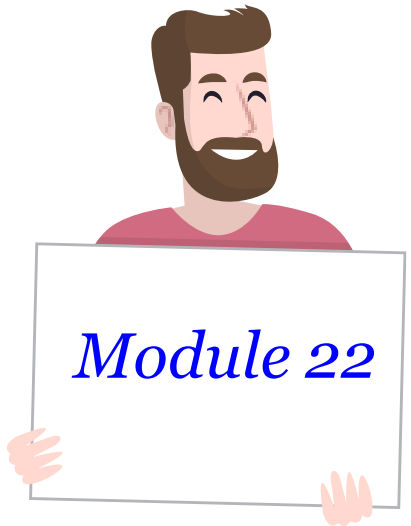
資料降維與視覺化



designed by freepik

Estimated time:
45 min.

學習目標

- 22-1: 降維與資料視覺化
- 22-2: PCA介紹
- 22-3: t-sne介紹

22-1:降維與資料視覺化

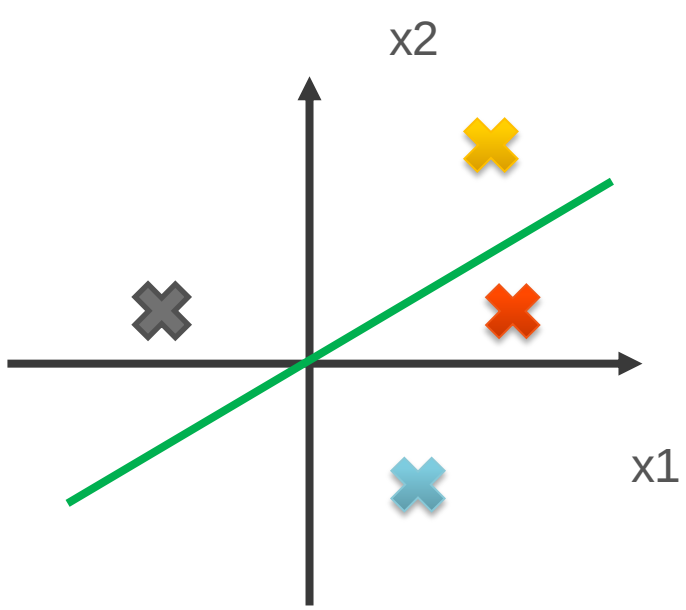
- 為什麼需要降維
- 什麼是降維

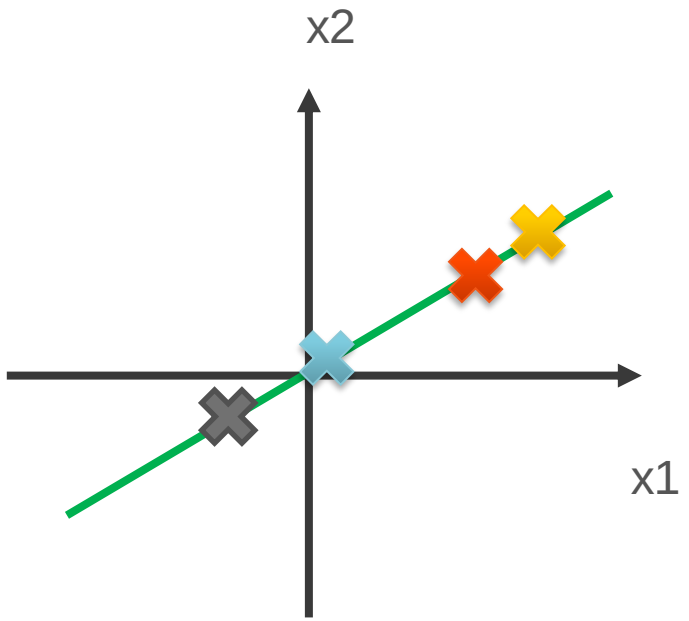
為什麼需要降維

- 降低特徵維度，增加訓練效率
- 降低特徵維度，減少儲存空間
- 易於可視化
  - 人類最多只能觀察到3D向量空間

什麼是降維

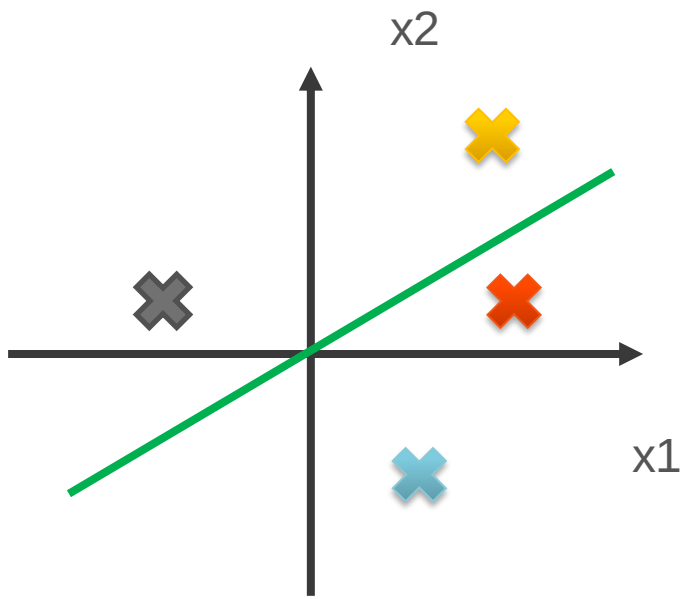
- 降維就是將原始資料用更少且不同的座標去表示原始的資料
  - 原本資料要用x1, x2表示，現在可以找到一個新的軸z，用來表示所有資料

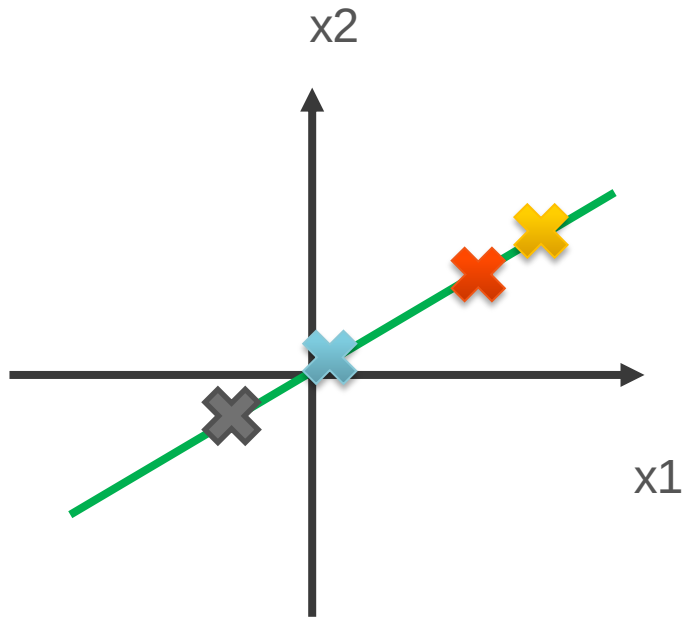




什麼是降維

- 雖然我們是用更少的座標軸來表示原始資料，但一個好的降維需要確保不同筆資料用新的座標軸表示的時候，不會被混在一起，無法區分出來



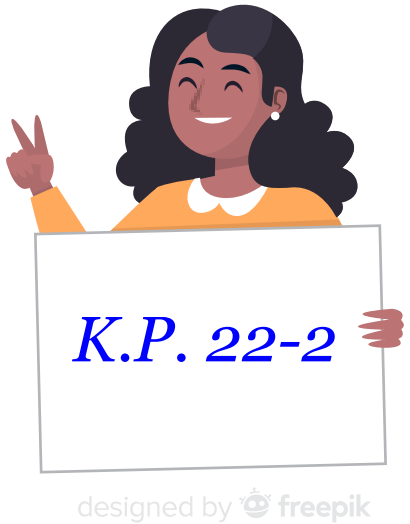


什麼是降維

- 一般來說，降維是一種非監督式學習
- 常見的降維方法有
  - SVD、PCA、t-sne、autoencoder等

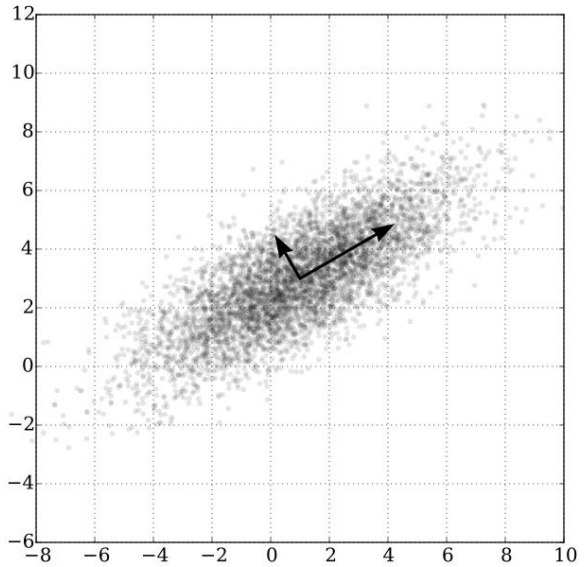
22-2: PCA介紹

- PCA是什麼
- PCA原理
- PCA數值範例



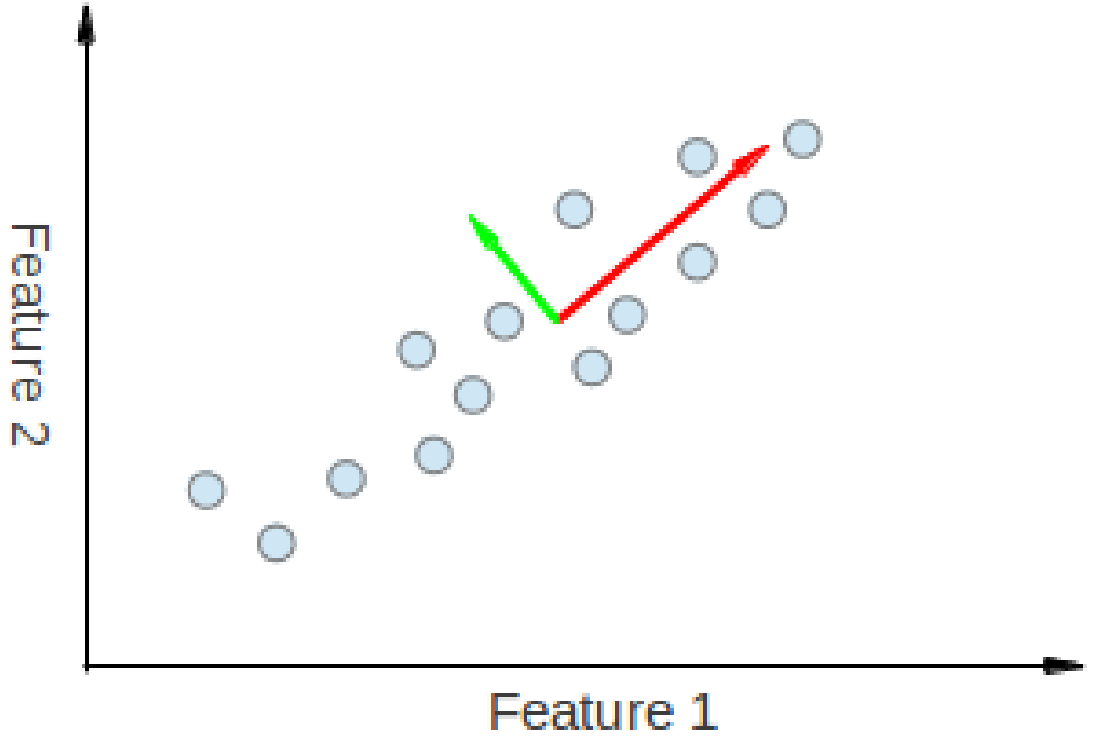
PCA是什麼

- **PCA(Principal component analysis)是一種降維的方法，它的核心思想是找到N個新的座標軸，使得所有資料投影在這些軸上後，離散程度最大**



PCA原理

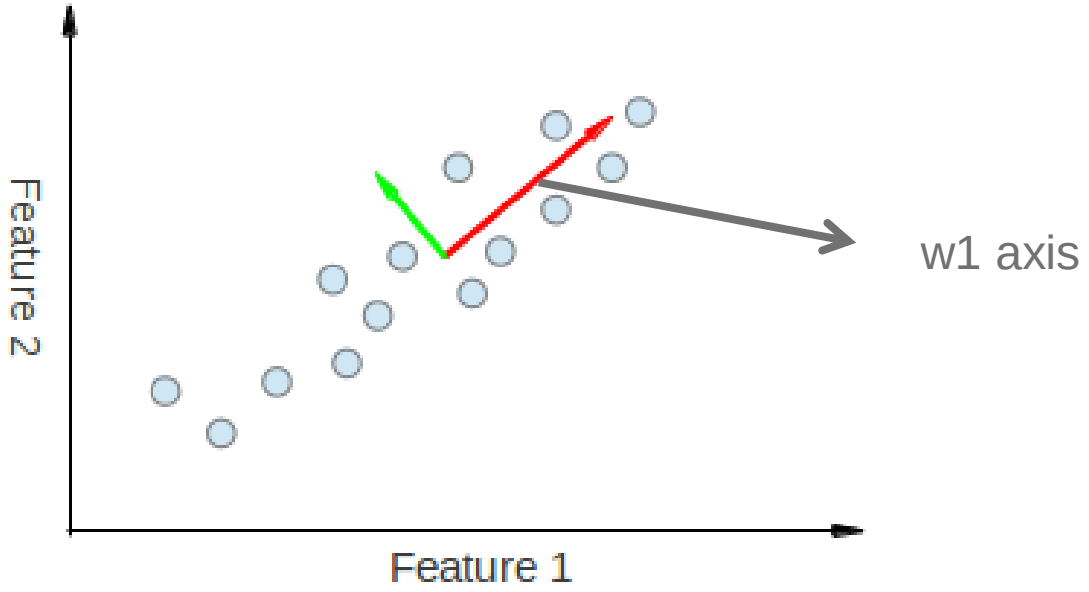
- 可以思考一下，將這些資料投影在哪個箭頭上比較好
  - 答案是紅色的箭頭，因為資料投影過後其座標值的離散程度比較高



PCA原理

- 根據上面所說的，PCA會先去找一個座標軸，這個座標軸的向量w1為單位向量，並確保所有投影在上面的資料標準差越大越好

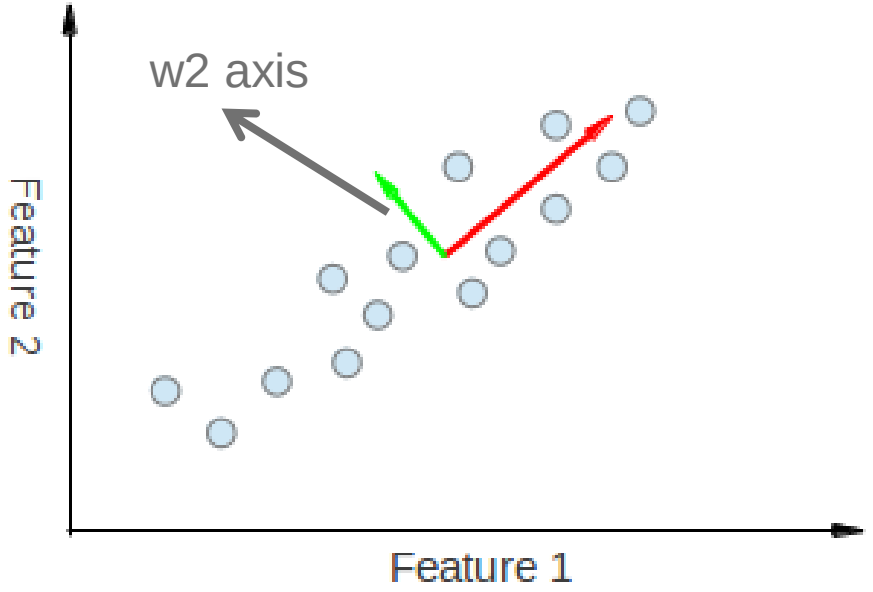
$var(z_1)$ 越大越好 且 $\|w^1\| = 1$



PCA原理

- 接下來PCA會再去找第二個軸去投影資料，一樣也會確保投影過後的資料標準差越大越好，但是為了確保不會跟上一次找到的軸一樣，所以會多一項與前幾個軸正交的限制式，確保找到不同軸

$var(z_2)$越大越好且$\|w^2\| = 1$，其中 $w^1 \cdot w^2 = 0$



PCA數值範例

- 假設我們的資料如下，在使用PCA時，我常常會先將資料標準化

Data =

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

PCA數值範例

- 計算標準化過後的共變異數矩陣

DataAdjust =

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

PCA數值範例

- 尋找此共變異矩陣的eigenvalues以及eigenvectors

$$cov = \begin{pmatrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{pmatrix}$$

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

PCA數值範例

- 將eigenvalues數值小的N個以及對應的eigenvectors刪除掉
  - eigenvalues越小代表其重要性越小，所以我們可以設定要刪除多少比較不重要的座標軸

$$eigenvalues = \begin{pmatrix} .0490833989 \\ 1.28402771 \end{pmatrix}$$

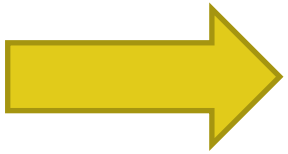
$$eigenvectors = \begin{pmatrix} -.735178656 & -.677873399 \\ .677873399 & -.735178656 \end{pmatrix}$$

PCA數值範例

- 使用留下來的座標軸投影原始資料
  - 我們可以發現資料從2D變成1D了

x	y
.69	.49
-1.31	-1.21
.39	.99
.09	.29
1.29	1.09
.49	.79
.19	-.31
-.81	-.81
-.31	-.31
-.71	-1.01

$$\begin{bmatrix} -0.6778 \\ -0.7351 \end{bmatrix}$$

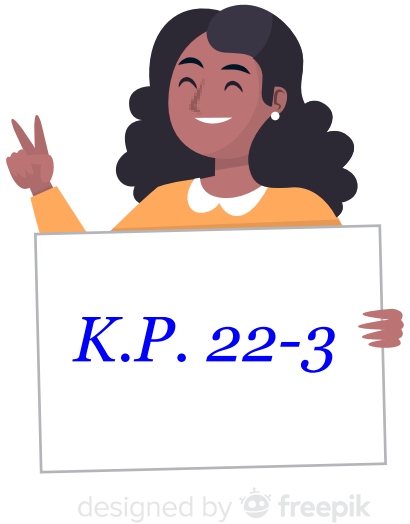


Transformed Data (Single eigenvector)

x
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056

22-3: t-sne介紹

- t-sne是什麼
- t-sne原理

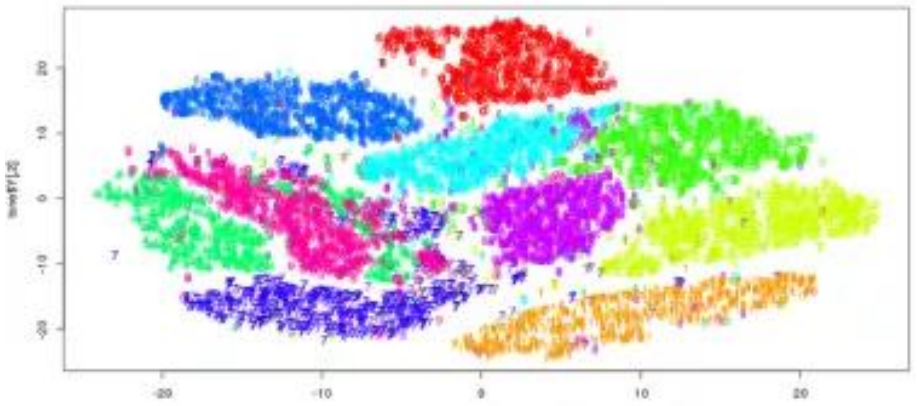


t-sne是什麼

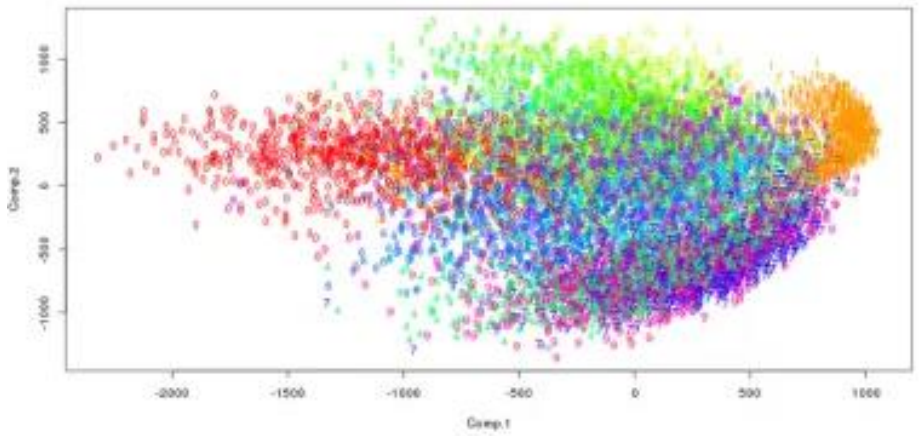
- t-sne是一種降維的演算法
  - 特別適合用來做可視化資料用
  - 它是一種非線性的降維演算法，所以所花的運算資源需要比較多

t-sne是什麼

- **t-sne與PCA最大的不同在於**
  - **t-sne在降維的時候，同一個類型的資料能在同一個區域且不同類型的資料彼此之間可以互相遠離**
  - **PCA只有確保同一個類型的資料能在同一個區域**



T-SNE



PCA

t-sne原理

- **t-sne的原理是將任兩筆資料丟入一個相似函數S去做計算，並將結果除以任兩筆資料相似函數的總和**
  - **在高維度空間會定義一個相似函數S，低維度的空間會定義另一個相似函數S’**
  - **這個值會介於0~1之間，可以把他們視為機率**

X(高維度空間) → Z(2D/3D空間)

$$P\left(x^j \middle| x^i\right) = \frac{S(x^i, x^j)}{\sum_{k \neq i} S(x^i, x^k)}$$

$$Q\left(z^j \middle| z^i\right) = \frac{S'(z^i, z^j)}{\sum_{k \neq i} S'(z^i, z^k)}$$

t-sne原理

- 在高維度空間的相似函數S以及低維度空間的相似函數S’函數如下

t-SNE

$$S(x^i, x^j) = e^{-\|x^i - x^j\|}$$

$$S'(z^i, z^j) = \frac{1}{1 + \|z^i - z^j\|}$$

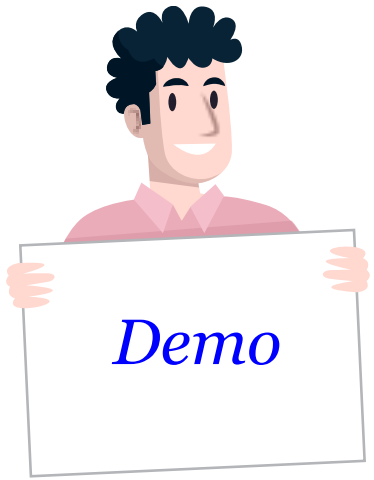
t-sne原理

- 我們將$P\left(x^j \middle| x^i\right)$以及$Q\left(z^j \middle| z^i\right)$丟入KL函數裡面
  - 並使用優化器去優化
  - 注意z^i是未知數，所以我們剛開始可以隨機初始化它

$$loss = \sum_i KL(P_i || Q_i) = \sum_{i \neq j} = P\left(x^j \middle| x^i\right) \log\left(\frac{P\left(x^j \middle| x^i\right)}{Q\left(x^j \middle| x^i\right)}\right)$$

Demo 22-3

- PCA將IRIS資料集降維到2維及3維
- PCA將MNIST資料集可視化
- t-sne將MNIST資料集可視化



線上Corelab

- 題目1：使用PCA將IRIS資料集可視化
  - 使用PCA演算法並完成程式碼，將IRIS資料可視化
- 題目2：使用PCA將MNIST資料集可視化
  - 使用PCA演算法並完成程式碼，將MNIST資料可視化
- 題目3：使用t-sne將MNIST資料集可視化
  - 使用t-sne演算法並完成程式碼，將MNIST資料可視化

本章重點精華回顧

- 降維與資料視覺化
- PCA原理
- t-sne原理

Lab:降維與資料可視化

- **Lab01: PCA將IRIS資料集降維到2維及3維**
- **Lab02: PCA將MNIST資料集可視化**
- **Lab03: t-sne將MNIST資料集可視化**

Estimated time:
20 minutes


THANK YOU