K-Means Clustering Implementation Report

1. Introduction

This report will provide a detailed explanation of the solution I implemented for the k-means clustering algorithm. Using C++, I designed a program that performs clustering on a dataset and minimizes the Sum of Squared Errors (SSE) between data points and their respective cluster centers. I will describe the data structures that have been used in each step and the algorithms.

2. Data Structures

The main data structure that is used in this program is vectors. I have decided to use vectors as I am familiar with them and they allow for flexible resizing. I implemented this using the C++ std::vector class which allowed me to use them effectively and efficiently. Throughout the program, I've used two-dimensional vectors and one-dimensional vectors to help me implement the algorithm.

The two primary two-dimensional vectors that have been used for this solution are the data points and the cluster centers. For the data vector, each element corresponds to a data point. Additionally, each data point is represented by a vector of doubles which will store the features of the points in a dimensional space that is determined by the dimension of the dataset. The centers vector stores the centroids of the clusters, with each element having a vector of doubles that represents the center of the cluster.

Other vectors have been used to help create the solution. For instance, the assignments vector allowed me to store the index of the cluster that each data point is currently assigned to. Each element in this vector holds the data point's closest cluster. Additionally, the cluster_sum and cluster_size vectors were used to calculate the new centers by finding the mean center of the points of each cluster and storing them in a new_centers array.

3. Functions and Algorithms

a. Reading the Dataset

The ReadFile function conducts the reading of the dataset. Firstly, this function validates if the file was successfully opened. Next, it opens the file of the dataset, iterates through each point, and stores the points in the data vector. Each point is stored as a vector of doubles.

b. Selecting the Initial Cluster Centers

The SelectCluster function randomly selects the initial cluster centers by uniformly generating random indices from the data points. The IndexCheck function is used as a helper function within this function to see if the randomly selected index has been generated to ensure

uniqueness. Once the initial centers have been selected, they are then returned as the starting centers for the algorithm.

c. Assigning the Points to the Clusters

The AssignClusters function assigns each data point to the cluster center that is nearest to them by using the squared Euclidean distance. This is done by using the helper function, SquaredEuclidean which calculates the distance to each cluster center for each data point. Once the lowest distance to the center has been found, the point is assigned to that center.

d. Calculating the Sum of Squared Errors

The SSE function calculates the Sum of Squared errors for each data point with its current cluster assignments. This is done by using the SquaredEuclidean function again, but this time it is calculated between the point and its assigned center. They are then summed up to compute the total SSE and it will also be used to track the convergence to know if it has reached its threshold.

e. Updating the Cluster Centers

The UpdateCluster function will reassign the cluster centers by calculating the mean of the points assigned to each cluster. It will iterate through all the points, and sum their features based on the clusters that they are assigned to. It will then divide the sum by the size of the cluster. An edge case has been implemented in this function where if a cluster only contains one point (singleton), then it will find the point with the greatest error (distance) from its current center and assign it to the new center.

f. The Main Loop

The main function will take in the user input (file_name, num_cluster, max_iter, threshold, and num_runs) and run the program for a certain number of runs or until the SSE is smaller than the threshold. The loop initializes the cluster centers, assigns points to the clusters, and then iteratively updates the centers and reassigns points. It will determine that the algorithm has converged when the SSE is smaller than the threshold. The program will also track the best run and the lowest SSE of the algorithm.

4. Conclusion

The k-means clustering algorithm that has been implemented has used random initialization for its starting clusters, uses distance as the factor for assignment, and updates the centers iteratively by reassigning the centers after calculating the mean of the points. Vectors have been used as the main data structure to ensure flexibility and efficiency. There was no issue with running the program on the bigger datasets as it completed its run within minutes. Some improvements could be made to make the algorithm more efficient. For instance, the use of a map, or a set could allow for a better program. Overall, the program has a way to clearly calculate the SSE to monitor the convergence. Thus, allowing for the approach to be efficient.