1	-3	-1	-1	-4	-2	-2	-3	-1	-3	-2	-2	7	-1	-2	-1	-1	-2	-4	-1	-3	-1
Q	-1	0	-3	0	2	-3	-2	0	-3	1	-2	0	0	-1	5	1	0	-1	-2	-2	-1	-1	2
R	-1	-2	-3	-2	0	-3	-2	0	-3	2	-2	-1	0	-2	1	5	-1	-1	-3	-3	-1	-2	0
S	1	0	-1	0	0	-2	0	-1	-2	0	-2	-1	1	-1	0	-1	4	1	-2	-3	-1	-2	0
T	0	-1	-1	-1	-1	-2	-2	-2	-1	-1	-1	-1	0	-1	-1	-1	1	5	0	-2	-1	-2	-1
V	0	-3	-1	-3	-2	-1	-3	-3	3	-2	1	1	-3	-2	-2	-3	-2	0	4	-3	-1	-1	-2
W	-3	-4	-2	-4	-3	1	-2	-2	-3	-3	-2	-1	-4	-4	-2	-3	-3	-2	-3	11	-1	2	-3
X	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
Y	-2	-3	-2	-3	-2	3	-3	2	-1	-2	-1	-1	-2	-3	-1	-2	-2	-2	-1	2	-1	7	-2
Z	-1	2	-4	2	5	-3	-2	0	-3	1	-3	-2	0	-1	2	0	0	-1	-2	-3	-1	-2	5

Protein Modeling - Node for Protein ARG

Local Substitution Vector

- Sample node will still be an index for the amino acid, while the model node will have a local substitution vector where each index representing how likely this node can match to each amino acids

When initializing the model protein ARG, we set the local substitution vector based on the BLOSUM matrix, so $\overrightarrow{N_i^w} = B[AA_i, :]$, where AA_i is the amino acid index for node i. To update the substitution vector with locality, we first calculate a "weighted" frequency vector $\overrightarrow{F_i^w}$:

We still initialize from BLOSUM

We count the frequency of this node matching to different amino acids in estimation step.

$$\overrightarrow{\zeta_i^w}[z] = \sum_{s=1}^{S} \sum_{a}^{\{a|\overrightarrow{N_a^s}=z\}} M_{ai}^{sw} P(G_s = \Phi_w) \qquad \forall z = 1, 2, 3...20 \qquad (3.6)$$

$$\overrightarrow{F_i^w}[z] = \frac{\overrightarrow{\zeta_i^w}[z]}{\sum_{x=1}^{20} \overrightarrow{\zeta_i^w}[x]} \qquad \forall z = 1, 2, 3...20 \qquad (3.7)$$

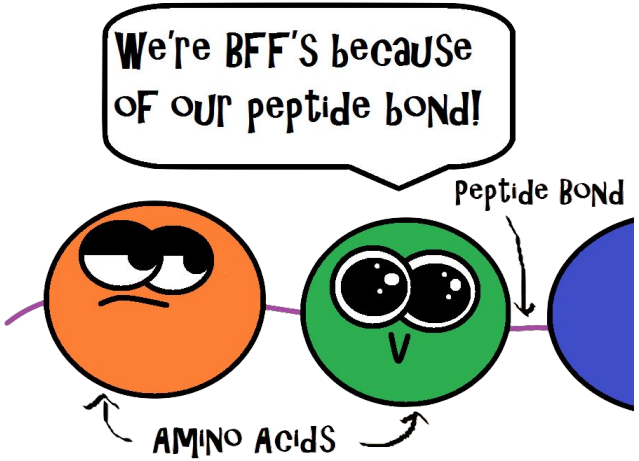
We normalize the count!

Once we calculate $\overrightarrow{F_i^w}$, we can calculate the log odd-ratio with an amino acid background frequency $\overrightarrow{F_B}$ that we can pre-defined or calculate based on the sample protein ARGs:

We calculate the log odd ratio similar to calculating BLOSUM matrix

$$\overrightarrow{N_i^w}[z] = log(\frac{\overrightarrow{F_i^w}[z]}{\overrightarrow{F_B}[z]}) \qquad \forall z = 1, 2, 3...20 \qquad (3.8)$$

Protein Modeling - Protein Backbone Encoding



$$E(M) = -\frac{1}{2}\sum_{a=1}^{G}\sum_{i=1}^{G'}\sum_{b=1}^{G}\sum_{j=1}^{G'} M_{ai}M_{bj}C_{abij} - \alpha\sum_{a=1}^{G}\sum_{i=1}^{G'} M_{ai}C_{ai} - \beta\sum_{a=1}^{G}\sum_{i=1}^{G'} M_{a-1i-1}M_{ai}M_{a+1i+1} \tag{3.9}$$

β controls the rigidity

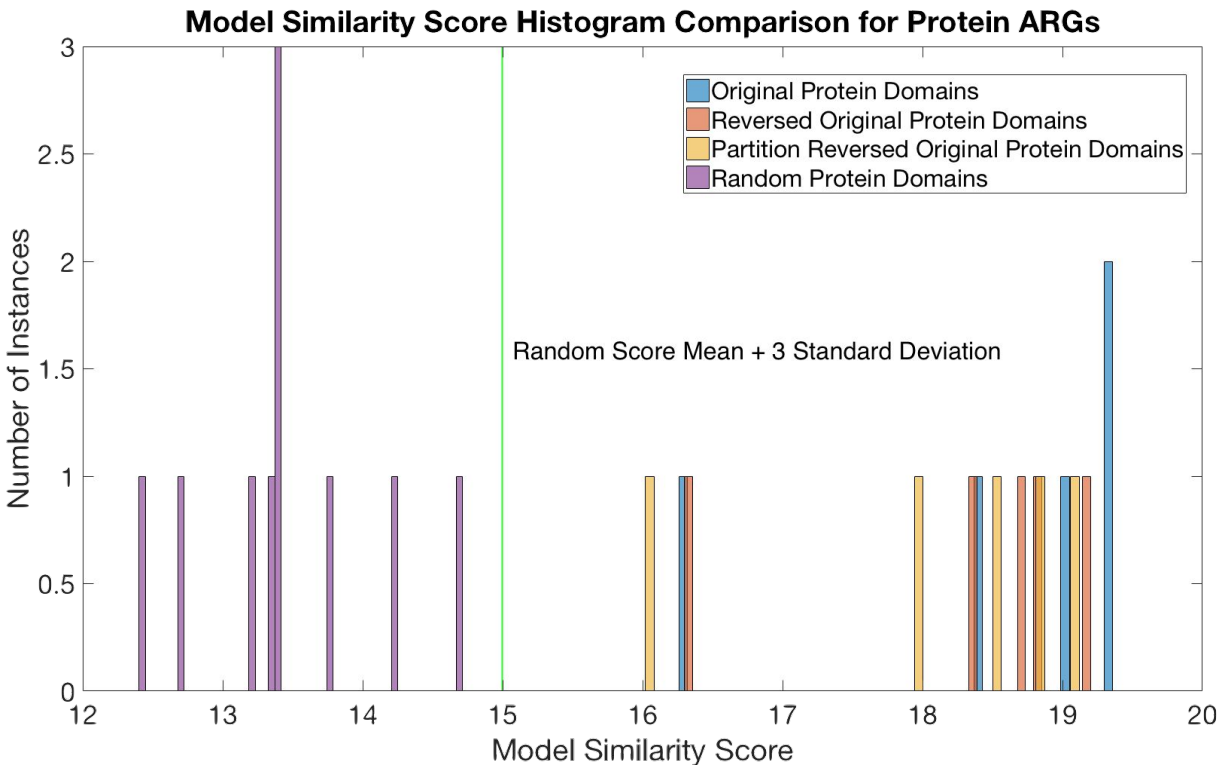
Three consecutive amino acids in one protein should match to three consecutive amino acids in another protein as much as possible

Protein Modeling - Proof of Concept

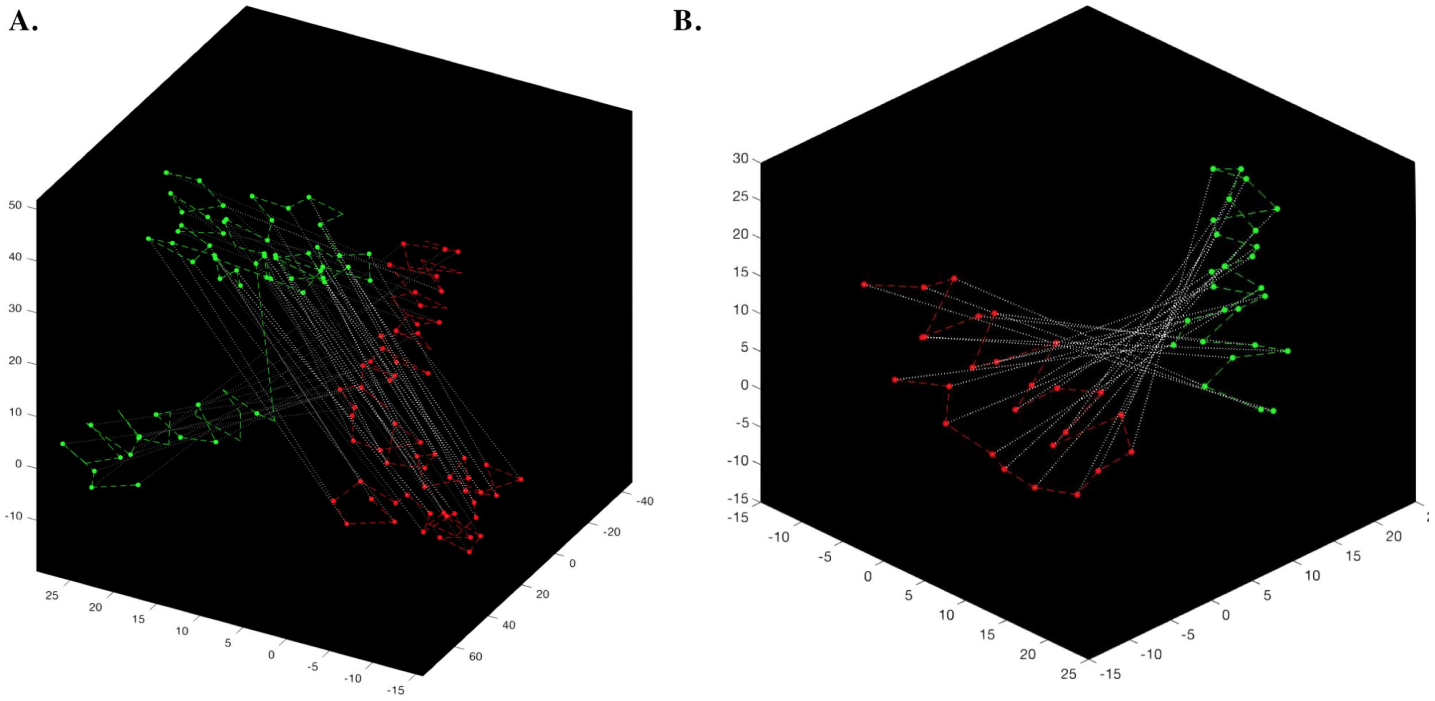
We did a proof of concept by training a model with five partial proteins sequence that contains part of the CH1 domain:

Training Sample	Reversed Test	Partition Reversed Test	Random Test
#$@ABCDEF	FEDCBA@$#	EF|CD|AB|@$#	@*#$*!
@^#ABCXEF	FEXCBA#^@	EF|CX|AB|#^@	&@&$(@
#@*#BCDEF	FEDCB#*@#	EF|CD|B|#*@#	@(#*%(!
@#ABCDYF	FYDCBA#@	YF|CD|AB|#@	(&*%@#
*#&ABCDE	EDCBA&#*	E|CD|AB|&#*	...

Protein Modeling - Proof of Concept



Protein Modeling - Proof of Concept



Key Contribution

- Introduce a stochastic process in the gradient search to effectively avoid local minima
- Add a null node network to each ARG to prevent force matches among background nodes
- Introduce an additional term in our objective function representing the protein backbone
- Create a local substitution vector to model the similarity between two different amino acids based on their specific local environment

Final Notes

I want to thank Porf. Hong for his continuous support and inspiration.

I also want to thank everyone here who makes Brandeis a warm and welcoming community.

Last but not least, I also want to give credits to my family, friends, and family who help me procrastinate.

Part of the grant also comes from Jerome A. Schiff Fellowship.

Q&A

Looking Forward - Graph Matching

More Efficient Implementation

- Parallel Computing
- Edge Representation
- GPU on Normalization
- Incorporate Constraints into Objective and Derive M Directly:

$$
\begin{aligned}
E(M) = & -\frac{1}{2}\sum_{a=1}^{G}\sum_{i=1}^{G'}\sum_{b=1}^{G}\sum_{j=1}^{G'} M_{ai}M_{bj}C_{abij} \\
& -\frac{1}{\beta}\sum_{a=1}^{G}\sum_{i=1}^{G'} M_{ai}(\log M_{ai} - 1) \\
& +\sum_{a=1}^{G}\mu_a(\sum_{i=1}^{G'} M_{ai} - 1) + \sum_{i=1}^{G;}\nu_a(\sum_{a=1}^{G} M_{ai} - 1)
\end{aligned} \tag{4.1}
$$

Looking Forward - Pattern Learning

Application in CV and NLP

- Model Relationship of Complex Scene

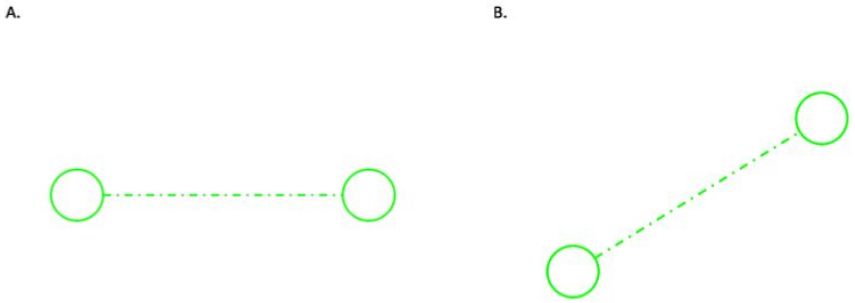


Figure 4.1: *A. Two objects (circles) have a relationship of a certain distance X. B. A rotated scene showing such relationship.*

- Model Dialogue Transitional Relationship (Q&A, Extension, etc)

Looking Forward - Protein Modeling

Protein as Documents and Amino Acid as Word

- Seq2Seq: protein structure prediction
- LSTM: predict protein biochemistry property