巨匠線上真人

Python 資料科學應用開發

第十六堂：非監督式學習（Unsupervised Learning）

同學，歡迎你參加本課程

- 請關閉你的FB 、Line等溝通工具，以免影響你上課。
- 考量頻寬、雜音，請預設關閉攝影機、麥克風，若有需要再打開。
- 隨時準備好，老師會呼叫你的名字進行互動，鼓勵用麥克風提問。
- 如果有緊急事情，你必需離開線上教室，請用聊天室私訊給老師，以免老師癡癡呼喚你的名字。
- 軟體安裝請在上課前安裝完成，未完成的同學，請盡快進行安裝。

課程檔案下載


巨匠電腦線上真人
開課查詢
免費體驗專區
課程總覽
專業師資
學員專區
講師專區
最新消息
您好!
登出
點數卡產品兌換
APCS檢測專區
公告專區
我的課表
IT真人課程劃位
電腦分校課程劃位
外語真人課程劃
美語分校課程劃
取消劃位
課程檔案下載
上課權益查詢
教學平台測試
學習諮詢
常見問題
個資維護
忘記密碼
登出
課程檔案下載
程式語言好難學?
那是因為
你還沒學過Python!
線上老師LIVE直播教學．搶先看
巨匠電腦真人課程

ZOOM 學員操作說明


www.pcschool.com.tw
您正在觀看蘇世宗的螢幕
查看選項
原始大小
請求遠端控制
共同註記
退出全螢幕
5 查看選項/共同註記/筆（連連看）
滑鼠
文字
筆
箭頭
擦除
格式
撤銷
重做
清除
與會者 (15)
查找參會者
張齡月 (我)
婷婷
解除靜音
舉手
2 共享螢幕（指導演練；點評作品）
老師須先停止共享螢幕
才能請學生共享螢幕
1 聊天
解除靜音
啟動視訊
邀請
與會者
共享螢幕
聊天
錄影
4 解除靜音
3 與會者/舉手

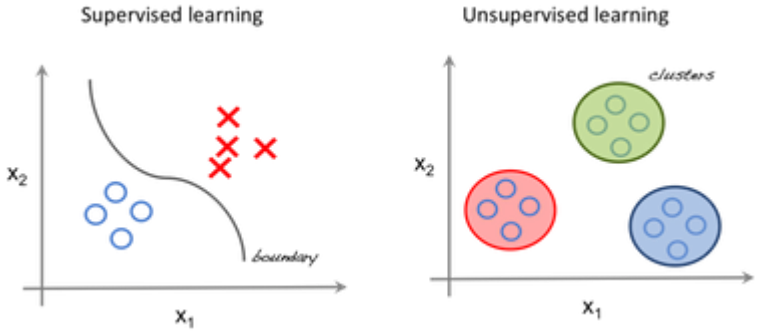
課程內容

整體學習方法

- K-Means 集群
- 分層集群
- 評估集群模型評估群集模型

監督與非監督式學習

- 在監督式學習中，系統試圖從先前給予的資料中學習。
- 在非監督式學習中，系統會嘗試直接從給予的資料中找到模式。
- 資料集：
  - 若被標記代表為監督式學習。
  - 若沒有被標記代表為非監督式學習。



K-means Clustering

- k-means clustering 是指將給定的資料分成 k 個聚群 (group)，每一群又稱為一個集群 (cluster)。
- 就是沒有訓練的過程，直接以資料進行處理。
- 演算法決定如何以給定的度量將資料集聚類於 k 個集群，也就是找出 k 個重心 (centroids) 以這些重心為集群的中心將資料收納。
- 度量一般可採用歐氏距離或距離平方，根據距離於多維的空間將資料點指配於最近的重心。
- 步驟

1. 事先指定 K值。
2. 隨機選 K 個點為重心，對每一個點算出到這 K 個重心的矩離，並依距離將所有點分配到最近的重心，以每一重心為中心形成集群，形成 K 個集群。
3. 再以每個集群的平均重心為新的重心重新計算每個點到每個新重心的距離，算過後再把點重新指配給最近的重心，形成新的 K 個集群。
4. 重覆上述動作，直至沒有任何點改變重心歸屬。

K-means Clustering 練習-初始化

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
df = pd.DataFrame({
    'x': [12, 20, 28, 18, 29, 33, 24, 45, 45, 52, 51, 52, 55, 53, 55, 61, 64, 69, 72],
    'y': [39, 36, 30, 52, 54, 46, 55, 59, 63, 70, 66, 63, 58, 23, 14, 8, 19, 7, 24]
})
np.random.seed(200)
k = 3
centroids = {
    i+1: [np.random.randint(0, 80), np.random.randint(0, 80)]
    for i in range(k)
}
fig = plt.figure(figsize=(5, 5))
plt.scatter(df['x'], df['y'], color='k')
colmap = {1: 'r', 2: 'g', 3: 'b'}
for i in centroids.keys():
    plt.scatter(*centroids[i], color=colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)
plt.show()
```

★操作檔案「K-means1.py 」

K-means Clustering 練習-開始作業

```
def assignment(df, centroids):
    for i in centroids.keys():
        # sqrt((x1 - x2)^2 - (y1 - y2)^2)
        df['distance_from_{}'.format(i)] = (
            np.sqrt(
                (df['x'] - centroids[i][0]) ** 2
                + (df['y'] - centroids[i][1]) ** 2
            )
        )
    centroid_distance_cols = ['distance_from_{}'.format(i) for i in centroids.keys()]
    df['closest'] = df.loc[:, centroid_distance_cols].idxmin(axis=1)
    df['closest'] = df['closest'].map(lambda x: int(x.lstrip('distance_from_')))
    df['color'] = df['closest'].map(lambda x: colmap[x])
    return df
df = assignment(df, centroids)
print(df.head())
fig = plt.figure(figsize=(5, 5))
plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i], color=colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)
plt.show()
```

★操作檔案「K-means1.py」

K-means Clustering 練習-更新作業

```
import copy
old_centroids = copy.deepcopy(centroids)
def update(k):
    for i in centroids.keys():
        centroids[i][0] = np.mean(df[df['closest'] == i]['x'])
        centroids[i][1] = np.mean(df[df['closest'] == i]['y'])
    return k
centroids = update(centroids)
fig = plt.figure(figsize=(5, 5))
ax = plt.axes()
plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i], color=colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)
for i in old_centroids.keys():
    old_x = old_centroids[i][0]
    old_y = old_centroids[i][1]
    dx = (centroids[i][0] - old_centroids[i][0]) * 0.75
    dy = (centroids[i][1] - old_centroids[i][1]) * 0.75
    ax.arrow(old_x, old_y, dx, dy, head_width=2, head_length=3, fc=colmap[i], ec=colmap[i])
plt.show()
```

★操作檔案「K-means1.py」

K-means Clustering 練習-重複分配

```
df = assignment(df, centroids)
fig = plt.figure(figsize=(5, 5))
plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i], color=colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)
plt.show()
```

★操作檔案「K-means1.py」

K-means Clustering 練習-直到最後作業

```
while True:
    closest_centroids = df['closest'].copy(deep=True)
    centroids = update(centroids)
    df = assignment(df, centroids)
    if closest_centroids.equals(df['closest']):
        break
fig = plt.figure(figsize=(5, 5))
plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i], color=colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)
plt.show()
```

★操作檔案「K-means1.py」

K-means Clustering 練習-改用 scikit-learn

```
df = pd.DataFrame({
    'x': [12, 20, 28, 18, 29, 33, 24, 45, 45, 52, 51, 52, 55, 53, 55, 61, 64, 69, 72],
    'y': [39, 36, 30, 52, 54, 46, 55, 59, 63, 70, 66, 63, 58, 23, 14, 8, 19, 7, 24]
})

from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3)
kmeans.fit(df)
```

★操作檔案「K-means1.py」

K-means Clustering 練習-改用 scikit-learn

```
df = pd.DataFrame({
    'x': [12, 20, 28, 18, 29, 33, 24, 45, 45, 52, 51, 52, 55, 53, 55, 61, 64, 69, 72],
    'y': [39, 36, 30, 52, 54, 46, 55, 59, 63, 70, 66, 63, 58, 23, 14, 8, 19, 7, 24]
})
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)
kmeans.fit(df)
labels = kmeans.predict(df)
centroids = kmeans.cluster_centers_
fig = plt.figure(figsize=(5, 5))
colors = map(lambda x: colmap[x+1], labels)
plt.scatter(df['x'], df['y'], color=colors, alpha=0.5, edgecolor='k')
for idx, centroid in enumerate(centroids):
    plt.scatter(*centroid, color=colmap[idx+1])
plt.xlim(0, 80)
plt.ylim(0, 80)
plt.show()
```

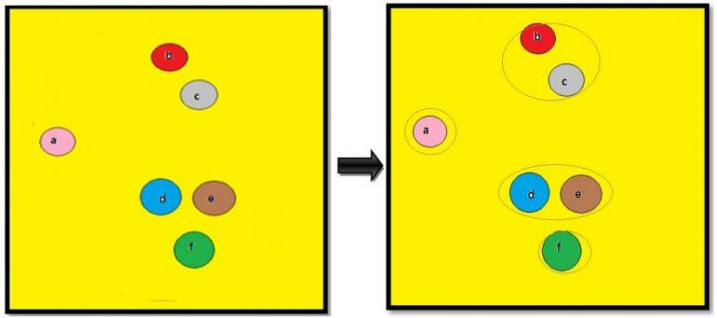
★操作檔案「K-means1.py」

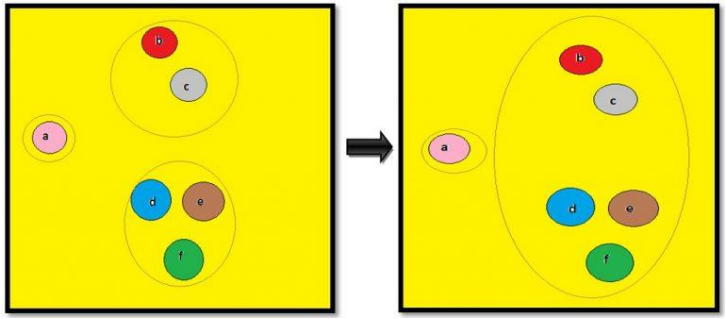
Hierarchical Clustering

- Hierarchical Clustering 分層集群對 Clustering 集群本身進行排序，共有兩種方法：
  - Agglomerative clustering 凝聚集群（由下而上方法）：
    - 將每個樣本視為單個 Clustering 集群，然後連續 merge 合併（或 agglomerate 聚集）Clustering 集群，直到所有 Clustering 集群已合併為單個 Clustering 集群。
  - Divisive clustering 分裂集群（自上而下）：
    - 將所有樣本的單個 Clustering 集群分成兩個最不相似的 Clustering 集群，直到每個 Clustering 集群有一個 Clustering 集群。
- 分裂集群比凝聚集群產生更準確的層次結構，但在概念上更複雜。

Hierarchical Clustering

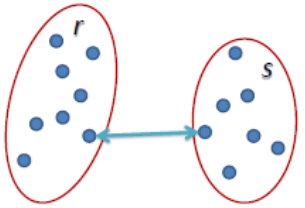
- 假設有6個樣本，您需要根據歐幾里德距離等一些潛在參數對它們進行 Clustering 集群分類。
- 點 c 和 b 可以分組為一個集群，點 d 和 e 有另一個集群。
- 還將 a 和 f 指向不同的單一集群。



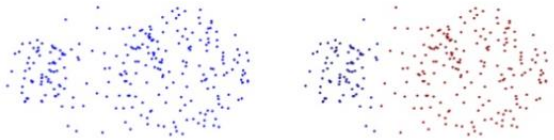


Linkage measures 聯動措施

- Linkage measures 聯動措施存在許多測量兩個集群之間距離的方法。
- Single link distance 單一連結距離
  - 單一連結距離定義為每個集群中兩點之間的最小距離。
  - 產生長而細長的集群。
  - 對異常值敏感。



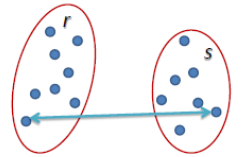
$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$



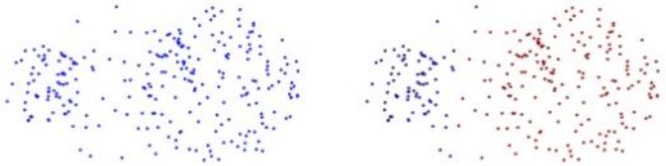
Original Points　　Two Clusters

Linkage measures 聯動措施

- Complete link distance 完整連結距離
  - 完整連結距離定義為每個集群中兩點之間的最大距離。
  - 產生更平衡的集群（具有相同的直徑）。
  - 不易受異常值影響。
  - 經常打破非常大的集群。
  - 小集群與大集群合併。



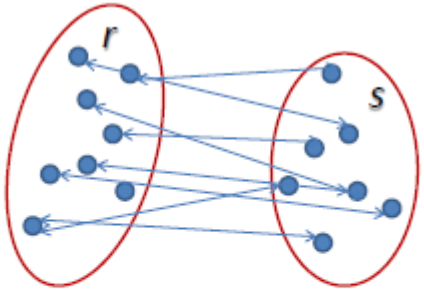
$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$



Original Points　　Two Clusters

Linkage measures 聯動措施

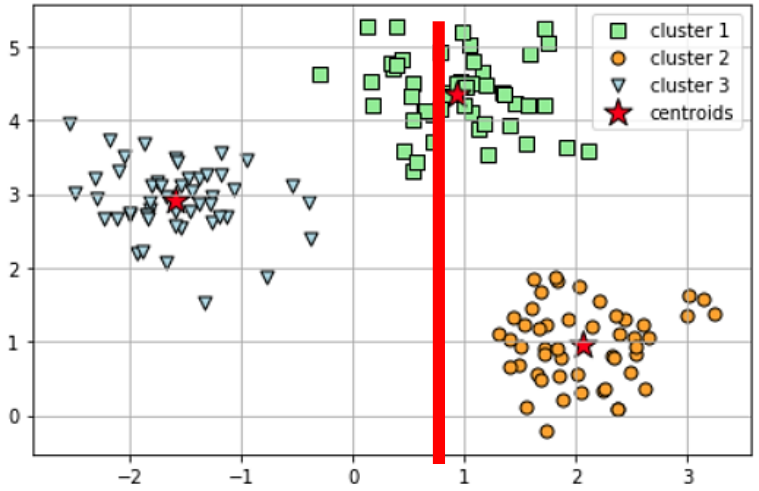
- Average link distance 平均連結距離
  - 一個群集中每個點與另一個群集中每個點之間的平均距離。
  - 不易受異常值的影響。
  - 偏向於球狀星團。

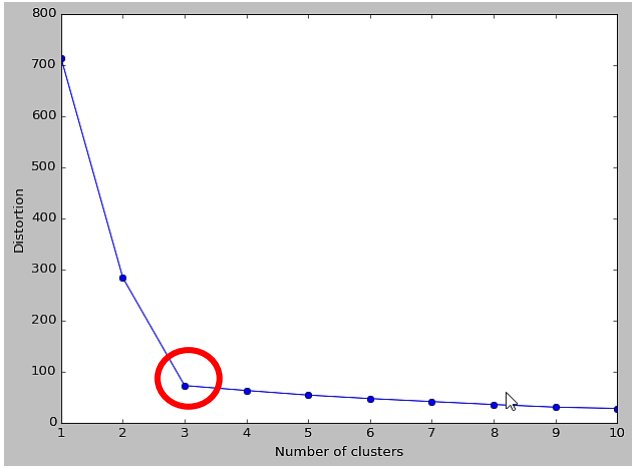


$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

決定集群數目 - Elbow

- 轉折判斷法（Elbow）：設定各種集群數目，執行 K-Means，計算集群內的『誤差平方和』（SSE），當作失真（Distortion）的程度
- 範例：kmeans_optimization_elbow.py





決定集群數目 - Silhouette

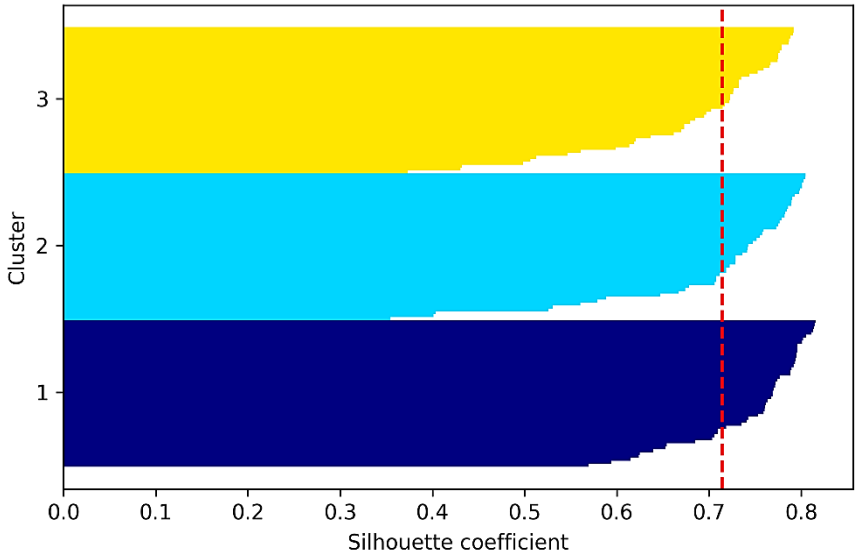
- 輪廓圖分析（Silhouette Analysis）：計算『輪廓係數』可檢驗樣本在集群中是否緊密在一起，步驟如下：

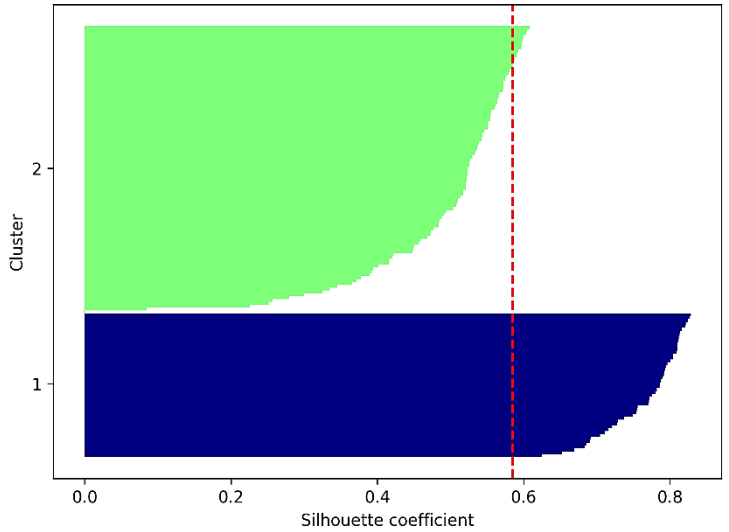
1. 對樣本點 $x^{(i)}$，計算其「集群內聚性」$a^{(i)}$，即該樣本與「同集群」其他樣本點之間的平均距離。
2. 對樣本點 $x^{(i)}$，計算其與最相近集群的「集群分離性」$b^{(i)}$，即該樣本與「最相近集群」中的所有樣本之間的平均距離。
3. 計算輪廓 $s^{(i)}$，即「集群分離性」與「集群內聚性」的差，除以兩者較大的值，公式如下：

$$s^{(i)} = \frac{b^{(i)} - a^{(i)}}{\max\left\{b^{(i)}, a^{(i)}\right\}}$$

決定集群數目 - Silhouette

- 範例：kmeans_optimization_silhouette.py
- 輪廓係數（silhouette Coefficient）介於 [-1, 1] 之間，愈接近1愈佳

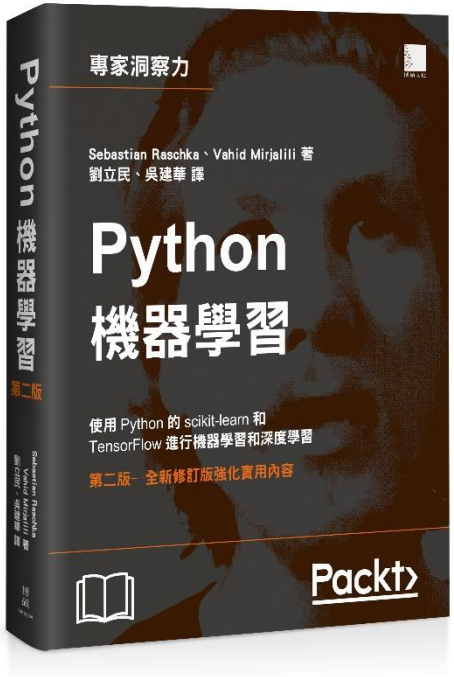




實作

- Lab (275x)
  - Module7-275 / IntroToUnsupervisedLearning.ipynb
  - Module7-275 / ApplicationOfClustering.ipynb

參考用書



- 書名：Python機器學習（第二版）

  http://www.drmaster.com.tw/bookinfo.asp?BookID=MP11804
- 作者：Sebastian Raschka, Vahid Mirjalili ISBN
- 譯者：劉立民、吳建華
- 出版社：博碩

問卷

http://www.pcschoolonline.com.tw

www.pcschoolonline.com.tw