Name: **Hend Aly**
ASU ID: **1229580406**

K-means-Strategy Project Part 2

Introduction

The K-means clustering algorithm is a widely used unsupervised learning technique in machine learning. It takes a dataset and partitions it into **k** clusters, where each data point belongs to the cluster with the nearest mean. The goal of this particular part of this project is to implement K-means clustering using a more advanced initialization strategy for the initial centers. This strategy involves picking the first center randomly and selecting each subsequent center based on the sample that maximizes the average distance to all previous centers.

Methodology

The implementation of the k-means algorithm with the new strategy consists of several steps, which include data preparation, initialization of centers based on strategy 2, calculation of the Euclidean distance, assignment of the points to the closest centroid, updating centroids, and iteration process to convergence. The execution of the code is in the following steps:

1. **Data Preparation**

   The dataset provided (**AllSamples.npy**) consists of a set of 2-D points that we clustered using the K-means algorithm. The dataset was loaded using the provided code. I applied the algorithm to this dataset for different **k** values (from 2 to 10).

2. **Initial Centers Selection (Strategy 2)**

   The provided code used the (**initial_S1()**) function to randomly choose the first center. For subsequent centers, I created a function called (**select_next_centroid**) which selects each subsequent centroid by maximizing its average distance to all previously selected centers using the (**euclidean_distance**) function. Points that are already selected as centroids are skipped to avoid duplication. It then selects the point with the maximum average distance to the existing centroids which ensures that the new centroid is as far away as possible from the previously selected centroids. Using this initialization strategy, we avoid choosing centroids that are too close to each other which results in better clustering.

3. **Euclidean Distance Calculation**

   The (**euclidean_distance**) function calculates the distance between a point and a centroid using the below formula, where x_i is the point and μ_i is the centroid.

$$\boxed{d = \sqrt{\sum (x_i - \mu_i)^2}}$$

4. **Points to Centroids Assignment**

   The (**assign_points_to_centroids**) function assigns each point to the closest centroid based on the calculated Euclidean distance. The data points are divided into clusters, which are stored in a dictionary called (**clusters**).

5. **Centroids Updating**

   The (**update_centroids**) function recalculates the centroids by finding the mean of the points in each cluster.

6. **Iteration Process**

   The (**k_means**) function iterates to assign points to centroids and updates centroids until the centroids converge. That was checked using the **(np.allclose())** function, which ensures that iterations stop when the centroids don’t change.

7. **Strategy 2 Implementation**

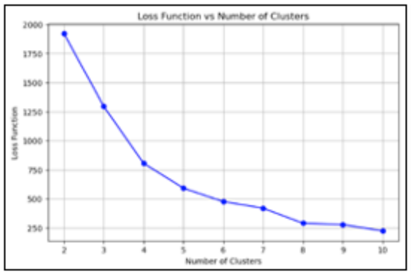
   The (**strategy_2**) function implements the second strategy for selecting initial centroids, where the first centroid is chosen randomly and for each subsequent centroid, it selects the point that maximizes the average distance to all previously chosen centroids.

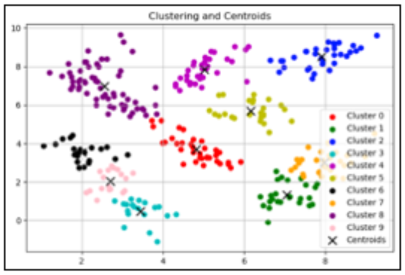
8. **Loss Function Evaluation & Results Plots**

   I calculated the loss function as the sum of squared distances between each data point and its assigned centroid. This loss is stored for each **k.** I generated two plots, one for the loss function versus the number of clusters, the other shows the clustering of data points and the final centroids for a selected **k** value.

Results & Analysis

I calculated the loss values for **k** ranging from 2 to 10 as shown below. These values represent the sum of squared distances between points and their cluster centroids which indicates the clustering quality. The loss decreases as the number of clusters increases, which is expected since the more clusters, the tighter the groupings of points. The decrease in loss indicates an optimal number of clusters. This point is used to select an appropriate number of clusters for the dataset. I also created a scatter plot for **k=5** to show the clustering and the positions of the final centroids as shown below.


Loss Function vs Number of Clusters
Loss Function
Number of Clusters


Clustering and Centroids
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4
Cluster 5
Cluster 6
Cluster 7
Cluster 8
Cluster 9
Centroids