Comp 598: Assignment 2

Protein Structure and System Biology

Due on December 8th, 2016.

- To some extent, collaborations are allowed, but you must indicate the name of all collaborators (including instructors) on your answers. Uncredited collaborations will be penalized.
- Unless specified, all answers must be justified.
- Partial answers will receive credits.
- Answers should be submitted electronically to the instructor.

Exercise 1 (10 points) Explain why secondary structure prediction methods are more accurate at predicting α-helices than β-sheets (N.B.: Provide a detailed answer).

Exercise 2 (30 points) We aim to develop a simple method to predict protein secondary structures. We will restrict the scope of this work to the prediction of residues in α-helices (noted `H`) or coil regions (noted `C`). Several propensity scale have been proposed for α-helices. Here, we will use the scale proposed by C.N. Pace and J.M. Scholtz in `http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1299714/`. We will benchmark our techniques on the myoglobin: `http://www.rcsb.org/pdb/explore/explore.do?structureId=1mbn` (extract the primary and secondary structure from the Protein Data Bank record).

1. Implement an algorithm that assigns a secondary structure type to each residue of an input sequence from the propensity scale introduced above. The program will require users to input a protein sequence ω and a threshold value λ. Residues with a propensity value lower than λ will be predicted to belong to an α-helix secondary structure.

2. Implement a program that compare a "real" secondary structure with a predicted one, and calculate the true positive rate (TPR) and false positive rate (FPR), respectively defined as $TPR = TP/(TP + FN)$ and $FPR = FP/(FP + TN)$ where TP are True Positive, FN are False Negatives, FP are False Positives, and TN are True Negatives.

3. Use these programs to plot the receiver operating characteristic (ROC) curve and calculate the area under the curve (AUC) (`http://en.wikipedia.org/wiki/Receiver_operating_characteristic`). Hint: You will vary the threshold λ to determine the coordinate of the points delimitating the hull of the ROC curve.

4. Improve your α-helix predictor. You will incorporate in your algorithm, a signal from sequence neighbour residues. In particular, the propensity value associate to a residue will now be the average of the propensity values of all residues located 4 positions before or after the current index.

5. Repeat the procedure of the third item, compare and discuss the performance of the two versions of the predictor.

Exercise 3 (30 points) We will analyze the result of our MD simulation. We simulated a system for 2000 pico seconds with a mutated version of amylin. Your job is to create the RMSD graph for this simulation along with a short movie showing the protein in action. To do this, you will have to:

1. Read and execute the molecular dynamics tutorial available at `http://www.cs.mcgill.ca/~jeromew/docs/MD_tutorial.pdf`. A zip file including all files you will need for this tutorial is available at `http://amyloid.cs.mcgill.ca/MD.zip` (Note: You do not have to return any answer for this, but it will provide you the knowledge to complete this exercise).

2. Retrieve the files `npt-nopr.tpr` and `npt-nopr.trr` in the tutorial material (`http://amyloid.cs.mcgill.ca/MD.zip`). These two files store the result of the simulation trajectory of every atom.

3. Use the 2 commands on slide 8 of the tutorial powerpoint that was given to you (found on http://cs.mcgill.ca/ ms-maou/MD). The first command will create the RMSD graph, and the second will create the movie.

4. Generate a graph from a software that opens .xvg extensions

5. Open your molecule-movie.pdb file in PYMOL and click "File->save as->Movie->choose .avi or .mov

Exercise 4 (30 points) We want to implement a simplified version of the ISORANK algorithm (`http://www.pnas.org/content/105/35/12763`) to compare 2 protein-protein interaction (PPI) networks N_1 and N_2. Here, we will not consider sequence similarity. The functional similarity score R_{ij} between two nodes i and j is then equal to the network similarity score, and defined as:

$$R_{ij} = \sum_{u \in \mathcal{N}(i)} \sum_{v \in \mathcal{N}(j)} \frac{R_{uv}}{|\mathcal{N}(u)| \cdot |\mathcal{N}(v)|} \tag{1}$$

Where i is a node of the network N_1, and j a node of the network N_2. $\mathcal{N}(x)$ is the list of neighbours of a node x. To calculate the R_{ij}, we represent the system of equations given by Equation 1 as a product of matrices $R = A \cdot R$, where:

$$R = \begin{pmatrix} R_{1,1} \\ R_{1,2} \\ \vdots \\ R_{m,n} \end{pmatrix} \quad (2) \quad A = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \dots & a_{1,m \cdot n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \dots & a_{2,m \cdot n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m \cdot n,1} & a_{m \cdot n,2} & a_{m \cdot n,3} & \dots & a_{m \cdot n,m \cdot n} \end{pmatrix}, \text{ where } a_{u,v} = \frac{1}{|\mathcal{N}(u)| \cdot |\mathcal{N}(v)|} \tag{3}$$

Then, we notice that solving this system of equations is equivalent to calculating the eigenvector R.

Finally, once the values of the R_{ij} have been determined, we determine the network alignment using a greedy approach. We align the node i and j with the largest R_{ij} and remove i and j from the list of available nodes. Then, we find the next largest value R_{xy} and align x with y. We iterate this process until no other nodes are available, or until all remaining values R_{ij} are below a threshold λ.

1. Download the files `N1.txt`, `N2.txt`, and `N3.txt`, storing networks N_1, N_2, and N_3 at `http://www.cs.mcgill.ca/~jeromew/docs/HW2/`. Each row stores an edge of the network, which are indicated by their identifiers separated by a space character (you can ignore the sign of the interaction if it has one).

2. Implement the proposed algorithm. You can use existing libraries to solve the eigenvector problem (e.g. `numpy.linalg` in Python). Your program must take as an input 2 networks A and B and return a list of values "$a_i\ b_j$", where a_i is the node of A aligned with node b_j of B. Each row must contain one and only one alignment of nodes, themselves separated by a space character. Note: You must specify the programming language used to implement this program, and the command line used to run it.

3. Use your program to align N_1 with N_2, N_1 with N_3, and N_2 with N_3. Print the common subgraphs. Which networks are the most similar? Justify your answer.

4. Let us assume now that the edges are weighted (e.g. the weight could represent the strength of the interaction). Propose modification of the algorithm that will align nodes connected by an edge with similar weight (Note: We do not ask you to implement this algorithm).