Implementing K-Means Clustering

Md. Zahid Fesabelilla
Computer Science and Engineering. Ahsanullah University of Science and Technology
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
160204082@aust.edu

***Abstract*—This document is about implementation of the K-Means Clustering.The task is to,take input from the given source data file and plot all the points.And then perform the k-means clustering algorithm applying Euclidean distance as a distance measure on the given dataset with k = user input.Then color the corresponding points on the clusters with different colors.**

***Index Terms*—K-Means Clustering, K-Means algorithm, Clustering etc.**

I. Introduction

Clustering is finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups.
For the clustering we use the most famous algorithm name K-Means clustering.
Now
K-means Algorithm
Given K, the K-means algorithm works as follows:

1) Choose k(random) data points(seeds) to be the initial centroids, cluster centers
2) Assign each data point to the closest centroid
3) Re-compute the centroids using the current cluster memberships
4) If the convergence criterion is not met, repeat steps 2 and 3.

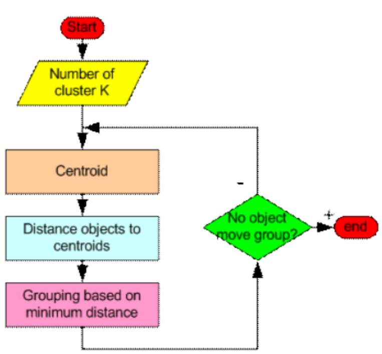

Fig. 1. Flow Chat of K-means algorithm

K-means convergence (stopping) criterion

1) No (or minimum) re-assignments of data points to different clusters, or
2) No (or minimum) change of centroids, or
3) Minimum decrease in the sum of squared error(SSE)

II. Experimental Design / Methodology

Firstly,We take input from the given source data file 'data_k_mean.txt' and plot all the points.

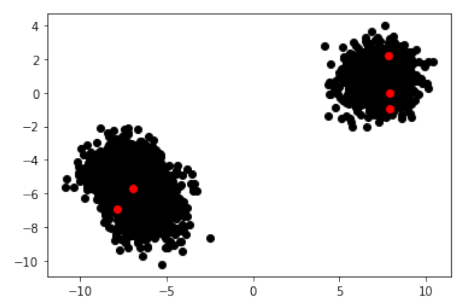

Fig. 2. Plot all the points

Here, red dots are k(random) data points(seeds) to be the initial centroids, cluster centers.

Secondly, then perform the k-means clustering algorithm applying Euclidean distance as a distance measure on the given dataset with k = user input.

And finally, Color the corresponding points on the clusters with different colors.

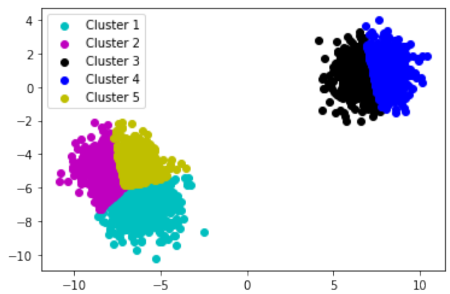

Fig. 3. Color the corresponding points on the clusters with different colors.

III. Result Analysis

for this dataset, number of iteration is too large. For this reason, algorithms take much more time to converge. So, I take a fixed number of iteration states and then converge the algorithm. Then we finally classify the K number of clusters.

IV. Conclusion

K-means is the most popular clustering algorithm. K-means clustering performs usually well.It is very efficient.Its solution can be used as a starting point for other clustering algorithms. It terminates at a local optimum if SSE is used. The global optimum is hard to find due to complexity.

V. Algorithm Implementation / Code

```
# -*- coding: utf-8 -*-
"""patt.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1YTl-
    g3Kzyt0c6c3MvoymaNrIteglLPPL
"""

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

from google.colab import drive

drive.mount(’/content/gdrive’)

root_path = ’/content/gdrive/My Drive/
    PatternAssignment5/’

df = pd.read_csv(’/content/gdrive/My Drive/
    PatternAssignment5/data_k_mean.txt’,sep=" ",
    header=None)

K = input("Enter number of clusters: ")

K = int(K)
# Select random observation as centroids
Centroids = (df.sample(n=K))
plt.scatter(df[0],df[1],c=’black’)
plt.scatter(Centroids[0],Centroids[1],c=’red’)
plt.show()

print(df)

df = df.to_numpy();

centroids = df[np.random.choice(df.shape[0],K), :]
minm = np.zeros(K)

centroids

print(df)

value = [ [ i[0] , i[1] , 5000 ] for i in df]

print(value)

for z in range(300):

    cnt = 0
    for i in range(len(value)):

        for j in range(K):
            minm[j] = np.sqrt( ((value[i][0] -
    centroids[j][0])**2) + ((value[i][1] - centroids
    [j][1])**2) )

        ## Return the minimum of an array
        temp1 = np.where(minm == np.amin(minm))
        #print("Temp1 : ", temp1,"\n")
        temp1 = np.array(temp1)

        if(value[i][2] != temp1.item(0)):
            value[i][2] = temp1.item(0)
            #print("Points : ",value,"\n")
            cnt = cnt + 1

    if(cnt == 0):
        break
    for i in range(K):
        temp = [[x[0],x[1]] for x in points if x
    [2]==i]
        temp = np.array(temp)
        centroids[i] = [ sum(x)/len(x) for x in zip
    (*temp)]

print(value)

centroids = pd.DataFrame(centroids)

centroids

plt.figure(figsize = (20, 30))

color = [’c’,’m’,’k’,’b’,’y’,’g’,’r’]
marker = [’o’,’o’,’o’,’o’,’o’,’o’,’o’,’o’]

a,b = plt.subplots()

for i in range(K):
    temp=[[j[0],j[1]] for j in value if j[2]==i]
    temp=np.array(temp)
    lvl ="Cluster " + str(i+1)
    b.scatter(temp[:,0], temp[:,1], marker = marker[
    i], color = color[i], label = lvl)

legend = b.legend()

plt.show()
```