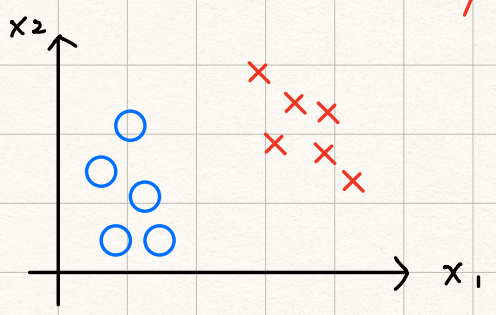
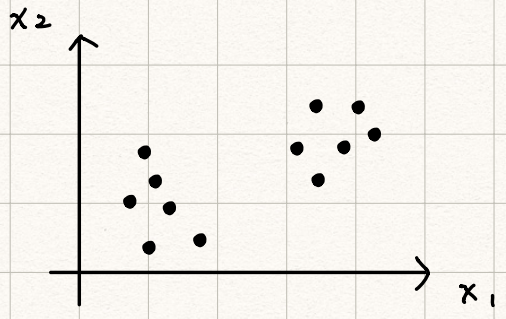
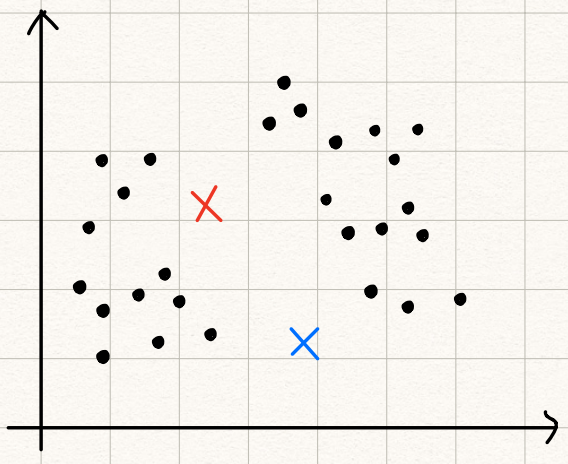
Clusting

Unsupervised Learning : Introduction

suppervised	unsupervised
find decision boundary  Training set has labels	find structure  Training set without label

K - Means Algorithm



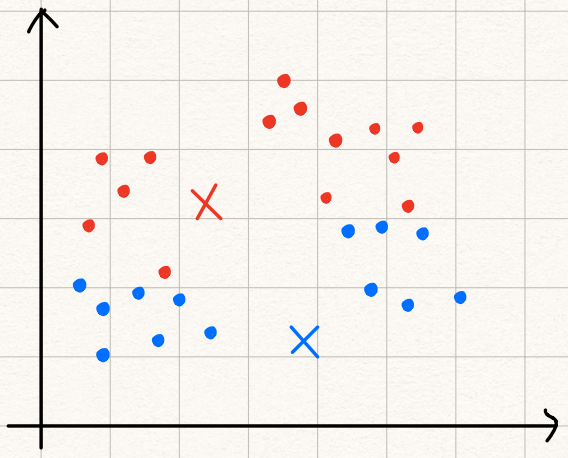
數量由想分幾個group來決定

↓

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

after ①



② move centroid 計算所有紅點和藍點的平均位置，之後把 cluster centroid 移到此平均位置上

重複 ① ② ① ② ① .... 的動作，便能完成 k-means

Steps of K-mens 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 $\mu_1, \mu_2, \mu_3 \dots \mu_K \in R^n$

Step 已: for i = 1 to m ← 資料樣本數 m
$c^{(i)}$:= index (from 1 to k) of cluster centroid closer to $x^{(i)}$

↑ cluster assignment

$\min_k \|x^{(i)} - \mu_k\|^2$，$x^{(1)}$，$x^{(3)}$，$x^{(6)}$ 近 μ_2 則 $c^{(1)} = 2$，$c^{(3)} = 2$，$c^{(6)} = 2$

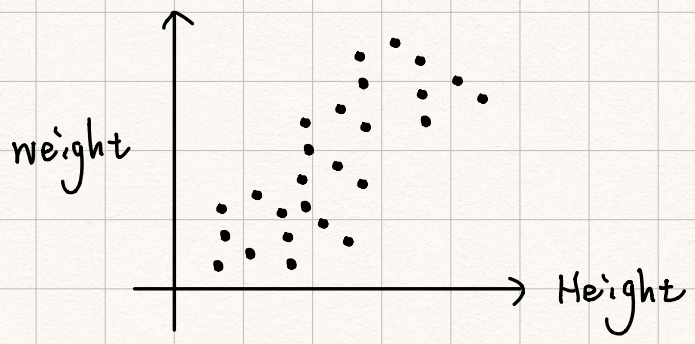
Step 3: for k = 1 to K

μ_k := average (mean of points assigned to cluster K)

move centroid

$$\mu_2 = \frac{1}{3}\left(x^{(1)} + x^{(3)} + x^{(6)}\right)$$

K-means for non-separated clusters



K-means 算法仍然可以將左圖分成一個個 cluster

K-means optimization objective

$c^{(i)}$ = $c^{(i)}$ 表示 x_i 屬於哪個 cluster

μ_k = μ_k 為第 k 個 cluster centroid

$\mu_{c^{(i)}}$ = $\mu_{c^{(i)}}$ 表示 x_i 的 cluster centroid

$x^{(i)} = 5 \rightarrow c^{(i)} = 5 \rightarrow \mu_{c^{(i)}} = \mu_5$

Optimization objective

$J(c^{(1)}, \dots c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$

找到 $c^{(1)} \sim c^{(m)}$, $\mu_1 \sim \mu_K$ 使得 $J(\ )$ 最小

Random initialization

should have $K < m$.

Randomly pick K training examples.

Set $\mu_1, \dots, \mu_K$ equal to these K examples.

$\mu_1 = x^{(i)}$, $\mu_2 = x^{(j)}$

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 $J(\ )$ 記下來

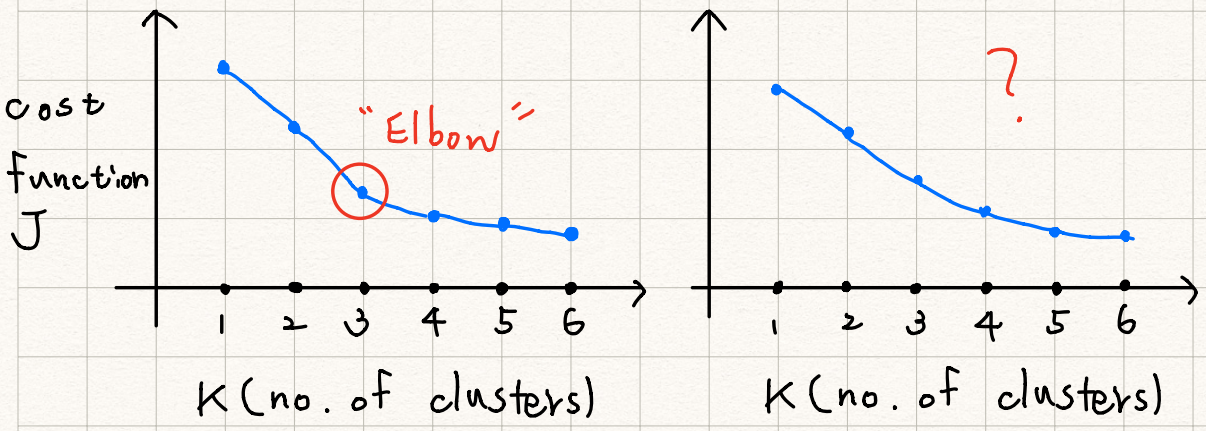
最後，從這 100 個 $J(\ )$ 中找最小的做為結果

Choosing the number of cluster

What is the right value of K?

因為 unsupervising learning 沒有標籤，每個人在看要分成幾個 cluster 時，都會有不同的想法

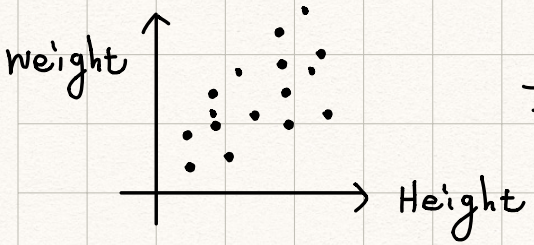
Elbow method



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 → 3 clusters

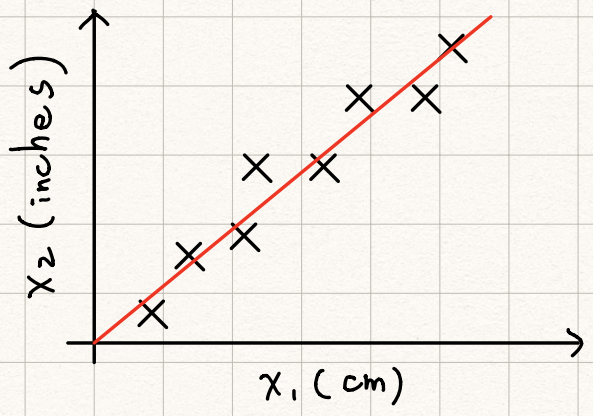
XS. S. M. L. XL → 5 clusters

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



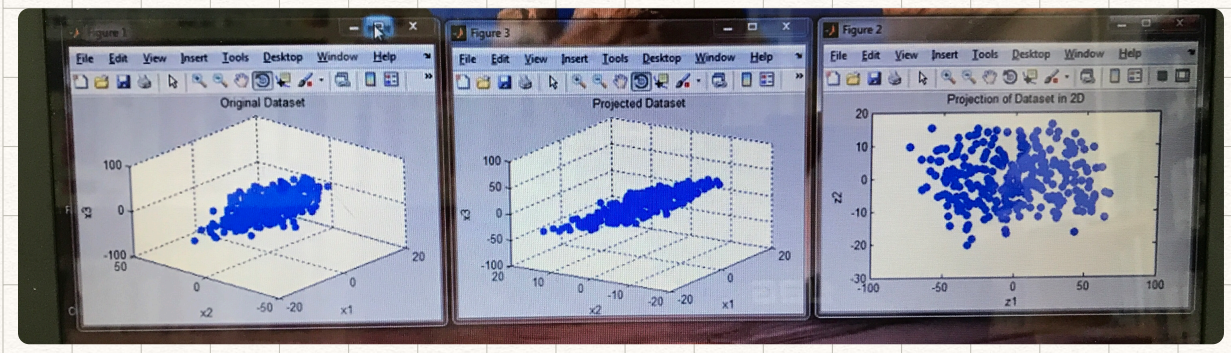
Reduce data from 2D to 1D

$x^{(1)} \in R^2 \rightarrow z^{(1)} \in R^1$

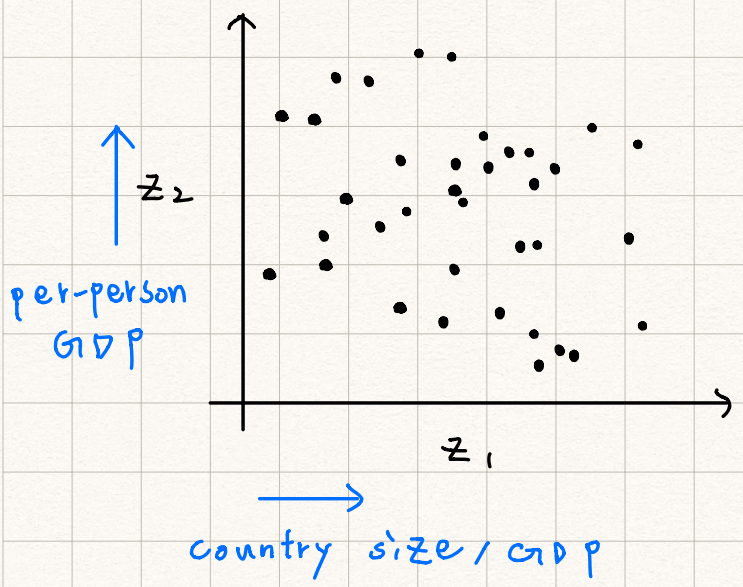
$x^{(2)} \in R^2 \rightarrow z^{(2)} \in R^1$

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



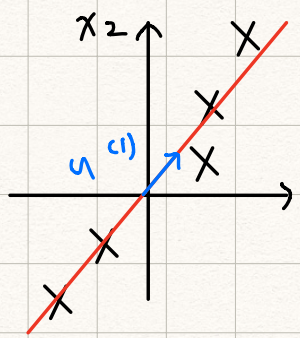
Reduce data from 50D to 2D (or 3D)

2D/3D 來視覺化之，因無手段繪製更高維的資料

Principal Component Analysis (PCA)

最常用的數據降維演算化
(主成分分析)

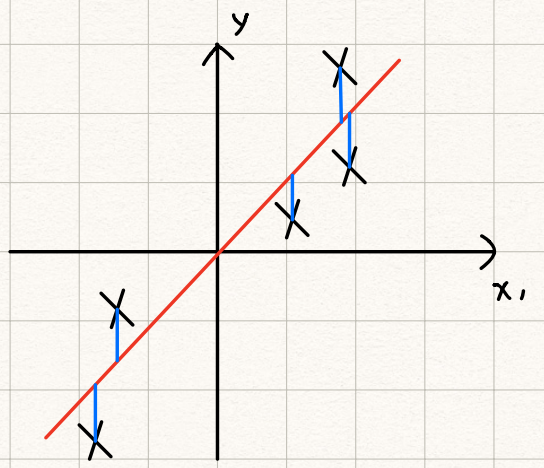
PCA problem formulation



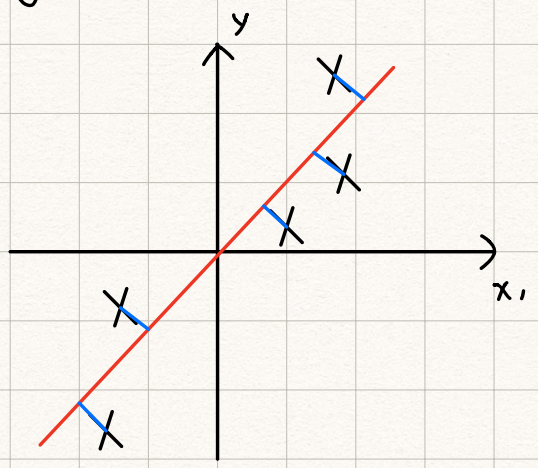
做PCA前，須先做 feature-scaling、normalization.

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X軸方向的藍線長度之平方和（有要預測的值y）

最小化的是垂直紅色線段的距離值之平方和（所有特徵被平等對待）

PCA Algorithm

Data processing

使每個feature的平均值等於零

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

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

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

PCA Algorithm

找到特定的向量、並且計算出Z

purpose Reduce data from n-dimension to k-dimension

Compute "covariance matrix"

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

Compute "eigenvectors" of matrix Σ

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

n×n matrix

k

$$z = \begin{bmatrix} \vdots & \vdots & & \vdots \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ \vdots & \vdots & & \vdots \end{bmatrix}^T \cdot x^{(i)}$$

$(n\times k)^T$ $\quad$ $n\times 1$ $\longrightarrow$ $k\times 1$ 矩陣

$$U_{reduce} = U(:, 1:k)$$

$$z = U_{reduce}' * x \qquad x\in\mathbb{R}^n,\ x_0 = 1$$

Applying PCA

Reconstruction from compressed repression

將壓縮後的 data 轉回壓縮前的 data ?

z_1 , $z = (U_{reduce})^T \cdot x$

$$x_{approx} = U_{reduce} \cdot z$$

$x_{approx} \in R^n$; U_{reduce}: $n \times k$; z: $k \times 1$; x_{approx}: $n \times 1$

Choosing the Number of Principle Components

Average squared projection error

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

PCA 就是最小化此值

Total variation in the data

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

資料離原點的平均值

Typically, choose k to be smallest value so that

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

"99% of variance is retained"

95 ~ 99 都算常用

具體步驟如下

Algorithm

① try PCA with $k=1$

② Compute U, Z

③ Check if

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

$\int_{k=n}$

$[U, S, V] = svd(Sigma)$

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

For given k:

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

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

Advice for Applying PCA

Supervised learning speedup

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

$x^{(1)}, x^{(2)}, \cdots\cdots, x^{(m)} \in R^{10000}$

↓ PCA

$z^{(1)}, z^{(2)}, \cdots\cdots, z^{(m)} \in R^{1000}$

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

Note: 應用PCA時，只能將之用在訓練集上，找出最優 U reduce 後，再將之用在驗証集／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會將特徵減少---

但有更好的方式實行之，regularization !!

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

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 $x_{test}^{(i)}$ to $z_{test}^{(i)}$.
  Run $h_\theta(z)$ on $\{(z_{test}^{(1)}, y_{test}^{(1)}), \ldots (z_{test}^{(m)}, y_{test}^{(m)})\}$

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 $z^{(i)}$.