Melroy Saldanha
saldanha@stanford.edu
CS 273 Project Report

Clustering of Proteins

Introduction

Numerous genome-sequencing projects have led to a huge growth in the size of protein databases. Manual annotation of the sequences found in these databases is expensive and not very feasible. Thus, there is a need for developing reliable algorithms that automate the functional classification of these sequences and the identification of protein families. Most of the methods that are currently used in practice employ evolutionary relationships between sequences to predict functional properties. Clustering of proteins is one such method for determining evolutionary relationships between proteins and thereby inferring functional properties. To perform a clustering on the proteins, we need an appropriate measure of distance between two protein sequences so that we have a distance matrix for the clustering algorithm. Since evolutionary relationships deal with how similar two organisms are, it is ideal to use the alignment score between two protein sequences as the distance measure between them because the alignments score itself implies how similar two sequences are. In fact it is widely accepted as a basis of functional assignment that proteins with high sequence similarity share a common evolutionary history (i.e. a common ancestor).

All of the alignment scores, i.e. the distances, between every two sequences were generated using CLUSTALW, which is a global multiple sequence alignment tool that works for both DNA and protein sequences. CLUSTALW takes in two sequences as an input and outputs the alignment score along with the alignment. For this project, we focus on determining the effects of picking different clustering algorithms and the effects of choosing different methods within a clustering algorithm on the results. The two clustering algorithms that seemed interesting to work on for this purpose were hierarchical clustering and the neighbor joining method, both of which have their advantages and disadvantages.

Hierarchical Clustering

Algorithm:
Let N be the number of protein sequences found in our dataset. Now our distance matrix will be an NxN matrix that's initially filled with the pair-wise alignment scores between every two sequences, as calculated by CLUSTALW. The algorithm begins by assigning every protein sequence in the dataset to its own cluster, thus leaving us with N clusters initially. Here the distance between

any two clusters is equal to the distances between the protein sequences found in each cluster. Since we are using alignment scores as the distances, the higher the alignment score, the closer in distance the two clusters are. The algorithm sorts all the alignment scores (distances) initially in decreasing order, using merge sort, and stores it in an array 'A'.

Now at each step, we look for the pair of clusters that have the smallest distance between them, which is the pair with the largest alignment score in the distance matrix. So in other words, we find the first element in array A whose clusters haven't already been merged in a previous step. We now merge the two clusters, whose element was found in array A, into a single new cluster, leaving us with one less cluster. We then re-compute the distances between this new cluster and every remaining cluster and update the distance matrix. We also add these new distances to the array A, in sorted order. We repeat this process until there is only one cluster left that contains all the sequences. The output is a rooted tree showing how each protein clusters at each successive level. We can cut off the tree at any level to generate a subset of clusters.

The re-computing of the distances, in the algorithm above, can be done in a number of ways, which is what distinguishes the different types of hierarchical clustering methods. The three hierarchical methods used in this project were the single-linkage method, the average-linkage method and the complete-linkage method.

Let Ci and Cj be the two clusters that were merged into the single cluster Cij for each of the following methods,

Single-Linkage method:
Now for every remaining cluster Ck in the distance matrix, we look at the distances between Ci & Ck and the distances between Cj & Ck, as given in the distance matrix, and we store the minimum of the two distances as the distance between Cij and Ck. This implies the distance between two clusters is the shortest distance from any member of one cluster to any member of the other cluster. Also since we store the minimum of the two distances, we are actually looking for the higher alignment score among Ci & Ck and Cj & Ck to store as the new distance between Cij and Ck.

Average-Linkage method:
Now for every remaining cluster Ck in the distance matrix, we calculate the distance between Cij and Ck using the following formula:

$$D(Cij.\ Ck) = \frac{|Ci| * D(Ci,Ck) + |Cj| * D(Cj,Ck)}{|Ci| + |Cj|}$$

So in other words we are actually calculating the average distance between any member of one cluster to any member of the other cluster.

Complete-Linkage method:
Now for every remaining cluster Ck in the distance matrix, we look at the distances between Ci & Ck and the distances between Cj & Ck, as given in the distance matrix, and we store the maximum of the two distances as the distance between Cij and Ck. This implies the distance between two clusters is the longest distance from any member of one cluster to any member of the other cluster. Also since we store the maximum of the two distances, we are actually looking for the lower alignment score among Ci & Ck and Cj & Ck to store as the new distance between Cij and Ck.

Complexity:
The initial merge sort takes $O(N^2 \log N)$ for sorting N^2 elements. During each iteration, merging the two clusters takes O(1) time while adding a new distance to the array A takes O(logN). Since we have to find the distance from each new cluster to every other cluster, the total time for adding every new distance to the array takes O(N logN). Finally these iterations runs N times until all clusters have been merged, which implies a total running time of O(NlogN * N). So this entire algorithm takes $O(N^2 \log N + N \log N * N)$ which equals $O(N^2 \log N)$.

Neighbor Joining Method

Algorithm:
Let N be the number of protein sequences found in our dataset. Now our distance matrix will be identical to the one used in hierarchical clustering, that is an NxN matrix that's initially filled with the pair-wise alignment scores between every two sequences, as calculated by CLUSTALW. However, here we initialize each sequence to be a leaf of a tree T. Let dij be the value found between pair i, j in the distance matrix. Now let's define the terms Dij and ri, such that

$$D_{ij} = d_{ij} - (r_i + r_j) \quad \text{and} \quad r_i = \frac{1}{|T| - 2} \sum d_{ik}.$$

At each step, we first calculate ri and Dij for all i and j. We then pick a pair i, j in T for which Dij is minimal. Since we are working with alignment scores, we actually look for the Dij that has the largest score (largest value). Next we assign a new node k and set

$$d_{km} = (d_{im} + d_{jm} - d_{ij})/2, \text{ for all m in T.}$$

Finally we remove the leaves i and j from the tree T and add the node k to it. We repeat this process until two leaves are remaining and we join these two leaves. This method guarantees that Dij is minimal for leaves i, j if they are neighboring leaves. The output for this algorithm is an unrooted tree with showing how each protein clusters with each other.

Complexity:

At each step, we reevaluate the values ri and Dij among all remaining leaves. So this takes $O(N^2)$. Since we run through each step N times in order to join all the N leaves, the total running time of this algorithm is $O(N^3)$.

Results

90 Protein Sequences were used as the dataset in this project. So the distance matrix, generated by CLUSTALW, had a size of 90x90. I passed in this matrix as a parameter into each of the four algorithms that I had written and generated trees to represent the output of these algorithms, shown in figures 1-4.

From the algorithms dealing with different methods within hierarchical clustering, the single-linkage method seemed to fare the worst. It actually had a cascading effect of adding one protein at a time to a cluster while the other two seemed to have a more balanced tree as shown below in figures 1, 2 and 3. Due to this cascading effect, we weren't able to obtain any useful information from splitting the tree into smaller clusters.

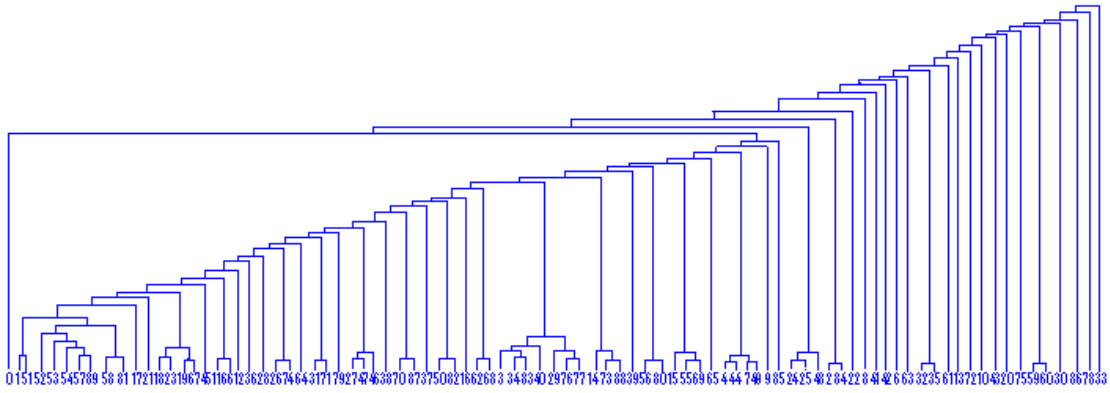

Figure 1: Single-Linkage Clustering

Between the average-linkage and complete-linkage methods, both methods actually did pretty good when looking at the clusters at the lower level that contained around 4-6 proteins. For example, they both found similar groups that were classified by PDB for oxidoreductase, nitrite reductase and ligase. However, when looking at larger clusters, these smaller clusters were actually clustered next to different groups of proteins in both methods, thus yielding different results overall for each method. It was hard to judge how well overall both clustering algorithms did but ideally, average linkage should have performed better since complete-linkage clustering is too strict a measure because of always picking the worse score as compared to average-linkage clustering that will pick the average of the scores of the clusters that were merged.

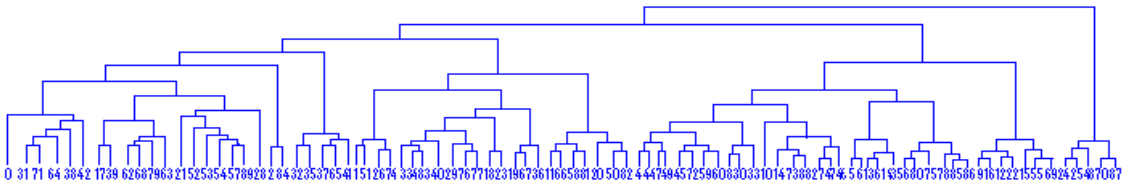

Figure 2: Average-Linkage Clustering

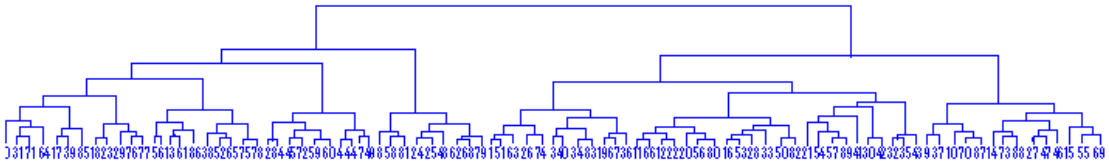

Figure 3: Complete-Linkage Clustering

Both Average-Linkage clustering and Neighbor Joining method are good methods to use when the data given is imperfect, that is, when we don't have ultrametric trees that follow the molecular clock. Even though according to the results neighbor joining had some cascading effects, we cannot judge it by that because of the fact that it outputs us an unrooted tree. Until we find a protein that is further away from all these 90 proteins than any of these 90 proteins are from themselves, in order to find the root, we cannot judge the overall nature of this tree. However, when looking at the smaller clusters generated at the lower levels (5-6 proteins), we see comparable results with the average-linkage method, that is, similar groups of proteins were clustered together in both. Ideally, however, the neighbor joining method is a better method to use since it reevaluates the distances from each node to every other node at each step while average-linkage is always biased towards larger clusters. However, hierarchical clustering does produce rooted trees which is easier to read and understand, so there is a tradeoff between ease in reading results and efficiency. Also since the data is imperfect, we are not guaranteed to obtain the optimal solution for either of the algorithms

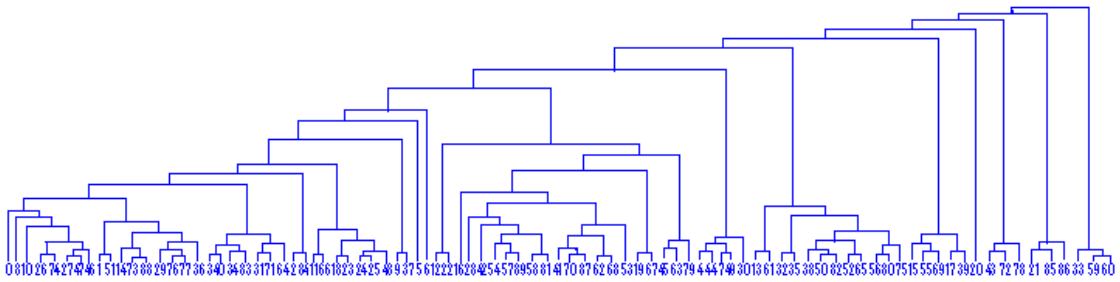

Figure 4: Neighbor Joining Method