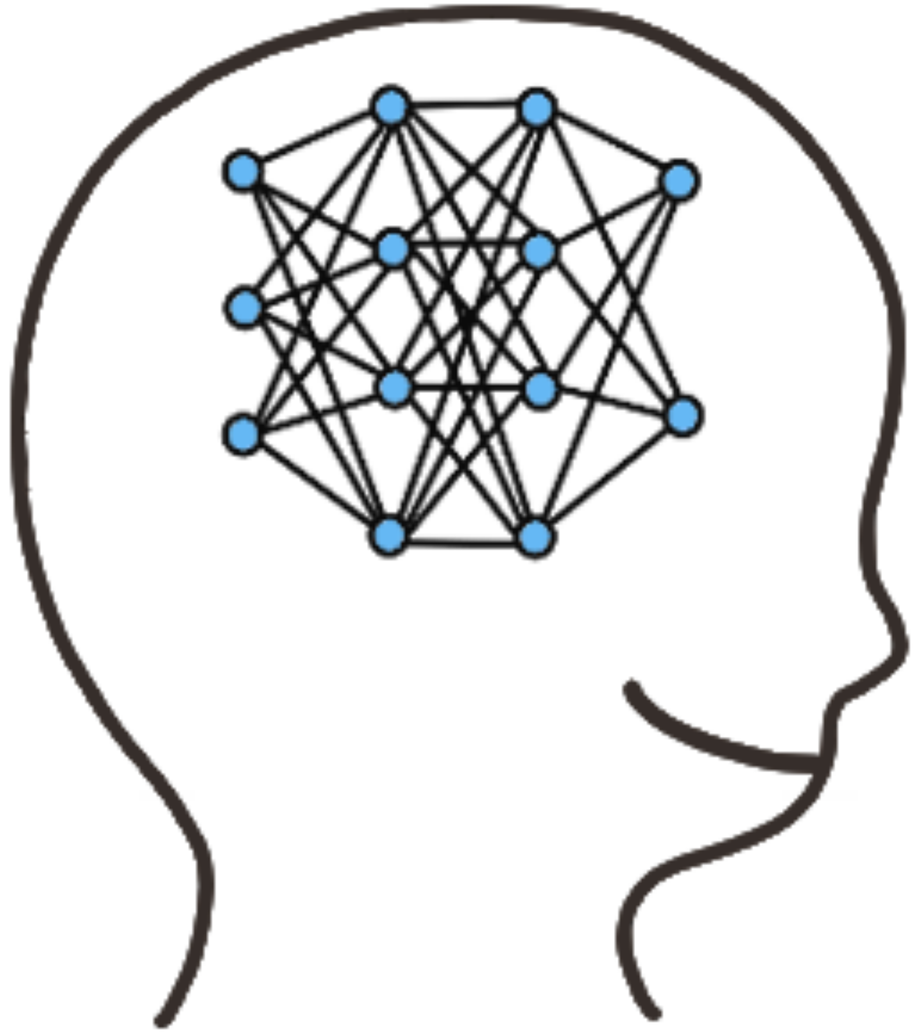
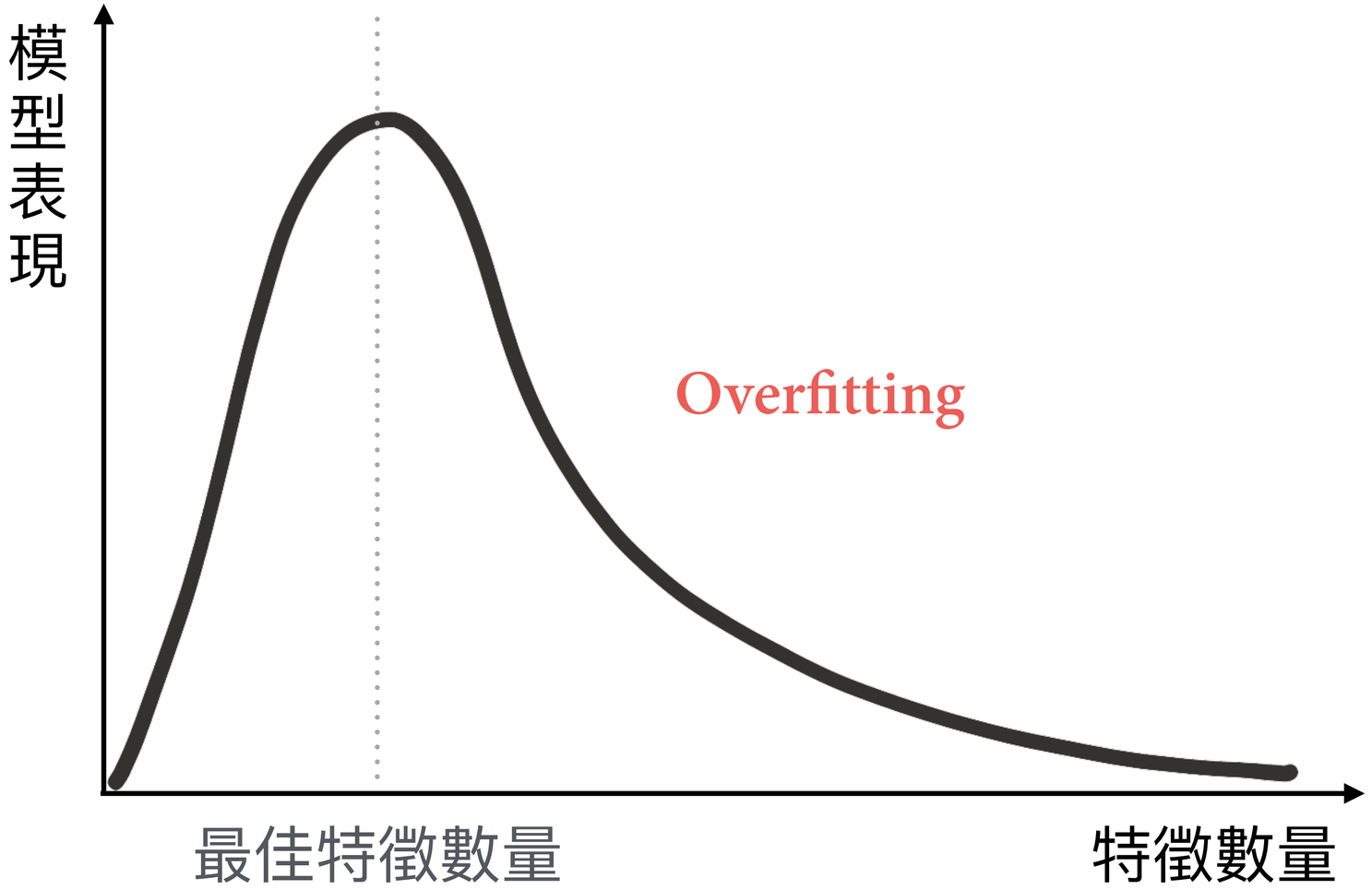


維度災難

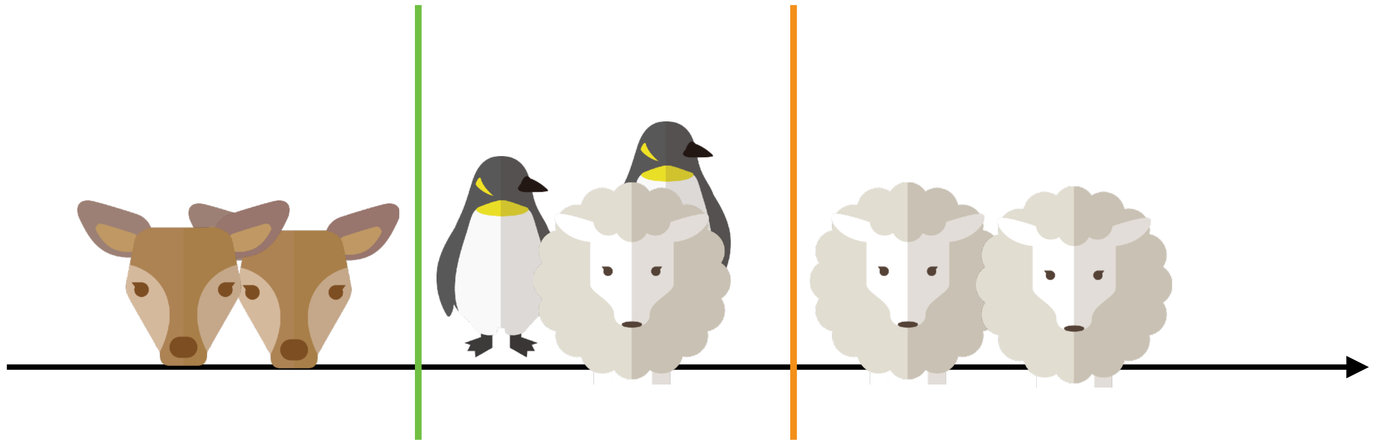
Curse of Dimensionality

維度災難 (Curse of Dimensionality)

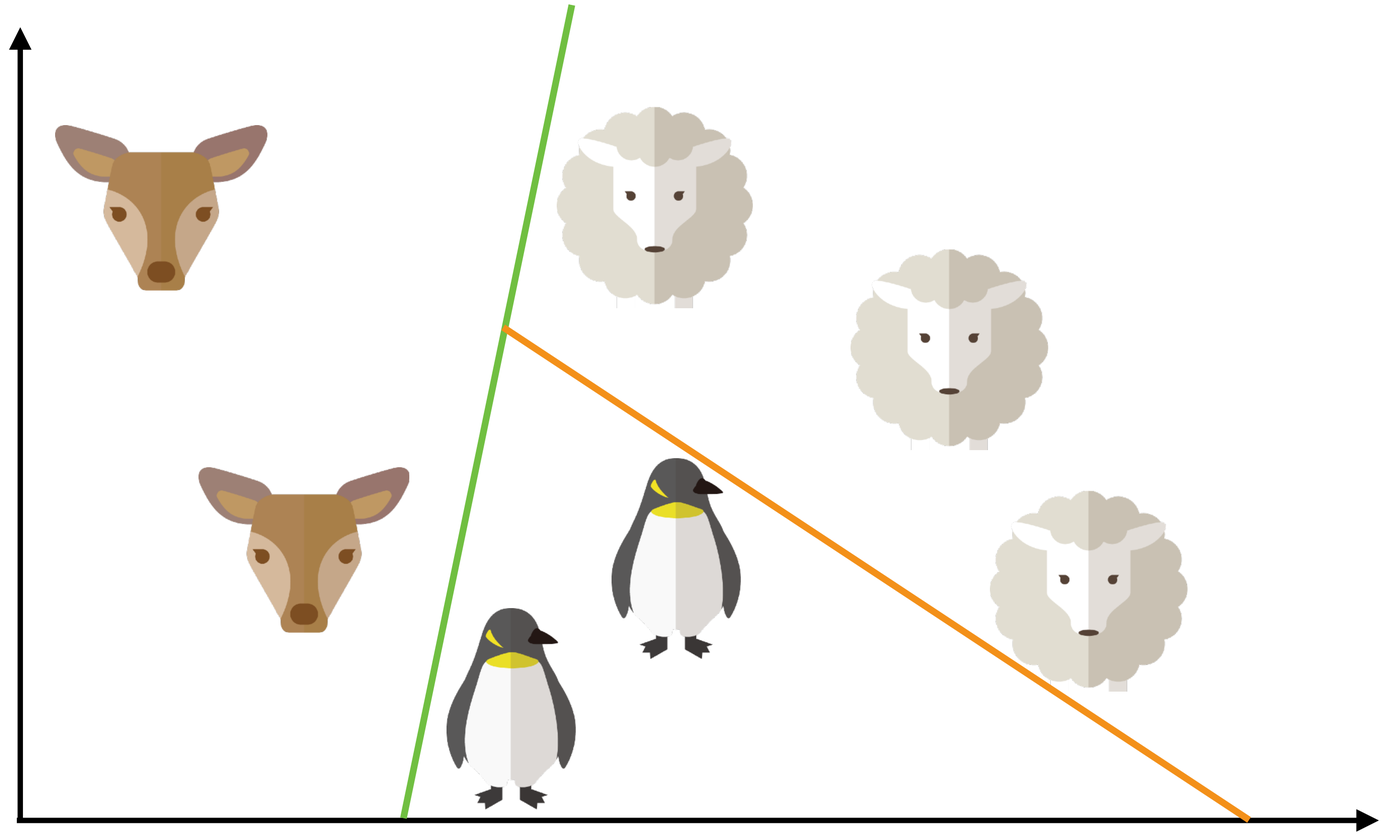


1維

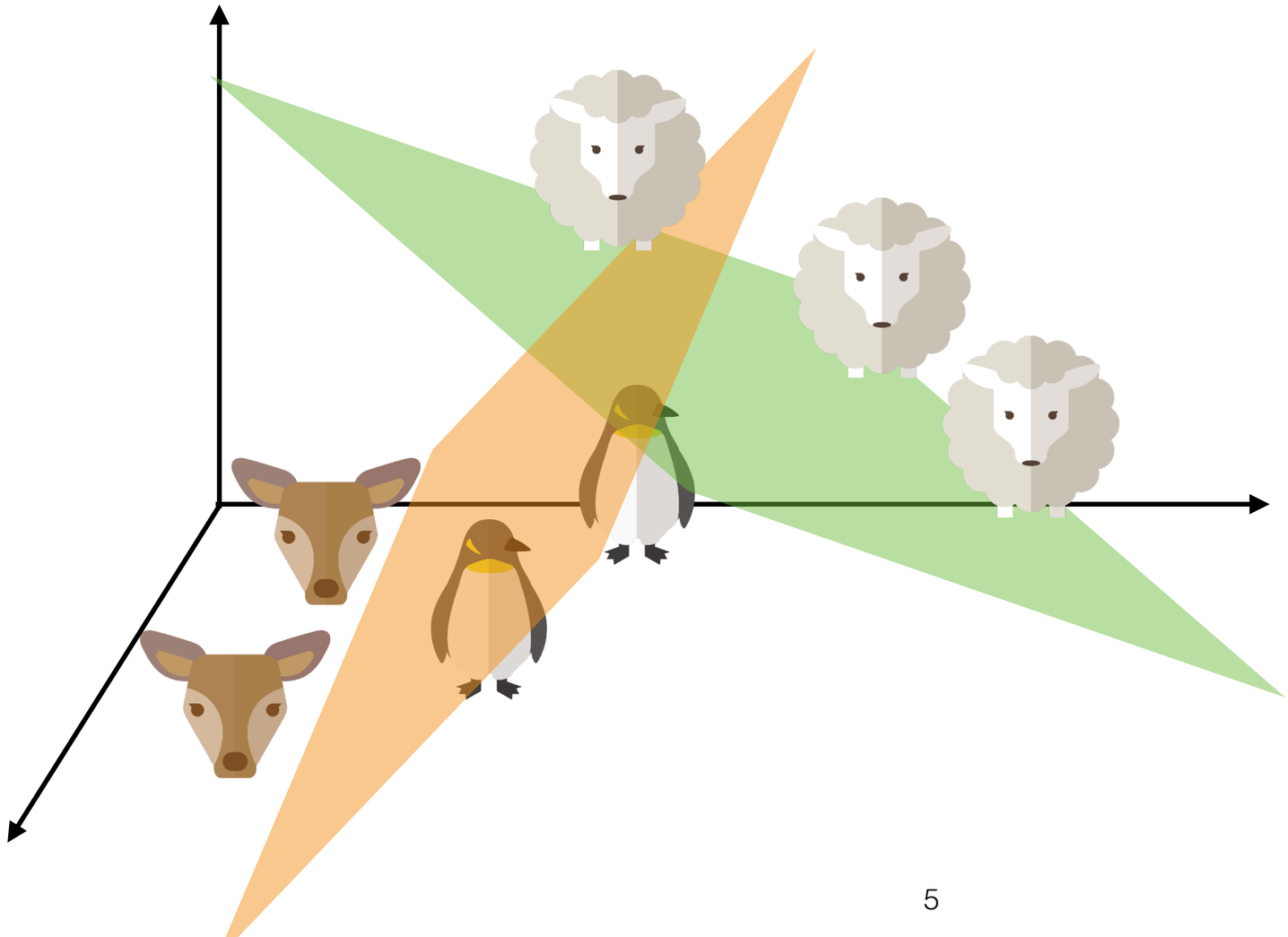
決策邊界



2維



3維



>3維

- 無法作圖
- 維度越來越高
  - 資料距離：越來越遠
  - 決策邊界：越來越有效分割
  - 模型複雜度：越來越高
  - 模型準確度：訓練準確度越來越高、測試準確度越來越低
    - 過擬合(overfitting)

How to solve it?

- 增加資料量
- 正規化 (Regularization)
- 降維技術/特徵提取(Feature Extraction)
  - e.g. 主成分分析(Principal Components Analysis, PCA)
- 深度學習
  - Good performance but slow and need more data

補充說明

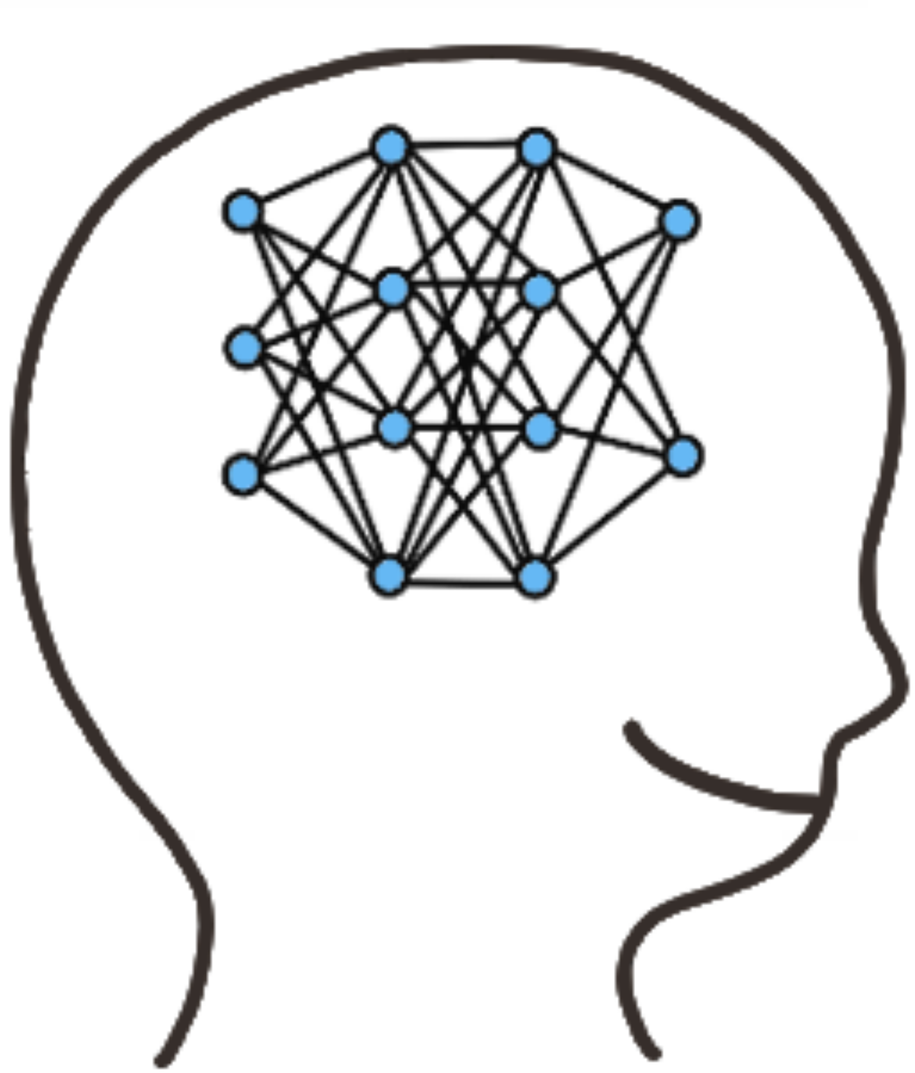
- 雖然PCA和正規化都有降低維度災難/overfitting的效果，但使用正規化效果通常較好，因為正規化同時把y考慮進去，而PCA只有針對X做降維，PCA更好的應用時機是：

  (1) 資料壓縮、加速建模

  (2) 降維後可以視覺化原本高維的資料

  (3) 特徵提取、去除雜訊 (選擇前k高的主成分)

- 所以如果遇到overfitting，還是請同學直接用正規化來解即可。



主成分分析

Principal Components Analysis

主成分分析 (PCA)

- 特徵降維/提取(Extraction)/壓縮
- 資料中有許多彼此相關的特徵

Notes

- Python 中 xe-n 指的是$x * 10^{-n}$
  - e.g. $5e\text{-}3 = 5 * 10^{-3} = 0.005$

	土地移轉總面積平方公尺	建物移轉總面積平方公尺	建物現況格局-房	建物現況格局-廳	車位移轉總面積平方公尺
土地移轉總面積平方公尺	1	0.678	0.34	0.22	0.39
建物移轉總面積平方公尺	0.678	1	0.197	0.06	0.736
建物現況格局-房	0.34	0.197	1	0.675	0.032
建物現況格局-廳	0.22	0.06	0.675	1	-0.043
車位移轉總面積平方公尺	0.39	0.736	0.032	-0.043	1

↓ PCA(n=3)

	0	1	2
0	1.000000e+00	7.869562e-16	1.817497e-15
1	7.869562e-16	1.000000e+00	5.334096e-16
2	1.817497e-15	5.334096e-16	1.000000e+00

PCA algorithm

- n維降到p維 (m(列/資料筆數)*n(行/特徵數量))

1. 標準化 features
2. 求共變異數矩陣 (Covariance Matrix) (n*n)
3. 矩陣分解成特徵向量(eigenvector)和特徵值(eigenvalue)
4. 選取最大的p個eigenvalues和對應的eigenvectors (最多有n個eigenvalues: n*1)
5. 合併p個eigenvectors成為「投影矩陣」(W) (W: n*p)
6. X.dot(W) 即為新的X'矩陣 (m*p)

PCA algorithm (cont.)

- Covariance Matrix

期望值　　x平均

$$Cov(X,Y) = E((X-\mu_x)(Y-\mu_y))$$

$$= E(XY - X\mu_y - Y\mu_x + \mu_x\mu_y)$$

$$= E(XY) - \mu_y E(X) - \mu_x E(Y) + \mu_x\mu_y$$

$$= E(XY) - \mu_y\mu_x - \mu_x\mu_y + \mu_x\mu_y$$

$$= E(XY) - \mu_x\mu_y$$

$$\rho(X,Y) = \frac{Con(X,Y)}{\sigma_x\sigma_y}$$

相關係數

x,y 標準差

- 共變異數、相關係數越大，線性相關性越高

PCA algorithm (cont.)

- 矩陣分解: Decompose Covariance Matrix into 特徵向量(eigenvector) and 特徵值(eigenvalue)

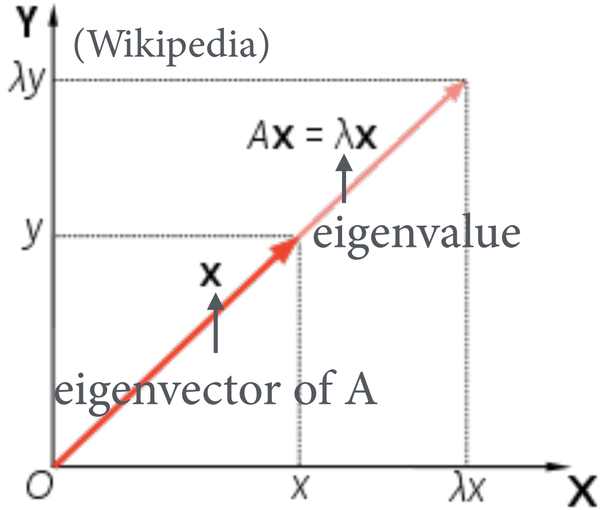
$$\begin{pmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix}$$

$$Av = w = \lambda v$$

eigenvector of A　　eigenvalue

$$\begin{pmatrix} 3 & 6 & 7 \\ 3 & 3 & 7 \\ 5 & 6 & 5 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ 4 \\ -2 \end{pmatrix} = -2 \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix}$$

eigenvector of A　　eigenvalue



PCA algorithm (cont.)

$A = \begin{pmatrix} 1 & 4 \\ 1 & -2 \end{pmatrix}$ 的eigenvector和eigenvalue?

行列式

$$\det(A - \lambda I) = \begin{vmatrix} 1-\lambda & 4 \\ 1 & -2-\lambda \end{vmatrix} = 0$$

$$(1-\lambda)(-2-\lambda) - 4$$

$$= \lambda^2 + \lambda - 6$$

$$= (\lambda - 2)(\lambda + 3) = 0$$

$\lambda = 2$

$\lambda = -3$

eigenvalues

$$A = \begin{pmatrix} 1 & 4 \\ 1 & -2 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = 2 \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

$$v_1 + 4v_2 = 2v_1$$

$$v_1 - 2v_2 = 2v_2$$

$$v = \begin{pmatrix} 4 \\ 1 \end{pmatrix}$$

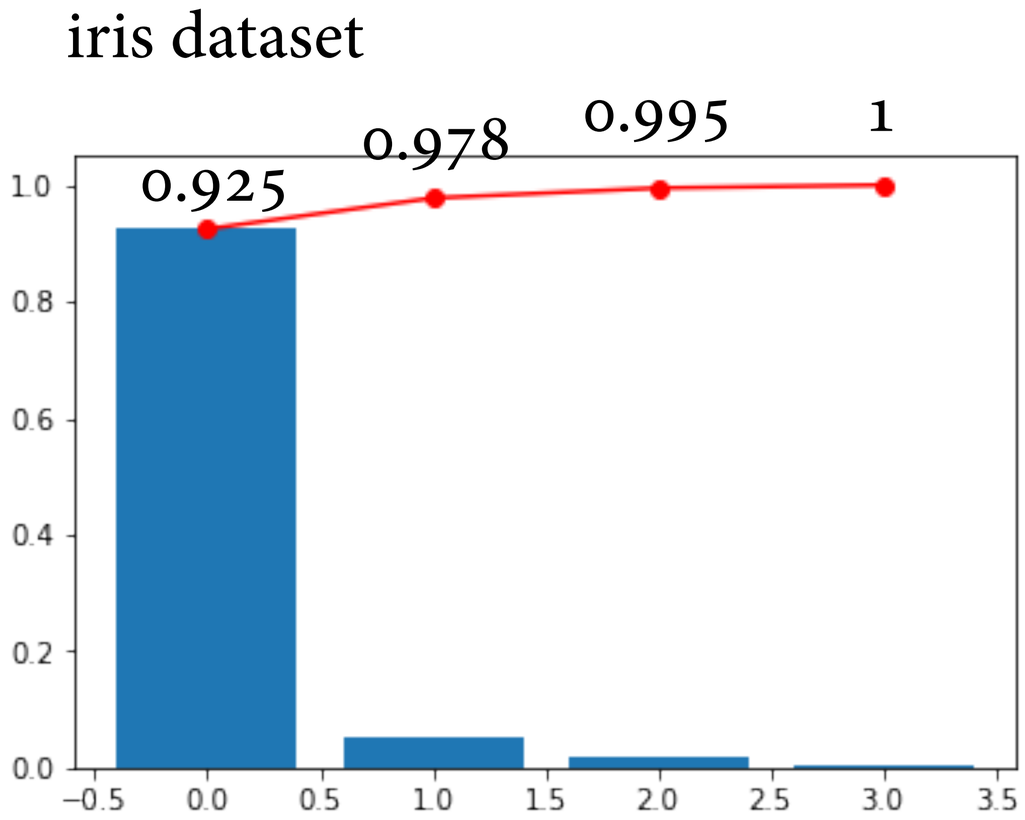
eigenvector
when eigenvalue=2

PCA algorithm

- n維降到p維 (m(列/資料筆數)*n(行/特徵數量))

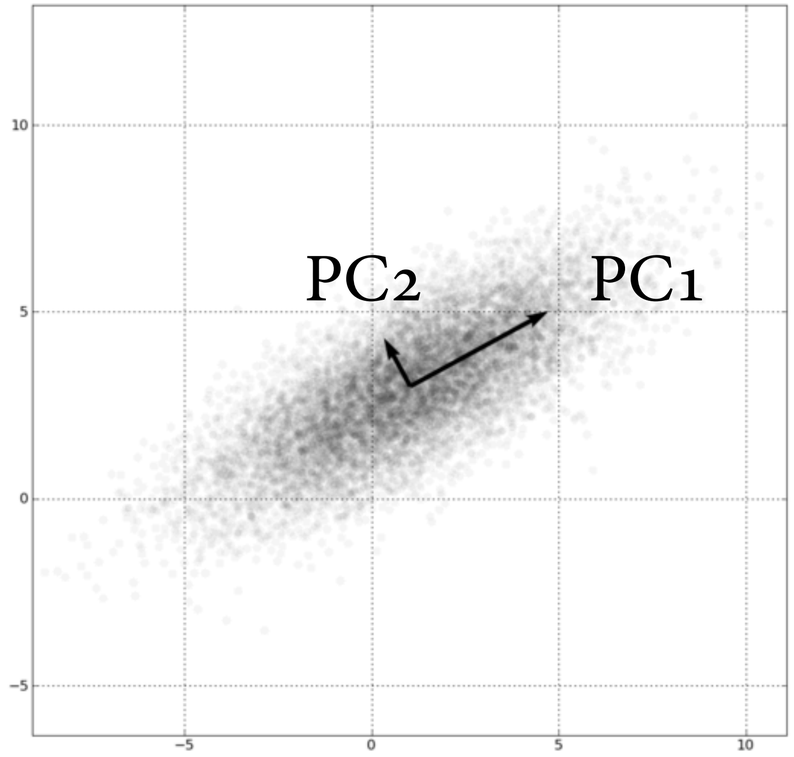
1. 標準化 features
2. 求共變異數矩陣 (Covariance Matrix) (n*n)
3. 矩陣分解成特徵向量(eigenvector)和特徵值(eigenvalue)
4. 選取最大的p個eigenvalues和對應的eigenvectors (最多有n個eigenvalues: n*1)
5. 合併p個eigenvectors成為「投影矩陣」(W) (W: n*p)
6. X.dot(W) 即為新的X'矩陣 (m*p)

解釋變異數比率

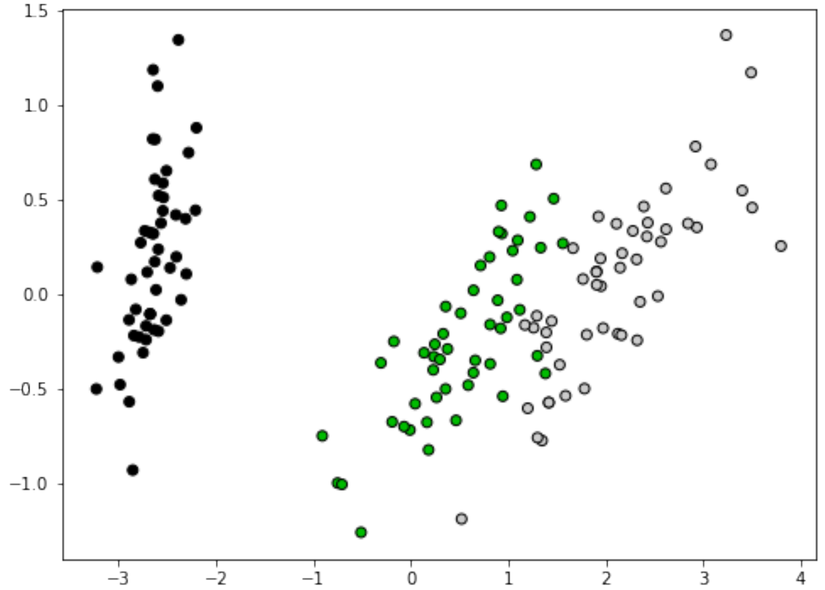


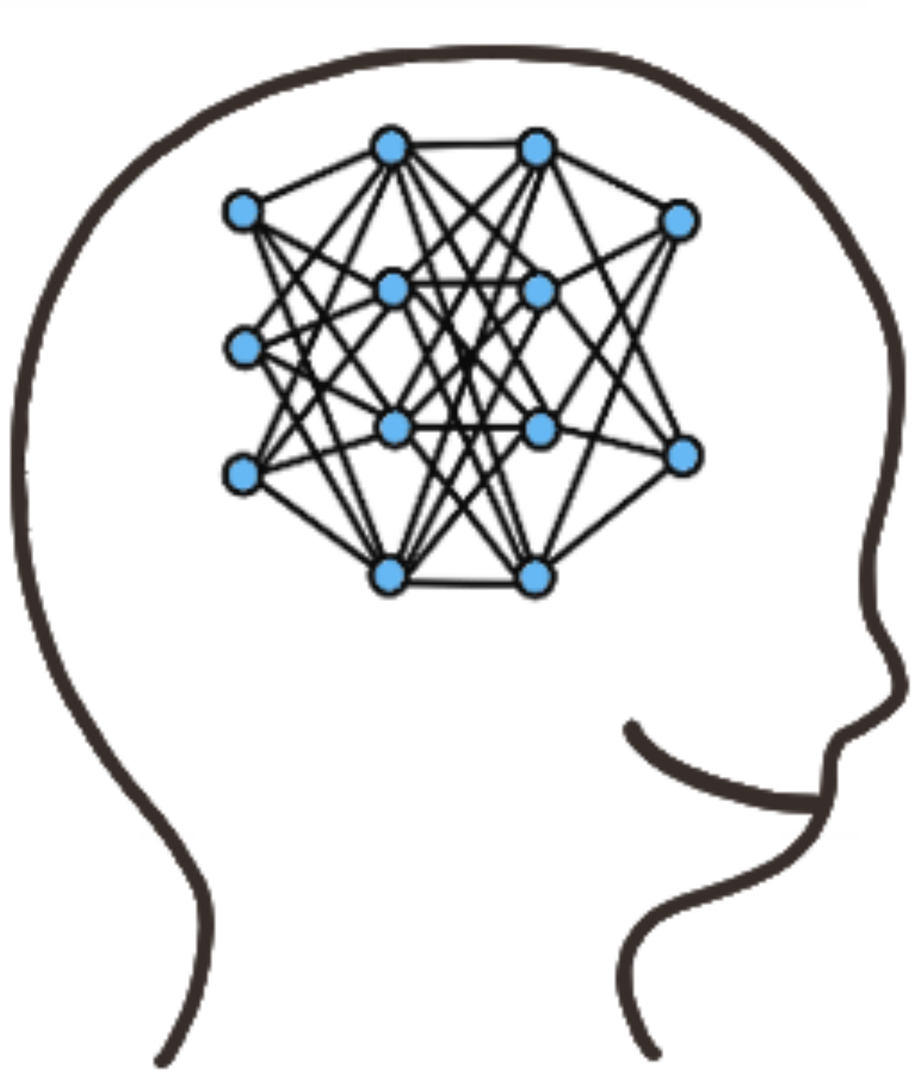
- 解釋變異數比率=特徵值(eigenvalue)/特徵值總和
- 主成份1占92.5%的變異數

PCA examples



iris: n=4 to 2





資料預處理(二)

Data Preprocessing- Part2

遺失值(missing data)處理

- 丟棄含遺失值的列:
  - `DataFrame.dropna`(axis=0, how='any', thresh=None, subset=None, inplace=False)
    - axis: 要丟棄的對象 {0 or 'index', 1 or 'columns'}
    - how: {'any'(任一na即丟棄), 'all'(所有na才丟棄)}
    - thresh: int, 超過(大於)多少個na才丟棄
    - subset: 要納入考慮的index/columns list (若丟棄index, 則考慮columns list)

遺失值(missing data)處理 (cont.)

- 補值: DataFrame.fillna()
  - `DataFrame.fillna`(value=None, method=None, axis=None, inplace=False, limit=None, downcast=None, **kwargs)
    - axis: 要丟棄的對象 {0 or 'index', 1 or 'columns'}