Implementing K-Means Clustering

Anas Sikder
Department of CSE
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
160204021@aust.edu

***Abstract*—My task for this experiment was to classify some unlabeled data points with the help of the K-Means Clustering. I implemented the algorithm using the distance formula i,e. Euclidean distance and labelling the data points where I vary the value of K and observe the clusters of data points of given samples. It works fine in my experimental data sets.**

***Index Terms*—K-Means ,Clustering ,data points, Euclidean distance , label , k integer value etc.**

I. Introduction

K means clustering is one of the simplest and popular unsupervised machine learning algorithms. In an unsupervised algorithm, the label of the data points is unknown. To label data points clustering samples has been used.

A. Motivation

My main goal is to implementing K-Means Clustering on given all data points and labelled them. Also, observe the change of labels of data points while changing the value of k.

B. Theory

1) Clustering: A cluster refers to a collection of data points aggregated together because of certain similarities.
The K-means algorithm identifies the k number of centroids, and then allocates every data point to the nearest cluster while keeping the centroids as small as possible.

2) How the K-means algorithm works: To process the learning data, the K-means algorithm in data mining starts with the first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids. It halts creating and optimizing clusters when either:

- The centroids have stabilized — there is no change in their values because the clustering has been successful.
- The defined number of iterations has been achieved.

3) Euclidean distance: For two data points P1(x_1, y_1) and P2(x_2, y_2) the distance using Euclidean formula calculated as:

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

II. Experimental Design / Methodology

A. Task 1

I took the data points and plot all the points with colour markers.

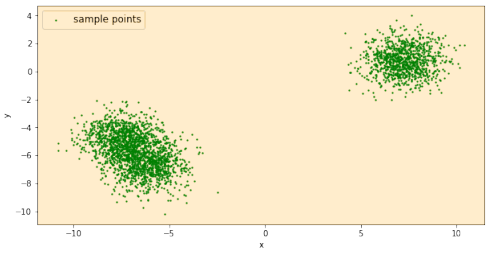

Fig. 1. Data Points

B. Task 2

Then I performed the k-means clustering algorithm applying Euclidean distance as a distance measure on the given data points with k having random integer values.

C. Task 3

Finally colouring the corresponding points on the clusters with different colours.

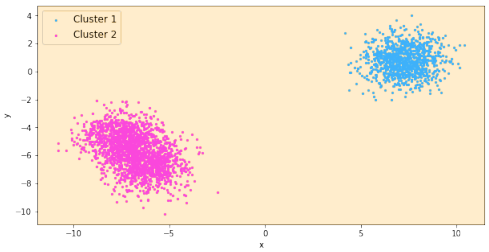

Fig. 2. Data Points with k = 2 and 2 Cluster

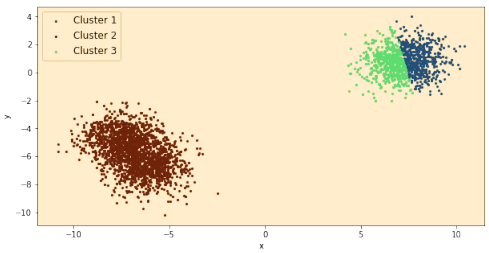

Fig. 3. Data Points with k = 3 and 3 Cluster

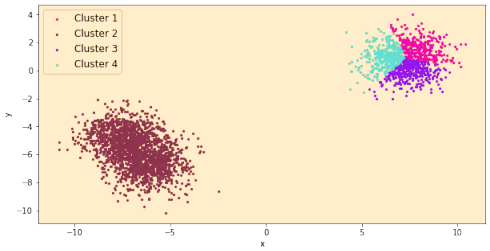

Fig. 4. Data Points with k = 4 and 4 Cluster

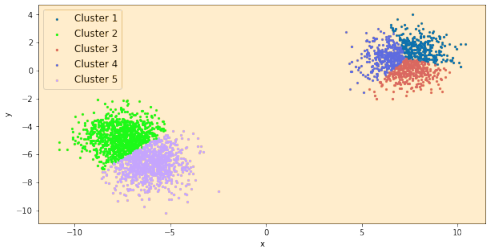

Fig. 5. Data Points with k = 5 and 5 Cluster

III. Result Analysis

For the initial value of centroids, I took the help of the python random choice function.

A. K Means clustering having $K = 2$

When I took the value of $K = 2$ I found **Cluster 1 having 1000 data points and Cluster 2 having 2000 data points.**

K = 2	
Cluster 1	1000
Cluster 2	2000

B. K Means clustering having $K = 3$

When I took the value of $K = 3$ I found **Cluster 1 having 474 data points , Cluster 2 having 2000 data points and Cluster 3 having 526 data points.**

K = 3	
Cluster 1	474
Cluster 2	2000
Cluster 3	526

C. K Means clustering having K=4

When I took the value of $K = 4$ I found **Cluster 1 having 294 data points , Cluster 2 having 2000 data points , Cluster 3 having 347 data points and Cluster 4 having 359 data points.**

K = 4	
Cluster 1	294
Cluster 2	2000
Cluster 3	347
Cluster 4	359

D. K Means clustering having K=5

When I took the value of $K = 5$ I found **Cluster 1 having 294 data points , Cluster 2 having 1008 data points , Cluster 3 having 347 data points , Cluster 4 having 359 data points and Cluster 5 having 992 data points.**

K = 5	
Cluster 1	294
Cluster 2	1008
Cluster 3	347
Cluster 4	359
Cluster 5	992

I can take $k = 6, 7, ..., n$ for further evaluation of the Clustering of data points.

IV. Conclusion

K-Means clustering performs well in my experimental data points, although my given data sets have 3000 data points. So algorithm scales to large data sets. So the was no problem with having convergence with value k with a small one.But **when I took the value of k larger algorithm took much time to convergence. Also, k means clustering being dependent on initial values of centroids called K-means seedings.**

V. Algorithm Implementation / Code

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

data = pd.read_csv('data_k_mean.txt'
,sep=" ",header=None)
data=data.to_numpy()

f, ax = plt.subplots()
f.set_figheight(5)
f.set_figwidth(10)
ax.scatter(data[:, 0], data[:, 1],
marker='.',s = 7, color = 'green',
label = 'sample points')
legend = ax.legend(loc='upper left
', shadow=False, fontsize='large'
,labelspacing=0.5)
legend.get_frame().set_facecolor('None')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_facecolor("orange")
ax.patch.set_alpha(0.2)

# Euclidean Distance
def euclidean_distances(data,centroids):
    t = np.zeros([len(data),k])
    for j in range(len(centroids)):
        x = []
```

```
        for i in range(len(data)):
                dis = np.sqrt(np.power
                ((data[i][0]
                - centroids[j][0]),2)+
                np.power((data[i][1]
                - centroids[j][1]),2))
                x.append(dis)
        t[:,j] = np.asarray(x)
    return t

def kmeans(data, k, maxiter):
    np.random.seed(25)
    centroids=data[np.random.choice(
    data.shape[0], k), :]
    #print(centroids)
    count = 1

    for itr in range(maxiter):
        count = count + 1
        data_dis = euclidean_distances
        (data, centroids)
        cluster_assignment = np.argmin
        (data_dis, axis = 1)
        #print(cluster_assignment)
        new_centroids = np.array([data
        [cluster_assignment == i].mean(axis = 0)
        for i in range(k)])

        xx = centroids == new_centroids
        xxx = xx.all()
        if(xxx==True):
            break;
        count = count + 1
        centroids = new_centroids
    print('iteration = ',count)

    return cluster_assignment

#k=1 to n
k = int(input("Enter your value of k: ") )
label = kmeans(data, k, maxiter = 1000)
print(label)
print(np.bincount(label))
dt = np.zeros([3000,3])
dt[:,0]=data[:,0]
dt[:,1]=data[:,1]
dt[:,2]=label[:]
print(dt)

from random import randint
colors = []

for i in range(50):
    colors.append('#%06X' % randint(0, 0xFFFFFF))
from matplotlib import cm
f, ax = plt.subplots()
f.set_figheight(5)
f.set_figwidth(10)
for i in range(k):
    arr = np.where(dt[:,2]==i)
    result = dt[arr]
    ax.scatter(result[:, 0], result[:, 1],s= 20,
    marker='.',label = 'Cluster'+ ' '+str(i+1),
    c = colors[i])
    legend = ax.legend(loc='upper left',
    shadow=False,
    fontsize='large',labelspacing=0.5)
legend.get_frame().set_facecolor('None')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_facecolor("orange")
ax.patch.set_alpha(0.2)
```