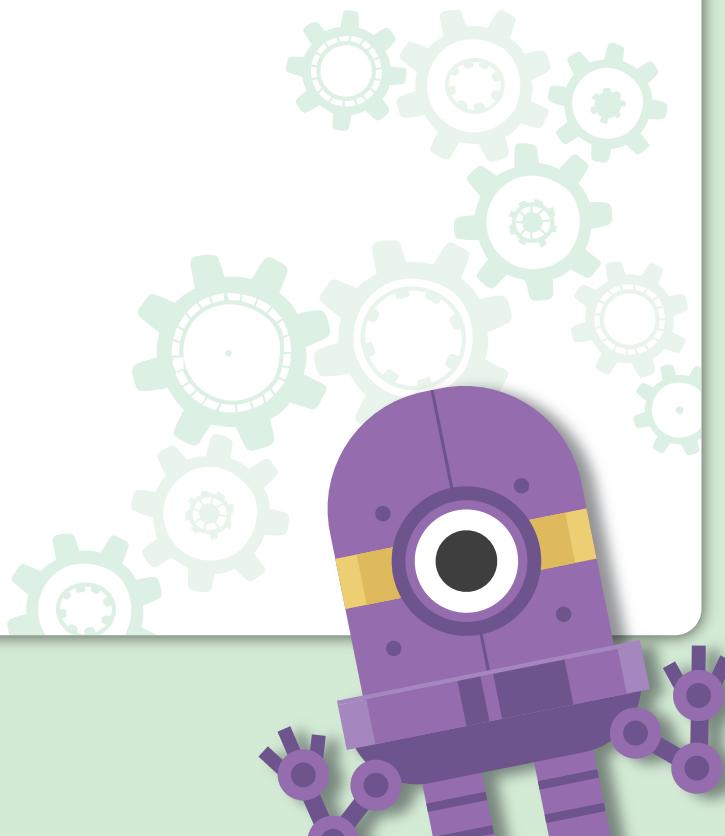


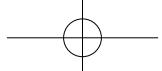
4



動手實作非監督式學習

-
- 4-1 K 平均分群演算法（K-means）的基本介紹與應用
 - 4-2 燕尾花的分群 -K-means 應用範例
 - 4-3 手寫數字的分群 -K-means 應用範例
 - 4-4 總結





上一章我們介紹了**監督式學習（Supervised Learning）**的迴歸與分類兩種應用。無論是房價預測或是乳癌診斷，都是使用**標籤資料（Label Data）**讓機器進行學習。不過，要將大量的資料進行標籤，這個繁複的任務對人們來說，實在是一件苦差事！

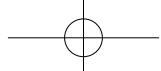
著名的電影訂閱網站 Netflix，每個月有成千上萬使用者觀看線上影片，且觀看電影節目超過 12,500 部，為了用機器學習自動推薦給會員個人化的影片，Netflix 除了透過直接的問答讓用戶選擇喜愛的類型電影，透過追蹤並分析用戶的網路點擊取得行為模式，最後由影視專家們協助將電影——手動標記更細緻的類別標籤，才能讓各種數據結合並發揮機器學習的更高效能。

但如果我們沒有時間一一標註資料，能不能讓機器自動分析資料中的特徵，自己學會如何做出判斷呢？例如：資料庫裡有一堆照片，能不能讓電腦自動學會將照片分群？亦或者有一堆零售店會員的資料，電腦能不能透過歷史消費行為自動將會員分群？

針對這些沒有標籤的資料，確實也能夠進行機器學習，我們稱之為**非監督式學習（Unsupervised Learning）**，方法通常有**分群分析（Cluster Analysis）**、**關聯規則（Association Rule）**、**維度縮減（Dimensionality Reduce）**等。

電腦的非監督式學習方式，也同樣是模擬人類的學習模式之一。瑞士心理學家皮亞傑（Jean Piaget）曾經提出的一個**認知發展理論（Theory of Cognitive Development）**，他認為人類與生俱來就具備了某種程度的**基模（Schema）**，可以透過自我不斷的同化與調適，最終會達成一個平衡結果而完成學習目的，無需外在強制力量介入，也就是說不一定要有其他師長的教導，就能夠自我學習。

本章我們就來介紹非監督式學習的分群分析中，常應用的演算法：**K 平均分群演算法（K-means）**。



知識補充

分群 (Clustering)

分群 (Clustering)，又可翻譯為「聚類」，是一種非監督式學習 (Unsupervised learning)，簡單地說就是把相似的資料歸到同一群。群內的資料越相似，分群的效果越好。但我們無法預估分群的結果將會是以何種特徵做出切分，例如：讓電腦對一堆動物的圖像進行分群，結果可能是將「四隻腳」或「兩隻腳」的動物分成兩群，也有可能將「有角」或「沒有角」的動物分成兩群。

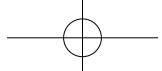
4-1 K 平均分群演算法 (K-means) 的基本介紹與應用

K 平均分群演算法 (K-means) 概念很簡單，就是「物以類聚」。例如在一個學生班級裡，通常會有小團體群聚的現象，例如：男生通常會自己聚成一群，女生也會自己聚成一群，愛運動的聚成一群，功課好的聚成一群等等，而每個小團體中通常會有一個中心人物。演算法 K-means 就有點類似這種小社會現象，K 是指將一大群未標籤的資料分成 K 個群，而 means 則是指 K 個分群中的質心點（可以想像成小團體的中心人物）。

K-means (K 平均分群演算法) 與上一章所介紹的監督式學習分類器 KNN (K- 最近鄰演算法) 不同，雖然都有「群聚」的概念，但 K-means 的 K 是分成 K 群，KNN 的 K 則從 K 個最靠近的鄰居來決定樣本資料的分類。

◎表 4-1 K-means 與 KNN 的比較

演算法	學習方法	資料	K 的意涵	學習目標	應用案例
K-means	非監督式學習	沒有標籤	分成 K 群	分群	將動物分成 K 群
KNN	監督式學習	有標籤	參考鄰近 K 個樣本數	分類	分辨動物是哪一種類



K-means 模型原理

1. 先設定將資料分成 K 群。
2. 在 d 維的資料空間，隨機指定 K 個質心。
3. 將每個資料點對 K 個質心計算歐式距離^{註1}（亦即直線距離公式）。
4. 將每筆資料分類判給距離最近的那個質心。
5. 每個質心內都會有被分類過來的資料，用這些資料點的中心位置再更新一次新的質心點。
6. 一直重複步驟 3 ~ 5，直到所有質心不再有太大的變動（收斂），結束。

K-means 模型概念圖解

若舉身高、體重兩個資料為例，在二維的平面將 12 位學生的身高、體重標示成座標點，K-means 分群將圖 4-1 與以下步驟：

Step1 設 K=2。

Step2 在二維資料空間，隨機選 C_1 、 C_2 為初始的質心。

Step3 將 12 個資料點分別對 C_1 、 C_2 的質心計算歐式距離。

Step4 將每筆資料分類判給距離最近的那個質心，因此可畫出一條分類線，將所有資料點分開成兩群。

Step5 將兩群資料點重新計算中心位置，找出新的質心點。

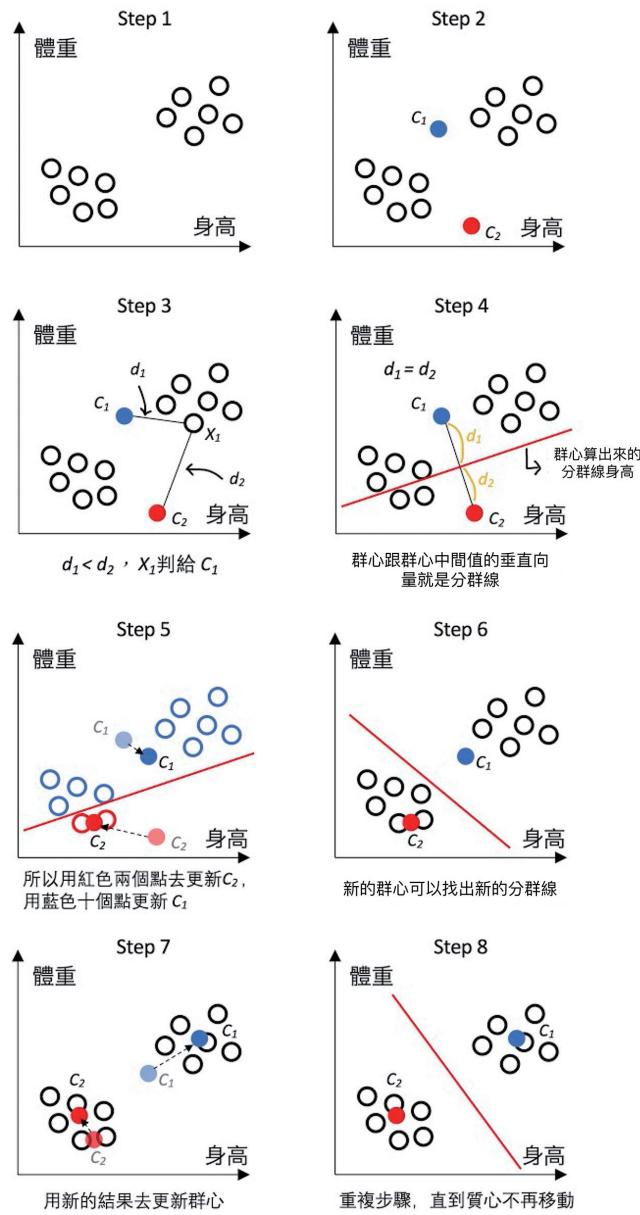
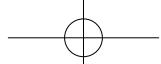
Step6 再次將每個資料點對新的質心計算歐式距離，重新將資料分群給新的質心，畫出新的分群線。

Step7 再次用兩群資料點重新計算中心位置，找出新的質心點。

Step8 重複步驟直到質心不再移動，就完成最終分群。

註 1 歐式距離又稱歐幾里得距離，用於計算兩點間的直線距離，計算公式如下：

$$d(x, y) := \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \dots + (x_n - y_n)^2}$$

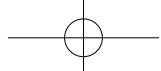


◆ 圖 4-1 K-means 模型概念圖解

K-means 的運作主要有重複執行兩個步驟：

1. 測量資料與質心距離來分群。
2. 從分群中產生新的質心，便可以將資料有效的分群。

如果已經了解 K 平均分群演算法的概念，接下來就著手用 Python 程式來實作分群的解題應用。



4-2

鳶尾花的分群 -K-means 應用範例

如果對於沒有標籤的資料要進行探索性的資料分析，K 平均分群演算法是常用的演算法，接下來我們要用一組花朵的資料集，運用非監督式學習的方式，讓電腦自己從資料中找特徵，並做出分群的判斷。

4-2-1 鳶尾花資料集介紹

鳶尾花資料集，亦稱安德森鳶尾花卉數據集（英文：Anderson's Iris data set），也稱費雪鳶尾花卉數據集（英文：Fisher's Iris data set）。它最初是由美國的植物遺傳學家埃德加·安德森（Edgar Anderson）從加拿大加斯帕半島上所蒐集來的鳶尾屬花朵資料，目的是想量化雜色鳶尾花在不同地理環境的變異，後來又由英國的統計學家羅納德·費雪（Ronald Fisher）拿來用在統計學領域，作為特徵判別分析的一個例子。

鳶尾花資料集包含了 150 個樣本，都屬於鳶尾屬，分別是山鳶尾、變色鳶尾和維吉尼亞鳶尾。用這三種花的四個特徵：花萼、花瓣的長度和寬度，以 K 平均分群演算法來實作看看花的分群預測結果是否準確。

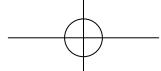
資料中除了鳶尾花的特徵，也提供相對應鳶尾花的類別資料^{註2}：

1. 山鳶尾（Iris-setosa）
2. 變色鳶尾（Iris-versicolor）
3. 維吉尼亞鳶尾（Irisvirginica）

◎表 4-2 摘取一部分鳶尾花資料集的數據概況

花萼長度	花萼寬度	花瓣長度	花瓣寬度	屬種
4.4	2.9	1.4	0.2	setosa
4.4	3.0	1.3	0.2	setosa
5.0	2.0	3.5	1.0	versicolor
5.0	2.3	3.3	1.0	versicolor
6.1	3.0	4.9	1.8	virginica
6.1	2.6	5.6	1.4	virginica

註 2 類別資料：雖然鳶尾花資料集是有類別的資料集，但是本章的機器學習過程並不會用到類別資料，因為我們要介紹的是非監督式學習的分群方法。

◆ 圖 4-2 山鳶尾 (*Iris-setosa*)◆ 圖 4-3 變色鳶尾 (*Iris-versicolor*)◆ 圖 4-4 維吉尼亞鳶尾 (*Irisvirginica*)

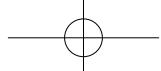
4-2-2 鳶尾花資料集實作

Step1 載入所需套件

鳶尾花資料集來源是在著名開源機器學習工具庫 scikit-learn 上，與第三章實作內容相同，我們會先下載所需的資料集與引用相關套件。其中有一個套件是比較特別的，專門用來繪製 3D 立體圖像的 Axes3D。

```
# 從 sklearn 載入鳶尾花資料集套件
from sklearn.datasets import load_iris

# 載入我們會用到的模型，K-means 模型
from sklearn.cluster import KMeans
```



```
# 載入用來做資料視覺化的畫圖套件
import matplotlib.pyplot as plt

# 載入 3D 繪圖工具
from mpl_toolkits.mplot3d import Axes3D
```

Step2 載入資料集，觀察資料集

當下載完資料集我們依序來觀察 150 筆鳶尾花資料集分別提供哪些資訊。

```
# 載入資料集，放到 digits 變數內
iris = load_iris()

# 觀察有哪些 key 在資料集內
print(iris.keys())
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_
names', 'filename'])

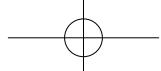
# 觀察資料筆數
print(" 資料筆數 :")
print(iris.data.shape, "\n")

# 觀察我們資料的欄位名稱
print(" 資料的欄位名稱，分別是 :")
print(iris.feature_names, "\n")

# 觀察我們第一筆的資料內容
print(" 第一筆的資料內容 :")
print(iris.data[0], "\n")

# 觀察我們第一筆的預測目標
print(" 第一筆的預測目標 :")
print(iris.target[0], "\n")
```

執行以上程式碼，我們會得到輸出結果，如圖 4-5 所示。總共有 150 筆花的資料，每一筆都有花萼長度 sepal length (cm)、花萼寬度 sepal width (cm)、



花瓣長度 petal length (cm)、花瓣寬度 petal width (cm) 四個特徵，將第一筆資料用 Print 印出來，分別可得到它的數值，而標籤是「0」也就是三種花的第一種 Setosa。

資料筆數：
(150, 4)

資料的欄位名稱，分別是：

```
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

第一筆的資料內容：
[5.1 3.5 1.4 0.2]

第一筆的預測目標：
0

◆ 圖 4-5 燕尾花資料集資訊

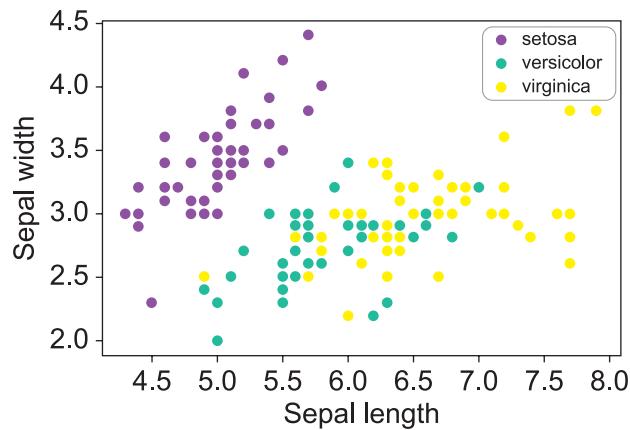
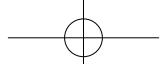
接著，我們可以利用資料視覺化的方法來進一步觀察，燕尾花的特徵共有四種，等於有四個維度，但因為平面圖只有兩個維度，所以我們分別挑選兩個特徵，來繪製圖表。

```
# 繪製散佈圖，利用花萼長度及花萼寬度作圖，不同的顏色代表不同種的燕尾花
scatter = plt.scatter(iris.data[:,0], iris.data[:,1], c=iris.target)

# 我們可以在 x,y 軸設定標題名稱
plt.xlabel('Sepal length')
plt.ylabel('Sepal width')

# 可以設定一個說明視窗，裡面分別放置顏色點及對應的類別名稱
# 第一個參數設定我們的點，第二個參數設定名稱
plt.legend( handles= scatter.legend_elements()[0], labels= iris.
target_names.tolist())
```

用花萼長度及花萼寬度作圖，如圖 4-6 所示，紫色的點是山燕尾 setosa、綠色的點是變色燕尾 versicolor、黃色的點是維吉尼亞燕尾 virginica，以花萼長度來說維吉尼亞燕尾偏長、山燕尾則偏短，以花萼寬度來說山燕尾則偏寬。而這就是我們畫成散佈圖與單純觀察數值資料不同的地方，圖表可以讓我們更直觀的判斷不同種類花朵的特徵差異之處。

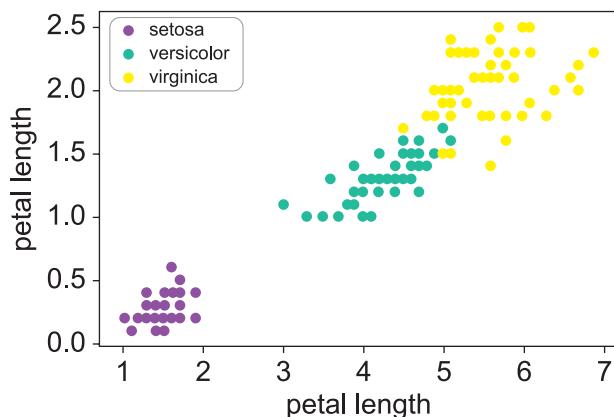


◆ 圖 4-6 三種鳶尾花的花萼長寬散佈圖

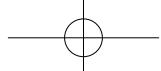
接著我們改用花瓣長度及花瓣寬度作圖

```
# 繪製散佈圖，利用花瓣長度及花瓣寬度作圖，不同的顏色代表不同種的鳶尾花  
scatter = plt.scatter(iris.data[:,2], iris.data[:,3], c=iris.target)  
  
# 我們可以在 x,y 軸設定標題名稱  
plt.xlabel('petal length')  
plt.ylabel('petal width')  
  
plt.legend( handles= scatter.legend_elements()[0], labels= iris.  
target_names.tolist())
```

輸出結果如圖 4-7 所示，以花瓣長度來說山鳶尾偏短、維吉尼亞鳶尾偏長，以花瓣寬度來說山鳶尾偏窄、維吉尼亞鳶尾偏寬。



◆ 圖 4-7 三種鳶尾花的花瓣長寬散佈圖



Step3 訓練模型及驗證模型

我們使用 K 平均分群演算法，並將 K 設為 3（因為我們想測試這個模型是否能依三種鳶尾花不同的特徵，將花朵分成正確的三群）

```
# 載入 K-means 模型，設定將資料分成三群
estimator = KMeans(n_clusters=3)

# 進行模型訓練，因為 K-means 是非監督式學習，故不用放入 label 的標籤資料
estimator.fit(iris.data)
# 印出分群好的標籤來觀察
print(estimator.labels_)
```

◆ 圖 4-8 K-means 模型的分群結果輸出

我們完成了模型的訓練，並且將分群好的標籤印出來如圖 4-8 所示，但單純的數字無法很直接的觀察電腦分群的依據為何，因此我們一樣可以將模型分群的結果用散佈圖的方式呈現出來，並且與正確答案的散佈圖做比較，將更容易了解分群的成效。

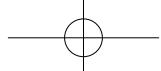
```
# 印出 K-means 分群好的質心點  
estimator.cluster_centers_
```

我們會得到輸出

```
array([[5.006    , 3.428    , 1.462    , 0.246    ],
       [5.9016129 , 2.7483871 , 4.39354839, 1.43387097],
       [6.85      , 3.07368421, 5.74210526, 2.07105263]])
```

代表分成三群後，三個質心點的數值

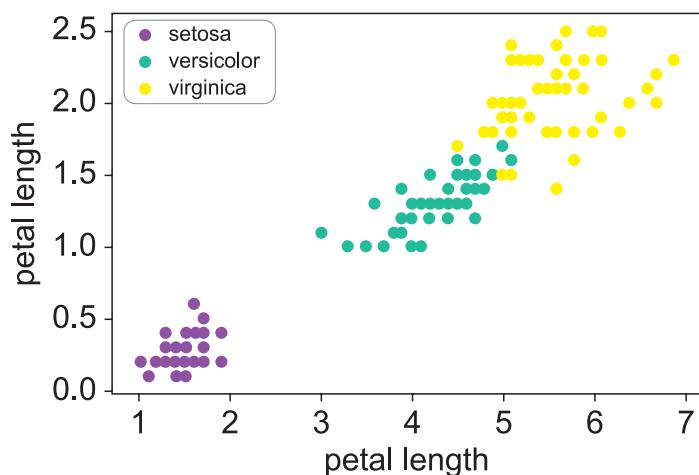
接著我們再用圖表的方式來比較真實資料與分群結果的差異：



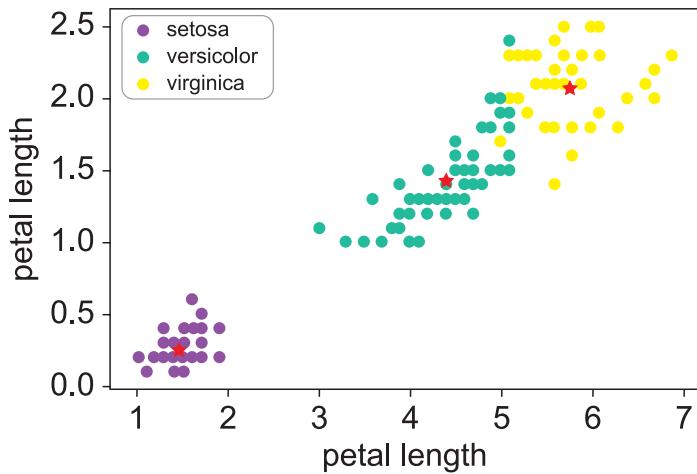
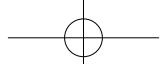
```
# 用原始的 label 來作圖
scatter = plt.scatter(iris.data[:,2], iris.data[:,3], c=iris.target)
plt.xlabel('petal length')
plt.ylabel('petal length')
plt.legend( handles= scatter.legend_elements()[0], labels= iris.
target_names.tolist())
plt.show()

# 用 K-means 分群好的 label 來作圖
scatter = plt.scatter(iris.data[:,2], iris.data[:,3], c=estimator.
labels_)
plt.xlabel('petal length')
plt.ylabel('petal length')

# 我們可以把 K-means 找到的質心位置給畫出來，用星號、配置紅色、s 為 size 的縮寫，及設置大小
plt.scatter(estimator.cluster_centers_[:,2], estimator.cluster_
centers_[:,3],marker='*', c='red' , s=100)
plt.legend( *scatter.legend_elements())
plt.show()
```



◆ 圖 4-9 取花瓣長寬特徵之真實花朵分類的散佈圖

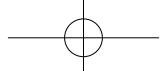


◆ 圖 4-10 取花瓣長寬特徵之 K-means 模型分群散佈圖

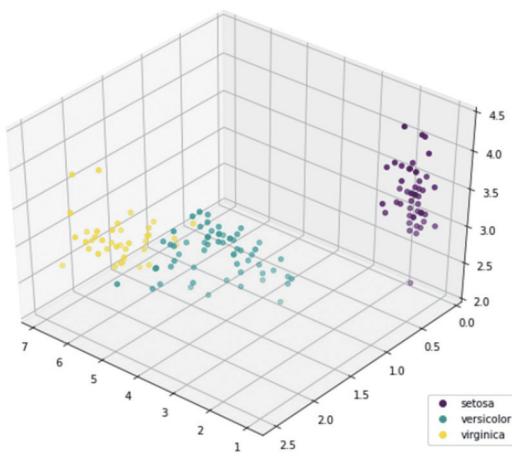
依據「花瓣長、寬」來繪製二維平面散佈圖，如圖 4-9 是正確的花朵分類，而圖 4-10 則是 K-means 的分群結果，可以發現紫色點的山鳶尾 setosa 都分對了，而綠色點變色鳶尾 versicolor、黃色點維吉尼亞鳶尾 virginica，則因為特徵較相近，因此少數幾個點，被分錯了群。不過大致上來說，兩個散佈圖是高度相似的，這表示我們的分群模型訓練得到不錯的成果。

接著我們更上層樓，使用 matplotlib 的 Axes3D 來繪製三維立體的視覺化圖示：

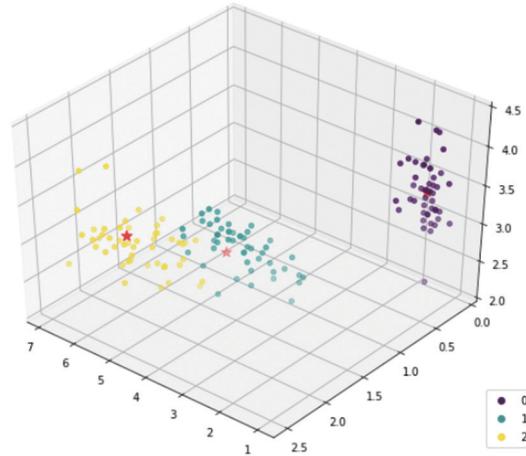
```
# 這邊的 3D 繪圖屬於較高階的作法，如果想要得到更詳細的資料可以到官網查詢：  
# https://matplotlib.org/mpl\_toolkits/mplot3d/tutorial.html#scatter-plots  
  
# 接下來三行是繪圖環境的設置  
# elev 是從不同的高度視角看過去  
# azim 是從不同的水平角度看過去  
fig = plt.figure(figsize=(8, 6))  
ax = Axes3D(fig, rect=[0, 0, .95, 1], elev=40, azim=130)  
ax.set_zlim(2, 4.5)  
# 首先我們先印出原始有正確標籤的資料  
# 依序用花瓣長度、花瓣寬度、花萼寬度這三個特徵畫圖，然後也需要 iris.target 標籤  
ax.scatter(iris.data[:, 2], iris.data[:, 3], iris.data[:, 1], c=iris.target)
```



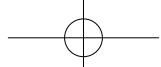
```
ax.scatter(estimator.cluster_centers_[:,2], estimator.cluster_centers_[:,3], estimator.cluster_centers_[:,1], marker='*', c='red', s=100)  
# 然後我們將圖例顏色點及對應的類別名稱放在右下角  
plt.legend(*scatter.legend_elements(), loc='lower right')  
plt.show()  
  
# 然後再將分群的資料匯成圖  
# 1.) 先將鳶尾花的資料匯出： iris.data[:, 2], iris.data[:, 3], iris.  
data[:, 1]  
# 2.) c 參數是顏色變數，三維空間的每一顆資料的顏色會根據 label 而決定，而  
label 是根據 estimator 也就是 K-means 訓練過的模型結果  
  
ax.scatter(iris.data[:, 2], iris.data[:, 3], iris.data[:, 1],  
c=estimator.labels_)  
plt.legend(handles= scatter.legend_elements()[0], labels= iris.  
target_names.tolist(), loc='lower right')  
plt.show()
```



◆ 圖 4-11 實真分類的鳶尾花散佈圖



◆ 圖 4-12 K-means 分群後的鳶尾花散佈圖



從 3D 立體的散佈圖來觀察，分群的結果確實也與真實的分類高度相似，因此我們可以透過這個實作初步體驗到，電腦採行非監督式學習的方式，確實能自行探索資料的特徵做出高精準度的判斷。

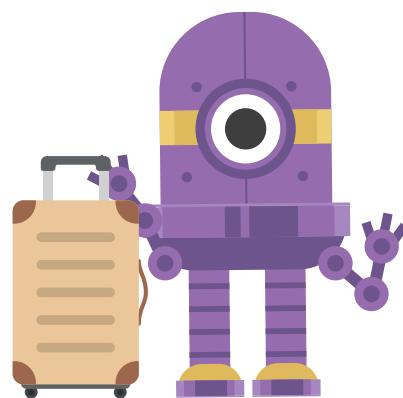


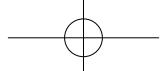
◆ 圖 4-13 莺尾花資料集實作流程圖



想想看

1. 如果分群程式重複執行多次，每次所繪出來的分群結果圖，還會是一樣的嗎？
2. 如果使用鳶尾花的另外三種特徵，用 K-means 模型程式訓練分群，結果會是如何？



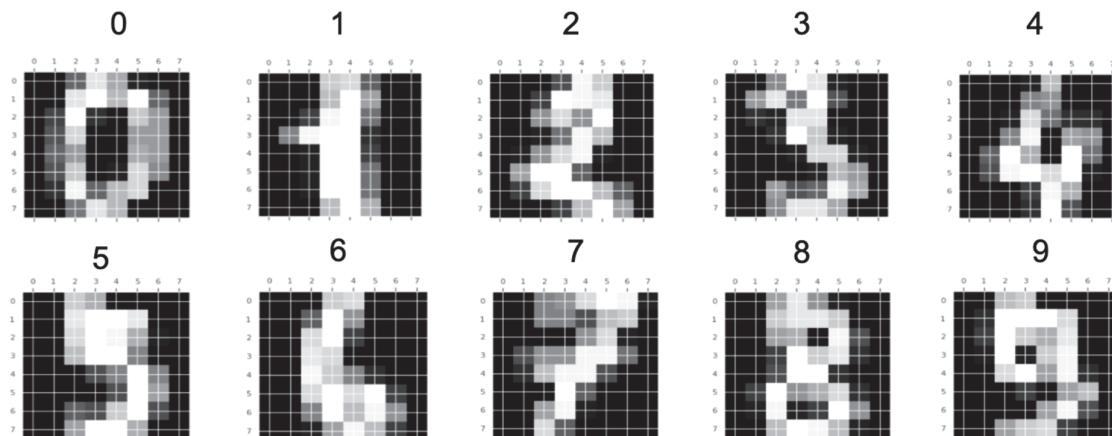


4-3 手寫數字的分群 -K-means 應用範例

K-means 分群方法不但可以用在如鳶尾花的統計數據上，在圖像分群應用上也很廣泛，如果用「k means image segmentation」為關鍵字在 Google 搜尋，可以找到很多案例，接下來我們就以手寫數字資料集來實作 K-means 分群。

4-3-1 手寫數字資料集介紹

手寫數字 digits 資料集來自於機器學習網站 scikit learn，共有 1797 個 0~9 手寫數字圖像，每個樣本均有成對的資料，分別是 8x8 像素的圖，以及對應的 0~9 整數標籤，如圖 4-14 所示。



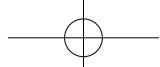
◆ 圖 4-14 scikit learn 的手寫數字 digits

4-3-2 手寫數字資料集實作

Step1 載入所需套件

我們需要載入資料集 digit，除此之外還需要載入 K-means 分群模型、matplotlib 視覺化的繪圖套件與 NumPy 數學運算套件。

```
# 從 sklearn 載入手寫辨識資料集
from sklearn.datasets import load_digits
```



```
# 載入我們會用到的模型，KMeans 分群模型
from sklearn.cluster import KMeans

# 載入用來做資料視覺化的畫圖套件
import matplotlib.pyplot as plt

# 載入做數學運算的套件 numpy
import numpy as np
```

Step2 載入與觀察資料集

手寫數字是圖像化的資料，我們載入資料之後，可進一步觀察資料的筆數、型態、圖像與標籤資訊等等。

```
# 載入手寫字資料庫到 digits 變數中
digits = load_digits()

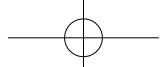
# 觀察有哪些 key 在資料集內
print(digits.keys())
dict_keys(['data', 'target', 'target_names', 'images', 'DESCR'])

# 觀察資料筆數
print(" 資料筆數 :")
print(digits.data.shape, "\n")

# 觀察我們資料的欄位名稱
print(" 資料的欄位名稱，分別是 :")
print(digits.target_names, "\n")

# 觀察我們第一筆的資料內容
print(" 第一筆的資料內容 :")
print(digits.data[0], "\n")

# 觀察我們第一筆的影像內容
print(" 第一筆的影像內容 :")
print(digits.images[0], "\n")
```



```
# 觀察我們第一筆的預測目標  
print(" 第一筆的預測目標 :")  
print(digits.target[0], "\n")
```

☛ 資料筆數：

(1797, 64)

資料的欄位名稱，分別是：

[0 1 2 3 4 5 6 7 8 9]

第一筆的資料內容：

```
[ 0.  0.  5. 13.  9.  1.  0.  0.  0. 13. 15. 10. 15.  5.  0.  0.  3.  
 15.  2.  0. 11.  8.  0.  0.  4. 12.  0.  0.  8.  8.  0.  0.  5.  8.  0.  
 0.  9.  8.  0.  0.  4. 11.  0.  1. 12.  7.  0.  0.  2. 14.  5. 10. 12.  
 0.  0.  0.  0.  6. 13. 10.  0.  0.  0.]
```

第一筆的影像內容：

```
[ [ 0.  0.  5. 13.  9.  1.  0.  0.]  
 [ 0.  0. 13. 15. 10. 15.  5.  0.]  
 [ 0.  3. 15.  2.  0. 11.  8.  0.]  
 [ 0.  4. 12.  0.  0.  8.  8.  0.]  
 [ 0.  5.  8.  0.  0.  9.  8.  0.]  
 [ 0.  4. 11.  0.  1. 12.  7.  0.]  
 [ 0.  2. 14.  5. 10. 12.  0.  0.]  
 [ 0.  0.  6. 13. 10.  0.  0.  0.]]
```

第一筆的預測目標：

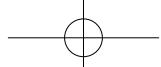
0

◆ 圖 4-15 觀察資料集

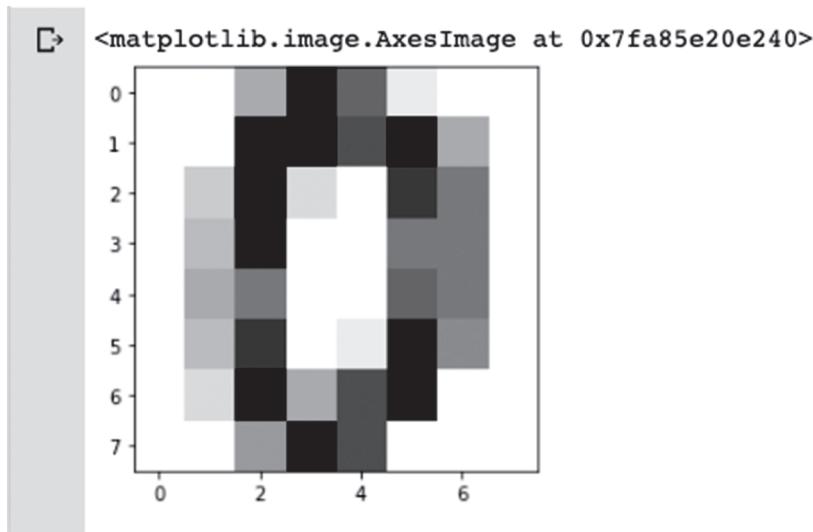
從 digits.data.shape 輸出結果 (1797, 64)，我們可知共有 1,797 張數字圖像，每一張圖有 64 個數字，是圖片的畫素資料（圖像原是 8×8 二維影像，攤平成一維就是 64）。每筆資料的 target_names 分別為數字 0、1、2、3、4、5、6、7、8、9，也是這些圖的標籤。

當輸出第一筆資料時，只能看到一群似乎沒有意義的數字，但如果轉換成二維的陣列，我們就幾乎可以從數字排列中觀察到，大於 0 的數值似乎圍成了一個圓，極可能就是 0 這個數字的圖像！

接著用 plt.imshow() 將像素值轉換成顏色，在電腦上顯示出來，以印證我們的猜測。



```
# 試試看用資料視覺畫的套件來畫出 image 影像資料  
# cmap 是指取用哪種顏色顯示，plt.cm.binary 是顯示灰階的顏色庫  
plt.imshow(digits.images[0] , cmap=plt.cm.binary)
```

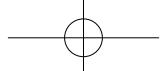


◆ 圖 4-16 第一筆資料的圖像

知識補充

什麼是像素？

像素，是影像顯示的基本單位，譯自英文「pixel」，pix 是英語單詞 picture 的常用簡寫，加上英語單詞「元素」element，就得到 pixel。在電腦的世界裡，所有的一切都是以數位的方式所組成，也就是最終會以 0 與 1 的方式呈現，讓電腦理解。而圖片是由許多的小方格所組合，就好像棋盤一般地構成一張圖片，如同上面數字 0 的手寫影像。在這些小方格裡，每個小方格都會存放一個代表這個顏色的數值，當夠多的小方格聚集，那麼我們在觀看圖片時，就彷彿是一張平滑的圖片。這些小方格我們稱之為像素（Pixel），所以說，圖片是由像素所組合而成的矩陣。



接下來我們可以將所有的數字都輸出，觀察一下這些手寫數字的圖像。

```
# 一次看前十筆的圖像資料
```

```
# 先畫第一列
```

```
# figsize 代表畫布的大小，這裡是希望設定成長 =10，寬 =10，單位是英寸
```

```
plt.figure(figsize=(10,10))
```

```
for i in range(1,6):
```

```
    plt.subplot(1,5,i)
```

```
    plt.imshow(digits.images[i], cmap=plt.cm.binary)
```

```
plt.show()
```

```
# 再畫第二列
```

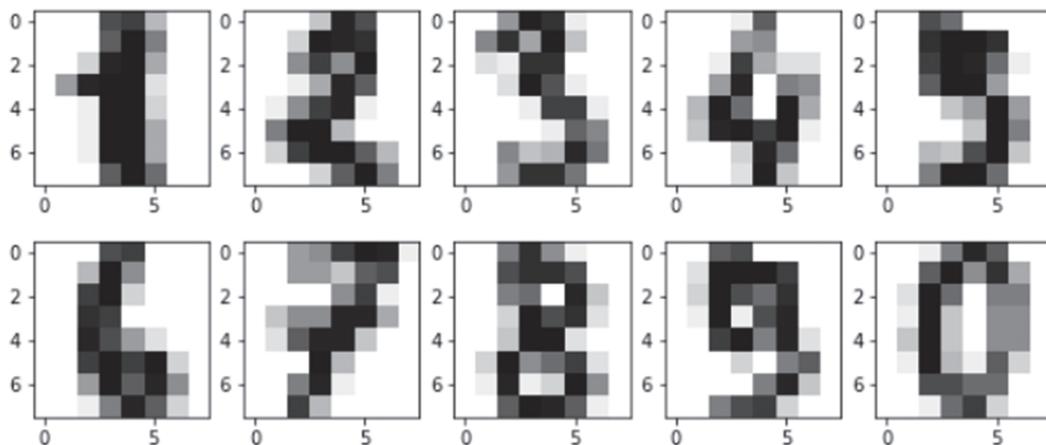
```
plt.figure(figsize=(10,10))
```

```
for i in range(6,11):
```

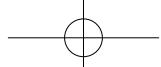
```
    plt.subplot(2,5,i)
```

```
    plt.imshow(digits.images[i], cmap=plt.cm.binary)
```

```
plt.show()
```



◆ 圖 4-17 前十筆資料分別是 0 ~ 9



Step3 訓練模型及驗證模型

觀察資料後，接下來我們就開始進到訓練模型的階段，因為 digits 有十種數字分類，使用 K-means 模型，我們將 K 設為 10，意即要分成十群，而且因為是非監督式學習，所以我們將不會使用到 target_names 這個標籤資料。

```
# 載入 K-means 模型，設定將資料分成十群
estimator = KMeans(n_clusters=10)

# 進行模型訓練，因為 K-means 是非監督式學習，故不用放入 label 的標籤資料
estimator.fit(digits.data)

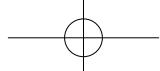
# 這邊分別取出 K-means 模型幫我們分好的其中三群出來看
# 分別就是 label 為 0,1,2 的三群，並把他們的位置 (index) 存到變數 c_0, c_1,
c_2 內
c_0 = np.where(estimator.labels_ == 0 )[0]
c_1 = np.where(estimator.labels_ == 1 )[0]
c_2 = np.where(estimator.labels_ == 2 )[0]

# 印出這三個變數的前十筆資料來看
# 他們代表著不同群集裡，分別在 digits 裡面的排序位置
print(c_0[0:10])
print(c_1[0:10])
print(c_2[0:10])
```

分群模型完成之後，我們輸出其中三個群的前十筆資料，結果是：

```
[ 1 2 18 28 38 40 50 51 53 57]
[ 0 10 20 30 36 48 49 55 72 78]
[ 3 13 23 45 59 60 62 63 83 89]
```

這代表 digits 資料集，第 1、2、18、28、38、40、50、51、53、57 筆，被電腦分為同一群，而第 0、10、20、30、36、48、49、55、72、78 筆被分到另一群，以此類推，而這到底分得好不好呢？我們得畫出圖來，才能判斷。



```
# 把這三個集群分成三列顯示出來看看
# 這下面有一個方法 interpolation='sinc'，他會讓影像用 sin 波取內插取值，讓影
像更清晰

# 先畫第一列
plt.figure(figsize=(20,20))

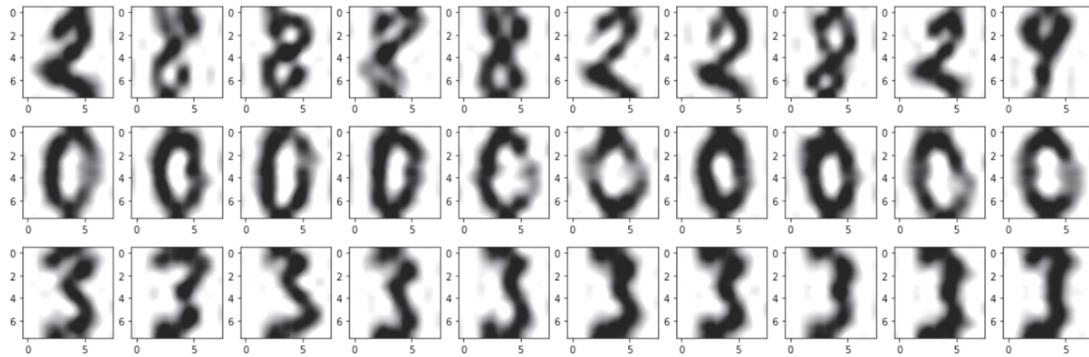
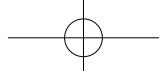
for i in range(1,11):
    plt.subplot(1,10,i)
    plt.imshow(digits.images[c_0[i]], cmap=plt.cm.binary, interpolation='sinc')
plt.show()

# 再畫第二列
plt.figure(figsize=(20,20))

for i in range(1,11):
    plt.subplot(2,10,i)
    plt.imshow(digits.images[c_1[i]], cmap=plt.cm.binary, interpolation='sinc')
plt.show()

# 再畫第三列
plt.figure(figsize=(20,20))

for i in range(1,11):
    plt.subplot(3,10,i)
    plt.imshow(digits.images[c_2[i]], cmap=plt.cm.binary, interpolation='sinc')
plt.show()
```



◆ 圖 4-18 K-means 分成十群後 取出其中三群圖像進行觀察

圖 4-18 就是我們用 K-means 模型分群後，用繪出圖像的方式，來呈現分群的成果。電腦自己依據圖像的特徵，將相似的分成一群，我們觀察到 0 與 3 數字圖像幾乎都分對了！而 2、8、9 這幾個數字，在電腦判斷起來則是較相似的，因此分成了同一群，讀者可以想想，這幾個數字到底有哪型態是相似的，以至於讓電腦做出了這樣的判斷。

因為我們知道數字本來就是 0~9，所以分十群是因為已經知道「答案」，但如果模擬非監督式學習的常態情況，我們其實不能精確判斷到底應該分成幾群，所以也可以試試從分三群或五群等來觀察看看 K-means 模型分群的結果會如何？程式如下，讀者可以參考並調整 K 值，來實驗不同的分群結果。

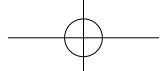
```
# 載入 kmeans 模型，設定將資料分成三群
estimator = KMeans(n_clusters= 3)
estimator.fit(digits.data)

c_0 = np.where(estimator.labels_== 0 )[0]
c_1 = np.where(estimator.labels_== 1 )[0]
c_2 = np.where(estimator.labels_== 2 )[0]

# 把這三個集群分成三列顯示出來看看

# 先畫第一列
plt.figure(figsize=(20,20))

for i in range(1,11):
```



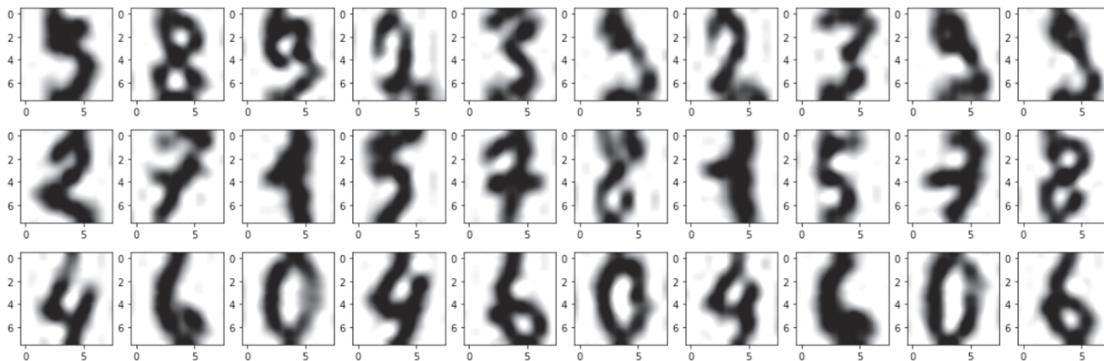
```
plt.subplot(1,10,i)
plt.imshow(digits.images[c_0[i]], cmap=plt.cm.binary, interpolation='sinc')
plt.show()

# 再畫第二列
plt.figure(figsize=(20,20))

for i in range(1,11):
    plt.subplot(2,10,i)
    plt.imshow(digits.images[c_1[i]], cmap=plt.cm.binary, interpolation='sinc')
    plt.show()

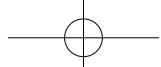
# 再畫第三列
plt.figure(figsize=(20,20))

for i in range(1,11):
    plt.subplot(3,10,i)
    plt.imshow(digits.images[c_2[i]], cmap=plt.cm.binary, interpolation='sinc')
    plt.show()
```



◆ 圖 4-19 K-means 分成三群後 取出三群圖像進行觀察

如圖 4-19 所示，便完成了手寫數字辨識 $K=3$ 分群模型。



與監督式學習不同，分群無法告訴我們一個預測數值或分類的答案，只能將一群資料做出分割，至於其中的意義，則需要自己去做出判斷與解釋。

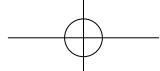


◆ 圖 4-20 手寫數字資料集實作流程圖



想想看

1. 為什麼把 K 設為 10，0 到 9 的圖像分群大致會按照真實數字分成 10 群？
2. 將數字分成 5 群，哪些數字可能被分到同一群？為什麼？



4-4

總結

上述範例是在不告訴電腦關於鳶尾花有哪幾類或是數字有十種（從 0~9），反而是一股腦地將資料全丟給電腦讓它自己去學，竟然可以得到還算準確的成果，是不是很神奇呢？

因為非監督式學習無需人工辛苦的標籤資料，因此在海量數據掛帥的金融商業領域應用非常廣泛，例如從大量的刷卡資料中，去評估某筆交易是否可能為詐欺；從網路會員的點擊行為中，去分析優化網頁轉換率等。

非監督式學習的優勢在於幫人類挖掘出資料中某些未知，但有意義的模式，將資料與資料間建立起關聯，但非監督式學習也有其缺點，那就是分群無法獲得有關數據排序的準確資訊，例如在 4-3 節手寫數字分群實作案例中，電腦並無法識別 0 ~ 9 的排列次序。因此非監督式學習的分群結果，還是需要由人主動去做評估，以找出其因果關係。

總結，本章的內容，我們學到的內容有：

- 非監督式學習的介紹。
- K-means 分群演算法的介紹。
- 經典資料集：鳶尾花資料集與手寫數字資料集的 K-means 實作。