Assignment-6: K-means Clustering

Munem Shahriar 2021251042
Tausif Mushtaque 2031020642
Email addresses: munem.shahriar07@nothsouth.edu
tausif.mushtaque@northsouth.edu

December 8, 2024

I. Introduction

K-means clustering is one of the most widely used unsupervised learning algorithms. It partitions a dataset into k clusters, where each data point belongs to the cluster whose centroid is the nearest. The algorithm works iteratively to minimize the within-cluster variance. It is often used in pattern recognition, image compression, and market segmentation. Despite its simplicity, K-means is effective for a variety of clustering tasks. This report describes a custom implementation of the K-means algorithm and demonstrates its application on a sample dataset.

II. Method

The K-means algorithm starts by initializing k centroids randomly from the dataset. It then iterates through two main steps until convergence:

1) **Assignment Step:** Each data point is assigned to the nearest centroid.
2) **Update Step:** The centroids are recalculated as the mean of all points assigned to them.

This process repeats until the centroids no longer change significantly, indicating convergence.

The custom implementation of K-means can be expressed as follows:

```
import numpy as np

def k_means(data, k, max_iters=100, tolerance=1e-4, random_state=None):
    if random_state:
        np.random.seed(random_state)

    n_samples, n_features = data.shape
    centroids = data[np.random.choice(n_samples, k, replace=False)]

    for _ in range(max_iters):
        distances = np.linalg.norm(data[:, np.newaxis] - centroids, axis=2)
        labels = np.argmin(distances, axis=1)

        new_centroids = np.array([data[labels == i].mean(axis=0) for i in range(k)])

        if np.all(np.abs(new_centroids - centroids) < tolerance):
            break

        centroids = new_centroids

    return centroids, labels
```

The core of the algorithm involves calculating the distance from each data point to all centroids. The distance between a data point x_i and a centroid c_j in an n-dimensional space is given by the Euclidean distance formula:

$$d(x_i, c_j) = \sqrt{\sum_{l=1}^{n} (x_{i,l} - c_{j,l})^2}$$

Where $x_{i,l}$ is the l-th feature of the i-th data point, and $c_{j,l}$ is the l-th feature of the j-th centroid.

Once the distances are computed, the data points are assigned to the closest centroid, and new centroids are computed by averaging the assigned points:

$$c_j^{new} = \frac{1}{|S_j|} \sum_{i \in S_j} x_i$$

Where S_j is the set of data points assigned to centroid j, and c_j^{new} is the updated position of centroid j.

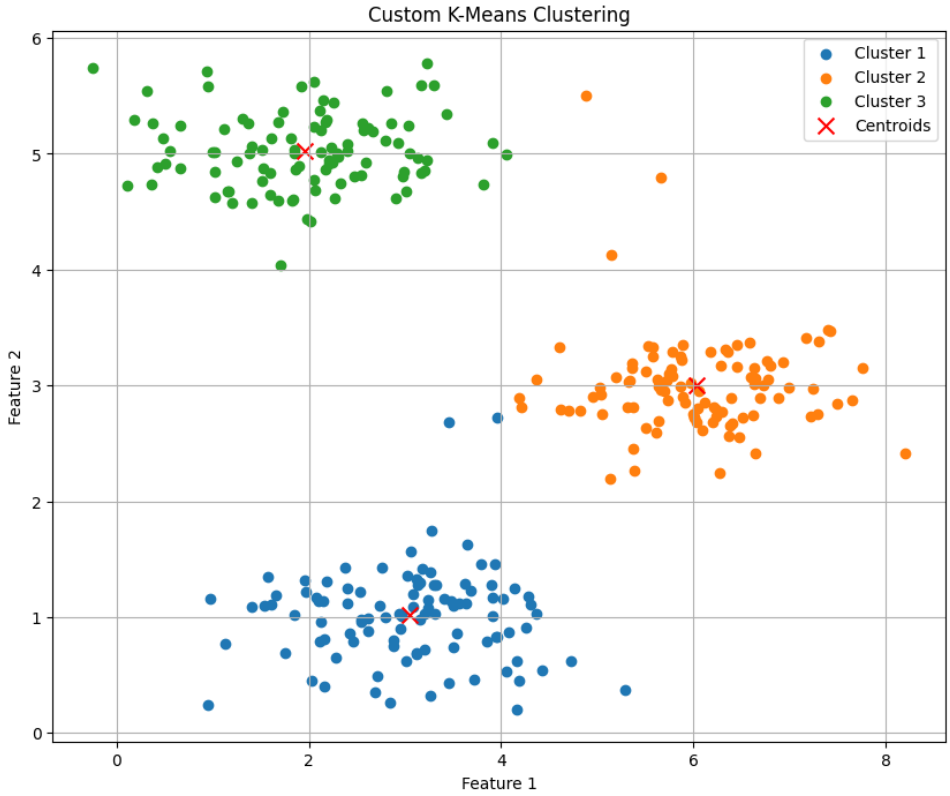

Fig. 1. Clustering result after applying the custom K-Means algorithm with 3 clusters.

The final result is a set of centroids and cluster assignments for each data point.

III. Conclusion

The K-means algorithm is a simple yet powerful method for clustering high-dimensional data. Our custom implementation, using basic numpy functions, performs clustering by iterating through the assignment and update steps. This method is particularly effective for datasets where the number of clusters k is known a priori. However, the algorithm is sensitive to the initialization of centroids and may converge to a local minimum.

Through the example presented, we observed that the clustering algorithm was able to successfully partition the data into 3 clusters, as indicated by the centroid locations. This technique can be applied to various types of datasets, including high-dimensional data, making it a versatile tool in machine learning and data analysis.