

Table of Contents

第 19 章 數據分析演算法-KNN 最近鄰居法.....	1
19.1 KNN 數學介紹.....	1
19.2 使用 sklearn 的 KNN 判斷水果種類.....	4
19.3 實戰案例-鳶尾花的種類判斷.....	5
19.3.1 鳶尾花資料下載和存到 Excel 檔案.....	6
19.3.2 使用 KNN 判別鳶尾花的種類.....	8

第19章 數據分析演算法-KNN 最近鄰居法

19.1 KNN 數學介紹

KNN (K Nearest Neighbor)最近鄰居法又譯 K-近鄰算法，此演算法在非常重要的分析演算法之一，以目前來說是廣泛使用的演算法，用於分類和回歸的非參數統計方法，輸出是分類。KNN 是所有的機器學習算法中最簡單也是使用最廣的演算法之一。

其數學原理透過 k 個最近的鄰居，並依照這些鄰居的分類，決定了賦予該對象的類別。也就是由其鄰居的“多數表決”確定的， k 個最近鄰居（ k 為正整數，通常較小），KNN 最近鄰居法採用向量空間模型來分類，概念為相同類別的案例，彼此的相似度高，而可以藉由計

算與已知類別案例之相似度，來評估未知類別案例可能的分類。

KNN(K-近鄰算法)的缺點是對數據的局部鄰居非常敏感，並且留意 KNN 與 k-means(K-平均算法)沒有任何關係，是二種不同的演算法，請勿混淆。

數學公式和原理：

假設我們有 $(X_1, Y_1), (X_2, Y_2), \dots, (X_n, Y_n)$ 取值 $R^d \times \{1, 2\}$ ，其中 Y 是 X 的類標籤，因此 $X \mid Y=r \sim P_r$ 而 $r=1,2$ (和概率分佈 P_r)。給定一些規範

$\|\cdot\|$ 在 R^d 和 $x \in R^d$ ，讓 $(X_{(1)}, Y_{(1)}), \dots, (X_{(n)}, Y_{(n)})$ 訓練數據的重新排序為 $\|X_{(1)} - x\| \leq \dots \leq \|X_{(n)} - x\|$ 。

而計算 KNN 彼此之間的距離，可以用以下的公式：

$$\text{Similarity}(A,B) = \frac{A \cdot B}{\|A\| * \|B\|} = \frac{\sum_{i=1}^n A_i * B_i}{\sqrt{\sum_{i=1}^n A_i^2} * \sqrt{\sum_{i=1}^n B_i^2}}$$

再用剛剛的柳丁和檸檬的範例來看一下 KNN 演算法，透過以下的程式，同樣的也把柳丁和檸檬的大小，把它放在圖表上面，這時候有一個未知的紅色的新的物體，一樣的把該物體的寬度和高度量出來之後，並且在圖表上面用紅色的三角形來表示，這個時候 KNN 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 $K=3$ ，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

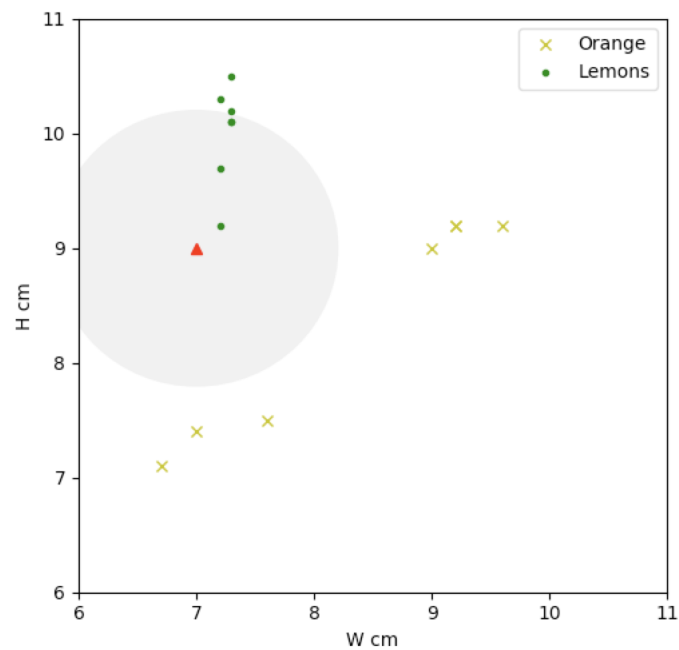


圖 1 執行結果

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。

19.2 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。

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.]]

19.3 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

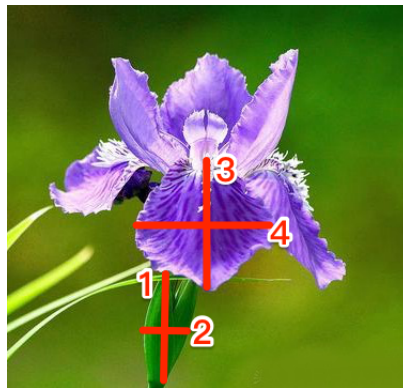


圖 2 鳶尾花的花萼和花瓣

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

19.3.1 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_iris()
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:
```

Python

```
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 xlswriter                                             # 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='xlswriter')    # 儲存到 Excel
27. df.to_excel(writer, sheet_name='Sheet1')
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

19.3.2 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

```

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

```


Python

```
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 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。