Table of Contents

第20章 數據分析演算法-K-means 最近鄰居法

20.1 k-means 數學介紹

k-平均演算法（k-means clustering）原本是用來做訊號處理中的一種向量量化方法，現在

則更多地作為一種聚類分析方法流行於資料探勘領域。k-平均聚類的目的是：把 n 個點的訓練樣本分到 k 個聚類中，使得每個點都屬於離他最近的均值和聚類中心所對應的聚類，以之作為聚類的標準。 聚類基本上就是依照著「物以類聚」的方式，也可能想成，相似的東西有著相似的特徵，所以相同種類的數據應該是非常的類似。

請注意 k-means 與 KNN 之間沒有任何關係的二種演算法。

數學公式和原理：

已知觀測集 （x_1， x_2， ...， x_n） ，其中每個觀測都是一個 d-維實向量，k-平均聚類要把這 n 個觀測劃分到 k 個集合中(k≤n),使得組內平方和最小。換句話說，它的目標是找到使得下式滿足的聚類 S_i 。

而計算 k-means， 可以用以下的公式：

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in = S_i} \|x - \mu_i\|^2$$

其中 μ_i 是 S_i 中所有點的均值。

再用剛剛的柳丁和檸檬的範例來看一下 k-means 演算法，透過以下的程式，首先準備 1.x 的數字，和 2.x 的數字，並且在圖表上面用不同的顏色來表示。

範例程式： 01-kmeans-Mat.py

```
1.  import matplotlib.pyplot as plt
2.  import numpy as np
3.
4.  plt.plot(X[:7,0], X[:7,1], 'yx' )
5.  plt.plot(X[7:,0], X[7:,1], 'g.' )
6.
7.  plt.ylabel('H cm')
8.  plt.xlabel('W cm')
9.  plt.legend(('A','B'),
10.         loc='upper right')
11. plt.show()
```

執行結果

Python

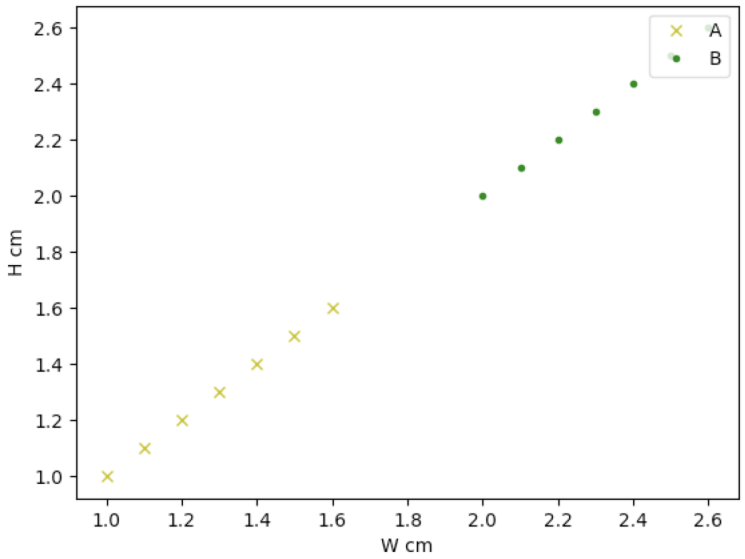

圖 1 執行結果

範例程式：

```
1.  import numpy as np
2.  from sklearn.model_selection import train_test_split
3.  from sklearn.cluster import KMeans
4.  from time import time
5.  import numpy as np
6.  import matplotlib.pyplot as plt
7.  from sklearn import metrics
8.  X= np.array([[1,1],[1.1,1.1],[1.2,1.2],[1.3,1.3],[1.4,1.4],[1.5,1.5],[1.6,1.6],
9.    [2,2], [2.1,2.1], [2.2,2.2], [2.3,2.3], [2.4,2.4], [2.5,2.5], [2.6,2.6]])
10. y=[1,1,1,1,1,1,1,
11.   0,0,0,0,0,0,0]
12. kmeans = KMeans(n_clusters=2, random_state=0).fit(X)
13. print("集群中心的坐標:",kmeans.cluster_centers_)
14. print("預測:",kmeans.predict(X))
15. print("實際:",y)
16. print("預測[1, 1],[2.3,2.1]:",kmeans.predict([[1, 1],[2.3,2.1]]))
17. plt.plot(X[:7,0], X[:7,1], 'yx' )
18. plt.plot(X[7:,0], X[7:,1], 'g.' )
19. plt.plot(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], 'ro')
20. plt.xticks(())
21. plt.yticks(())
```

```
22.  plt.show()
23.
```

執行結果

集群中心的坐標: [[2.3 2.3]

[1.3 1.3]]

預測: [1 1 1 1 1 1 1 0 0 0 0 0 0 0]

實際: [1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]

預測[1, 1],[2.3,2.1]: [1 0]

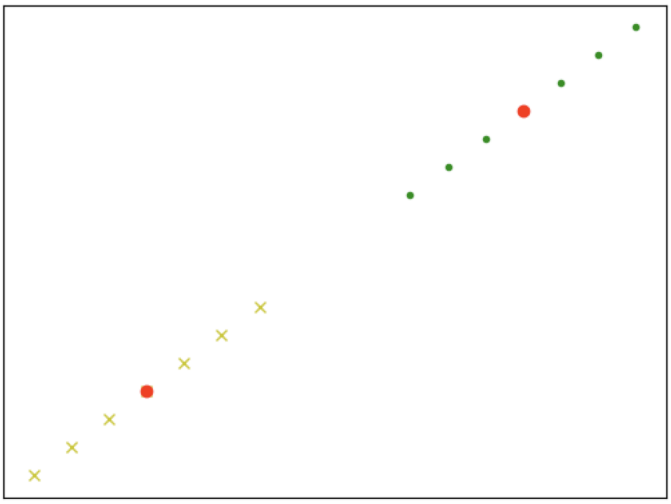

圖 2 執行結果

Python

範例程式：03-Iris-kmeans.py

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.cluster import KMeans
from sklearn import metrics
# Load the diabetes dataset
iris = datasets.load_iris()
iris_X_train , iris_X_test , iris_y_train , iris_y_test = train_test_split(iris.data,iris.target,test_size=0.2)
# KMeans 演算法
kmeans  = KMeans(n_clusters = 3)
kmeans_fit =kmeans.fit(iris_X_train)
print("實際",iris_y_train)
print("預測",kmeans_fit.labels_)
#調整標籤的數字
iris_y_train[iris_y_train==1]=11
iris_y_train[iris_y_train==0]=10
iris_y_train[iris_y_train==11]=0
iris_y_train[iris_y_train==10]=1
print("調整",iris_y_train)
score = metrics.accuracy_score(iris_y_train,kmeans.predict(iris_X_train))
print('準確率:{0:f}'.format(score))

```

執行結果

實際 [2 1 2 2 1 2 0 2 0 0 1 0 2 1 0 2 0 1 0 0 2 1 2 2 1 2 0 0 2 1 0 0 2 2 2 1 0

0 1 1 1 2 2 2 0 1 1 2 0 1 1 0 0 2 0 0 0 2 1 2 0 0 0 0 2 1 0 1 0 1 1 1 1 0

2 0 1 1 0 0 1 1 1 1 2 0 1 1 2 1 0 0 1 1 2 2 2 0 2 0 1 1 1 1 1 1 1 2 2 2 2

2 2 2 1 1 2 0 2 0]

預測 [0 0 0 2 0 2 1 2 1 1 0 1 2 0 1 0 1 0 1 1 0 0 2 2 0 0 1 1 2 0 1 1 2 2 2 0 1

1 0 0 0 2 2 2 1 2 0 2 1 0 0 1 1 2 1 1 1 2 0 2 1 1 1 1 2 0 1 0 1 0 0 0 0 0 1

0 1 0 0 1 1 0 0 0 0 2 1 0 0 2 0 1 1 0 0 2 0 2 1 0 1 0 0 0 0 0 0 0 0 2 2 2 0

2 2 0 0 0 0 1 2 1]

調整 [2 0 2 2 0 2 1 2 1 1 0 1 2 0 1 2 1 0 1 1 2 0 2 2 0 2 1 1 2 0 1 1 2 2 2 0 1

Python

```
1 0 0 0 2 2 2 1 0 0 2 1 0 0 1 1 2 1 1 1 2 0 2 1 1 1 1 2 0 1 0 1 0 0 0 0 1
2 1 0 0 1 1 0 0 0 0 2 1 0 0 2 0 1 1 0 0 2 2 2 1 2 1 0 0 0 0 0 0 0 2 2 2 2
2 2 2 0 0 2 1 2 1]
準確率:0.900000
```

這個時候 k-means 就可以拿出來使用了，首先需要先設定好 K 的數量，這裡我們用 K=2，然後以這一個紅色的位置的位置，來尋找一下附近最靠近的三個水果，透過畫出一個灰色的圓形，可以看得出來在這個範圍之中的三個水果都是檸檬，所以 大膽的說這一個未知的物體就是檸檬。

範例程式： 01-kNN-Mat.py

```
import matplotlib.pyplot as plt
import numpy as np
plt.plot([9,9.2,9.6,9.2,6.7,7,7.6], [9.0,9.2,9.2,9.2,7.1,7.4,7.5 ], 'yx' )
plt.plot([7.2,7.3,7.2,7.3,7.2,7.3,7.3 ], [10.3,10.5,9.2,10.2,9.7,10.1,10.1 ], 'g.' )
plt.plot([7], [9], 'r^' )
circle1=plt.Circle((7,9),1.2,color='#eeeeee')
plt.gcf().gca().add_artist(circle1)
plt.axis([6, 11, 6, 11])
plt.ylabel('H cm')
plt.xlabel('W cm')
plt.legend(('Orange','Lemons'),
        loc='upper right')
plt.show()
```

執行結果

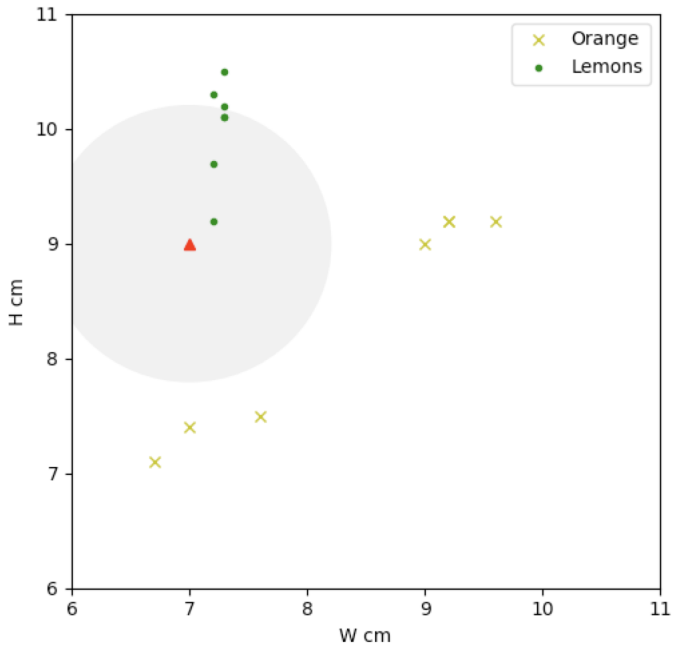

圖 1 執行結果

演算法邏輯

kNN 分類演算法簡單來說就是要找和新數據最近的 K 個鄰居，這些鄰居是什麼分類，那麼新數據就是什麼樣的分類。

現在給定一個特徵樣本 (我們稱為訓練集合)，我們輸入一個新樣本要把一個該樣本分藍色、紅色，我們可以從訓練集合找跟這新樣本距離最近的 K 個特徵樣本，看這些 K 個點是什麼顏色，來決定該點的最終顏色。

20.2 使用 sklearn 的 KNN 判斷水果種類

在本章節的範例之中將要使用 KNN，透過收集到的檸檬和柳丁的體積大小寬度和高度之間的訓練資料，並加以判別當新未知的水果量測相關的寬度和高度之後，並使用 KNN 的計算法，來判別這個位置水果到底是檸檬還是柳丁。

Python

範例程式

```
1. from sklearn.neighbors import KNeighborsClassifier      #匯入 KNN 函示庫
2. X=[[9,9],[9.2,9.2],[9.6,9.2],[9.2,9.2],[6.7,7.1],[7,7.4],[7.6,7.5],
3.    [7.2,10.3], [7.3,10.5], [7.2,9.2], [7.3,10.2], [7.2,9.7], [7.3,10.1], [7.3,10.1]]
4. y=[1,1,1,1,1,1,1,
5.    2,2,2,2,2,2,2]
6. neigh = KNeighborsClassifier(n_neighbors=3)             #使用 KNN, K=3
7. neigh.fit(X, y)                                         #訓練
8. print("預測答案＝",neigh.predict([[7,9]]))               #預測
9. print("預測樣本距離＝",neigh.predict_proba([[7,9]]))  # 測試數據 X 的返回概率估計。
```

執行結果

預測答案＝ [2]

預測樣本距離＝ [[0. 1.]]

20.3 實戰案例-鳶尾花的種類判斷

在這一個章節，將用植物數據範例，來探討 KNN 的在農業上的研究，這個植物數據資料來源是由，scikit-learn 函示庫中還附帶一些開發練習時的數據集，load_iris 鳶尾花數據集。

在本章節，將介紹植物學家透過尋找數據分析，對每個鳶尾花進行分類，而本章將會根據萼片和花瓣的長度和寬度測量來分類鳶尾花。

花萼是一朵花中所有萼片的總稱，位於花的最外層，一般是綠色，樣子類似小葉，但也有少數花的花萼樣子類似花瓣，有顏色。花萼在花還是芽時包圍著花，有保護作用。

本章節將會使用 load_iris 鳶尾花數據集，這是一個判別花的種類的數據集，主要包括 150

筆數據，4 個屬性值，分別是：

- Sepal Length 花萼長度
- Sepal Width 花萼寬度
- Petal Length 花瓣長度
- Petal Width 花瓣寬度

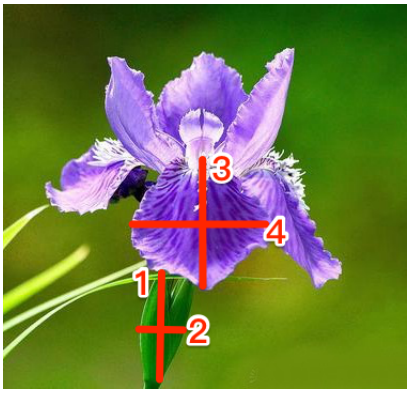

圖 2 鳶尾花的花萼和花瓣

而結果的部分 Target，鳶尾花目前有 300 多種，但範例的數據庫將只有以下三種：

- 柔滑鳶尾花 Iris setosa
- 弗吉尼亞鳶尾花 Iris virginica
- 雜色鳶尾花 Iris versicolor

圖 3 左到右，分別是 setosa, virginica, versicolor

透過以下的程式，確認相關函示庫是否有安裝成功，並取得版本編號

20.3.1 鳶尾花資料下載和存到 Excel 檔案

首先將透過以下的程式將資料下載取得，並瞭解這個鳶尾花的數據的樣貌。這個鳶尾花數據的特徵值只有 4 種，而判別的種類 Target 有三種，在本章節將透過 pandas 函式庫，將所取得的數值儲存在 Excel 表之中，這樣的話方便觀看這一個鳶尾花數據的內容。

範例程式 03-Iris.py

```
1.  import numpy as np                                          # 矩陣函示庫
2.  from sklearn import datasets                                # 範例數據函示庫
3.  from sklearn.neighbors import KNeighborsClassifier          # KNN 函示庫
4.
5.  # 取得鳶尾花的數據
6.  iris = datasets.load_diabetes()
7.  print("iris.data.shape=",iris.data.shape)                   # 輸出(150, 4)
8.  print("dir(iris)",dir(iris))
    # 輸出['DESCR', 'data', 'feature_names', 'target', 'target_names']
9.  print("iris.target.shape=",iris.target.shape)              # 輸出 (150,)
10. try:
11.   print("iris .feature_names=",iris .feature_names)        # 顯示特徵值名稱
12. except:
13.   print("No iris.feature_names=")
14. import xlsxwriter                                          # Excel 函示庫
15. import pandas as pd                                        # pandas 函示庫
16. # 轉換資料型態
17. try:
18.   df = pd.DataFrame(iris .data, columns=iris .feature_names)  # 處理特徵值
19. except:
20.    df = pd.DataFrame(iris .data, columns= ['sepal length (cm)', 'sepal width (cm)', 'petal
    length (cm)', 'petal width (cm)'])
21. df['target'] = iris.target                                 # 處理結果 Target
22.
23. #print(df.head())                                          # 顯示前五筆資料
24. df.to_csv("iris .csv", sep='\t')                           # 儲存到 CSV
25.
26. writer = pd.ExcelWriter('iris .xlsx', engine='xlsxwriter')  # 儲存到 Excel
27. df.to_excel(writer, sheet_name='Sheet1')
```

Python

```
28.  writer.save()
```

執行結果

iris.data.shape= (150, 4)

dir(iris) ['DESCR', 'data', 'feature_names', 'target', 'target_names']

Backend TkAgg is interactive backend. Turning interactive mode on.

iris.target.shape= (150,)

iris.feature_names= ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']

	A	B	C	D	E	F
1		sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
2	0	5.1	3.5	1.4	0.2	0
3	1	4.9	3	1.4	0.2	0
4	2	4.7	3.2	1.3	0.2	0
5	3	4.6	3.1	1.5	0.2	0
6	4	5	3.6	1.4	0.2	0
7	5	5.4	3.9	1.7	0.4	0
8	6	4.6	3.4	1.4	0.3	0
9	7	5	3.4	1.5	0.2	0
10	8	4.4	2.9	1.4	0.2	0
11	9	4.9	3.1	1.5	0.1	0
12	10	5.4	3.7	1.5	0.2	0
13	11	4.8	3.4	1.6	0.2	0
14	12	4.8	3	1.4	0.1	0
15	13	4.3	3	1.1	0.1	0
16	14	5.8	4	1.2	0.2	0
17	15	5.7	4.4	1.5	0.4	0
18	16	5.4	3.9	1.3	0.4	0
19	17	5.1	3.5	1.4	0.3	0
20	18	5.7	3.8	1.7	0.3	0

圖 4 執行結果

20.3.2 使用 KNN 判別鳶尾花的種類

在章節中將透過 KNN 的方法，訓練已知的鳶尾花的種類，找出其關聯性，並且預測出未知的鳶尾花，並預測該鳶尾花的種類。

範例程式 09-LinearRegression-diabetes.py

Python

```
1.  import matplotlib.pyplot as plt                                  # 繪圖函示庫
2.  import numpy as np                                               # 矩陣函示庫
3.  from sklearn import datasets                                     # 範例數據函示庫
4.  from sklearn.neighbors import KNeighborsClassifier               # KNN 函示庫
5.  from sklearn.model_selection import train_test_split             # 切割資料函示庫
6.
7.  iris = datasets.load_iris()   # 取得鳶尾花的數據
8.
9.  # 切割 80%訓練和 20%的測試資料
10. iris_X_train , iris_X_test , iris_y_train , iris_y_test =
    train_test_split(iris.data,iris.target,test_size=0.2)
11.
12. # 研究和計算
13. knn = KNeighborsClassifier()                                    # 建立 KNN
14. knn.fit(iris_X_train, iris_y_train)                             # 訓練
15.
16. print("預測",knn.predict(iris_X_test))
17. print("實際",iris_y_test)
18. print('準確率: %.2f' % knn.score(iris_X_test, iris_y_test))
```

執行結果:

預測 [1 0 2 1 0 2 2 1 1 2 1 0 0 1 0 0 1 2 2 2 2 2 0 1 2 0 0 2 1 1]

實際 [1 0 2 1 0 2 2 1 1 2 1 0 0 1 0 0 1 2 2 1 1 2 2 0 2 2 0 0 2 1 1]

準確率: 0.93

為什麼準確率只有 93%？這個程式所判別出來的預測結果，還是會有一筆的答案和實際是不一樣的，在實際的分類的資料很難會出現 100%，這就是數理統計實際的情況，所以改善的方法需要再補充大量的數據讓準確率再更精準一點，準確能夠再高一些。