Ryuho Kudo
Kathrine Blizard
CPE466 Lab 4 Report

1. Study Design:
For this assignment we implemented kmeans and hclustering with the following features/ options:
kmeans

- Used euclidean distance for all distance measurements.
- Choice of SelectCentroid and RandomSelect method for finding initial clusters.
- Number of reassigned pointers, total distance change of all centroids, and difference of sum of squared error (SSE) are the choices for stopping conditions.
- No feature was implemented to filter out outliers

Agglomerative

- Euclidean distance between points
- Centroid centers used as comparison distance measure between clusters
- Gives the sum of squared error as evaluation measure. Machine Learning (making the clusters into classes and seeing if the computer through C45 can learn those clusters) was attempted but proved to be too complicated to integrate in time.
- No feature was implemented to filters out outliers.

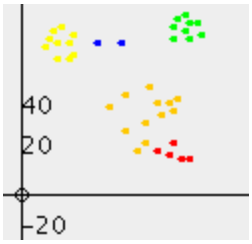
2. Results:
We decided to evaluate how well the clusters did by the total SSE value. We realize that if we have a cluster for each data point, the total SSE value would be 0 so we decided to pick the results with minimum number of clusters with decent SSE value.

kmeans uses euclidean distance for all distance measurements, SelectCentroid method as initial cluster selector, thresh hold is set for difference of SSE for stopping condition.

Input	Best kmeans result	Best hclustering result
4clusters.csv	k=5 TSSE=894.17	threshold = 12, Total SSE: 2189.1618315, 4 Clusters
mammal_milk.csv	k=5 TSSE=199.26	threshold 10: Total SSE: 255.65331, 5 Clusters
many_clusters.csv	k = 12 TotalSSE Value=1185.50	threshold 10, Total SSE: 1421.3469438552856, 12 Clusters.
AccidentsSet01.csv	k=5	threshold 3,

	TotalSSE Value=23.25	Total SSE: 29.693359375, 5 Clusters.
AccidentsSet02.csv	k=5 TSSE==1016.58	threshold 10, Total SSE: 1023.2980534, 4 Clusters.
AccidentsSet03.csv	k = 6 TSSE=55.75	threshold 2, Total SSE: 68.54197367307631 7 Clusters

We also created a 2D scatter plot for kmeans function to help visualize the result of the program.



This is a visualization of kmeans algorithm with k=5 for 4clusters.

3. Discussion:

We found that many_clusters and AccidentSets02 were hard to cluster. We usually can get a decently low total SSE value if we create around 3-6 numbers of clusters, but with many_clusters, we had to make 12 clusters and we still get very high total SSE value. This is because the points are spread out, even a person eyeballing a graph would have difficulty in pinpointing clusters.

While running hclustering we realized that AccidentSets03 had 2 outliers ([10,0,4,1,1] and [1,0,4,3,3]) which made it hard to cluster with thresh hold of 3.

Similarly, we found a few outliers while running hclustering on many_clusters ([46,30], [50,19] and [27,3]) with a thresh hold of 10.

4. Analysis:

The kmeans clustering method takes in a single parameter which is the number of clusters to make. This parameter greatly affects the output of the program since if the k value is too small, it forces the data points to combine and create a cluster with a large spread. We implemented both random selection and SelectCentroid algorithm for seeding the initial centroids and we concluded that while Random method sometimes produce better results, SelectCentroid performs better in most case, and that is why we used it for majority of data analysis.

The Agglomerative hierarchical clustering method depended entirely on the threshold taken as input. If the threshold was too low, the sum of squared would report a good, low number,

but many clusters would be created, and often they would contain only one point in their collection. If the number was too high, then a low number of very large clusters were created (and it may include outliers), and the SSE would report a high error level with the cluster being too spread apart. At this point, the evaluation of the Agglomerative appears to be a subjective measure, and a compromise between a large threshold and a small one seems to be the most appropriate. At a compromise, the performance of the Agglomerative method is similar to the performance of the K means method. When setting the k in K-means to equal the number of clusters Agglomerative output in an earlier run, K-means tends to have a lower sum of squared error. The Agglomerative method is better adapted to finding outliers because it creates clusters based on distance, whereas the clusters K means may be shifted over by outliers.

The Euclidean distance measure between points is not weighted, and there is nothing in the input data the specifies a weight. This would be making the assumption that all the fields in the data set are of equal importance. Our implementation does not consider data sets where the user must specify which variables are most important.