Clustering and Classification Exercises

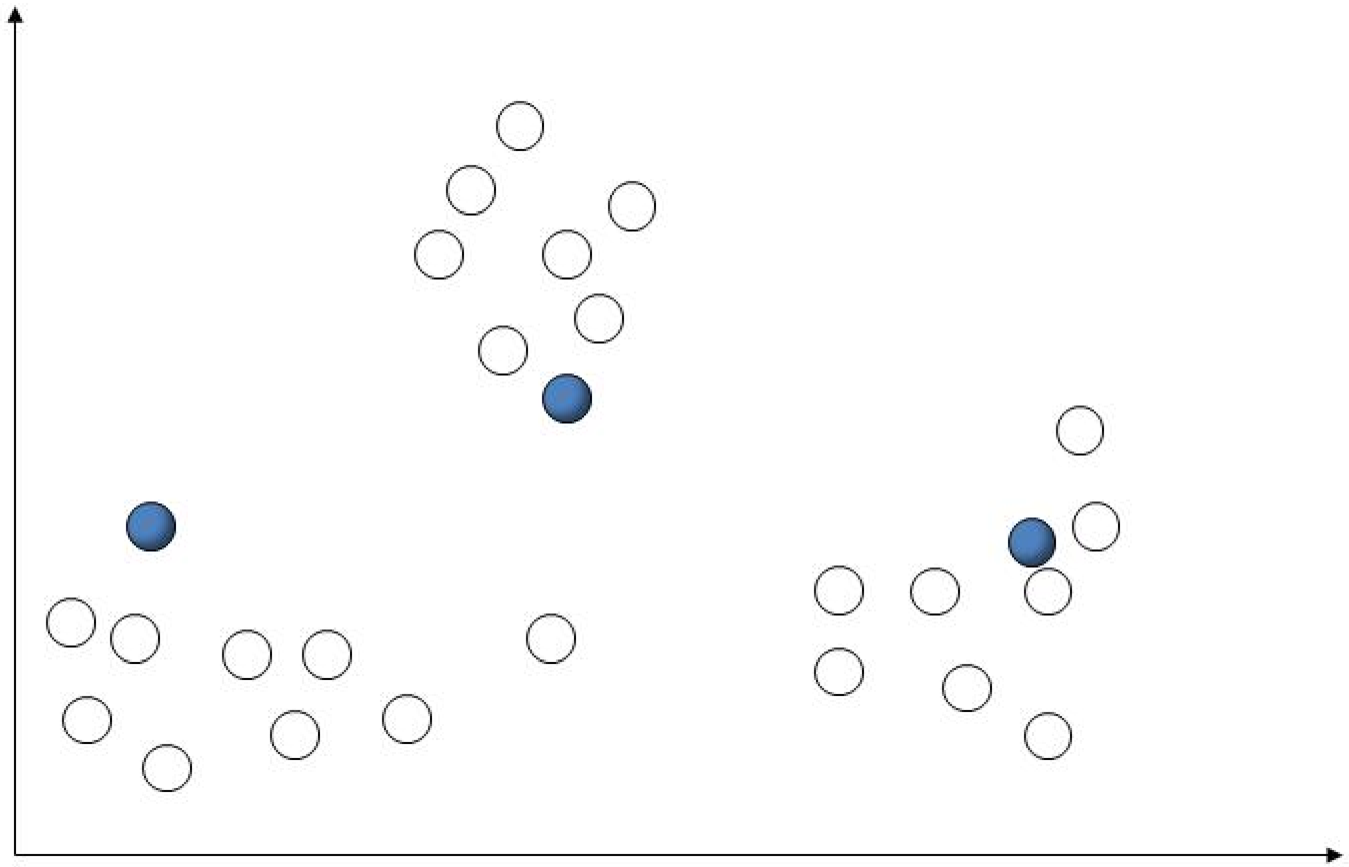
張家瑋 博士
副教授
國立臺中科技大學資訊工程系

分群概念

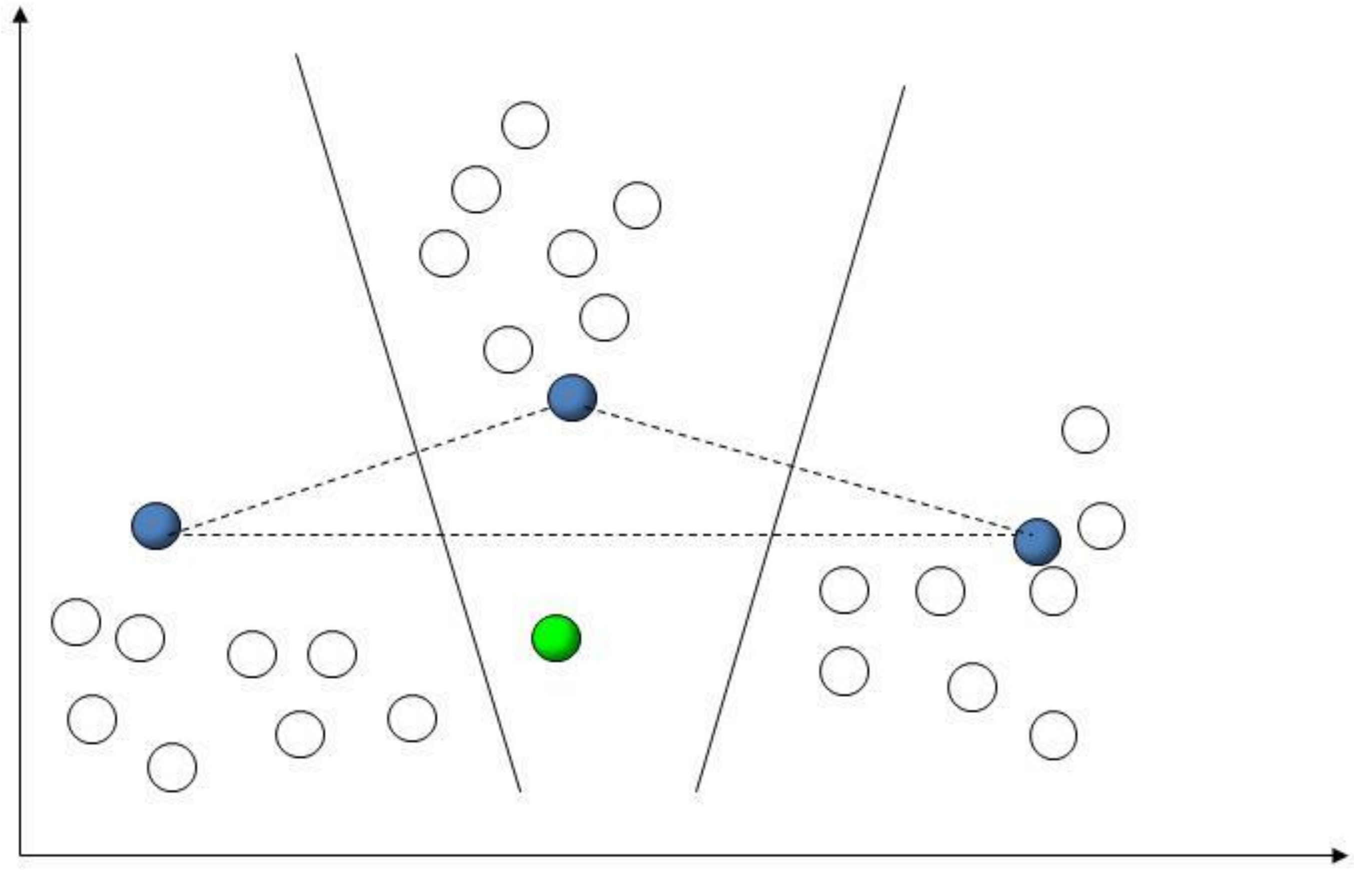
- 把許多事物按照某種標準歸為數個類別，其中較為相近/類似的聚為一類，反之較不相近的則聚為不同類。
- 目的是企圖從一大堆雜亂無章的原始資料中，找出少數幾個較小的群體，使得群體內的分子在某些變項的測量值均很類似，而群體與群體間的分子在該測量值上差異較大。
- 同一組樣本會因不同目的、資料輸入方式、所選擇分群特徵或資料屬性，形成不同的分群結果

K-means

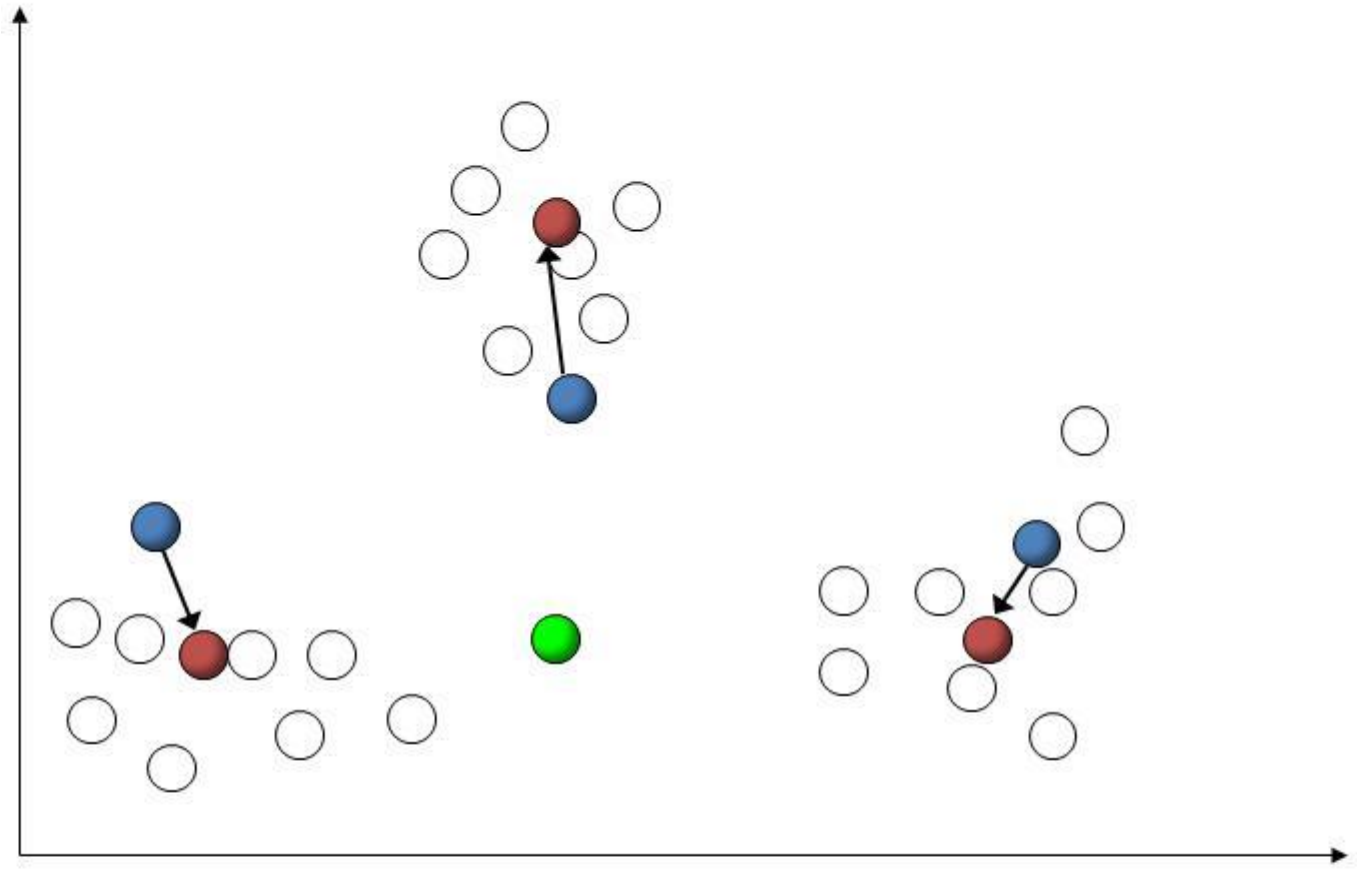
Step 1. 隨機指派群集中心



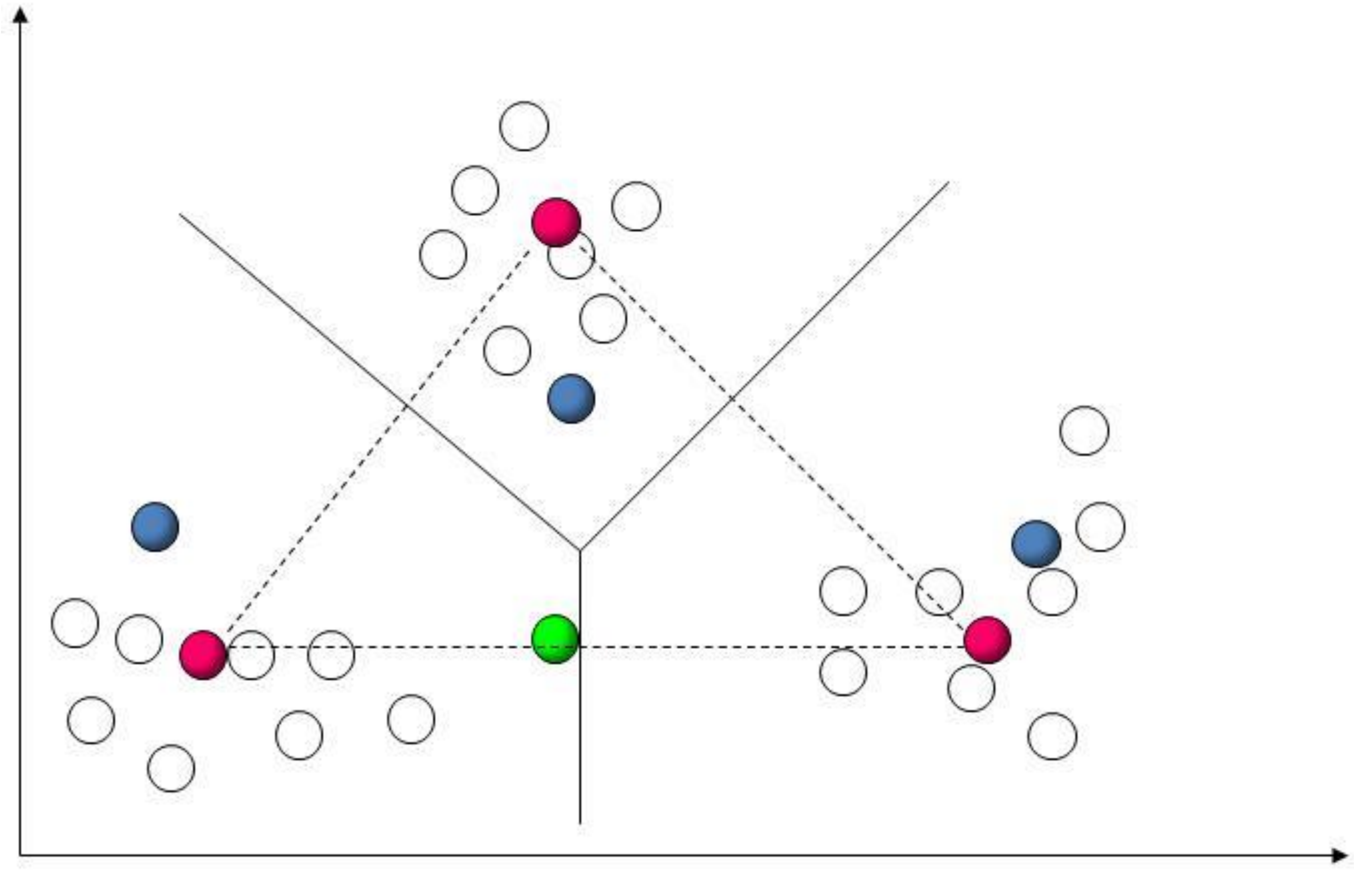
Step 2. 產生初始群集



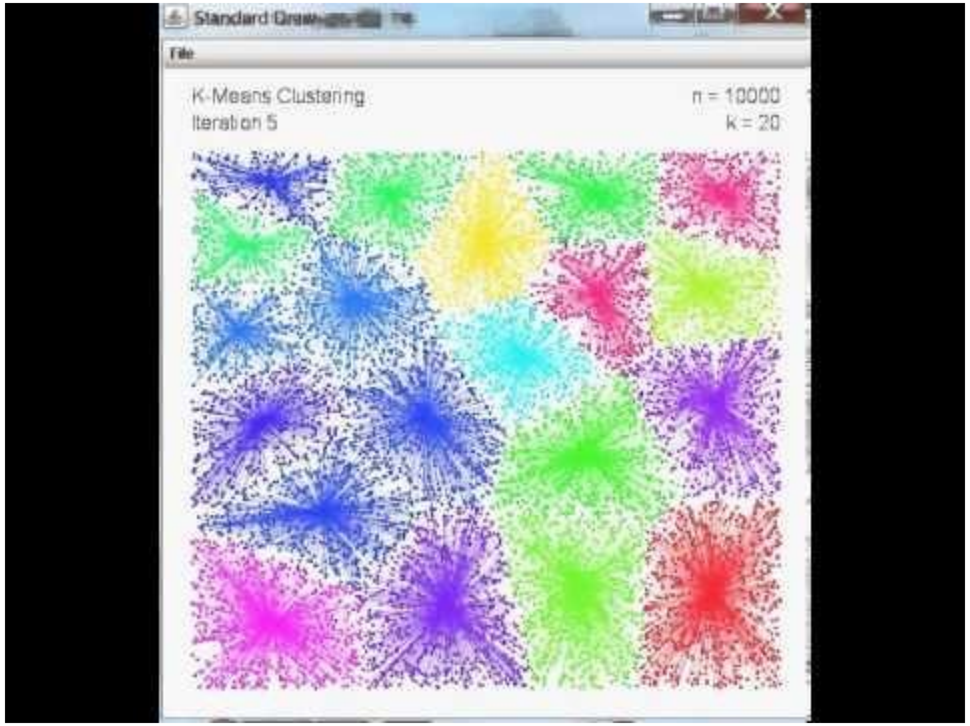
Step 3. 產生新的質量中心



STEP 4. 變動群集邊界



Example



https://youtu.be/BVFG7fd1H30

事前要件

安裝SKlearn模組

```
pip3 install -U scikit-learn
```

鳶尾花資料集

- 花瓣（Petal）的長
- 花瓣（Petal）的寬
- 花萼（Sepal）的長
- 花萼（Sepal）的寬

[5.1 3.5 1.4 0.2]

在設定某K的Kmeans

```python
from sklearn import cluster, datasets

# 讀入鳶尾花資料
iris = datasets.load_iris()
iris_X = iris.data

# KMeans 演算法
kmeans_fit = cluster.KMeans(n_clusters = 3).fit(iris_X)

# 印出分群結果
cluster_labels = kmeans_fit.labels_
print("分群結果：")
print(cluster_labels)
print("---")

# 印出品種看看
iris_y = iris.target
print("真實品種：")
print(iris_y)
```

在設定某K的Kmeans

```
[5.1 3.5 1.4 0.2]
分群結果：
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 0 2 0 0 0 0 2 0 0 0 0
 0 0 2 2 0 0 0 0 2 0 2 0 2 0 0 2 2 0 0 0 0 0 2 0 0 0 0 2 0 0 0 2 0 0 0 2 0
 0 2]
---
真實品種：
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

K從2到10的Kmeans效能

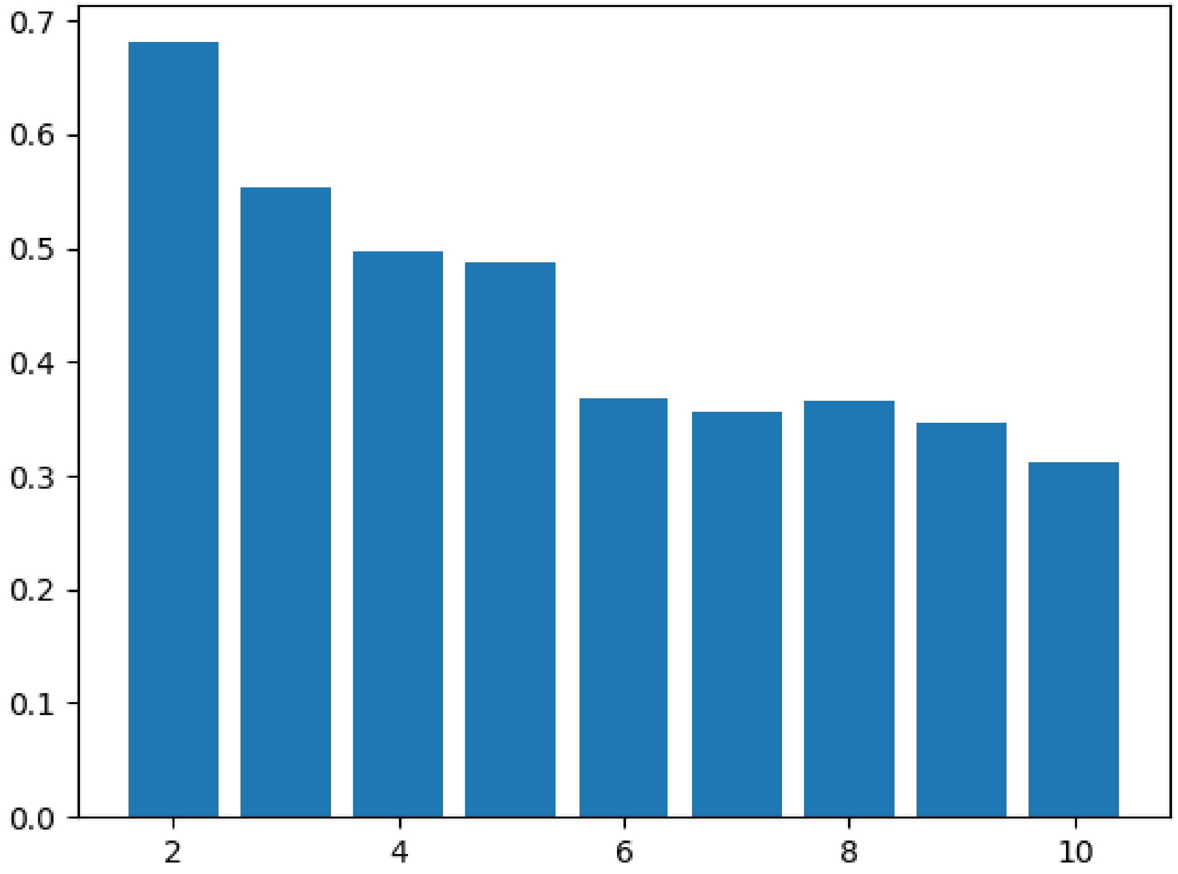
```python
from sklearn import cluster, datasets, metrics
import matplotlib.pyplot as plt

# 讀入鳶尾花資料
iris = datasets.load_iris()
iris_X = iris.data

# 迴圈
silhouette_avgs = []
ks = range(2, 11)
for k in ks:
    kmeans_fit = cluster.KMeans(n_clusters = k).fit(iris_X)
    cluster_labels = kmeans_fit.labels_
    silhouette_avg = metrics.silhouette_score(iris_X, cluster_labels)
    silhouette_avgs.append(silhouette_avg)

# 作圖並印出 k = 2 到 10 的績效
plt.bar(ks, silhouette_avgs)
plt.show()
print(silhouette_avgs)
```

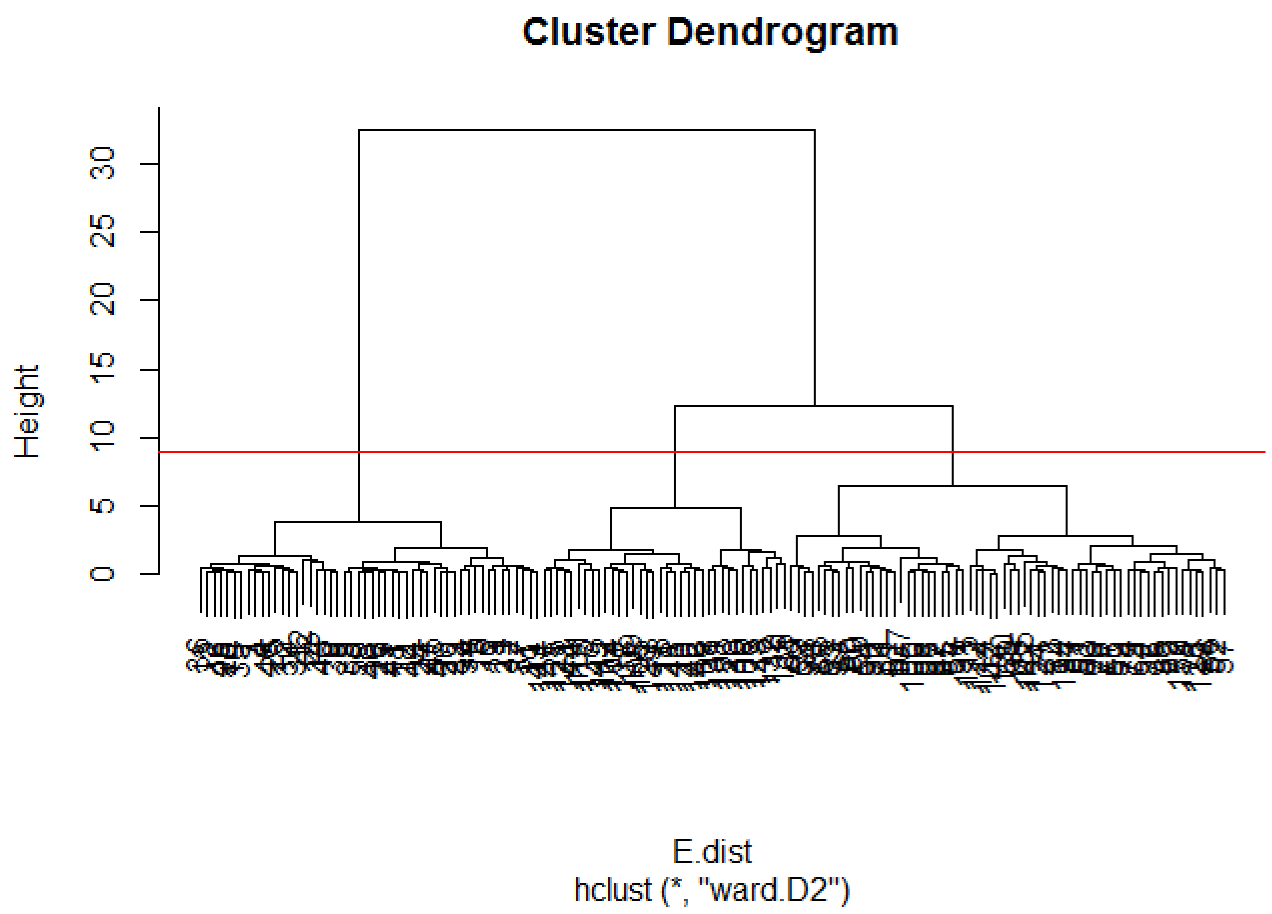
K從2到10的Kmeans效能



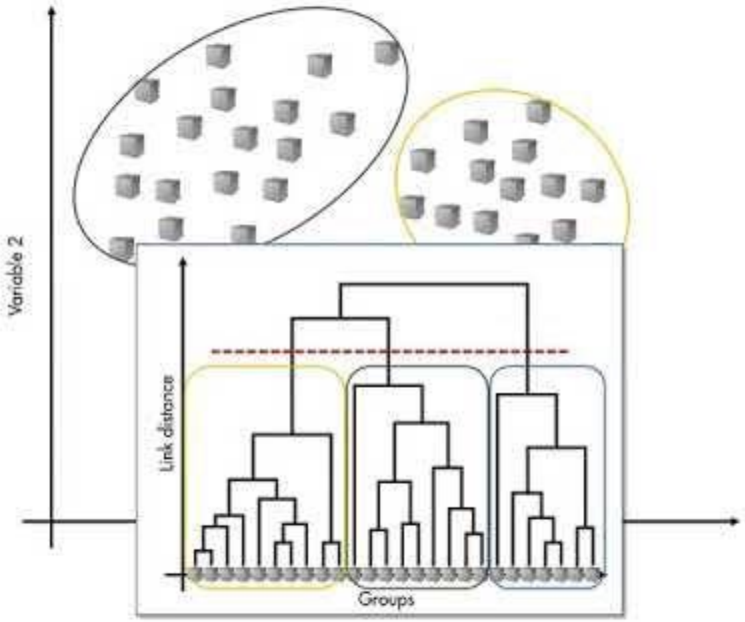
階層式分群法

hierarchical clustering

Processes


Cluster Dendrogram
Height
0
5
10
15
20
25
30
E.dist
hclust (*, "ward.D2")

Example



https://youtu.be/iy7-Q7Y1Klk

鳶尾花資料集

- 花瓣（Petal）的長
- 花瓣（Petal）的寬
- 花萼（Sepal）的長
- 花萼（Sepal）的寬

[5.1 3.5 1.4 0.2]

在設定某K的Hierarchical clustering

```python
from sklearn import cluster, datasets

# 讀入鳶尾花資料
iris = datasets.load_iris()
iris_X = iris.data

# 印出單筆測資
print(iris_X[0])

# Hierarchical Clustering 演算法
hclust = cluster.AgglomerativeClustering(linkage = 'ward', affinity = 'euclidean', n_clusters = 3)

# 印出分群結果
hclust.fit(iris_X)
cluster_labels = hclust.labels_
print(cluster_labels)
print("---")

# 印出品種看看
iris_y = iris.target
print(iris_y)
```

在設定某K的Hierarchical clustering

```
[5.1 3.5 1.4 0.2]
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 0 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
---
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

K從2到10的效能

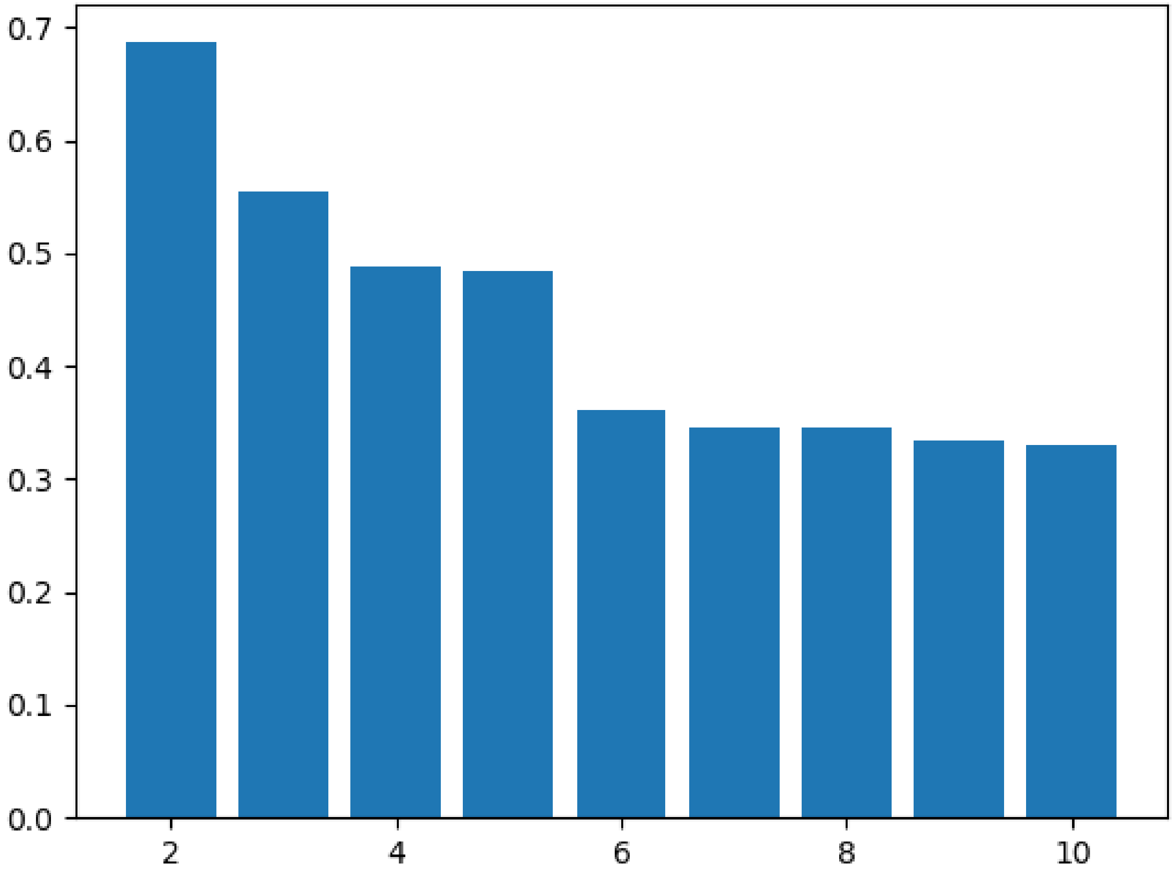
```python
from sklearn import cluster, datasets, metrics
import matplotlib.pyplot as plt

# 讀入鳶尾花資料
iris = datasets.load_iris()
iris_X = iris.data

# 迴圈
silhouette_avgs = []
ks = range(2, 11)
for k in ks:
    # Hierarchical Clustering 演算法
    hclust_fit = cluster.AgglomerativeClustering(linkage = 'ward', affinity = 'euclidean', n_clusters = k).fit(iris_X)
    cluster_labels = hclust_fit.labels_
    silhouette_avg = metrics.silhouette_score(iris_X, cluster_labels)
    silhouette_avgs.append(silhouette_avg)

# 作圖並印出 k = 2 到 10 的績效
plt.bar(ks, silhouette_avgs)
plt.show()
print(silhouette_avgs)
```

K從2到10的效能



自行練習

Wine Dataset

[1.207e+01, 2.160e+00, 2.170e+00, 2.100e+01, 8.500e+01, 2.600e+00, 2.650e+00, 3.700e-01,
1.350e+00, 2.760e+00, 8.600e-01, 3.280e+00, 3.780e+02]

(1) Alcohol	→	1.207e+01	(2) Malic acid	→	2.160e+00
(3) Ash	→	2.170e+00	(4) Alcalinity of ash	→	2.100e+01
(5) Magnesium	→	8.500e+01	(6) Total phenols	→	2.600e+00
(7) Flavanoids	→	2.650e+00	(8) Nonflavanoid phenols	→	3.700e-01
(9) Proanthocyanins	→	1.350e+00	(10)Color intensity	→	2.760e+00
(11)Hue	→	8.600e-01	(12)OD280/OD315 of diluted wines	→	3.280e+00
(13)Proline	→	3.780e+02			

使用SKlearn及Sklearn預設資料集實作

KNN 的曼哈頓、歐幾里得距離及決策樹

分類器

題目敘述

1. 使用SKlearn中的預設的wine資料集進行作業
2. wine資料集中每筆資料都含有13種特徵
3. 使用KNN的曼哈頓、歐幾里得及決策樹分類器將13種特徵進行演算並且分類

載入SKlearn預設資料集

```
# ---導入模塊---
from sklearn import datasets
from sklearn.model_selection import train_test_split
import pandas as pd
# ---資料處理---
wine = datasets.load_wine()
print(wine)
# 載入SKlearn內建資料集
```

print(wine)

```
{'data': array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
        1.185e+03],
       ...,
       [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
        8.350e+02],
       [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
        8.400e+02],
       [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
        5.600e+02]])
```

← data為酒的特徵

print(wine)

```
'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2])
```

← target為上頁各項特徵

所對應到的酒種類

類別分為0,1,2三種標籤

```
wine_data =  wine.data
# 定義資料特徵
wine_target = wine.target
# 定義資料標籤
print(pd.DataFrame(wine.data))
# 印出資料特徵查看
print(pd.DataFrame(wine.target))
# 印出資料標籤查看
x_train, x_test, y_train, y_test = train_test_split(wine_data, wine_target, test_size = 0.2)
# 使用"train_test_spit"將數據分成訓練和測試兩類,test_size = 0.2,代表測試數據佔20%
```

將data打印出一列，來查看一下特徵有哪些

[1.207e+01, 2.160e+00, 2.170e+00, 2.100e+01, 8.500e+01, 2.600e+00, 2.650e+00, 3.700e-01,
 1.350e+00, 2.760e+00, 8.600e-01, 3.280e+00, 3.780e+02]

(1) Alcohol	→	1.207e+01	(2) Malic acid	→	2.160e+00
(3) Ash	→	2.170e+00	(4) Alcalinity of ash	→	2.100e+01
(5) Magnesium	→	8.500e+01	(6) Total phenols	→	2.600e+00
(7) Flavanoids	→	2.650e+00	(8) Nonflavanoid phenols	→	3.700e-01
(9) Proanthocyanins	→	1.350e+00	(10)Color intensity	→	2.760e+00
(11)Hue	→	8.600e-01	(12)OD280/OD315 of diluted wines	→	3.280e+00
(13)Proline	→	3.780e+02			

查看訓練及測試資料集數據

```
print('x_test:測試用特徵')
print(x_test)
print('------------------------------------------')
print('x_train:訓練用特徵')
print(x_train)
print('------------------------------------------')
print('y_test:測試用標籤')
print(y_test)
print('------------------------------------------')
print('y_train:訓練用標籤')
print(y_train)
```

```
x_test:測試用特徵
[[1.207e+01 2.160e+00 2.170e+00 2.100e+01 8.500e+01 2.600e+00 2.650e+00
  3.700e-01 1.350e+00 2.760e+00 8.600e-01 3.280e+00 3.780e+02]
 [1.382e+01 1.750e+00 2.420e+00 1.400e+01 1.110e+02 3.880e+00 3.740e+00
  3.200e-01 1.870e+00 7.050e+00 1.010e+00 3.260e+00 1.190e+03]
 [1.369e+01 3.260e+00 2.540e+00 2.000e+01 1.070e+02 1.830e+00 5.600e-01
  5.000e-01 8.000e-01 5.880e+00 9.600e-01 1.820e+00 6.800e+02]
 [1.141e+01 7.400e-01 2.500e+00 2.100e+01 8.800e+01 2.480e+00 2.010e+00
  4.200e-01 1.440e+00 3.080e+00 1.100e+00 2.310e+00 4.340e+02]
 [1.182e+01 1.720e+00 1.880e+00 1.950e+01 8.600e+01 2.500e+00 1.640e+00
  3.700e-01 1.420e+00 2.060e+00 9.400e-01 2.440e+00 4.150e+02]]
------------------------------------------------------------
x_train:訓練用特徵
[[1.358e+01 1.660e+00 2.360e+00 ... 1.090e+00 2.880e+00 1.515e+03]
 [1.406e+01 2.150e+00 2.610e+00 ... 1.060e+00 3.580e+00 1.295e+03]
 [1.243e+01 1.530e+00 2.290e+00 ... 6.900e-01 2.840e+00 3.520e+02]
 ...
 [1.216e+01 1.610e+00 2.310e+00 ... 1.330e+00 2.260e+00 4.950e+02]
 [1.200e+01 3.430e+00 2.000e+00 ... 9.300e-01 3.050e+00 5.640e+02]
 [1.182e+01 1.470e+00 1.990e+00 ... 9.500e-01 3.330e+00 4.950e+02]]
------------------------------------------------------------
y_test:測試用標籤
[1 0 2 1 1 0 1 1 1 2 1 2 0 1 1 2 2 2 0 1 1 0 2 2 1 0 0 1 2 1 2 2 0 1 1 0]
------------------------------------------------------------
y_train:訓練用標籤
[0 0 1 0 1 2 0 1 0 0 1 1 1 0 1 1 0 0 0 0 0 0 2 2 2 2 2 0 2 0 1 1 2 1 0 0 2
 1 1 0 1 2 0 2 0 2 2 1 0 1 1 2 1 0 1 0 1 1 0 0 1 0 2 2 2 1 1 2 1 2 0 1 1 1
 1 0 0 1 0 0 1 1 2 1 2 0 2 1 0 2 2 1 1 1 1 2 0 0 2 1 2 1 2 1 0 0 1 1 1 0 1
 1 1 0 0 0 1 2 0 0 1 2 2 0 0 2 1 2 0 2 1 0 2 1 0 0 2 0 2 1 1 1]
```

← 20%特徵
(因數據過多只打印出5組)

← 80%特徵

← 20%標籤

← 80%標籤

KNN-曼哈頓距離分類器

```
# ---最短距離---
```

```
knn = KNeighborsClassifier(p = 1)
# 定義模塊,設定p值為1,p值為Minkowski metric參數,p=1使用曼哈頓距離
knn.fit(x_train, y_train)
# 注入訓練數據使用x_train為訓練數據y_train為標籤
print(knn.predict(x_test))
# 預測x_test的標籤類
print(y_test)
```

```
[1 0 2 1 1 0 1 1 1 2 2 1 0 1 2 0 2 0 0 1 0 0 1 2 1 0 0 2 1 1 2 2 0 1 1 1]
[1 0 2 1 1 0 1 1 1 2 1 2 0 1 1 2 2 2 0 1 1 0 2 2 1 0 0 1 2 1 2 2 0 1 1 0]
```

← 下方為預測結果

KNN-歐幾里得距離分類器

```
# ---KNN分類---
```

```
from sklearn.neighbors import KNeighborsClassifier
# 導入模塊
```

```
knn = KNeighborsClassifier(p = 2)
# 定義模塊,設定p值為2,p值為Minkowski metric參數,p=2使用歐幾里得距離
knn.fit(x_train, y_train)
# 注入訓練數據使用x_train為訓練數據y_train為標籤
print(knn.predict(x_test))
# 預測x_test的標籤類
print(y_test)
```

```
[1 0 2 1 1 2 2 1 1 2 2 2 0 1 2 0 2 0 1 1 0 0 1 2 1 0 0 2 1 1 2 1 0 1 1 2]
[1 0 2 1 1 0 1 1 1 2 1 2 0 1 1 2 2 2 0 1 1 0 2 2 1 0 0 1 2 1 2 2 0 1 1 0]
```

← 下方為預測結果

決策樹分類器

```
# ---決策樹---
```

```
from sklearn.tree import DecisionTreeClassifier
# 導入模塊
```

```
tree = DecisionTreeClassifier()
# 定義模塊
tree.fit(x_train, y_train)
# 注入訓練數據使用x_train為訓練數據y_train為標籤
print(tree.predict(x_test))
# 預測x_test的標籤類
print(y_test)
```

```
[1 0 2 1 1 0 1 1 1 2 1 2 0 1 1 2 1 2 0 1 1 0 2 2 1 1 1 1 2 1 2 2 0 1 1 1]
[1 0 2 1 1 0 1 1 1 2 1 2 0 1 1 2 2 2 0 1 1 0 2 2 1 0 0 1 2 1 2 2 0 1 1 0]
```

← 下方為預測結果

參考資料

- https://scikit-learn.org/stable/index.html
- https://morvanzhou.github.io/tutorials/machine-learning/sklearn/

Thank you