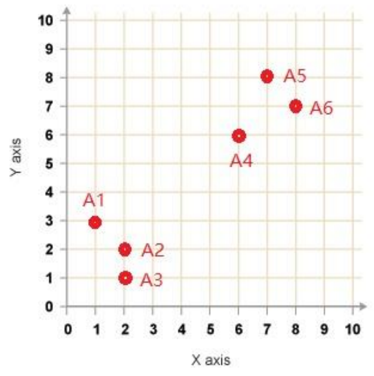
The K-means algorithm

Data Processing Using Python   by Dazhuang@NJU

The K-means algorithm is a typical distance-based cluster analysis algorithm which uses distance as the evaluation measure. The smaller the distance of two object is, the more similar the two objects are.
In the process, a data cluster is made up by objects close to each other.
Therefore, the algorithm is aimed to calculate compact and standalone clusters.
Suppose to divide objects into k clusters. The algorithm will be:
(1) Randomly pick up k objects to be the centroid of each cluster. Initially each cluster will be represented by the only object.
(2) For each object which is not one of the centroids, assign it to the cluster of which the centroid is the nearest.
(3) Re-calculate the centroid of each cluster.
(4) Repeat step (2) and (3) until the new centroid and old centroid of each cluster are identical or closer than a preset threshold. At the point, the algorithm ends.

As an example, if we randomly pick up a few data, the algorithm will be:



Given six points - A1, A2, … A6:

	X	Y
A1	1	3
A2	2	2
A3	2	1
A4	6	6
A5	7	8
A6	8	7

If we want to have two cluster, the procedure will be:
(1) Assume A1 and A2 as initial centroid:
(2) Calculate the Euclidean distance of each non-centroids (A3-A6) and each centroids (A1-A2) with the formular d = sqrt($(x1-x2)^2+(y1-y2)^2$)

	A1	A2
A3	2.24	1
A4	5.83	5.66
A5	6.4	6.4
A6	8.06	7.81

(3) Based on the table above, A3,A4 and A6 are all closer to A2 while A5 is evenly close to both A1 and A2. Let's just assign A5 to the cluster where A2 is. Then we will have two new clusters with all objects included:

Cluster 1： A1

Cluster 2： A2， A3， A4， A5， A6

(4) Calculate new centroids

The new centroid of cluster 1： A1

The new centroid C_temp of cluster 2 is the average of all dimenons.

((A2.x+A3.x+A4.x+A5.x+A6.x)/5 , (A2.y+A3.y+A4.y+A5.y+A6.y)/5)=(5, 4.8)

	A1	C_temp
A2	1.41	4.1
A3	2.24	4.84
A4	5.83	1.56
A5	6.4	3.77
A6	8.06	3.72

(5) Base on the distances to the new centroids, objects are divided into two new clusters.

Cluster 1： A1， A2， A3

Cluster 2： A4， A5， A6

New centroid 1 "C_temp1"： ((A1.x+A2.x+A3.x)/3 , (A1.y+A2.y+A3.y)/3)=(1.67, 2)

New centroid 2 "C_temp2"： ((A4.x+A5.x+A6.x)/3 , (A4.y+A5.y+A6.y)/3)=(7, 7)

	C_temp1	C_temp2
A1	1.2	7.21
A2	0.33	7.07
A3	1.05	7.81
A4	5.89	1.41
A5	6.66	1
A6	6.71	1

(6) bingo ㄟ(◑‿◐)ㄏ

The two new clusters are having the same objects as the previous two clusters respectively and the processing ends here.

Cluster 1： A1， A2， A3

Cluster 2： A4， A5， A6

Note

(1) The selection of k in K-means algorithm is subject to human experience.

(2) Euclidean distance or cosine similarity is usually selected to calculate distance. Euclidean distance is based on position coordinates which mainly

represent the difference of individual values. Cosine similarity represents more directional difference than value difference. The more the value of cos θ is close to 1, the more similar the two objects are similar. Cosine similarity can offset the difference caused by various measure systems.
(3) Initialization of centroid chosen from the space is very vital and sometimes it will get stuck in local optima.

Discussion about the K value
For a bunch of data, how to determine the optimal solution of K value (cluster numbers)? Common methods are Elbow method and Silhouette Coeffient method. The basic ideas of these two methods are described as follows. Specific formulas can be consulted in relevant documents.
① Elbow method: The core index is SSE (sum of the square errors), that is the clustering errors of all samples (cumulative square of the distance from the sample to the centroid in each cluster). With the increase of K, the degree of clustering of each cluster will increase and the decreasing extent of SSE will increase. With the increase of K, the decreasing extent of SSE will decrease and tend to be flat, and the relationship between SSE and K will be presented the shape of an elbow. The K value corresponding to the elbow is the best clustering number.
② Silhouette Coeffient method: Considering the Cohesion and Separation of clustering, the cohesion is the average distance between the sample and other samples in the same cluster, and the separation is the average distance between the sample and all samples in the nearest cluster. The larger the value, the better the clustering effect. The value is between - 1 and 1.

Take the iris data as an example.

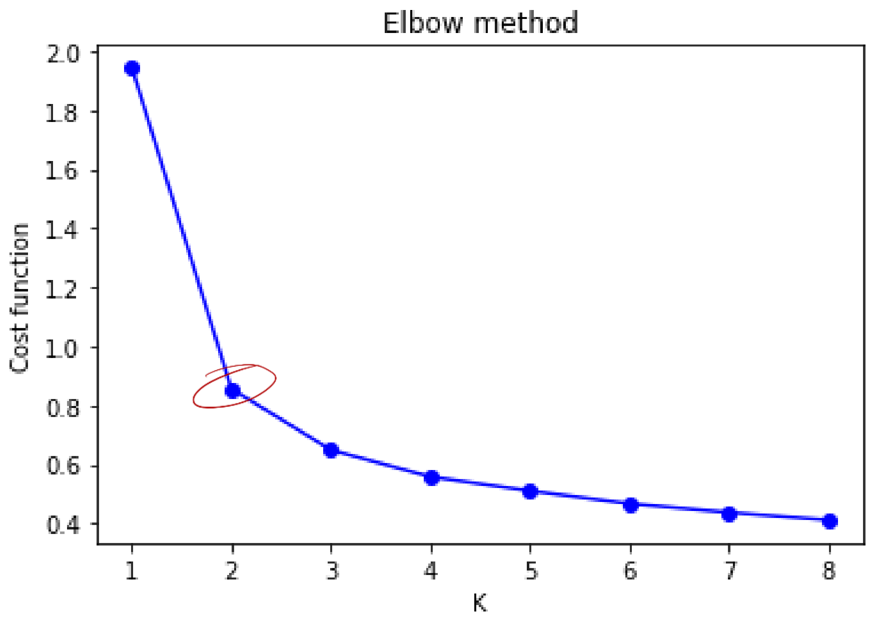
[Elbow method]
```
import numpy as np
from sklearn.cluster import KMeans
from scipy.spatial.distance import cdist
import matplotlib.pyplot as plt
from sklearn import datasets

iris = datasets.load_iris()
X = iris.data

K = range(1, 9)      # assumpt group into 1~8 clusters
lst = []
for k in K:
    kmeans = KMeans(n_clusters = k)
    kmeans.fit(X)
    # Calculate the average value of the sum of minimum lists for
corresponding K Value
    lst.append(sum(np.min(cdist(X,kmeans.cluster_centers_,
      'euclidean'), axis = 1)) / X.shape[0])
    # cdist(X, kmeans.cluster_centers_, 'euclidean')  find the sum of
      squares of distance from X to every cluster_centers_, shape is (150,
      k), 'euclidean' means using Euclidean Distance
    # np.min(cdist(X,kmeans.cluster_centers_, 'euclidean'), axis = 1)
      Calculate the minimum in each row
    # sum(np.min(cdist(X,kmeans.cluster_centers_, 'euclidean'), axis = 1))
      calculate the sum of the list of minimum values under each round of K
      value

plt.plot(K, lst, 'bo-')
plt.title('Elbow method')
plt.xlabel("K")
plt.ylabel("Cost function")
plt.show()
```

We can see the elbow is at the point where K equals to 2, or K equals to 3 and the iris data is group into 3 categories.



[Silhouette Coeffient]

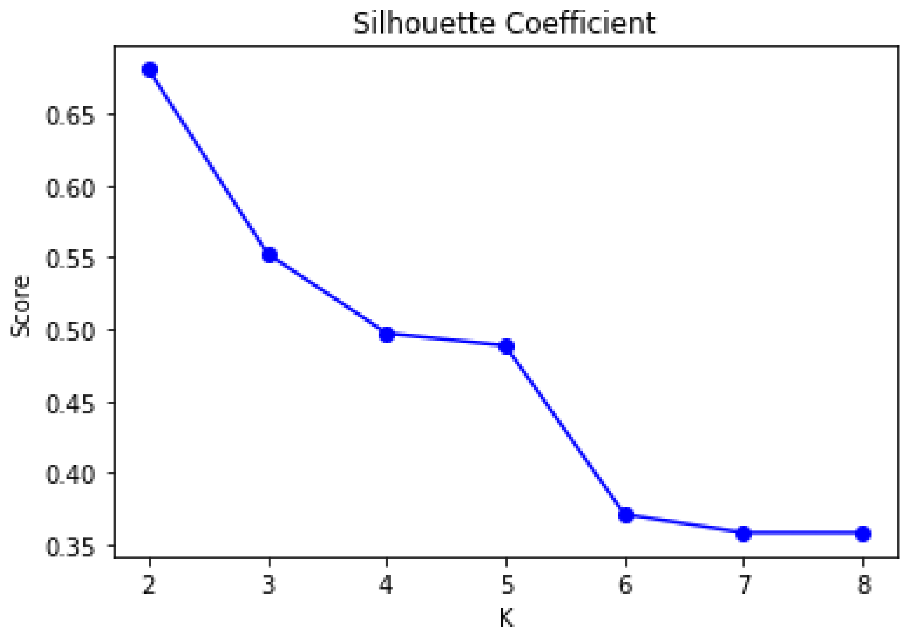
```
from sklearn. cluster import KMeans
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn import metrics
```

```
iris = datasets.load_iris()
X = iris.data

K = range(2, 9)     # assumpt group into 2~8 clusters
lst = []
for k in K:
    kmeans_model = KMeans(n_clusters = k).fit(X)
    sc_score = metrics.silhouette_score(X, kmeans_model.labels_,
                 metric = 'euclidean')
    # silhouette_score() calculate the average silhouette coeffient of all
samples
    # kmeans_model.labels_ class labels predicted by each sample
    # metric = 'euclidean'
    lst.append(sc_score)

plt.plot(K, lst, ' bo-')
plt.title('Silhouette Coefficient')
plt.xlabel("K")
plt.ylabel("Score")
plt.show()
```

Drawing the relationship between K and the corresponding silhouette coefficient is as follows, which shows that K = 2 is the best. The actual iris data are divided into three clusters.


Silhouette Coefficient
Score
K