Implementing K-Means Clustering

Maliha Mahjabin Chowdhury
Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
160204043@aust.edu

***Abstract*—K-means clustering is one of the simplest and popular unsupervised machine learning algorithms. The objective of K-Means is very simple: group similar data points together and discover underlying patterns to get a meaningful intuition of the structure of the data. K-Means model looks for a fixed number of clusters, K in order to achieve this objective. The model is used in a variety of applications such as market research, document clustering, image segmentation, data analysis, image compression, etc. A cluster refers to a collection of data points aggregated together because of certain similarities. Each cluster is defined by its centroid, the imaginary or real location representing the center of the cluster.**

***Index Terms*—Centroid, Mean, Cluster, Squared Eucildean Distance.**

I. Introduction

K-Means Clustering is an unsupervised learning algorithm, that means, the data are not labeled. Because of this, there is no train set neither test set. The model starts with randomly selected K numbers of centroids and then performs repetitive calculations to optimize the positions of the centroids. Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares. The model targets to maximize the inter-cluster distances and minimize the intra-cluster distances. The algorithm terminates when the centroids remain unchanged and no data point changes its assigned cluster anymore, indicating that the clustering is successful.

The rest of the report is organised in the following manner: section 2 discusses our experimental design. In section 3, we illustrate the results and present our findings. Finally, section 4 summarizes the report and section 5 demonstrates the algorithm implementation.

II. Methodology

In this section, we illustrate the basic algorithm of K-Means Clustering. Additionally, we demonstrate the implementation of the algorithm.

A. Algorithm

The K-Means Clustering algorithm is given below.

Algorithm 1 K-Means Clustering

```
1: Read dataset
2: Observe the dataset and initialize k, number of clusters
3: Randomly select K number of centroids as clusters' means
4: Calculate distance between each data point and every
   centroid using the following formula:
   $$d = \sqrt{(x_1 - X_1)^2 + (x_2 - X_2)^2 + ..... + (x_i - X_i)^2} \quad (1)$$
5: Assign each data point to its nearest cluster
6: Define the new centroids by calculating means of each
   cluster
7: Repeat step 4 to 6 until the centroids stabilize and data
   points stop changing assigned clusters
```

B. Implementation Process

We design the classifier in Python. This is accomplished using the most popular libraries: NumPy [1], pandas [2] and Matplotlib [3]. NumPy and pandas help processing the dataset and performing necessary operations and calculations respectively. We visualize the results using Matplotlib. We cluster the data points using the equation 1. We use different colors and markers for different clusters. Figure 1 presents clustered data on a plot.

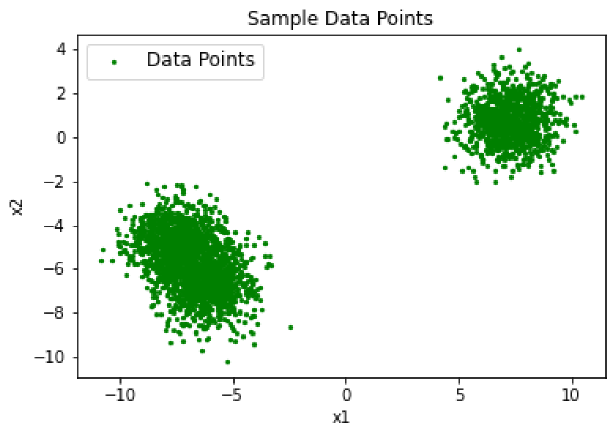

Fig. 1. Given Dataset

III. Result Analysis

We plot the data points and observe them in figure 1. For the given dataset, value 2 seems a good choice for K. However, figure 2, figure 3 and figure 4 show clustering for different values of K: 2, 3 and 5 respectively.

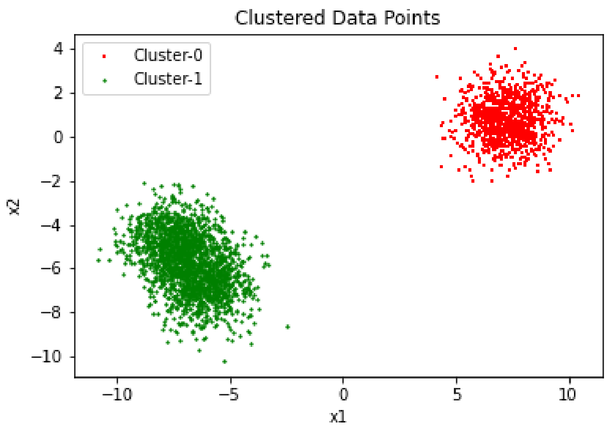

Fig. 2. Clustered Data Points (K = 2)

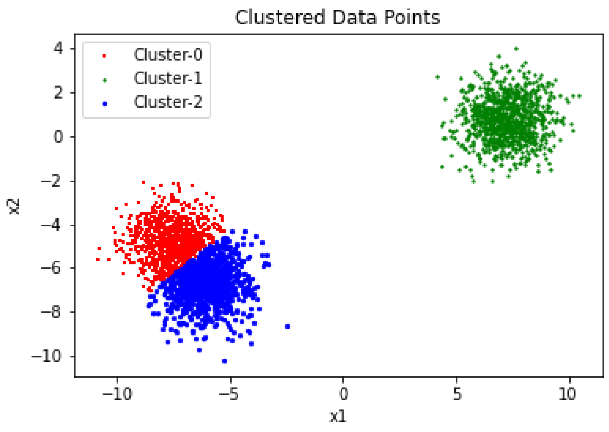

Fig. 3. Clustered Data Points (K = 3)

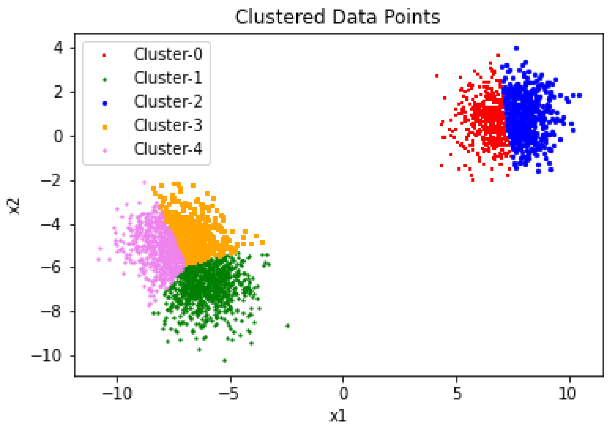

Fig. 4. Clustered Data Points (K = 5)

We use equation 1 to calculate distance between data points and centroids. When the centroids stay unchanged, the algorithm terminates. Upon thoroughly observing figure 2, figure 3 and figure 4, value 2 for works best for the given dataset. Values 3 and 5 seem to cluster data unnecessarily. Different colors and markers represent different cluster for easier understanding. As this is unsupervised learning, we cannot calculate the accuracy of model's performance, but intuitively, the clustering seems to be successful.

IV. Conclusion

K-Means Clustering is an extensively used technique for data analysis. It is easy to understand and delivers results quickly. However, the model has some limitations. The very first problem is determining k, number of clusters. For the given dataset, we intuitively choose 2 and it performs satisfactorily. But this may turn out very difficult to predict value for K in case of large and complex datasets. Also, the model performs poorly with global clusters. The algorithm is sensitive to outliers and initial centroids.

V. Code

choose The model is designed in Python. The implementation is given below.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random


dataset = pd.read_csv('data_k_mean.txt',
sep = '␣', header = None)


dataset = np.array(dataset)


plt.scatter([dataset[i][0]
for i in range(len(dataset))],
[dataset[i][1]
for i in range(len(dataset))],
label = 'Data␣Points',
marker = 'o', c = 'green', s = 5)


plt.xlabel("x1")
plt.ylabel("x2")
plt.title("Sample␣Data␣Points")
plt.legend(loc = 'upper␣left',
fontsize = 12)
plt.show()


print('Enter␣the␣number␣of␣clusters␣:␣')
k = int(input())
mean = []
```

```
for i in range(k):
  x = random.randint(0, len(dataset))
  mean.append(dataset[x])


mean = np.array(mean)


iteration = 0


while iteration < 100000:
  cluster = []
  for i in range(len(dataset)):
    x = dataset[i]
    distance =
    np.iinfo(int_type = int).max
    for j in range(k):
      temp_distance =
      ((mean[j][0]-x[0])**2 +
      (mean[j][1]-x[1])**2)**0.5
      if temp_distance < distance:
        distance = temp_distance
        temp_cluster = j
    cluster.append(temp_cluster)


  temp_mean = []


  for i in range(k):
    x = [j for j in range(len(cluster))
    if cluster[j] == i]
    temp_dataset =
    np.array([dataset[j] for j in x])
    temp_mean.append(np.mean(temp_dataset,
    axis = 0))


  temp_mean = np.array(temp_mean)
  mean = np.array(mean)
  if np.any(temp_mean - mean) == True:
    mean = temp_mean
  else:
    break
  iteration = iteration + 1


print('Number␣of␣Iterations␣Needed␣:␣',
iteration)


color = ['red', 'green', 'blue',
'orange', 'violet']
marker = ['+', '1', 'o', 's', '2']
```

```
for i in range(k):
  x = [j for j in range(len(cluster))
  if cluster[j] == i]
  plt.scatter([dataset[j][0] for j in x],
  [dataset[j][1] for j in x],
  marker = marker[i], color = color[i],
  s = 5, label = 'Cluster-'+str(i))


plt.xlabel("x1")
plt.ylabel("x2")
plt.title("Clustered␣Data␣Points")
plt.legend(loc = 'best', fontsize = 10)
plt.show()
```

References

[1] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[2] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020.

[3] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.