Cluster_Analysis___using_Kmeans

July 7, 2020

0.1 Introduction

K-means clustering algorithm represents each cluster by its corresponding cluster centroid. The algorithm will partition the input data data into k-disjoint clusters by using the following steps:

1. Form K clusters by assigning each instance to its nearest centroid.
2. Recompute the centroid of each cluster.

[1]:
```
import pandas as pd
%config IPCompleter.greedy=True
```

[2]:
```
ratings=[['John',5,5,2,1],␣
 ↪['Mary',4,5,3,2],['Bob',4,4,4,3],['Lisa',2,2,4,5],['Lee',1,2,3,4],['Harry',2,1,5,5]]
```

[3]:
```
ratings
```

[3]:
```
[['John', 5, 5, 2, 1],
 ['Mary', 4, 5, 3, 2],
 ['Bob', 4, 4, 4, 3],
 ['Lisa', 2, 2, 4, 5],
 ['Lee', 1, 2, 3, 4],
 ['Harry', 2, 1, 5, 5]]
```

[4]:
```
titles=['User','Jaws','Star Wars','Exorcist','Omen']
```

[5]:
```
titles
```

[5]:
```
['User', 'Jaws', 'Star Wars', 'Exorcist', 'Omen']
```

[6]:
```
movies=pd.DataFrame(ratings,columns=titles)
```

[7]:
```
movies
```

[7]:
```
    User  Jaws  Star Wars  Exorcist  Omen
0   John     5          5         2     1
1   Mary     4          5         3     2
2    Bob     4          4         4     3
3   Lisa     2          2         4     5
4    Lee     1          2         3     4
```

```
5  Harry     2          1          5     5
```

0.2 Data Discusion

From the dataset the first 3 users like action movies, and the last 3 users enjoy horror movies. Our goal is to apply K-means clustering on the users to identify groups of users with similar movie preferences. K=2 from the dataset.

```
[8]: from sklearn import cluster
```

```
[9]: data=movies.drop('User',axis=1)
```

```
[10]: data
```

```
[10]:    Jaws  Star Wars  Exorcist  Omen
      0     5          5         2     1
      1     4          5         3     2
      2     4          4         4     3
      3     2          2         4     5
      4     1          2         3     4
      5     2          1         5     5
```

```
[11]: k_means=cluster.KMeans(n_clusters=2,max_iter=50,random_state=1)
```

```
[12]: k_means
```

```
[12]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=50,
             n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
             random_state=1, tol=0.0001, verbose=0)
```

```
[13]: k_means.fit(data)
```

```
[13]: KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=50,
             n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
             random_state=1, tol=0.0001, verbose=0)
```

```
[14]: labels = k_means.labels_
```

```
[15]: labels
```

```
[15]: array([0, 0, 0, 1, 1, 1])
```

```
[16]: pd.DataFrame(labels,index=movies.User,columns=['Cluster ID'])
```

```
[16]:        Cluster ID
      User
      John            0
```

```
Mary           0
Bob            0
Lisa           1
Lee            1
Harry          1
```

K-means clustering has assigned the first 3 users to one cluster and the last 3 users to the second cluster. These results are consistent with our expectations. We can also display the centroid for each of the two clusters.

```
[17]: centroids=k_means.cluster_centers_
```

```
[18]: centroids
```

```
[18]: array([[4.33333333, 4.66666667, 3.        , 2.        ],
             [1.66666667, 1.66666667, 4.        , 4.66666667]])
```

```
[19]: pd.DataFrame(centroids,columns=data.columns)
```

```
[19]:       Jaws  Star Wars  Exorcist      Omen
      0  4.333333   4.666667       3.0  2.000000
      1  1.666667   1.666667       4.0  4.666667
```

The cluster centroids can be used to determine other users cluster assignments.

```
[20]: import numpy as np
```

```
[21]: testData = np.array([[4,5,1,2],[3,2,4,4],[2,3,4,1],[3,2,3,3],[5,4,1,4]])
```

```
[22]: testData
```

```
[22]: array([[4, 5, 1, 2],
             [3, 2, 4, 4],
             [2, 3, 4, 1],
             [3, 2, 3, 3],
             [5, 4, 1, 4]])
```

```
[23]: labels=k_means.predict(testData)
```

```
[24]: labels
```

```
[24]: array([0, 1, 0, 1, 0])
```

```
[25]: labels=labels.reshape(-1,1)
```

```
[26]: labels
```

[26]:
```
array([[0],
       [1],
       [0],
       [1],
       [0]])
```

[27]:
```
usernames=np.array(['Paul','Kim','Liz','Tom','Bill']).reshape(-1,1)
```

[28]:
```
cols=movies.columns.tolist()
```

[29]:
```
cols
```

[29]:
```
['User', 'Jaws', 'Star Wars', 'Exorcist', 'Omen']
```

[30]:
```
cols.append('Cluster ID')
```

[31]:
```
cols
```

[31]:
```
['User', 'Jaws', 'Star Wars', 'Exorcist', 'Omen', 'Cluster ID']
```

[32]:
```
newusers=pd.DataFrame(np.concatenate((usernames,testData,labels),␣
 ↪axis=1),columns=cols)
```

[33]:
```
newusers
```

[33]:
```
   User Jaws Star Wars Exorcist Omen Cluster ID
0  Paul    4         5        1    2          0
1   Kim    3         2        4    4          1
2   Liz    2         3        4    1          0
3   Tom    3         2        3    3          1
4  Bill    5         4        1    4          0
```

[34]:
```
data
```

[34]:
```
   Jaws  Star Wars  Exorcist  Omen
0     5          5         2     1
1     4          5         3     2
2     4          4         4     3
3     2          2         4     5
4     1          2         3     4
5     2          1         5     5
```

[40]:
```
centroids=k_means.cluster_centers_
```

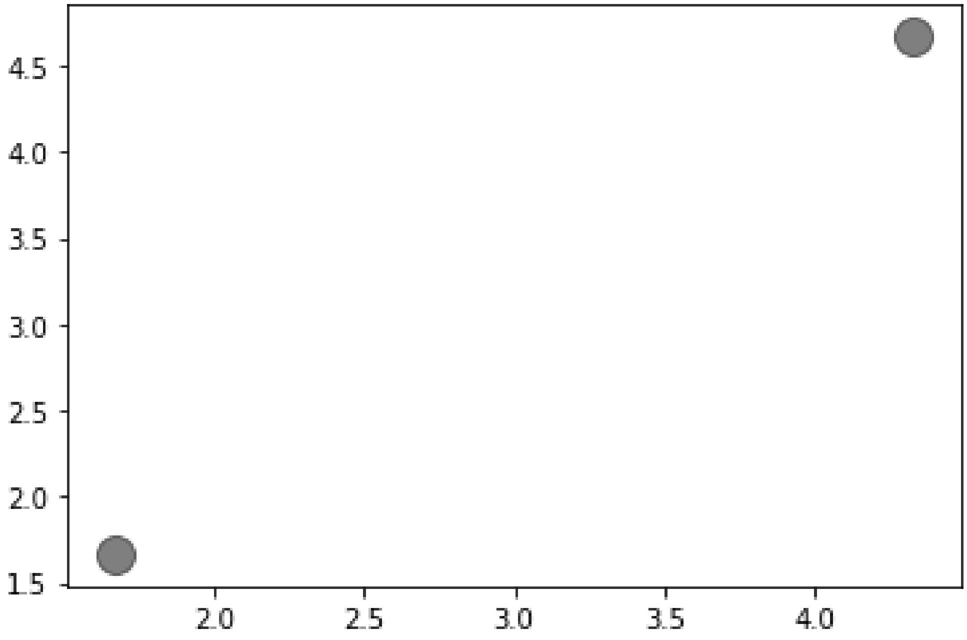
0.3 Future Work

How to determine the number of clusters. We looked at the data and concluded that 2 clusters are okay. How about if you have a lot of data, or high dimensional data

[39]:
```
import matplotlib.pyplot as plt
```

[41]:
```
plt.scatter(centroids[:,0],centroids[:,1],c='black', s=200,alpha=.5)
```

[41]:
```
<matplotlib.collections.PathCollection at 0x1e519cb3708>
```



[43]:
```
k_means.inertia_
```

[43]:
```
9.333333333333334
```

1 How To Determine The Number of *Clusters*

We can determine the number of clusters in the data using the k-means clustering by varying the number of clusters within a range (this is a trial and error method). For example in this case we can vary from 1 to 6, and then compute the sum-of-squared-errors(SSE). The elbow in the plot of the SSE vurses the number of clusters can be used to estimate the number of clusters.

[45]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

[46]:
```
numClusters=[1,2,3,4,5,6]
```

[47]:

```
SSE=[]
```

[51]:

```
data
```

[51]:

```
   Jaws  Star Wars  Exorcist  Omen
0     5          5         2     1
1     4          5         3     2
2     4          4         4     3
3     2          2         4     5
4     1          2         3     4
5     2          1         5     5
```

[50]:

```
for k in numClusters:
    k_means=cluster.KMeans(n_clusters=k)
    k_means.fit(data)
    SSE.append(k_means.inertia_)
```

[ ]:

```

```