


KMeans Clustering Algorithm in Machine Learning

 ddas.tech/kmeans-clustering-algorithm-in-machine-learning/

January 2, 2023

In this post we will explore the KMeans Clustering algorithm using various approaches. KMeans is a simple algorithm which starts with initializing a number of random centroids and updating the cluster labels based on its proximity to each centroid. After the first pass each point in the data set is assigned a cluster label. Then the subsequent iteration would calculate the new centroids as a mean of all the items in the cluster and then all points will be re-evaluated for its proximity to the updated centroids.

The process of updating the cluster labels and recalculating the centroids will continue till the time the centroids don't change. The number of iterations for updating the cluster can be controlled through a max iteration variable.

Although a simple algorithm, the KMeans clustering has the problem of identifying the initial centroids and based on the initial choice the algorithm might give varied results.

The initial centroid issue can be tackled to some extent by using bisecting KMeans approach which starts with two initial centroids and progressively adding more centroids. There is an option to use `kmeans++` in scikit-learn that initializes the centroid distant from each other providing better results than random initialization.

KMeans also struggles with noise points and doesn't perform well with non-globular structures or irregular shapes. With addition of more dimensions (curse of dimensionality) the distance calculation can slow down the computation. One should use Principal component Analysis (PCA) to address this problem and to speed up the computation

KMeans Clustering using SKLearn

Using SKLearn to explore KMeans Clustering and Plotting the cluster centroids along with the cluster points

```

import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

A = [
    [10,10],
    [11,10],
    [12,10],
    [10,11],
    [15,17],
    [13.5,15],
    [15,16],
    [17,18],
    [14.5,19],
    [20,20],
    [13.5,14.5]
]

df = pd.DataFrame(A)
number_clusters = 3
pca = PCA(n_components=2)
pca.fit(df)
x_pca = pca.fit_transform(df)
km = KMeans(
    n_clusters=number_clusters, init='random',
    n_init=10, max_iter=300,
    tol=1e-04, random_state=0
).fit(x_pca)

kmeans_labels = km.labels_
unique_labels = np.unique(kmeans_labels)
for i in unique_labels:
    plt.scatter(
        x_pca[kmeans_labels == i, 0], x_pca[kmeans_labels == i, 1]
    )
cluster_centroids = km.cluster_centers_
print("cluster_centroids = ",cluster_centroids)
print("kmeans_labels = ",kmeans_labels)
plt.scatter(cluster_centroids[:,0], cluster_centroids[:,1],linewidths = 3,
s=150,marker="x", color='r')

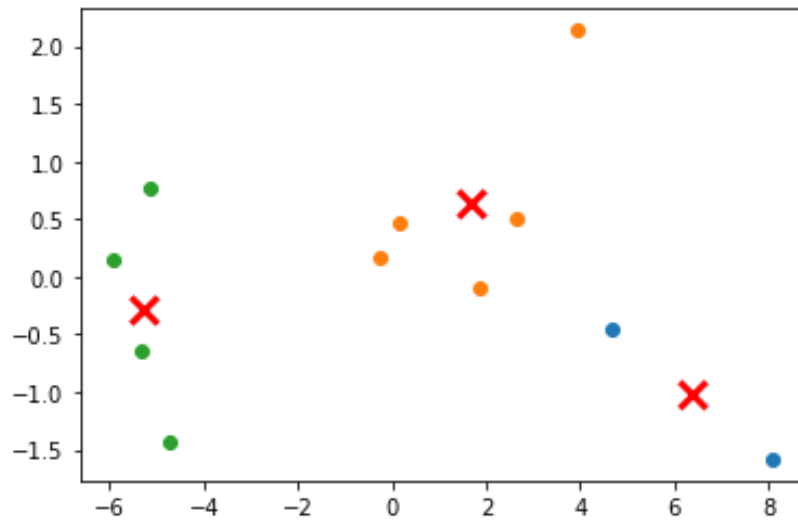
```

Output

```

cluster_centroids = [[ 6.38423083 -1.0142913 ]
 [ 1.67176795  0.63576739]
 [-5.28182536 -0.28756359]]
kmeans_labels = [2 2 2 2 1 1 1 0 1 0 1]

```



KMeans Clustering Output with Centroids denoted by X marks

References

<https://scikit-learn.org/stable/modules/clustering.html#k-means>