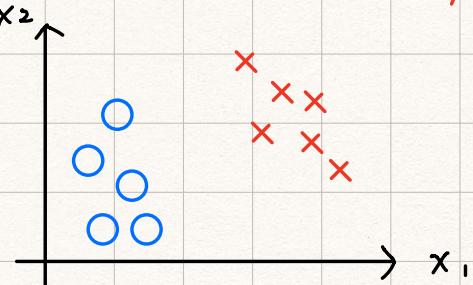
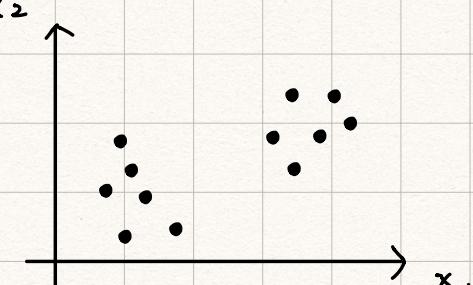
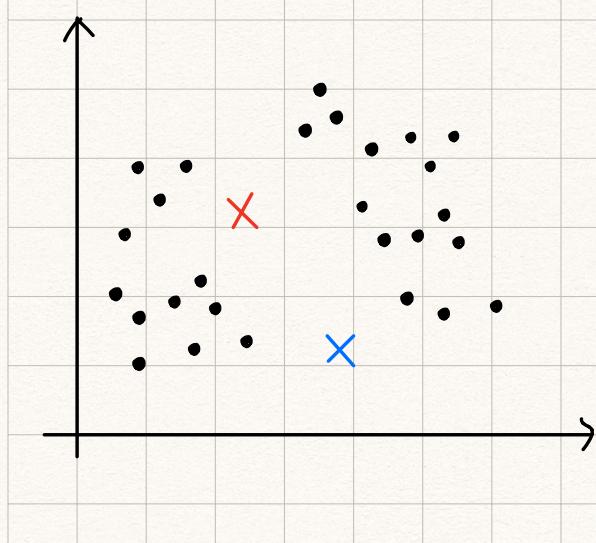


Clustering

Unsupervised Learning : Introduction

Supervised	unsupervised
<p>find decision boundary</p>  <p>Training set has labels</p>	<p>find structure</p>  <p>Training set without label</p>

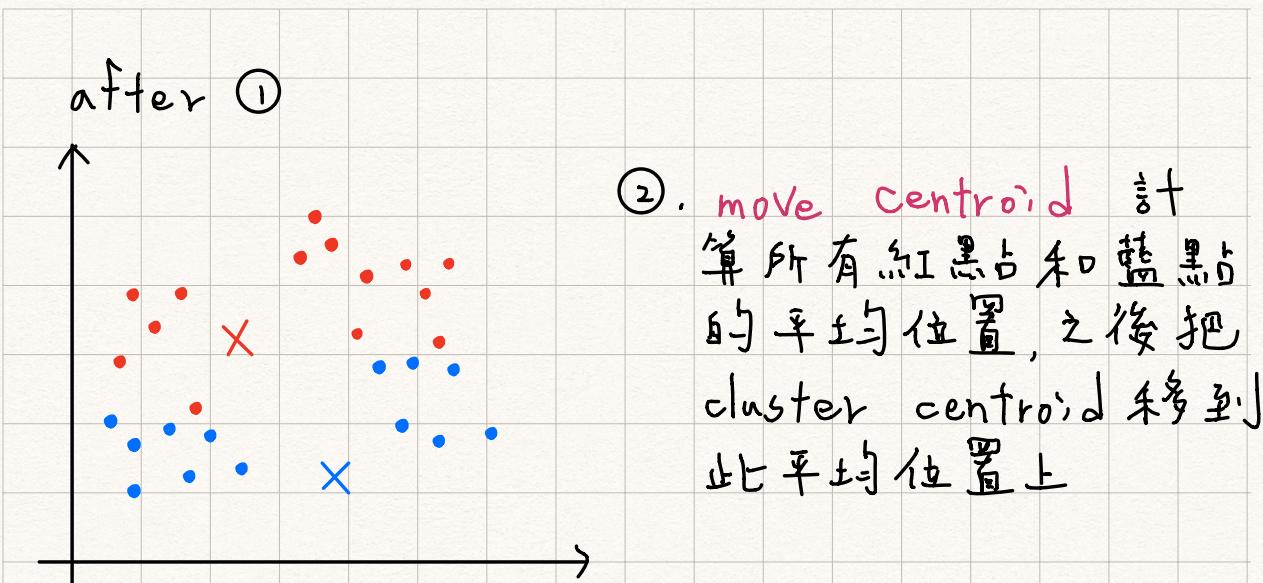
K - Means Algorithm



數量由想分幾個 group 來決定

↓
X, X: Cluster Centroid

- ①. Cluster assignment 比較樣本點和 Cluster Centroid 的距離，比較接近的便把它歸為此類 group



重複 ① ② ① ② ① ... 的動作，便能完成 k-means

Steps of K-means algorithm

Input:

- k (number of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$

$x^{(i)} \in R^n$ (而不是 $n+1$ 個維度)

Step 1: Randomly initialize K cluster centroids
 $M_1, M_2, M_3, \dots, M_K \in R^n$

Step 2: for $i=1$ to m ← 資料樣本數 m

↑
cluster
assignment

$c^{(i)} := \text{index}(\text{from } 1 \text{ to } k) \text{ of cluster}$
centroid closer to $x^{(i)}$

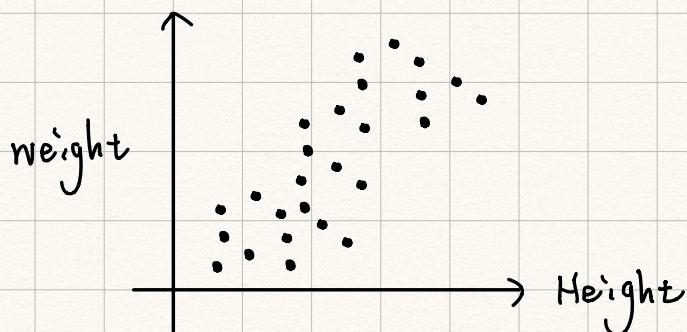
$$\min_k \sum_{i=1}^n \|x^{(i)} - M_k\|^2, \quad x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)} \text{ 靠近 } M_2 \text{ 則 } C^{(1)} = 2, C^{(2)} = 2$$

Step 3: for $k=1$ to K

\uparrow
more centroid
 $M_k := \text{average}(\text{mean of points assigned to cluster } k)$

$$M_2 = \frac{1}{3}(x^{(1)} + x^{(2)} + x^{(3)})$$

K-means for non-separated clusters



K-means 虽然可以将左图分成一个簇 cluster

K-means optimization objective

$C^{(i)}$ 表示 x_i 属於哪個 cluster

M_k 為第 k 個 cluster centroid

$M_{C^{(i)}}$ 表示 x_i 的 cluster centroid

$$x^{(i)} = 5 \rightarrow C^{(i)} = 5 \rightarrow M_{C^{(i)}} = M_5$$

Optimization objective

$$J(c^{(1)}, \dots, c^{(m)}, M_1, \dots, M_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - M_{c(i)}\|^2$$

找到 $c^{(1)} \cup c^{(m)}$, $M_1 \cup M_K$ 使得 $J(\cdot)$ 最小

Random initialization

should have $K < m$.

Randomly pick K training examples.

Set M_1, \dots, M_K equal to these K examples.

$$M_1 = x^{(i)}, M_2 = x^{(j)}$$

隨機挑選 training examples 中的 K 個值最為 cluster centroid.

→ 可能陷入局部最佳

Global optima

→ $K = 2 \sim 10$, 用此方法效果都不錯

For $i=1$ to 100 :

隨機初始化 cluster centroid 的位置，
爾後 RUN K-means algorithm, 並把最後的 $J(\cdot)$ 記下來

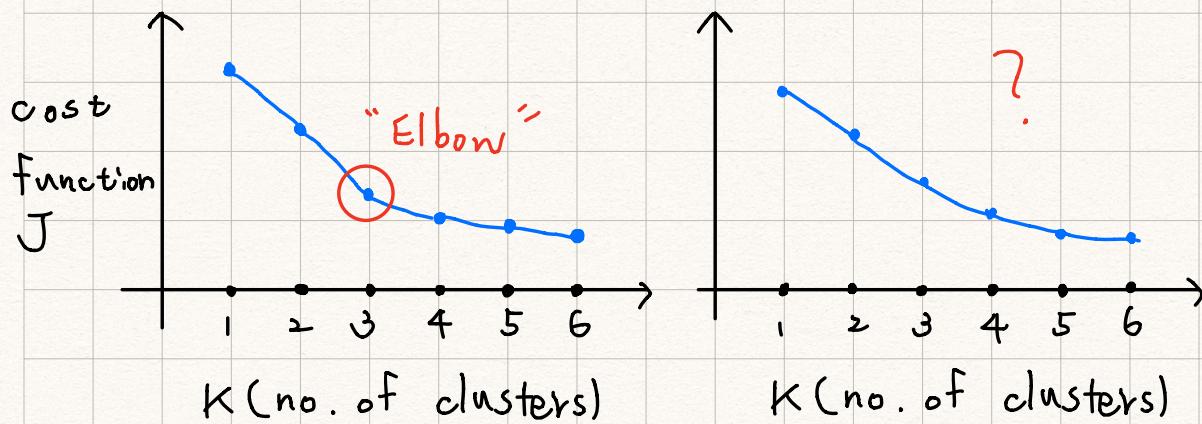
最後，從這 100 個 $J(\cdot)$ 中挑出最小的做為結果

Choosing the number of cluster

What is the right value of k ?

因為 unsupervised learning 沒有標籤，每個人在看要分成幾個 cluster 時，都有會有不同
的想法

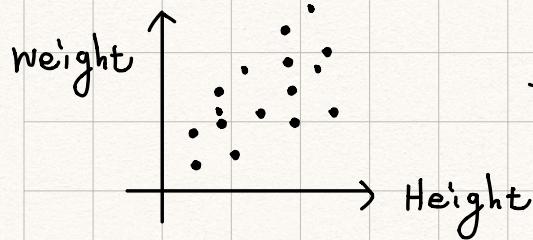
Elbow method



Choosing the value of K

依照人所想的目的去挑選 K 值

比如以 T-shirt 製造來說，在得到



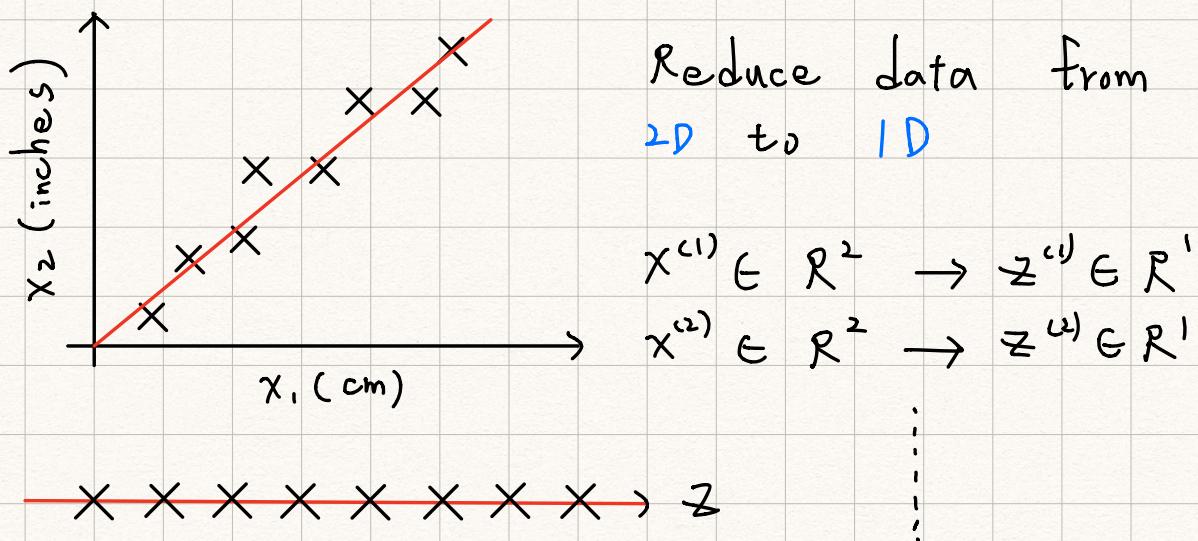
如要分成 S. M. L \rightarrow 3 clusters

X S. S. M. L. X L \rightarrow 5 clusters

Dimensionality Reduction

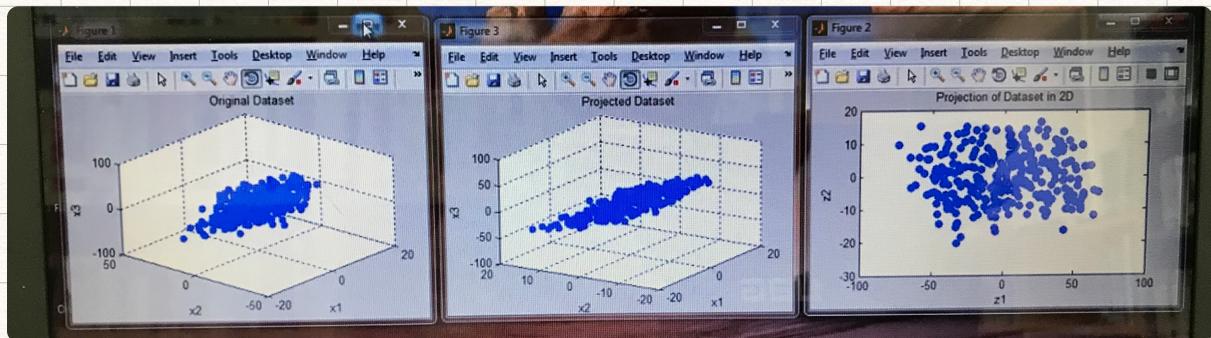
數據壓縮使計算速度上升並減少數據大小
將相似度高的數據壓縮

Data Compression

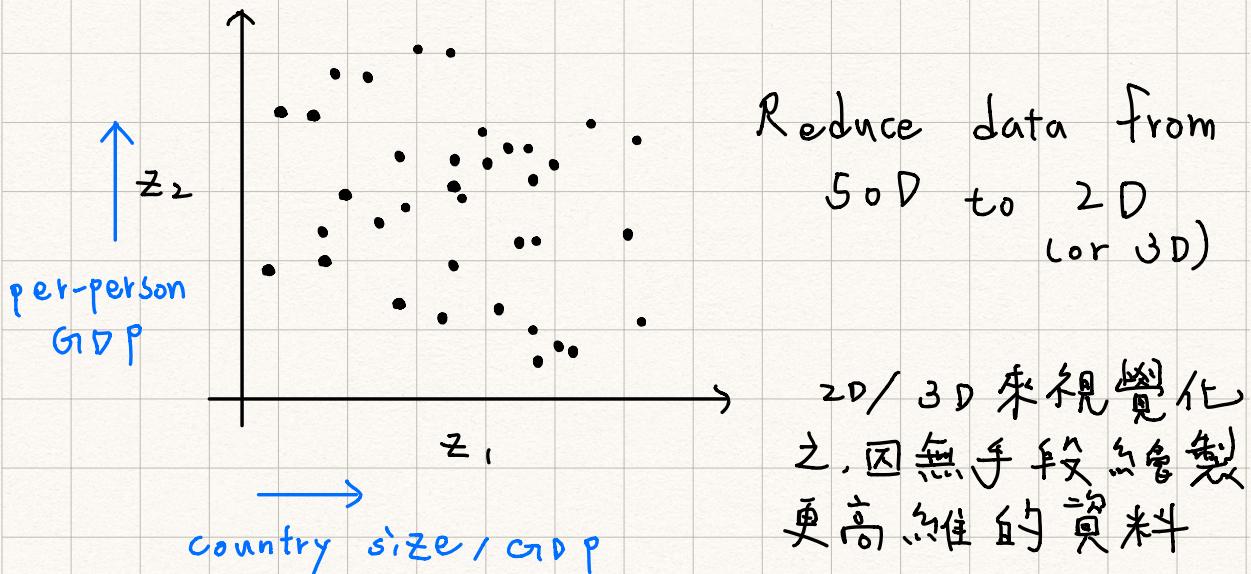


將二維的數據投影到一條線上

3D 轉 2D：將三維數據投影到二維平面



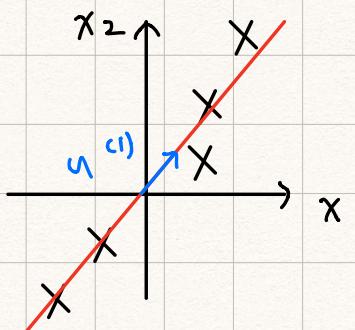
Data Visualization



Principal Component Analysis (PCA)

最常用的數據降維演算
(主成分分析)

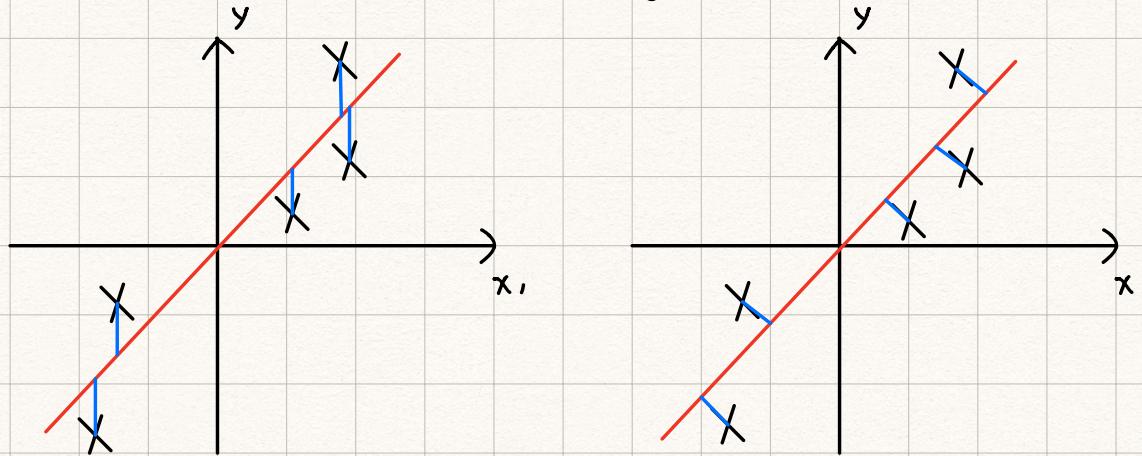
PCA problem formulation



做 PCA 前，須先做
feature-scaling & normalization.

Reduce from n-D to k-D:
Find k vectors $u^{(1)}, u^{(2)}, \dots, u^{(k)}$
onto which to project the data,
so as to min the projection error.

PCA is not linear regression



Linear regression

PCA

最小化的是垂直於 x_1 軸方向的藍線長度之平方和
(有要預測的值 y)

最小化的是垂直於紅色線段的正切距離之平方和
(所有特徵被平等對待)

PCA Algorithm

Data processing

1. mean normalization : $M_j = \frac{1}{m} \sum_{i=1}^m x_j^{(i)}$

Replace each $x_j^{(i)}$ with $x_j^{(i)} - M_j$

2. feature scaling : $\frac{x_j^{(i)} - M_j}{S_j} = x_j^{(i)}$

使每個 feature 的平均值等於零

PCA Algorithm

找到特定的向量、並且計算出之

purpose Reduce data from n-dimension to k-dimension

Compute "covariance matrix"

$$\Sigma = \frac{1}{m} \sum_{i=1}^n (\mathbf{x}^{(i)}) (\mathbf{x}^{(i)})^T = \text{sigma}$$

Compute "eigenvectors" of matrix Σ

$$(\mathbf{U}, \mathbf{S}, \mathbf{V}) = \text{svd}(\text{Sigma}) \quad \mathbf{U} = \begin{bmatrix} \vdots & \vdots & \vdots & \cdots & \vdots \\ u^{(1)} & u^{(2)} & & \cdots & u^{(n)} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \end{bmatrix} \in \mathbb{R}^{n \times n}$$

$$\mathbf{Z} = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots \end{bmatrix}^T \cdot \mathbf{x}^{(i)} \quad \text{nx1} \quad \rightarrow k \times 1 \text{ 矩陣}$$

$$\mathbf{U}_{\text{reduce}} = \mathbf{U}(:, 1:k)$$

$$\mathbf{Z} = \mathbf{U}_{\text{reduce}} * \mathbf{x} \quad \mathbf{x} \in \mathbb{R}^n, \quad \mathbf{x}_0 = 1$$

/

Applying PCA

Reconstruction from compressed representation

將壓縮後的 data 轉回壓縮前的 data ?

$$x \rightarrow z, \quad z = (U_{\text{reduce}})^T \cdot x$$

$$\left[\begin{array}{c} \approx x \\ X_{\text{approx}} = U_{\text{reduce}} \cdot z \\ \in \mathbb{R}^n \end{array} \right] \quad \begin{array}{c} n \times k \\ K \times 1 \\ n \times 1 \end{array}$$

Choosing the Number of Principle Components

Average squared projection error

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{\text{approx}}^{(i)}\|^2$$

PCA 就是最小化此值

Total variation in the data

$$\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2$$

資料離原點的平均值

Typically, choose k to be smallest value so that

$$\frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{\text{approx}}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2} \leq 0.01 (1\%)$$

"99% of variance is retained"

95~99都算常用

具骨量步馬眾

Algorithm

- ① try PCA with $k=1$
 \int
 $K=n$

- ② Compute U, Z

- ③ Check if

$$\frac{\frac{1}{m} \sum_{i=1}^m \|x^{(i)} - x_{\text{approx}}^{(i)}\|^2}{\frac{1}{m} \sum_{i=1}^m \|x^{(i)}\|^2} \leq 0.01$$

$$\{U, S, V\} = \text{svd}(Sigma)$$

$$S = \begin{pmatrix} S_{11} & & & \\ & S_{22} & & \text{Zero} \\ & & \ddots & \\ \text{Zero} & & & S_{nn} \end{pmatrix}$$

For given k :

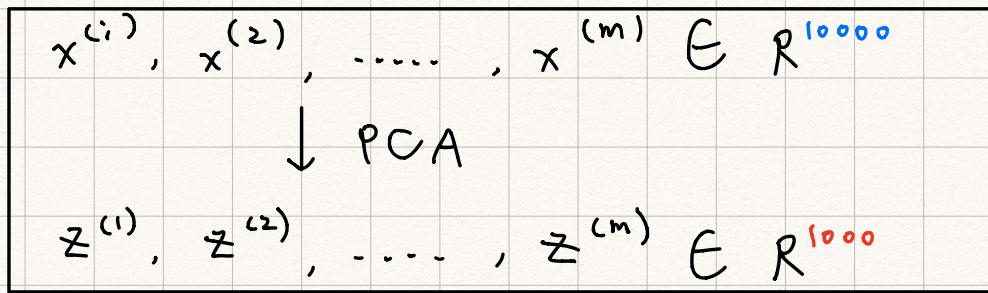
$$1 - \frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} \leq 0.01$$

$$\text{or } \frac{\sum_{i=1}^k S_{ii}}{\sum_{i=1}^n S_{ii}} \geq 0.99$$

Advice for Applying PCA

Supervised learning speedup

$$(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \dots \dots (x^{(m)}, y^{(m)})$$



$$(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \dots, (z^{(m)}, y^{(m)})$$

Note: 應用PCA時，只能將之用在訓練集上，找出最優 Ureduce 後，再將之用在驗證集 / 測試集

Application of PCA

- Compression: Reduce memory / Speed up
- Visualization ($k=2$ or 3)

Bad use of PCA: To prevent overfitting

This might work OK, but isn't a good way to address overfitting. 使用PCA會使得少 ---

但有更好的方式實行之, regularization !

$$\min_{\theta} \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2m} \sum_{j=1}^n \theta_j^2$$

PCA is sometimes used where it shouldn't be

Design of ML system:

- Get training set
- Run PCA
- Train logistic regression
- Test on test set: Map $x_{\text{test}}^{(i)}$ to $z_{\text{test}}^{(i)}$.
Run $h_{\theta}(z)$ on $\{(z_{\text{test}}^{(1)}, y_{\text{test}}^{(1)}), \dots, (z_{\text{test}}^{(m)}, y_{\text{test}}^{(m)})\}$

How about doing the whole thing without using PCA

Before implementing PCA. first try running whatever you want to do with the original / raw data $x^{(i)}$. Only if that doesn't do what you want, then implement PCA and consider using $z^{(i)}$.