Implementing K-Means Clustering

Ahmad Subaktagin Jabir
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh.
160204061@aust.edu

***Abstract*—K-Means Clustering Algorithm is an algorithm that is used to solve the clustering problems in machine learning or data science. This algorithm is used in market segmentation, document clustering, image segmentation and image compression, etc. We will see in this experiment how this algorithm works.**

***Index Terms*—Euclidean Distance, Machine Learning, Unsupervised Learning, Centroid, Mean, Variance.**

I. Introduction

K-Means Clustering is an Unsupervised Learning algorithm, which groups the unlabeled dataset into different clusters. Here K defines the number of predefined clusters that need to be created in the process, as if $K = 2$, there will be two clusters, and for $K = 3$, there will be three clusters, and so on. This algorithm mainly performs two tasks. One is to determine the best value for K centroids by an iterative process. Another is to assign each data point to its closest k-center. Data points which are near to the particular k-center, create a cluster.

II. Experimental Design / Methodology

A. **Plotting All Points:** We will read from the "*data_k_mean.txt*" file and we will plot all points. After plotting, I got the following figure:

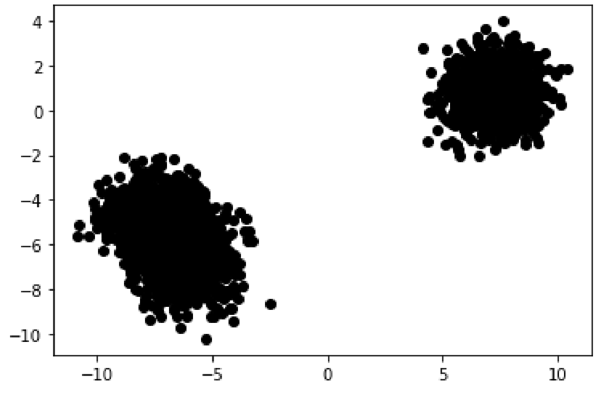


Fig. 1. Plotting All Data Points

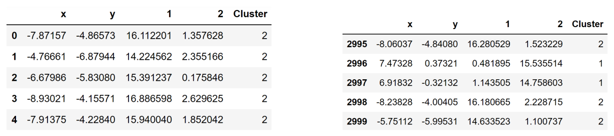
B. **Implementing K-Means Clustering Algorithm:** First of all, we will select the number of K to decide the number of clusters. Then we will select random K points or centroids from our dataset. After that, we will assign each data point to their closest centroid, which will form the predefined K clusters. For this, we will use Euclidean Distance to calculate Distance between Centroid and Data Point. The Formula is given below:

$$\text{Euclidean Distance} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Then we will calculate the variance and place a new centroid of each cluster. We will then reassign each datapoint to the new closest centroid of each cluster. If any reassignment occurs, we will calculate the variance and place a new centroid of each cluster again. Otherwise, our model is ready.

III. Result Analysis

1) **Determining Cluster of Data Points:** Using $k = 2$, $(-5.34386, -7.11979)$ and $(-5.20518, -7.98521)$ as initial centroids I got results like following:



	x	y	1	2	Cluster
0	-7.87157	-4.86573	16.112201	1.357628	2
1	-4.76661	-6.87944	14.224562	2.355166	2
2	-6.67986	-5.83080	15.391237	0.175846	2
3	-8.93021	-4.15571	16.886598	2.629625	2
4	-7.91375	-4.22840	15.940040	1.852042	2

	x	y	1	2	Cluster
2995	-8.06037	-4.84080	16.280529	1.523229	2
2996	7.47328	0.37321	0.481895	15.535514	1
2997	6.91832	-0.32132	1.143505	14.758603	1
2998	-8.23828	-4.00405	16.180665	2.228715	2
2999	-5.75112	-5.99531	14.633523	1.100737	2

Fig. 2. First and Last 5 rows After Applying K-Means Clustering.

2) **Plotting all corresponding points with different colors:** I used red and green color to separate clusters, The Figure is given below:

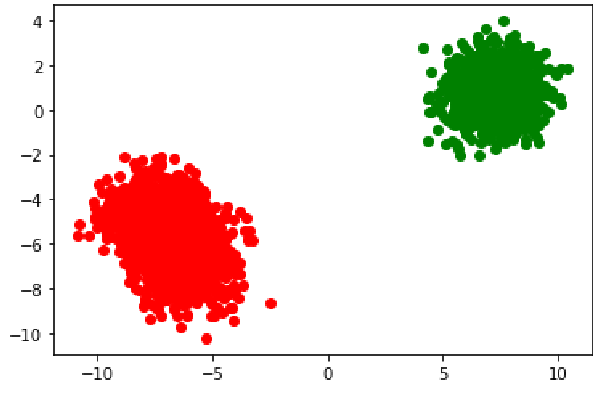


Fig. 3. Plotting All Data Points with Different Colors

IV. Conclusion

The main purpose of K-Means Clustering Algorithm is to minimize the sum of distances between the data. But there are some weaknesses of this algorithm. One of them is this algorithm is only applicable when the mean is defined. Also the user needs to specify k. Besides that , this algorithm is sensitive to outliers. In spite of these weaknesses, this algorithm is the most popular clustering algorithm as it is easy to understand and to implement. Also, the time complexity of this algorithm is $O(tkn)$, which makes this algorithm more efficient (n is the number of data points, k is the number of clusters, t is the number of iterations). Since both k and t are small, this algorithm is considered as a linear algorithm.

V. Algorithm Implementation / Code

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

train_set = []
with open('data_k_mean.txt','r') as file:
    new_reader = csv.reader(file,delimiter=' ')
    for row in new_reader:
        train_set.append(row)

for i in range(len(train_set)):
    for j in range(len(train_set[i])):
        train_set[i][j] = float(train_set[i][j])

x = []
for train in train_set:
    x.append(train[0])
y = []
for train in train_set:
    y.append(train[1])

train_set_array = np.array(train_set)

plt.plot(train_set_array[:,0:1],train_set_array[:,
    1:], linestyle = '', marker='o', color='k')
plt.show()

X = pd.DataFrame({'x': x,
                  'y': y})

k = int(input("Enter the Value of K: "))
Centroids = (X.sample(n=k))
Centroids

diff = 1
j=0

while(diff!=0):
    XD=X
    i=1
    for index1,row_c in Centroids.iterrows():
        ED=[]
        for index2,row_d in XD.iterrows():
            d1=(row_c["x"]-row_d["x"])**2
            d2=(row_c["y"]-row_d["y"])**2
            d=np.sqrt(d1+d2)
            ED.append(d)
        X[i]=ED
        i=i+1

    C=[]
    for index,row in X.iterrows():
        min_dist=row[1]
        pos=1
        for i in range(k):
            if row[i+1] < min_dist:
                min_dist = row[i+1]
                pos=i+1
        C.append(pos)
    X["Cluster"]=C
    Centroids_new = X.groupby(["Cluster"]).mean()[["
    x","y"]]
    if j == 0:
        diff=1
        j=j+1
    else:
        diff = (Centroids_new['x'] - Centroids['x'])
    .sum() + (Centroids_new['y'] - Centroids['y']).
    sum()
    Centroids = X.groupby(["Cluster"]).mean()[["x","
    y"]]

X.head()
X.tail()

color=['green','red','cyan']
for l in range(k):
    data=X[X["Cluster"]==l+1]
    plt.scatter(data["x"],data["y"],c=color[l])
plt.show()
```