From Principal Subspaces to Principal Components with Linear Autoencoders

翁婉庭

PCA
(Principal Component Analysis)

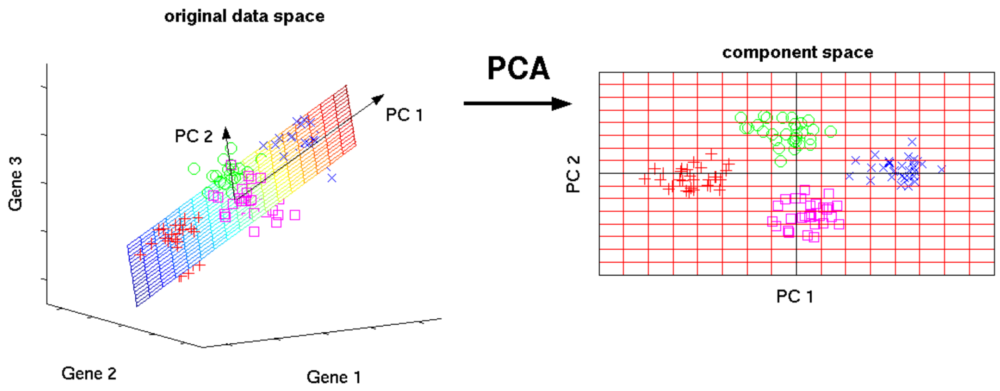
- Linear transformation:

  a set of observations ⟶ coordinate system

- Unsupervised
- Dimensionality reduction:

  n dimentions ⟶ the first m principal components

  (m<n)



PCA
(Principal Component Analysis)

$$\boxed{X = W^T Y}$$

Y：觀測向量(input vector)

W ：orthogonal matrix $(w_i \perp w_j，i \neq j)$

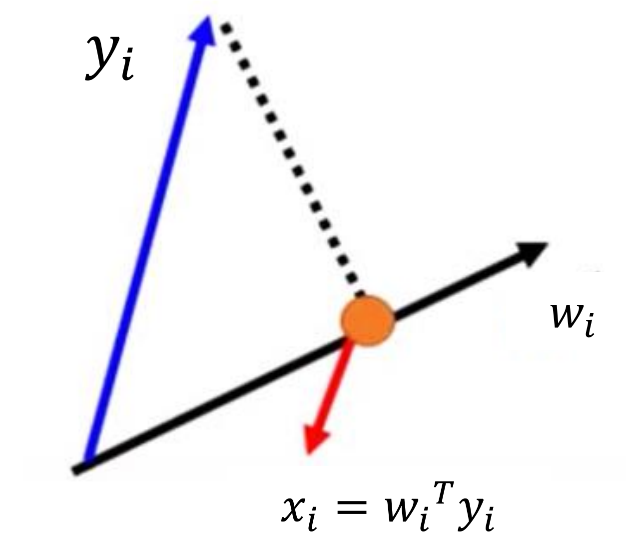
X ：投影出的純量

N ：觀測向量個數

n ：每個觀測向量的維度

m ：預降維到m個向量投影

w_i: *loading vector*



PCA

(Principal Component Analysis)

- 目標:

找一個W_i向量組成的W，經過投影後，使$Var(X)$愈大愈好，其中$W_i \perp W_j$，$i \neq j$

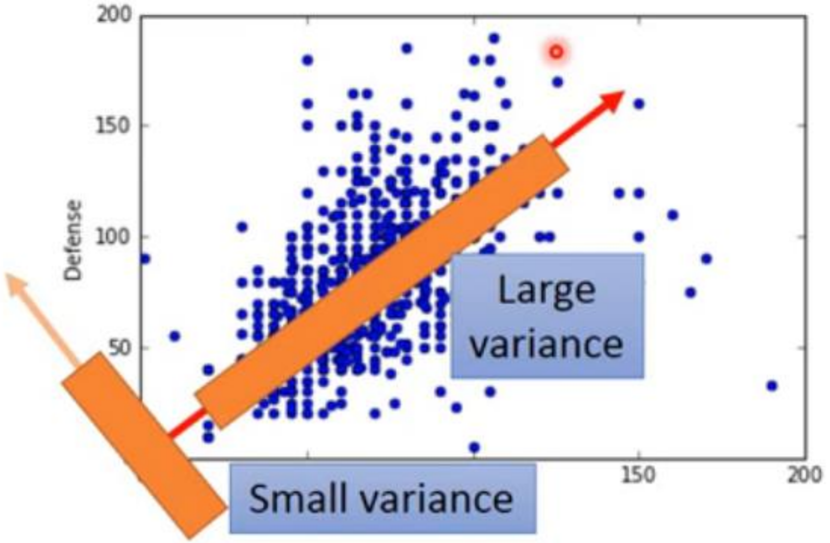
$$Var(x_1) = \sum(x_{1i} - \overline{x_1})^2 = \cdots = w_1^T Y_0 Y_0^T w_1 = w_1^T \text{cov(Y)} w_1$$

Dimensionality reduction:(n⟶m)

$\because$ *losing information*

$\therefore$ *minimize the loss of information = maxmux variance*

$$\Rightarrow \max_{w_1} Var(x_1) = \max_{w_1} w_1^T \text{cov(Y)} w_1 \quad \text{, s.t. } w_1^T w_1 = 1.$$



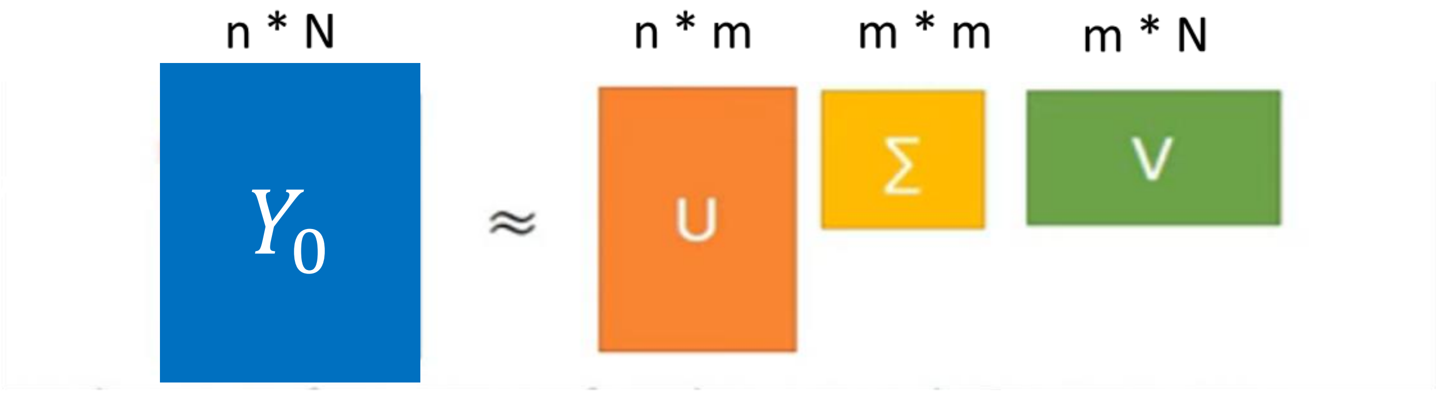
w_1 is eigenvector of cov(Y) ,corresponding to the largest eigenvalue

SVD

(Singular-value decomposition)

- 找$Y_0Y_0^T$ 的eigenvectors:$w_1, w_2, w_3 .... w_m$

  ∵high dimensional data 不好做 ⇒做SVD分解



$$Y_0Y_0^T = U\Sigma V^T V\Sigma^T U^T = U\Sigma\Sigma^T U^{-1} \quad = U\Sigma^2 U^{-1}$$

∴ m columns of U =the loading vectors of Y :(w_1,w_2, $w_3 .... w_m$)

Autoencorder

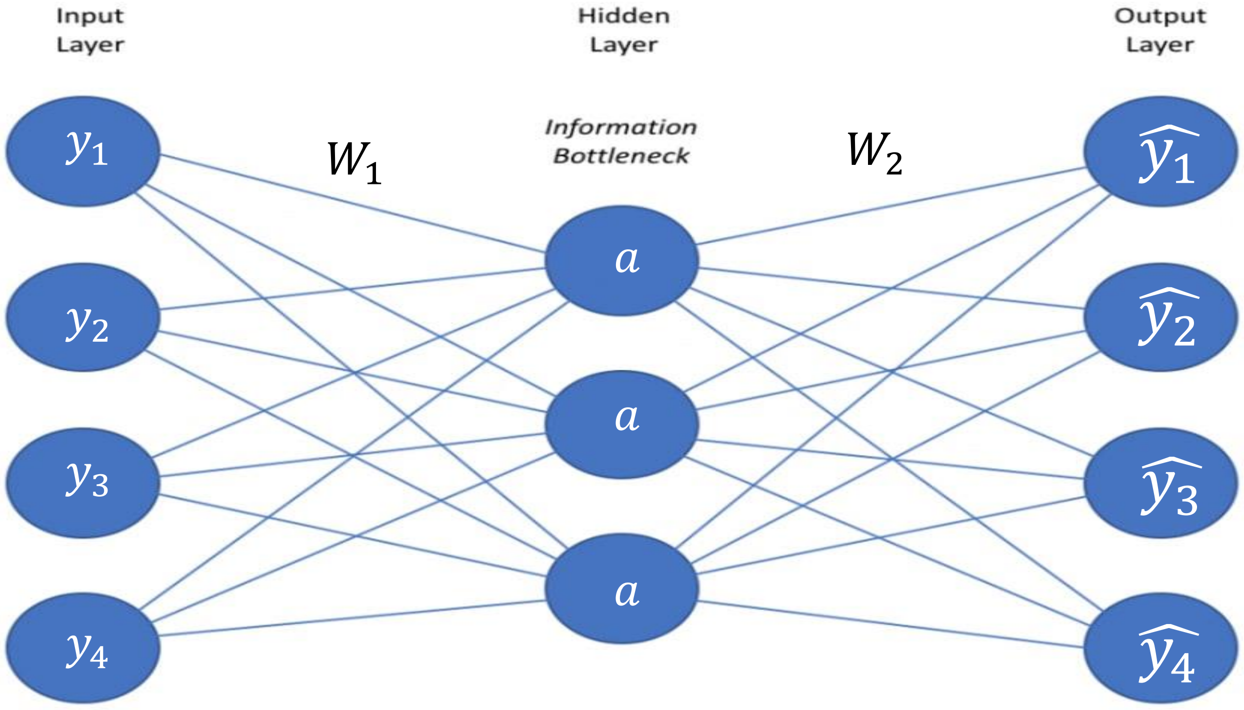
- unsupervised learning
- A single fully-connected hidden layer
- numbers of input layer = numbers of output layer
- Dimensionality reducation

- First layer

$x_i = a(W_1 y_i + b_1)$, a = activation function

- second layer

$\hat{y_i} = a(W_2 x_i + b_2) = a(W_2\, a(W_1 y_i + b_1) + b_2)$



Linear autoencoders

- 把activation function 改成線性
- Project data onto the principal subspace
- The optimization algorithm used to train the neural network
- No orthonormality constraint

- find W_1 , b_1, W_2 , b_2 by

  minimum the total squared difference between output and input

$$\min_{\mathrm{W}_1,\mathbf{b}_1,\mathrm{W}_2,\mathbf{b}_2} \left\|\mathrm{Y} - \left(\mathrm{W}_2\left(\mathrm{W}_1\mathrm{Y} + \mathbf{b}_1\mathbb{1}_N^T\right) + \mathbf{b}_2\mathbb{1}_N^T\right)\right\|_F^2.$$

Linear autoencoder ⟶ PCA

Advantage:

*Process high-dimensional data

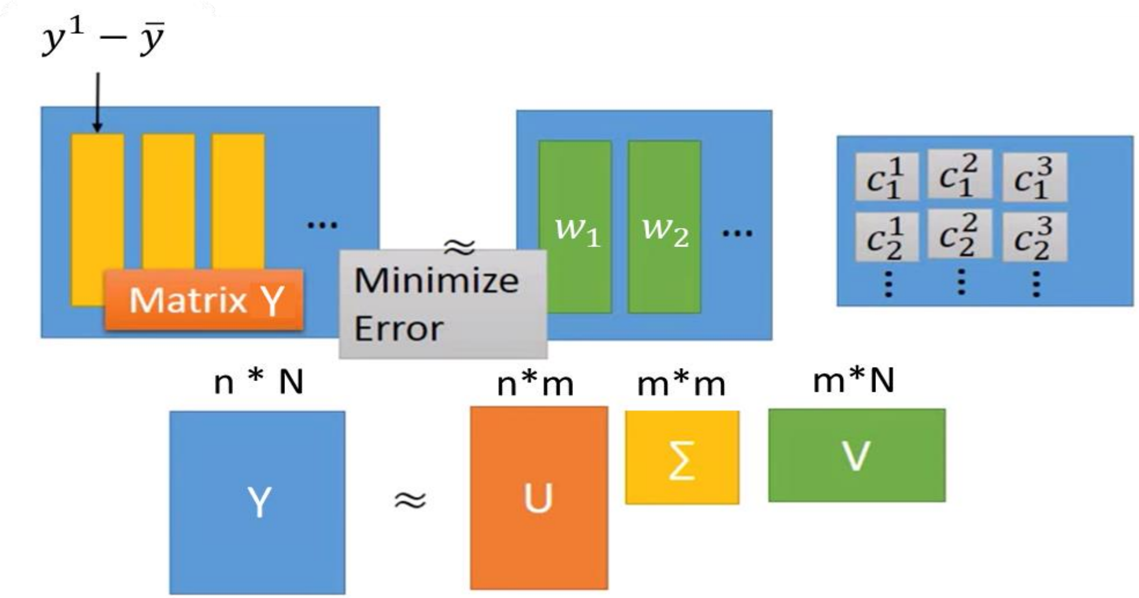
*Process datasets with large numbers of observations

*avoid the need to center the data to element-wise zero mean

- $y \approx c^1 w_1 + c^2 w_2 + \cdots + c^m w_m + \bar{y}$
- $y - \bar{y} \approx c^1 w_1 + c^2 w_2 + \cdots + c^m w_m = \hat{y}$

⇒*reconstruction error* = $\|(y - \bar{y}) - \hat{y}\|_2$

⇒對$y^1 - \bar{y}$ 作SVD



PCA v.s. Linear autoencoder

	PCA	Linear Autoencoder
Eigenvector比較	Eigenvector彼此是正交的，在新空間裡的每個軸彼此垂直	找出的W不一定正交，做SVD後才與PCA相等
在不同維度下的運行情況	在高維度時，要切割運算（Local PCA）	維度較無限制（可作高維度）
優缺點	1. 若data有分組，做PCA會混在一起（因為無label） 2. 無法做non-linear	1. 可自行加深hidden layer 2. 在linear下，PCA運行較快