Udacity → Machine learning

Arthur Samuel:

The field of study that gives computers the abili
to learn without being explicitly programmed.

Tom Mitchell:

The field of machine learning is concerned with the
quest of how to construct computers program that
automatically improve with experience. A com. prog.
is said to learn from experience E with respect to
some class of tasks T and performance measure
P, if its performance at tasks in T, as measured
by P, improves with experience E.

i.e. checkers Game:

E = the experience of playing

T = the task of playing checkers

P = the probability that the program will win the
next game.

Machine learning algorithms:

1. Supervised learning
2. Unsupervised learning → Semi Supervised.
3. Reinforcement learning

Acerous Vs. Non-Acerous

1. Supervised Learning:

SL is where you have input variables (x) and output variables (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(x)$$ (level data)

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict output variable (Y) for that data.

Labels

Training Data

↓

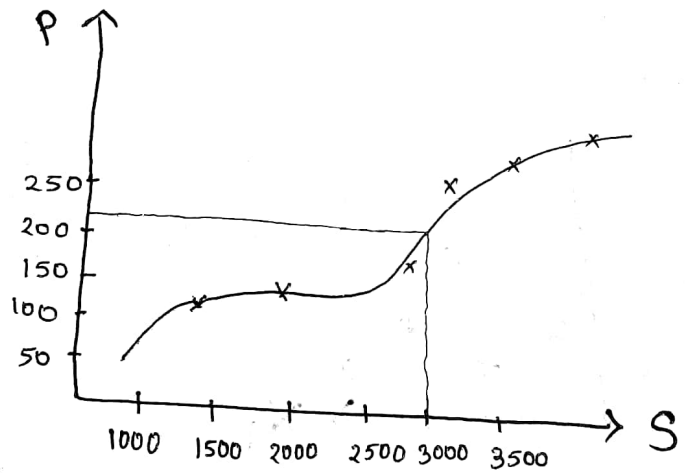
ML. Algo.

↓

New Data ⟶ Predictive Model ⟶ Prediction.

Test data

Model



$y = f(x)$

$P = f(S)$

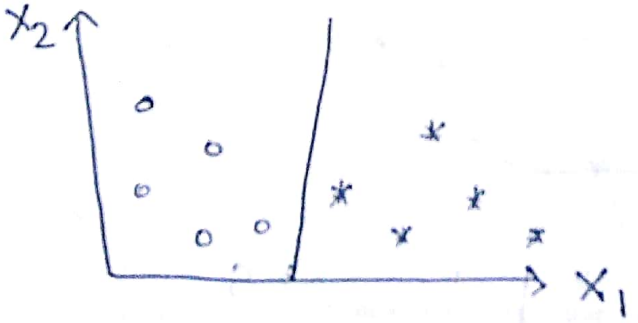
⟶ × ⟵

08.05.18

Supervised Learning → Regression
↳ Predict results within a continuous output

→ Classification
↳ Predict result in discrete output.

* Classification:



* Unsupervised Learning:

Here we have little or no idea what our result should look like

input variable (x) but no Y	Underlying structure

Unsupervised Learning
- Clustering
  - want to discover the inherent grouping in the data such as groups customers by purchasing behavior. (UL classification)
- Association.

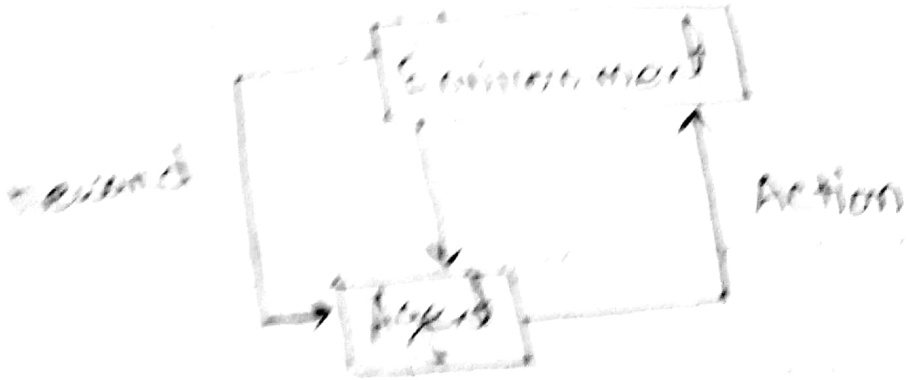
labeled → supervised

* Clustering:
  - Share certain degree of similarity.
  - more dissimilar to object in other clusters.

* Association:
  - Discovers rules that describe large portions of our data.
  - Such as: People that buy x also tend to buy Y.

* Reinforcement Learning:

↳ The goal is to develop a system (agent) that improves its performance based on interactions with the environment, no answer key.



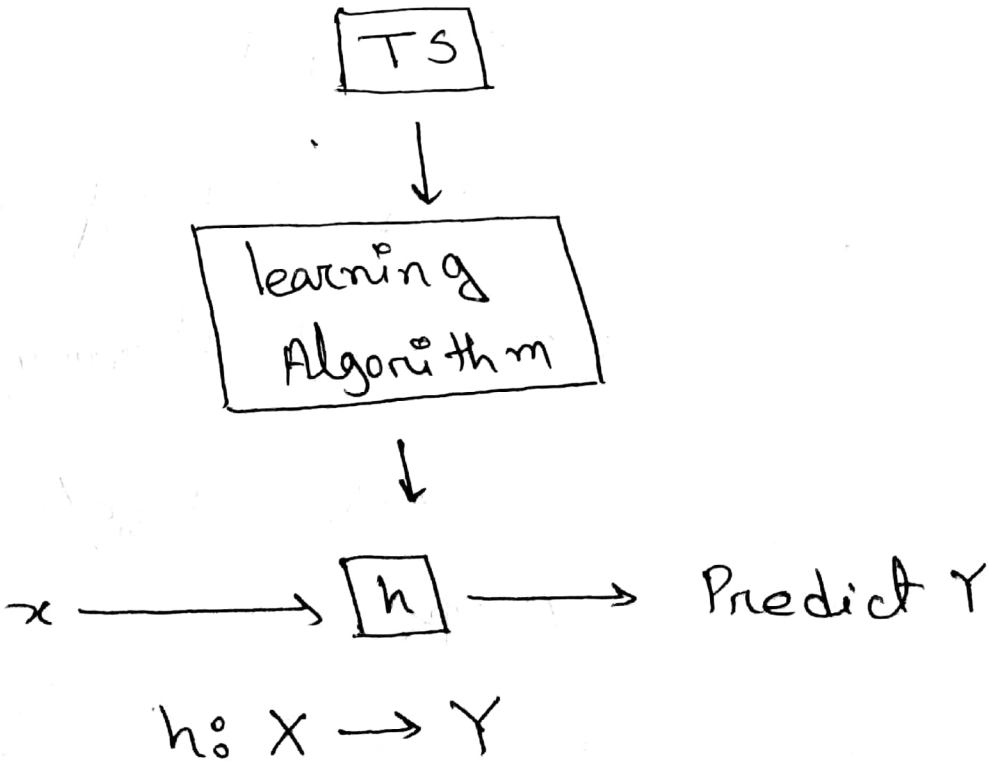
13.05.18.

* Linear regression with one variable:

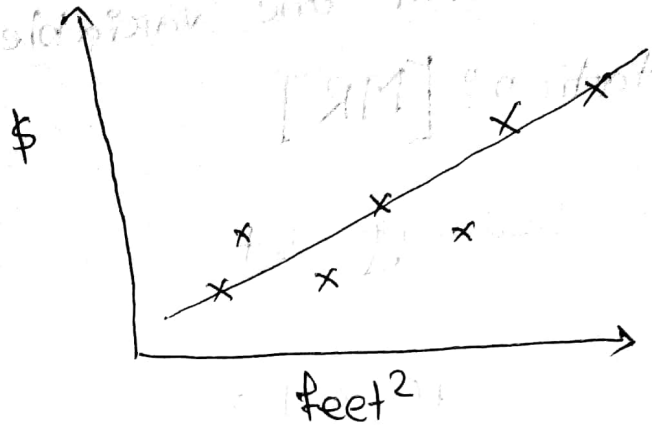
Model Representation: [MR]

TS → Training Set

Feet^2	1000$s
2104	400
1600	330
2400	369
1416	540
⋮	⋮



$h: X \rightarrow Y$



*Univariant linear regression.

$h_\theta(x) = \theta_0 + \theta_1 x$

m = # No. of training example

n = # No. of variables

x = input

y = output

(x, y) = training example

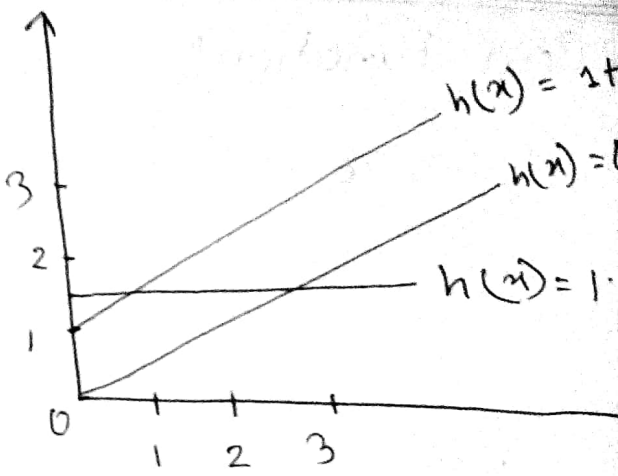
$(x^{(i)}, y^{(i)})$ = i^{th} training examp

$\theta \rightarrow$ Parameters/weights

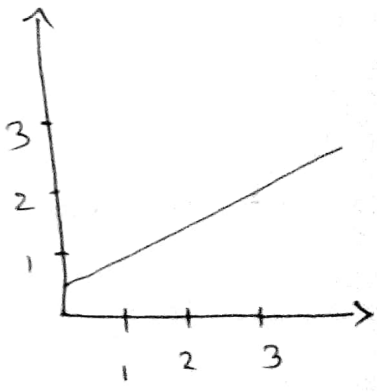
$\theta \in \mathbb{R}$

$h_\theta(x) = \theta_0 + \theta_1 x$

$\theta_0 = 1.5$	$\theta_0 = 0$	1
$\theta_1 = 0$	$\theta_1 = 0.5$	0.5



$\theta_0 =$	0	0.5	1	1
$\theta_1 =$	1	1	0.5	1



i.e°

X	Y
0	4
1	(7)
2	7
3	8

$h_\theta(x) = \theta_0 + \theta_1 (x)$; $\theta_0 = \theta_1 = 2$

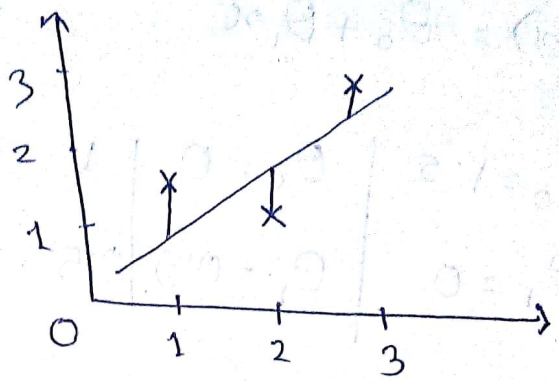
$= 2 + 2 * x$

if $x = 0 \rightarrow 2 + 2 \cdot 0 = 2$

$1 \longrightarrow 2 + 2 \times 1 =$ (4) x

* Cost Function:

minimize
θ_0, θ_1



minimize
$\theta_0\theta_1 \quad (h_\theta(x) - y)^2 \rightarrow \min_{\theta_0\theta_1} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Least square Error.

Least mean square Error (LMS):

$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

↓

Minimize
θ_0, θ_1

Hypothesis : $h_\theta(x) = \theta_0 + \theta_1 x$

Parameter : θ_0 and θ_1

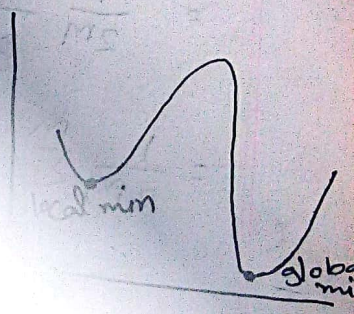
Cost function:

$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Goal: minimize
$\theta_0, \theta_1 \quad J(\theta_0, \theta_1)$

(any value)

until we have minim



em in

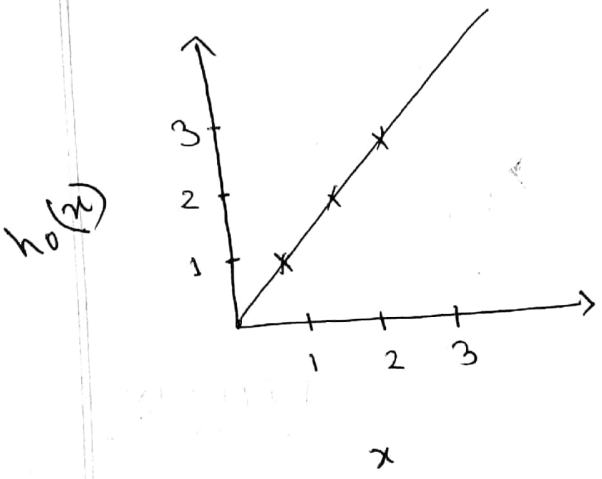
t desce

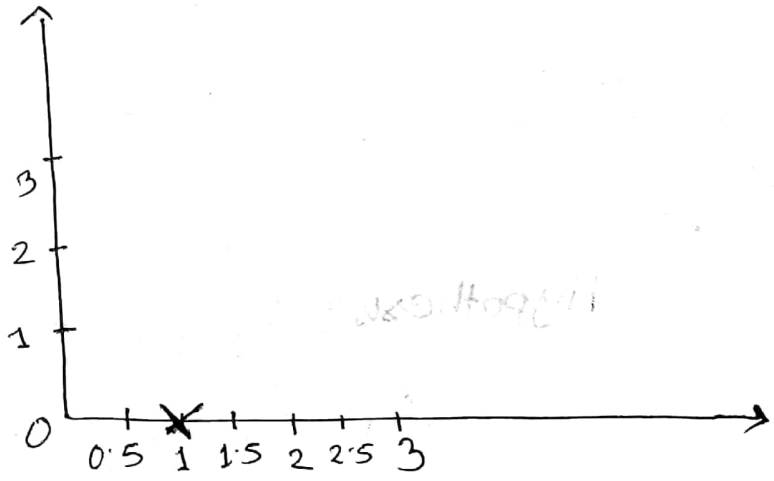
$h_\theta(x) = \theta_0 + \theta_1 x$ Let, $\theta_0 = 0$

$h_\theta(x) = \theta_1 x$

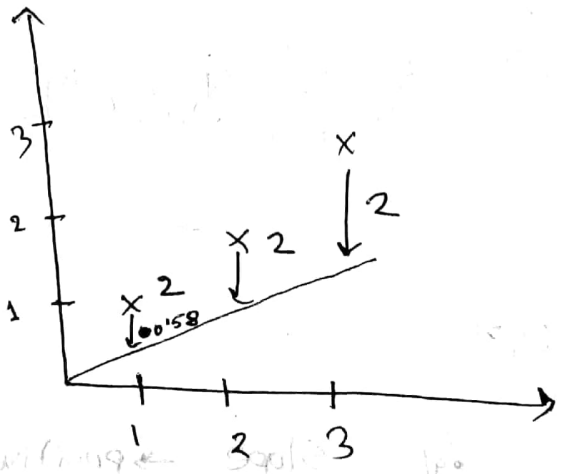
if $\theta_1 = 1$

then, $h_\theta(x) = x$ [$Y = x$]





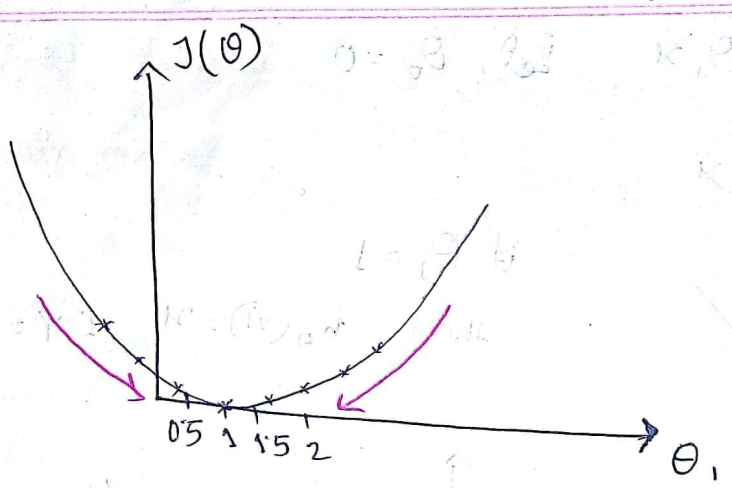
As distance 0 (1-1 = 0)



$\theta_1 = 0.5$

$$J(0.5) = \frac{1}{2m}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right]$$

$$= \frac{1}{2\times 3}(\quad) \qquad = 0.58$$



15.05.18

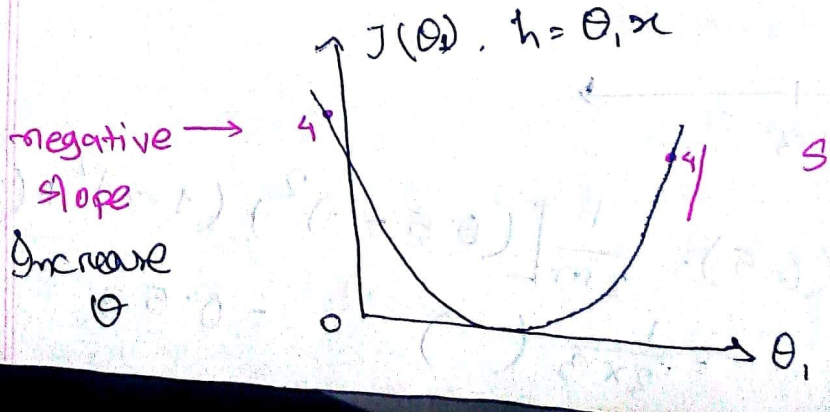
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

θ_1, θ_0 → Parameters

Cost function: $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Goal: Minimize

θ_0, θ_1

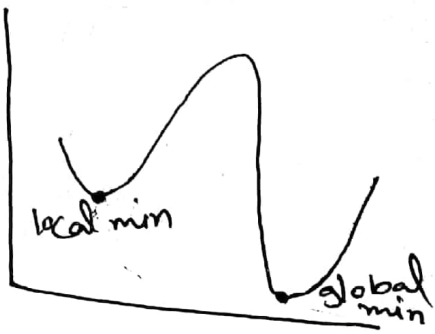


Gradient Descent:

→ start with θ_0, θ_1 (any value)

→ Keep changing θ_0, θ_1, until we have minimum $J(\theta)$ (Hopefully)

→



problem in gradient descent

// we get more global solution

repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \quad ; \quad j = 0, 1$$

}

repeat

α = learning rate

{

$$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta) = \theta_0 - \alpha \frac{\partial \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}{\partial \theta_0}$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta) = \theta_1 - \alpha \frac{\partial \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}{\partial \theta_1}$$

}

When $j=0$,

$$\theta_0 \longrightarrow \frac{\partial}{\partial \theta_0} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$= \frac{1}{2m} \frac{\partial}{\partial \theta_0} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$= \frac{1}{2m} \sum_{i=1}^{m} 2 \cdot \left(h_\theta(x^{(i)}) - y^{(i)} \right) \frac{\partial}{\partial \theta_0} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$= \frac{1}{2m} \cdot 2 \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \frac{\partial}{\partial \theta_0} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)$$

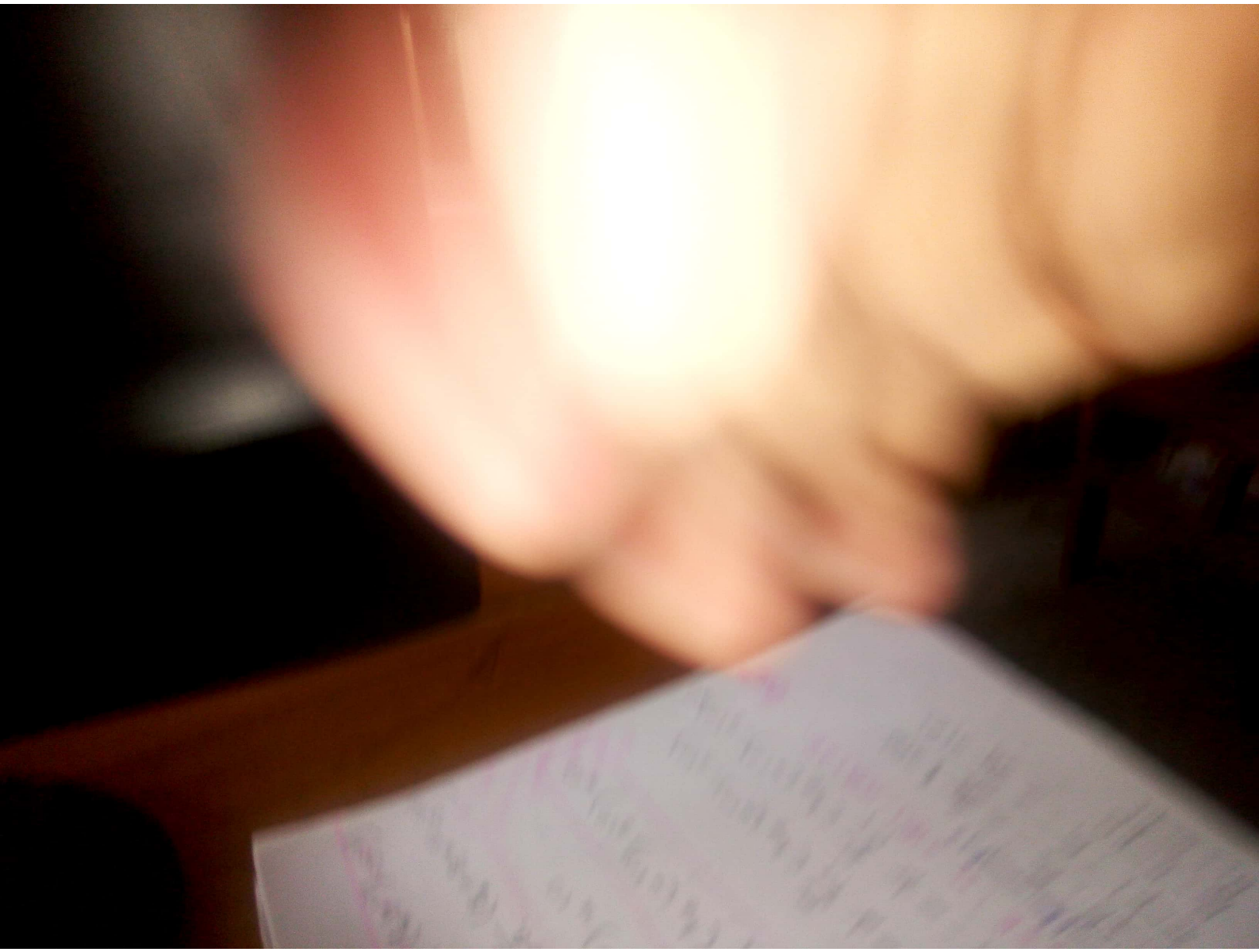
$$= \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot 1$$

$$\therefore \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

When, $j=1$,

$$\theta_1 = \frac{\partial}{\partial \theta_1} \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$= \frac{1}{2m} \frac{\partial}{\partial \theta_1} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$



$$= \frac{1}{2m^2} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \frac{\partial}{\partial \theta_1} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \frac{\partial}{\partial \theta_1} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)}\right)$$

$$= \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

$$\therefore \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

repeat

{

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}$$

} $\theta_0 - \theta_0 \geq \varepsilon$ or $J(\theta_1) - J(\theta_0) \geq \varepsilon$ ..... //Condition to stop.

update simultaneously:

{ $temp0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta)$

$temp1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta)$

$\theta_0 = temp0$

$\theta_1 = temp1$

}

Batch Gradient descent:

↓ Solution

Incremental / Stochastic Gradient descent:

(fast, but possibility of getting global solution is less.)

Loop {

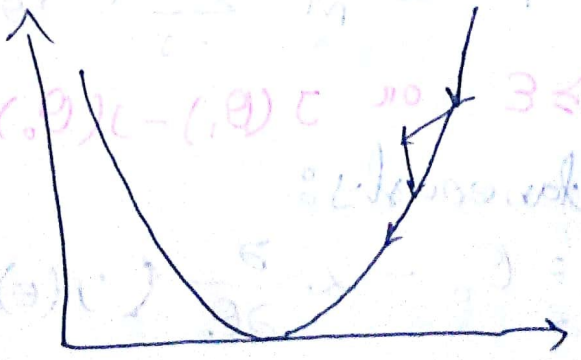
for i = 1 to m {

$$\theta_j = \theta_j - \alpha\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$$

}

; for j = 0, 1

}



—x—

"Special class"

17.05.18.

eliademy.com

Python

Introduction to Machine Learning and Decision Tree

aktarhossain@daffodilvarsity.edu.bd

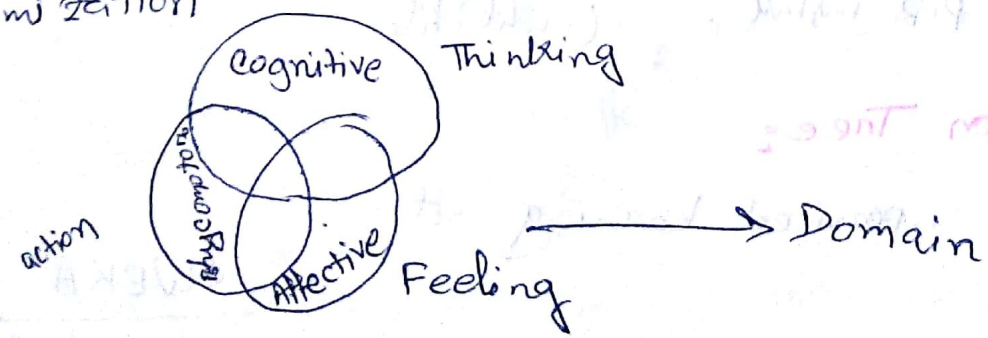
Industry 4.0

* Three components of ML:

Representation

Evaluation

Optimization



Bloom's Taxonomy of learning

Supervised or inductive learning:

markany.com

2**100 → Power

every single type is tuple

Python Basics.pdf.

Everything in python is an object.

Pallindrome

1. get-pip.py

   Install pip installer

   C:\> Python get-pip.py (Internet)

2. Install Numpy & sci-pi

   C:\> pip install ---(.whl)file

Decision Tree:

Supervised learning set

"WEKA" → DM Tool

New Zealand Waikato University

20.05.18.

x_0	x_1 feet^2	x_2 # bedrooms	x_3 # floor	x_4 Age	Price
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

n = # features

$x^{(i)}$

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix} \begin{matrix} \rightarrow x_1 \\ \rightarrow x_2 \\ \rightarrow x_3 \\ \rightarrow x_4 \end{matrix}$$

$$x^{(2)}_3 = 2$$

$$x^{(i)} = \begin{bmatrix} x_1^{(i)} \\ x_2^{(i)} \\ x_3^{(i)} \\ x_4^{(i)} \end{bmatrix} \in \mathbb{R}^4$$

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

$$h_\theta(x) = \theta_0 + \theta_1 x + \dots + \theta_n x_n$$

Let, $x_0 = 1$ for all $i = 1$ to m

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

$$= \sum_{j=0}^{n} \theta_j x_j = \theta^T x$$ (no. of features).

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

$n+1$ ~~dimen~~ dimensional vector

$(n+1) \times 1$ matrix

$$\theta^T x = \begin{bmatrix} \theta_0 & \theta_1 & \cdots & \theta_n \end{bmatrix} \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix}$$

* Gradient Descent:

$$J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(\theta^T x^{(i)} - y^{(i)} \right)^2 \quad \text{(no. of examples } m\text{)}$$

$$= \frac{1}{2m} \sum_{i=1}^{m} \left(\left(\sum_{j=0}^{n} \theta_j x_j^{(i)} \right) - y^{(i)} \right)^2$$

when, $n = 1$

Repeat {

$$\theta_0 = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$

$$\theta_1 = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}$$

}

when $n \geq 2$

Repeat {

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

}

Update simultaneously

Assignment

Derivation of → Next class

$$\theta_0 = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_0^{(i)}$$

$$\theta_1 = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_1^{(i)}$$

$$\vdots$$

$$\theta_n = \theta_n - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_n^{(i)}$$

Gradient Descent in practice:

→ Feature Scaling

→ Mean ~~no~~ normalization

→ Debug

→ select α

x_1 = size = 100 - 2000 feet2

x_2 = # bedrooms = 2 - 5

* θ will descent quickly on small range so, set input value in same range (roughly).

$$-1 \le x^{(i)} \le 1$$

$$-0.5 \le (x^{(i)}) \le 0.5$$

$$\boxed{-3 \le x^{(i)} \le 3}$$

Feature Scaling:

$$x_j = \frac{x_j}{\text{range}} = \frac{2104}{2400} \approx 1 \quad ; \text{for } x_1$$

$$x_2 = \frac{5}{3} = 1.5$$

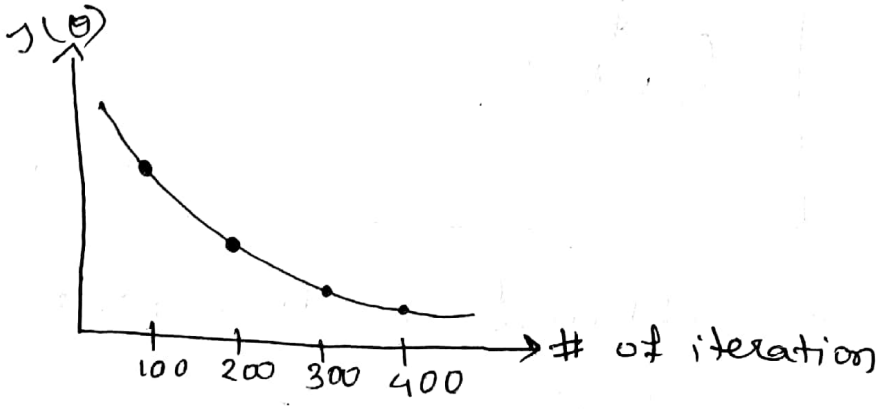
Mean normalization:

$$x_j = \frac{x_j - \mu_j \ (\text{mean avg.})}{s_j \ (\text{range})}$$

$$\mu_j = \frac{5+3+3+2}{4}$$

$$x_2^{(1)} = \frac{5 - 3.25}{3}$$

Debug:



$J(\theta) = 5$ (2)

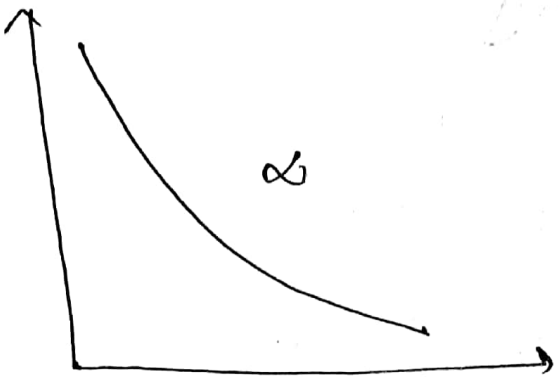
$J(\theta) = 4.99999$ (3)

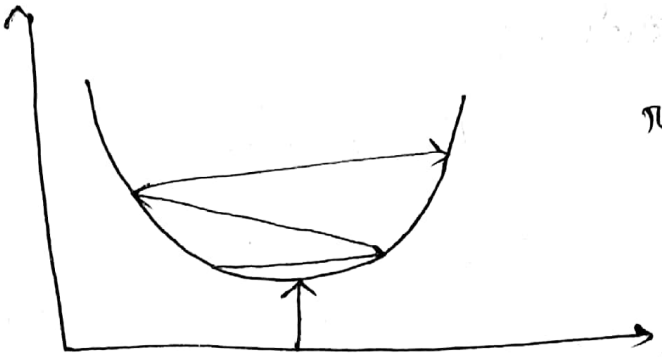
Change learning rate, α

→ Threshold value $\approx 10^{-3}$

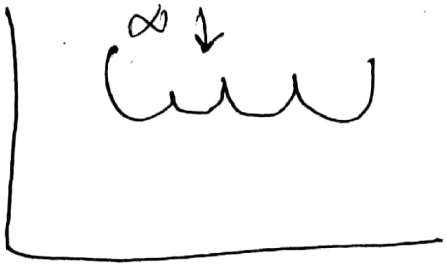
* If the change is below threshold value then we can assume to get optimal solution.

Select α:





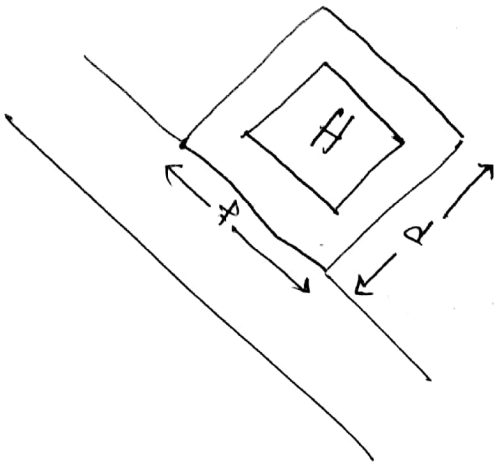
reduce the value of α.



$\alpha = 0.001, 0.003, 0.01, 0.03$

* for sufficient small α, $J(\theta)$ increase for every iteration.

22.05.18.

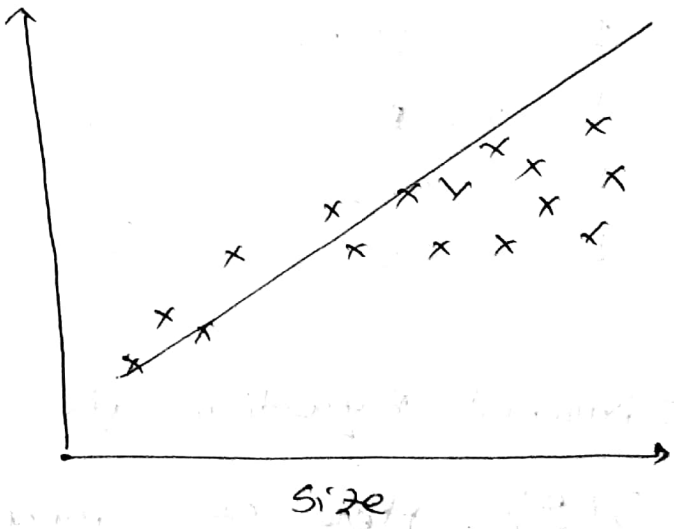


$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

x_1 = area

$$= \theta_0 + \theta_1 x$$

*Polynomial Regression



$x_1 = size$

$x_2 = size^2$

$x_3 = size^3$

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$h_\theta(x) = \theta_0 + \theta_1 x^2$$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$

Feature Scaling:

$1 - 1000$

$x_1 = 1000$

$x_2 = 1000000$

$x_3 = 10^9$

$$h_\theta(x) = \theta_0 + \theta_1 \text{size} + \theta_2 \sqrt{\text{size}}$$

Normal Equation:

x_0	feet^2	#bedrooms	#floor	Age	Price
1	2104	5	1	45	316
1	1416	3	2	40	420
1	1534	3	2	30	290
1	852	2	1	35	330

Normal Equation gives other way of minimizing $J(\theta)$. Here we minimize $J(\theta)$ explicitly taking its derivatives w.r.to the θ_j and setting them to zero. This allow us to find the optimal θ without iteration.

$$\theta = (X^T X)^{-1} X^T y$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} = \mathbb{R}^{n+1} \qquad \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \text{output}.$$

X = input matrix

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1535 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 35 \end{bmatrix}$$

$$x^{(1)} = \begin{bmatrix} 1 \\ 2104 \\ 5 \\ 1 \\ 45 \end{bmatrix} \qquad x^{(2)} = \begin{bmatrix} 1 \\ 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix}$$

$$X = \begin{bmatrix} x_i^{(1)T} \\ x_i^{(2)T} \\ \vdots \\ x_i^{(m)T} \end{bmatrix}$$

X X^T

$m \times (n+1)$ $(n+1) \times m$ → dimension

4×5

Feature scaling not needed

Gradient Descent	Normal Equation
1 Need to choose α.	1. No need to choose α.
2. Need many iteration.	2. No iteration needed
3. $O(kn^2)$	3. $O(n^3)$ [$(X^TX)^{-1}$] [if no. of feature is huge, then complexity would increase]
4. Works well if n is large	4. Slow if n is large [n = no. of features]

* If $n \geq 1000$ it is better to use gradient descent

$(X^TX)^{-1}$ not invertible,

- On regularization
- 1. Redundant features
- 2. Delete feature $n \geq m$

* Assignment → finding weight from height using Python. Before → 5th June.

Sklearn linear regression

udacity

↓

Introduction to machine learning.

———x———

27.05.18

Classification :

Email : Spam / Not

Online Tran. : Normal / Not

Tummer : Malignant / Benign

Binary classification : $y \in \{0, 1\}$

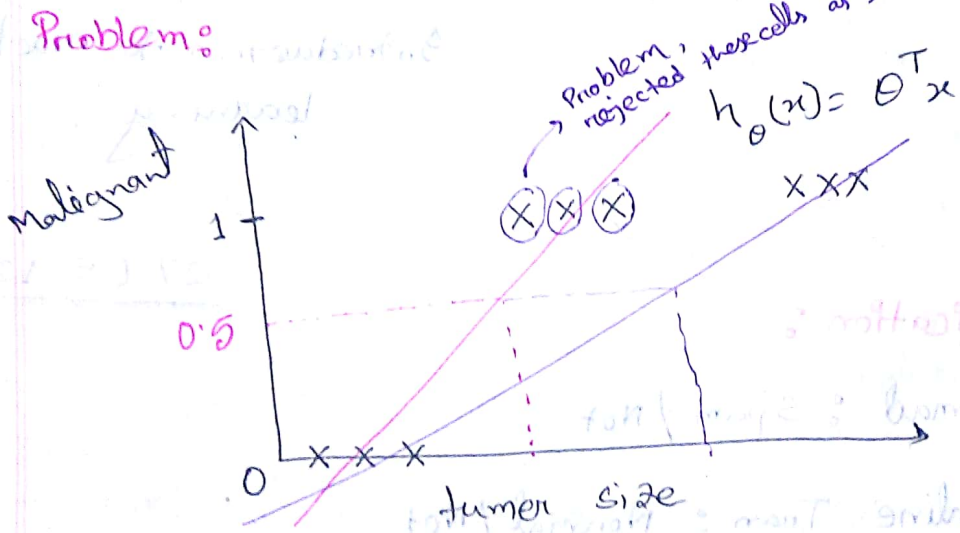
Multiclass classification : $y \in \{0, 1, 2, 3, \ldots\}$

0 : Negative class / Absence of something

1 : Positive class / Presence of something

* Problem with linear Regression for classification

Problem:



Threshold value : 0.5

$h_\theta(x) \geq 0.5, \quad y = 1$

$h_\theta(x) < 0.5, \quad y = 0$

Problem:

$Y = 0$ or 1

$h_\theta(x)$ using Linear Regression may produce value

< 0 or > 1

* Solution :- →

Logistic Regression (Logistic function, sigmoid function)

[classification algorithm]

$$\frac{1}{1 + e^{-x}}$$

always produces result -

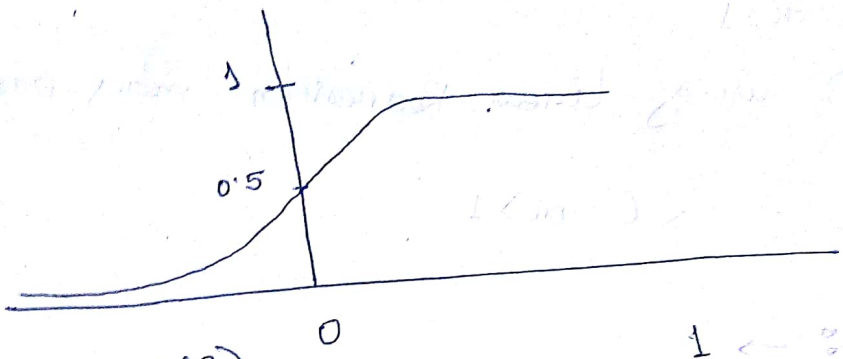
$$0 \leq h_\theta(x) \leq 1$$

Hypothesis:

$$h_\theta(x) = g(z) \quad ; \quad z = \theta^T x \,/\, \theta_0 + \theta_1 x$$

$$= \frac{1}{1 + e^{-z}}$$

$$= \frac{1}{1 + e^{-\theta^T x}}$$



$g(z)$

$g(0) = 0.5$

$g(\infty) = 1$

$g(-\infty) = 0$

$\frac{1}{1+e^{-z}}$

$\frac{1}{1+e^{-\infty}}$

$h_\theta(x)$

→ Estimate Probability that $y = 1$ on input x.

If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumor\_size} \end{bmatrix}$ $= h_\theta(x) = 0.7$

$h(x) = P(y = 1 \mid x, \theta)$ Probability that $y = 1$, given x parameterized by θ

$y = 0$ or 1

$P(y = 1 \mid x, \theta) + P(y = 0 \mid x, \theta) = 1$

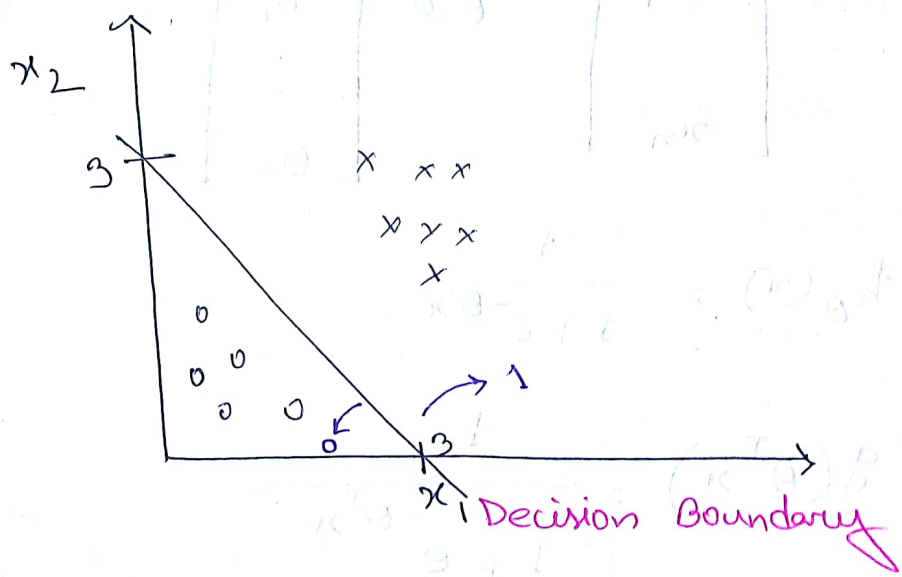
$P(y = 1 \mid x, \theta) = 1 - P(y = 0 \mid x, \theta)$

$P(y = 0 \mid x, \theta) = 1 - P(y = 1 \mid x, \theta)$

Predict, $y = 1 \longrightarrow h_\theta(x) \geq 0.5 \longrightarrow \theta^T x \geq 0$

$y = 0 \longrightarrow h_\theta(x) < 0.5 \longrightarrow \theta^T x < 0$

Decision Boundary: is the line that separates the area where $y = 0$ and $y = 1$. It is created by hypothesis function.



$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 = \theta^T x$$

$$\begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

$$-3 + 1x_1 + 1x_2$$

$$-3 + x_1 + x_2 \geq 0$$

$$-3 + x_1 + x_2 = 0$$

$$x_1 + x_2 = 3$$

— x —

29.05.18.

Logistic Regression:

Tranning Example: $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

Example : m

Features : n

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad \theta \in \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

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

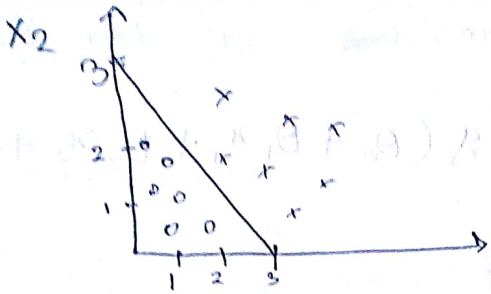
$$g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

$Y = 1, \quad g(z) \geq 0.5$

$\theta^T x = 0$

if $\theta^T x \geq 0, \quad Y = 1$

else $Y = 0$



$g(z) \longrightarrow h_\theta(x)$

$\theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

$$\begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix} = -3 + x_1 + x_2$$

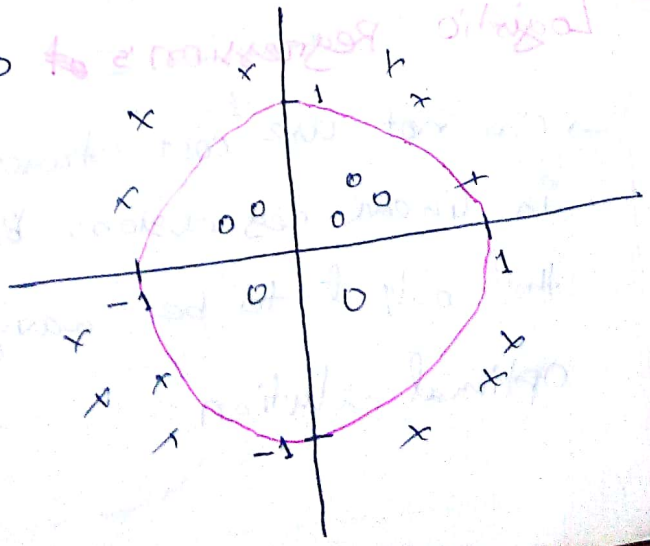
$Y = 1$ when $\theta^T x \geq 0$

$Y = 0$ when $\theta^T x < 0$

$-3 + x_1 + x_2 \geq 0$

$x_1 + x_2 \geq 3$

$x_1 + x_2 = 3$



$h_\theta(x)$

$$g(\theta^T x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = -1 + x_1^2 + x_2^2$$

$$-1 + x_1^2 + x_2^2 \geq 0$$

$$x_1^2 + x_2^2 = 1 \quad [\text{equ. for circle}]$$

* Decision boundary is a property of hypothesis, not of training set.

Logistic Regression's cost function:

→ Can not use cost function that we've used in linear regression. Because, it will cause the output to be wavy, causing many local optimal solution.

→ it will not be ~~coin~~ convex function.

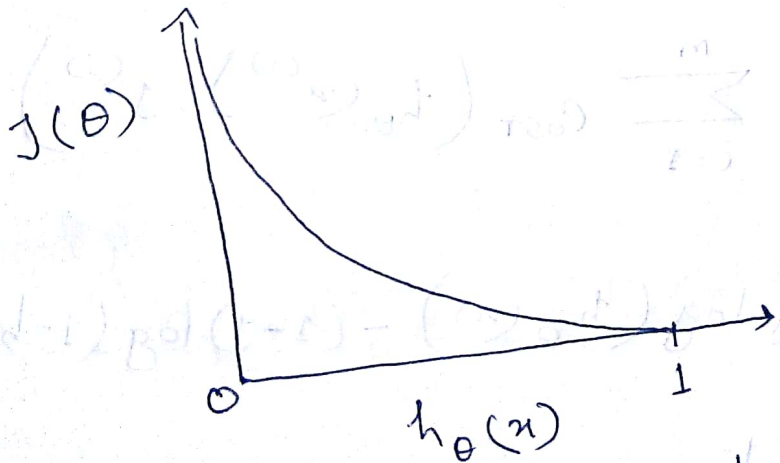
* Cost function:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

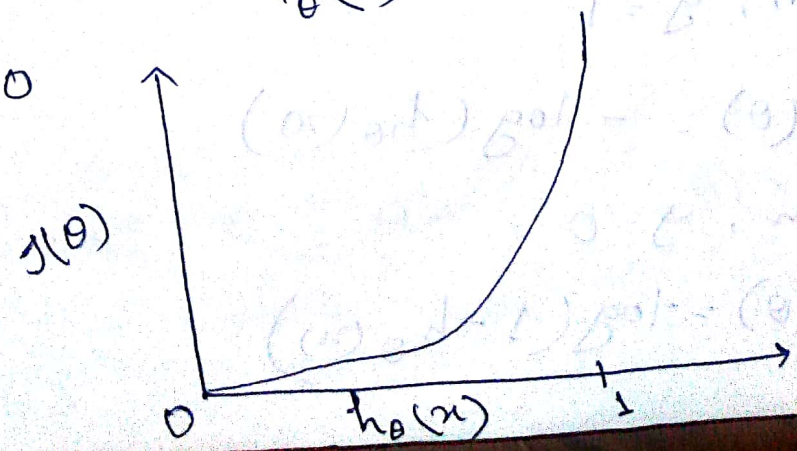
$$\text{Cost}\left(g\left(\sum_{i=0}^{n} \theta^T x^{(i)}\right), y^{(i)}\right)$$

$Y = 1$ or 0

when $y = 1$



$y = 0$



$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$-\log(1) = 0$

$J(\theta) = 0, \quad y = h_\theta(x)$

$J(\theta) = \infty, \quad y = 1, \quad h_\theta(x) = 0$

$J(\theta) = \infty, \quad y = 0, \quad h_\theta(x) = 1$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

$$J(\theta) = -y \log(h_\theta(x)) - (1-y) \log(1 - h_\theta(x))$$

when, $y = 1$

$$J(\theta) = -\log(h_\theta(x))$$

when, $y = 0$

$$J(\theta) = -\log(1 - h_\theta(x))$$

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log\left(h_\theta(x^{(i)})\right) + \left(1-y^{(i)}\right) \log\left(1-h_\theta(x^{(i)})\right)\right]$$

$$J(\theta) = \frac{1}{m}\left(-y^T \log(h) - (1-y)^T \log(1-h)\right)$$

Gradient Descent:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

// update simultaneously

}

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

}

Vector form: $\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - y)$

- BFGS
- L-BFGS

03.06.18.

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

↓

$g(z)$

0.5

$y = 1$ $\quad \theta^T x \geq 0$

$$\begin{cases} 1 & -\log(h_\theta(x)) \\ 0 & -\log(1 - h_\theta(x)) \end{cases}$$

↓

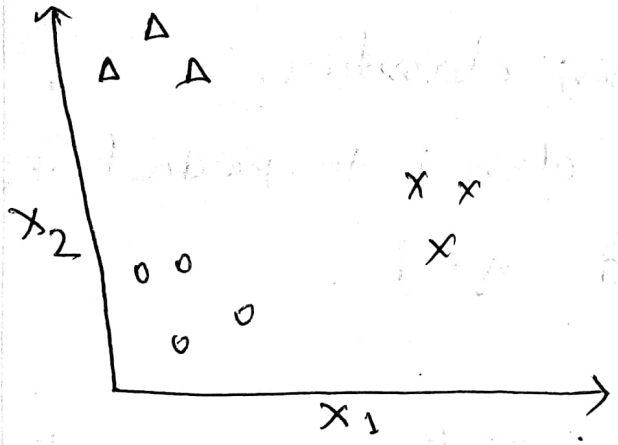
$g(z)$

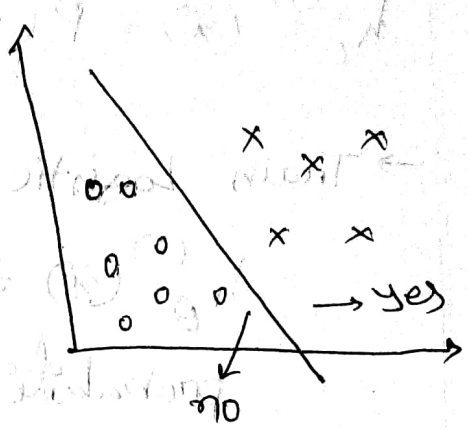
$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left(y \cdot \log(h_\theta(x)) + (1-y) \log (1 - h_\theta(x)) \right)$$

$y = 1, \quad h_\theta(x) \to 0$; cost = infinity

$y = h_\theta(x)$; cost = zero

Multiclass classification:





binary classification

multiclass classification

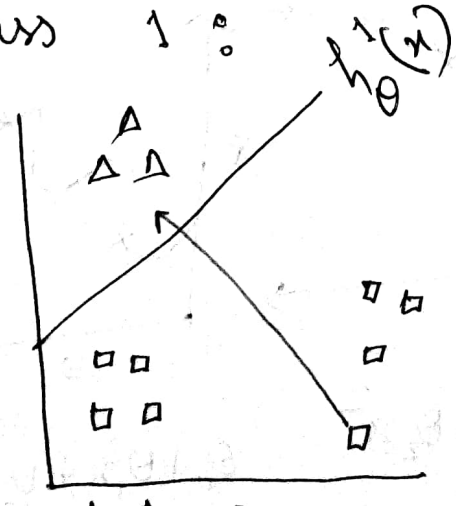
* One Vs all (one Vs rest):

class 1: △

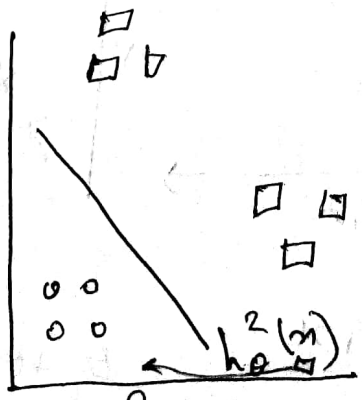
class 2: □

class 3: x

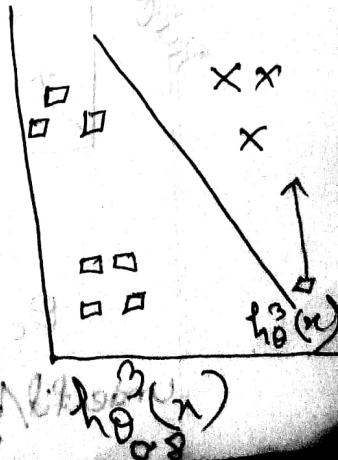
class 1:



class 2:



class 3:



$h_\theta^{(i)}(x) = P(y = i | x, \theta) \quad (i = 1, 2, 3)$

→ Train Logistic Regression classifier.

$h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$.

→ One new input x, to make a prediction pick the class i th maximize

$$\max_i h_\theta^{(i)}(x)$$

Overfiting Vs Underfiting:

$\theta_0 + \theta_1 x$

Underfit/high bias

$\theta_0 + \theta_1 x + \theta_2 x^2$

Just fit

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Overfit/High variance

* Underfit:

Structure not captured by model.

$h_\theta(x)$ maps poorly.

* Overfit:

Fits training data set very well ($J(\theta) \approx 0$),
but fails to generalize to new input

Addressing overfiting problem:

1. Reduce the # of features
   - → manually select which feature to keep.
   - → Use a model selection algorithm
2. Regularization
   - → keep all features, but reduce the magnitude of θs.
   - → works well when we have a lot of slightly useful features.

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

θ_3^2 ↓ tends to $\theta \to 0$; θ_4^2 ↓ 0

$x_0 \quad x_1 \quad x_2 \quad x_3 \quad \cdots \quad x_n$

$\theta_0 \quad \theta_1 \quad \theta_2 \quad \theta_3 \quad \cdots \quad \theta_n$

} don't know which one to reduce.

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

λ = regularization constant.

—x—

R

05.06.18.

Regularization: (for linear Regression)

⇒ Small value for θs

→ simple hypothesis

→ less prone to overfitting

| if λ high, θ will be small.

Features: $x_0, x_1, x_2, \ldots, x_n$

P: $\theta_0, \theta_1, \theta_2, \ldots, \theta_n$

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

Gradient descent:

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)} \right.$$

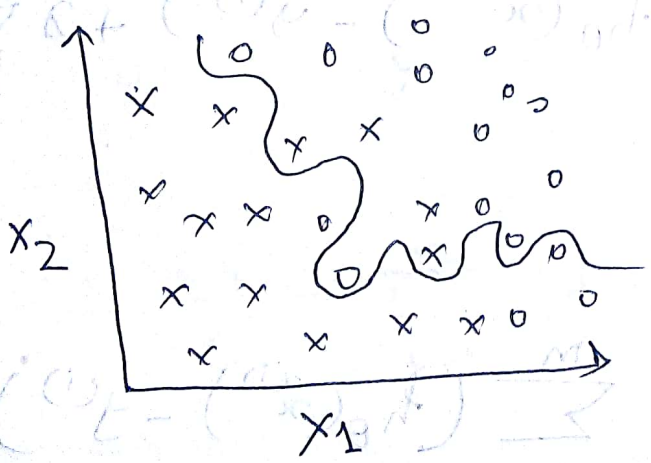
$$\left. + \frac{\lambda}{m} \theta_j \right] ; j = 1, 2, 3, \ldots, n$$

}

$$\theta = (X^T X + \lambda L)^{-1} X^T Y \quad ; \quad L = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{n+1}$$

—x—

Regularization for Logistic Regressions:



$$h_\theta(x) = g(z)$$

$$= g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2 + \theta_6 x_1^2 x_2 + \dots)$$

* Cost :

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

Gradient descent:

repeat {

$$\theta_0 = \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j\right]; \; j = 1, 2, 3 \ldots n$$

}

—x—

Function Linear-Reg (theta-0, theta-1, x)

hy° $h_\theta(x) = \theta_0 + \theta_1 x$

cost° $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

GD°

repeat {

$\theta_0 := \theta_0 - \alpha \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)$

$\theta_1 := \theta_1 - \alpha \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x^{(i)}$

}

Fun Linear (theta-0, theta-1, x)

return $\theta_0 + \theta_1 * x$

costfun (x)

for i = 1 to n

cost = cost + (Linear($\theta_0, \theta_1, x^{(i)}$) - $y^{(i)}$)^2

return cost * $\frac{1}{2m}$

x[80] = { . . . } //Global
y[80] = { . . . }

GDfun (θ_0, θ_1)

for i = 1 to n //repeat 1000 times

$\theta_0 = \theta_0 - (x^{(i)} - y^{(i)})$

$\theta_1 = \theta_1 - (x^{(i)} - y^{(i)})\, x^{(i)}$

Kaggle

↓

registration.

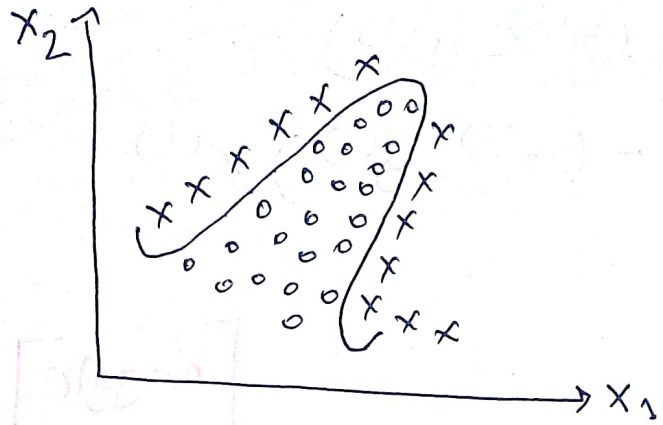
—x—

After Mid

24.06.18.

Neural Networks:

* classification Problem:



g(z)

↓

$h(\theta) = (\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + \theta_5 x^2 + \theta_6 x_1^2 x_2 + \theta_7 x_1 x_2^2 + \ldots)$

If so many features.

x_1 = size

x_2 = # room

x_3 = # floor

x_4 = area

. . . . .

x_{100} = - - - .

(n) = 100

$x_{100} = \ldots$

2nd order equation is needed,

$$x_1^2 + x_1x_2 + x_1x_3 + \ldots + x_1x_{100} + x_2^2 + x_2x_3$$

$$+ x_2x_4 + \ldots + x_2x_{100} + \ldots$$

Complexity $O(n^2)$

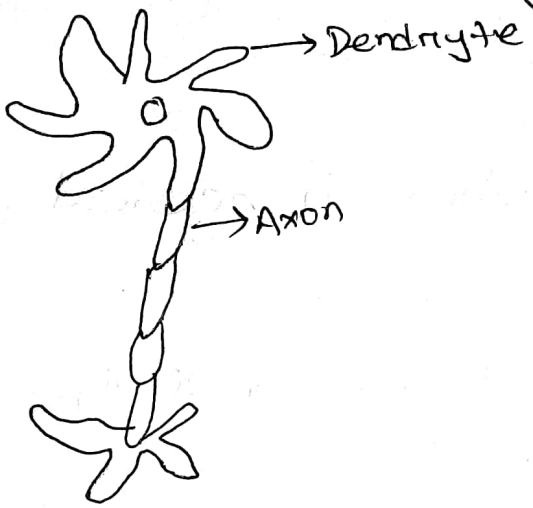
Gray image → 8 byte

Color image → 32 byte

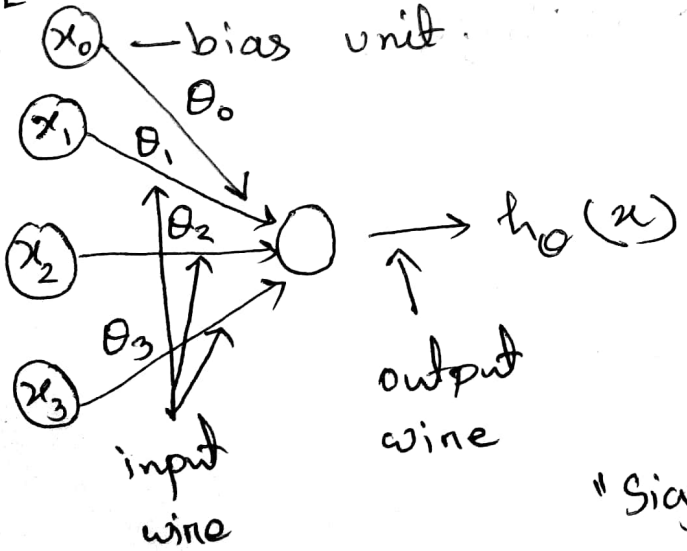
—x—

Artificial Neural Network: 26.06.18.

(for classification problem)



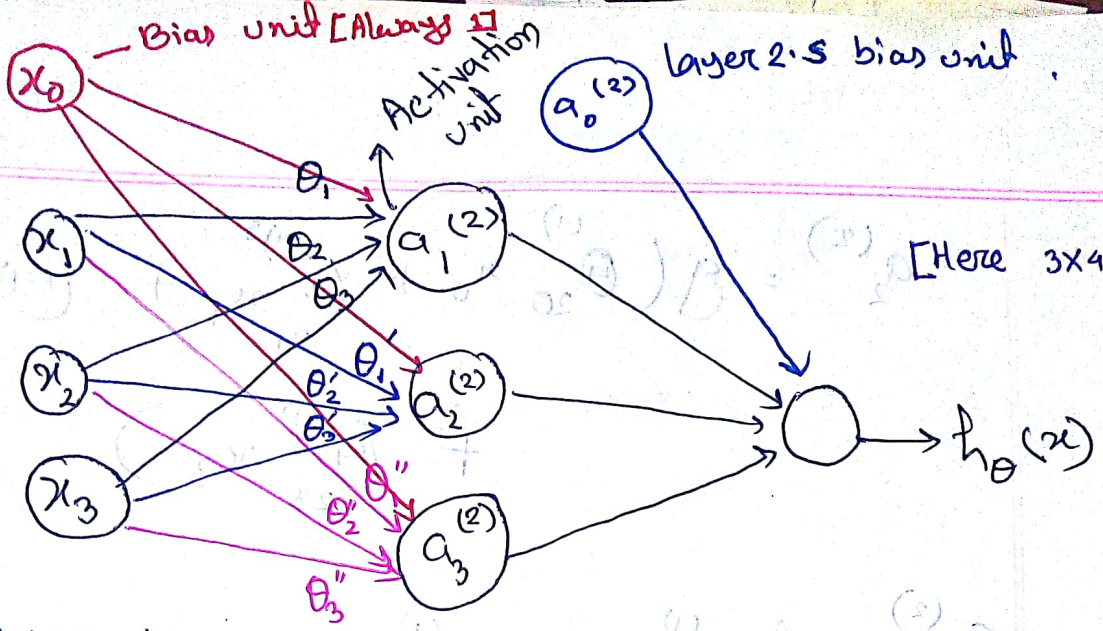
$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$ Parameter / weight.



"Sigmoid Activation function"

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

18



layer 1 (input)

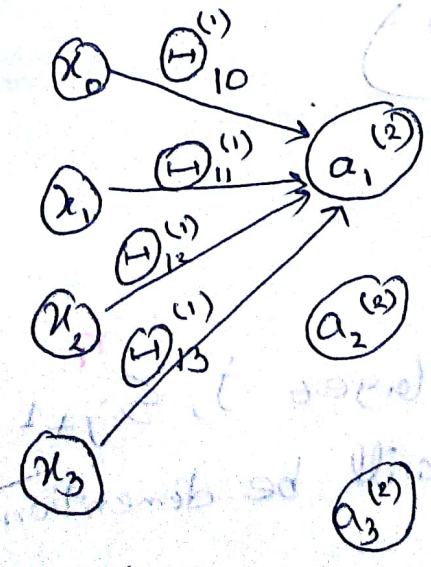
layer 2 (Hidden)

layer 3 (output)

(one or more than one hidden layer)

$a_i^{(j)}$ = "activation" unit i of j^{th} layer.

$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer j to $j+1$



$$a_1^{(2)} = g(z)$$
$$= g(\theta^T x)$$
$$= g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3\right)$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3\right)$$

$$h_\Theta(x) = g(z)$$

$\rightarrow$ layer 3

$$= a_1^{(3)}$$

$\rightarrow$ unit 1

$$= g\left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}\right)$$

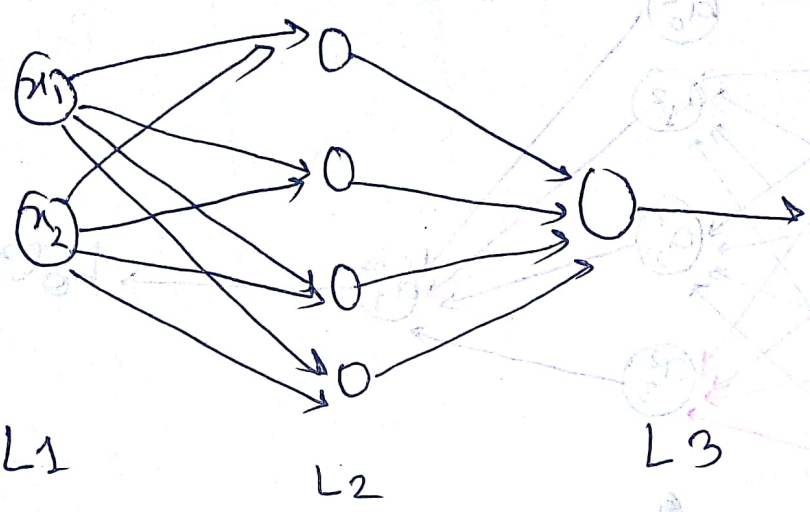
$$\Theta^{1} \in \mathbb{R}^{3*4}$$

* If a network has S_j (m) units in layer j, S_{j+1} (n) units in layer $j+1$, then $\Theta^{(j)}$ will be dimention of $S_{j+1} \times (S_j + 1)$

$n \times (m+1)$; if we calculate bias unit, else $n \times m$.

20

i.e.



if we use bias unit, Dimention would be

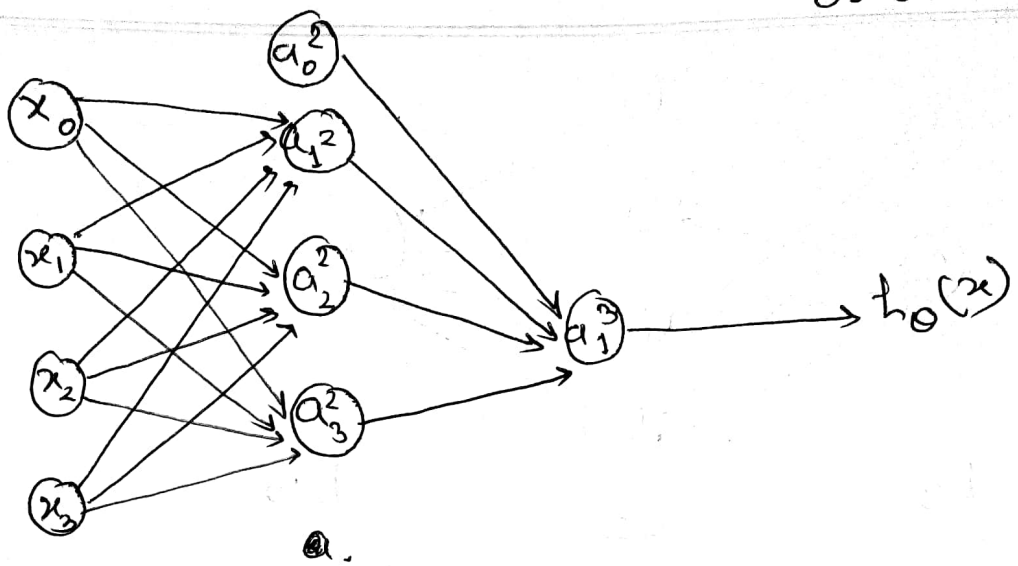
$(4 \times (2+1))$

$= 4 \times 3$

else, 4×2.

(Any)

01.07.18



@.

$$a_1^{(2)} = g(\theta_{10}^1 x_0 + \theta_{11}^1 x_1 + \theta_{12}^1 x_2 + \theta_{13}^1 x_3)$$

$$a_2^{(2)} = g(\theta_{20}^1 x_0 + \theta_{21}^1 x_1 + \theta_{22}^1 x_2 + \theta_{23}^1 x_3)$$

$$a_3^{(2)} = g(\theta_{30}^1 x_0 + \theta_{31}^1 x_1 + \theta_{32}^1 x_2 + \theta_{33}^1 x_3)$$

$$a_1^3 = g(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)})$$

Using logistic Function $g(z)$

$$a_1^{(2)} = g(z_1^{(2)})$$

$$a_2^{(2)} = g(z_2^{(2)})$$

$$a_3^{(2)} = g(z_3^{(2)})$$

$$z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

$$z^{(2)} = \Theta^{1} x$$

$\mathbb{R}^3$

3×4

$$a^{(2)} = g(z^2)$$

$\mathbb{R}^3 \quad \mathbb{R}^3$

For layer $j = 2$ and k unit,

$$z_k^{2} = \Theta_{k0}^{(1)} x_0 + \Theta_{k_1}^{(1)} x_1 + \ldots.$$

$$z^3 = \Theta^{(2)} a^{(2)}$$

$\mathbb{R} \quad \mathbb{R} \quad 3$

1×4

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad z^j = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ \vdots \\ z_n^{(j)} \end{bmatrix} = z^{(j)} = \Theta^{(j-1)} a^{(j-1)}$$

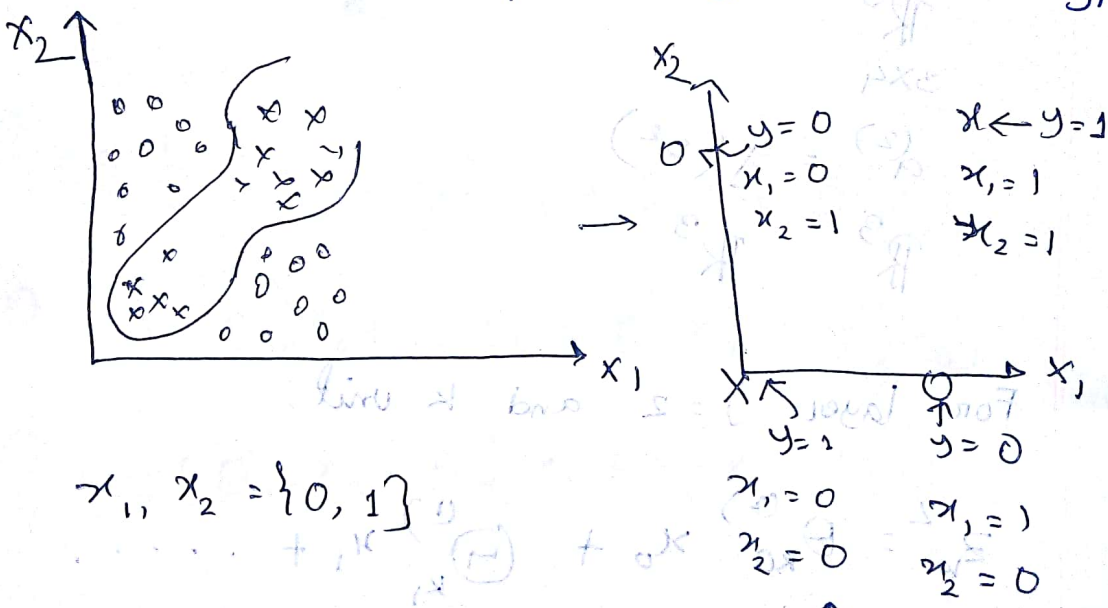
$$\Rightarrow a^j = g(z^{(j)})$$

$$a_1^3 = g(z_1^3)$$

$$= \frac{1}{1+e^{-z_1^3}}$$

(logistic Regression)

Q. Why neural networks is imp. to design non-linear hyp.?



$x_1, x_2 = \{0, 1\}$

x_1 on x_2

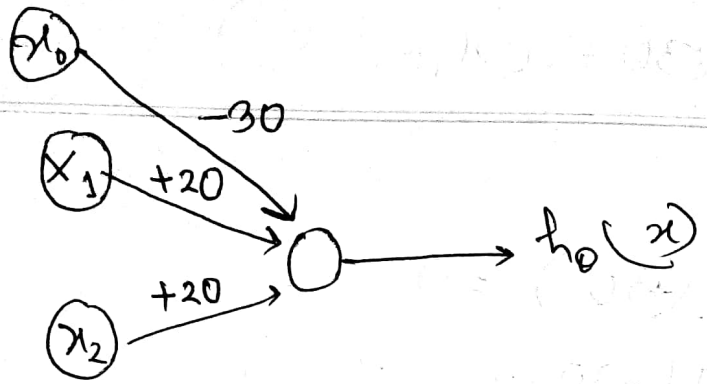
NOT (x_1 XOR x_2)

x_1 XNOR x_2

AND, OR, NOT

x_1	x_2	$x_1 \oplus x_2$	$\neg(x_1 \oplus x_2)$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

24

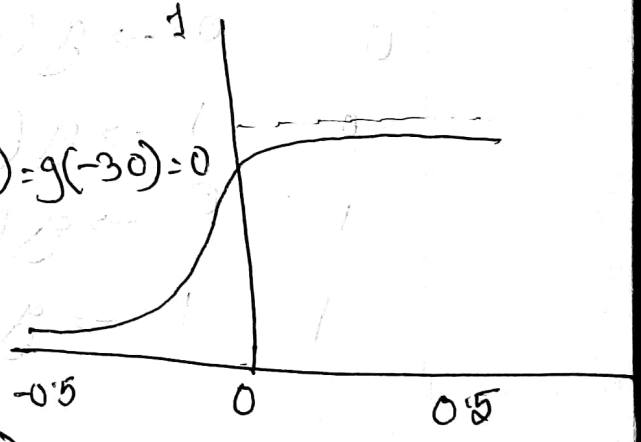


$$h_\theta(x) = g(z_1^{(2)})$$

$$= g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2)$$

$$= g(-30 + 20x_1 + 20x_2)$$

x_1	x_2	
0	0	$\rightarrow g(-30+20.0+20.0) = g(-30) = 0$
0	1	$\rightarrow g(-30+20) = g(-10) = 0$
1	0	$\rightarrow g(-30+20) = g(-10) = 0$
1	1	$\rightarrow g(-30+20+20) = g(10) = 1$



$g(z) = 1$ if $z \geq 0$

x_1 AND x_2

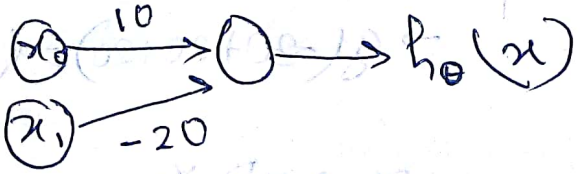
0
0
0
1

$g(-30 + 20x_1 - 20x_2)$

x_1	x_2	
0	0	$\rightarrow g(-30) = 0$
0	1	$\rightarrow g(-30-20) = g(-50) = 0$
1	0	$\rightarrow g(-30+20) = g(-10) = 0$
1	1	$\rightarrow g(-30+20-20) = (-30) = 0$

$g(-10 + 20x_1 + 20x_2)$

x_1	x_2	
0	0	$\rightarrow g(-10) = 0$
0	1	$\rightarrow g(10) = 1$
1	0	$\rightarrow g(10) = 1$
1	1	$\rightarrow g(30) = 1$



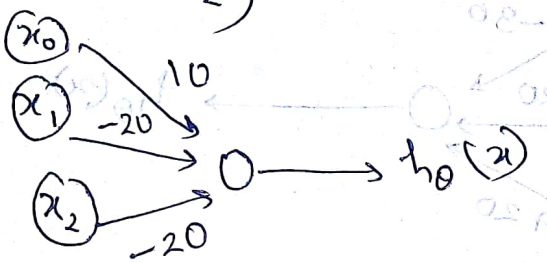
x_1	
0	$\rightarrow$ ~~$g(0-20) = g(0)$~~ $g(10) = 1$
1	$\rightarrow g(10-20) = 0$

Assignment :-

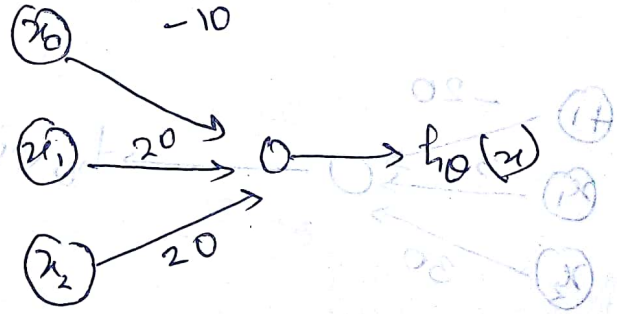
26

(NOT x_1 AND NOT x_2)

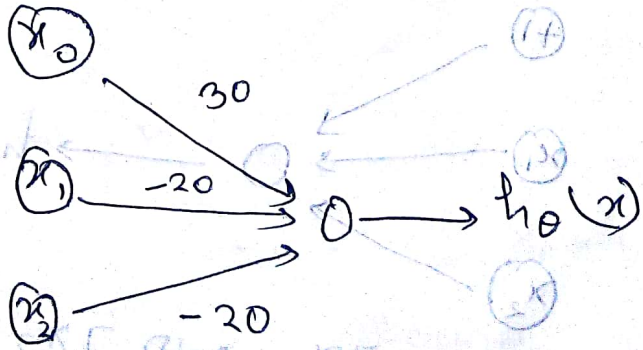
(a)



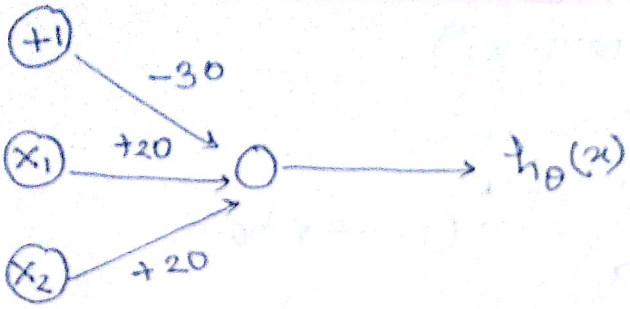
(b)



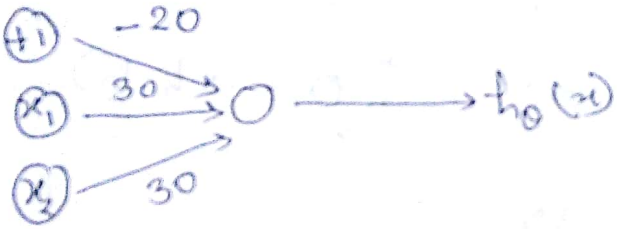
(c)



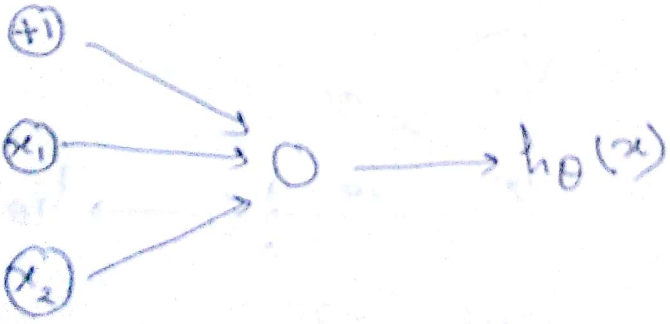
Scanned by CamScanner



AND

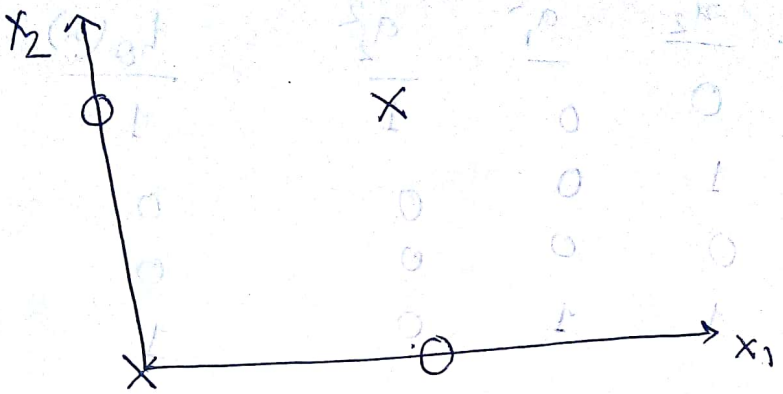


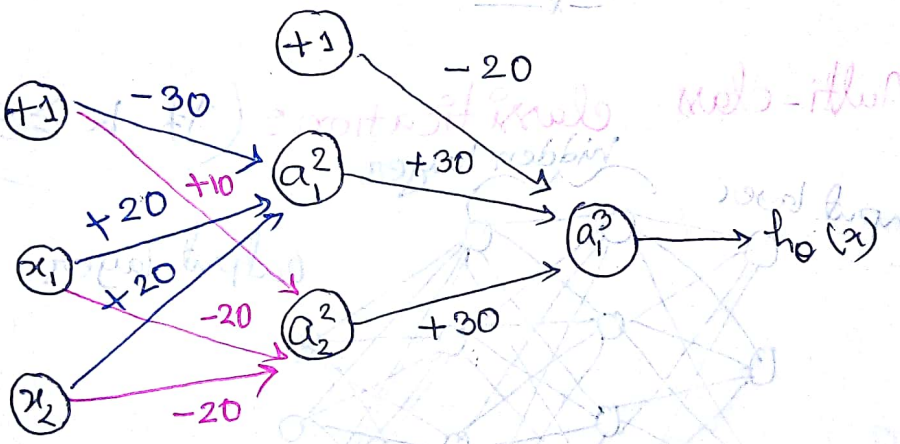
OR



$\overline{x_1}$ AND $\overline{x_2}$

Assignment No. 28





(Forward propagation)

$a_1^2 = 1$ iff $x_1 = x_2 = 1$

otherwise 0

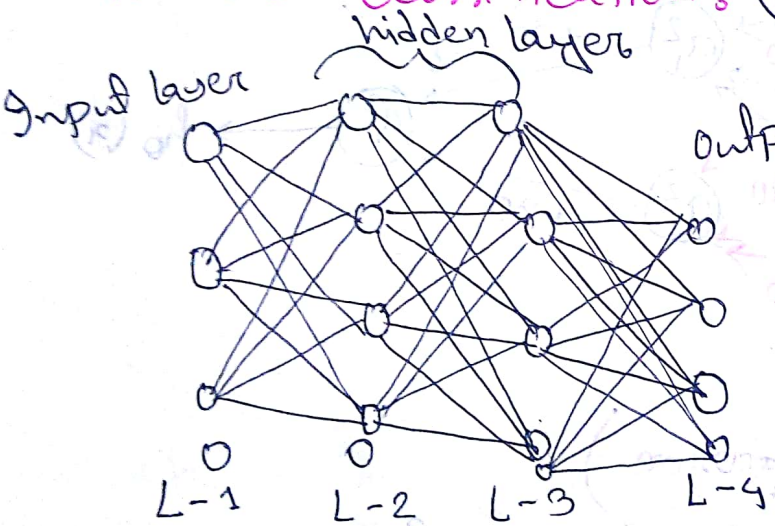
$a_2^2 = 1$ iff $x_1 = x_2 = 0$

otherwise $= 0$

x_1	x_2	a_1^2	a_2^2	$h_\theta(x)$
0	0	0	1	1
0	1	0	0	0
1	0	0	0	0
1	1	1	0	1

—x—

Multi-class classification: (if $k \geq 3$)



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

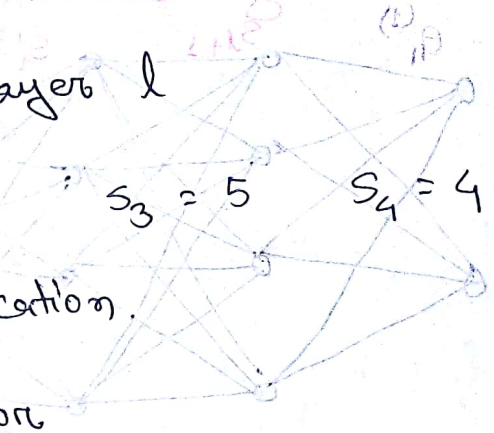
car truck Human motor cycle.

30

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \ldots, (x^{(n)}, y^{(n)})\}$$

L = # layers in network

s_l = # unit in layer l



$s_1 = 4 \quad s_2 = 5 \quad s_3 = 5 \quad s_4 = 4 = s_L$

K = K class classification.

$K = h_\theta(x) \rightarrow$ vector $\rightarrow \mathbb{R}^K$

Binary classification

$s_l = ?\ 1$

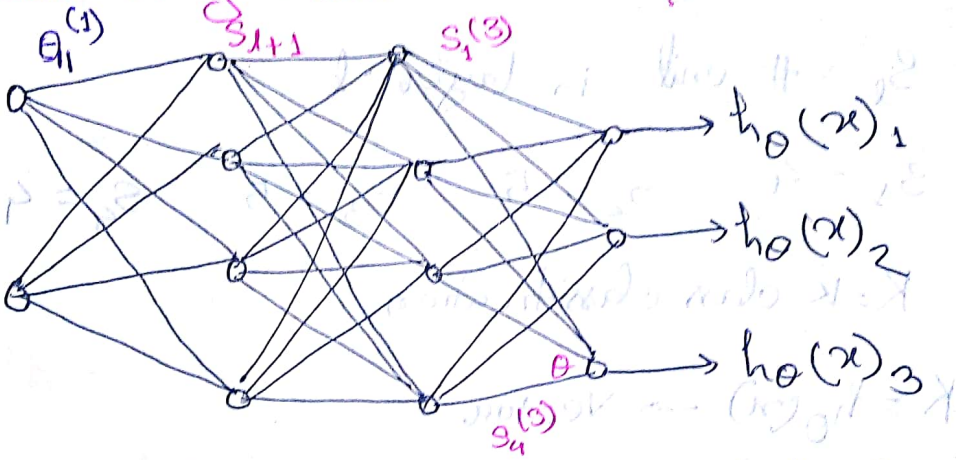
$h_\theta(x) = 1$ *

* If two class, then Binary classification.

08.07.18.

Today's Topic:

→ How to find out Cost function and Gradient Descent using Neural Networks?



K = 3
multiclass classification.

* Cost function:

Logistic Regression:

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\Theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\Theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n} \Theta_j^2$$

$h_\theta(x) \in \mathbb{R}^K \quad (h_\theta(x))_i$ ith output unit.

* Neural Network:

$$J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\theta(x^{(i)})\right)_k + \left(1 - y_k^{(i)}\right)\log\left(1 - \left(h_\theta(x^{(i)})\right)_k\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\theta_{ji}^{(l)}\right)^2$$

The double sum adds up the logistic regression costs calculated for each cell in output layer.

The triple adds up the square of all the individual Θs in the entire network, except all Θ_{j0} → because i starts with 1, so Θ_{j0} will be eliminated.

i^{th} feature

l^{th} layer

j^{th} unit.

* Gradient Descent:

$\min J(\theta) \mid m = 1$ → example.

Forward propagation.

$a^{(1)} = x$

$z^{(2)} = \Theta^{(1)} a^{(1)}$

$a^{(2)} = g(z^2)$ add $(a_0^{(2)})$

$z^3 = \Theta^{(2)} a^{(2)}$

$a^{(3)} = g(z^3)$

$z^4 = \Theta^{(3)} a^{(3)}$

$a^4 = g(z^4) = h_\theta(x)$

* Gradient computation: Back propagation algorithm:

Intution: $\delta_j^{(l)}$ = error of node j in layer l.

For each output unit ($l = 4$)

$\delta_j^{(4)} = a_j^{(4)} - y_j \;//\; \delta^{(4)} = a^{(4)} - y$

last layer calculates the distance ...

$\delta^{(3)} \rightarrow \delta_1^{(3)}\ \delta_2^{(3)}\ \delta_3^{(3)}\ \delta_4^{(3)}$

$\delta^{(3)} = \left(\Theta^{(3)}\right)^T \delta^{(4)} .* g'(z^{(3)})$

$g'(z^{(3)})$ is the ~~derivation~~ derivative of activation function $g(z^{(3)})$

.* = element wise multiplication

$g'(z^{(3)}) = a^{(3)} .* (1 - a^{(3)})$

$\delta^{(3)} = \left(\Theta^{(3)}\right)^T \delta^{(4)} .* a^{(3)} .* (1 - a^{(3)})$

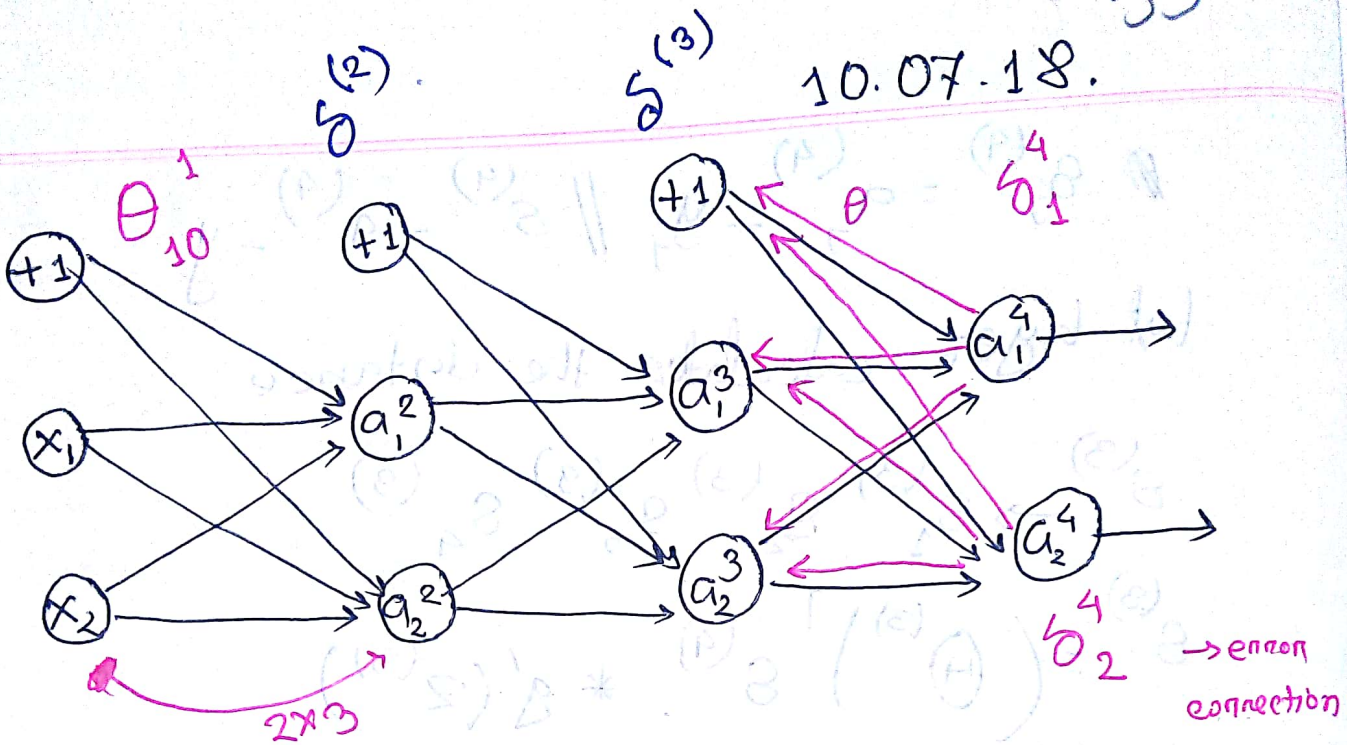
$\delta^{(2)} = \left(\Theta^{(2)}\right)^T \delta^{(3)} .* a^{(2)} .* (1 - a^{(2)})$

$\delta^{(1)}$ ~~ed~~ → no need to calculate, because input has no error.

$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\theta) = a_j^{(l)} \delta_i^{(l+1)}$

[Ignoring λ, if $\lambda = 0$].

10.07.18.

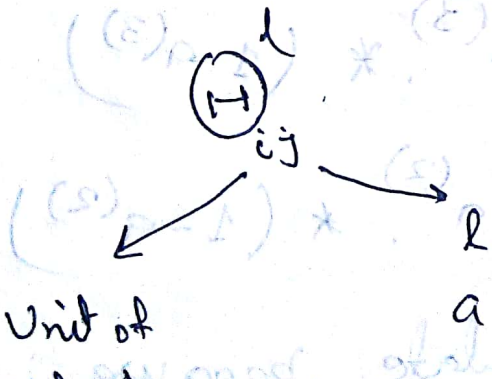


Cost Function: $J(\theta)$

$\min_{\theta} J(\theta)$ $\quad \frac{\partial}{\partial \theta} J(\theta)$

layer 1 to layer 2

2's Unit × 1's unit + 1



$\delta^{(l)}_j$ = l layer, j unit is error.

$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\frac{\partial}{\partial \Theta_{ij}^{1}} J(\theta)$$

BFSG

↓ takes

$J(\theta)$

& partial derivative as input.

For layer -1

Θ_{10}^{1} Θ_{20}^{1}

Θ_{11}^{1} Θ_{21}^{1}

Θ_{12}^{1} Θ_{22}^{1}

$$\delta^{(4)} = a_j^{(4)} - y_j = a^{(4)} - y$$

$$\delta^{(3)} = (\theta^{(3)})^T \delta^{(4)} .* g'(z^{(3)})$$

$$= (\theta^{(3)})^T \delta^{(4)} .* a^{(3)} .* (1 - a^{(3)})$$

(Back propagation)

$$\frac{\partial}{\partial \Theta_{ij}^{l}} J(\theta) = a_j^{l} \delta_i^{(l+1)}$$

Back propagation algorithm:

Given training set $\{(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})\}$

- Set $\Delta_{ij}^{(l)} := 0$ for all (l, i, j)

  ↳ use to compute $\frac{\partial}{\partial \Theta_{ij}} J(\Theta)$

- For $t = 1$ to m

  1. Set $a^{(1)} := x^{(t)}$

  2. Perform forward propagation to compute $a^{(l)}$ for $l = 2, 3, \ldots, L$

x_1	a_1^1
x_2	a_2^1
x_3	a_3^1

  3. Using $y^{(t)}$, compute $\delta^{(L)} = a^{(L)} - y^{(t)}$

  4. Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots \delta^{(2)}$ using formula,

$$\delta^{(l)} = \left((\Theta^{(l)})^T \delta^{(l+1)}\right) .* a^{(l)} .* (1 - a^{(l)}) \quad \text{/} \quad g'(z^{(l)})$$

5. $\Delta_{ij}^{(l)} := \Delta_{ij}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

[Vector form,
$\Delta^{(l)} = \Delta^{(l)} + \delta^{(l+1)} (a^{(l)})^T$]

Hence we update our new Δ matrix (out of for loop)]

- $D_{ij}^{(l)} := \frac{1}{m} \left(\Delta_{ij}^{(l)} + \lambda \Theta_{ij}^{(l)} \right)$ if $j \neq 0$

(Cost function → feature scaling to avoid overfitting problem)

- $D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}$ if $j = 0$

* The capital-delta matrix D is used as an "accumulator" to add up our values as we go along and eventually compute our partial derivative, this we get,

$$\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta) = D_{ij}^{(l)}$$

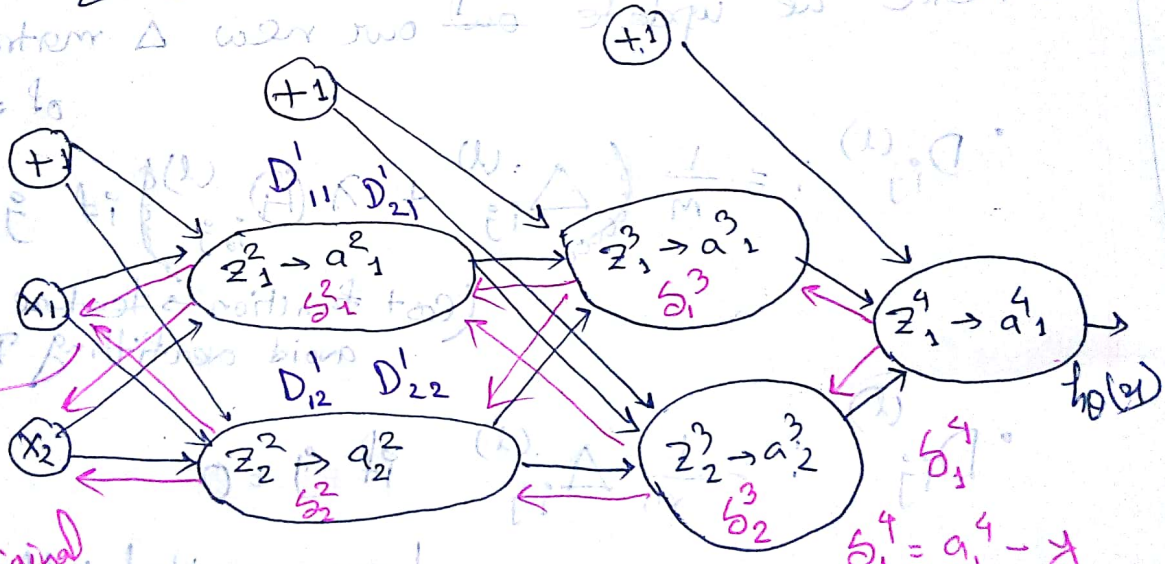
To reduce Cost function, $J(\theta)$:

→ Gradient Descent

→ Conjugate GD

→ BFGS

→ L-BFGS.



not needed to calculate, because original input has not error.

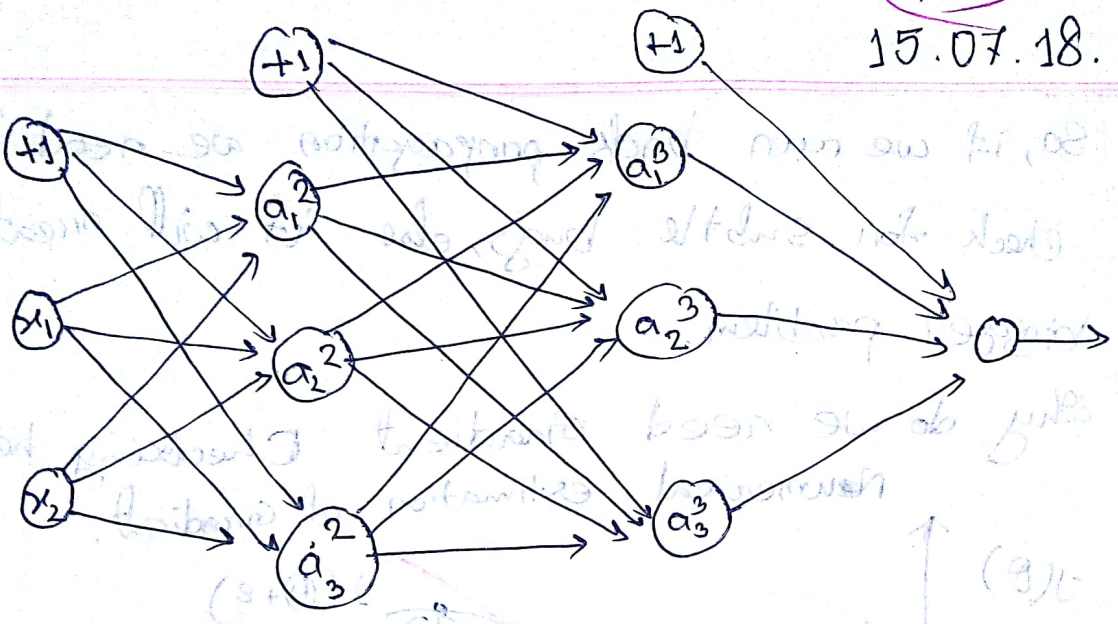
$\delta^4_1 = a^4_1 - y$

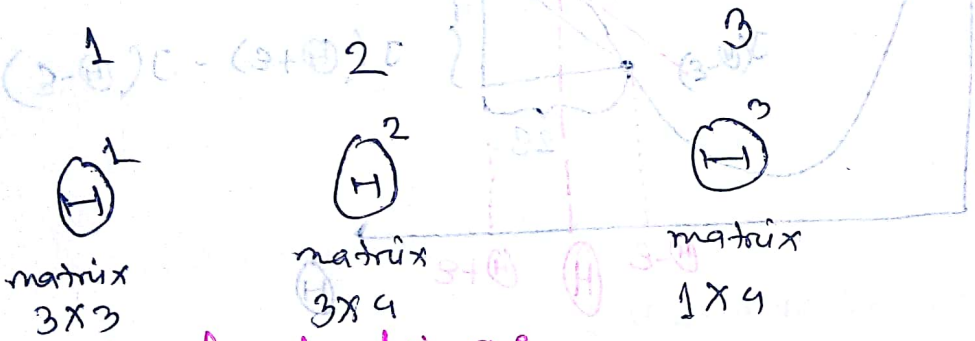
$z^3_1 = \Theta^2_{10} \cdot 1 + \Theta^2_{11} a^2_1 + \Theta^2_{12} a^2_2$

—x—

(Final)

15.07.18.





Gradient checking:

We know → How to do forward propagation

→ " " " backward "

Backward propagation

→ has a lot of details

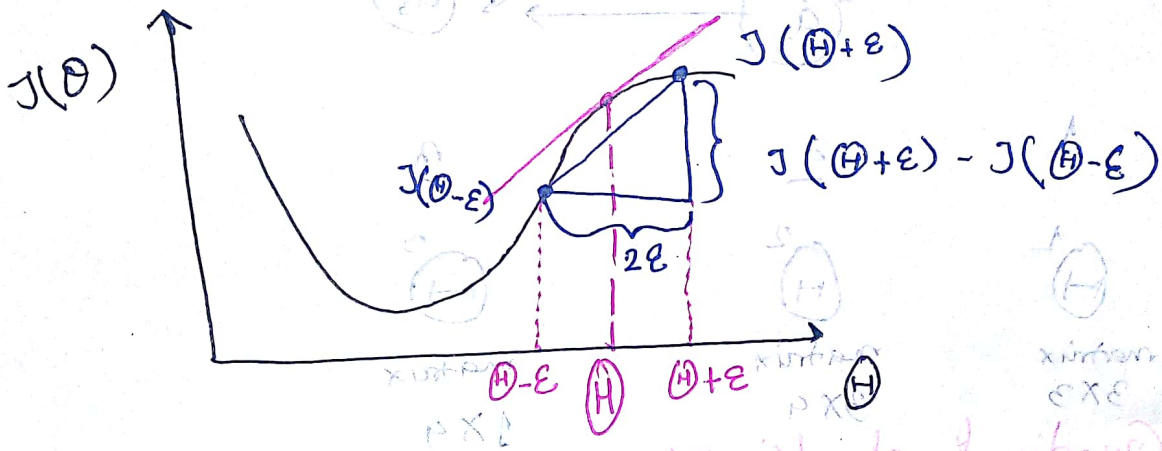
→ tricky to implement

→ there are many ways to have subtle bugs in back propagation.

So, if we run back propagation we need to check for subtle bugs, else it will create bigger problem.

* Why do we need Gradient Checking here?

Numerical estimation of Gradient.



$$\theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \qquad \tan\theta = \frac{y}{x}$$

$$\frac{d}{d\theta} J(\theta) \approx \frac{J(\theta+\varepsilon) - J(\theta-\varepsilon)}{2\varepsilon} \quad \Big| \quad \varepsilon = 10^{-4}$$

Parameter Vector θ

Unrolling Θ:

$\Theta^{(1)} = 10 \times 11$

$\Theta^{(2)} = 10 \times 11$

$\Theta^{(3)} = 1 \times 1$ $\quad \Theta^{(1)}_{11}$

thetavector = [theta1(:), theta2(:), theta3(:)]

1-110 111-220 221-232

thetavector[1]

Back to matrix,

theta1 = reshape (thetaVector (1:110), 10, 11)

theta2 = reshape (thetavector (111:220), 10, 11)

Parameter vector Θ

Let, $\Theta \in \mathbb{R}^n$ (Θ is unrolled version of $\Theta^{(1)}, \Theta^{(2)}, \ldots, \Theta^{(n)}$)

$\Theta = [\Theta_1, \Theta_2, \Theta_3, \ldots, \Theta_n]$

$\frac{\partial}{\partial \Theta_{10}} J(\Theta)$

D_{10} $$\frac{\partial}{\partial \theta_1} J(\theta) \approx \frac{J(\theta_1+\varepsilon, \theta_2, \theta_3, \ldots, \theta_n) - J(\theta_1-\varepsilon, \theta_2, \theta_3, \ldots, \theta_n)}{2\varepsilon}$$

D_{11} $$\frac{\partial}{\partial \theta_2} J(\theta) \approx \frac{J(\theta_1, \theta_2+\varepsilon, \theta_3, \ldots, \theta_n) - J(\theta_1, \theta_2-\varepsilon, \theta_3, \ldots, \theta_n)}{2\varepsilon}$$

$$\vdots$$

D_{1m} $$\frac{\partial}{\partial \theta_n} J(\theta) \approx \frac{J(\theta_1, \theta_2, \ldots, \theta_n+\varepsilon) - J(\theta_1, \theta_2, \ldots, \theta_n-\varepsilon)}{2\varepsilon}$$

```
for i = 1 : m
    thetaP = theta
    thetaP(i) = thetaP(i) + EPSILON
    thetaM = theta
    thetaM(i) = thetaM(i) - EPSILON
    gradApprox = (J(thetaP) - J(thetaM))/2*EPSILON
end;
```

check that,

gradApprox $\approx$ D Vec

* Implementation Note:

1. Implements back propagation to compute D.
   (Unrolled $D^1, D^2, \ldots, D^n$)

2. Implement gradient check to compute gradApprox.

3. Make sure they give similar result.

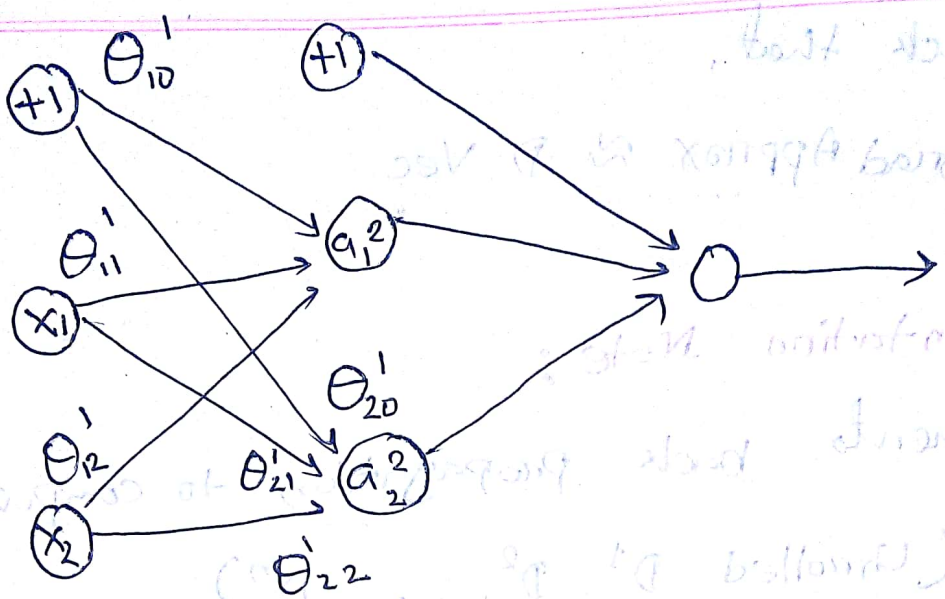
4. Turn off gradient checking, using back propagation code for learning, train your algorithm

Random Initialization:

→ Initiall all the Θ's to 0 does not work with neural networks.

→ When back propagation all nodes will update to the same value repeatedly.

→ Instead we can randomly initialize our Θ's for our Θ matrix.



$a_1^2 = \theta_{10}^{1} \quad \theta_{11}^{1} \quad \theta_{12}^{1}$ initialize

if $0 \quad 0 \quad 0$

$a_2^2 = \theta_{20}^{1} \quad \theta_{21}^{1} \quad \theta_{22}^{1}$

if $0 \quad 0 \quad 0$

$a_1^2 \;,\; a_2^2$

$\downarrow \qquad \downarrow$

$\theta^T x \quad \theta^T x$

$a_1^2 = a_2^2$

17.07.18.

Summary (all together):

First, pick a Network Architecture. i.e. 3 5 4 / 3 4 4 4 / 3 5 5 5 4

↳ choose the layout of your Neural Network.

↳ # hidden units in each layer

↳ # layers in total Network.

- # input units = dimention of features ($x^{(i)}$)
- # output units = # classes
- # hidden units / layer = (*1)

Second, Training a Neural Network.

1. Randomly initialize the Θs.
2. Implement forward Propagation to get $h_\Theta(x)$ for any $x^{(i)}$
3. Implement the cost function $J(\Theta)$
4. Implement Back propagation to compute partial derivative. $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\Theta)$ (*2)

Symetry Breaking: Random Initialization.

Initial each $\Theta_{ij}^{(l)}$ to random value in

$[-\varepsilon, \varepsilon]$.

i.e. $-\varepsilon \leq \Theta_{ij}^{(l)} \leq \varepsilon$

1

- # Hidden unit / layer = Usually more is better.
  must balance with cost as it increase with more Hidden Unit.

- Default: 1 hidden layer.
  If you have more than one hidden layers, you should have same Hidden units in every Hidden layers. (Suggested)

2

when perform Back propagation & forward propagation, we loop every training example:

for $i = 1 : m$,

perform FP & BP using $(x^{(i)}, y^{(i)})$

(Get activations $a^{(l)}$, delta terms $\delta^{(l)}$ for $l = 2, 3, \ldots L$)

↓ for

Then compute $D^{(1)}$ out of for loop.

5. Use gradient checking to compare $\frac{\partial}{\partial \theta_{ij}} J(\theta)$. Compute using BP vs using numerical estimation of gradient of $J(\theta)$ to confirm that your BP works. Then disable gradient checking.

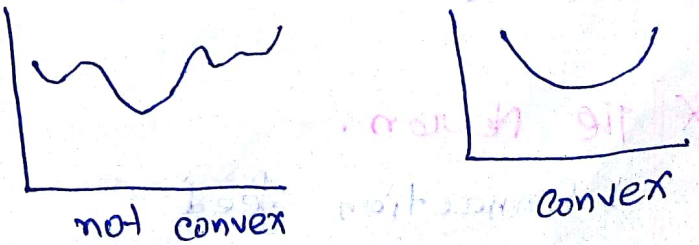
6. Then use GD or a built in optimization function to minimize the cost function with weight in theta.

→ $J(\theta)$ in NN is not a convex function.

It could be local minimum as well as global minimum.

↓

a good local optimal can get.



CT-3

—X—

"Special class" 19.07.18.

Neural Networks:

Zain Protein Molecule.

Human's Unsupervised learning → Imagination (new, innovation).

Architecture:

1. single - layer feed-forward
2. multi - "
3. Recurrent -

Que → How to choose hidden layer's unit or node?

Data preprossing → fit data in model.

Recurrent network:

* The Neuron.

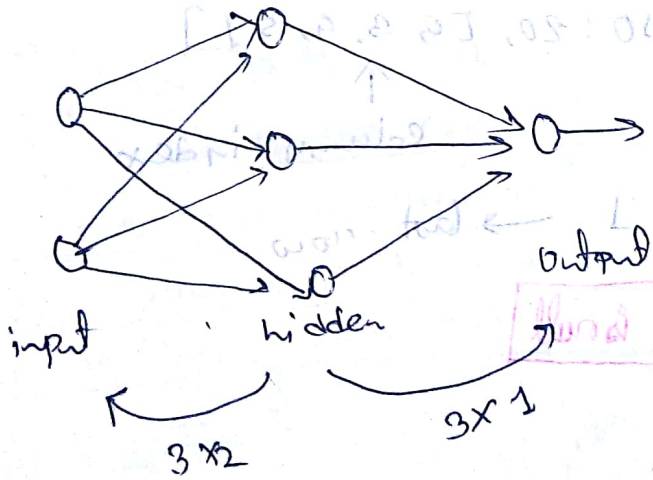
1. Connection feed
2. Adder
3. Activation

Dimensions of a NN:

1. Neurons
2. Architecture
3. learing algo.
4. Applications.

* learning Algorithm: Backpropagation:

$$w'_{46} = w_{46} + \eta \delta \frac{d f_6(e)}{d e} y_4$$



"Python"

Numpy → Numerical python

Scipy → Scientific python

pandas

Scikit - learn

matplotlib

~~Seaborn~~ Seaborn.

jupyter notebook
↓
1st p to implement ML.

Pandas → all data processing

[10:20, [0, 3, 4, 5]]
↑
column index

df.iloc[-1] → last row

isnull

[

Linear regression.

epoke size

Association Rule Mining:

↓

Market Basket Analysis

1. Identify frequent item set → based on "support."
2. Based on frequent item set, discover association rule → satisfying "confidence."

i.e.: Consider the following transactions:

t_1	Bread, Butter, Milk
t_2	Bread, Butter, Coke, Diaper
t_3	Coke, Diaper, Butter
t_4	Bread, Milk, Coke, Diaper
t_5	Coke, Butter, Milk

requirement: let,
(support) Minimum support = 60% (of total transaction)
Minimum Confidence = 70%

Single item:

$\{Bread\} = \frac{3}{5} = 0.6$ ✗

$\{Butter\} = \frac{4}{5} = 0.8$ ✓

{Milk} = $\frac{3}{5}$ = 0.6 ✗

{Diaper} = $\frac{3}{5}$ = 0.6 ✗

{Coke} = $\frac{4}{5}$ = 0.8 ✓

∴ {Butter}, {coke}

Two items:

{Butter, coke} = $\frac{3}{5}$ = 0.6 ✗

Association Rule: (confidence)

{Butter → coke} = $\frac{3}{4}$ = 0.75 (75% chance who buys butter, also buys coke)

↑

(Apriori Algorithm) → (FP Growth Algorithm (Best ⇒ frequent pattern (FP) (Faster))

$$= \frac{\text{How many Butter, coke}}{\text{How many Butter}}$$

Min support & Min confidence
↳ depends on Customers requirements.

could be

{ Butter → Diaper, coke }

{ coke → Butter, Diaper }

{ Diaper → Butter, coke }

* All possible combination depends on min. support.

Butter, Diaper, Coke

Butter

—×—

"Decision tree"

* Decision tree classification:

↓

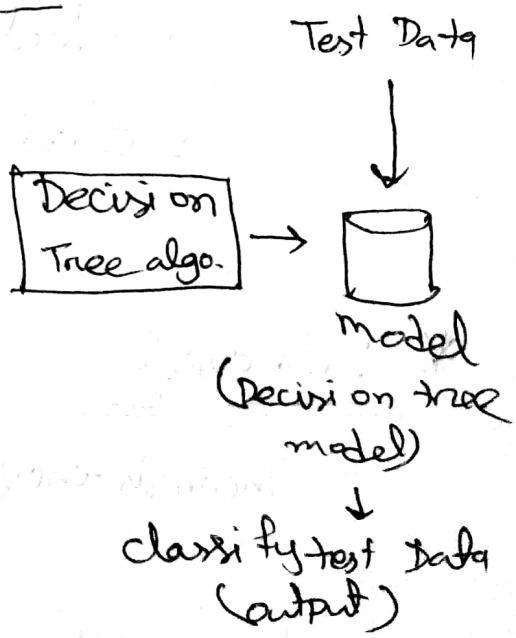
Supervised learning

→ Phases

Input training set

(Training Data)

Inference → to model creation



i.e.: Example training Data:

SL	Colors	Age	class
1	White	15	Tender
2	White	20	Tender
3	Gray	25	Midranger
4	Gray	30	Midranger
5	Red	35	Mature
6	Red	36	Mature

1. Calculate "Entropy"
2. Calculate "Information gain" for each attribute except the class.

* Entropy:

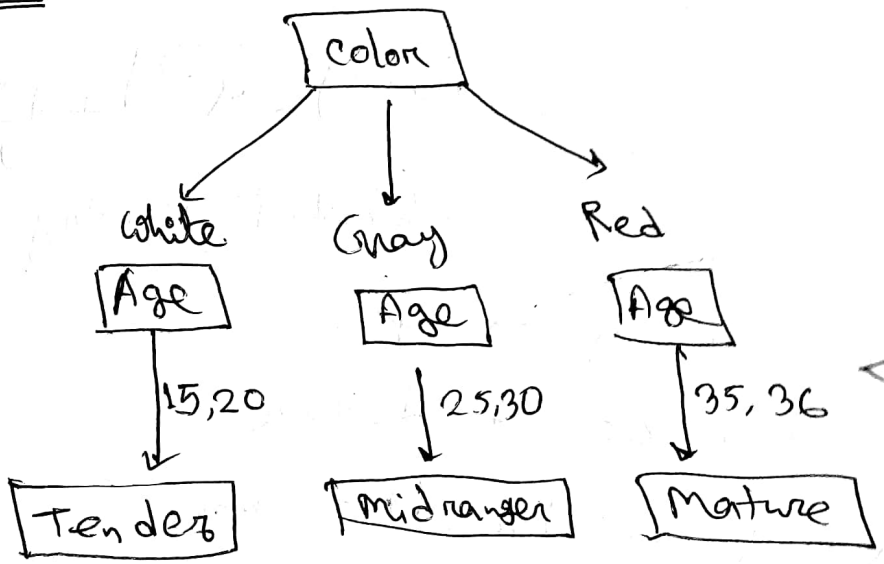
→ Inconsistency in data. [How entropy is expected]

→ Higher the entropy, the higher the inconsistency.

Information Gain:

→ To decide which attribute goes where in the tree.

Trees:



← Decision tree

New Data:

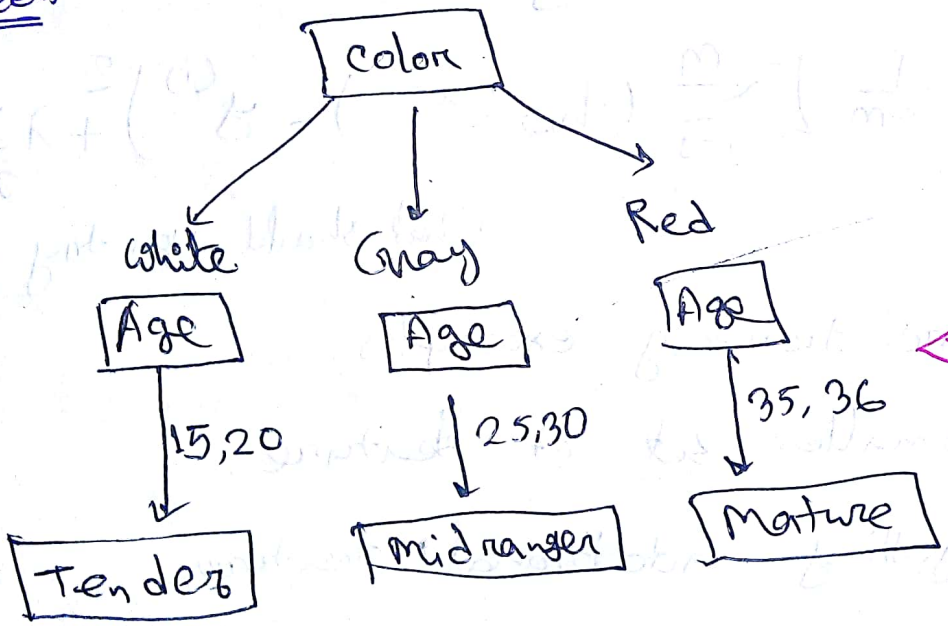
Red 32 (Not available)

—x—

Information Gain:

→ To decide which attribute to use as 'root' of the tree.

Tree:



← Decision tree

New Data:

Red 32 (Not available)

—x—

22.07.18.

→ ML system Design :

→ Understand multiple parts.

→ How to deal with skew data.

* Regularized linear Regression (LR):

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

what should you try next?

→ Get more training examples.

→ Try smaller set of feature.

→ Try getting additional features.

→ Try adding polynomial features (x_1^2, x_2^2, x_1, x_2 etc).

→ Decrease λ

→ Increase λ

Do not try anything randomly

ML Diagnostic:

A test that you can run to gain insight what is/is not working with a learning algorithm, and gain guidance as to how best to improve its performance.

Evaluating Hypothesis: (HT):

→ A HT may have lower error for the training examples, but still be inaccurate. (i.e. overfitting problem)

→ Thus evaluate a HT, given a data set of training example, we can split up the data into two sets,

(I) Training set: (70% Data may be 1st)

(II) Test set: (30% " " last)

x^1, y^1

x^2, y^2

$\vdots$

x^m, y^m

} training Set (70%)

x^1_{test}, y^1_{test}

x^2_{test}, y^2_{test}

$\vdots$

x^m_{test}, y^m_{test}

} Test Set (30%).

→ New procedure using these two set is then,

1. Learn Θ and minimize $J_{train}(\theta)$ using training set.

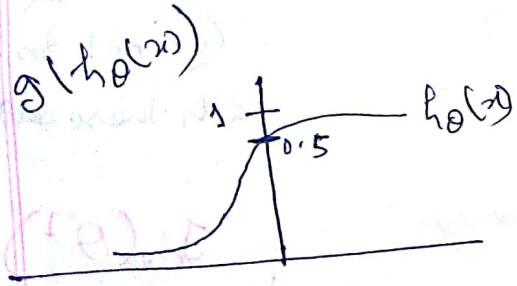
2. Compute the test set errors $J_{test}(\theta)$.

↳ LR

↳ LR : $J_{test}(\theta) = \frac{1}{2m_{test}} \sum_{i=1}^{m_{test}} \left(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)} \right)^2$

↳ classification Problem :

$$J_{test}(\theta) = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} \left[y_{test}^{(i)} \log h_\theta(x_{test}^{(i)}) + (1 - y_{test}^{(i)}) \log(1 - h_\theta(x_{test}^{(i)})) \right]$$



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

Missclassification error,

$$err(h_\theta(x), y) = \begin{cases} 1 \text{ if } h_\theta(x) \geq 0.5 \text{ \& } y = 0 \\ \text{or, } h_\theta(x) < 0.5 \text{ \& } y = 1 \\ 0 \text{ otherwise} \end{cases}$$

$$\text{Test error} = \frac{1}{m_{test}} \sum_{i=1}^{m_{test}} err\left(h_\theta(x_{test}^{(i)}) - y_{test}^{(i)} \right)$$

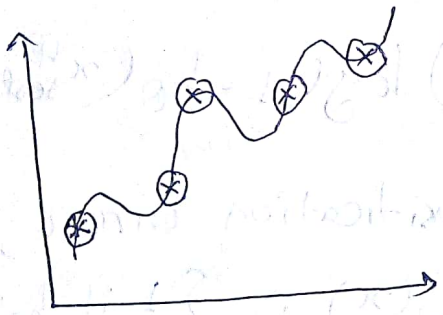
24.07.18.

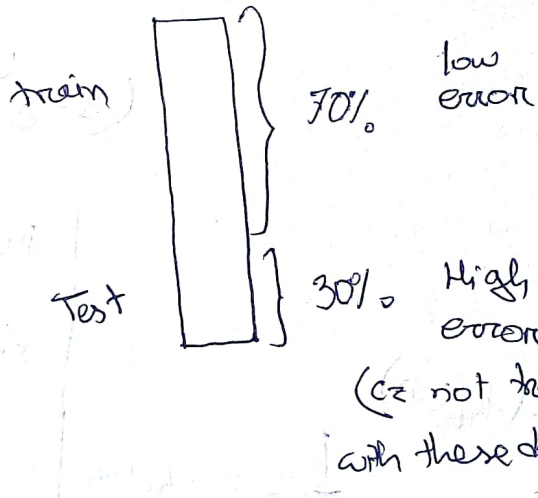
$\theta^T x$ θx

$\theta_0 + \theta_1 x_1 + \ldots\ldots + \theta_n x_n$

$\overset{0.5}{\theta_0} + \overset{1}{\theta_1 x_1} + \overset{3}{\theta_2 x_3}$

$J(\theta) \to \min$





Model selection:

$J_{cv}(\theta^1)$

d = 1 1. $h_\theta(x) = \theta_0 + \theta_1 x \to J_{train}(\theta^1) \to \theta^1 \to J_{test}(\theta^1)$

d = 2 2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \to J_{train}(\theta^2) \to \theta^2 \to J_{test}(\theta^2)$

d = 3 3. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 \to J_{train}(\theta^3) \to \theta^3 \to J_{test}(\theta^3)$

. . . . . .

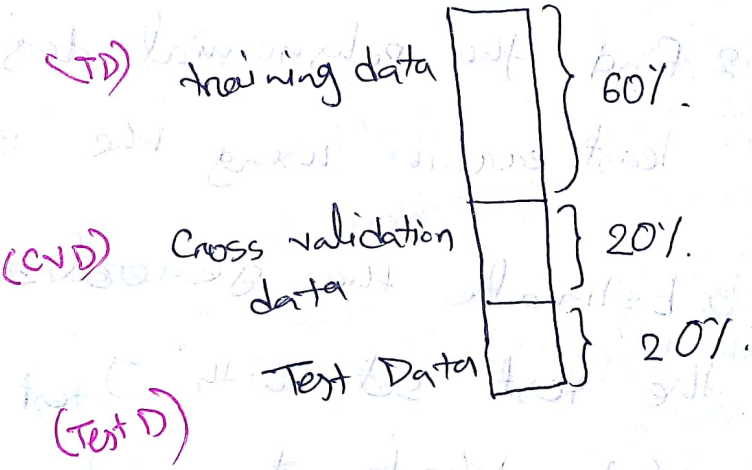
d = 10 10. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_{10} x^{10}$

$\to J_{train}(\theta^{10}) \to \theta^{10} \to J_{test}(\theta^{10})$

d = degree of polynomial

When, $d = 5$, $J_{test}(\theta^5)$ [assume] will give lowest value.

* d has been trained using test data (30%)



* Break down data set into three sets:
  - training set: 60%
  - Cross validation set: 20%
  - Test set: 20%

* We can now calculate three separate error values for the three different sets using the following method:

1. Optimize the parameters in (H) using the training set for each polynomial degree.

2. Find the polynomial degree d with the least error using the cross validation set.

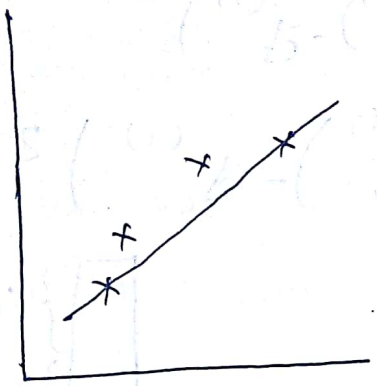
3. Estimate the generalization error using the test set with $J_{test}(\theta)$.
(d = theta from polynomial with lower error).

* "This way the degree of the polynomial d has not been trained using the test set."

—x—

29.07.18.

Diagnosing Bias Vs Variance:

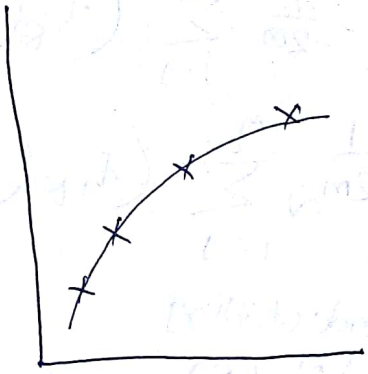


$\theta_0 + \theta_1 x$

High Bias

Underfit

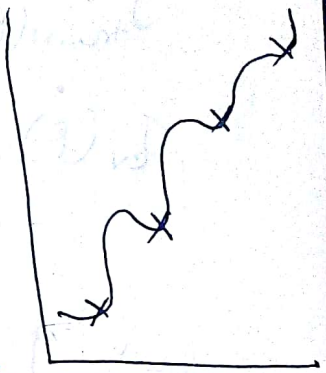
$d = 1$



$\theta_0 + \theta_1 x + \theta_2 x^2$

Just right

$d = 2$



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

overfit

High variance

$d = 4$

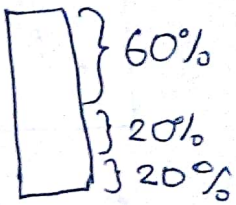
- Distinguish weither Bias or variance is the Problem for Bad prediction.
- High bias - underfit
- High variance - overfit
- golden ratio.

cv = Cross validation.

Bias/variance:

$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} \left(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)} \right)^2$$



For high Bias,

$J_{train}(\theta)$ & $J_{cv}(\theta)$ will be high

$J_{train}(\theta) \approx J_{cv}(\theta)$.

For high variance,

$J_{train}(\theta) \rightarrow$ low and $J_{cv}(\theta) \rightarrow$ High

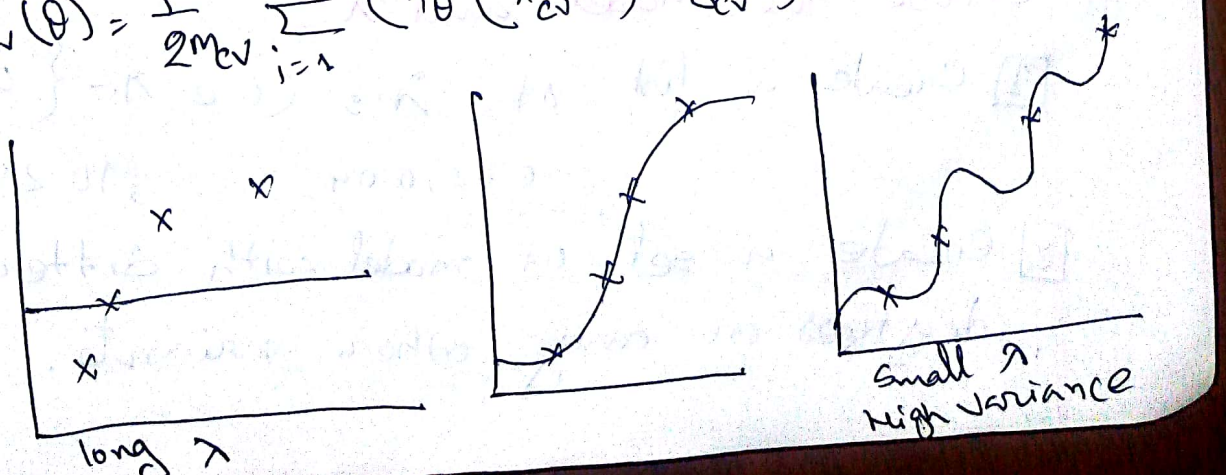
$J_{cv}(\theta) >> J_{train}(\theta)$.

Regularization & Bias/variance:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

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

$$J_{cv}(\theta) = \frac{1}{2m_{cv}} \sum_{i=1}^{m_{cv}} \left(h_\theta(x_{cv}^{(i)}) - y_{cv}^{(i)} \right)^2.$$



$\theta_1 \approx 0, \theta_2 = 0 \ldots \theta \approx 0$

$h_\theta(x) \approx \theta_0$

After selecting d, How to Judge λ?

1. Try $\lambda = 0 \rightarrow \min J(\theta) \rightarrow \textcircled{H}^{(1)} \rightarrow J_{cv}(\theta)$

without regularization

2. $\lambda = 0.01 \rightarrow \min J(\theta') \rightarrow \textcircled{H}^{(2)} \rightarrow J_{cv}(\theta^2)$

$\lambda = 0.02$ $J_{cv}(\theta^3)$

$\lambda = 0.04$ $J_{cv}(\theta^4)$

. . . . . .

$\lambda = 10.24 \rightarrow \min J(\theta) \rightarrow \textcircled{H}^{(10)} \rightarrow J_{cv}(\theta^{10})$.

λ, with less error will be selected.

Choose the model and λ:

1. Create a list of λ's (i.e. $\lambda \in \{0, 0.01, 0.02, 0.04, \ldots, 10.24\}$)
2. Create a set of model with different degrees or any others variants.

3 Iterate through the λ's and for each λ go through all models to learn Θ.

4 Compute the CV error using the learned Θ (compute with λ) on the $J_{cv}(\theta)$ without regularization or $\lambda = 0$

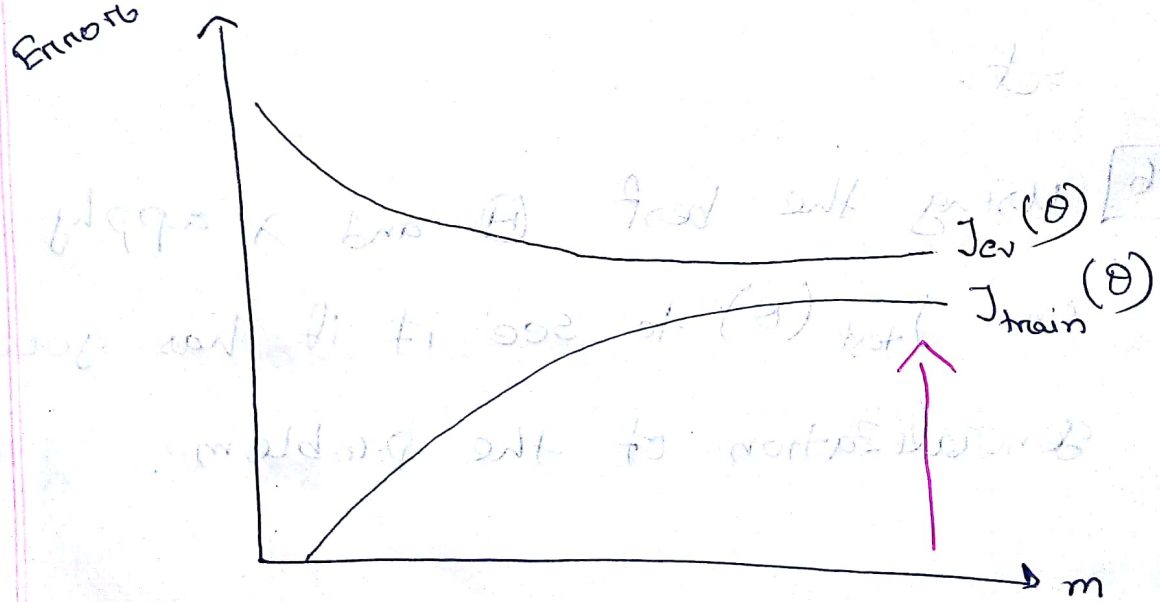
5 Select the best combination that produces the lowest error on the cross validation set.

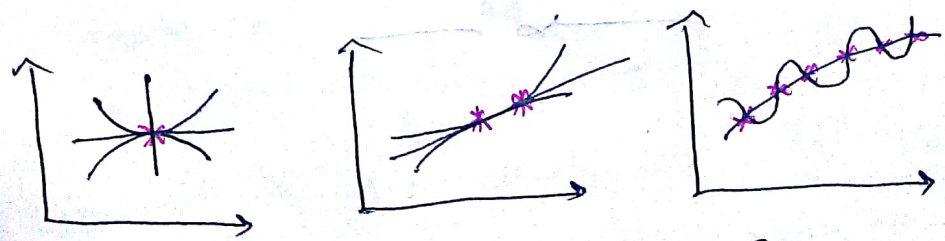
6 Using the best Θ and λ apply it on $J_{test}(\theta)$ to see if it has good generalization of the problem.

31.07.18

Learning Curve:

CV = Cross validation





$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

→ Train algo. on a very few data,

↳ easily have 0 error.

↳ can find a quadratic function that touches all point.

→ Training set larger,

↳ Error increases

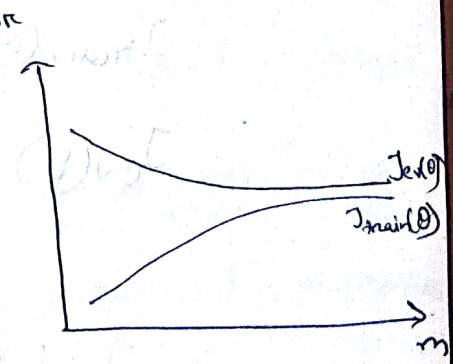
→ Error value will plateu out after a certain m.

Experiencing High Bias:

→ Lower training set size.

$J_{train}(\theta)$ → low

$J_{cv}(\theta)$ → High



→ Higher training set size.

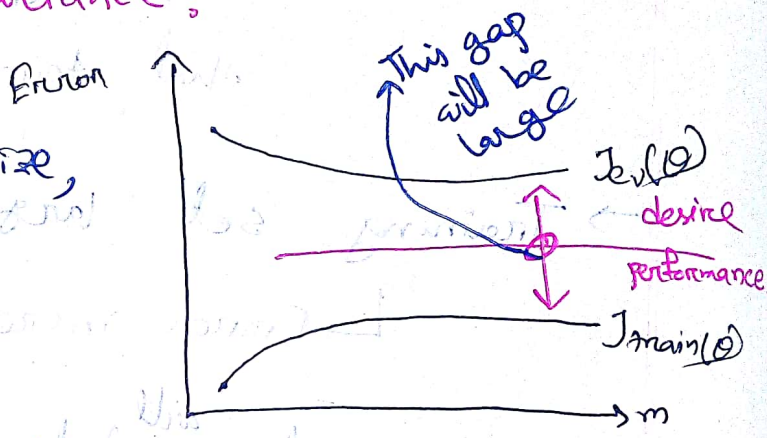
$J_{train}(\theta)$ → High

$J_{cv}(\theta)$ → High

$J_{cv}(\theta) \approx J_{train}(\theta)$

* If learning algo. is suffering from high bias, getting more data will not (by itself) help much.

Experiencing High Variance:

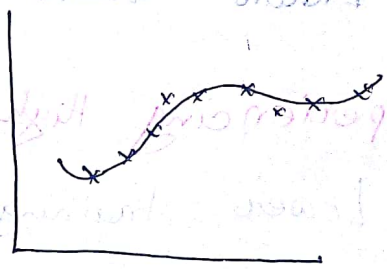


→ Lower training set size,

$J_{train}(\theta) \rightarrow$ low

$J_{cv}(\theta) \rightarrow$ High

→ Higher training set size,



$J_{train}(\theta) \rightarrow$ increase

$J_{cv}(\theta) \rightarrow$ Decrease (but not low)

$J_{cv}(\theta) >> J_{train}(\theta)$

$J_{cv}(\theta) \rightarrow$ Decrease without leveling off.

* difference must be significant.

Colab

λ = regularization parameter.

* What to do next? How to fix?:
  - → Getting more training examples : Fixes HV
  - → Trying smaller features : Fixes HV
  - → Adding features : Fixes HB
  - → Adding polynomial features : Fixes HB
  - → Decrease λ : ~~Hurt~~ fixes HB
  - → Increase λ : Fixes HV.

Diagnosing Neural Network:

→ Fewer Parameters.
- → Prone to underfitting
- → Computationally cheaper.

→ Higher Parameters.
- → Prone to overfitting (λ increase)
- → Computationally expensive.

* Using a single hidden layer is a good default. You can train your of hidden layers using cv set. You can then select the one that perform most.

28.08.18.

Handling Skewed Data:

→ Have a lot more of example from one class than from the other class(es).

* Cancer classification Problem:

Train logistic model $h_\theta(x) = \begin{cases} 1 & \text{if cancer} \\ 0 & \text{if not} \end{cases}$

Your algorithm got 99% accuracy

01% Error.

0.05%.

function predcan(x)

y=0 # ignore X

99.5% Accuracy

0.05% Error.

Error Matrix:

$Y = 1$. cancer

$h_\theta(x) = 1$

Precision / Recall

Predicted class \ Actual class	1	0
1	True Positive	False Positive
0	False Negative	True Negative

* **Precision:** of all patients where we predict $y=1$, what fraction actually has cancer?

$$P = \frac{\text{True Positive}}{\text{Total no. of Predicted positive}}$$

$$= \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

* **Recall:** of all patient ~~the~~ that actually have cancer what fraction did we correctly detect as having cancer?

* High Precision & Recall gives better performance.

$$R = \frac{\text{True Positive}}{\#\text{Actually Positive}}$$

$$= \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

Algo. $\longrightarrow$ High $\begin{cases} \nearrow P \\ \searrow R \end{cases}$

* It is not possible to get high P and R at the same time setting $Y = 1$ or $Y = 0$ always.

If $Y = 1$, $P \to 0$

If $Y = 0$, $R \longrightarrow 0$

P and R

Logistic Regression

$0 \leq h_\theta(x) < 1$

Predict 1, if $h_\theta(x) \geq 0$

Predict 0, if $h_\theta(x) < 0.5$

Predict $Y = 1$, only when very confident

Predict, $Y = 1$ if $h_\theta(x) \geq 0.7$.

" $Y = 0$ if $h_\theta(x) < 0.7$.

(Higher P, Lower R)

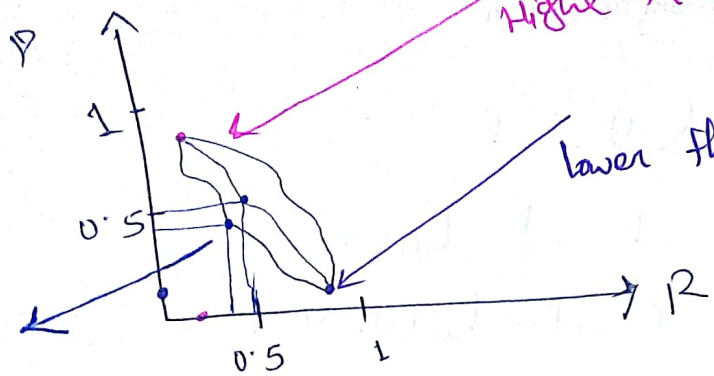
* Want to avoid missing.

Predict 1, if $h_\theta(x) \geq 0.3$

" 0, if $h_\theta(x) < 0.3$

(Lower P, Higher R)

Predict 1 if $h_\theta(x) \geq$ threshold.



Higher threshold (0.99)

lower threshold (0.001)

Both P & R are High

F_1 Score (F score):

	P	R	$(P+R)/2$	F_1
Algo. 1	0.5	0.4	0.45	0.44
Algo. 2	0.7	0.1	0.4	0.175
Algo. 3	0.02	1.0	0.501	0.0392

If $Y = 1$ (always) $R = 1$ $P \approx 0$

If $Y = 0$ $R = 0$ $P \approx 1$

$$F_1 = \frac{2PR}{P+R}$$

* What will be the approach in case of choosing algo of large Data? (v.v.g)

$$F_1 = \frac{2PR}{P+R}$$

if $P = 0$ $F_1 = 0$

$R = 0$ $F_1 = 0$

if $P = 1$ $R = 1$ then $F_1 = 1$

* what is Support Vector Machine ? search

(SVM)

———— x ————

04.09.18

Support Vector Machine :

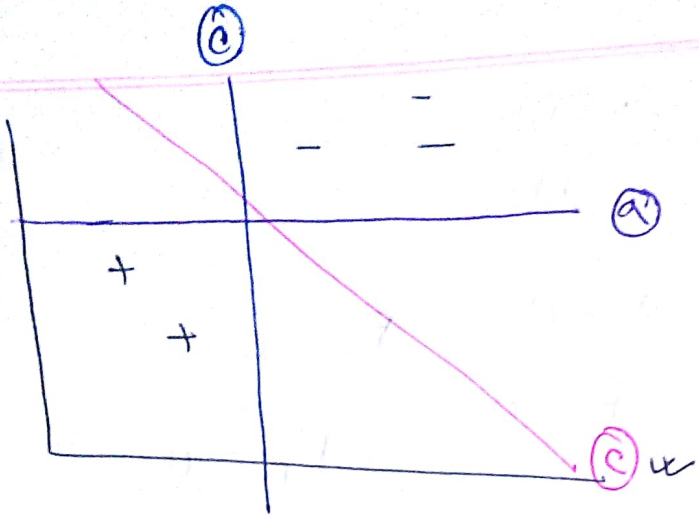
"It is not who has the best algo. that wins.

It is who has the most data."

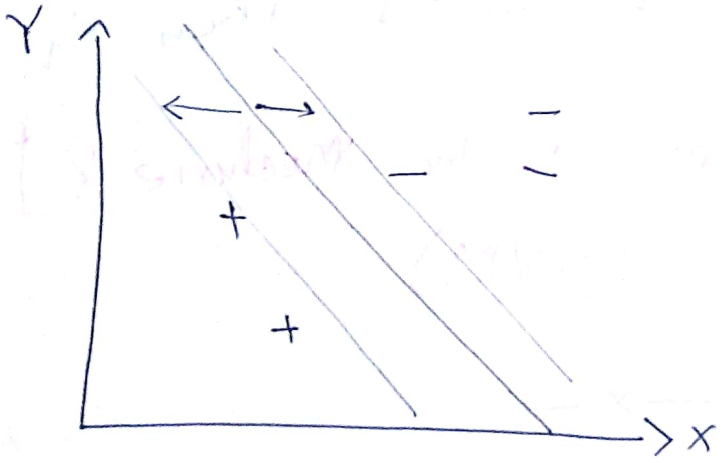
when TRUE, when FALSE ?

i.e: for breakfast I ate ___ eggs.

to, two, too ?



widest street approach.

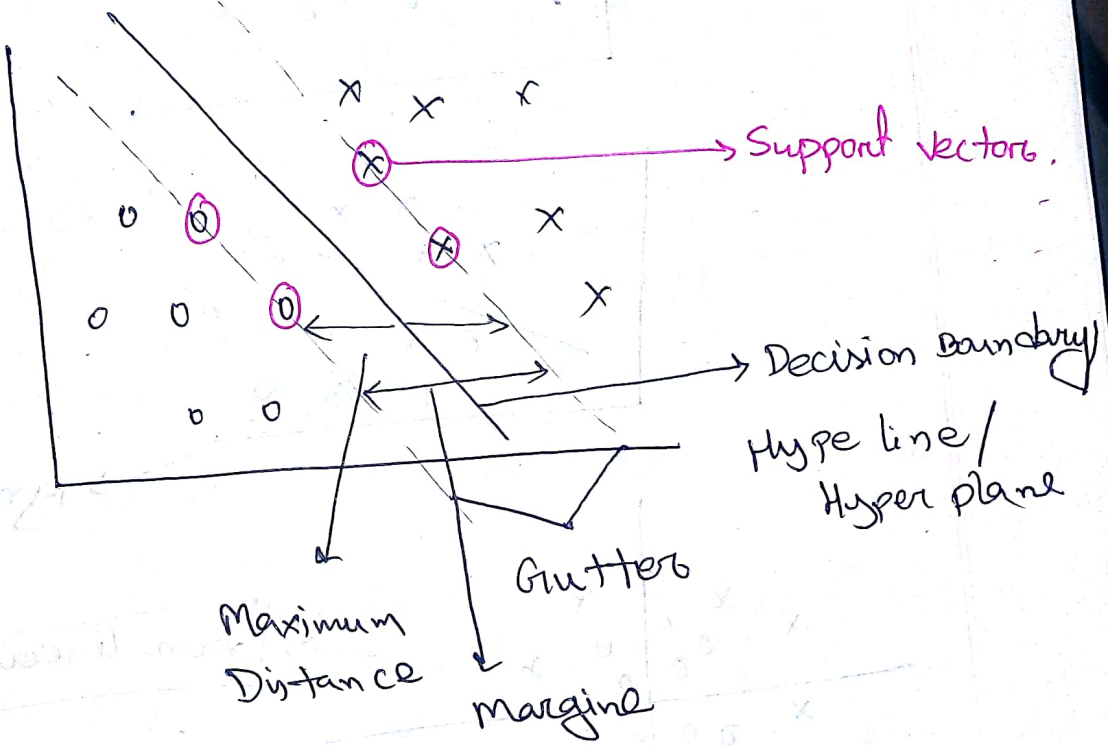


SVM → ML. Algo. (Supervised)

→ Numerical classifiers regression/classifier.

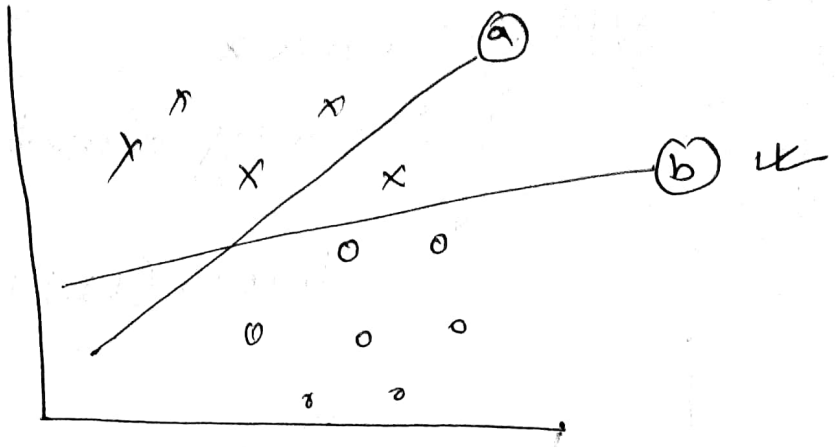
⇒ That draws a single decision boundary to maximize the margins between two region.

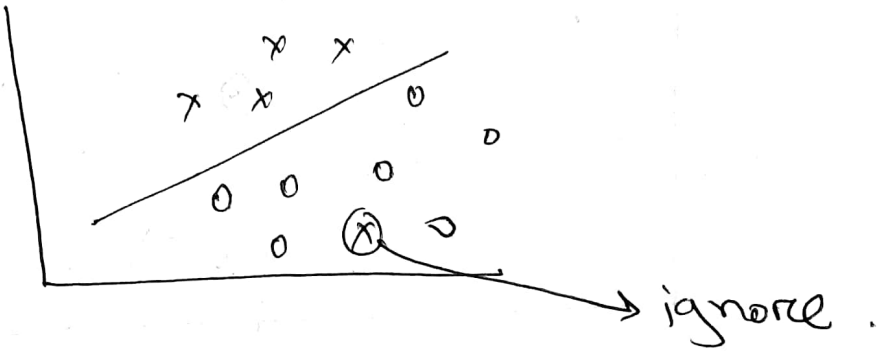
SVM → Convex
→ Binary classifier
→ non-linear (KERNEL)

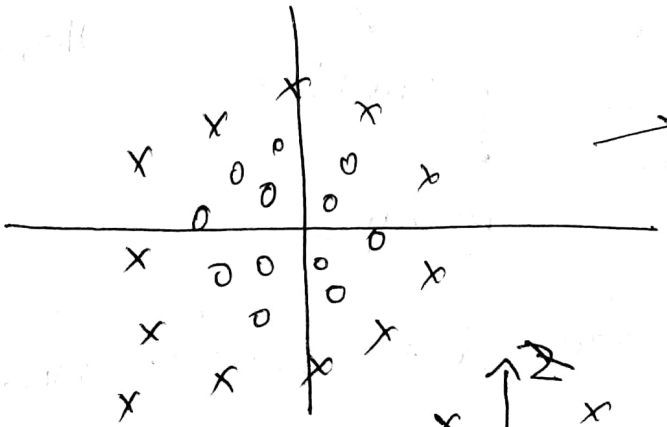


Support Vectors: are data points nearest to hyper plane. If removed, would alter the position of hyper plane.

Hyper plane: is a linear decision surface that splits the space into two parts.
↓
Binary classifier.





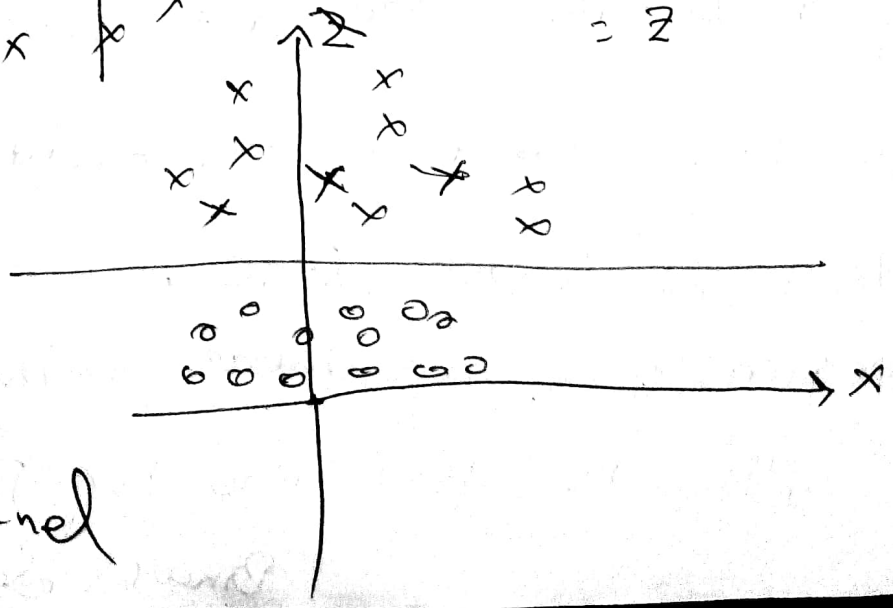


non-linear

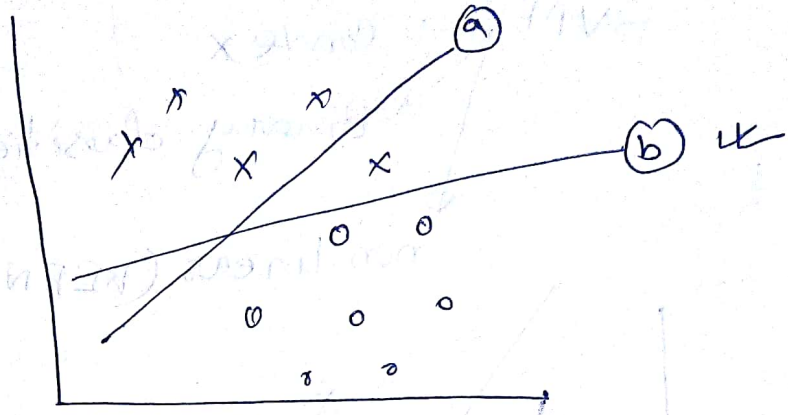
(Kernel tricks)

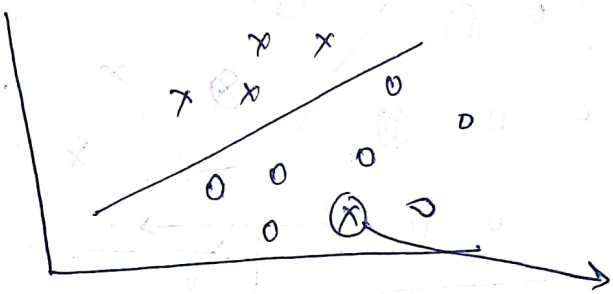
$f = x^2 + y^2$

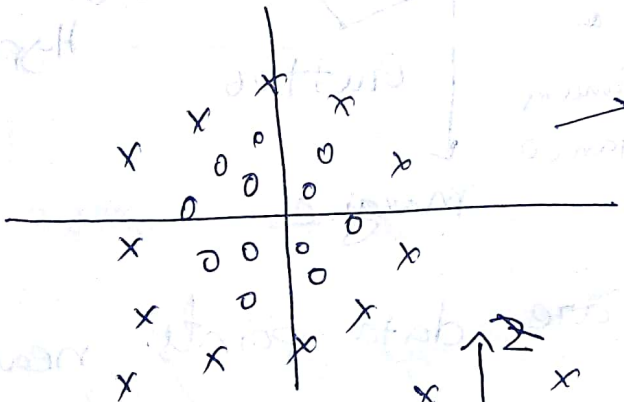
$= z$



* What is Kernel tricks ?





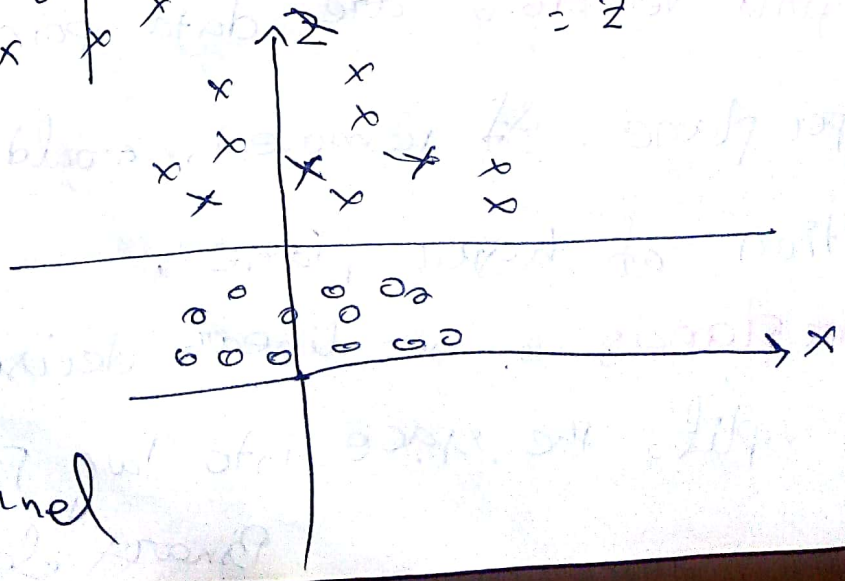


non-linear

(Kernel tricks)

$f = x^2 + y^2$

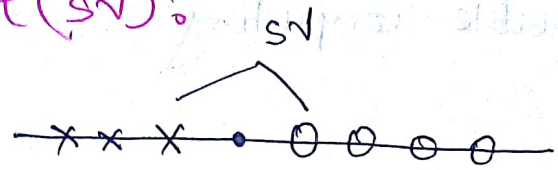
$= z$



* What is Kernel tricks?

* # of Support Vector (SV):

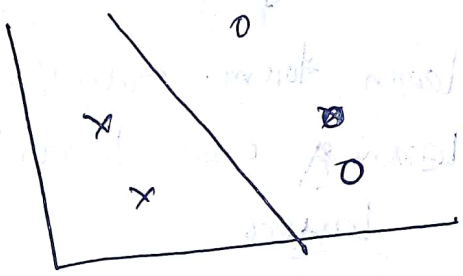
1 Dimension:



# of SV = 2

D: point.

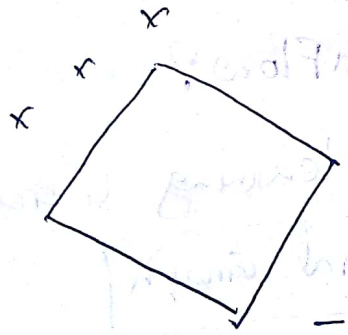
2 Dimension:



# of SV = 2/3/4

line.

3 Dimension:



* How to select SV? [Convex rule] → H.C.

(AI) 2016

Jessica Notes [MIT]

"Special Class" 06.09.18.

Wearable Computing — Ubicomp

Machine Learning Using TensorFlow Library:

Machine learning Vs Deep learning (DL)

learn form solution → DL
learning over learning
layers

DL → representation learning
Hierarchical representation
Dynamic learning

* what is TensorFlow?

A deep learning library.

Computational Graph

import tensorflow as tf

machine translation flow to TensorFlow.

```
import tensorflow as tf

a = tf.constant(1)

b = tf.constant(2)

c = tf.add(a,b)

session = tf.Session()

value_of_c = session.run(c)
```

* Graph building

  prepare execution env

  Initialize variables

  Run the computation.

* Two computation phrases of a Graph:

  (i) construction phrase

  (ii) Execution phrase

Everything is an object in Tensorflow.

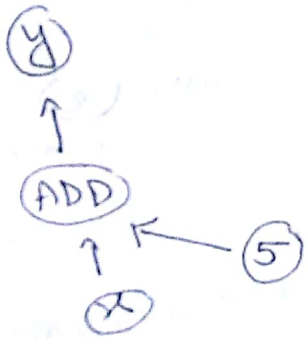
* What is a Tensor?

Tensor data structure to represent all data.

R

* Basic of TensorFlow:

(1) Variable



* Linear Algebra:

(1) Dimensionality

(11)

1 2 3
4 5 6

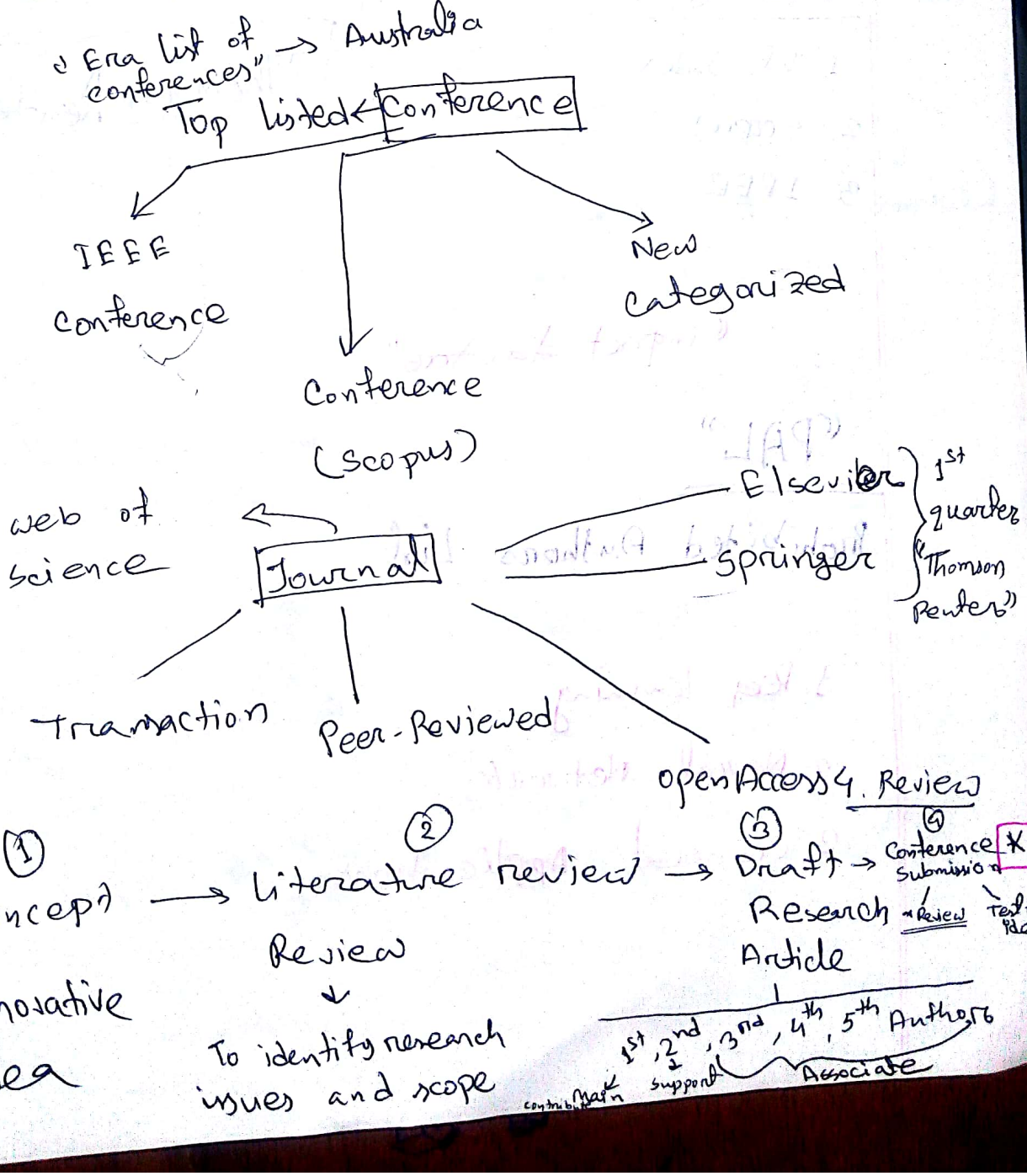
1 2 3
4 5 6

2 4 6
8 10 12

Linear Regression:

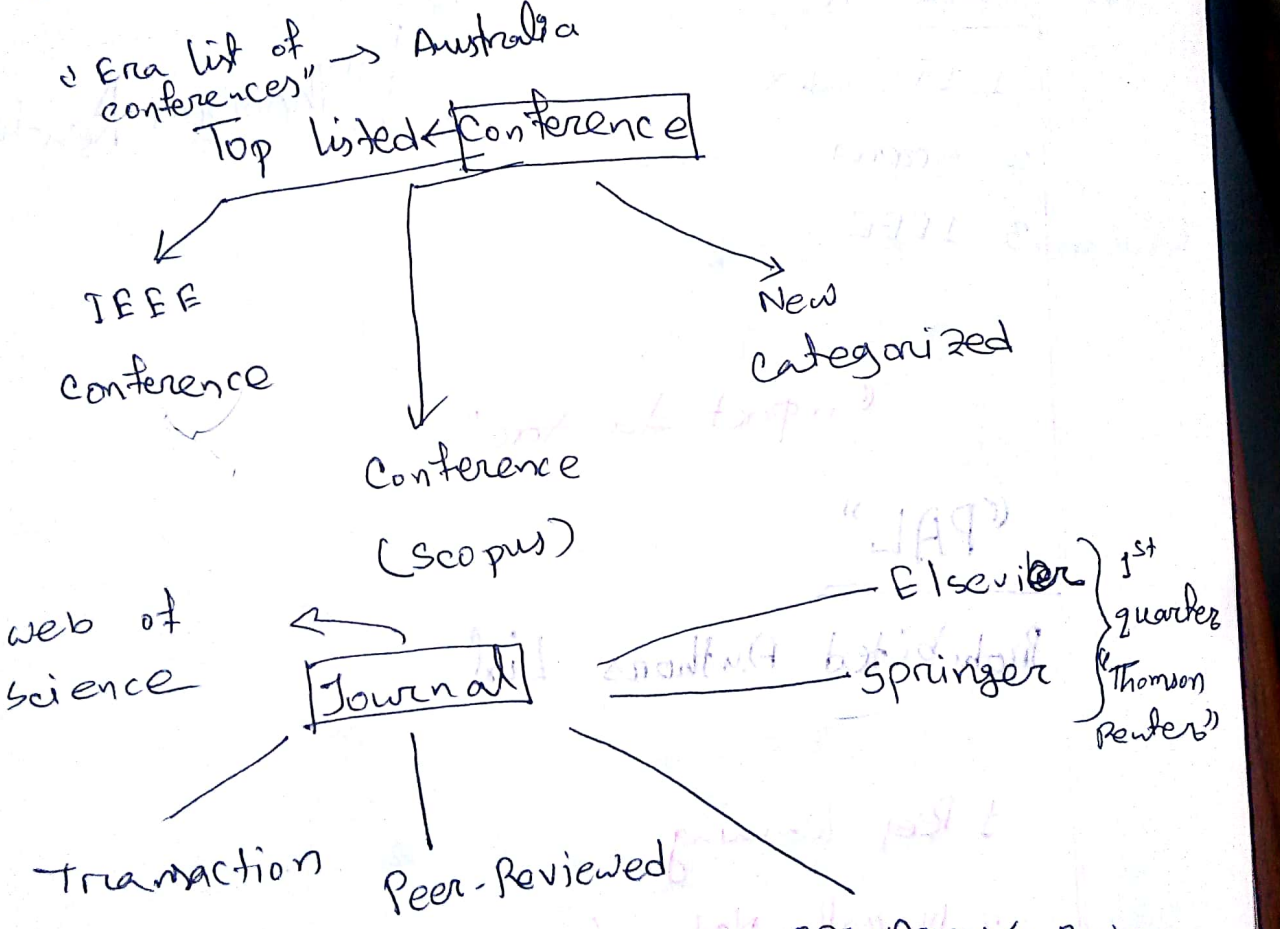
Research Discussion:

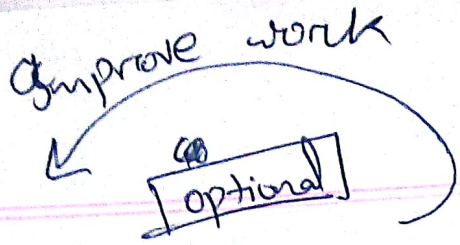
1. Plagiarism
2. Paraphrasing
3. Citation..



Research Discussion:

1. Plagiarism
2. Paraphrasing
3. Citation..





** (4) Conference Submission → (5) Journal Submission "Archive"

Indexing

1. ISI Index
2. Scopus
3. IEEE

"Thomson Reuter"

"impact factor"

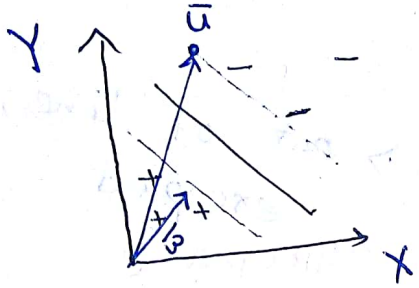
"PAL"

Prohibited Authors List

1. Deep learning
2. Neural Networks
3. General Applications

09.09.18.

Support Vector Machine:



Vector is perpendicular with hyperplane.

u = unknown point / new example.

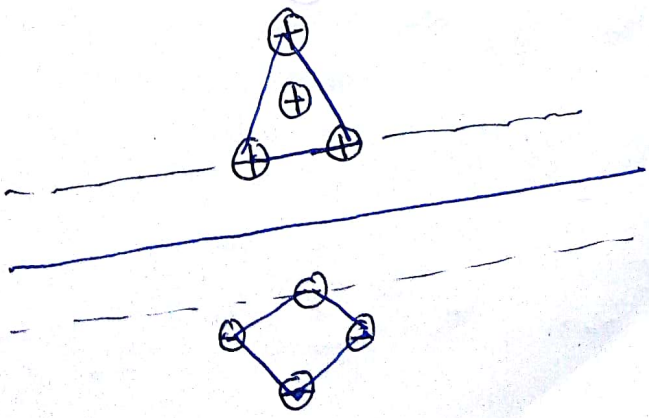
if, $\bar{w} \cdot \bar{u} \geq c$ [b,c, constant]

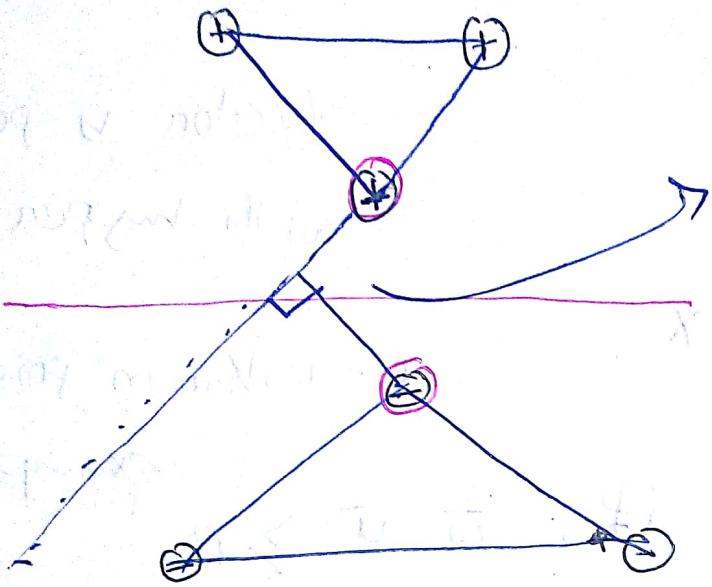
then +

or, $\bar{w} \cdot \bar{u} + b \geq 0$

α_i = Supportitiveness

Convex hull Method:

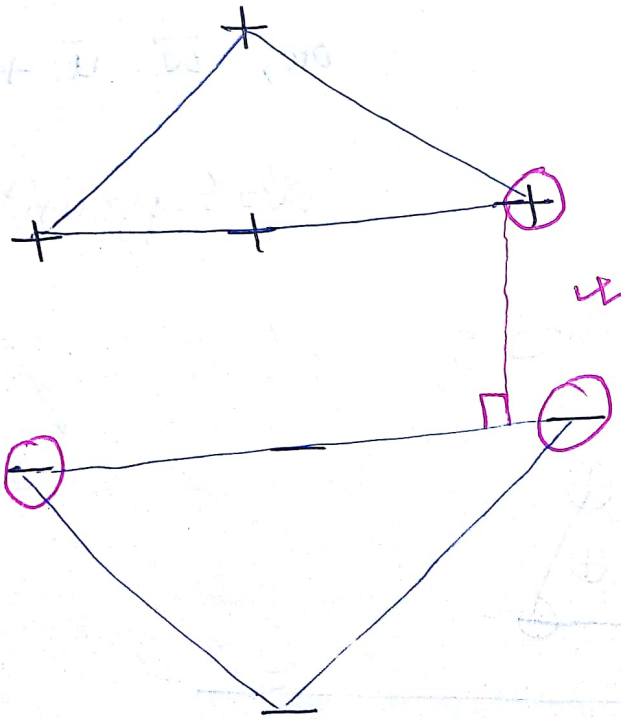




out of line, so not excepted

Hyperplane

2 point / SV

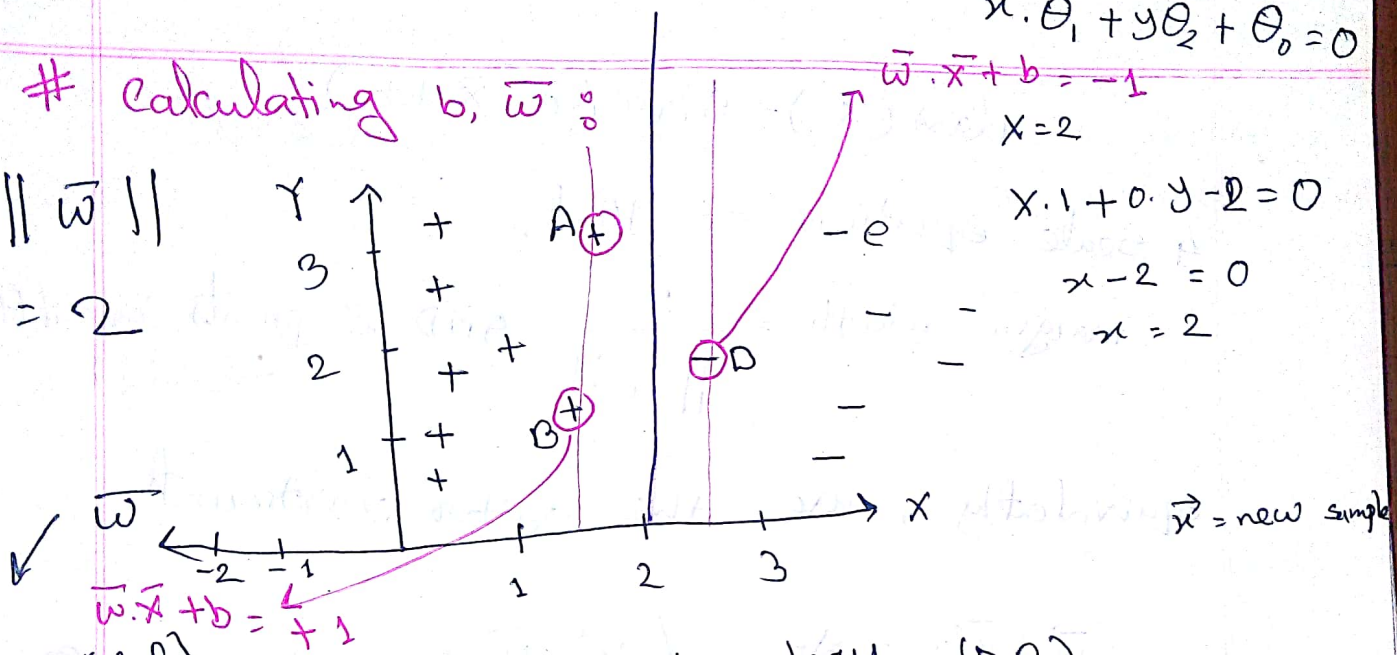


3 points / SV

$x.\theta_1 + y\theta_2 + \theta_0 = 0$

Calculating b, $\vec{w}$:

$\|\vec{w}\| = 2$



$\vec{w}.\vec{x} + b = -1$

$X = 2$

$X.1 + 0.y - 2 = 0$

$x - 2 = 0$

$x = 2$

(As $(\vec{w}) = [-2\ 0]$)

1. Draw the decision boundary. (DB)

2. Write the equation for the DB.

$x = 2$

[2 dimensional vector as here two features]

3. Rewrite the equation as $\vec{w}.\vec{x} + b = 0$ — (1)

$\vec{w}.\vec{x} + b = 0 \quad \begin{bmatrix} x \\ y \end{bmatrix}$

$x.\theta_1 + y\theta_2 + \theta_0 = 0$

$\theta^T x$

$\vec{w}.\vec{x} + b = 0$

$[1\ \ 0] \begin{bmatrix} x \\ y \end{bmatrix} + (-2) = 0$

$$class(\bar{x}) = sign(\bar{w} \cdot \bar{x} + b)$$

4. Scale equation so that,

margin width $= \frac{2}{\|\bar{w}\|}$ AND $\bar{w}$ points toward (+) ——— (2)

equivalently, use the gutter constraint.

$\vec{w} \cdot \vec{x}_i = y_i$ / $\vec{w} \cdot \vec{u} + b = y_i$ ——— (3)

for all support vectors

where $y_i = class(i) = +1$ or -1

C. $\bar{w} \cdot \bar{x}_i + bC = y_i$

Using SVD,

$\vec{w} \cdot \vec{x} + b = 0$

C. $\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} 2.5 \\ 2.25 \end{bmatrix} + C(-2) = y_i = -1$

$2 - 2C = -1$

$-0.5 = -1$

$C \cdot 2 - C \cdot (1.5) = 1$

$\Rightarrow 0.5\,C = -1$

$\Rightarrow C = -2$

$\vec{w} = \vec{w} \cdot C$

$= [-2 \;\; 0]$

$b = +3$

margin, $\omega = \frac{2}{\|\omega\|}$ $\qquad \sqrt{(-2)^2 + 0^2}$

$= \frac{2}{2}$ $\qquad = 2$

$= 1$

→ if margin decreases, ω will increase else, decrease if margin increases.

margin width, $= \frac{2}{\|\omega\|}$

$= \frac{2}{2}$

$= 1$.

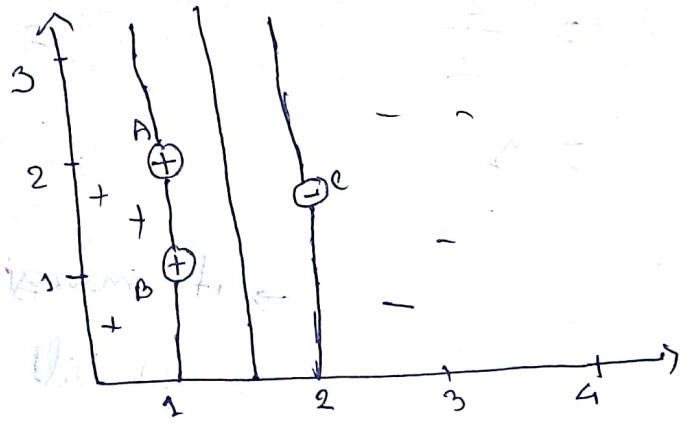
H.W → Winston MIT professor SVM

11.09.18

Calculating α (Lang Lagrange multipliers) / (Sequential minimal optimizers)

Supportiveness value:

↓
find local min/max of a function



* α represents how big of a role each point plays in determining / supporting the decision boundary.

1. For all non-support vectors, $\alpha = 0$

   otherwise, the point is a SV and lies on the Gutter and $\alpha > 0$

   [Positive Supportiveness]

2. $\sum_{\text{Support vectors}} \alpha_i y_i = 0$

$$\Longleftrightarrow \sum_{\text{Positive SV } p} \alpha_p = \sum_{\text{negative SV } n} \alpha_n$$

$$\alpha_A + \alpha_B - \alpha_C = 0 \quad \Rightarrow \alpha_A + \alpha_B = \alpha_C \quad \text{——} (1)$$

3. $$\sum_i \alpha_i y_i \bar{x}_i = \bar{w}$$

$$\Rightarrow w = \sum_p \alpha_p \bar{x}_p - \sum_n \alpha_n \bar{x}_n$$

$$\begin{bmatrix} -2 \\ 0 \end{bmatrix} = \alpha_A \begin{bmatrix} 1 \\ 3 \end{bmatrix} + \alpha_B \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \alpha_C \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

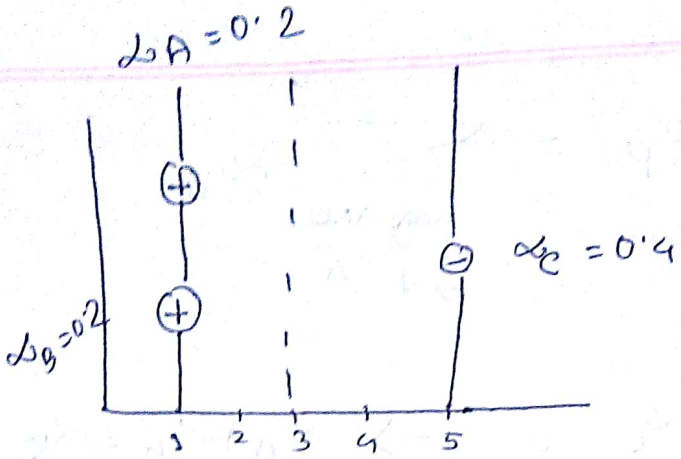
$$\Rightarrow -2 = \alpha_A \cdot 1 + \alpha_B \cdot 1 - \alpha_C \cdot 2 \quad \text{——} (2)$$

$$0 = \alpha_A \cdot 3 + \alpha_B \cdot 1 - \alpha_C \cdot 2 \quad \text{——} (3)$$

$$\alpha_A = 1$$

$$\alpha_B = 1$$

$$\alpha_C = 2$$

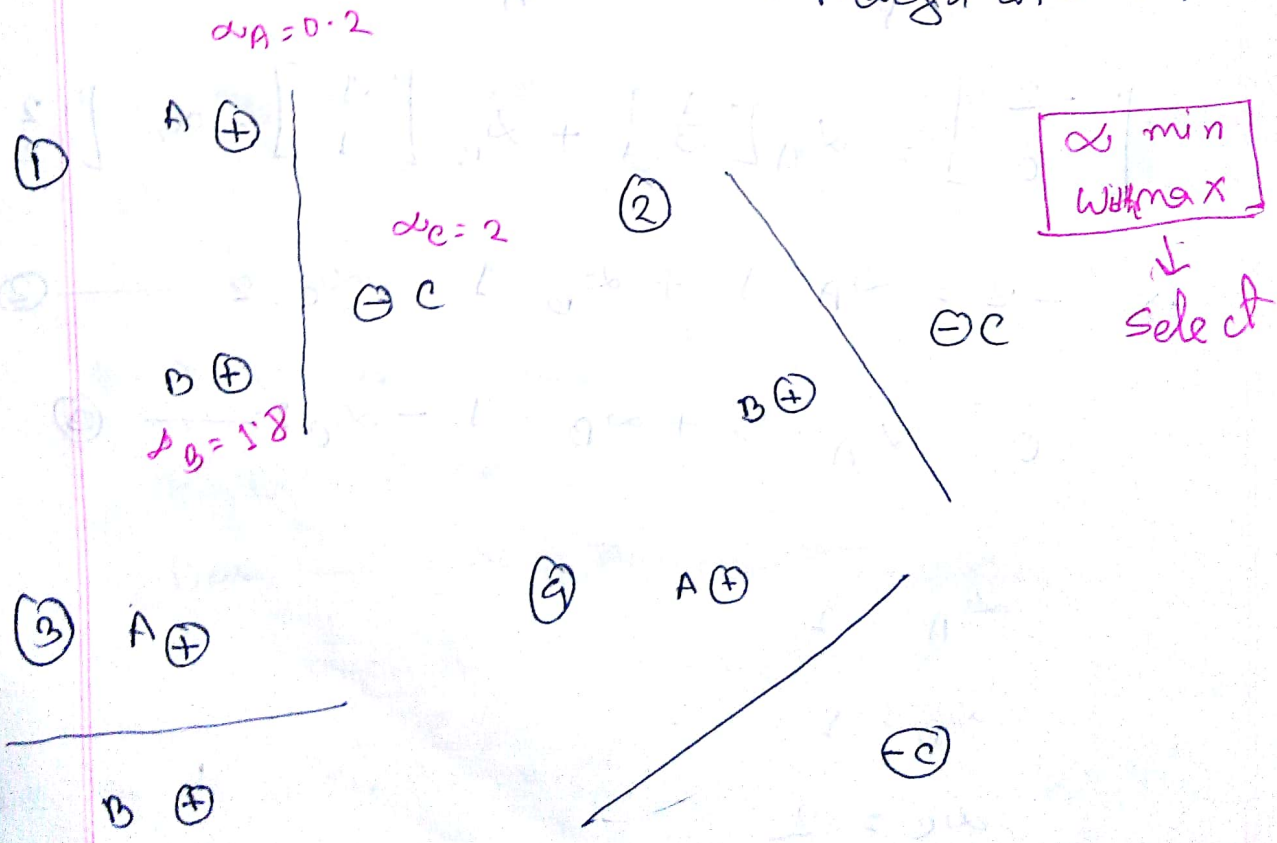


$$\text{margin width} = \frac{2}{\|w\|}$$

$w \downarrow$

$\alpha \downarrow$

Margin width $\uparrow$



When data is not linearly seperable,

- Apply transformation
- Replace dot product with another function (kernel), effectively transforming the space.

(A kernel function $K(\bar{u}, \bar{v})$ is a similarity measure)

$$\text{class}(\bar{x}) = \text{sign}(\bar{w} \cdot \bar{x} + b)$$

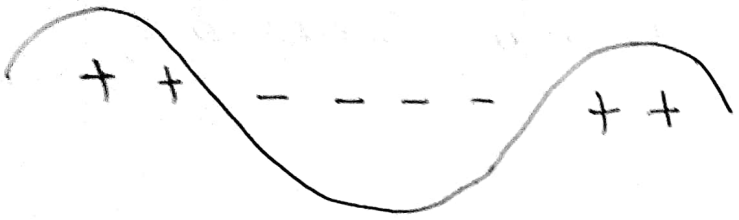
$$= \text{sign}\left(\sum_i (\alpha_i y_i \bar{x}_i) \cdot x + b\right)$$

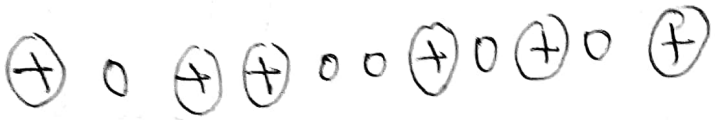
$$= \text{sig}\left(\sum_i (\alpha_i y_i) K(\bar{x}_i \cdot \bar{x}) + b\right)$$

↳ Transforms to higher order/ function/ dimension.

Common kernel function:

- linear, e.g. $K(\bar{u} \cdot \bar{v}) = \bar{u} \cdot \bar{v} + 1$; draw line.
- quadratic, e.g. $K(\bar{u} \cdot \bar{v}) = (\bar{u} \cdot \bar{v} + 1)^2$; draw conic



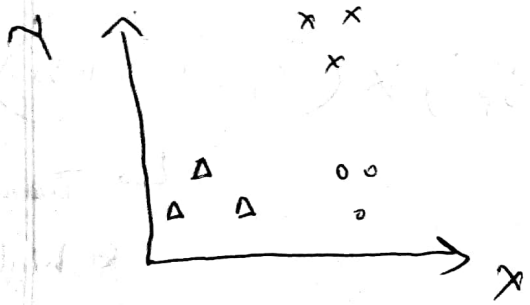


— Polynomial

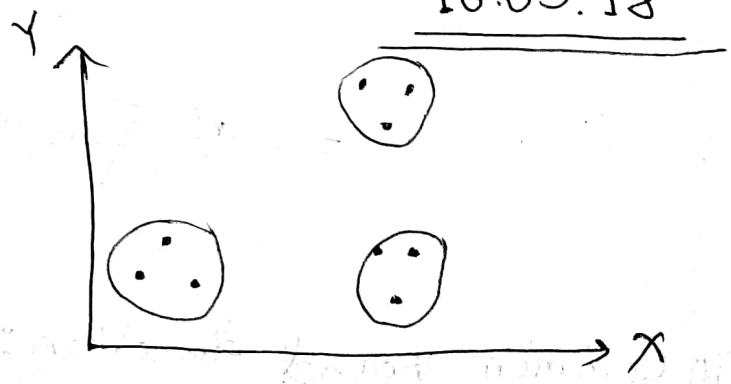
- ~~Radi~~ Radio bias Kernel

$$k(\bar{u}, \bar{v}) = e^{-|\bar{u} - \bar{v}|/6}$$

16.09.18



Supervised



Unsupervised (not labeled)

⇒ Supervised learning:

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

⇒ Unsupervised learning:

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

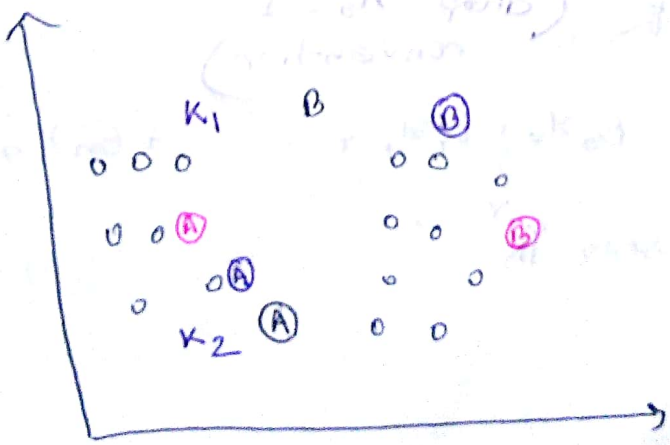
Applications of clustering algorithms:

→ Social Network analysis.

→ Market segmentation.

→ Astronomical data analysis.

* K-means Algorithm: K = No. of clusters.



K = 2

→ Select Random point K_1, K_2 on A, B.

* K-means:

→ iterative algo

→ has two steps

(i) Cluster assignment step
(ii) Move cluster centroid step.

repeat.

K-means Algorithm:

Input:

- K (# of cluster)
- Training set $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$

$x^{(i)} \in \mathbb{R}^n$ (drop $x_0 = 1$ convention)

$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_n x_n$

→ Dimension $\mathbb{R}^{n+1}$.

* Algorithm:

→ Randomly initialize K-cluster centroids $\mu_1, \mu_2, \ldots, \mu_K$

→ Repeat {

Cluster Assignment step

for $i = 1$ to m

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

min J(–) w.r.t. $c^{(1)}, c^{(2)}, \ldots, c^{(m)}$

$\mu_1, \mu_2, \ldots, \mu_K$ → fixed

move centroid step

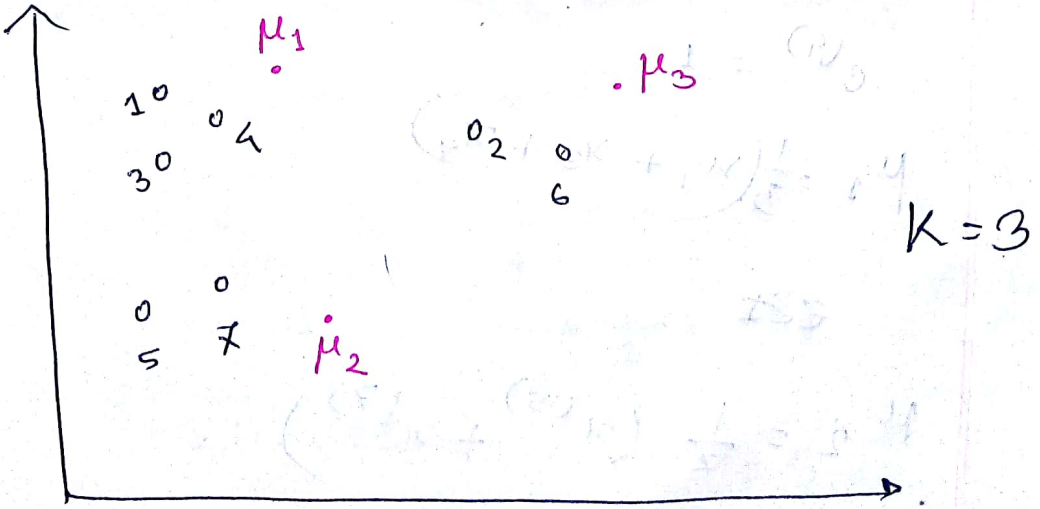
for $K = 1$ to K

μ_K = average (mean) of points assigned to cluster K..

min J(–) w.r.t. $\mu_1, \mu_2, \ldots, \mu_K$

$c^{(1)}, c^{(2)}, \ldots, c^{(m)}$ → fixed

}



$K = 3$

$x^{(1)} \Rightarrow c^{(1)} = 1$

$x^{(2)} \Rightarrow c^{(2)} = 3$

$x^{(3)} \Rightarrow c^{(3)} = 1$

$x^{(4)} \Rightarrow c^{(4)} = 1$

$x^{(5)} \Rightarrow c^{(5)} = 2$

$x^{(6)} \Rightarrow c^{(6)} = 3$

$x^{(7)} \Rightarrow c^{(7)} = 2$

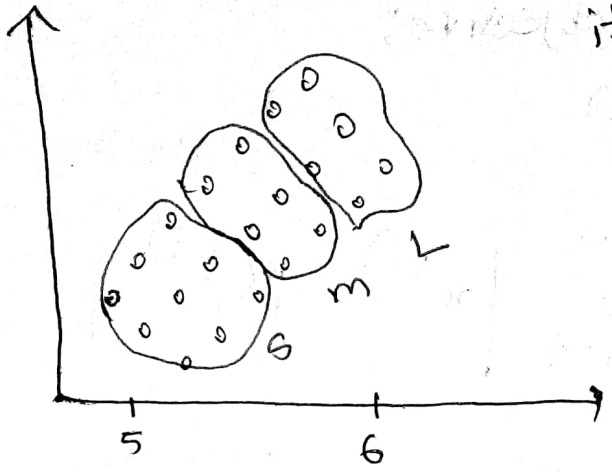
$c^{(1)} = 1$

$c^{(3)} = 1$

$c^{(4)} = 1$

$\mu_1 = \frac{1}{3}(x_1 + x_3 + x_4)$

$\mu_2 = \frac{1}{2}(x^{(5)} + x^{(7)})$

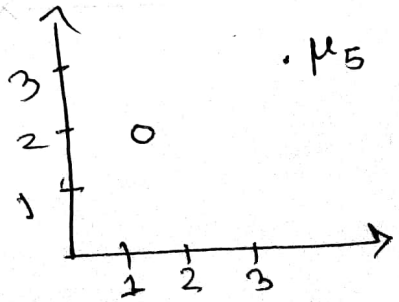


Optimization objective:

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

μ_k = cluster centroid k ($\mu_k \in \mathbb{R}^n$) (same with the sample dimension)

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



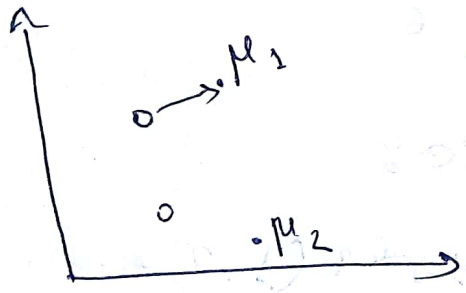
$x^{(1)} = (1,2)$ $\quad \therefore c^{(1)} = 5$

$\mu_5 = (2,3)$ $\quad \mu_{c^{(1)}} = \mu_5 = (2,3)$.

* Optimization objective:

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

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



$x^{(1)} - \mu_1$

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

—x—

18.09.18

K-mean Algo.:

- Randomly initialize k - cluster centroid

$\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

- Repeat {

for $i = 1$ to m

$c^{(i)}$ = index of cluster centroid (1 to k) which is closest to $x^{(i)}$

for $K = 1$ to K

μ_K = average (mean) of points assigned to cluster μ_K

Optimization:

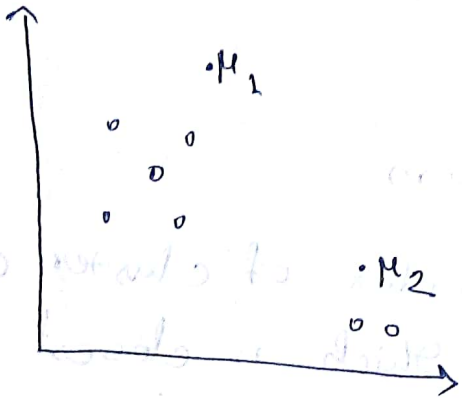
$J(c^{(1)}, c^{(2)}, \ldots, c^{(m)}, \mu_1, \mu_2, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} |x^{(i)} - \mu_{c^{(i)}}|^2$

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

i^{th} index point $\longrightarrow \mu_5$

$c^{(i)} = 5$

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

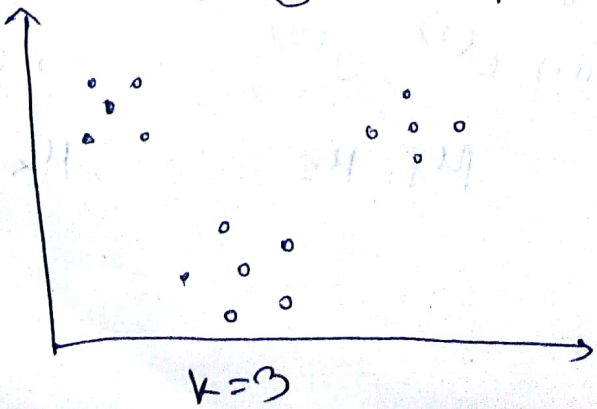


Important:

* assignment of each point
* Position selection of μ_k

Random Initialization:

- select k such that $k < m$.
- Randomly pick k training examples.



- Set $\mu_1, \mu_2, \ldots, \mu_K$ equal to these examples.

Drawbacks:

* Local optimal:

Random initialization to avoid local optima

- for $i = 1$ to 100 (50 - 1000)

{ - Randomly initialize K cluster centroid.

- Run K-means algorithm

- Get $c^{(1)}, c^{(2)}, \ldots, c^{(m)}, \mu_1, \mu_2, \ldots, \mu_K$

- Compute cost function (distortion)

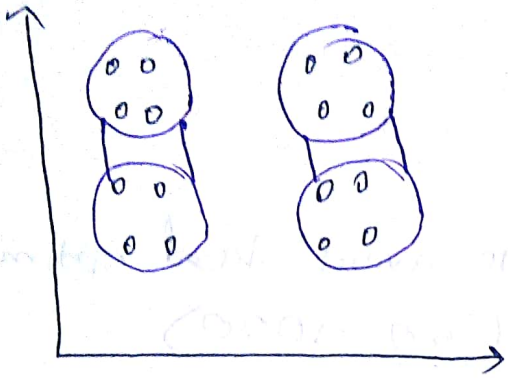
$$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \mu_2, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} |x^{(i)} - \mu_{c^{(i)}}|^2$$

}

pick the cluster that gives the lowest cost value.

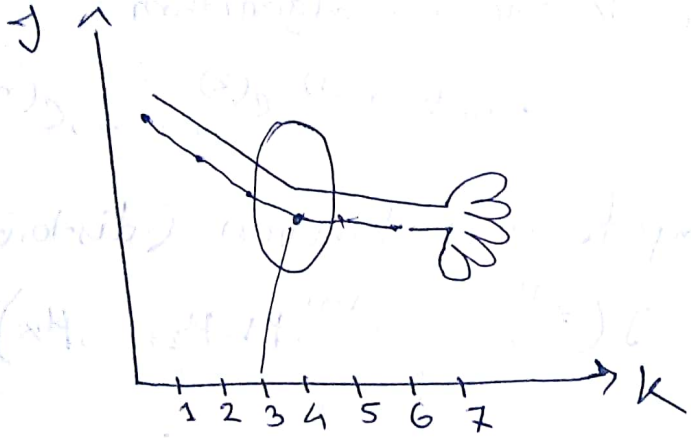
* (If $K = 2$ to 10) this algo. works really well.
* If $K = 100$ + expected result can be get within few iterations. So, no. of iterations can be decreased.

How to choose value of K ?:

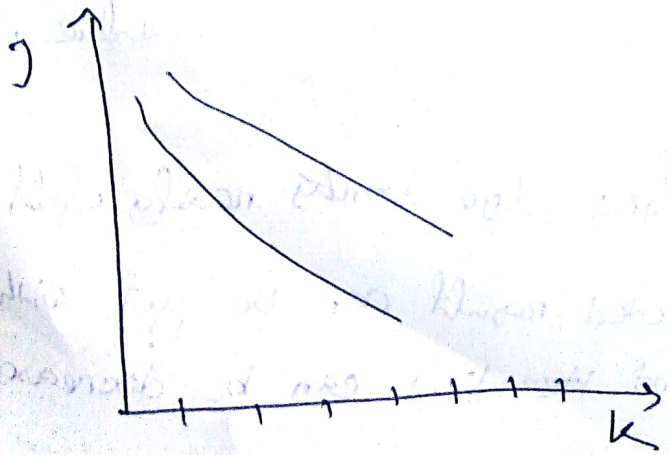


K = 4 / 2

* Elbow Method:



↑ Select 3 as the value of K.



→ Elbow method doesn't work well in these kinds of cases.

* choosing K's value based on purpose.

T-Shirt size

3 → S, M, L

5 → XS, S, M, L, XL