BMI203 HW 2 Writeup

Matt Jones

February 10, 2018

Contents

1 Introduction

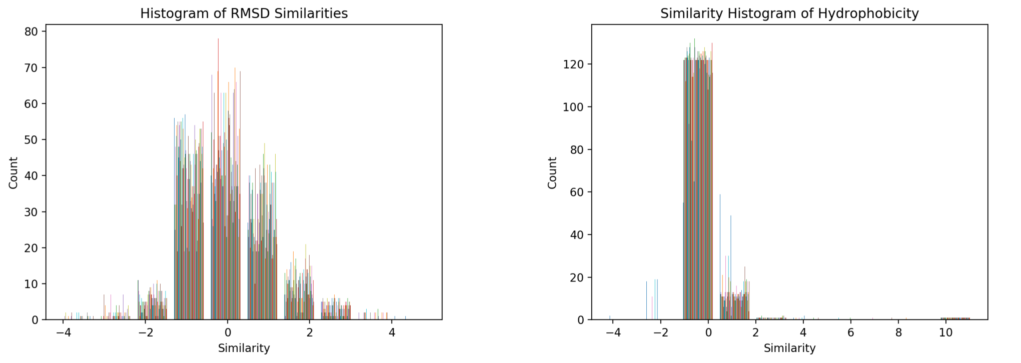
In this assignment, we developed clustering algorithms for protein sequences. These protein sequence came in `.pdb` form, so not only were sequences available but also their coordinates in 3D space. This document will outline how sequences were compared to one another, how they were clustered together, and the results from these clustering.

There were two procedures implemented - one partitioning method and one hierarchical method. K-means was used as the partitioning method, and an agglomerative method was used for clustering hierarchically. Both algorithms resulted in different clusterings, even with the same similarity criterion. The results, along with potential reasons for these differing results, are presented in this document.

2 Similarity Criterion

The first thing required for clustering samples together - whether they be protein sequences as in this assignment or images or any other type of subject - is a similarity criterion. Particular to this assignment, there are many different ways to compare protein sequences, each of which can yield different results. Because of this, two similarity metrics were implemented and compared - RMSD of backbone structures and difference in hydrophobicity between sequences. In both cases, we z-normalize the similarity data to obtain an optimal distribution to work with.

Figure 1: Comparison of Similarity Distributions across all peptides



(a) RMSD Similarity Distribution (b) Hydrophobicity Similarity Distribution

2.1 RMSD

The first similarity metric implemented compared the distance of the amino acid backbones, namely ± 1 atoms from C_α. In specific, we compared the RMSD of the sums of these distances:

$$RMSD = \sqrt{mean((d_{i,j}^2)}$$

where $d_{i,j}$ is the distance between i and j amino acid backbones. However, not all peptide sequences are of the same length – as a consequence, we employed an exhaustive search across all possible contiguous (i.e. no gaps) alignments, computed the RMSD of each alignment, and then returned one of many RMSDs (min, max, mean, median, or total) depending on the user input. Intuitively, RMSD makes sense when comparing proteins because much of a protein's activity is dependent on the geometry of the active site. Thus, it would make sense that proteins that look similar ought to have similar function, or bind with similar partners.

After employing this algorithm, we present the distribution of sequence similarities in Figure 1a (for the `min` option).

2.2 Hydrophobicity

A second pass at developing a similarity metric was focused on overall sequence similarity. Aside from a protein's geometry, its overall charge and polarity play a large role in its activity and which co-factors it has. As such, we employed a hydrophobicity criterion (many are out there, we chose one by Eisenberg and Weiss described in *Hydrophobic Moments and Protein Structure*, 1982) which scores each amino acid according to a relative hydrophobicity criterion. In short, we compared peptide hydrophobicity similarities - i.e. $d_{i,j} = (h_i - h_j)^2$, where $h_i = \sum_{k=1}^{n} hs(a_k)$ and $hs(a_j)$ is the hydrophobicity score of amino acid a_j for a length n peptide i. After computing all similarities, we formed a histogram which is presented in Figure 1b.

2.3 Evaluating the Similarity Scores

In Figure 1 I present the distribution of similarity scores across all peptide sequences. Interestingly, it seems that the th RMSD metric behaves better than the hydrophobicity index, most likely because there is more room for variance in the RMSD calculation. Given this information, it is most likely that the RMSD metric will produce better clusters than the hydrophobicity metric.

3 Metrics of Comparison

Before I detail the performance of my algorithms, I'll deal with the metrics I use to assess this performance. To this end, there are two important ways I quantify this performance - coherence of clusters from a single run and between clustering runs (i.e. two different clustering assignments). These quantities will be used to assess quality and stability of clusterings when I discuss my implementations in Section 4.

3.1 Quantifying Cluster Quality

To assess the quality of a particular clustering, I implemented a simple Sum of Distances calculation, which will report the average sum of squared distances within each cluster. Mathematically, for K clusters this is expressed as

$$Q = \frac{1}{K} \sum_{n=1}^{K} \sum_{i,j \in S_n} D(X_i - X_j) + C$$

where S_n is some cluster n of K, X_i and X_j are points assigned to that cluster, and $D(\bullet)$ is the distance. This metric gives an intuitive assessment of how homogeneous a cluster is, or how similar its points are within it. When I present my clustering algorithms, I will rely primarily on this metric to assess the comparative quality of the algorithm's assignments.

3.2 Quantifying Clustering Similarity

Comparing assignments across runs is a crucial step not only in assessing stability of assignments, but also to assess correctness of different algorithms and similarity metrics. Thus, I've implemented a Rand index calculator which takes two clustering assignments C_1 and C_2 and returns the fraction of "True", or consistent assignments. Formally, this quantity is defined as

$$R = \frac{TP + TN}{TP + TN + FP + FN}$$

where we have a laundry list of variables to define. One by one, we have

- TP: The number of pairs of objects in the dataset that are in the same cluster in both C_1 and C_2
- TN: The number of pairs of objects in the dataset that are in different clusters in both C_1 and C_2
- FP: The number of pairs of objects in the dataset that are in the same cluster in C_1 but different clusters in C_2
- FN: The number of pairs of objects in the dataset that are in different clusters in C_1 but are in the same cluster in C_2

This index will be used to compare results from different similarity metrics, and with different algorithms.

4 Algorithm Performance

As mentioned before, I've implemented both K-means and Agglomerative Hierarchical clustering which will be evaluated in this section. I'll deal with both individually and then assess their comparative performance.

4.1 K-means

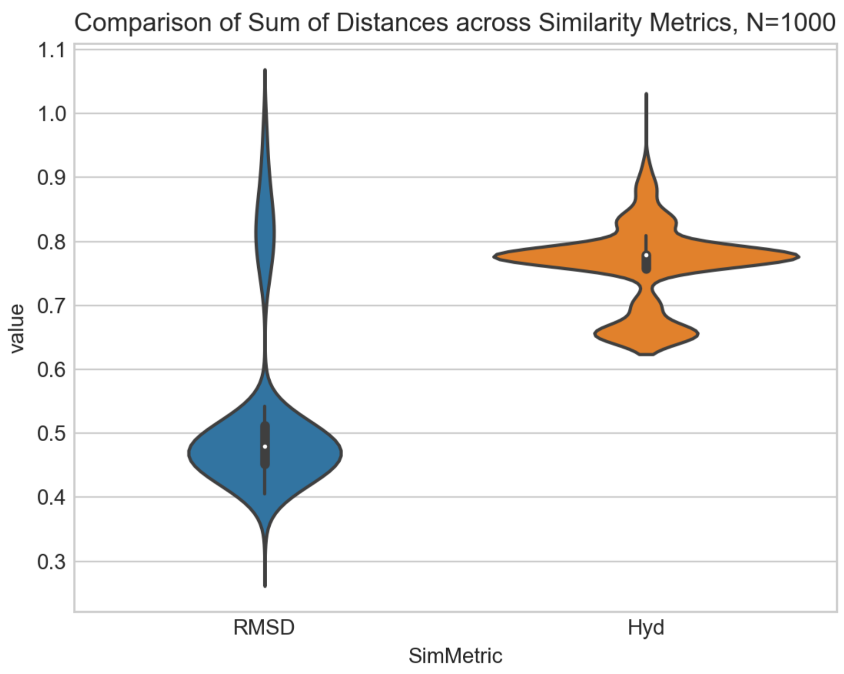
As a partitioning algorithm, I implemented K-means - an interative algorithm which will find K centroids that optimally partition the space according to the objective function:

$$S^* = argmin_S \sum_{j=1}^{K} \sum_{n \in S_j} D(X_n - \mu_n)^2$$

where S is a set of K clusters, S_j is the j^{th} cluster and S^* is the optimal clustering. Briefly, I use the following procedure:

1. Instantiate clusters by randomly assigning sequences to be cluster centers
2. Assign points by optimizing the objective function and finding $S*$.

Figure 2: Comparison of Similarity Metrics with SSD's for the K-means Partitioning algorithm.



3. Find new cluster centers by assigning the sequence with the shortest average distance to all other points in its cluster as the cluster center.

4. If no points moved clusters between iterations or if the maximum number of iterations was reached, return $S*$. Else return to step 2.

I report the sum of squared distances for both similarity metrics after 1000 assignments in Figure 2. To note, it's not exactly fair to compare the two at a raw scale, so the values are normalized to the maximum value seen across all 1000 simulations. As was expected with the similarity metrics, it's apparent that the RMSD metric creates much more coherent clusters, as the normalized scores are all fairly low. Interestingly, the hydrophobicity score has much less dispersion, most likely because the similarity distribution is much more compact than the RMSD similarity distribution.

4.2 Agglomerative Hierarchical Clustering

I implemented an agglomerative hierarchical clustering algorithm where I select can specify the number of clusters, K the algorithm should return. Roughly, I use a procedure as follows:

1. Assign each point to a cluster of its own

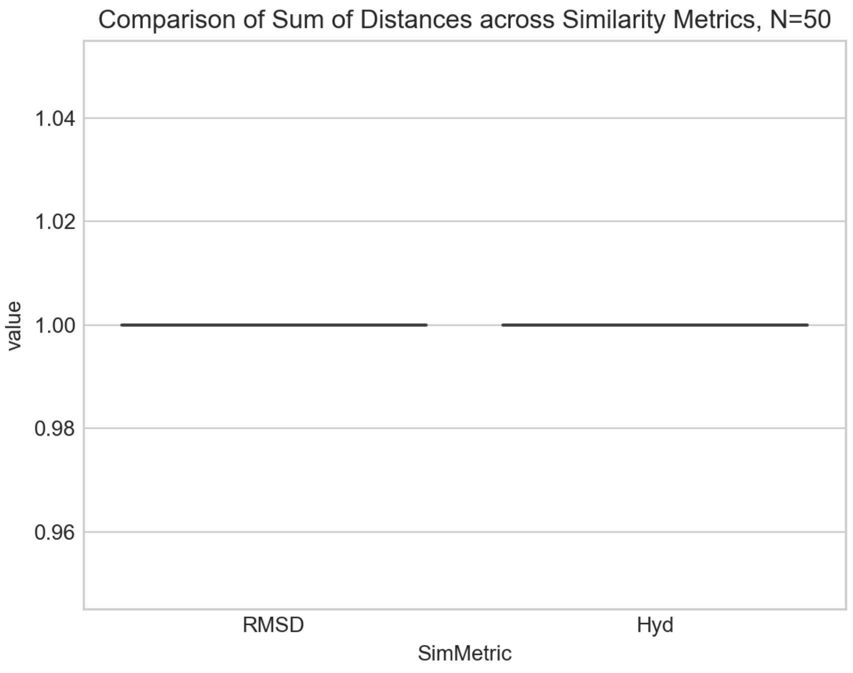
2. Select two clusters to merge according to some criterion (by default I use "single" linkage)

3. Repeat procedure until I have reduced the number of clusters from N to K.

Since I produced clusters using the "single" linkage criterion, I'd expect that most clusters are fairly drawn out. In short, "single" linkage picks two clusters to merge based on the distance between nearest neighbors To obtain more centroid-like clusters, I could implement a "complete" linkage; alternative linkage criteria will be left for further work.

Interestingly, the clusters produced by the agglomerative method is heavily biased towards only one cluster – i.e. one cluster has most samples and the rest of the $K-1$ clusters have one or two clusters. This may have to do with the data, the similarity metric, or the linkage criteria that I use when choosing which clusters to merge. I report the quality of clusterings in Figure 3; I assign clusters 50 times according to each similarity metric and report the Sum of Distances as in Figure

Figure 3: Comparison of Similarity Metrics with Sum of Distances for the Hierarchical Clustering Algorithm



2. As would be expected, since the hierarchical clustering algorithm is deterministic (as long as there are no ties in similarity), the Sum of Distances does not change across iterations.

4.3 Comparing Hierarchical and Partitioning Algorithms

Here I report briefly the relative Sum of Distances using the same metric between Partitioning and Hierarchical clustering algorithms. Because the hierarchical benchmarking only used 50 iterations rather than the 1000 in the K-means benchmarking, I sample from the 1000 values 100 times to obtain a distribution of values to compare against. I report the squared difference between these sum of squared distances, as well as a comparison between raw values. These results are presented in Figure 3.

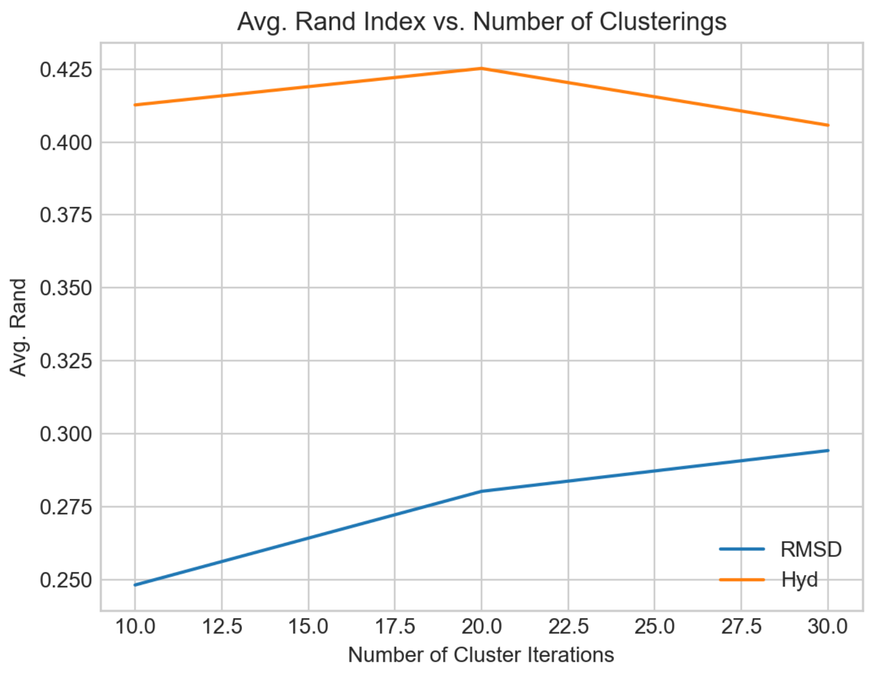
5 Assessing Cluster Stability

Aside from measuring cluster quality, we can also assess the stability of assignments by using the Rand index as described in Section 3. Here we'll assess the stability of assignments in K-means clustering and the comparison between K-means and Hierarchical clustering. Given that clustering is deterministic in the agglomerative algorithm (assuming the same linkage function), it is not possible to test stability across iterations for only hierarchical clustering.

5.1 K-means Cluster Stability

We assess the stability of K-means clustering by computing the mean Rand index after 10, 20, and 30 clustering iterations. The results are presented in Figure 4. Interestingly, the average Rand index is much lower for the RMSD similarity metric, even though this metric had on average more homogeneous clusters. This is most likely because the similarity scores had higher variance than the hydrophobicity metric, allowing the latter similarity score to produce more similar results even though the clusters were more heterogeneous.

Figure 4: Average Rand Index vs. Number of Clustering Iterations



5.2 K-means vs. Hierarchical Clusters

For K= 5, 10, 15, and 20, I generated clusters with my agglomerative method and with the K-means method and report the Rand index in Figure 5. It's clear from the results that assignments don't agree well between the two clusterings – mostly because of the cluster skew discussed in Section 4.2. Still, as the number of clusters allowed, the RMSD similarity metric results in clustering assignments that are far more stable than the hydrophobicity metric.

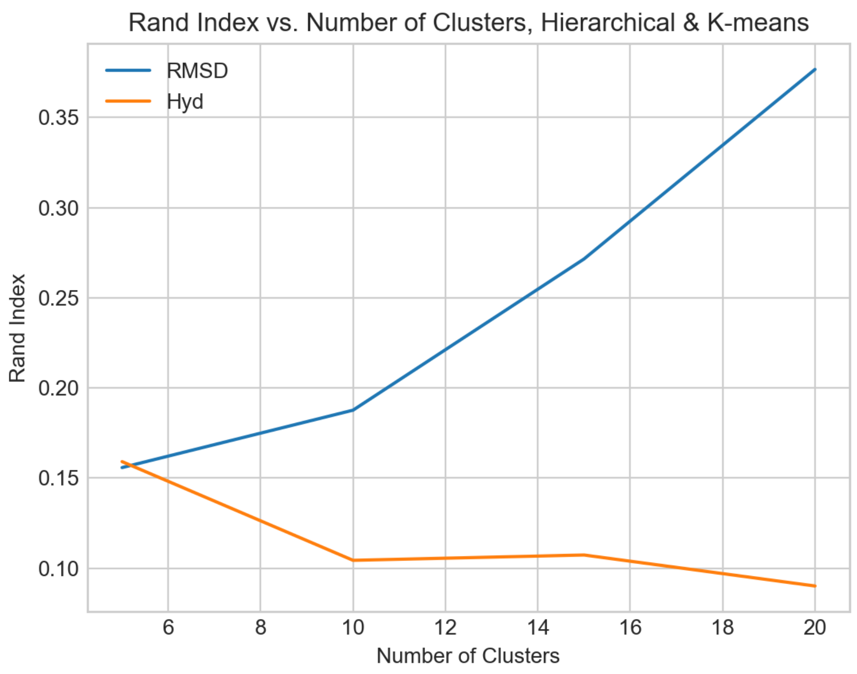
6 Discussion

Here I've presented two algorithms for clustering protein sequences, based on various similarity metrics. I've shown that the K-means algorithm especially performs fairly well with the RMSD metric, providing relative consistent assignments and homogeneous clusters. These observations were less so for the hierarchical clustering results, which for both similarity metrics provided skewed clusters. I've noted that this could be due to the linkage function, or perhaps due to the similarity metric. Here I'd like to offer one more possiblity: the number of clusters specified is far less than the number of protein "families" in the collection of proteins. If this were the case, perhaps as we ascend up this "phylogeny" of protein families by merging clusters, we miss the actual meaningful clusters by merging too much. This can be explored further by more intensely analyzing the quality of clusters as you modulate the number of clusters specified for the hierarchical clustering algorithm.

A key motivation for these algorithms is to find real biological signal from the unsupervised clustering assignments. Although no biological semantics go along with these files, it's difficult to assign actual biological meaning to the active sites. Still, given my similarity metrics it's hard to imagine a scenario where these clusters didn't have any signal.

Further analyses should focus on comparing the impact of similarity metrics and linkage functions on the clustering assignments. Additionally, more sequences with known biological function should be used in clustering.

Figure 5: Evaluating K-means & Hierarchical cluster similarity as a function of number of clusters.



7 Code Availability

Code implementing these tests, benchmarks, and algorithms can be found on my github repository.