Implementing K-Means Clustering

Devopriya Tirtho
16.02.04.033
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology
Dhaka, Bangladesh

***Abstract*—'Machine Learning' is the kind of learning which tries to train a device according to learning to work smarter. In Machine Learning, there are two types of learning. One is Supervised Learning and the other one is Unsupervised Learning. In first intuition, we may think without any labelling how we can train the device to learn something. There are some advanced algorithms which are based on Unsupervised Learning. K-Means Clustering is one of them. In this experiment, we will implement the K-Means Clustering algorithm.**

***Index Terms*—Machine Learning, K-Means Cluster, Unsupervised Learning, Supervised Learning.**

I. Introduction

K-Means Clustering is a form of **Unsupervised Learning**. We have already known about the **Supervised Learning**, now it is the time to work with **Unsupervised Learning**. In **Unsupervised Learning**, there is no labelled data. We have to predict the classes according to their happenings. In **K-Means Clustering**, we are given **K** number of clusters to divide our datapoints. The clusters are generated according to their **mean** value, that is why this model is called **K-Means Clustering**. The distance measurement we consider here is the **Euclidean Distance** from one point to another. The word **cluster** means there are some datapoints together and the main goal is to keep the distance low of in-cluster points and keep the distance high of the clusters.

II. Task

As this classifier is a form of **unsupervised learning**, there is no labelled data. So, we are given some datapoints at random. These data have to be clustered. Here, there are some examples of our given datapoints:
$W = \{(-7.87157, -4.86573), (5.86288, 0.99790), ...\}$
For implementing the **K-Means Clustering**, there are some gradually incremented tasks which we have to solve on the way of implementing. These tasks are:

- Firstly, we have to take all the input datapoints from the file and plot the points.
- Secondly, the implementation part comes. We have to implement the **K-Means Clustering** model with the cluster value of **k** which is taken from user.
- Lastly, we have to color the corresponding datapoints according to clusters.

III. Experimental Design

- For the first task, we have to plot the datapoints which are taken from the input file and visualize the occurances.

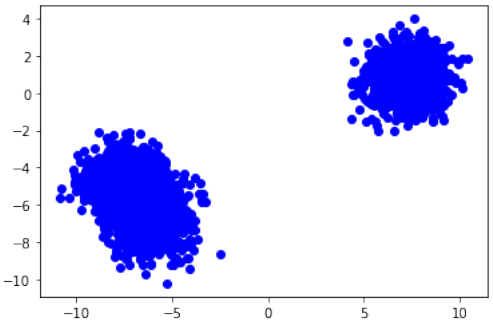

Fig. 1. Visualization of the Datapoints

- Now, it is the time for the implementation of **K-Means Clustering** model. Before the implementation, we need to understand how the algorithm works.

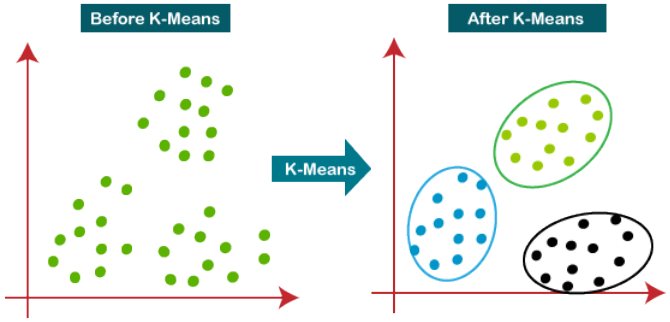

Fig. 2. K-Means Clustering

  - Firstly, we need to take the input of cluster number **K**
  - Then, we have to find the centroids by shuffling the datapoints and take random datapoints as centroids.
  - After that, we have to iterate over all the datapoints and keep changing the centroid according to the newest mean value. For finding the distance, we will follow the **Euclidean Distance's** value.

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad (1)$$

- For stopping, we may define a fixed iteration number or we have to keep track of the changing datapoints and centroids. When there is no change in fixing the centroid, the algorithm stops. Our main task is to plot the datapoints close to the centroids.
- Finally, we need to color the corresponding datapoints of each cluster differently.

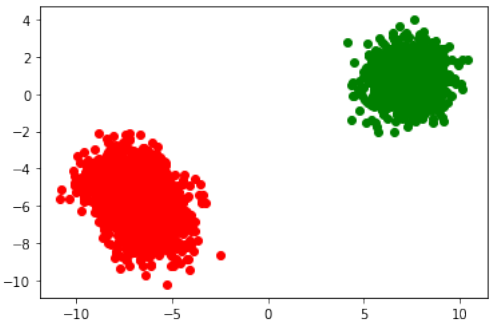

Fig. 3. Visualization of the Clusters

IV. Result Analysis

After implementing the algorithm, we have seen different colored datapoint at different clusters. From the picture, we can see that, datapoints which are very close to each other are colored in same and datapoints which are far, they are of different color. The in-cluster distance of the datapoints is less while, the distance between different clusters are high.

V. Python Code

```
# -*- coding: utf-8 -*-
"""160204033_A2_05.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1OKyARfCG4HyAg-Ff-MnHkmlsT_gSiIYP
"""

import io
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
data = pd.read_csv('data_k_mean.txt',sep=" ",header = None)
x=[i for i in data[0]]
y=[i for i in data[1]]
plt.plot(x ,y ,marker="o",linestyle = 'None',color="blue")

k = int(input("Enter the Value of Clusters: "))
datalen=len(data[0])
print(datalen)
centroids=[]
for i in range(k):
  index=random.randint(0,datalen-1)
  centroids.append([data[0][index],data[1][index]])
print(centroids)

import math
lst=[]
converge=False
for m in range (0,200):

  if(m>0):
    centroidsTmp=[]
    for j in range(k):
      classwisedata1=[i[0] for i in lst if i[2]==j]
      classwisedata2=[i[1] for i in lst if i[2]==j]

      centroidsTmp.append([sum(classwisedata1)/len(classwisedata1),sum(classwisedata2)/len(classwisedata2)])

    if(centroids==centroidsTmp):
      converge=True
    else:
      centroids=centroidsTmp

  if(converge):
    break
  lst=[]
  print("iteration: ",m)
  for i in range(0,datalen):
    distance=[]
    for j in range(k):
      distance.append(math.sqrt(pow(centroids[j][0]-data[0][i],2)+pow(centroids[j][1]-data[1][i],2)))


    lst.append([data[0][i],data[1][i],distance.index(min(distance))])

  print(lst)

import matplotlib.pyplot as plt
st=['red','green','blue','yellow']
for i in range(k):
  x1=[j[0] for j in lst if j[2]==i ]
  y1=[j[1] for j in lst if j[2]==i ]
  colIndex=i%4
  plt.plot(x1,y1 ,marker="o",linestyle = 'None',color=st[colIndex])
```

VI. Conclusion

K-Means Classifier is a classifier which is based on **Unsupervised Learning**. There are several classifiers, but **K-Means Classifier** works better and the algorithm is simple as well. The knowledge of implementation will help us to consturct other models in future.