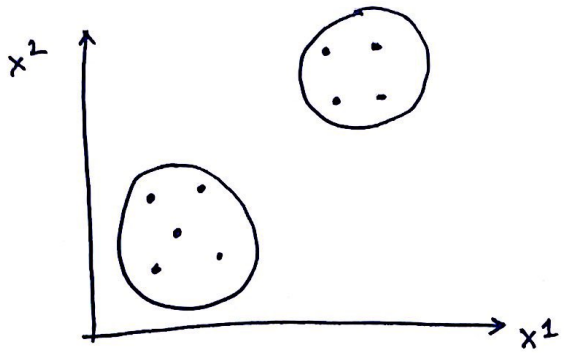
WEEK-8

UNSUPERVISED LEARNING

* Clustering : don't have any labels.

Training set : $\{x^1, x^2, \ldots x^m\}$ without labels y^i



* K-means Algorithm

① • cluster Assignment
② • Move centroid
} Iterative.

Inputs :

- K (number of clusters)
- Training set $\{x^1, x^2, \ldots x^m\}$
  ↳ $x^i \in \mathbb{R}^n$ {drop $x_0 = 1$ convention}

Algorithm :

- Randomly initialize K cluster centroids

$$\mu^1, \mu^2, \ldots, \mu^K \in \mathbb{R}^n$$

- Repeat

{

cluster assignment step

for $i = 1 : m$

$c^{(i)}$:= index (from 1 to K) of cluster centroid closet to $x^{(i)}$

$$c^i = \min_K \|x^i - \mu_K\|$$

eg. $c^{(1)} = 3$, $c^{(2)} = 3$ } cluster 3

$c^{(3)} = 5$ } cluster 5.

move centroid step

for $k = 1 : K$

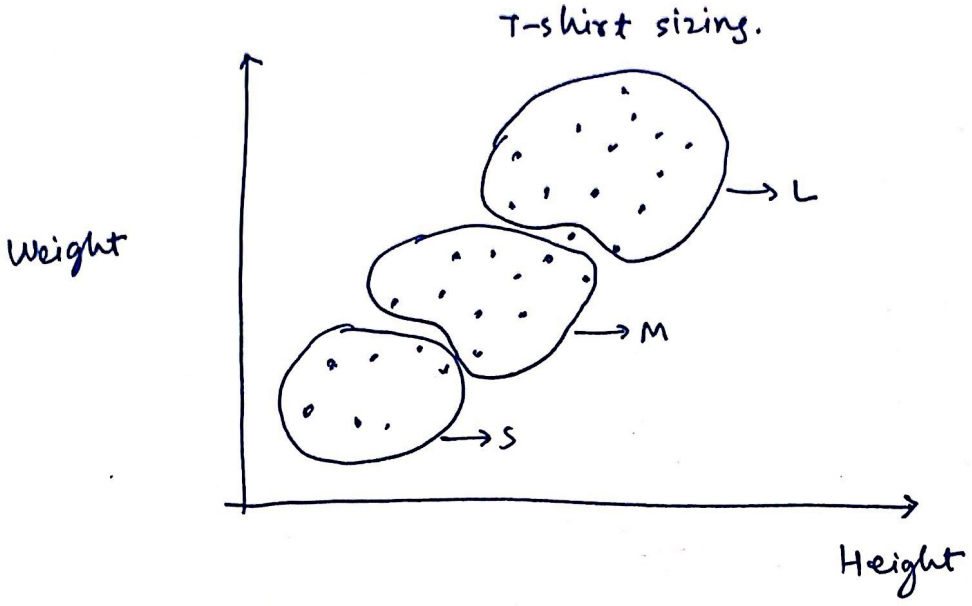
μ_k := average(mean) of points assigned to cluster K

eg. $x^{(1)}, x^{(5)}, x^6, x^{10} \Rightarrow$ $c^1 = 2$, $c^5 = 2$, $c^6 = 2$, $c^{10} = 2$

$\therefore \mu_2 = \frac{1}{4}[x^1 + x^5 + x^6 + x^{10}] \in \mathbb{R}^n$

}

⊛ K-means for non-seperated clusters.



Optimization Objective : K-means

c^i : index of cluster to which x^i is assigned.

μ_k : cluster centroid

μ_{c^i} = cluster centroid of cluster i to which x^i has been assigned

$x^i \Rightarrow 5$

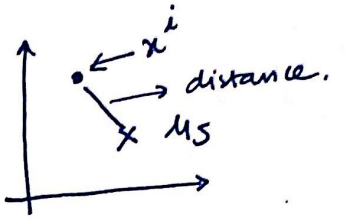
$c^i = 5.$

$\mu_5 \Leftarrow \mu_{c^i}$ •

Optimization Objective : Distortion fn.

$$\left[J(c^1, c^2, \ldots c^m, \mu_1, \mu_2 \ldots \mu_K) = \frac{1}{m} \cdot \sum_{i=1}^{m} \| x^i - \mu_{c^i} \|^2 \right]$$

$$\min_{\substack{c^1, c^2, \ldots c^m \\ \mu_1, \mu_2 \ldots \mu_K}} J(c^1, c^2 \ldots c^m, \mu_1 \ldots \mu_K)$$



↕ cluster assignment step: minimize $J(\ldots)$ with resp. to. $c^1, c^2 \ldots c^{(m)}$ holding $(\mu_1, \mu_2 \ldots \mu_K)$ fixed

↕ Move centroid step: min. $J(c \ldots)$ w.r.t $\mu_1, \mu_2 \ldots \mu_K$ holding $c^1, c^2, \ldots c^{(m)}$ fixed

Random Initialization

(1) should have $K < m$

(2) Randomly pick K training examples

(3) Set $\mu_1 \ldots \mu_K$ equal to these K examples

for $i = 1 : 100$

{

① Randomly initialize K-means.

② Run K-means., Get c^i, μ_K for all

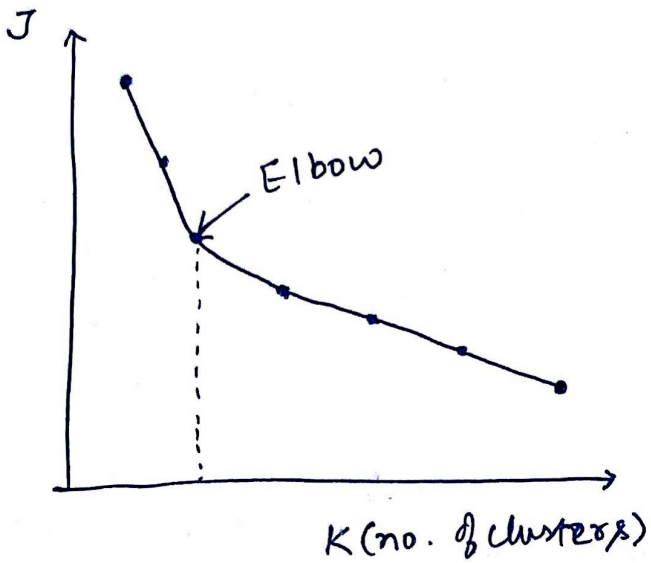
③ Compute $J(\ldots)_{c,\mu}$ → cost/ distortion fn.

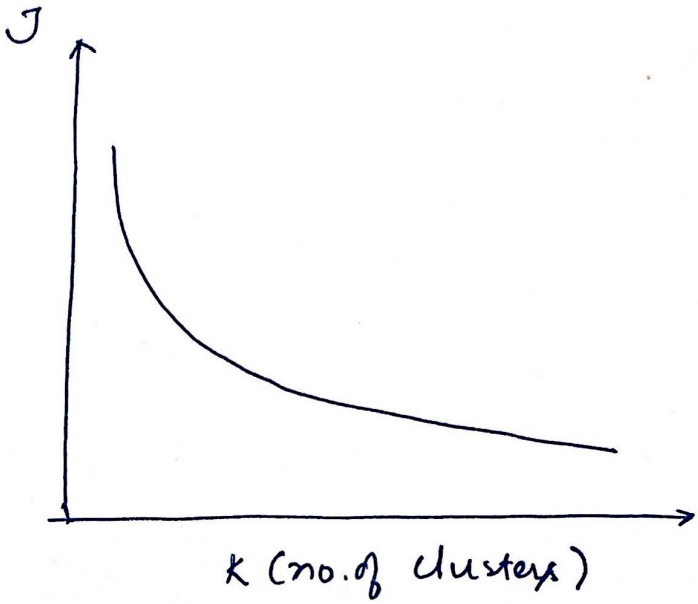
}

- pick clustering that gave lowest cost $J(c^1, c^2 \ldots c^m, \mu_1, \ldots \mu_K)$

Choosing the Number of clusters

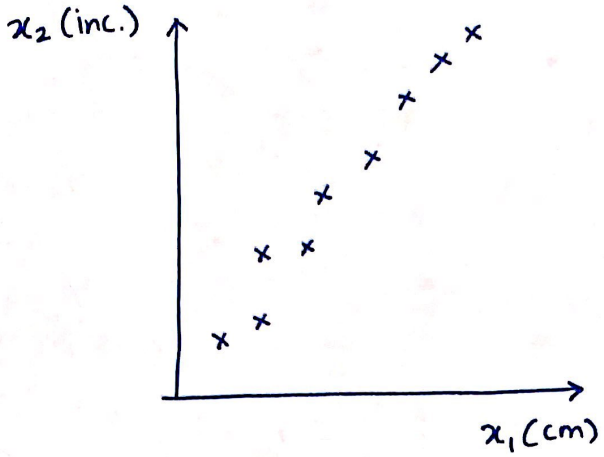
Elbow Method.





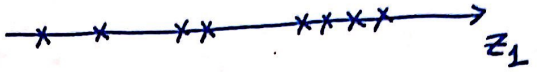
J goes ~~out~~ low rapidly till Elbow.

Dimensionality Reduction : Data Compression



$x^i \in \mathbb{R}^2$

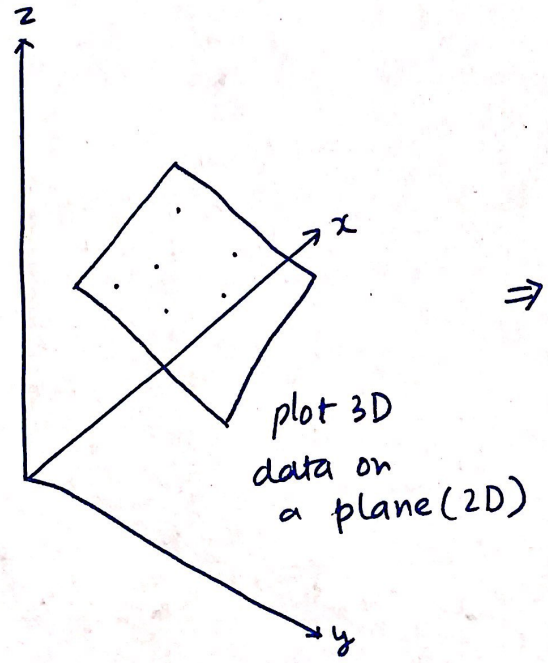
Reduce data from 2D to 1D



$x^i \in \mathbb{R}^1$

* approximation of the training set

Reduce data from 3D to 2D



$x^i \in \mathbb{R}^3$

$z^i \in \mathbb{R}^2$

$$z^i = \begin{bmatrix} z_1^i \\ z_2^i \end{bmatrix}$$

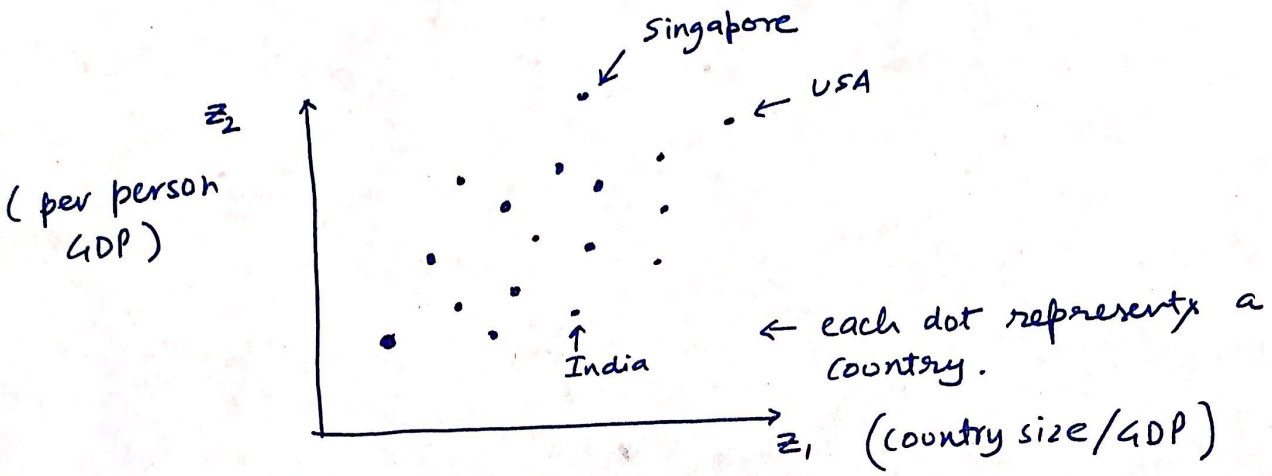
Dimensionality Reduction: Data Visualization

$x_i = \{$ GDP, per capita GDP, HDI, Life exp., Gini coeff, ... $\}$

$x^i \in \mathbb{R}^{50}$

⇓ Reduce from 50D to 2D

$z^i = \{ z_1, z_2 \}$ $\quad z^i \in \mathbb{R}^2$

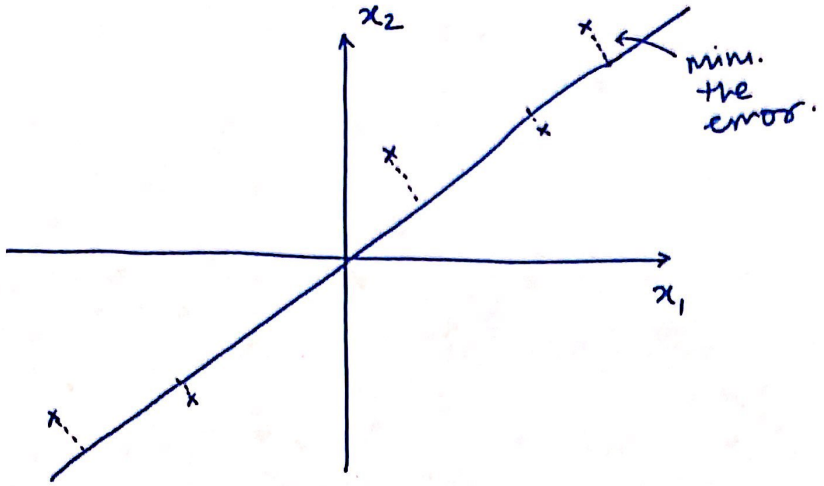


Suppose, I have a data $x^i \in \mathbb{R}^n$. In order to visualize it, we apply dimensionality reduction & get $z^i \in \mathbb{R}^K$ where $z^i \in \mathbb{R}^K$.

① $K \leq n$

② $K = 2$ or 3, because don't have ways to visualize higher dimension data.

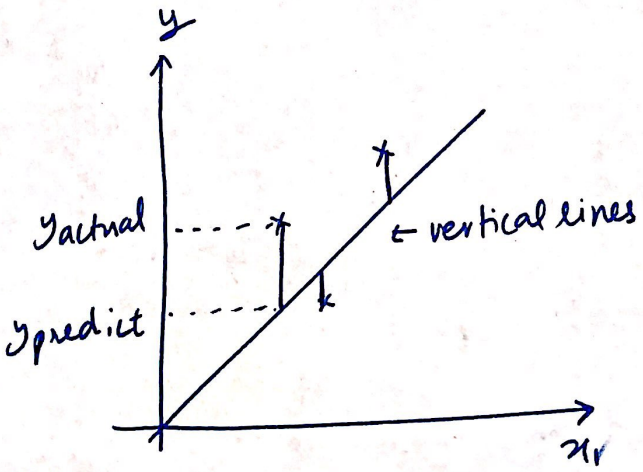
Principal Component Analysis (PCA)

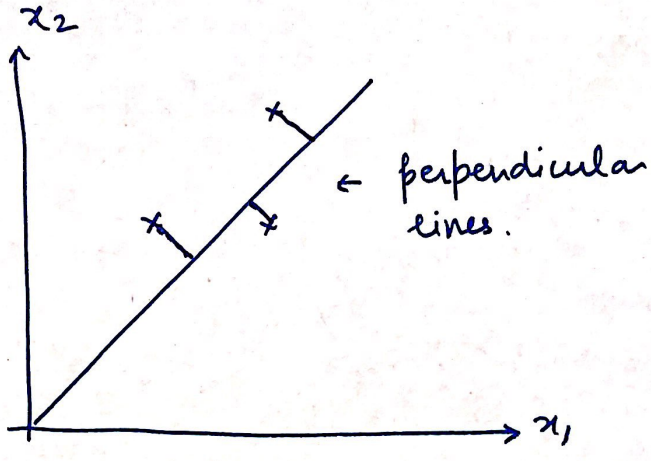


- Reduce from 2-dimension to 1-dimension : Find a direction (a vector $u^1 \in \mathbb{R}^n$) onto which to project the data to minimize the projection error

- Reduce from n-dimensional to K-dimension : Find K vectors ($u^1, u^2 \ldots u^K$) onto which to project the data so as to minimize the projection error

PCA is not linear regression





PCA Algorithm :

⊛ Data preprocessing (Feature scaling / mean normalization)

Training set: $x^1, x^2 \dots x^m$

Step 0 :

① Compute mean, μ_j for each feature.

$$\therefore \mu_j = \frac{1}{m} \sum_{i=1}^{m} x_j^i$$

② Replace x_j^i with $x_j^i - \mu_j$

⊛ PCA : Reduce data from n-dimension to K-dimension

Step 1 : Covariance Matrix

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} \underbrace{(x^i)}_{n\times 1} \underbrace{(x^i)^T}_{1\times n}$$

(sigma)

Step 2 : Compute "eigenvectors" of matrix Σ

[U, S, V] = svd (Sigma) ↳ nxn

(Octave command)

% svd : ~~singular~~ singular value decomposition

$$U = \underbrace{\begin{bmatrix} | & | & & | \\ u^1 & u^2 & \dots & u^m \\ | & | & & | \end{bmatrix}}_{K}$$

$U \in \mathbb{R}^{n\times n}$

↳ Take k vectors.

$x \in \mathbb{R}^n \longrightarrow z \in \mathbb{R}^K$

Octave

```
Ureduce = U(:,1:K);
z= Ureduce' * x;
```

Step 3 :

$$z^K = \underbrace{\begin{bmatrix} | & | & & | \\ u^1 & u^2 & \dots & u^K \\ | & | & & | \end{bmatrix}^T}_{U_{reduce}} * X$$

$n \times K$

$\hookrightarrow (K \times n) \circledast (n \times 1)$

$K \times 1$

$$\therefore \quad z^K \rightarrow \begin{bmatrix} z_1 \\ z_2 \\ \vdots \end{bmatrix}_{K \times 1}$$

Reconstruction from compressed representation

$z^K = (U_{reduce})_K' * X$: compression

$\Downarrow$

$X_{approx} \underbrace{}_{\mathbb{R}^{n \times 1}} = \underbrace{U_{reduce}}_{n \times K} * \underbrace{z}_{K \times 1}$: De-compression

choosing the number of principal components.

k for PCA

→ Avg. squared projection error : $\frac{1}{m}\sum_{i=1}^{m} \|x^i - x^i_{approx}\|^2$

→ Total Variation in the data : $\frac{1}{m}\sum_{i=1}^{m} \|x^i\|^2$

→ Typically, choose k to be smallest value so that

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

"99% variance is retained"

(ratio ≤ 0.05, means 95% is retained)

Algorithm: (inefficient)

step 1: Try PCA with K=1

step 2: Compute U_{reduce}, z^i, x^i_{approx}

step 3: compute & check if

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

} iterate until step 3 is satisfied.

[U, S, V] = svd (sigma)

$$S = \begin{bmatrix} S_{11} & & 0 \\ & S_{22} & \\ & & \ddots \\ 0 & & S_{nn} \end{bmatrix}_{n \times n}$$

square matrix of nxn size

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \| x^i - x_{approx}^i \|^2}{\frac{1}{m}\sum_{i=1}^{m} \| x^i \|^2} \Rightarrow 1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}}$$

$\therefore$ [redacted]

check $\dfrac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \geq 0.99$ for a given k. } iterate

$\Downarrow$

pick smallest value of k. **

Advice for applying PCA

→ Supervised learning speedup

$(x^1, y^1)(x^2, y^2) \ldots (x^m, y^m)$

$x^i \in \mathbb{R}^n$ [where n is large]

eg: computer vision 100 x 100 pixel image

$n = 10,000$

- Extract inputs:

unlabeled inputs dataset:

$x^1, x^2 \ldots x^m \in \mathbb{R}^{10,000}$ → PCA → $z^1, z^2 \ldots z^m \in \mathbb{R}^{1000}$

$x^{test} \rightarrow z^{test}$

- New Training set:

$(z^1, y^1)(z^2, y^2) \ldots (z^m, y^m)$

$h_\theta(x) = \frac{1}{1+e^{-\theta^T z}}$

** Note: Mapping $x^i \rightarrow z^i$ should be defined by running PCA only on the training set (not on x^i_{cv} or x^i_{test})

Application of PCA

- compression
  - Reduce memory / disk needed to store data
  - Speed up learning algorithm. ** (good!)

  → choose K by % of variance retain

- Visualization

  → choose K=2 or K=3

Bad use of PCA: To prevent overfitting

✓ use regularization instead

× less features, less likely to overfit (! Bad)
{does not look at y^i} thus throwing away some useful information

Note: Before implementing PCA, first 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 & consider using (z^i)

Octave application

K- means Algorithm :

if $X \in [m \times n]$

then,

$K \in [k \times n]$

} if x^i has n- dimensions then μ_K will also have n- dimensions.

$\|X - \mu\| = \text{sqrt}\left[\text{sum}\{(\text{power}(X - \mu), 2)\}\right];$