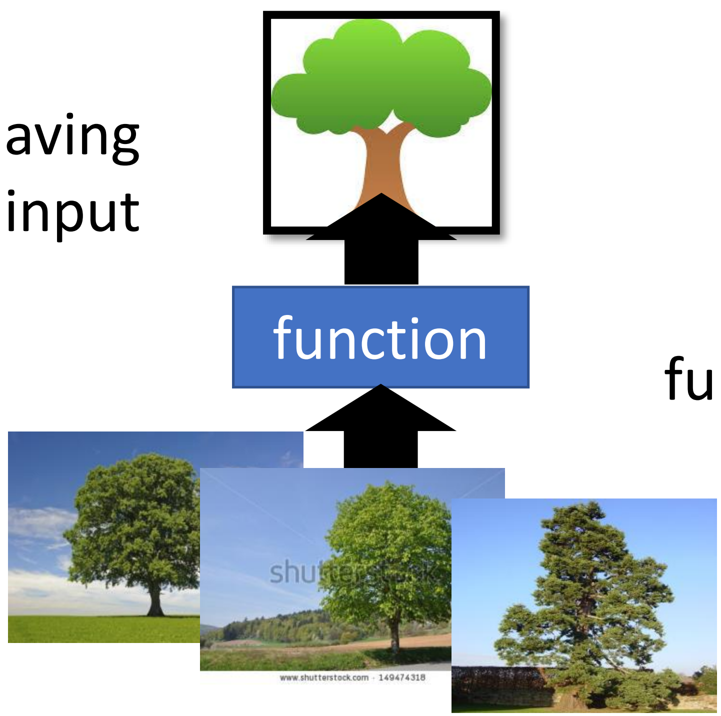
Unsupervised Learning: Principle Component Analysis

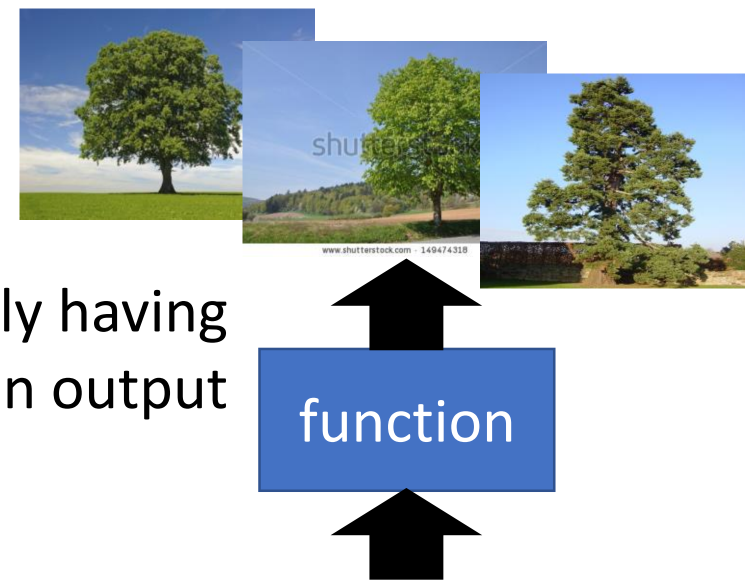
Unsupervised Learning

- Dimension Reduction (化繁為簡)

only having function input



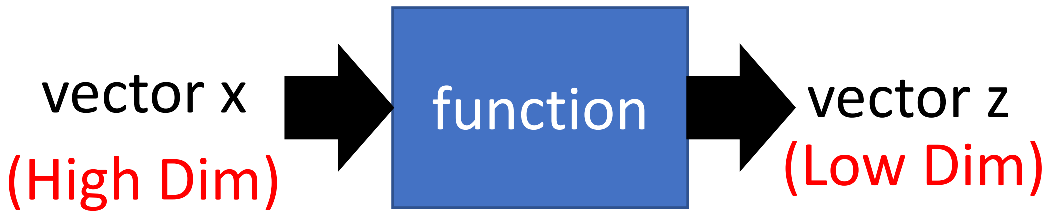
- Generation (無中生有)

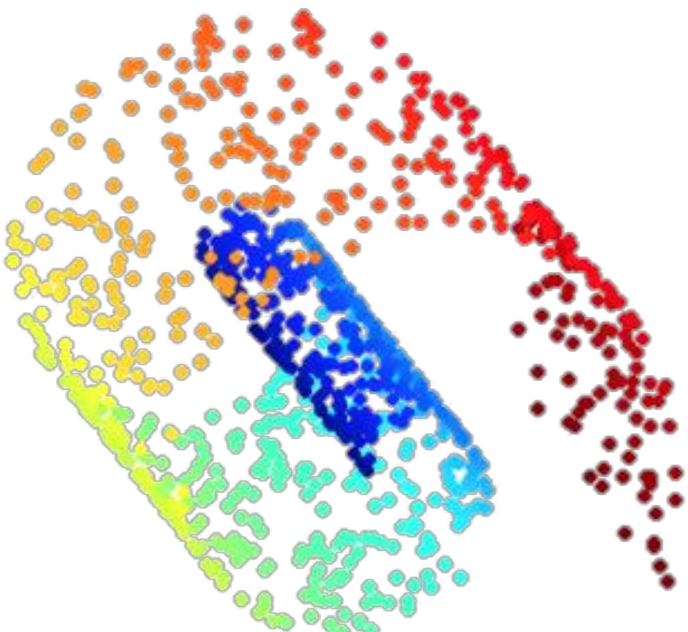


only having function output

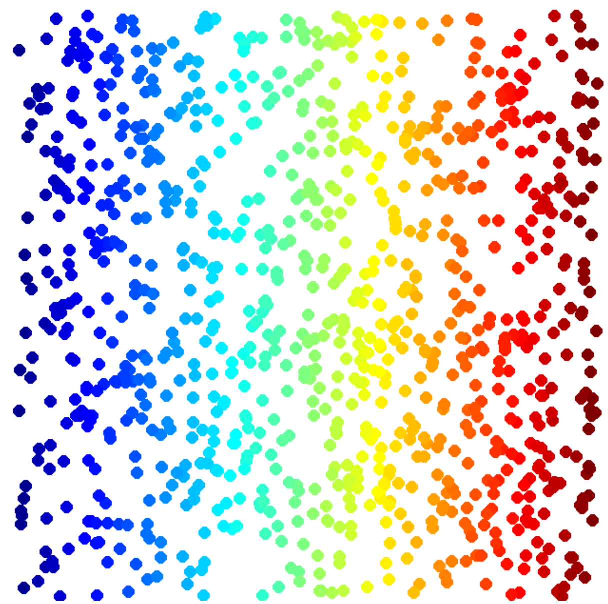
Random numbers

Dimension Reduction



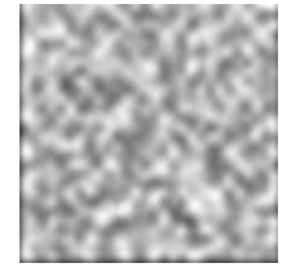


Looks like 3-D

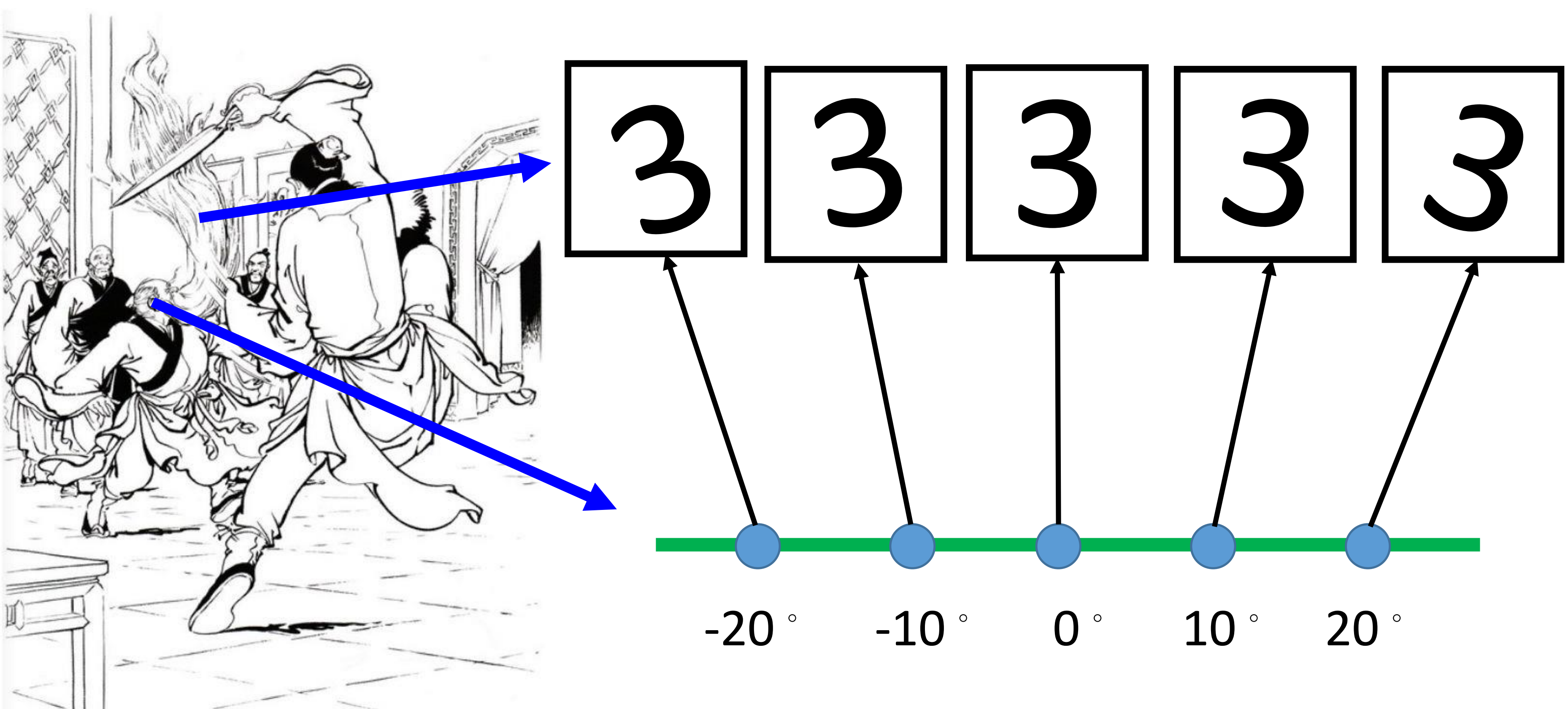


Actually, 2-D

Dimension Reduction



- In MNIST, a digit is 28 x 28 dims.
    - Most 28 x 28 dim vectors are not digits



分群的結果屬於一種dimension reduction

Clustering

$\begin{bmatrix}1\\0\\0\end{bmatrix}$ Cluster 1

$\begin{bmatrix}0\\1\\0\end{bmatrix}$ Cluster 2

Cluster 3 $\begin{bmatrix}0\\0\\1\end{bmatrix}$

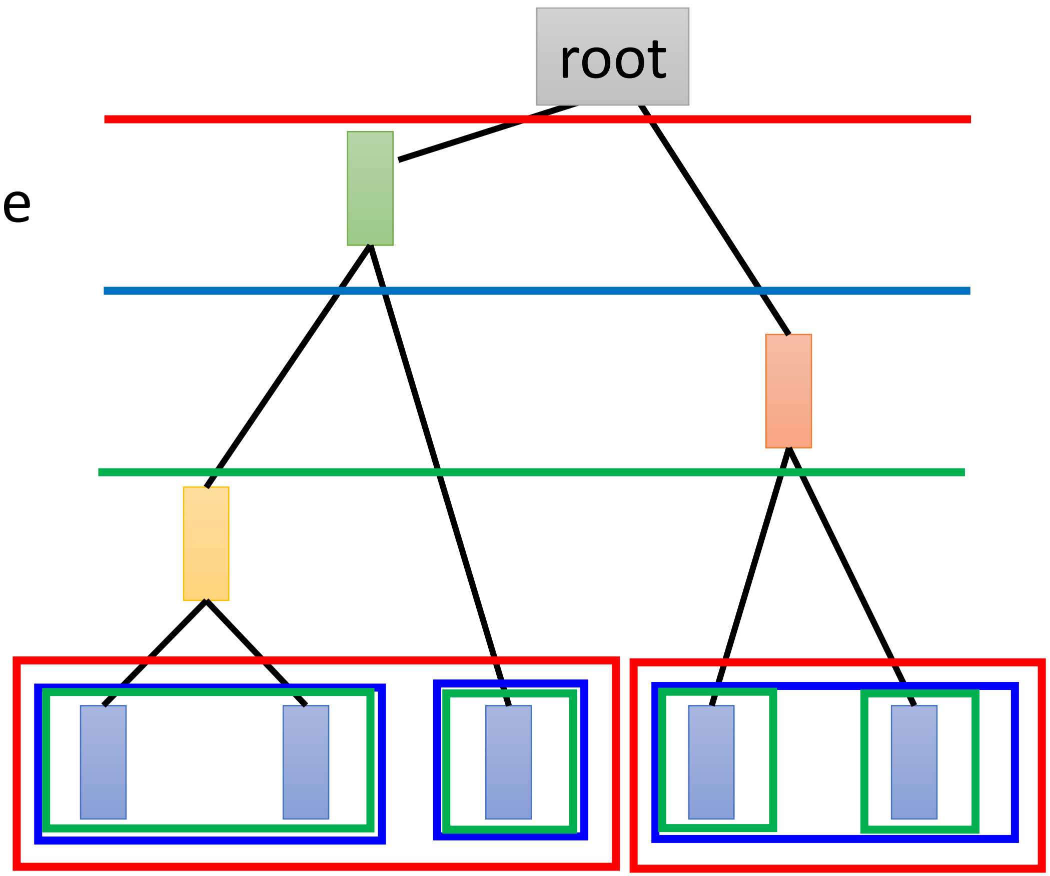
- K-means
  - Clustering $X = \{x^1, \cdots, x^n, \cdots, x^N\}$ into K clusters
  - Initialize cluster center c^i, i=1,2, ... K (K random x^n from X)
  - Repeat
    - For all x^n in X: $b_i^n \begin{cases} 1 & x^n \text{ is most “\textbf{\textit{close}}” to } c^i \\ 0 & \text{Otherwise} \end{cases}$
    - Updating all c^i: $c^i = \sum_{x^n} b_i^n x^n \Big/ \sum_{x^n} b_i^n$

Clustering

- Hierarchical Agglomerative Clustering (HAC)

Step 1: build a tree

Step 2: pick a threshold

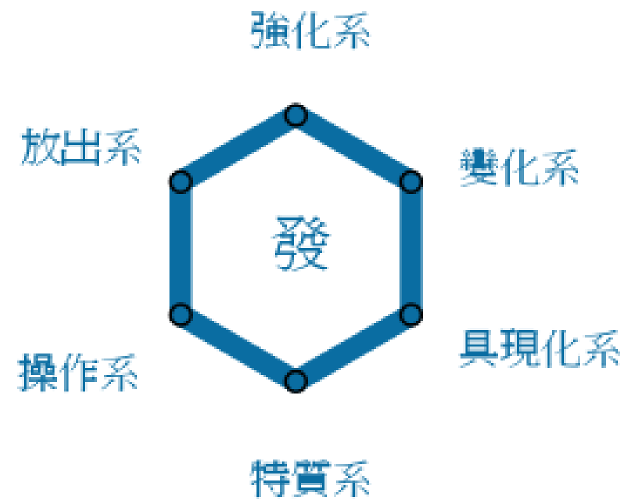


Distributed Representation

- Clustering: an object must belong to one cluster

小傑是強化系

- Distributed representation

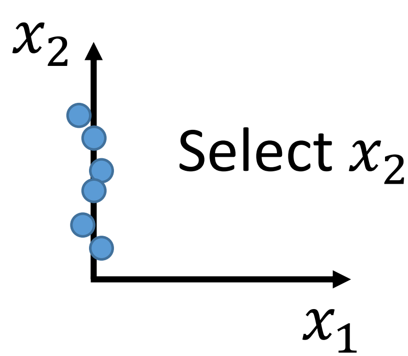


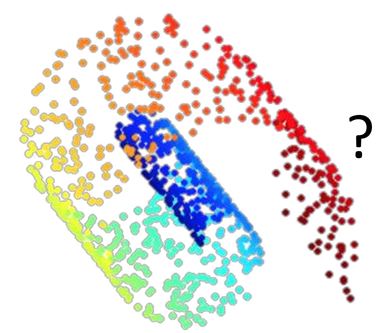
小傑是

強化系	0.70
放出系	0.25
變化系	0.05
操作系	0.00
具現化系	0.00
特質系	0.00

Distributed Representation

- Feature selection





- Principle component analysis (PCA)
  [Bishop, Chapter 12]

linear transform

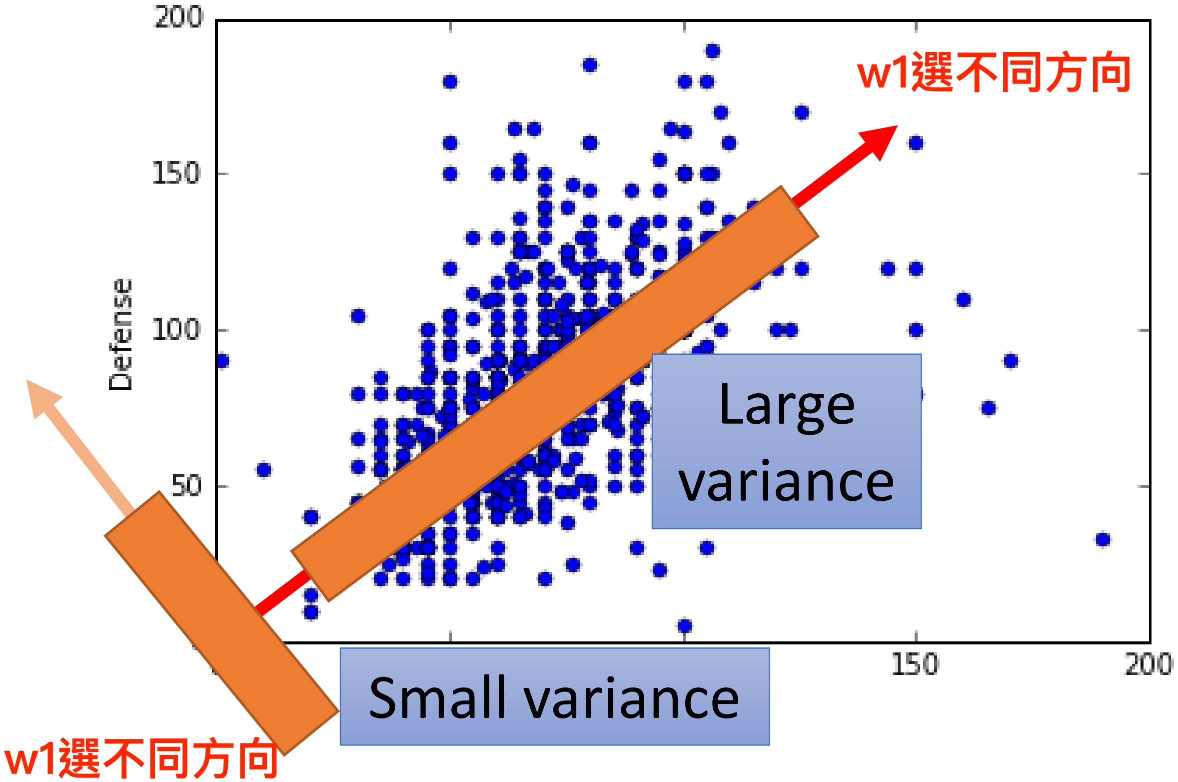
$$z = Wx$$

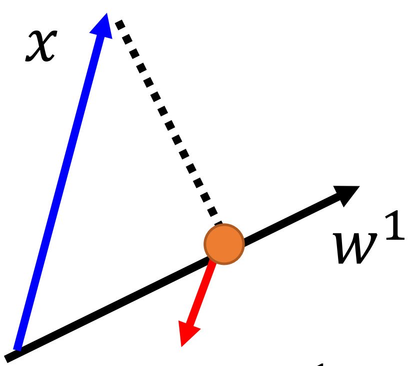
PCA

$$z = Wx$$

Reduce to 1-D:

$$z_1 = w^1 \cdot x$$





$$z_1 = w^1 \cdot x$$

z1為x投影在w1上的距離

Project all the data points x onto w^1, and obtain a set of z_1

選擇project後投影範圍越大的方向越好

We want the variance of z_1 as large as possible

$$Var(z_1) = \sum_{z_1} (z_1 - \bar{z_1})^2 \qquad \boxed{\|w^1\|_2 = 1}$$

Constrain

PCA

$$z = Wx$$

Reduce to 1-D:

固定w的長度，找出最大Var

$$z_1 = w^1 \cdot x$$

$$z_2 = w^2 \cdot x$$

$$W = \begin{bmatrix} (w^1)^T \\ (w^2)^T \\ \vdots \end{bmatrix}$$

Orthogonal matrix

Project all the data points x onto w^1, and obtain a set of z_1

We want the variance of z_1 as large as possible

$$Var(z_1) = \sum_{z_1} (z_1 - \bar{z_1})^2 \quad \|w^1\|_2 = 1$$

We want the variance of z_2 as large as possible

$$Var(z_2) = \sum_{z_2} (z_2 - \bar{z_2})^2 \quad \|w^2\|_2 = 1$$

$$w^1 \cdot w^2 = 0$$

orthogonal,否則會找到與w1相同

Warning of Math

PCA

$$z_1 = w^1 \cdot x$$

$$\bar{z_1} = \frac{1}{N}\sum z_1 = \frac{1}{N}\sum w^1 \cdot x = w^1 \cdot \frac{1}{N}\sum x = w^1 \cdot \bar{x}$$

$$Var(z_1) = \frac{1}{N}\sum_{z_1}(z_1 - \bar{z_1})^2$$

$$= \frac{1}{N}\sum_{x}(w^1 \cdot x - w^1 \cdot \bar{x})^2$$

$$= \boxed{\frac{1}{N}\sum\left(w^1 \cdot (x - \bar{x})\right)^2}$$

$$= \frac{1}{N}\sum (w^1)^T(x - \bar{x})(x - \bar{x})^T w^1$$

降維前的feature的covariance matrix

w1都提出來

$$= (w^1)^T \boxed{\frac{1}{N}\sum (x - \bar{x})(x - \bar{x})^T} w^1$$

$$= (w^1)^T Cov(x) w^1$$

$$S = Cov(x)$$

$$(a \cdot b)^2 = (a^T b)^2 = a^T b a^T b$$

純量可做transpose

$$= a^T b (a^T b)^T = a^T b b^T a$$

Find w^1 maximizing

$$(w^1)^T S w^1$$

constrain

$$\|w^1\|_2 = (w^1)^T w^1 = 1$$

Find w^1 maximizing $(w^1)^T S w^1$ $\quad (w^1)^T w^1 = 1$

$S = Cov(x)$ Symmetric positive-semidefinite
(non-negative eigenvalues)

Using Lagrange multiplier [Bishop, Appendix E]

$$g(w^1) = (w^1)^T S w^1 - \alpha\left((w^1)^T w^1 - 1\right)$$

constrain

其gradient必為0，找極值

$$\partial g(w^1)/\partial w_1^1 = 0$$

$$\partial g(w^1)/\partial w_2^1 = 0$$

$$\vdots$$

其實兩邊應該還有一個*2，但是同除2消掉了

$$Sw^1 - \alpha w^1 = 0$$

$$Sw^1 = \alpha w^1$$

w^1 : eigenvector

$$(w^1)^T S w^1 = \alpha (w^1)^T w^1$$

$$= \alpha$$

Choose the maximum one

w^1 is the eigenvector of the covariance matrix S

Corresponding to the largest eigenvalue λ_1

Find w^2 maximizing $(w^2)^T S w^2$ $\quad (w^2)^T w^2 = 1 \quad (w^2)^T w^1 = 0$

第1個constrain 第2個constrain

$$g(w^2) = (w^2)^T S w^2 - \alpha\left((w^2)^T w^2 - 1\right) - \beta\left((w^2)^T w^1 - 0\right)$$

第1個constrain 第2個constrain

$\partial g(w^2)/\partial w_1^2 = 0$

$\partial g(w^2)/\partial w_2^2 = 0$

$\vdots$

$$S w^2 - \alpha w^2 - \beta w^1 = 0$$

同乘transpose(w1)

$$\boxed{0} - \alpha \boxed{0} - \beta \boxed{1} = 0$$

純量轉置

$$= \left(\left(w^1\right)^T S w^2\right)^T = (w^2)^T S^T w^1$$

orthogonal

$$= (w^2)^T S w^1 = \lambda_1 (w^2)^T w^1 = 0$$

得到beta=0

$$\boxed{S w^1 = \lambda_1 w^1}$$

$\beta = 0$: $\quad S w^2 - \alpha w^2 = 0 \quad S w^2 = \alpha w^2$

w^2 is the eigenvector of the covariance matrix S

Corresponding to the 2nd largest eigenvalue λ_2

做完PCA降維後可使dimension之間沒有corelation

PCA - decorrelation

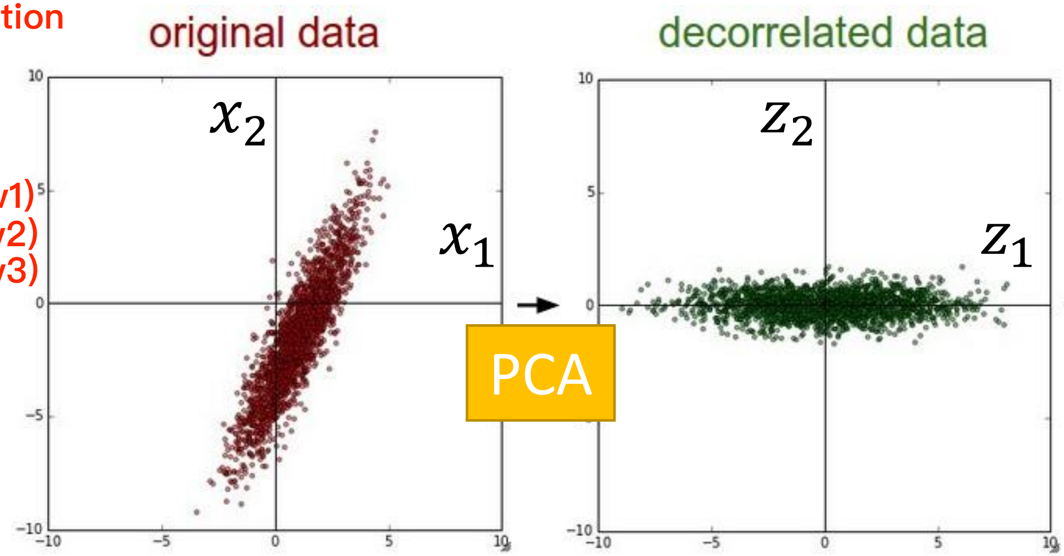
$$z = Wx$$

W = transpose(w1)
transpose(w2)
transpose(w3)
.
.

$$Cov(z) = D$$

dimension間沒有corelation

Diagonal matrix



做完PCA後丟給其他的model，其他的model可以假設dimension間沒有corelation，因此可以選擇比較簡單的model避免overfitting

z=wx

$$Cov(z) = \frac{1}{N}\sum(z-\bar{z})(z-\bar{z})^T = WSW^T \qquad S = Cov(x)$$

$$= WS[w^1 \quad \cdots \quad w^K] = W[Sw^1 \quad \cdots \quad Sw^K]$$

$$= W[\lambda_1 w^1 \quad \cdots \quad \lambda_K w^K] = [\lambda_1 W w^1 \quad \cdots \quad \lambda_K W w^K]$$

$$= [\lambda_1 e_1 \quad \cdots \quad \lambda_K e_K] = D$$

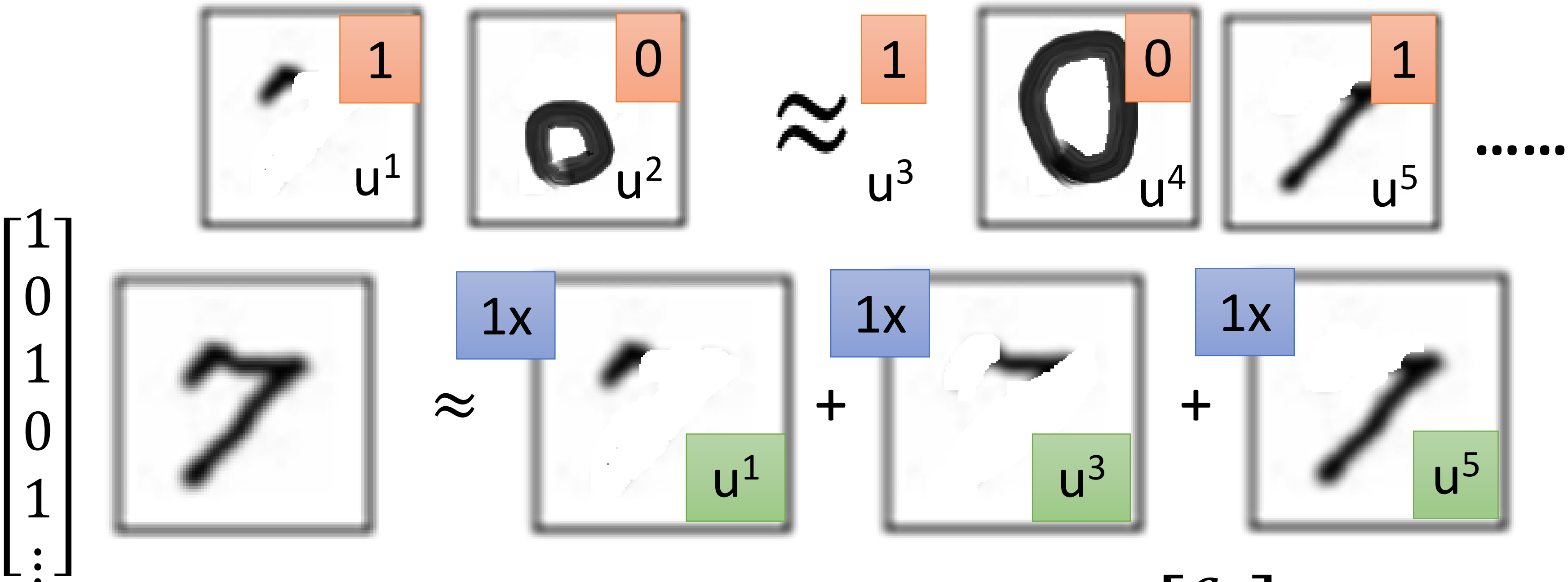
Diagonal matrix

因為orthogonal，所以只有對角線有值(lambda)，其餘為零

End of Warning

PCA – Another Point of View

Basic Component: 把pixel-wise降維至component-wise



$$x \approx c_1 u^1 + c_2 u^2 + \cdots + c_K u^K + \bar{x}$$

Pixels in a digit image

component

$$\begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_K \end{bmatrix}$$

Represent a digit image

PCA – Another Point of View

如果component數目不夠則x^無法跟x接近

$$x - \bar{x} \approx c_1 u^1 + c_2 u^2 + \cdots + c_K u^K \quad = \hat{x}$$

Reconstruction error:
$\| (x - \bar{x}) - \hat{x} \|_2$

Find $\{u^1, \ldots, u^K\}$ minimizing the error

$$L = \min_{\{u^1,\ldots,u^K\}} \sum \left\| (x - \bar{x}) - \underbrace{\left(\sum_{k=1}^{K} c_k u^k \right)}_{\hat{x}} \right\|_2$$

PCA: $z = Wx$

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} (w_1)^{\mathrm{T}} \\ (w_2)^{\mathrm{T}} \\ \vdots \\ (w_K)^{\mathrm{T}} \end{bmatrix} x$$

from PCA(w1...wk are Eigen vector)

$\{w^1, w^2, \ldots w^K\}$ is the component $\{u^1, u^2, \ldots u^K\}$ minimizing L

Proof in [Bishop, Chapter 12.1.2]

$$x - \bar{x} \approx c_1 u^1 + c_2 u^2 + \cdots + c_K u^K \;\; = \hat{x}$$

Reconstruction error:
$\| (x - \bar{x}) - \hat{x} \|_2$

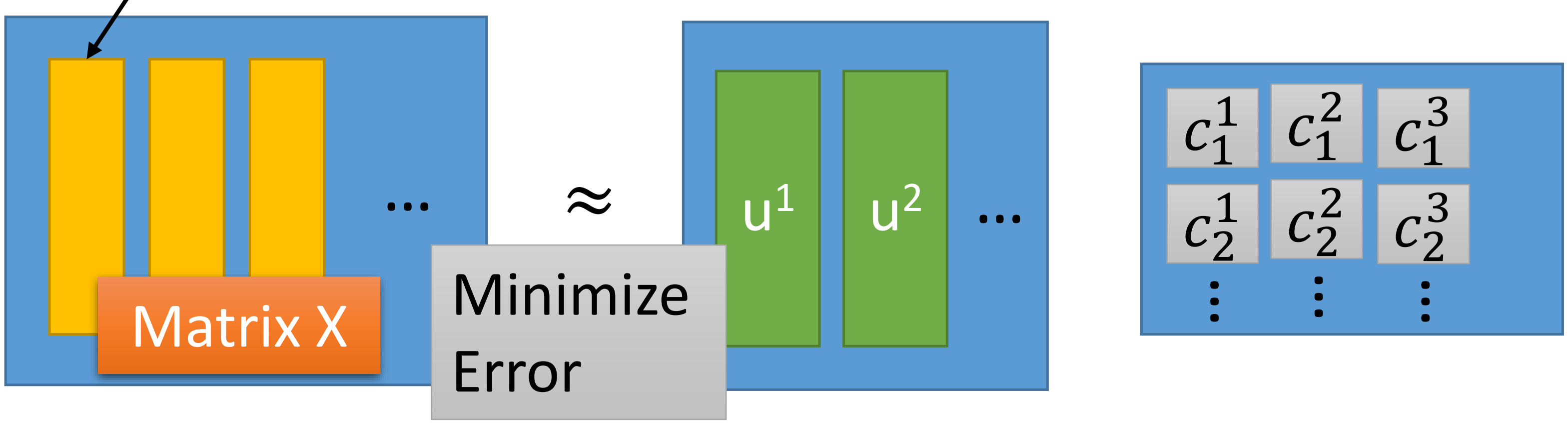
Find $\{u^1, \ldots, u^K\}$ minimizing the error

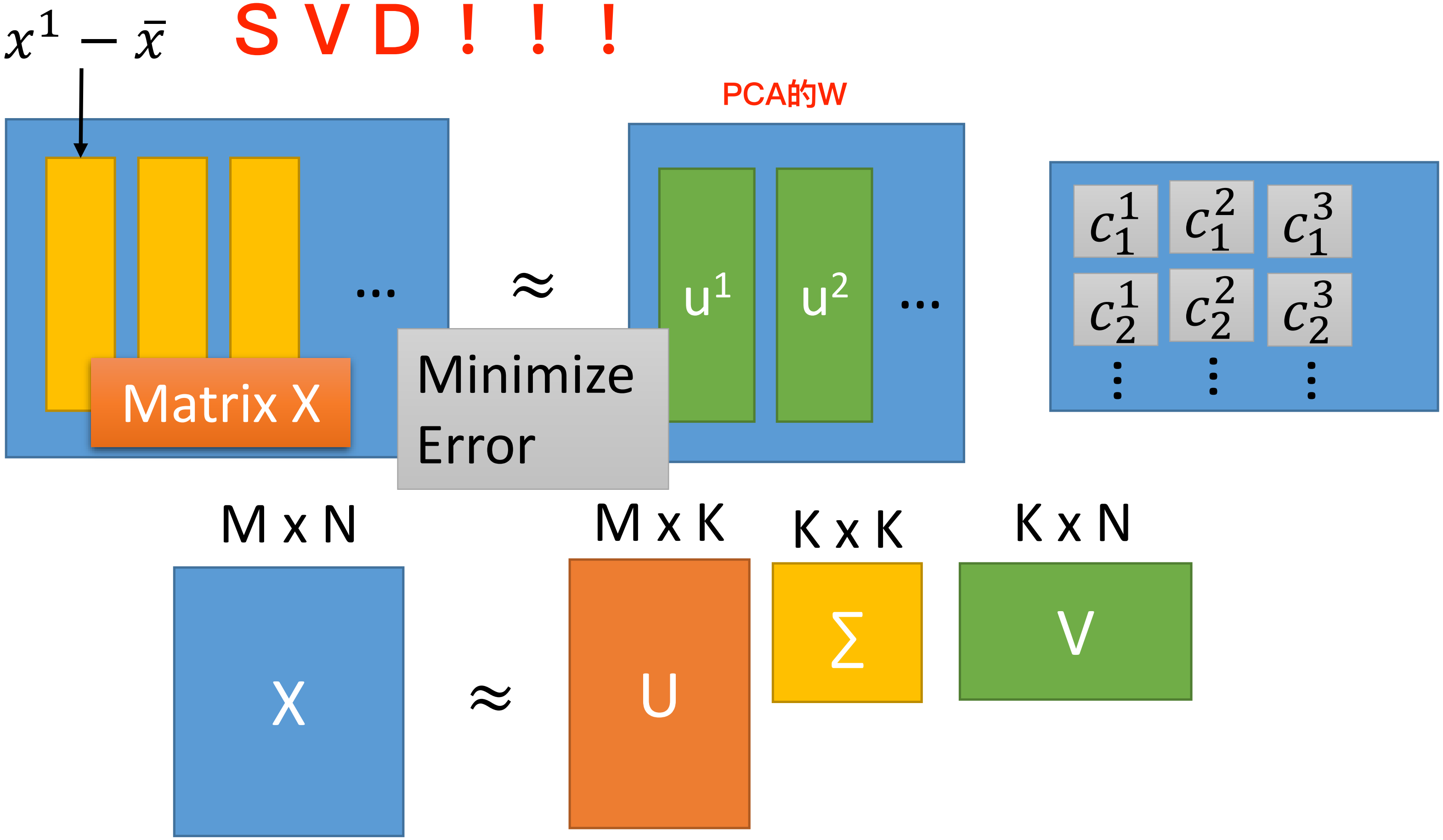
$$x^1 - \bar{x} \approx c_1^1 u^1 + c_2^1 u^2 + \cdots$$

$$x^2 - \bar{x} \approx c_1^2 u^1 + c_2^2 u^2 + \cdots$$

$$x^3 - \bar{x} \approx c_1^3 u^1 + c_2^3 u^2 + \cdots$$

$$\vdots$$





K columns of U: a set of orthonormal eigen vectors corresponding to the k largest eigenvalues of XX^T

U 的 解 即 為 P C A 的 解 ：
X*transpose(X)的eigenvalue

This is the solution of PCA

SVD:
http://speech.ee.ntu.edu.tw/~tlkagk/courses/LA_2016/Lecture/SVD.pdf

PCA looks like a neural network with one hidden layer (linear activation function)

Autoencoder

If $\{w^1, w^2, \dots w^K\}$ is the component $\{u^1, u^2, \dots u^K\}$

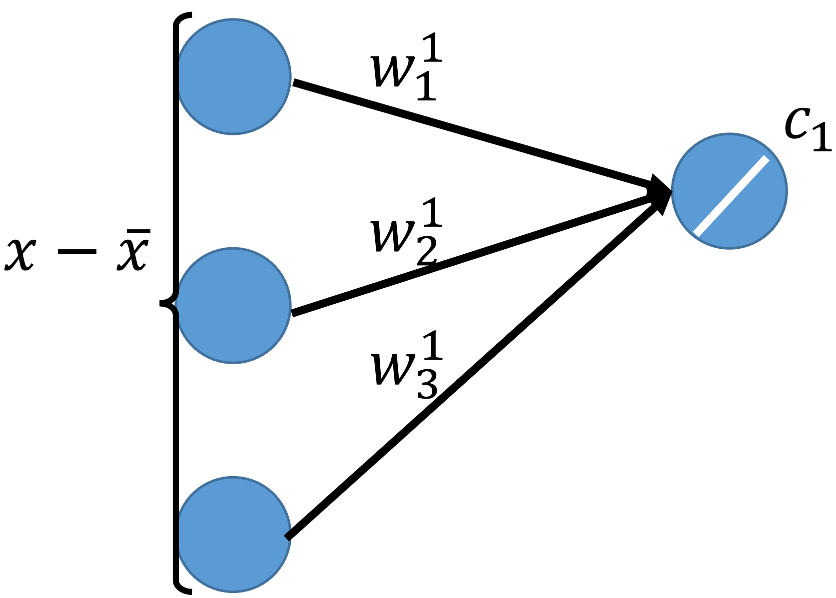
$$\hat{x} = \sum_{k=1}^{K} c_k w^k \Leftrightarrow x - \bar{x}$$

To minimize reconstruction error:

$$c_k = (x - \bar{x}) \cdot w^k$$

因為w1...wk為orthonormal

$K = 2$:



PCA looks like a neural network with one hidden layer (linear activation function)

Autoencoder

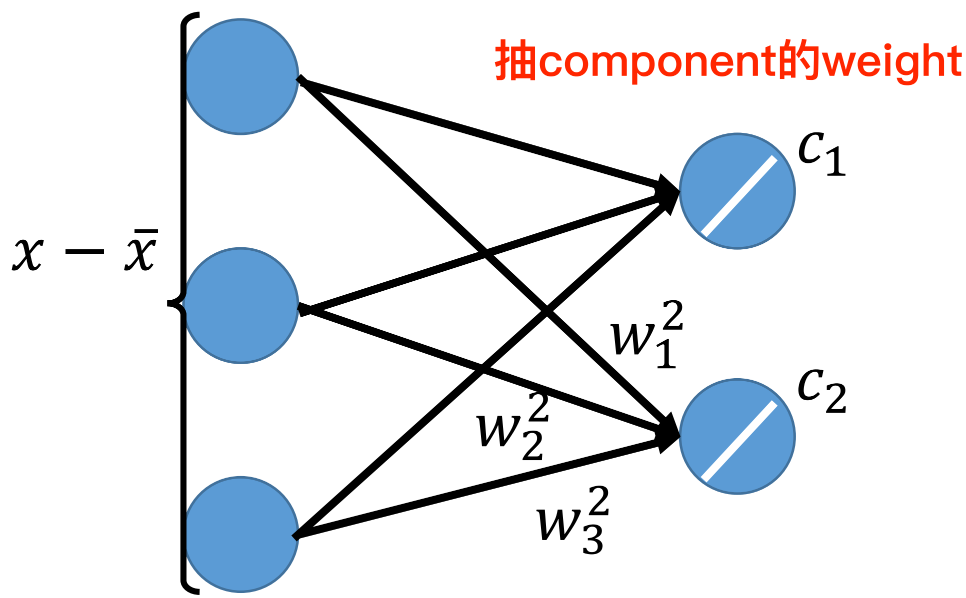
If $\{w^1, w^2, \ldots w^K\}$ is the component $\{u^1, u^2, \ldots u^K\}$

$$\hat{x} = \sum_{k=1}^{K} c_k w^k \Longleftrightarrow x - \bar{x}$$

To minimize reconstruction error:

$$c_k = (x - \bar{x}) \cdot w^k$$

$K = 2$:



PCA looks like a neural network with one hidden layer (linear activation function)

Autoencoder

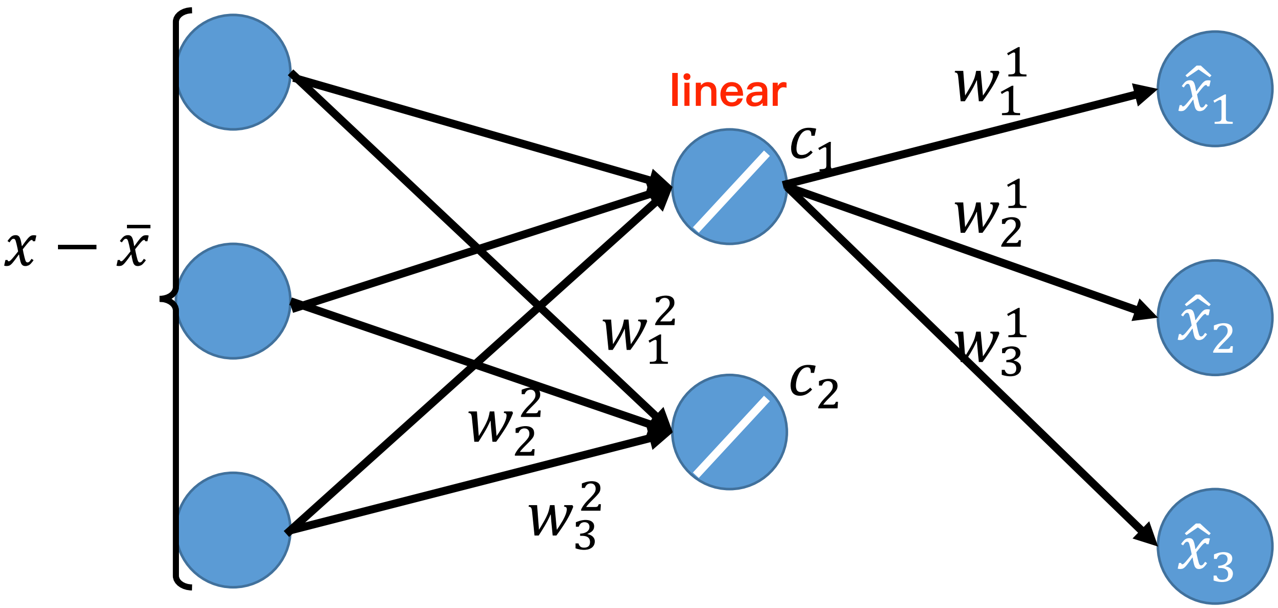
If $\{w^1, w^2, \dots w^K\}$ is the component $\{u^1, u^2, \dots u^K\}$

$$\hat{x} = \sum_{k=1}^{K} c_k w^k \Leftrightarrow x - \bar{x}$$

To minimize reconstruction error:

$$c_k = (x - \bar{x}) \cdot w^k$$

$K = 2$:



PCA looks like a neural network with one hidden layer (linear activation function)

Autoencoder

If $\{w^1, w^2, \dots w^K\}$ is the component $\{u^1, u^2, \dots u^K\}$

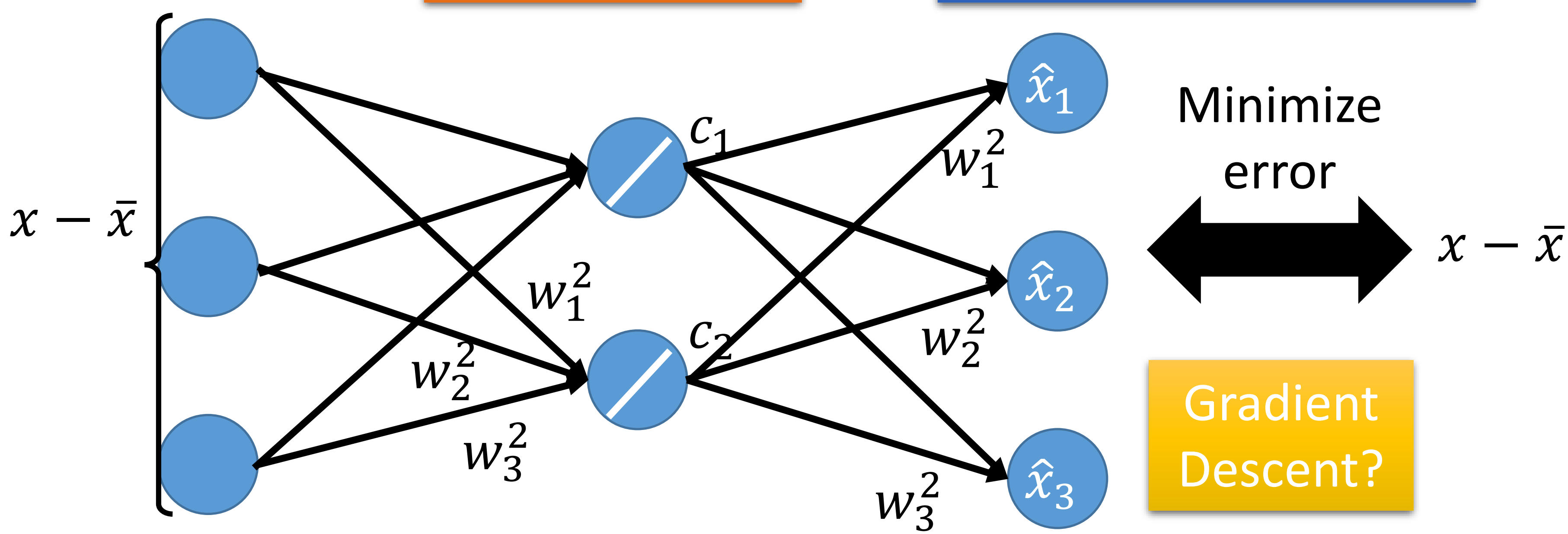
$$\hat{x} = \sum_{k=1}^{K} c_k w^k \Leftrightarrow x - \bar{x}$$

To minimize reconstruction error:

$$c_k = (x - \bar{x}) \cdot w^k$$

$K = 2$:

It can be deep. → Deep Autoencoder



Train DNN，其結果與PCA會不同，因為PCA有些限制例如其不同dimension間必須是orthogonal

PCA - Pokémon

- Inspired from: https://www.kaggle.com/strakul5/d/abcsds/pokemon/principal-component-analysis-of-pokemon-data
- 800 Pokemons, 6 features for each (HP, Atk, Def, Sp Atk, Sp Def, Speed)
- How many principle components?

eigen value

$$\frac{\lambda_i}{\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 + \lambda_5 + \lambda_6}$$

	λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
ratio	0.45	0.18	0.13	0.12	0.07	0.04

Using 4 components is good enough

PCA - Pokémon

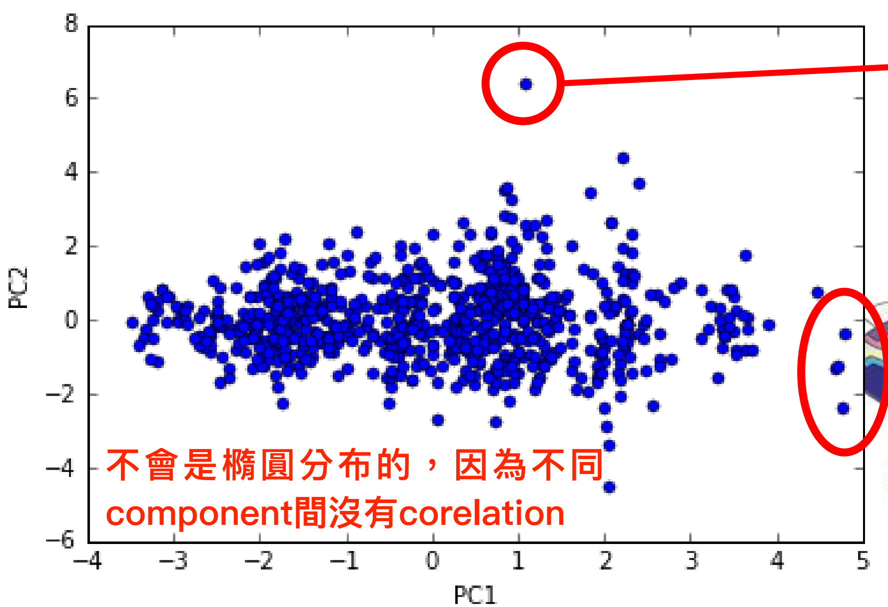
每隻pokemon的素質都是由PC1~PC4四個component做linear combination

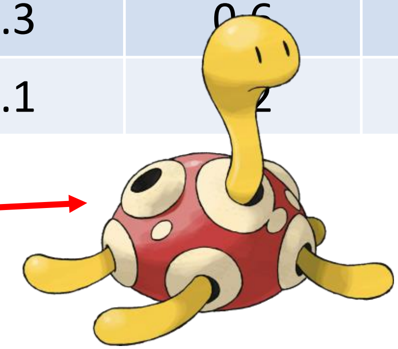
四個component

	HP	Atk	Def	Sp Atk	Sp Def	Speed
PC1	0.4	0.4	0.4	0.5	0.4	0.3
PC2	0.1	0.0	0.6	-0.3	0.2	-0.7
PC3	-0.5	-0.6	0.1	0.3	0.6	[illegible]
PC4	0.7	-0.4	-0.4	0.1	[illegible]	-0.3

強度

防禦(犧牲速度)



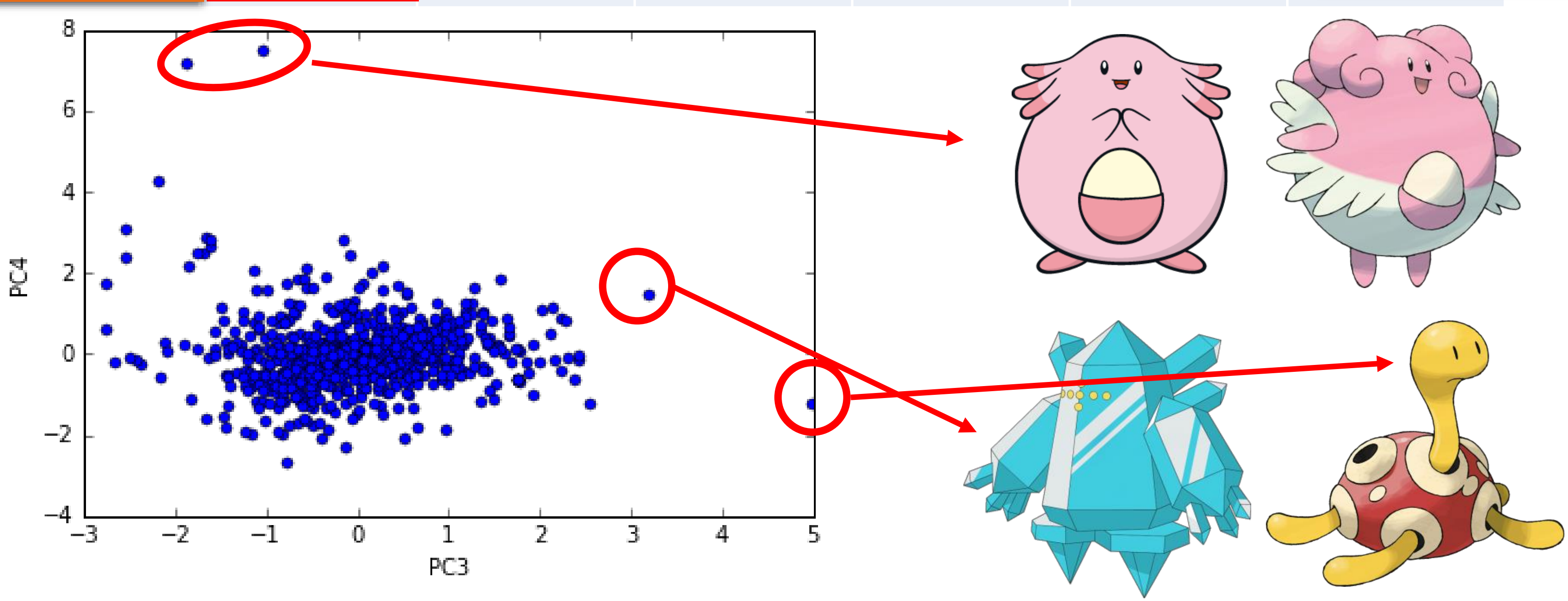


PCA - Pokémon

	HP	Atk	Def	Sp Atk	Sp Def	Speed
PC1	0.4	0.4	0.4	0.5	0.4	0.3
PC2	0.1	0.0	0.6	-0.3	0.2	-0.7
PC3	-0.5	-0.6	0.1	0.3	0.6	
PC4	0.7	-0.4	-0.4	0.1	0.2	

特殊防禦(犧牲攻擊和生命)

生命力強



PCA - Pokémon

- http://140.112.21.35:2880/~tlkagk/pokemon/pca.html
- The code is modified from
  - http://jkunst.com/r/pokemon-visualize-em-all/

PCA - MNIST

針對image做PCA

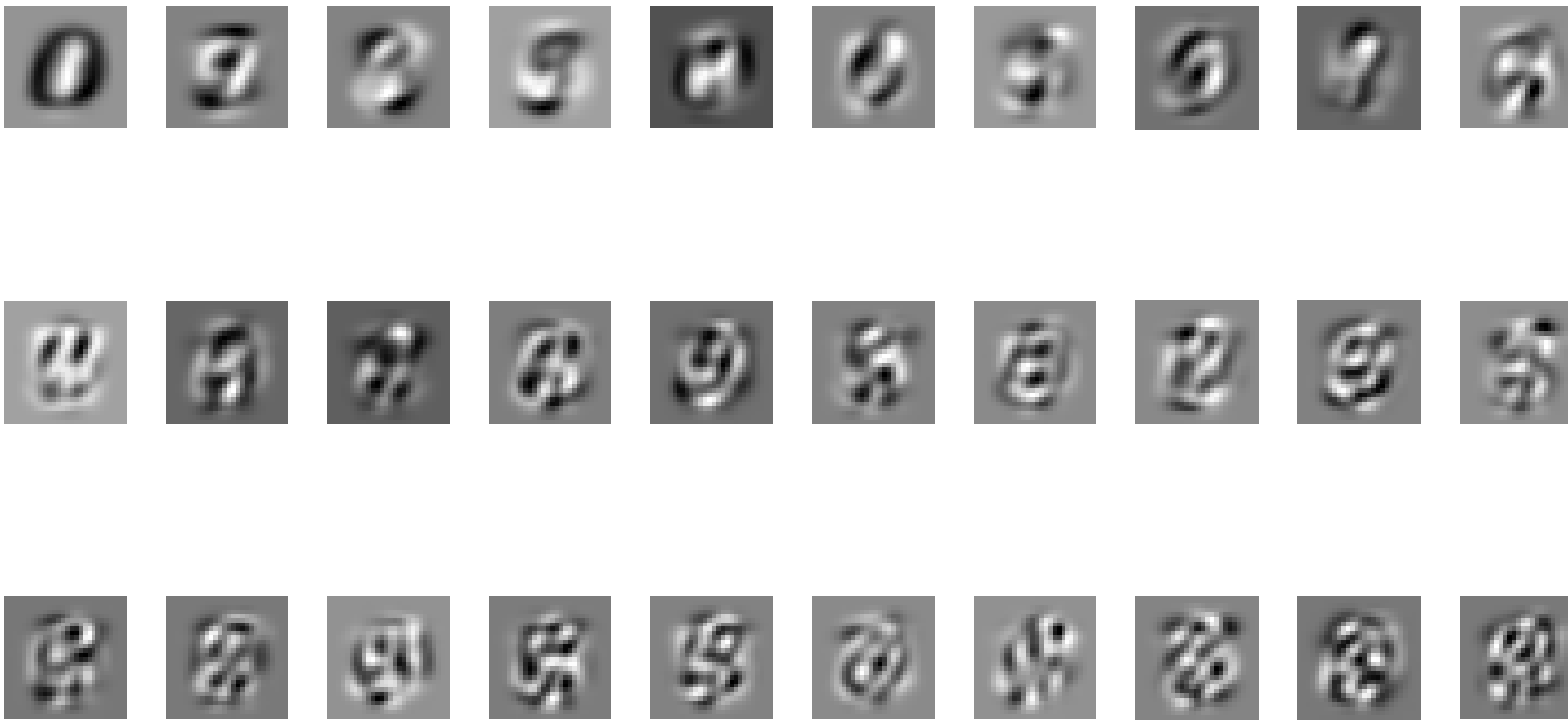
$$= a_1 w^1 + a_2 w^2 + \cdots$$

每個weight可以是正的或是負的實數

images

30 components:

Non–negative matrix factorization(NMF)使所有wight變成正的，因此所有component會變成類似筆畫的樣子



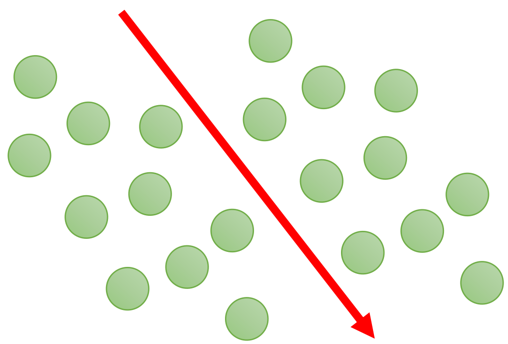
Eigen-digits

Weakness of PCA

PCA是linear transform的，因此做完reduction會直接打平在平面，無法拉直

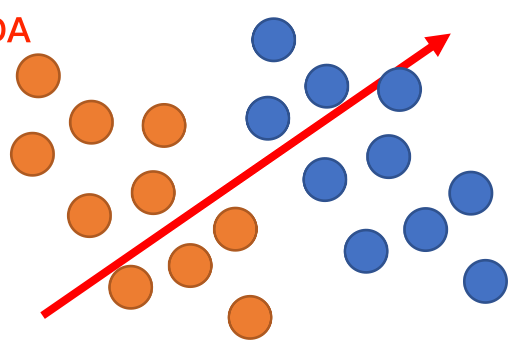
靠variance決定distribution

- Unsupervised



PCA

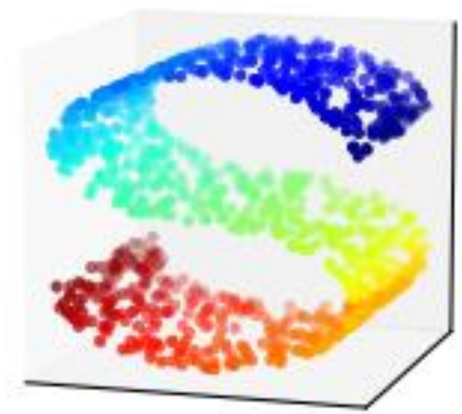
如果資料本身已經有label，做PCA反而會混在一起
因此改用LDA

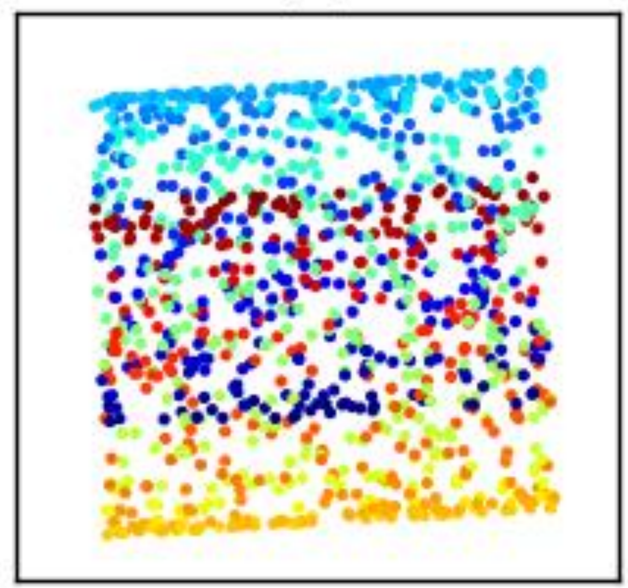


LDA

LDA考慮label data的降維，但是事supervis

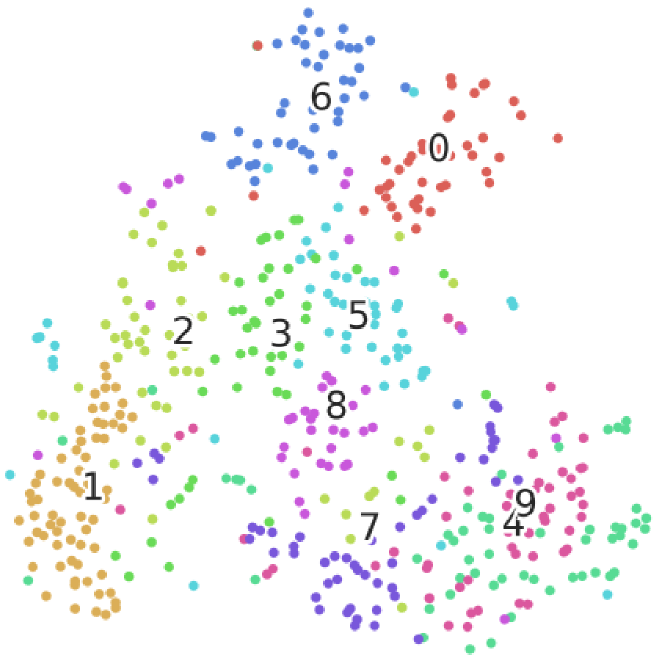
- Linear



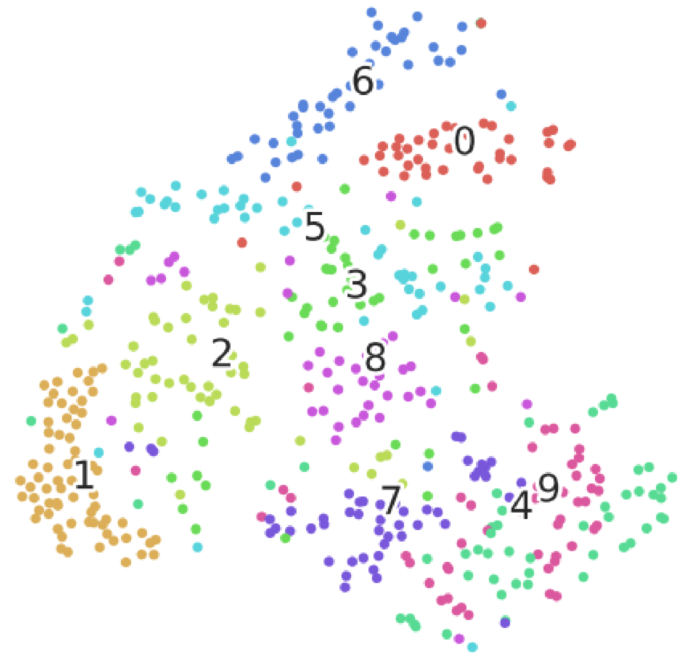


http://www.astroml.org/book_figures/chapter7/fig_S_manifold_PCA.html

Weakness of PCA



Pixel (28x28) -> tSNE (2)



PCA (32) -> tSNE (2)

非線性的轉換

Non-linear dimension reduction in the following lectures

Acknowledgement

- 感謝 彭冲 同學發現引用資料的錯誤
- 感謝 Hsiang-Chih Cheng 同學發現投影片上的錯誤

Matrix Factorization可以用在推薦系統，去計算每個人會購買的東西的矩陣的SVD，找出兩個矩陣相乘。
找出兩個矩陣後其對應的row/column做inner product後即為預測這個人購買這個東西的可能度

Appendix

- http://4.bp.blogspot.com/_sHcZHRnxlLE/S9EpFXYjfvI/AAAAAAAABZ0/_oEQiaR3WVM/s640/dimensionality+reduction.jpg
- https://lvdmaaten.github.io/publications/papers/TR_Dimensionality_Reduction_Review_2009.pdf