Assignment-6

Zarin Akter (2011704042), Rashik Iram Chowdhury (2111336642) and Md. Mutasim Farhan (2013123642)
Emails: {zarin.akter@northsouth.edu}, {rashik.chowdhury@northsouth.edu}, {mutasim.farhan@northsouth.edu}

May 21, 2024

I. Introduction

This assignment delves into implementing and applying the K-Means clustering algorithm, a fundamental unsupervised machine learning technique. K-Means is designed to partition data into distinct groups (clusters) based on similarity, where data points within a cluster are more similar than points in other clusters. We will develop a K-Means algorithm from scratch without relying on pre-built libraries. We will apply our K-Means algorithm to a two-dimensional dataset to visualize how it groups data points into clusters. In this paper, we will try to analyze the relationship between the number of clusters and the distortion cost (a measure of how well the data fits the clusters). As a practical application, we will leverage K-Means to compress an image. This will involve reducing the number of colors in the image by grouping similar colors into clusters and representing each pixel by its cluster's centroid color. The paper is structured into three main sections.

In this assignment, Section [2] outlines the methodology for implementing the K-Means Algorithm, and Section [3] presents the experimental results and analyzes the relationship between clusters and the distortion cost function. Finally, Section [4] offers conclusions drawn from the findings and discusses the implications for practical applications.

II. Method

In this section, a sequential structured approach was followed to implement the K-Means clustering algorithm without using any machine learning library and apply it to both data clustering and image compression tasks.. Fig. 1 shows the workflow of this study.

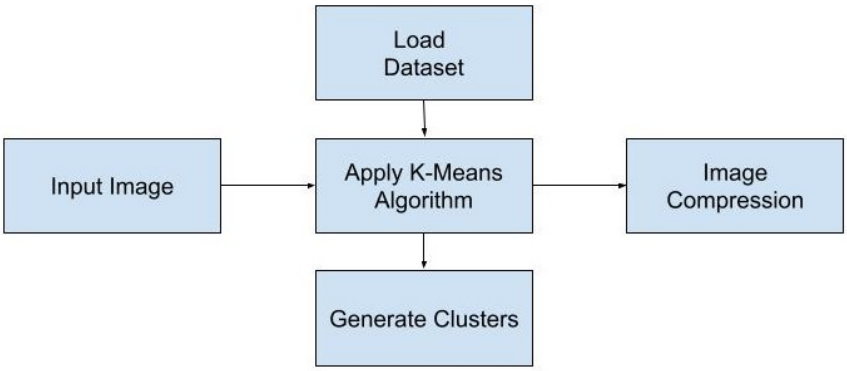

Fig. 1. Methodology Block Diagram

1. Load Dataset: A two-dimensional dataset named 'kmeans.npy' and an image was used for the data clustering visualization and image compression task. This 'kmeans.npy' dataset was loaded using the NumPy library and used as the input for the K-Means algorithm, where each data point represents a coordinate in a two-dimensional space.

```
X = np.load('G:\\Assignment\\Assignment-6\\kmeans2d.npy')
print("First five elements of X are:\n", X[:5])
print('The shape of X is:', X.shape)
```

2. Apply K-Means Algorithm: A custom K-Means algorithm was implemented using NumPy. The steps of the algorithm include:

- **Initialization:** Randomly initialize k cluster centroids from the input dataset.
- **Assignment:** Assign each data point to the nearest cluster based on the lowest Euclidean distance.
- **Update:** Recalculate the cluster centroids of each cluster as the mean of the data points assigned to it.
- **Iteration:** Repeat the assignment and update steps until convergence (centroids no longer change significantly) or a maximum number of iterations is reached.

The algorithm was configured to run multiple initializations (100 by default) and select the clustering solution with the lowest distortion cost (sum of squared distances between data points and their assigned centroids).

```
import numpy as np

def kmeans(data, k, max_iterations=100, num_inits=100):

    best_centroids = None
    best_labels = None
    min_cost = np.inf

    for _ in range(num_inits):
        centroids = data[np.random.choice(data.shape[0], k, replace=False), :]

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

            new_centroids = np.zeros_like(centroids)
            for i in range(k):
                if np.any(labels == i):
                    new_centroids[i] = data[labels == i].mean(axis=0)
                else:
                    new_centroids[i] = data[np.random.choice(data.shape[0])]

            if np.all(centroids == new_centroids):
                break

            centroids = new_centroids

        cost = np.sum(np.square(distances[np.arange(data.shape[0]), labels]))

        if cost < min_cost:
            min_cost = cost
            best_centroids = centroids
            best_labels = labels

    return best_centroids, best_labels, min_cost
```

3. Generate Clusters: The implemented K-Means algorithm was applied to the X features of the dataset using k = 3 clusters, and the resulting clusters of the corresponding data points and their adjacent cluster centroids were plotted using the matplotlib library. The hyperparameter k was assigned the value of 3 based on the distortion cost function vs k clusters curve, as shown in Figure 2. Here, it was observed that the curve mirrors an elbow, for which the elbow method was considered to assess the optimal number of clusters, where the elbow of the curve is at k=3. The clusters were visualized by plotting the data points with different colors representing their cluster assignments and marking the cluster centers with 'x' markers.

```
original_img = plt.imread('G:\\Assignment\\Assignment-6\\bird_small.png')
plt.imshow(original_img)
print("Shape of original_image is: ", original_img.shape)

original_img = original_img / 255
X_img = np.reshape(original_img, (original_img.shape[0] * original_img.shape[1], 3))
```

```

K = 4
cluster_centers, labels, min_cost = kmeans(X_img, K)
X_recovered = cluster_centers[labels].reshape(original_img.shape)

fig, ax = plt.subplots(1,2, figsize=(8,8))
plt.axis('off')
ax[0].imshow(original_img*255)
ax[0].set_title('Original')
ax[0].set_axis_off()
ax[1].imshow(X_recovered*255)
ax[1].set_title('Compressed␣with␣%d␣colours'%K)
ax[1].set_axis_off()
```

4. Image Compression: A bird image was used as the input for the K-Means algorithm, where each data point represents a coordinate in a two-dimensional space. The image was loaded using the matplotlib library and normalized by dividing the pixel values by 255 to bring them within a range of 0 to 1. The image was then reshaped into a two-dimensional array, where each row represents a pixel, and the columns represent the red, green, and blue color channels. The K-Means algorithm was applied to the reshaped image data (X_img) with varying numbers of clusters (K = 4, 8, 16, 32). Each pixel's color was replaced with the centroid color of its assigned cluster, effectively reducing the number of unique colors in the image. The compressed images were reconstructed from the cluster assignments and centroids. The original and compressed images were displayed side by side to visualize the impact of compression with different K values.

```
k = 3
cluster_centers, labels, min_cost = k-means(X, k)
print('cluster_center', cluster_centers.shape)
print('labels', labels.shape)
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], marker='x', c='r',
linewidths=3)
```

III. Experiment Results

A. Data Analysis

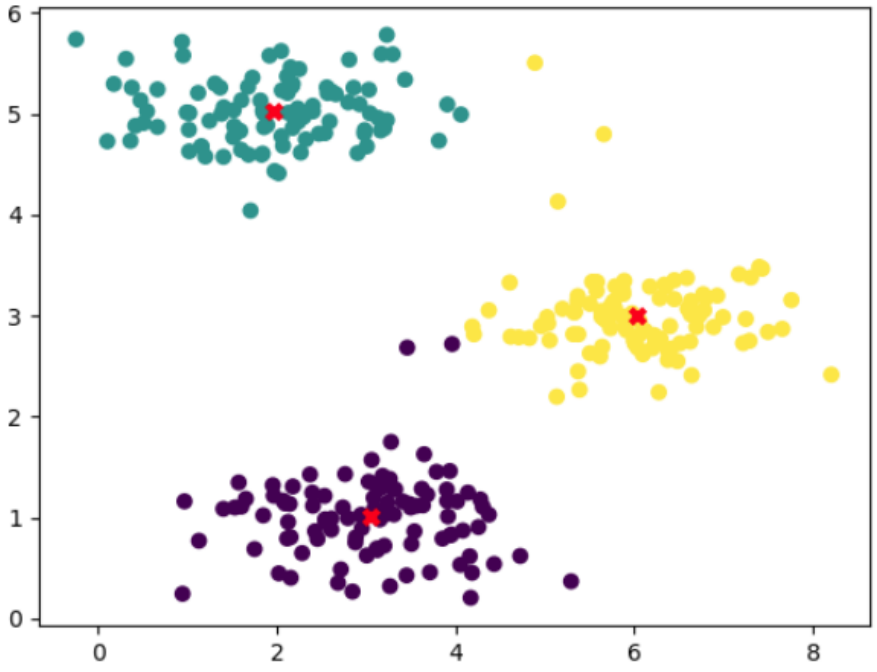

Fig. 2. Cluster Assignments and Centroids (K-Means, K=3)

Fig. 3. Image Compression using K-Means Clustering with Varying Color Palettes

- **Distortion cost function**

$$J = \sum_{i=1}^{m} \sum_{j=1}^{K} ||x^{(i)} - \mu_j||^2 \tag{1}$$

Here, J represents the distortion cost, m is the number of data points, K is the number of clusters, $x^{(i)}$ represents the i-th data point, μ_j represents the centroid of the j-th cluster, $||\cdot||$ denotes the Euclidean distance, $\sum_{i=1}^{m} \sum_{j=1}^{K}$ represents the summation over all data points and clusters, and $||x^{(i)} - \mu_j||^2$ calculates the squared Euclidean distance between each data point and its assigned centroid.

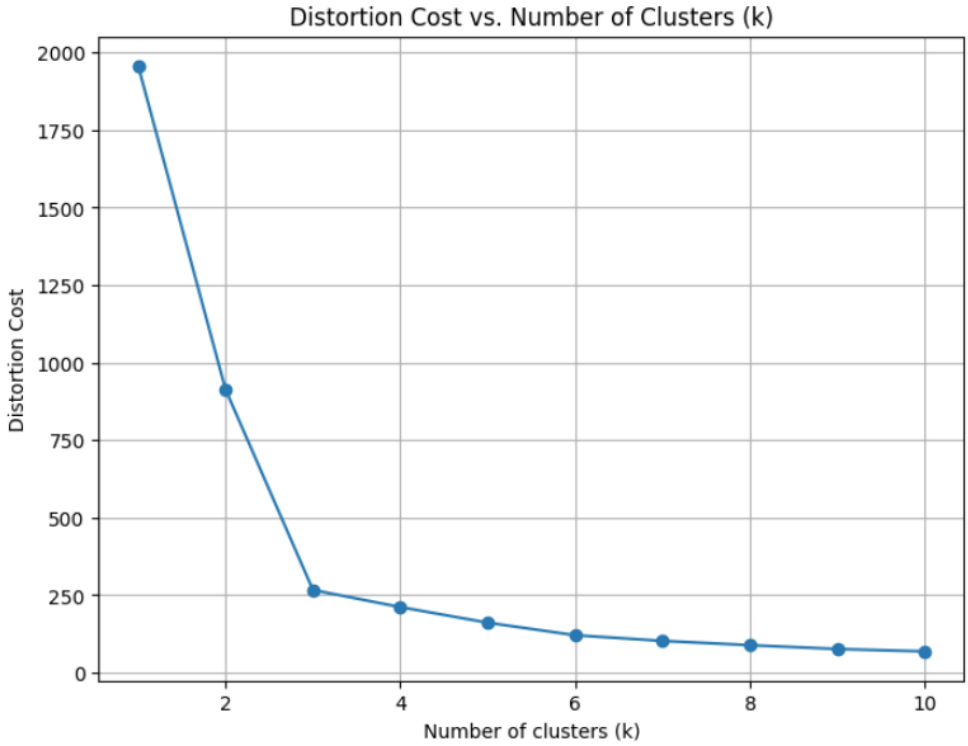

Fig. 4. Distortion Cost vs Number of Clusters (k)

IV. Discussion

In this assignment, we successfully implemented the K-Means algorithm without using any machine learning library and applied it to both data clustering and image compression tasks. Moreover, we try to visually analyze the relationship between the distortion cost function and the number of cluster centroids in order to find the optimal cluster centroid k. It was observed that for both cases, the distortion function decreases non-linearly with the increase in the number of cluster centroids as the clusters become smaller. Furthermore, in the case of image compression, it was observed that there is a trade-off relationship between the quality of the image and compression. As the value of k (color palette) increases, the quality of the image increases, while decreasing the compression.