K-Means Clustering

Md. Yeasir Arafat
Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh
160204093@aust.edu

***Abstract*—K-means clustering is a type of unsupervised learning, which is used when we have unlabeled data.**

***Index Terms*—Clustered the unsupervised data based on given k value.**

I. Introduction

The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. The algorithm works iteratively to assign each data point to one of K groups based on the features that are provided. Data points are clustered based on feature similarity.

II. Experimental Design / Methodology

1. **Plotting sample points:** 3000 Sample points are given to 'data-k-means.txt' file. We need to plot all the sample points. First I read the file from google drive then extract the data for plotting. Finally I plotted the points using matplotlib package.

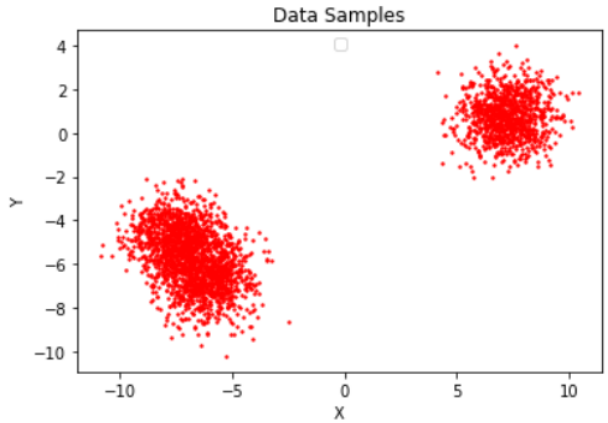

Fig 1: Given Sample points

2. **Perform the k-means clustering algorithm:** I take user input of the value of K. Then randomly pick k-th clustered centroid from the given dataset. Then calculate the euclidean distance between clustered centroid and each given dataset and assigned that point to a cluster for which the euclidean distance is minimum. After assigning all the data point to cluster, we update the cluster centroid based on assigned data points. Cluster centroid update is simply the average of those assigned cluster data points. This process continues until the cluster centroids are remain same for previous iteration.

3. **Plotting clustered points:** After clustering each data points, I plotted those clustered points with different colored marker as follows:

Fig 2: Clustered Sample points

III. Result Analysis

This algorithm is relatively easy to implement and works for large dataset and also this algorithm can easily adapts to new examples.But in this algorithm it is difficult to predict K-value. Initial seeds have a strong impact on the final results. Rescaling the dataset will completely changed the results.

IV. Conclusion

In conclusion, the K-means clustering technique is a simple quick algorithm that can be applied to large datasets to separate them into different partitions; analysis of these partitions may provide a better characterization of neck pathologies.

V. Algorithm Implementation / Code

```
"""k means clustering.ipynb
Read file from gdrive
"""

import numpy as np
import math
from google.colab import drive
```

```
drive.mount('/content/gdrive')
# change working directory on the drive
%cd '/content/gdrive/My Drive/Data/'

# read train.txt file line by line
with open('data_k_mean.txt', "r") as file:
    FileasList = file.read().splitlines()

# split the string and store it into another list
    classwise
data = []
x = []
y = []
for i in range(len(FileasList)):
  data.append(FileasList[i].split())
  x.append(float(data[i][0]))
  y.append(float(data[i][1]))

for i in range(len(data)):
  data[i] = [float(data[i][0]),float(data[i][1])]
data = np.array(data)

"""Plotting all sample points from train data """

import matplotlib.pyplot as plt

plt.scatter(x, y, c = 'r', marker = 'o', s = 2)
plt.title("Data Samples")
plt.xlabel("X")
plt.ylabel("Y")
plt.legend(loc = 'upper center')

"""Euclidean Distance"""

def eu_dis(a,b):
  dis = math.sqrt((a[0] - b[0])**2 + (a[1] - b[1])
    **2)
  return dis

"""Assigned cluster's Index finder"""

def find_index(list,key):
  indexes = []
  for i in range (len(list)):
    if(list[i] == key):
      indexes.append(i)
  return indexes

"""Cluster Update"""

def cluster_update(k, data, cluster_assign):
  cluster = []
  for i in range(k):
    index = find_index(cluster_assign ,i+1)
    temp = np.array([0,0])
    for j in index:
      temp = temp + data[j]
    temp = temp / len(index)
    cluster.append(temp)
  return cluster

"""**K means Clustering**"""

def k_means_clustering(k, data):
  cluster = []
  for i in range(k):
    cluster.append(np.array([x[i],y[i]]))

  flag = 0
  while(flag != k):
    flag = 0
    cluster_assign = []
    for d in data:
      c_dis = []
      for i in range(k):
        c_dis.append(eu_dis(cluster[i],d))
      minpos = c_dis.index(min(c_dis)) + 1
      cluster_assign.append(minpos)

    # cluster update
    temp = cluster
    cluster = cluster_update(k,data,cluster_assign)

    for i in range(k):
      if(temp[i][0] == cluster[i][0] and temp[i][1]
    == cluster[i][1]):
        flag = flag + 1

  return cluster_assign

"""**Clustering and plotting**"""

k = int(input("Enter the value of K: "))
clustered_label = k_means_clustering(k,data)

x1 = []
y1 = []
x2 = []
y2 = []

for i in range(len(clustered_label)):
  if(clustered_label[i] == 1):
    x1.append(data[i][0])
    y1.append(data[i][1])
  else:
    x2.append(data[i][0])
    y2.append(data[i][1])

plt.scatter(x1, y1, c = 'r', marker = 'o', label = '
    Class 1', s = 2)
plt.scatter(x2, y2, c = 'g', marker = 'o', label = '
    Class 2', s = 2)
plt.title("Clustered Data")
plt.xlabel("X")
plt.ylabel("Y")
plt.legend(loc = 'upper left')
```