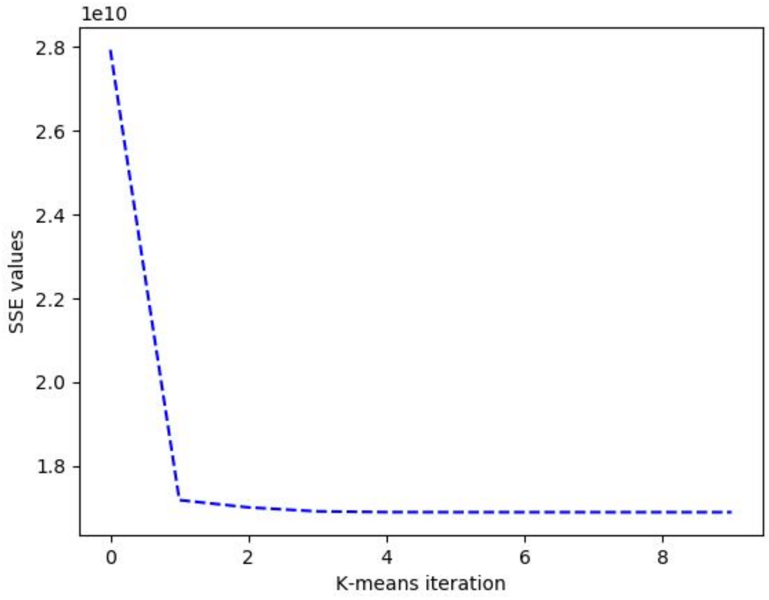
CS434: #4 Clustering Algorithms
Sam Jacobs, Erich Kramer, Markus Woltjer

Part I: Non-hierarchical clustering - K-Means algorithm

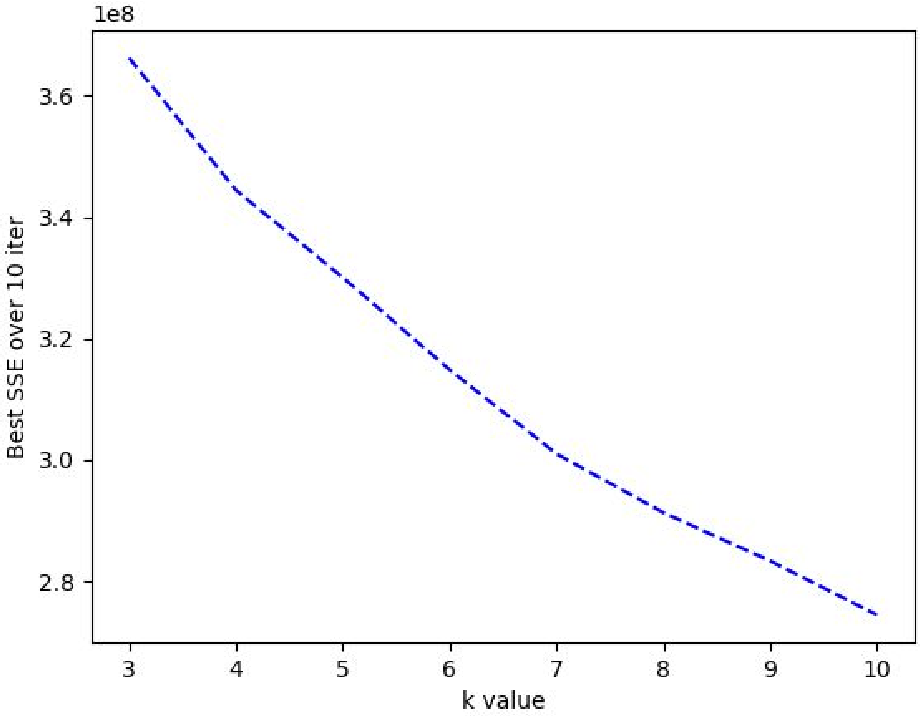
1. Implement K-means algorithm. Run your K-means algorithm with k = 2. To verify that your algorithm actually converges, please plot the objective of the K-means algorithm (i.e., the SSE) as a function of the iterations. From one run to another run, this curve may look different. Just present the results of a typical run.

Each instance of the k-means starts off with randomly selected points from the data set which are used as a starting points for the means.

The first iteration moves the means to a much improved local minimum. The Algorithm then continues to iterate until a stable SSE has been found. This takes anywhere from 5 to 20 steps on average, as the means begin slowly to approach an optimum. After stabilizing the points have found the local optimum for their initial values.



2. Now apply your K-means implementation to this data with different values of k (consider values 2, 3, · · · , 10). For each value of k, please 1 run your algorithm 10 times, each time with a different random initialization, record the lowest SSE value achieved in these 10 repetitions for each value of k. Plot the recorded SSE values against the changing k value. What do you think would be a proper k value based on this curve? Please provide justification for your choice.



As the number of means is increased, the value of the best SSE decreases proportionately. Upon observing the trend at values of k=4 or k=7 there is a knee in the graph. This suggests that the data set has a cluster amount equal to these respective k values, possibly containing subclusters at k = 7.

Part II: Hierarchical agglomerative clustering (HAC)

1. HAC algorithm using single link. Report the dendrogram starting with 10 clusters. Start building your dendrogram with those 10 clusters until you get only one cluster. Clearly indicate the heights of the tree branches, i.e., the distances between merged clusters at that particular step. Looking at the dendrogram, can you determine the number of clusters? Explain your choice.

One method of determining the number of clusters is to cut at the point where the gap between two distances is the largest. In the case of the singly linked clustering, the largest delta in the distances is between 3 and 4 clusters. Cutting before this combination leaves us with an estimate of four clusters. This coincides with the previous prediction based on the error of the different k values.

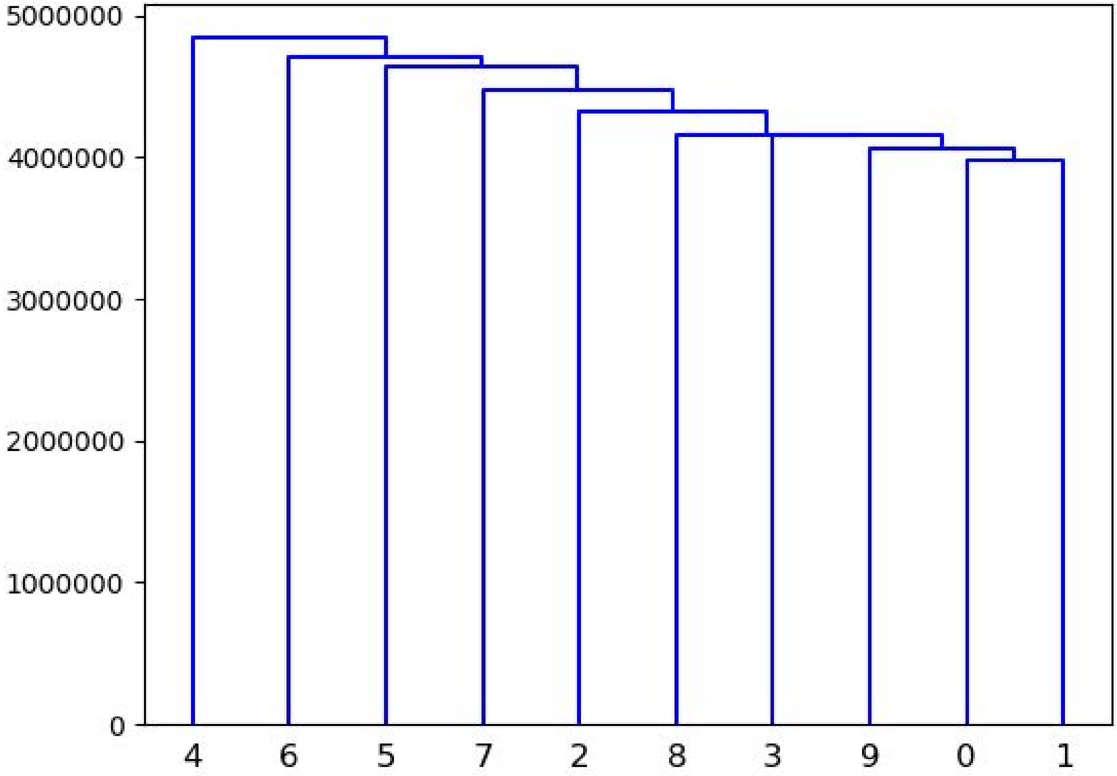


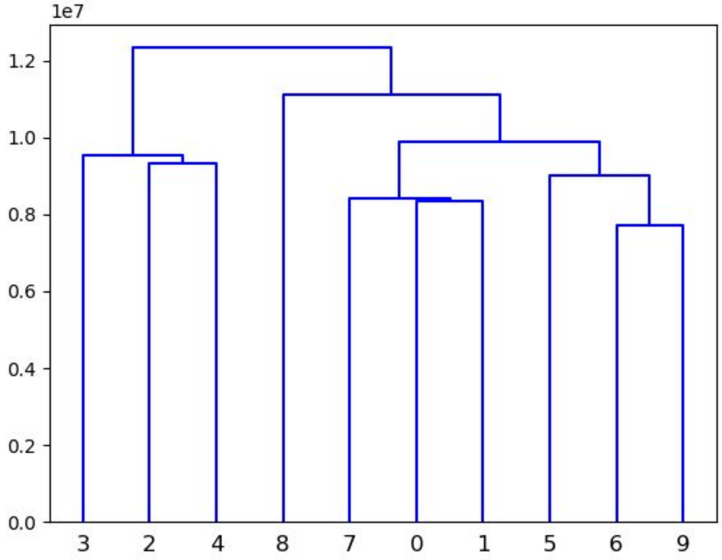
Table of exact heights of merges:

Merges	Height
{0,1}	3974902
{9, {0,1}}	4062523
{3, {0,1,9}}	4156466
{8, {0,1,3,9}}	4158115
{2, {0,1,3,8,9}}	4316733
{7, {0,1,2,3,8,9}}	4472301
{5, {0,1,2,3,7,8,9}}	4632352
{6, {0,1,2,3,5,7,8,9}}	4703695

{4, {0,1,2,3,5,6,7,8,9}}	4841502

2. HAC algorithm using complete link. Report the dendrogram starting with 10 clusters. Clearly indicate the heights of the tree branches, i.e., the distances between merged clusters at that particular step. Looking at the dendrogram, can you determine the number of clusters? Explain your choice.

If we examine this dendrogram instead of the singly linked cluster dendrogram, we observe higher order of combining in the final stage. The largest deltas in distance occur in-between the clustering into 3 clusters and 4 clusters. This suggests that the best k value might be 3. However, previous graphing and dendrograms form the singly-linked HAC suggest that the best k value would be k=4. This leads to the conclusion that an outlier may be skewing the Completely linked HAC and that the true cluster count should be four.



Values of merges

Merges	Merge Height

{6,9}	7732517
{0,1}	8335838
{7, {0,1}}	8411717
{5, {6,9}}	9022978
{2,4}	9314567
{3, {2,4}}	9528207
{{7,0,1} , {5,6,9}}	9884848
{8, {0,1,5,6,7,9}}	11126301
{{2,3,4} , {0,1,5,6,7,8,9}}	12334294