SENG474 A2 Report

Jake Houston

Feb 24, 2025

1 Introduction

In this report, we analyze the performance of Lloyd's algorithm using uniform random initialization and k-means++ initialization, as well as hierarchical agglomerative clustering with single linkage and average linkage. These algorithms are run on two datasets, one containing 3500 two-dimensional points generated by a Gaussian mixture model, and the other having 14,801 three-dimensional points. We will be running experiments with the 4 models to learn which configuration creates the best results for clustering the datasets.

For both uniform random initialization and k-means++ initialization in Lloyd's algorithm, we will be varying the amount of clusters k from 2 to 15, and for each k, we will make a plot of the cost as the number of clusters increases to help us determine the optimal amount of clusters to use. After running this experiment with each dataset, we will determine the optimal configurations for both versions of Lloyd's algorithm for two- and three-dimensional datasets.

Then for both single linkage and average linkage, we will generate the corresponding dendrogram, only looking at the first 100 splits in order to truncate the graph and avoid looking at too many lower levels of the dendrogram; then using the graph we will decide where to make a cut in the dendrogram to result in the best clustering configuration. This will allow us to compare the difference between single and average linkage and to find the optimally tuned hierarchical agglomerative model.

2 Lloyd's Algorithm

We first implemented two versions of Lloyd's algorithm in Python using Euclidean distance, one using uniform random initialization and the other using k-means++ initialization. We first experimented with uniform random initialization, varying the number of clusters k between 2 and 15, and running each one 5 times to get the best results. This procedure is run on the two datasets

separately to determine the optimal configuration for 2- and 3-dimensional examples. In Figure 1, we have shown how the cost decreases as the number of clusters k increases for the 2-dimensional dataset 1. This trend helps us determine an appropriate number of clusters that achieves the best generalization in a minimal number of clusters. Based on the plot, we select 4 as the optimal number of clusters since the cost decreases significantly up to k = 4 and flattens out beyond this point, showing that there is decreasing returns and a risk of overfitting. This can be seen in the final clustering shown in Figure 2. We then repeat this process for the 3-dimensional dataset 2, which we can see in Figure 3. Here we notice a similar pattern leading us to select k = 7 as the optimal number of clusters. Beyond 7, the decrease in cost is in very small increments, reflecting the same pattern as seen in the two-dimensional dataset where increasing is risking overfitting. Figure 4 shows our final clustering for dataset 2.

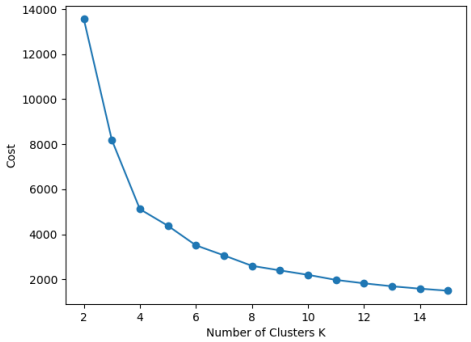

Figure 1: Cost

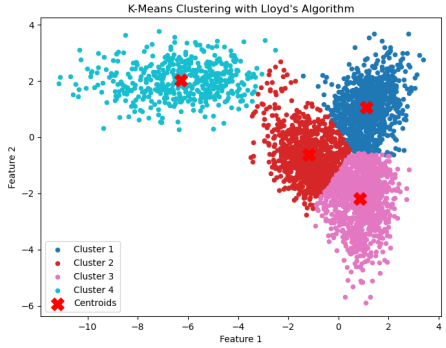

Figure 2: 4 Clusters

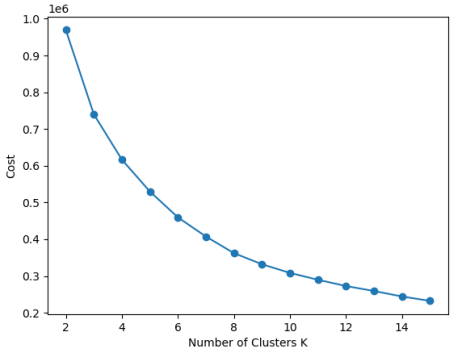

Figure 3: Cost

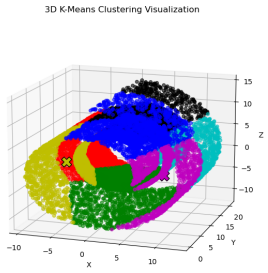

Figure 4: 7 Clusters

Next, we repeated the same process using the second model, which instead uses k-means++ initialization. For both datasets, we varied the number of clusters k from 2 to 15 and ran each configuration five times to ensure reliable results. For the two-dimensional dataset 1, a plot of the cost can be seen in Figure 5. Similar to the previous experiment, we observed that the cost decreases significantly up to k = 4, then the curve starts to flatten out. So again, we select k = 4 as the optimal number of clusters to balance generalization with the number of clusters. The final clustering using k = 4 can be seen in Figure 6. We then applied the same process to the three-dimensional dataset 2, which is visualized in Figure 7. Reflecting the same results as uniform random initialization, the cost follows a steep curve down to k = 7 then begins flattening out, leading us to select k = 7 as the optimal cluster count. Figure 8 represents the final clustering using k = 7.

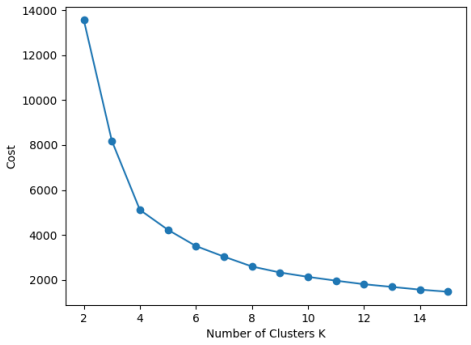

Figure 5: Cost

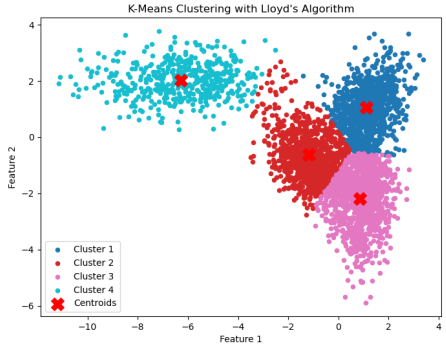

Figure 6: 4 Clusters

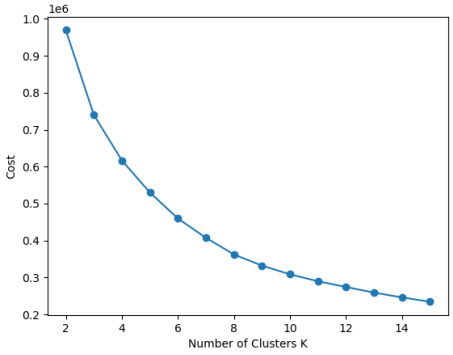

Figure 7: Cost

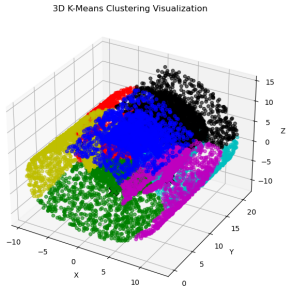

Figure 8: 7 Clusters

3 Hierarchical Agglomerative Clustering

Using Python's scikit-learn library, we implemented two versions of hierarchical agglomerative clustering. Both using Euclidean distance for the dissimlarity measure between examples. For the dissimilarity measure between clusters, one model uses single linkage and the other uses average linkage. For each of the two models, we generated clusters and visualized the clustering process using dendrograms. To avoid seeing too many of the lower levels of the dendrograms, we truncated them. This visualization helps us determine the proper cut point in the dendrogram to produce the optimal final clustering which balances cluster separation and tightness. Figure 9 presents the dendrograms for both single linkage and average linkage models on the two-dimensional dataset 1. When selecting the cut point, we are looking for a distance that balances creating separate and tight clusters. In the single linkage model, we notice that the chaining effect is leading to long and loosely connected clusters. However, there is a larger gap between merges at a distance of 0.3, making this our ideal cut point. For the average linkage model, we realize the chaining effect is not present, and the structure of the dendrogram is much more balanced. Looking for the largest vertical gap between merges, we notice it at a distance of 3, which creates three separated clusters without the chaining issue and more balanced clustering. A similar process is applied to the three-dimensional dataset 2, with the dendrograms shown in Figure 12. The same pattern is noticed where, in the single linkage model the chaining effect is present and we decide to make the cut at a distance of 0.9 as the merges start to level off, in the average linkage model we see a large vertical gap around a distance of 10 where we select our cutting point.

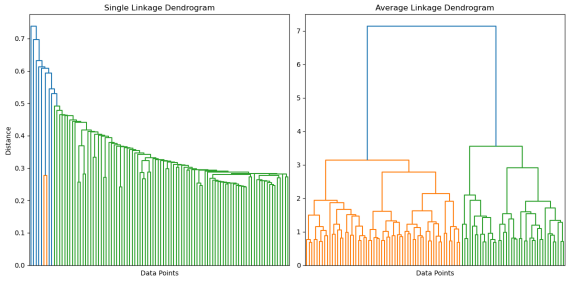

Figure 9: 2D Dataset

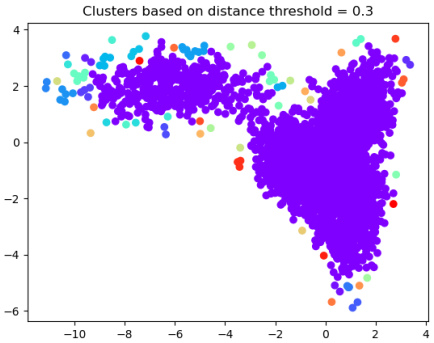

Figure 10: Single Linkage

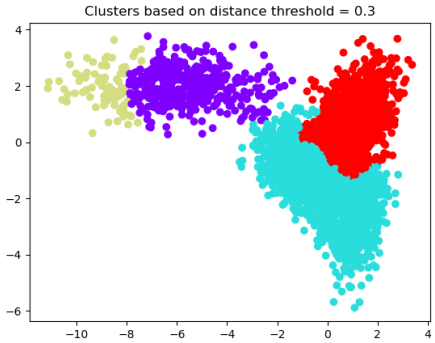

Figure 11: Average Linkage

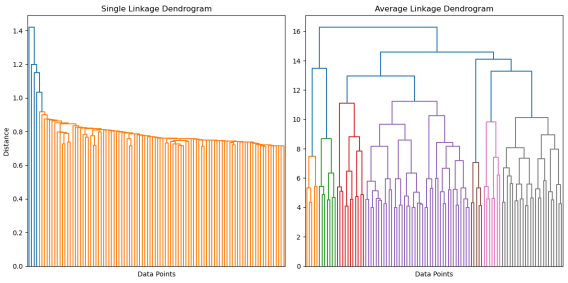

Figure 12: 3D Dataset

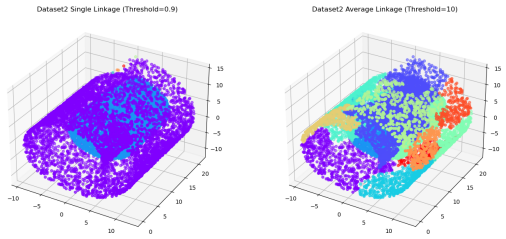

Figure 13: 3D Dataset Scatterplots

4 Conclusion

To conclude this report, we will compare the results of the 2 Lloyd's algorithm models and the 2 hierarchical agglomerative clustering models. We observed

from experimenting with uniform random initialization and k-means++ initialization that there is very little difference in the final results of the two methods. However, the speed at which k-means++ converged was much faster than that of uniform random initialization. This is attributed to its initialization strategy, which selects centers based on probabilities related to Euclidean distance, encouraging more separated starting points. Whereas, uniform random initialization can result in poor randomly chosen initial centers, leading to a slower convergence. Experimenting with the hierarchical agglomerative clustering methods, there is a significant difference in results between the single linkage model and the average linkage model. We notice the single linkage models structure from Figures 9 and 12 is heavily influenced by the chaining effect, and thus creates poor generalization and far too many early merges. Using average linkage led to better generalization of the data points for both the two- and three-dimensional datasets, and the structure of the dendrograms is much more interpretable. Overall, single linkage hierarchical agglomerative clustering struggled to generalize the data in a small amount of merges, average linkage resulted in a few more clusters required for the 3D dataset, and a slightly worse generalization of data points for the two-dimensional dataset than the Lloyd's algorithm models, and the 2 Lloyd's algorithm models performed at a very similar level with k-means++ initialization being faster to converge.