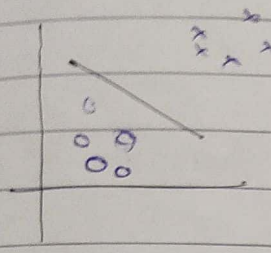
Week 8

27/8/18
Pgno. 125

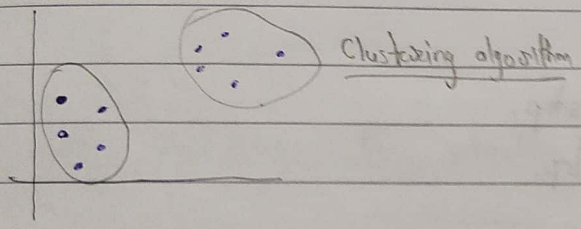
ⓐ Unsupervised learning Introduction

- Supervised learning



Training set → $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots (x^{(m)}, y^{(m)})\}$

- Unsupervised learning

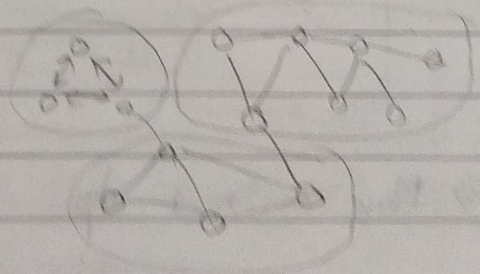


In unsupervised learning what we do is we give this sort of unlabbled training set to an algorithm and we just ask the algorithm to find some structure for us.

Training set → $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots x^{(m)}\}$

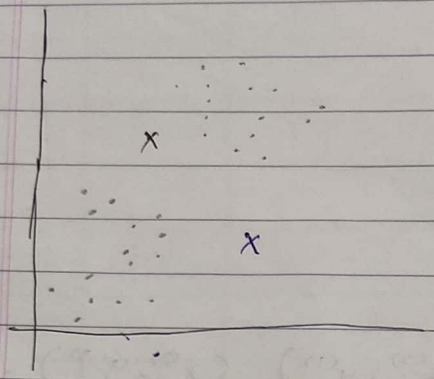
- Applications of clustering

- market segmentation
- Social network analysis.
- organise computing clusters.
- Astronomical data analysis.



⑥ K-Means Algorithm

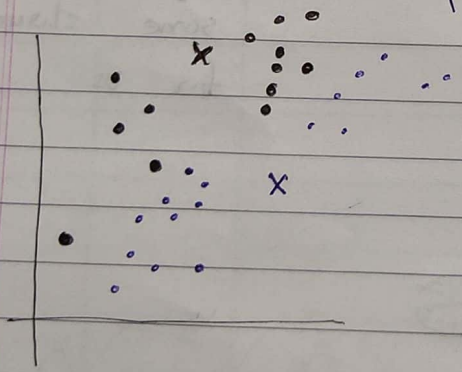
A clustering (Iterative) algo.



S1) Randomly initilize two clustroids. centroid.

2 steps →
- cluster assignment step.
- Move centroid step.

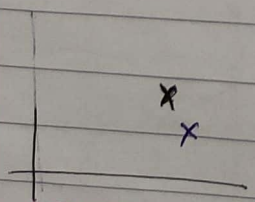
S2) Cluster assignment step.



colour each depending its closer to blue or black cluster centroid.

S3) Move centroid step.

Move centroid to average of points coloured the same colour.



S4) (Repeat S2, S3) → Repeat.

What if cluster centroid is assigned with zero points.

- Eliminate that cluster centroid
- K-1 clusters

(common)

- Randomly reinitialize that cluster centroid
- K clusters.

After many iterations, centroids will not change, colours of points will not change.

K-Means converged.

☆ K-means algorithm.

Input:

- K (no of clusters)
- Training set $\{x^{(1)}, x^{(2)}, \dots x^{(m)}\}$

$x^{(i)} \in \mathbb{R}^n$ ~~$\mathbb{R}^{n+1}$~~ ~~$x_0 = 1$~~ ← we will not use this so (drop $x_0 = 1$ convention)

Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

✗ μ_1 ✗ μ_2

Repeat {

Cluster assignment step

for $i = 1$ to m

$c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$ — (number b/w 1 to K)

[or] $\min_k \|x^{(i)} - \mu_k\|$ → $= c^{(i)}$

same

[or] Here pick $c^{(i)}$ = cluster centroid with smallest squared distance to training e.g. $x^{(i)}$

(but we use this) → $\min_k \|x^{(i)} - \mu_k\|^2$

Move centroid step

for $k = 1$ to K — for each cluster centroid

μ_k := average of points assigned to cluster k

$x^{(1)}, x^{(5)}, x^{(6)}, x^{(10)}$

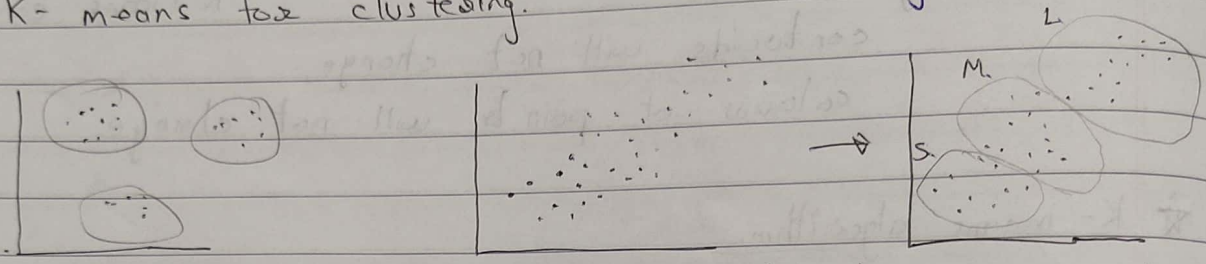
$c^{(1)} = 2, c^{(5)} = 2, c^{(6)} = 2, c^{(10)} = 2$

This e.g. were assigned to cluster centroid 2.

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

★ K- means for clustering.

T shirt sizing



Thats how k- means works
(Market segmentation like)

Optimization Objective — — —

upto now each algo had optimization objective / cost fun which we were trying to minimize..

K-means also has cost fun.

cost fun Useful for →

i) To debug learning algo. (running right/wrong)

ii) How k-means avoid local mining by finding better cost fun.

notations

$c^{(i)}$ = index of cluster $(1, 2, \ldots, K)$ to which example $x^{(i)}$ is currently assigned.

μ_k = cluster centroid k $(\mu_k \in \mathbb{R}^n)$ $\quad k \in \{1, 2, \ldots, K\}$

$\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned.

$x^{(i)} \rightarrow 5$ $\quad x^{(i)}$ is assigned to cluster no. 5.

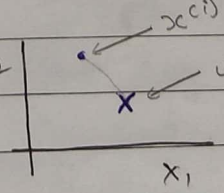
$c^{(i)} = 5$

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

• Optimization objective.

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

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



Distortion cost function /

Distortion of k-means algo.

☆ K-means algo.

Randomly initialize K cluster centroids $\mu_1, \mu_2, \dots \mu_K \in \mathbb{R}^n$

Repeat {

for i = 1 to m

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

- Cluster assignment step
- minimize $J(\dots)$ wrt $c^{(1)}, c^{(2)}, \dots c^{(m)}$ (holding $\mu_1, \mu_2, \dots, \mu_K$ fixed)

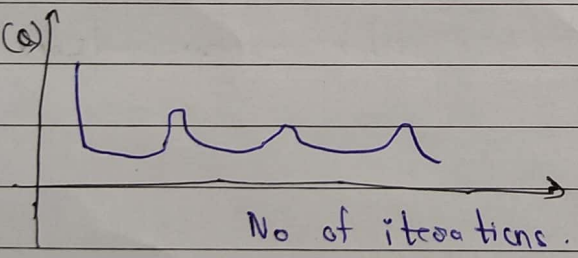
for k = 1 to K

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

- Move centroid.
- minimize $J(\dots)$ wrt $\mu_1, \dots, \mu_K$

}

creates 2 sets $c^{(i)}$'s & μ_k's & works

Q) 

→ It is not possible for the cost fun to some times increase. There must be bug in the code.

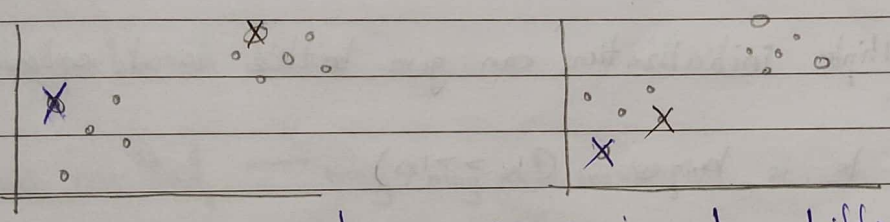
K-means Random Initialization

- K-means initialization (How)
- How to avoid local optimum.

Random initialization.

- Should have $K < m$.
- Randomly pick K training examples.
- Set $\mu_1, \ldots, \mu_K$ equal to these K examples.

e.g.

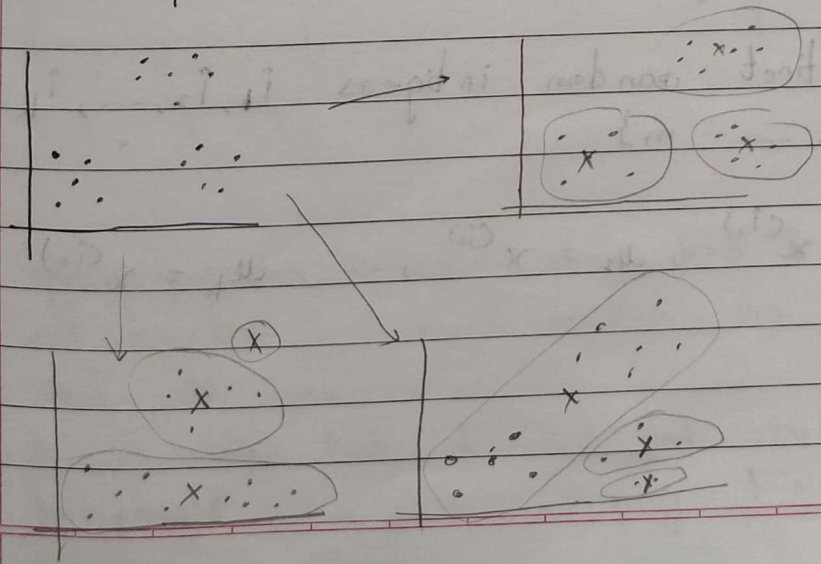


K-means can end up converging to different solutions Depending on exactly how clusters were initialized, (so random initialization)

K-mean can actually end up at local optima.

Local optima.

local optima refers to local optima of distortion fun.



Global optima

To get aid of local optima Try multiple random initialization And try to get as good solution as good.

Random initialization.

for $i = 1$ to 100 → 50-100
{
Randomly initialize k means
Run k-means. Get $c^{(1)}, \ldots, c^{(k)}, \ldots$
Compute cost fn.
$J(c^{(1)}, \ldots, c^{(k)}, \ldots)$
}

Compute for 50-100 times.
Run code for multiple / initialize multiple times.

Pick clustering that gave lowest cost $J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots \mu_k)$

If k is small (k = 2-10)

Multiple initializations can give better local optimum.

If k is large (k >> 10)

Multiple initializations will not make huge difference.
First random initialization will also be close to global optima.

Q11 Recommended way to initialize k-means.

→

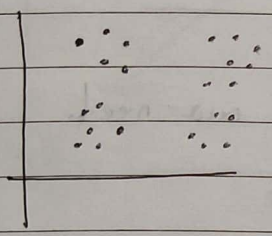
Pick k distinct random integers $i_1, i_2, \ldots, i_k$ from $\{1, \ldots, m\}$

Set $\mu_1 = x^{(i_1)}$, $\mu_2 = x^{(i_2)}$, ..., $\mu_k = x^{(i_k)}$

-- @ Choosing the Number of clusters — — —

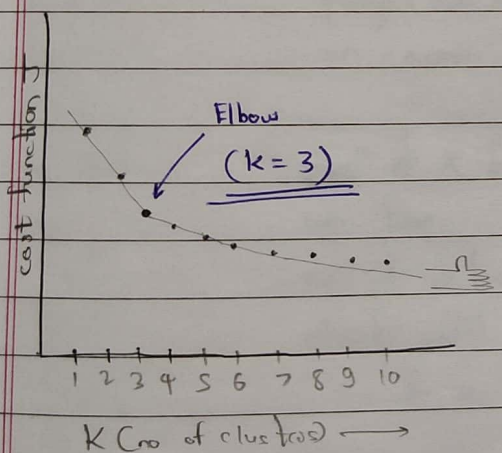
No good method to choose automatically
We have to choose by visualization / o/p of clustering algo manually.
By hand (manually)

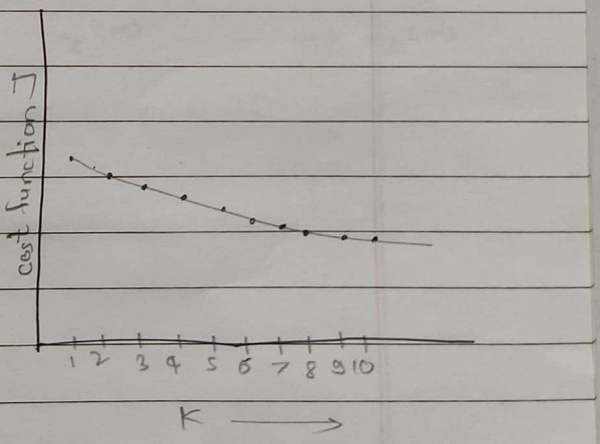
- What is value of k -- here it might be 2/4.



- Choosing value of K.

#Elbow method → may / may not work





Q Suppose you run k-means using k=3 & k=5. You find that the cost function J is much higher for k=5 than for k=3. What can you conclude?

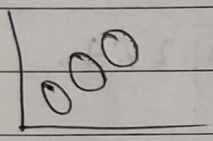
→ In the run k=5, k-means got stuck in bad local minima, You should ~~re~~ retry running k-means with multiple times.

★ Choosing the value of k.

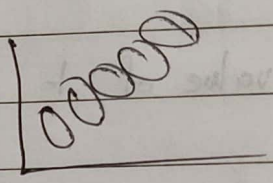
Sometimes, you are running k-means to get clusters to use for some later/downstream purpose. Evaluate k-means based on a metric for how well it performs for that later purpose.

sizing.

K=3



K=5



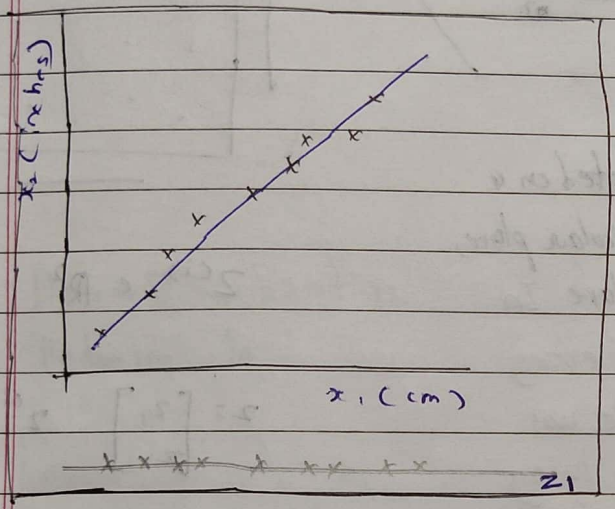
Take value of k, according to our need.

optimization, To train faster.

Motivation I: Data Compression

Dimensionality reduction

Out of 100, let say x_1 ... x_2 are features (redundant features) & we want to reduce features, to train faster.



Reduce dat from 2D to 1D
Some how using only one feature, $z^{(2)}$ which has feature of x_1, x_2 both.

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

$x^{(2)} \in \mathbb{R}^2 \longrightarrow z^{(2)} \in \mathbb{R}$

$x^{(m)} \longrightarrow z^{(m)}$

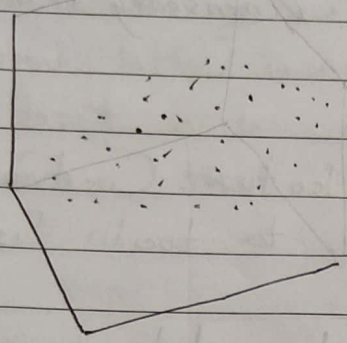
⟶ a line (best fit)

⟶ projection of x on green line

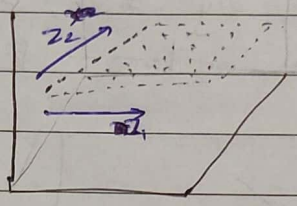
z_1 ⟶ pos^n of x ~~form~~ on line, so, it is feature which will replace x_1 & x_2

• Data compression.

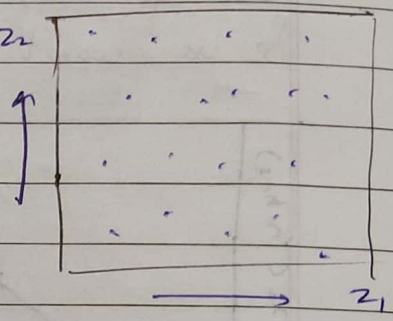
Reduce data 3D to 2D.



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



projected on a particular plane, A plane in which every example was almost lying just little bit distortion.



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

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

vectorized form. ith example

Qv After applying dimensionality reduction on $\{x^{(1)}, x^{(2)}, \ldots x^{(m)}\}$

$x^{(i)} \in \mathbb{R}^n$

we will get

→ lower dimension dataset $\{z^{(1)}, \ldots z^{(m)}\}$ where $z^{(i)} \in \mathbb{R}^k$ $(k \leq n)$

Motivation II : Visualization

(To visualize data.)

1. Dataset.

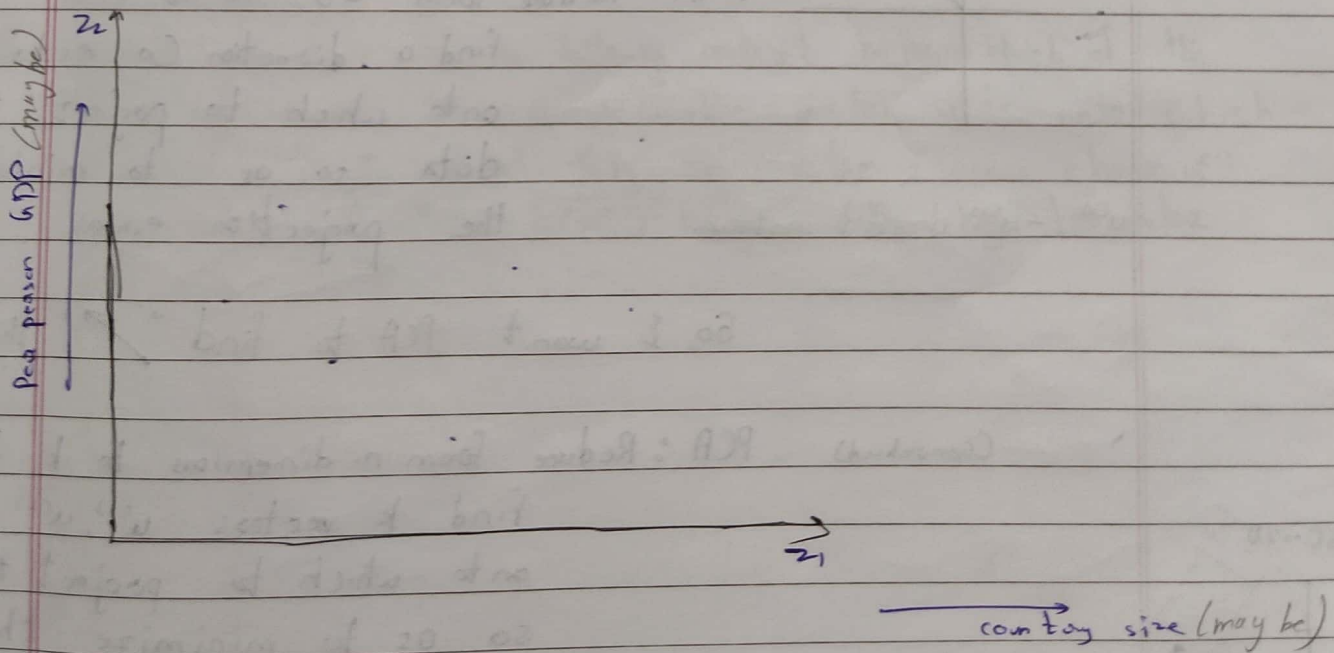
country	GDP x_1	Per capita GDP x_2	life expectancy x_3	x_4	x_5	...	$x \in \mathbb{R}^{50}$ $x^{(i)} \in \mathbb{R}^{50}$

- Data Visualization.

> Reducing to

country	z_1	z_2	$z^{(i)} \in \mathbb{R}^2$

Reduce data from 50 D to 2 D.



→ used for dimension reduction

Principal Component Analysis Problem formulation.

PCA problem formulation.

Sum of square of this line must be minimized (i.e. projection error)

worst line to project data on

good line

projection error is high

dataset

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

We want to reduce dimension, so, we will find projection of dataset on a line.

before applying PCA, do first mean normalization at feature scaling on x_1, x_2.

PCA will choose "/" this line

$u^{(1)}$ / $-u^{(1)}$ doesn't matter.

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

So, I want PCA to find "↗" this vector

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

3D → 2D

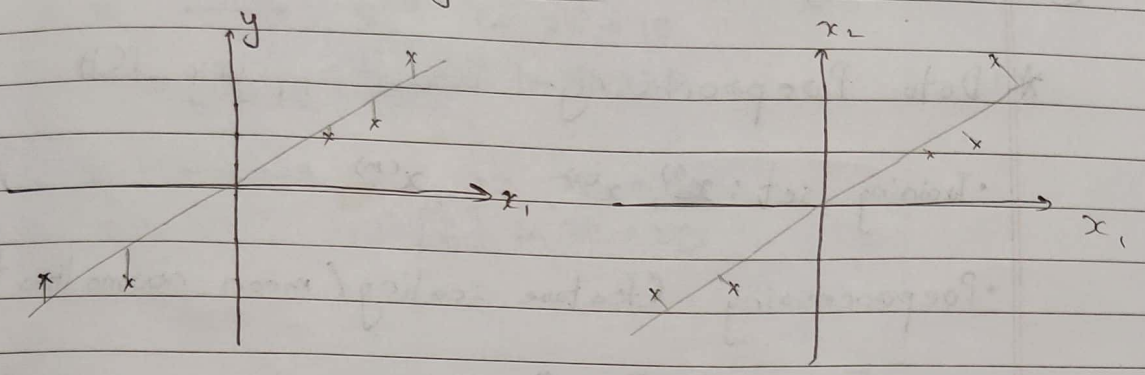
$u^{(1)}$

$u^{(2)}$

Project data onto the linear subspace spanned by set of k vectors.

NOTE: Finding k directions (instead of 1) onto which to

PCA is not linear regression.



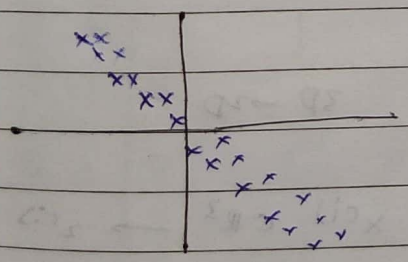
LR

- Vertical dist
  $(x^{(1)}, x^{(1)}, \ldots, y^{(1)}), \ldots$
- To predict y. y is special variable.

PCA

- $\perp^{a}$ dist.
- $x_1, x_2, \ldots, x_n$
- All variable are treated equally.
  [Even in 3D we are treating all feature similar]

Suppose you run PCA on following dataset below. Which of the following would be a reasonable vector $u^{(1)}$ onto which to project the data? (By convention, we choose $u^{(1)}$ so that $\|u^{(1)}\| = \sqrt{(u_1^{(1)})^2 + (u_2^{(1)})^2}$, the length of vector $u^{(1)}$, equals 1.).



✓ $u^{(1)} = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$

-- (a) PCA algorithm

★ Data Preprocessing: Before applying PCA

- Training set: $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$
- Preprocessing (feature scaling / mean normalization):

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

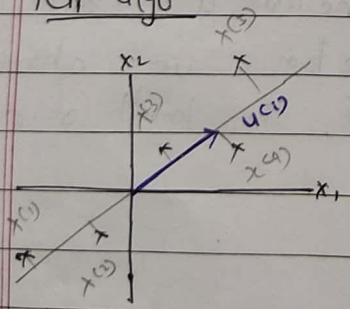
Replace each $x_j^{(i)}$ with $x_j - \mu_j$

If different features on different scales
(e.g. x_1 = size of house
x_2 = no. of bedroom),
Scale features to have comparable range of values.

$$x_j^{(i)} \longleftarrow \frac{x_j^{(i)} - \mu_j}{s_j}$$

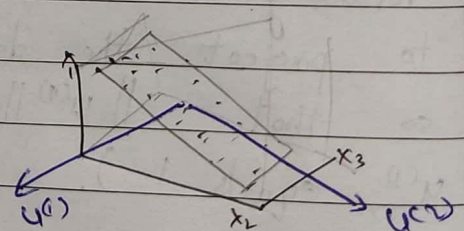
s_j → max − min / standard deviation of feature

PCA algo



2D → 1D

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



3D → 2D

$x^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$

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

We have to find :

$u \longrightarrow u_0^{(1)}$ in 2D → 1D

$u^{(1)}, u^{(2)}$ in 3D → 2D

$z \longrightarrow z_1$ in 2D → 1D

$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ in 3D → 2D

✱ PCA algo

before this → • Reduce data from n-dimension to k-dimensions

do computer

Compute "~~con~~ covariance matrix" :

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

$n \times 1$ $\quad$ $1 \times n$ $= n \times n$

store this variable called Sigma.

Compute "eigen vectors" of matrix Σ :

$[U, S, V] = \text{svd}(\text{Sigma})$ / eig(sigma)

we need only U

single value decomposition.

Covariance matrix always satisfies a mathematical property symmetric positive definite

sigma → n × n matrix

$$U = \begin{bmatrix} | & | & | & & | \\ u^{(1)} & u^{(2)} & u^{(3)} & \cdots & u^{(m)} \\ | & | & | & & | \end{bmatrix} \quad U \in \mathbb{R}^{n \times n}$$

k

To make k Dimensions pick first k vectors. $u^{(1)}, \ldots, u^{(k)}$

PCA algo.

From [U, S, V] = svd (Sigma), we get:

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

K

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

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

$n \times k$

U_{reduce}

$k \times n$ $n \times 1$

$k \times 1$

$z = \mathbb{R}^k$

☆ PCA algo summary.

After mean normalization (ensure every feature has zero mean) and optionally feature scaling.

- $\text{Sigma} = \frac{1}{m}\sum_{i=1}^{m}(x^{(i)})(x^{(i)})^T$ $\qquad X = \begin{bmatrix} — & x^{(1)T} & — \\ & \vdots & \\ — & x^{(m)T} & — \end{bmatrix}$
- `Sigma = (1/m) * X' * X;`
- `[U, S, V] = svd(Sigma);`
- `Ureduce = U(:, 1:k);`
- `z = Ureduce' * x;`

$x \in \mathbb{R}^n$ ~~$x_0 = 1$~~

Q. In PCA we obtain $z \in \mathbb{R}^k$ from $x \in \mathbb{R}^n$ as follows.

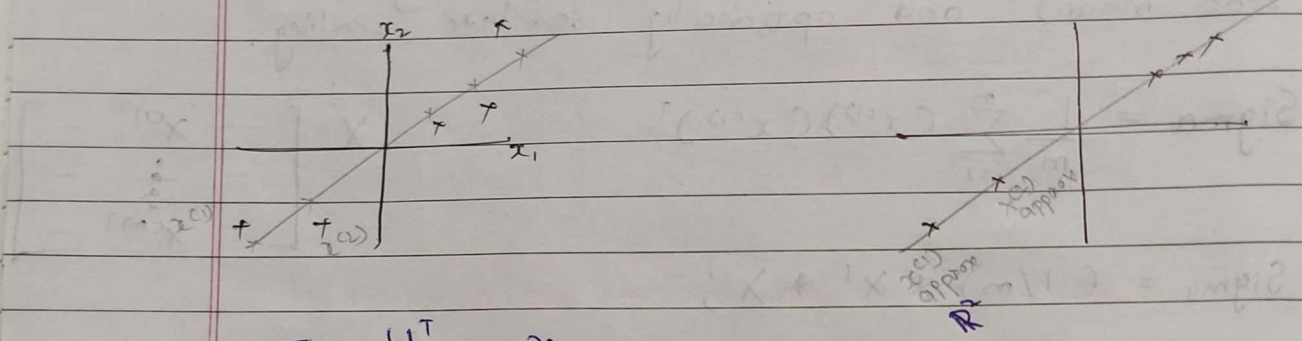
$$z = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(k)} \\ | & | & & | \end{bmatrix}^T x = \begin{bmatrix} (u^{(1)})^T \\ \\ (u^{(k)})^T \end{bmatrix} x$$

~~✓~~ $z_j = (u^{(j)})^T x$

- We haven't proved mathamatically PCA of getting minimized squared distance.

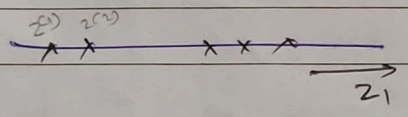
--(a) Reconstruction from Compressed Representation - —

3D to 2D now 2D → 3D going back in this video.



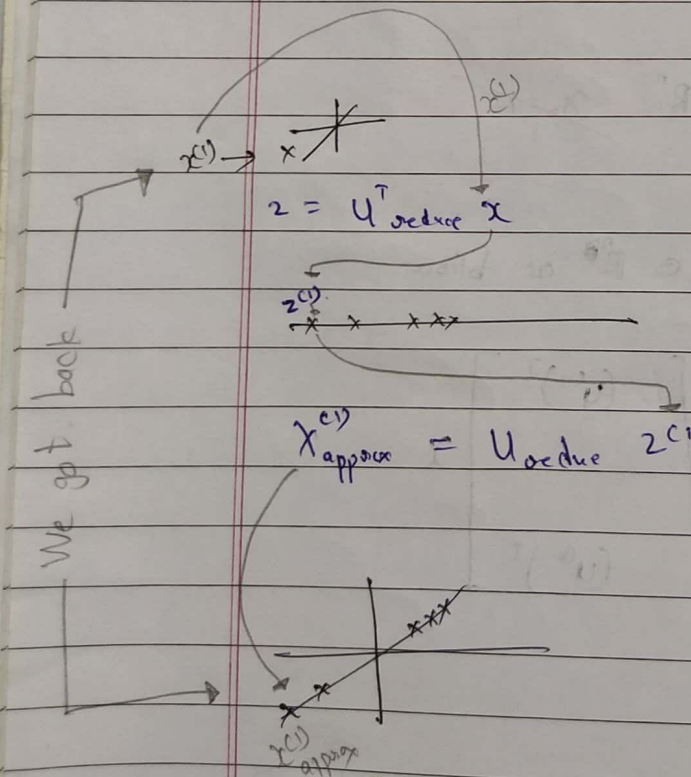
$$z = U_{reduce}^T x$$

$$z \in \mathbb{R} \longrightarrow x \in \mathbb{R}^2$$



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

$\mathbb{R}^{n\times}$; $n\times k$; $k\times 1$; $n\times 1$



(11) Suppose we run PCA with $k=n$ so that the dimension of the data is not reduced at all. (This is not useful in practice but is a good thought exercise.)
Recall that the percent / fraction of variance retained is given by: $\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}}$. Which of the following will be true?

→ ✓ U_{reduce} will be an $n \times n$ matrix

✓ $x_{approx} = x$ for every example x

✓ The percent of variance retained will be 100%.

✗ • We have that $\frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \rightarrow 1$

Choosing value of Principal components.

Choosing k

Average squared projection error: $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x_{approx}^{(i)}\|^2$

Total variance in the data: $\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2$

→ On average how far are my vectors from origin (training eg.)

Typically, choose k to be smallest value so that.

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

~~0.05~~ ~~5%~~

0.10 10%.

~~99%~~. variance is retained.

~~95%~~.

90%.

85 above is good but toy 95%-100%.

★ Choosing k (number of principal components)

Algorithm:

- Try PCA with ~~k=1~~ k=2 k=3 ... and so on, till we get ≤ 0.01

(we want k to minimum)

- Compute $U_{reduce}, z^{(1)}, \ldots z^{(m)}, x^{(1)}_{approx}, \ldots x^{(m)}_{approx}$.
- Check if

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

How we deal with algorithm.

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

$$S = \begin{bmatrix} S_{11} & & & \\ & S_{22} & & \\ & & S_{33} & \\ & & & \ddots \\ & & & & S_{nn} \end{bmatrix}$$

For given k.

$$1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}} \leq 0.01 \quad \longleftarrow \text{check this cond}^n.$$

find minimum k which satisfies

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

Previously we said that PCA chooses a direction $u^{(1)}$ (for k directions $u^{(1)}, \ldots u^{(k)}$) onto which to project the data so as to minimize the squared projection error. Another way to say is PCA tries to minimize

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

-- @ Advice for applying PCA: — — —

☆ Supervised learning speed up.

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

suppose $x^{(i)} \in \mathbb{R}^{10000}$

Extract inputs.

unlabbled dataset : $x^{(1)}, x^{(2)}, \ldots x^{(m)} \in \mathbb{R}^{10,000}$

↓ PCA

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

New training set :

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

Use these

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

Note :- Mapping $x^{(i)} \longrightarrow z^{(i)}$ should be defined by running PCA only on training set. This mapping can be applied as well as to the examples $x^{(i)}_{cv}$ & x_{test} in the cross validation & test set.

Applications of PCA.

- Compression.
  - Reduce memory/disk needed to store data.
  - Speed up learning algo
  - → choose k by % of variance retain.

- Visualization. ⟶ k=2 / k=3

Bad use of PCA:

Use $z^{(i)}$ instead of $x^{(i)}$ to reduce the number of features, less likely to overfit

Bad application of PCA

This might work OK, but isn't a good way to address overfitting. Use regularization method. instead.

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

In regularization knows about label y PCA don't know about y, PCA may throw important information.

Note: 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 & consider $z^{(i)}$.

- Get training set $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots (x^{(m)}, y^{(m)})\}$.
- Get Run PCA to reduce $x^{(i)}$ in dimension to get $z^{(i)}$.
- Train model on $\{(z^{(1)}, y^{(1)}), (z^{(2)}, y^{(2)}), \ldots, (z^{(m)}, y^{(m)})\}$
- Test on test set: Map $x^{(i)}_{test}$ to $z^{(i)}_{test}$. Run $h_\theta(z)$ on $\{(z^{(1)}_{test}, y^{(1)}_{test}), \ldots, (z^{(m)}_{test}, y^{(m)}_{test})\}$