Machine Learning —
Andrew Ng.

ML

$x = \{x^1, x^2, x^3 \ldots\}$
$x^1_{2.2}$

① $h: x \to y$

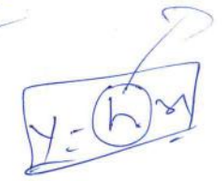
x: set of inputs of ndh

$x = (x^n)$

y: set of outputs

$y = y^n$

$h: x \to y$

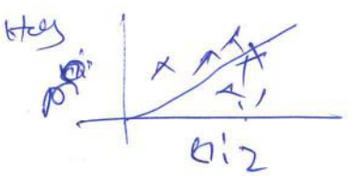


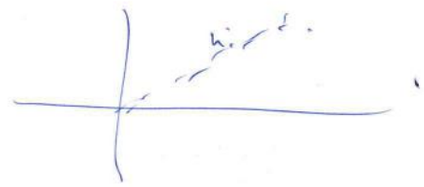
→ $h: x \to y$, ~~when $x \in R^n$, $y \in R$~~

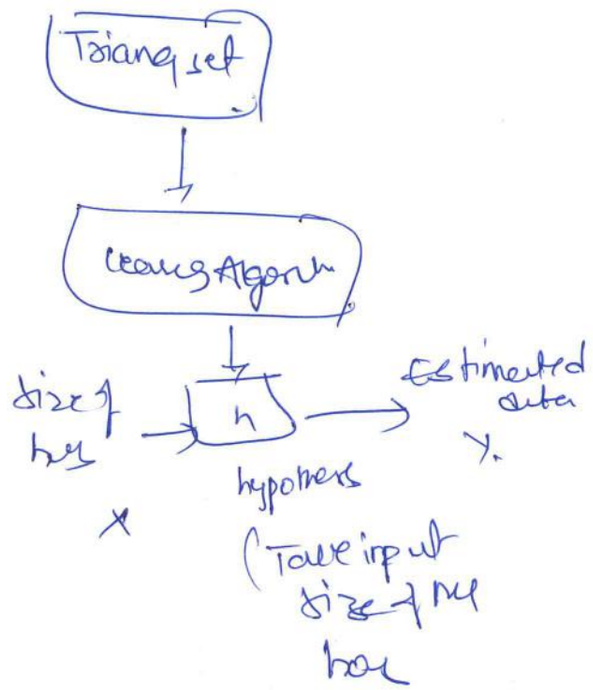
→ when $x = R^n$ and $y = R$, the simplest choice is to approximate y as a linear function of x

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

Size feet^2 (x)	Price ($1000s) (y)
$x^{(1)}$	$y^{(1)}$
$x^{(2)}$	$y^{(2)}$
$x^{(3)}$	$y^{(3)}$





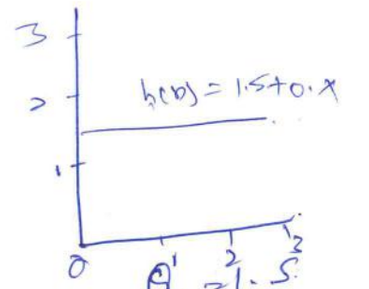


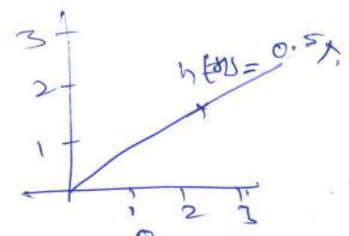
cost function:- How to fit best possible straight line to our data.

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

θ_i's = parameters

How to choose θ_0 and θ_1





Idea:- Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training samples (x, y)

$\Rightarrow$ So, minimize (θ_0, θ_1) so $\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$ (size of training set)

θ_0, θ_1

Parameters to be minimized

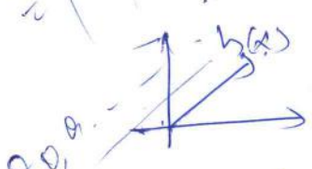
$\min_{\theta_0, \theta_1} \left[\sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \right]$

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

minimize θ_0, θ_1 $J(\theta_0, \theta_1)$ ⟶ Squared error cost function

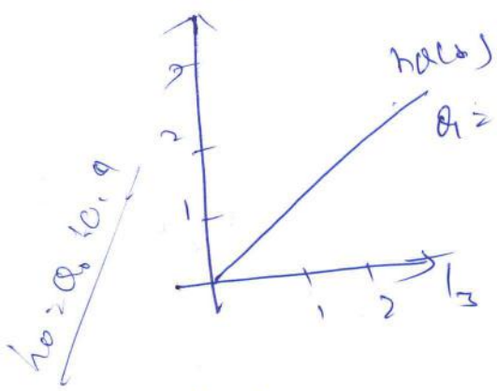
simplified

$h_\theta(x) = \theta_1 x$ ∴ $\theta_0 = 0$

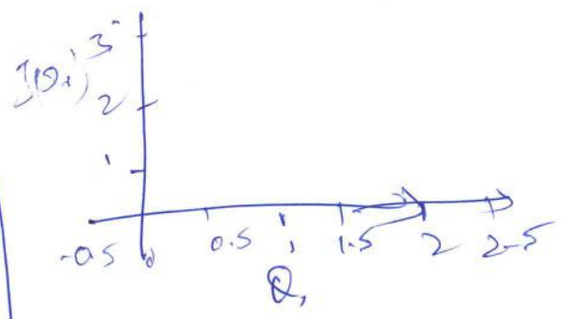


$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$h_\theta(x)$



$J(\theta_1)$



$\theta_1 = 1$

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

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

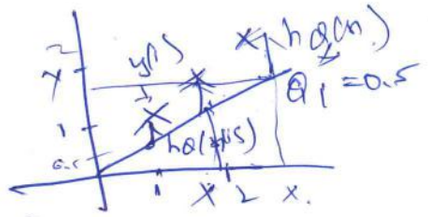
correct for x_1 and x_2

$= \frac{1}{2m} \left(0^2 + 0^2 + 0^2 \right) = 0^2$

$J(1) = 0$

1780.0 e
≤0

If $\theta_1 = 0.5$



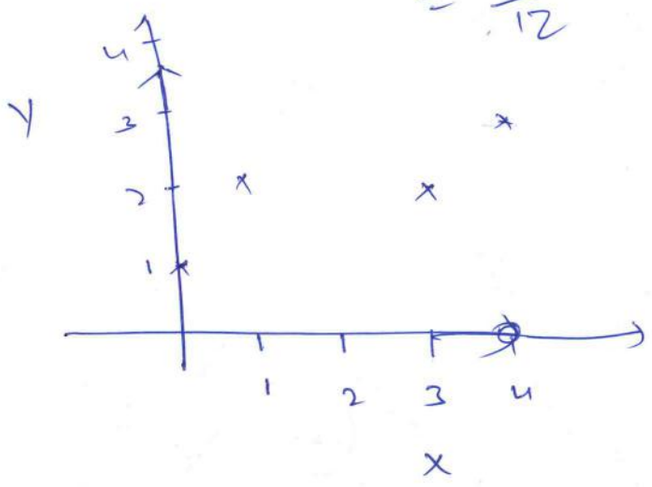
$$J(0.5) = \frac{1}{2 \times 3} \left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2 \right]$$

$m = 3$

$$= \frac{1}{6} \left[(0.5)^2 + (1)^2 + (1.5)^2 \right]$$

$$= \frac{1}{6} \left[\frac{7}{2} \right]$$

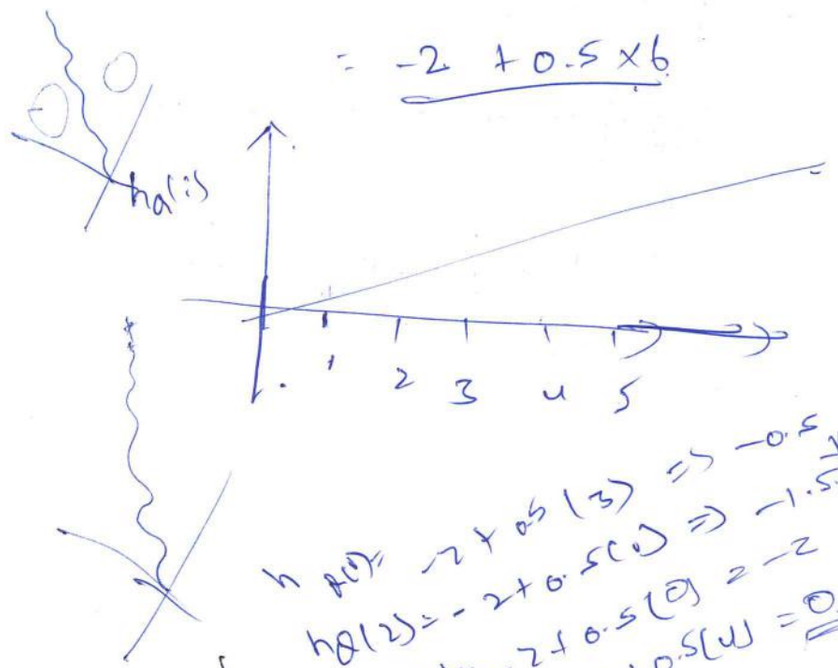
$$= \frac{7}{12}$$



$-1780.0 + (1)(530.9)$

$h_\theta(6) = \theta_1 + \theta_2 x_6$

$= -2 + 0.5 \times 6$



$h_\theta(1) = -2 + 0.5(1) \Rightarrow -1.5$

$h_\theta(2) = -2 + 0.5(2) \Rightarrow -1.5$

$h_\theta(3) = -2 + 0.5(3) = -0.5$

$h_\theta(4) = -2 + 0.5(4) = 0$

$h_\theta(5) = -2 + 0.5(5) = 0.5$

$h_\theta(x^1) = -2 + 0.5(1) \Rightarrow h_\theta(1) = -1.5$

$h_\theta(x^2) = -2 + 0.5(2) \Rightarrow h_\theta(2) = -1$

$h_\theta(x^3) = -2 + 0.5(3) \Rightarrow -0.5 \Rightarrow h_\theta(3) = -0.5$

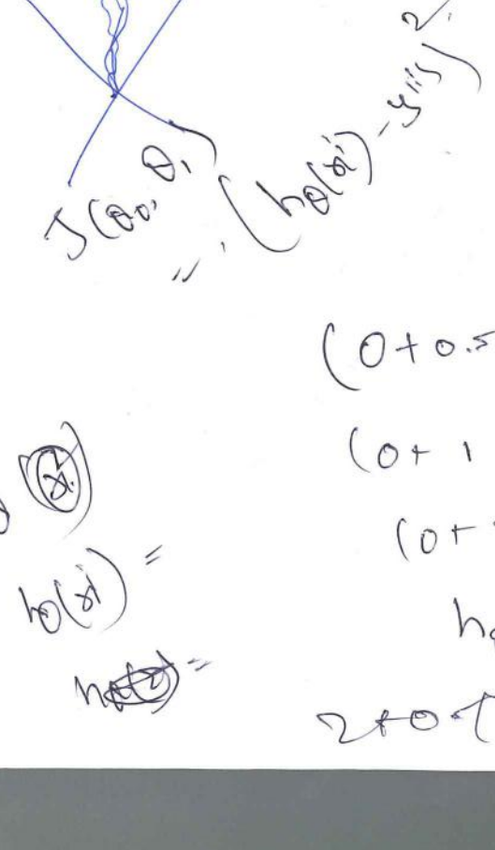
$h_\theta(x^4) = -2 + 0.5(4) = h_\theta(4) = -2 + 2 = 0$

$h_\theta(x^5)$

$h_\theta(x^5) = -2 + 0.5(x^5)$

$h_\theta(6) = -2 + 0.5(6)$

$= -2 + 3 = 1$



$(0 + 0.5 - 0.5)^2 = 0$

$(0 + 1 - 1)^2 =$

$(0 + 2 \cdot$

$h_\theta(2) = 0 + 1 \cdot (2)$

$= 0 + 3 \quad (1)$

$2 + 0 \cdot (3) =$

matrices :- Week - 2

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 0 & 5 \end{bmatrix}, \begin{bmatrix} 10 \\ 23 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} 10 \\ 23 \end{bmatrix} = \begin{bmatrix} 79 \\ 94 \\ 115 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & -4 \\ 2 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 3 \\ 5 & 8 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & -2 \\ -4 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 5 \\ 3 & 8 \end{bmatrix} \quad A+B = \begin{bmatrix} 1 & 3 \\ -1 & 9 \end{bmatrix}$$

$$x = \begin{bmatrix} 2 & 2 & 4 & 1 \end{bmatrix}$$

$$v = \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}, \quad v = \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix}$$

$$v v^T = \begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix} \begin{bmatrix} 2 & 2 & 4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 4 \\ 6 & 6 & 12 \\ 2 & 2 & 4 \end{bmatrix}$$

$3 \times 1 \quad 1 \times 3$

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} 2 \\ 2 \\ 4 \end{bmatrix}$$

$$\begin{bmatrix} -3 \\ 4 \\ 3 \end{bmatrix} \begin{bmatrix} 3 & 1 & 5 \end{bmatrix} = \begin{bmatrix} -9 & -3 & -15 \\ 12 & 4 & 20 \\ 9 & 3 & 15 \end{bmatrix}$$

$$v = \begin{bmatrix} -3 & 4 & 3 \end{bmatrix}$$

$$v = \begin{bmatrix} 3 & 1 & 5 \end{bmatrix}$$

* Linear regression - multivariate.

x	y

⇒ bivariate.

x_1	x_2	x_3	y
x_1^1	x_2^1	x_3^1	y^1
x_1^2	x_2^2	x_3^2	y^2
x_1^3	x_2^3	x_3^3	y^3

⇒ multivariate.

n = number of features.

$x^{(i)}$ = input (i)

$x_j^i = x^{(i)}_{(j)}$ (i) → row, (j) → column

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 \cdots \theta_n x_n \quad (x_0 = 1)$$

$$h_\theta(x^i) = \theta_0 + \theta_1 x_1^i + \theta_2 x_2^i + \theta_3 x_3^i + \cdots \theta_n x_n^i$$

$$h_\theta(x^1) = \theta_0 + \theta_1 x_1^1 + \theta_2 x_2^1 + \cdots$$

$$h_\theta(x^2) = \theta_0 + \theta_1 (x_1^2) + \theta_2 (x_2^2) \cdots$$

$$\begin{bmatrix} h_\theta(x^1) \\ h_\theta(x^2) \\ h_\theta(x^3) \end{bmatrix} = \begin{bmatrix} x_0^1 & x_1^1 & x_2^1 & x_3^1 \\ x_0^2 & x_1^2 & x_2^2 & x_3^2 \\ x_0^3 & x_1^3 & x_2^3 & x_3^3 \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$= \theta^T \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$= \begin{bmatrix} \theta_0 \theta_1 \theta_2 \theta_3 \end{bmatrix} \begin{bmatrix} x_0^1 & x_0^2 & x_0^3 \\ x_1^1 & x_1^2 & x_1^3 \\ x_2^1 & x_2^2 & x_2^3 \\ x_3^1 & x_3^2 & x_3^3 \end{bmatrix}$$

$$= \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ a_2 \end{bmatrix}, \quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

$x_1, x_2, x_3 \ldots$ $x=1$

$h_\theta = \theta_0 + \theta_1 x_1 + \theta_2 x_2$

$= \theta_0 + \theta \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

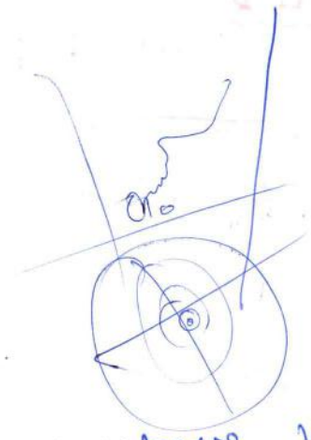
$= \theta^T x$

$h = \theta^T x$

* Octave :

```
a = [1 2 3; 4 5 6]
a =
   1  2  3
   4  5  6
b = a(:,1)
b =
   1
   4
```

```
b = a(:,3)
b =
   3
   6
b = (1,:)
  = 1 2 3
```



$h = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots$

Gradient descent for multiple variables

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

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

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

update θ_0, θ_1

Update θ_j's calculations

Update algorithm

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

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

One variable Gradient descent method

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

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

Week 3 : WEEK 3 :

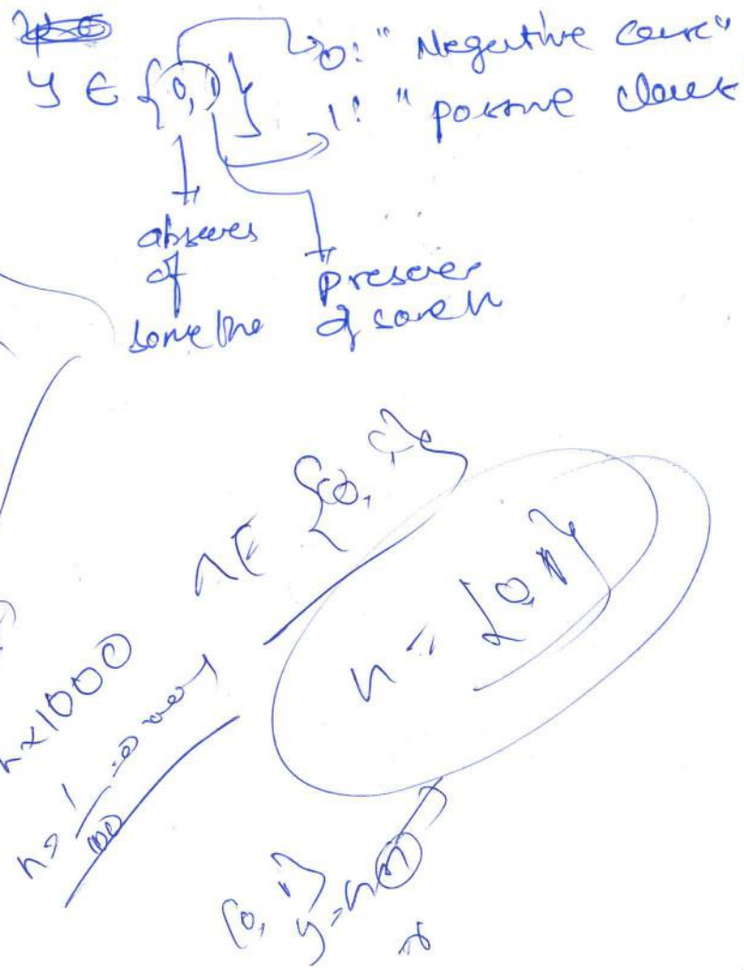
CLASSIFICATION AN Representation

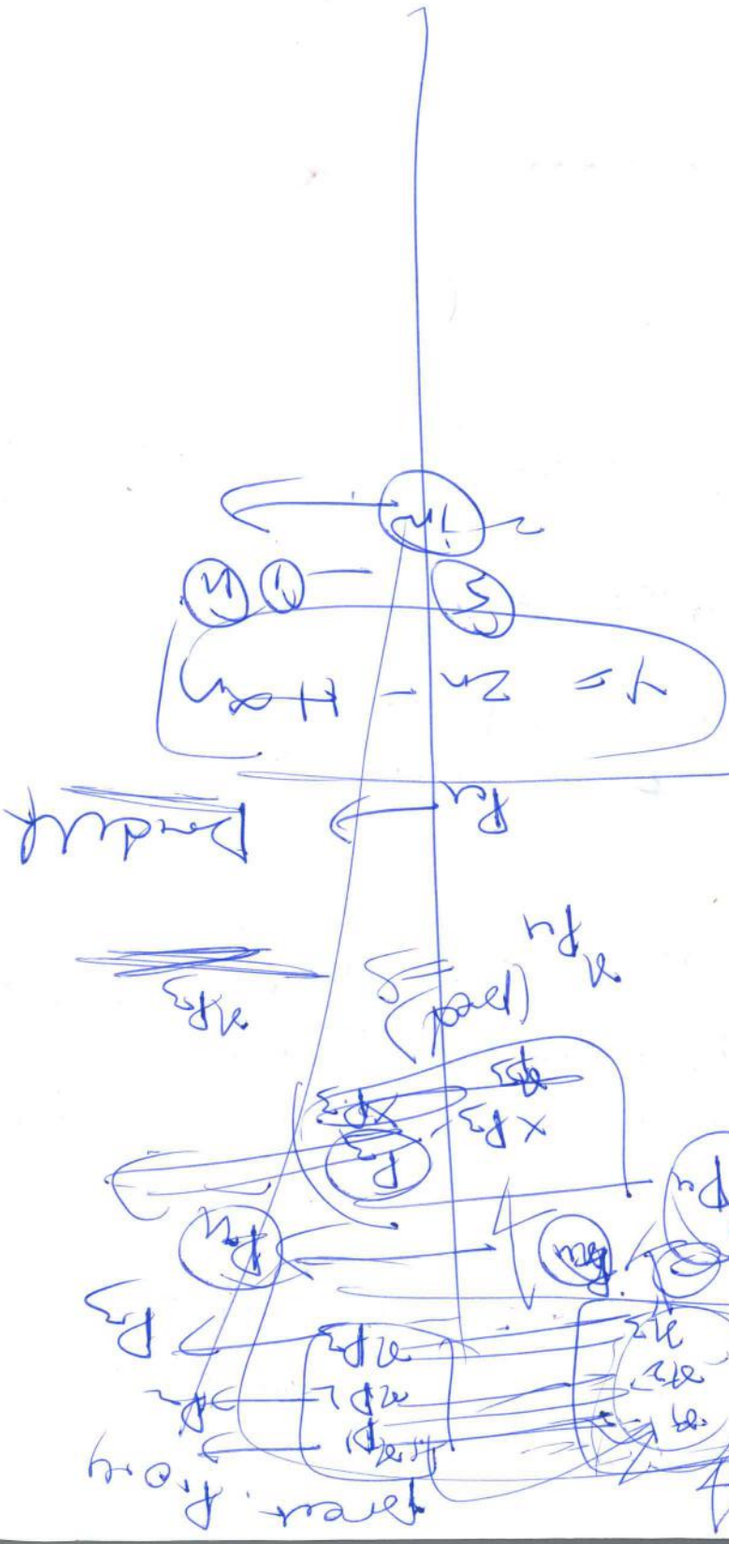
Logestic Regression

Classification :-

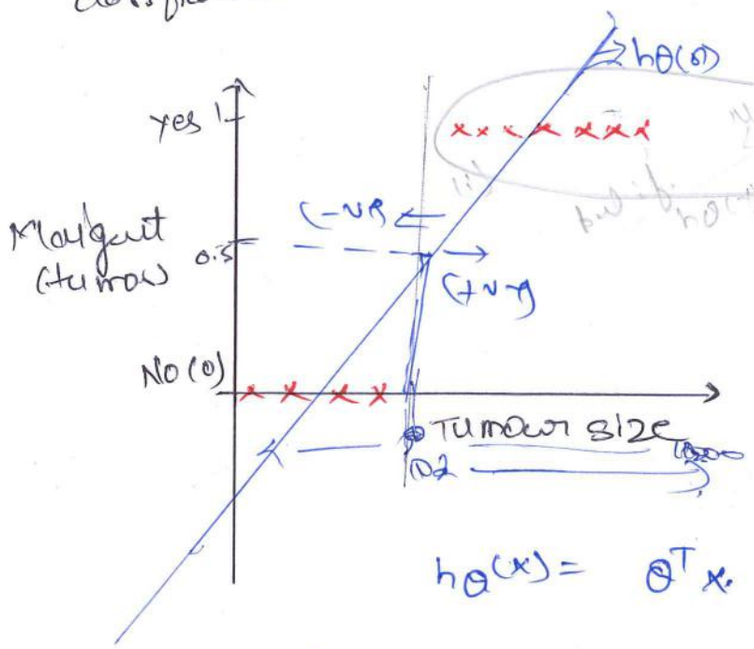
Email :- spam / Not spam ?

online :- fraudulent (Yes / NO) ?





multy classfication: $y \in \{0,1,2,3\}$ = classification



$h_\theta(x) = \theta^T x$.

If $h_\theta(x) \geq 0.5$, predict "$y=1$"
$h_\theta(x) < 0.5$, predict "$y=0$"

→ For Classification: $y = 0$ or 1
$h_\theta(x)$ can be > 1 or < 0 } so error not good.

→ So, we go to
Logistic Regression: $0 \leq h_\theta(x) \leq 1$
(Also name of classification)

Logistic Regression: Hypothesis Regression.

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

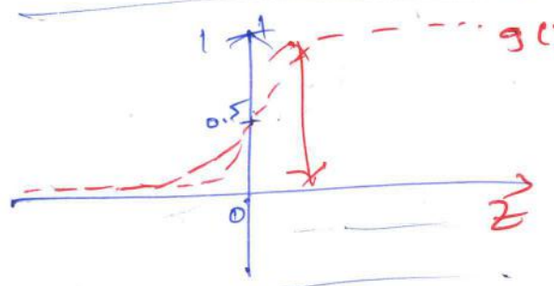
So $h_\theta(x) = \theta^T x$.

↳ $h_\theta(x) = g(\theta^T x)$
$= g(z)$

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

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

Sigmoid function or Logistic function.



Interpretation of Hypothesis Output

$h_\theta(x)$ = estimated probability that $y=1$ on input x

Ex: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$y=1$

$h_\theta(x) = 0.7$, probability of $y=1$ is 0.7

(or) 70% chance of tumor being malignant.

$h_\theta(x) = P(y=1 | x; \theta)$ ⇒ "probability that $y=1$, given x, parameterized by θ"

→ As this is Classification $y = 0$ or 1.

so we can calculate $y=0$ for given

$P(y=0 | x; \theta) = 0.3$.

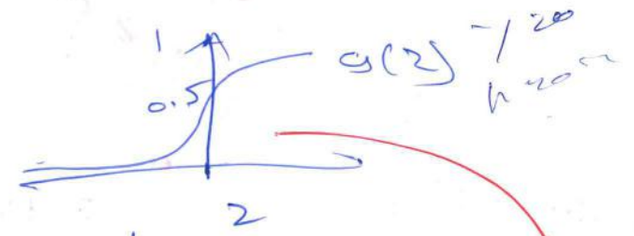
as $P(y=1 | x; \theta) + P(y=0 | x; \theta) = 1$

$P(y=0 | x; \theta) = 1 - P(y=1 | x; \theta)$ $= 1 - 0.7 = 0.3$

Decision Boundary

Logistic:-

$h_\theta(x) = g(\theta^T x) = P(y=1 | x; \theta)$



Suppose predict "$y=1$" if $h_\theta(x) \geq 0.5$

predict "$y=0$" if $h_\theta(x) < 0.5$

→ To understand better, we need when is y given $h_\theta(x) \geq 0.5$.

when $g(z) \geq 0.5$

when $z \geq 0$

so $h_\theta(x) = g(\theta^T x) \geq 0.5$, when $z \geq 0$.

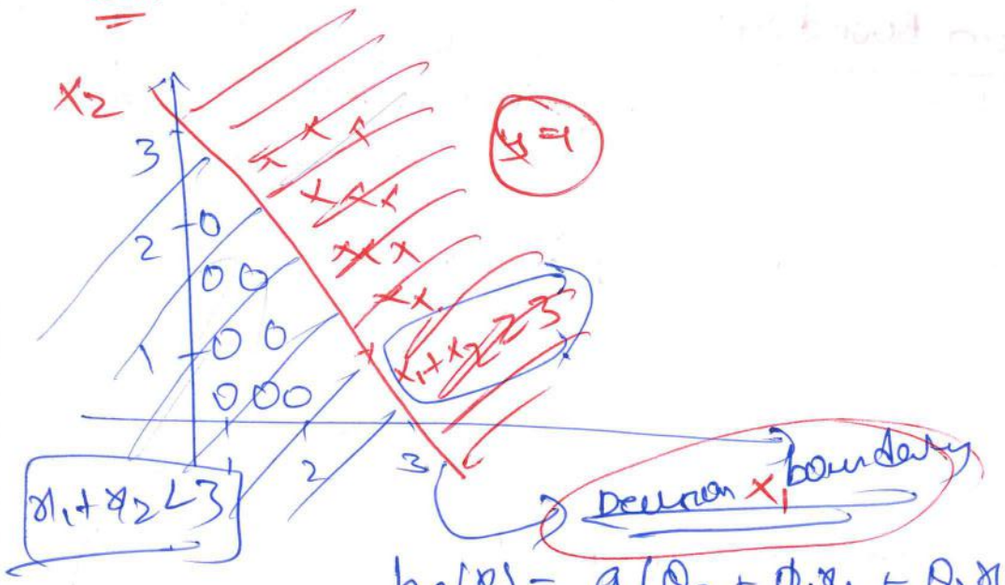
$\theta^T x \geq 0$

$g(\theta^T x) \geq 0.5$

→ Similarly $h_\theta(x) < 0.5$, when

$\theta^T x < 0$

Ex1)-



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

$-3 \quad 1 \quad 1$

Predict $\hat{y}=1$, if $(-3 + x_1 + x_2 \geq 0)$

$x_1 + x_2 \geq 3$

$y=1$, If $x_1 + x_2 \geq 3$

~~$x_1 + x_2$~~

$y=0$, if $(-3 + x_1 + x_2 < 0)$

$x_1 + x_2 < 3 \Rightarrow y=0.$

→ ~~These are used~~,

⇒ Decision boundary is a property of $h_\theta(x)$ including $\theta_0, \theta_1, \theta_2$.

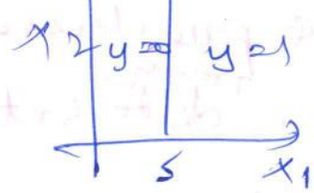
↳ These are in genuine property of hθ(x) not dataset

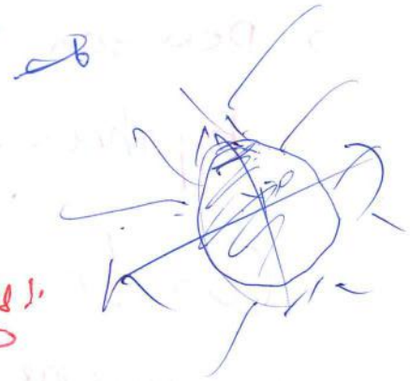
Ex1-

$h_\theta(x) = g(5 - x_1)$, $\theta_0 = 5$, $\theta_1 = -1$, $\theta_2 = 0$

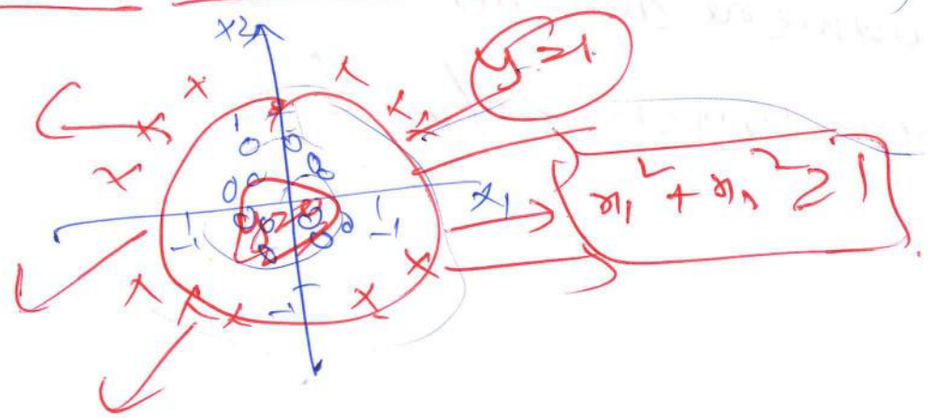
$5 - x_1 \geq 0$

$5 = x_1 \Rightarrow$ decision boundary





Non linear decision boundaries:



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

$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

Predict $y = 1$ if $-1 + x_1^2 + x_2^2 \geq 0$

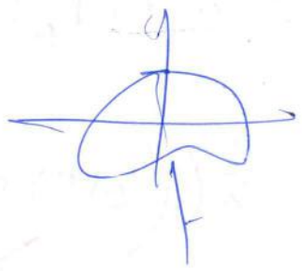
$x_1^2 + x_2^2 \geq 1 \Rightarrow \begin{cases} x_1 = 0, & x_2 \geq \pm 1 \\ x_2 = 0, & x_1 \geq \pm 1 \end{cases}$

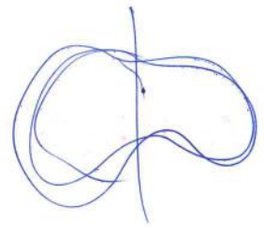
→ Decision boundary is the parameters of hypothesis, not training set. Training dataset.

Can be used to fit the $h_\theta(x)$ parameters and then the decision boundary is defined.

* More complex decision boundary

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_3^2 + \dots)$





Logistic Regression: cost function: How to auto fit parameters data to hθ to fit training set

eg:

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

m examples $x \in \begin{Bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{Bmatrix}$, $x_0 = 1, y \in \{0, 1\}$

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

How to choose. θ?

cost function

$\frac{1}{\text{train}} = J(\theta) = \frac{1}{m} \sum \frac{1}{2}(h_\theta(x^{(i)}) - y^{(i)})^2$

Logistic Regression $= \text{cost}(h_\theta(x^{(i)}), y)$

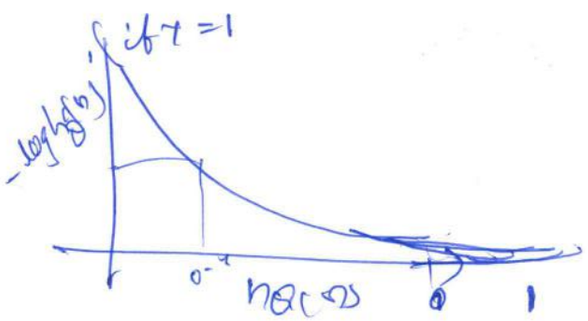
$\text{Cost}(h_\theta(x), y) = \frac{1}{2}(h_\theta(x) - y)^2$ where $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

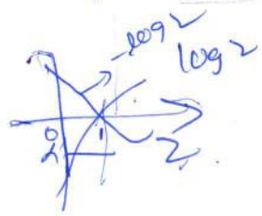
→ * So we cannot use the same cost function for logistic, inorder to avoid wavy.

Logistic regression cost function.

so use

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





→ Cost = 0, if $y=1$, $h_\theta(x) = 1$

But as $h_\theta(x) \to 0$

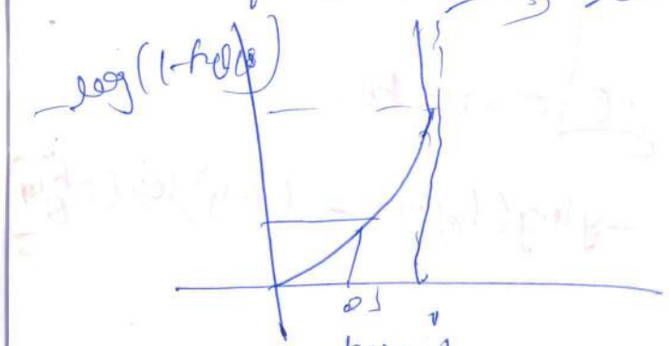
Cost $\to \infty$

If $h_\theta(x) = 0$,

(Predict, $P(y=1|x;\theta) = 0$) but $y=1$

we'll penalize learning algorithm by very large cost.

→ If $y=0$ → large cost.



Sim

$\frac{\partial}{\partial \theta}\left[\frac{1}{1+e^{-\theta^T x}}\right]$

$-\log\left(\frac{1}{1+e^{-\theta^T x}}\right)$

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

$-\log(1-h_\theta(x))$ if $y=0$

$-\log\left(\frac{1}{h_\theta(x)} - 1\right)$ if $y=1$

$-\mu(\text{more} - 1)$

Simplified cost function and Gradient descent.

Logistic regression CF:

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

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

Note $y = 0$ or 1 always

So the cost function can be rederived to

comprises ② in ① equation

$$\text{Cost}(h_\theta(x), y) = -y\log(h_\theta(x)) - (1-y)\log(1-h_\theta(x))$$

If $y=1$; $\text{cost}(h_\theta(x), y) = -\log(h_\theta(x))$

If $y=0$; $\text{cost}(h_\theta(x), y) = -\log(1-h_\theta(x))$

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

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

To fit parameters θ:

$\min_\theta J(\theta)$ → Get θ

To make a prediction given new x.

Output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} \quad P(y=1|x;\theta)$

→ By Gradient descent

want $\min_\theta J(\theta)$

Repeat {

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

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

(simultaneously update all θ_j)

}

Logistic Regression : Advanced optimization

cost function $J(\theta)$. Want $\min_\theta J(\theta)$

Given θ, → code to compute
- $J(\theta)$
- $\frac{\partial}{\partial \theta_j} J(\theta)$

Gradient descent

Repeat {

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

}

Optimization Algorithms

→ Gradient descent
- conjugate gradient
- BFGS
- L-BFGS

} Out of store

Examples:

$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$ $J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$

$\min_\theta J(\theta)$

$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$

$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$

~~# octave example~~

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

Gerlugem

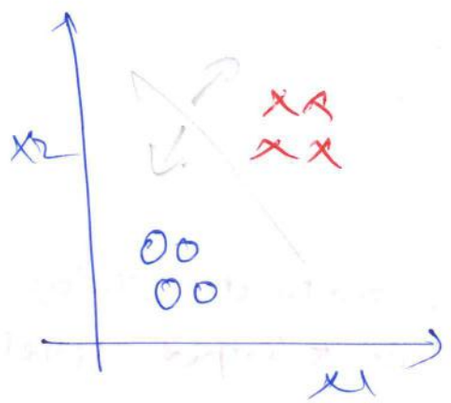
```
function [jVal, gradient] = costFunction(theta)
  jVal = [- - - code to compute J(theta)]
  gradient = [- - - code to compute derivative
                d J(theta)];
end.
```

Logestic Regression : MULTI CLASSIFICATION : One-Vs-all.

Email :- work, friends, Family, Hobby
↑ y=1 ↑ y=2 ↑ y=3 ↑ y=4

Medical diagrams: ~~Not ill~~, cold, Flu
↑ y=1 ↑ y=2 ↑ y=3

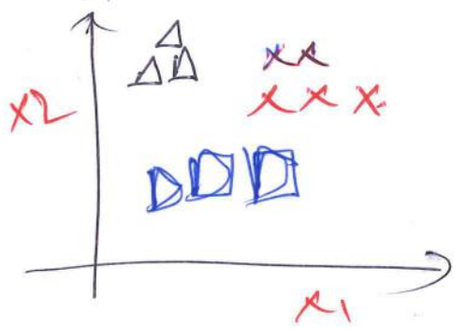
Weather : Sunny, Cloudy, Rain, Snow
y=1 y=2 y=3 y=4
0 1 2 3



Binary Classifier

Multi-class Classification
(one-vs-all)

one-vs-all (one-vs-rest)

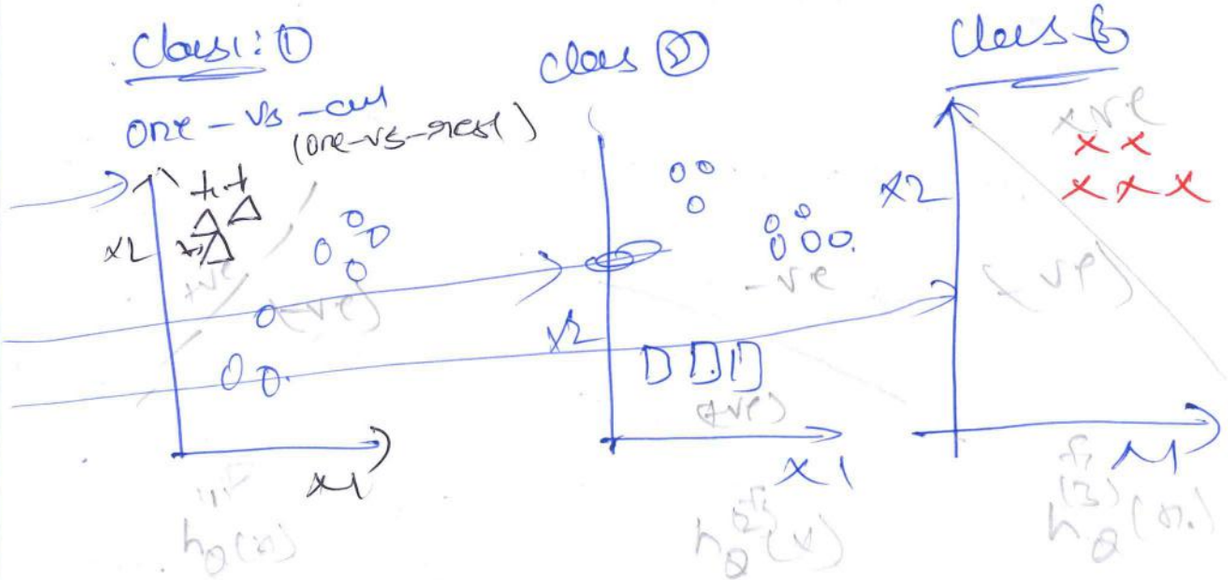


Class 1 : △
Class 2 : □
Class 3 : x

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

one-vs-all Classifier $h_\theta(x)$ for every class of i to predict the probability that $y=i$.

On a new input x, to make a prediction, pick the class i that maximises $\max_i h_\theta^{(i)}(x)$.



eg: Quiz

x_0	x_1	x_2	y
1	1	0.5	0
1	1	1.5	0
1	2	1	1
1	3	1	0

$$h = \theta^T x$$

$$\begin{bmatrix} h_0 \\ h_1 \\ h_2 \end{bmatrix} = \theta^T \begin{bmatrix} \theta_0 & \theta_1 & \theta_2 \end{bmatrix} \begin{bmatrix} x_0^0 & x_0^1 & x_0^2 & x_0^3 \\ x_1^0 & x_1^1 & x_1^2 & x_1^3 \\ x_2^0 & x_2^1 & x_2^2 & x_2^3 \end{bmatrix}$$

$$h = \begin{bmatrix} \theta_0 + \theta_1 + 0.5\theta_2 \\ \theta_0 + \theta_1 + 1.5\theta_2 \\ \theta_0 + 2\theta_1 + \theta_2 \\ \theta_0 + 3\theta_1 + \theta_2 \end{bmatrix} \quad y = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\cos(h_\theta(x) - y) =$$

$$h = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

$\theta_0 = -x_1 \geq 6$

$x_1 \leq 6$

$\theta_0 + x_1 \geq 0$

$x_1 \geq 0$

$\theta_0 + \theta_1 \geq 0$

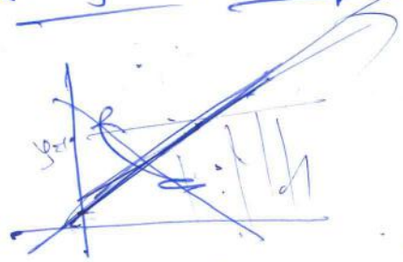
$\theta_0 + 3\theta_1 \geq 0$

$x_1 \geq 0$

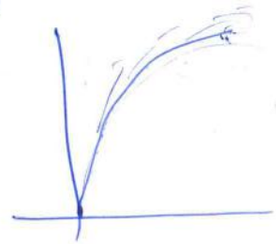
$x_1 \geq 0$

$x_1 \geq 0$

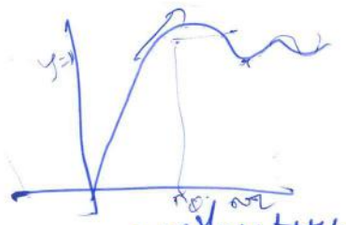
* How to reduce overfitting



underfit

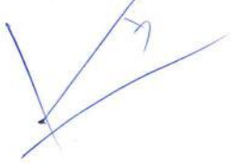


OK

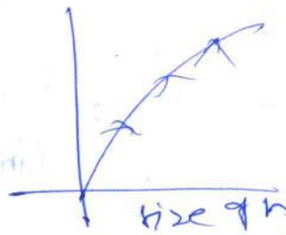


overfitting

Can be reduced by reducing the no of parameters.



cost function



size of tree

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

size of tree

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

$$\rightarrow \min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\theta_3^2 + 1000\theta_4^2$$

$\theta_3 \approx 0, \theta_4 \approx 0$

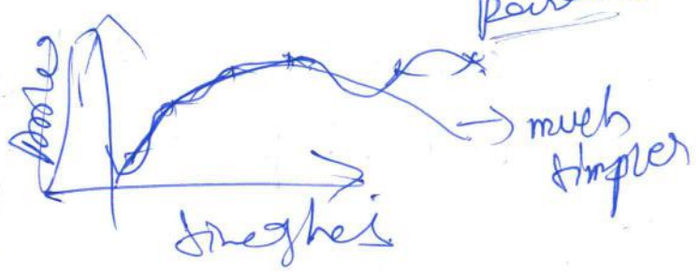
$\rightarrow$ Smaller values of $\theta_0, \theta_1 \ldots \theta_4$

Less overfitting

Regularized

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

↑ regularization parameter



Regularized Linear

→ Gradient descent
(linear regression)

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

$j = 0, 1, 2, 3 \dots n$

* Gradient descent
linear regression

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

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

$j = 1, 2, 3 \dots n$

Regularized

$$\theta_j = \theta_j - \alpha\left[\frac{1}{m}\sum(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j\right]$$

$j = 1$

Regularized $J(\theta)$

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

$\therefore (1 - \alpha\frac{\lambda}{m}) < 1$

$\theta_j = 6000$

→ 0.99

$\theta_j \times 0.99$ → θ_j by a little

Normal equation

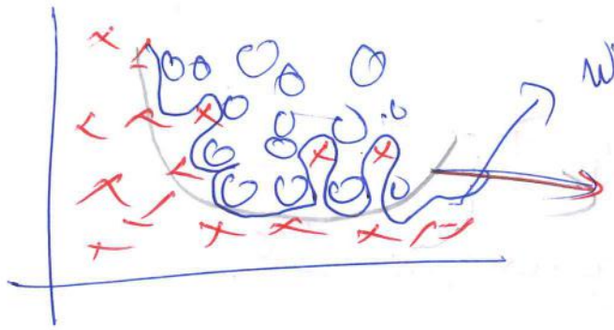
$$X = \begin{bmatrix} x^{(1)T} \\ \vdots \\ x^{(m)T} \end{bmatrix}$$

$$y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^n$$

$\min J(\theta)$

$\theta = (X^TX + \lambda \begin{bmatrix} 0 & & \\ & 1 & \\ & & \ddots \end{bmatrix})^{-1} X^Ty$

Regularised logistic Regression



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

$\theta_1 \dots \theta_n$

Gradient =

Repeat {

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

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

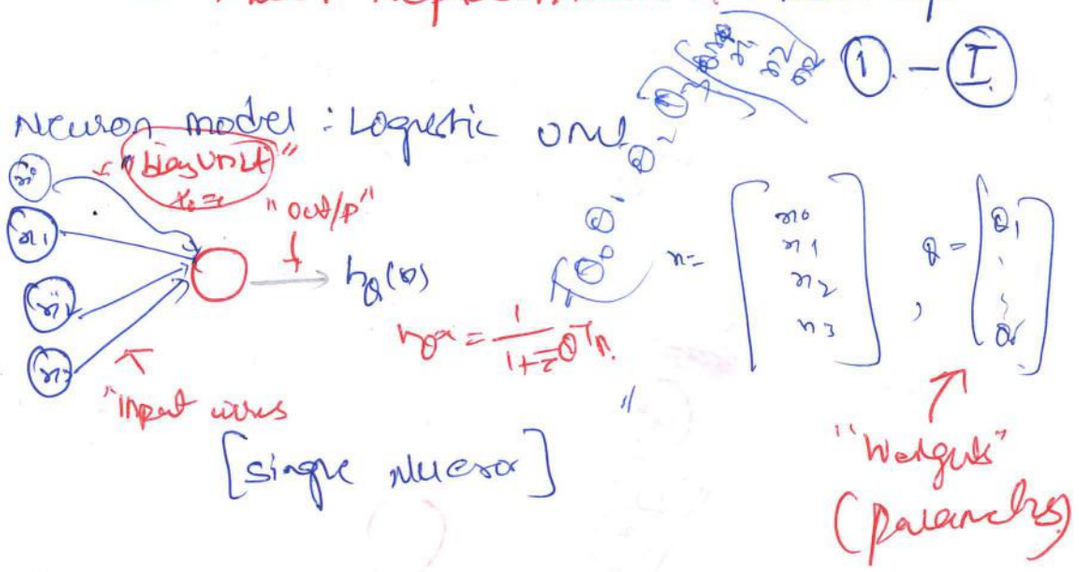
}

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

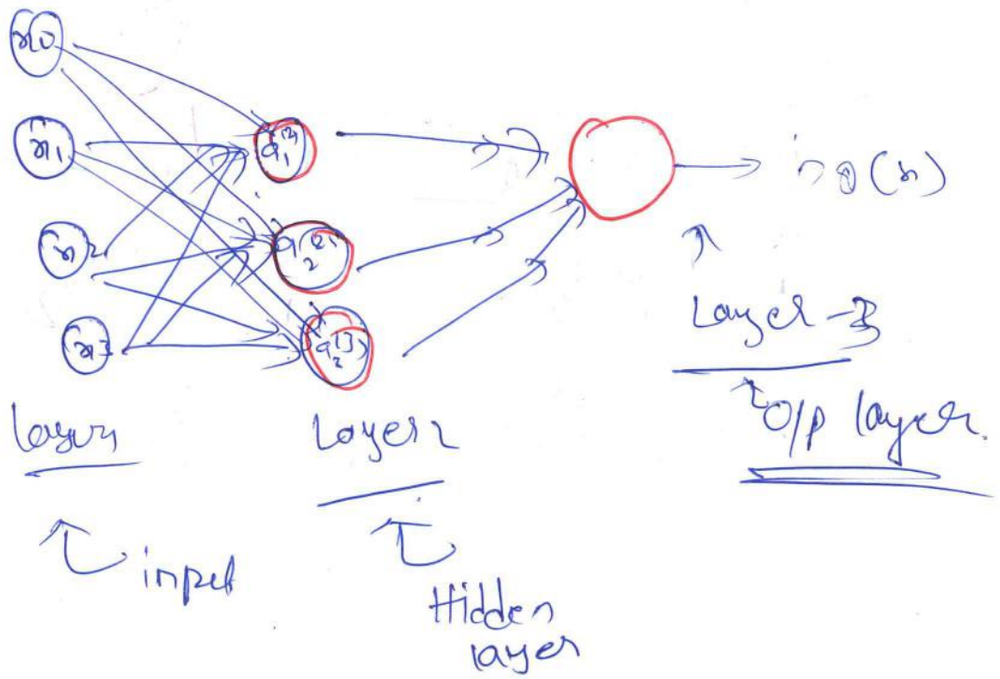
WEEK-4

NEURAL NETWORK: REPRESENTATION

— Model Representation: Model Representation

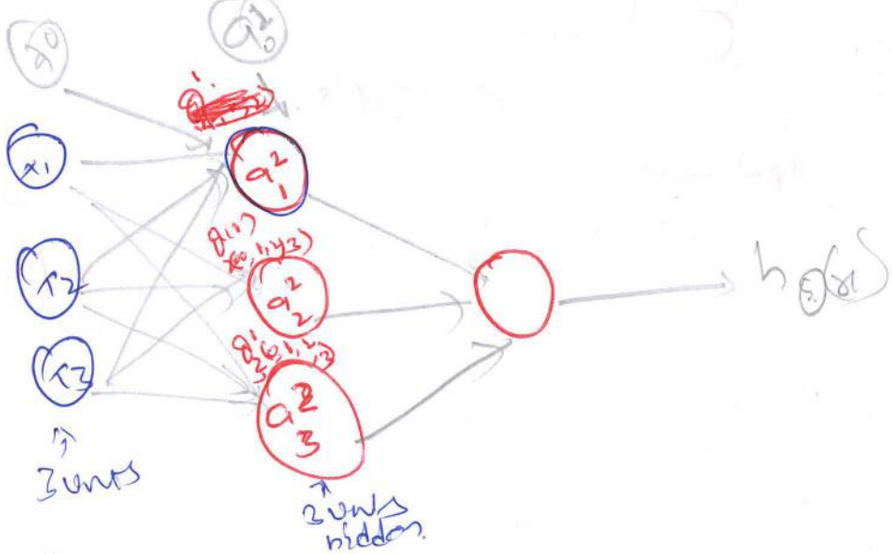


Neural Network [Multiple]



a_i^j = "activation" of unit i in layer j

Θ^j = matrix of weights controlling function mapping from layer j to layer $j+1$



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

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

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

If in layer j: $= s_j$ units

If in layer $j+1 = s_{j+1}$ units

Then $\Theta^{(j)}$ with dimension = $s_{j+1} \times (s_j + 1)$

↑ increment.

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

Model representation: II

consider here.

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

$$a_2^{(3)} = g(z_2^{(3)})$$

$$a_3^{(3)} = g(z_3^{(3)})$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

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

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

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

Add $a_0^1 = 1$

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

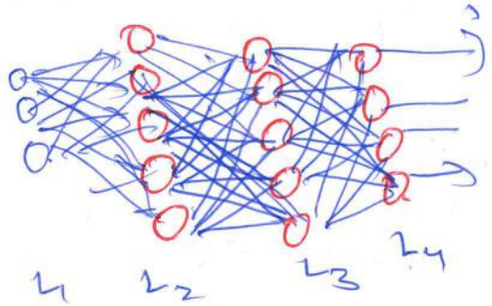
$h_\Theta(x) = a^3 = g(z^3)$

WEEK: 5

NEURAL NETWORKS: ~~Representation~~ LEARNING.

COST FUNCTION FOR FITTING Parameters

Neural Networks (classification)



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

L = total no of layers = 4

s_l = no of units (not counting bias unit)

$s_1 = 3$, $s_2 = 5$, $s_3 = 4$

$s_4 = s_L = 4$

binary classification

only one O/P

so

$h_\theta \in \mathbb{R}$

$s_L = 1$

multiclass

$h_\theta(x) \in \mathbb{R}^K$

K = no of O/P.

$s_L = K$, $(K \geq 3)$

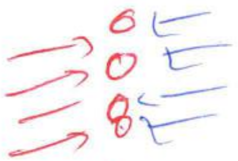
cost func (from logistic regression)

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

⟹ In general for a neural network we have 'k' o/p more '1's. i.e., $h_\Theta(x) \in \mathbb{R}^k$, $(h_\Theta(x))_i = i^{th}$ o/p.

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

For k o/p
if k=1 / Binary



$y_k = \begin{bmatrix} 0 \\ 0 \\ \vdots \end{bmatrix}$

No regularization of bias unit, we need only sums Θ_{i0}

BACK PROPOGATION : MIN COST function

$\min_\Theta J(\Theta)$

↳ need to compute $-J(\Theta)$

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

i = Length of each $x_j = x_j^1, x_j^2, x_j^3 \dots$ (car1, ...

j = No of e

Remember

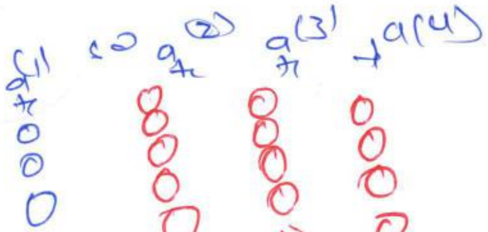
i = length of each $x_j = x_j^1, x_j^2 \dots$ = (car1, car2, car3 ...

j = No of pixels = (car, bus, lorry ...)

→ Gradient computation

Given one training example (x, y)

not of $x^{(i)}, y^{(i)}$



$a^{(1)} = x$

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

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

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

$a^3 = g(z^3)$ (add a_0^3)

$z^4 = \Theta^3 a^3$

$a^4 = g(z^4)$ (add a_0^4)

$\hookrightarrow h_\Theta(x) = g(z^4)$

for one (x, y)

$\Leftrightarrow$

: Back propagation algorithm

→ Intuition: Consider an error, $\delta_j^{(l)}$ = error of node j in layer l.

For each output unit (layer $L = 4$)

$\delta_j^{(4)} = a_j^{(4)} - y_j$

$= h_\Theta(x)_j - y_j$

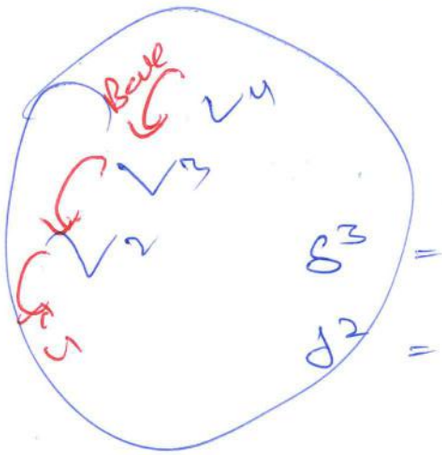
In vectorial $\delta^{(4)} = a^{(4)} - y$

$\delta^3 = (\Theta^{(3)})^T \delta^{(4)} .* g'(z^3)$ → $g'(z^3) = a^3 .* (1 - a^3)$

$\delta^2 = (\Theta^{(2)})^T \delta^{(3)} .* g'(z^2)$ → $g'(z^2) = a^2 .* (1 - a^2)$

↳ Steps of derivation

Back



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

(ignore λ if $\lambda = 0$)

$(=$

For $(x^{(i)}, y^{(i)})$:- Algorithm

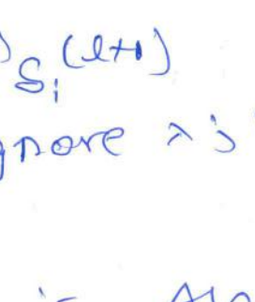
Set $\Delta_{ij}^{(l)} = 0$ (for all l, i, j)

(used to compute $\frac{\partial}{\partial \Theta_{ij}^{(l)}} J(\theta)$)

For $i = 1$ to m

Set $a^{(1)} = x^{(i)}$

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



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

Compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots \delta^{(2)}$

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

[Writing Algm in Foreach] : Backpropagation Algm

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

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

For $i = 1$ to m

set $a^{(1)} = x^{(i)}$

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

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

compute $\delta^{(L-1)}, \delta^{(L-2)}, \ldots \delta^{(2)}$

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

→ After this

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

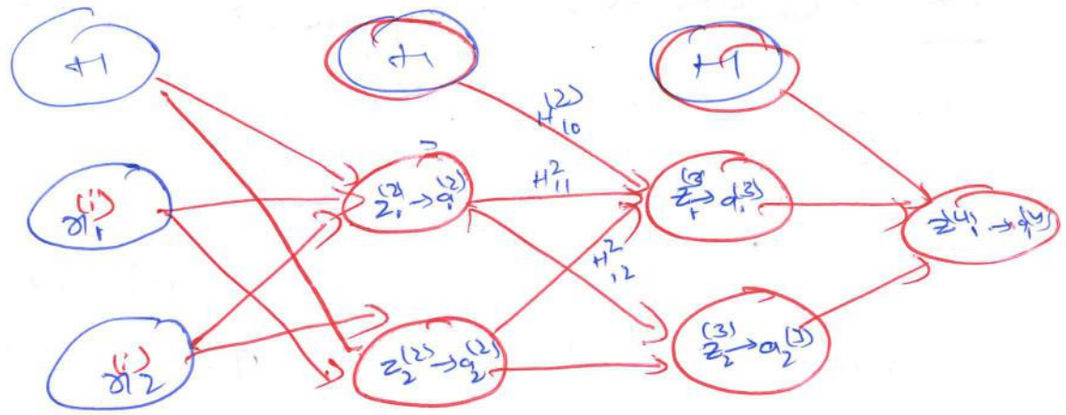
$D_{ij}^{(l)} := \frac{1}{m} \Delta_{ij}^{(l)}$

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

~~Out of Context:~~

Out of Context:

Forward propagation [~~Input units~~ Understanding Back Propagation]



$(x^{(1)}, y^{(1)})$

$z_1^{(2)} = \Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} a_1^{(1)} + \Theta_{12}^{(1)} a_2^{(1)}$

→ In general $\delta_{ij}^{(l)}$ = "error" of cost for $a_j^{(l)}$ (unit j in layer l)

→ Formally, $\delta_j^{(l)} = \frac{\partial}{\partial z_j^{(l)}} \text{cost}(i)$ (for $j \geq 0$), where

→ $\text{cost}(i) = y^{(i)} \log h_\Theta(x^{(i)}) + (1 - y^{(i)}) \log h_\Theta(x^{(i)})$

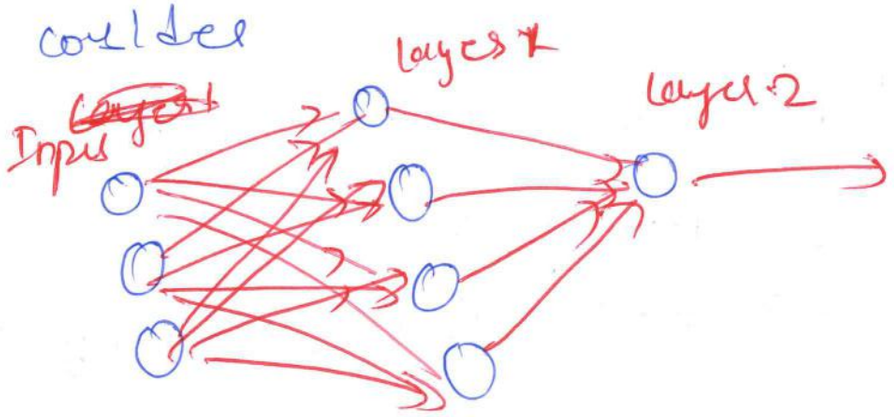
Back propogation In Practice

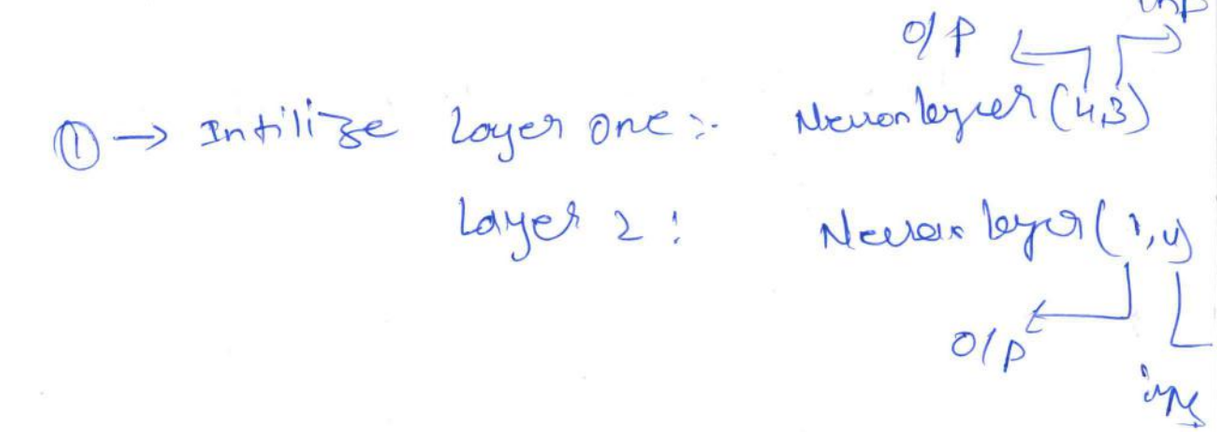
Implementation note: unrolling parameters.

Out of Coursera: Implementing multilayer neural networks. In general Single layer causes error when there is large input data in which it cannot receive to the o/p training data

So.

Steps for implementing





② → Intilize training set Input matrix

③ → Initilize training set Out put matrix

④ → Intilize random weights

⑤ → find o/p → find error b/w actual (training) & o/p and obtained

feed this ans

⑥ → find o/p from L_2 by feeding L_1 input to L_2

(7) → find real L_2 and O/P L_2 error

(8) → Using error from 'L' find ΔL_2, then for ΔL_2 can be found using diagonal feature derivative and input.

(9) → Use ΔL_2 to find ΔL_1

(10) → Adjust weights

(11) → Again perform (3)

(1)

Debugging learning Algoritham:

When new test is performed. We find that there is the huge different between actual & predicted

We do follows

→ Get more training set. ✗ (bad idea) → fixes high bias (high variance)

→ Try smaller sets of features → fixes high variance

→ Try getting additional features → fixed high bias

→ Try adding polynomial features ($x_1^2, x_2^2, x_1 x_2$, ...etc) → fixes high bias

→ Try decreasing λ → fixes high bias

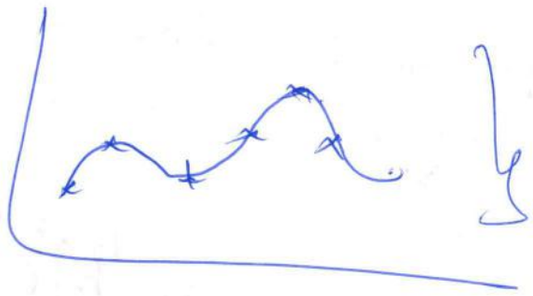
→ Try increasing λ → fixes high variance.

↳ so we find what to do.

Machine learning Diagnostic

Diagnostic:- finding if it is/isn't working and gives guidance on how to improve best

Evaluating your dataset



→ Here the curve is hard and can be error.

Data:

Size	Price
x_1^1	x_2^1
x_1^2	x_2^2
x_1^3	x_2^3
x_1^4	x_2^4
x_1^5	x_2^5
x_1^6	x_2^6
x_1^7	x_2^7
x_1^8	x_2^8
x_1^9	x_2^9
x_1^{10}	x_2^{10}

Training data

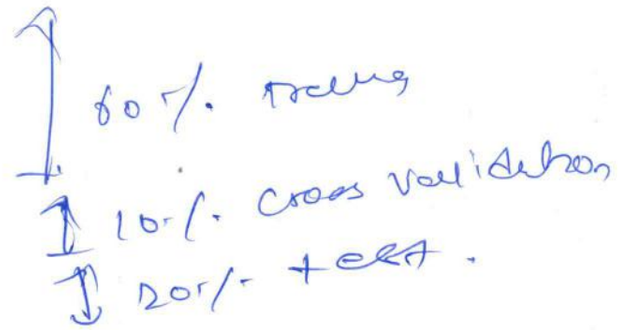
Test data

70% Training dataset

30% Test dataset

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

$(x^{(1)}_{test1}, x^{(1)}_{test2}, \ldots x^{(3)}_{test1}, x^{(3)}_{test2})$



So →

Training error:

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

Cross validation error:

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

Test error

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

Model selection

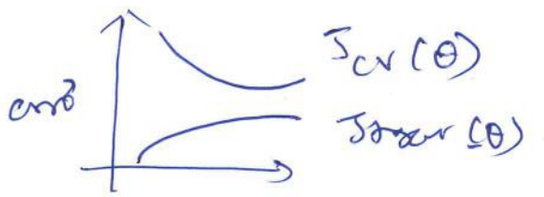
1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x x \ldots \theta_{10} x^{10}$

Choose best hang. $\min(J_{CV})$

→ Apply $J_{test}(\theta)$

Learning curves :-

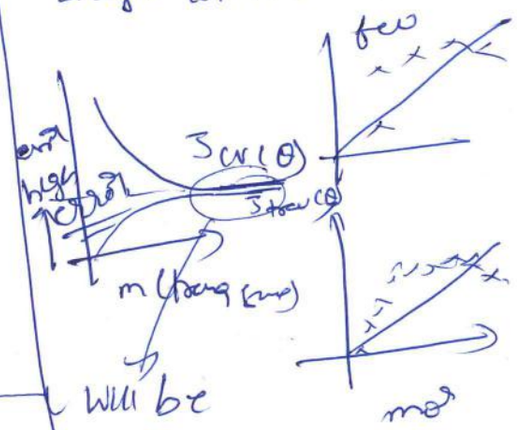
$J_{train}(\theta)$ $J_{cv}(\theta)$



For high bias

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

Image lot more training

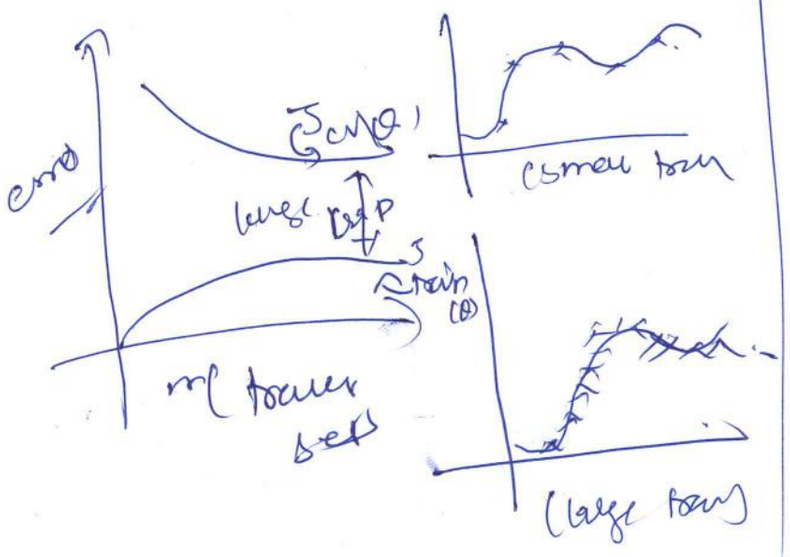


Will be close

→ high bias, high error with $J_{cv}(\theta)$, J_{train} almost same.

For high variance

$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_{100} x^{100}$



Small Neural Network

computationally cheaper

Large neural Networks

computationally more expensive

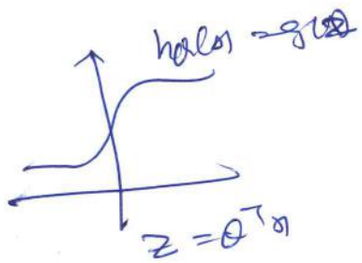
Use regularization (λ) to address overfitting.

SUPPORT VECTOR MACHINES

I) OPTIMIZATION objective :-

Alternate view of logistic regression

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



If $y = 1$, we want $h_\theta(x) \approx 1$, $\theta^T x >> 0$

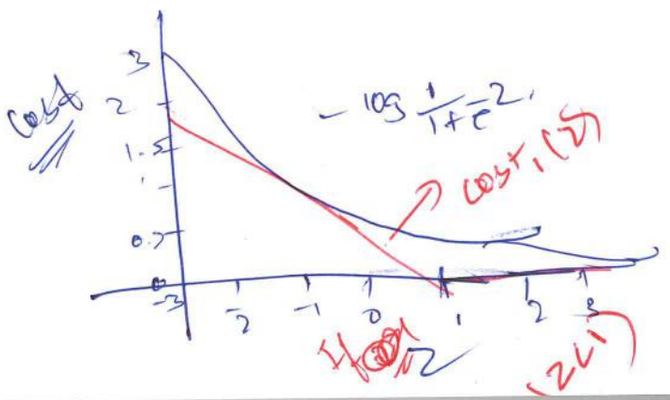
$y = 0$ " $h_\theta(x) \approx 0$, $\theta^T x << 0$

* cost function of logistic.

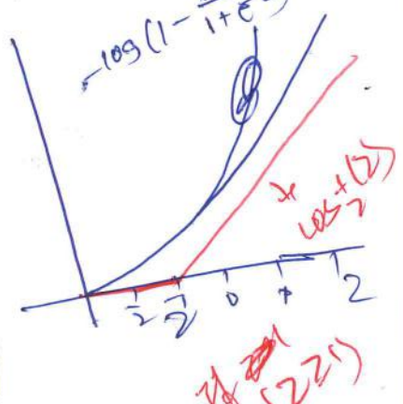
$$= -(y \log h_\theta(x) + (1-y) \log(1 - h_\theta(x)))$$

$$= -y \log \frac{1}{1+e^{-\theta^T x}} - (1-y) \log\left(1 - \frac{1}{1+e^{-\theta^T x}}\right)$$

If $y = 1$, (want $\theta^T x >> 0$)



If $y = 0$



SVM :

Logistic Regression :

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

Support Vector Machine:

$$\min_\theta \frac{1}{m} \sum_{i=1}^{m} y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

so

SVM Hypothesis :

$$\min_\theta C \sum_{i=1}^{m} \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

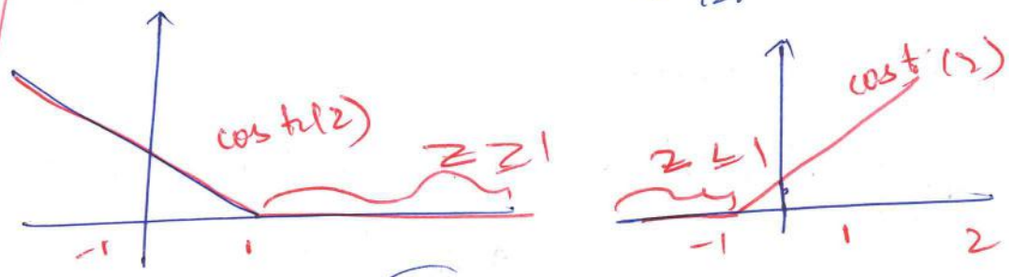
Hypothesis :

$$h_\theta(x) = \begin{cases} 1, & \text{if } \theta^T x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

SVM

SVM: Large Margin Classification

$$\min_\theta \; C \sum_{i=1}^{m} \left[y^{(i)} \text{cost}_1(\theta^T x^{(i)}) + (1-y^{(i)}) \text{cost}_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$



If $y=1$, we want $\theta^T x \geq 1$ (not just ≥ 0), $\theta^T x \not\geq 0$

$y=0$, we want $\theta^T x \leq -1$ (not just < 0), $\theta^T x \not< 0$

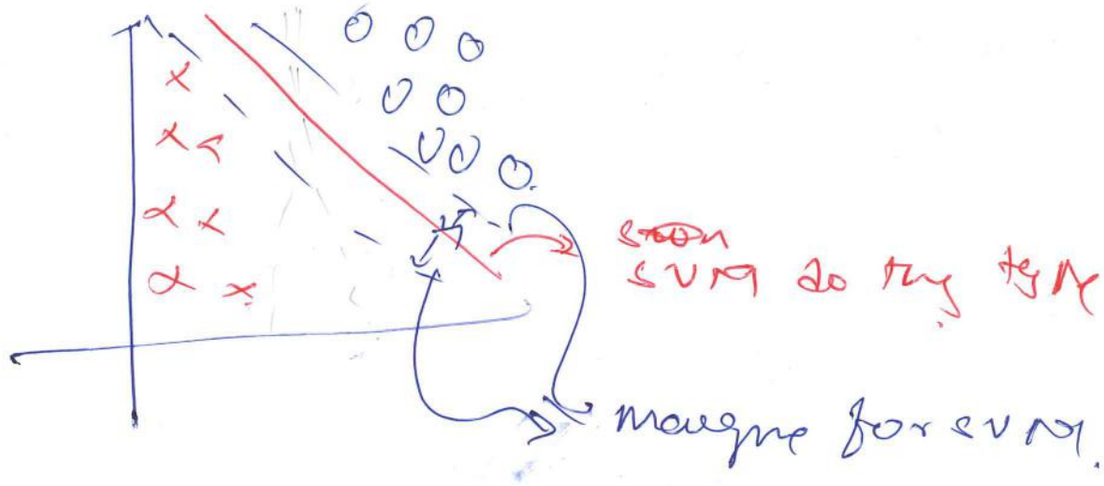
SVM does this

SVM :- Boundary ≥ 0.

So

$$\min \; C \cdot 0 + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

$$\text{s.t} \quad \theta^T x^{(i)} \geq 1 \quad \text{If } y^{(i)} = 1$$
$$\theta^T x^{(i)} < -1 \quad \text{If } y^{(i)} = 0$$



SVM with Kernels:

Hypothesis: Given x, compute features $f \in \mathbb{R}^{m+1}$

→ predict "$y=1$" if $\theta^T f \geq 0$

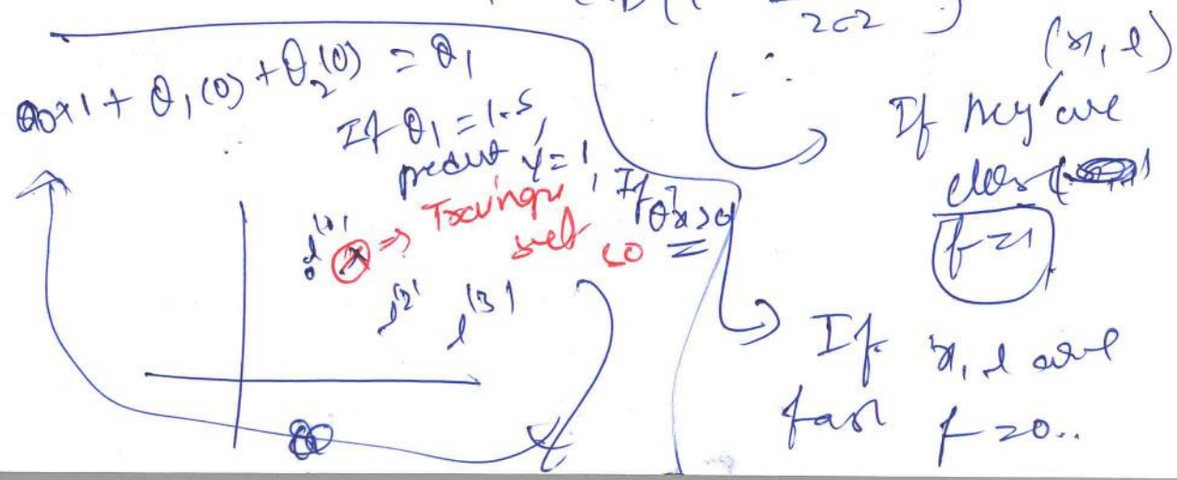
↳ $\theta_0 f_0 + \theta_1 f_1 + \cdots \theta_n f_n$

$\therefore$ $f_0 = x_0$

$f_1 = x_0 x_1$

$f_2 = x_0 x_2$ ..

$f_1 = \text{exp}\left(1 - \frac{\|x - l^{(1)}\|^2}{2\sigma^2}\right)$



Training

$$\min_{\theta} C \sum_{i=1}^{m} y^{(i)} \text{cost}_1(\theta^T f^{(i)}) + (1-y^{(i)}) \text{cost}_0(\theta^T f^{(i)}) + \frac{1}{2} \sum_{j=1}^{n=m} \theta_j^2$$

$\sum_{j=1}^{n} \theta_j^2 = \theta^T \theta$

$\theta = \begin{bmatrix} \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ (ignore θ_0)

$= \theta^T M \theta$ { minimize we equivalent for training.

TO use SVM efficiently

SVM Parameters:

$C (= \frac{1}{\lambda})$, Large C = lower bias, high variance
Small C = Higher bias, low variance

σ^2 Large σ^2: features f_i vary smoothly higher Bias, loud variance.

Logistic Regression Vs SVMs

n = number of features, ($x \in \mathbb{R}^{n+1}$), m = number of training examples

If n is large (relative to m):

↳ Use logistic regression & SVM without kernel (linear kernel)

→ If n is small, m is intermediate:

→ Use SVM with Gaussian kernel

$\therefore f = \exp\left(-\frac{(x-l)^2}{\sigma^2}\right)$

→ If n is small, m is large

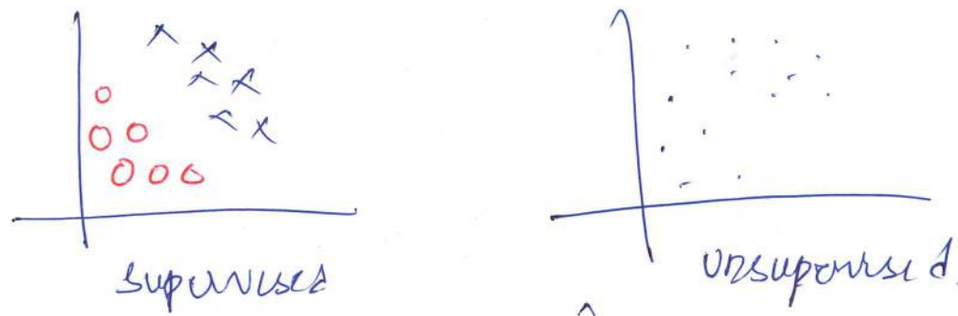
→ create/add more features, then use logistic regression & SVM without a kernel.

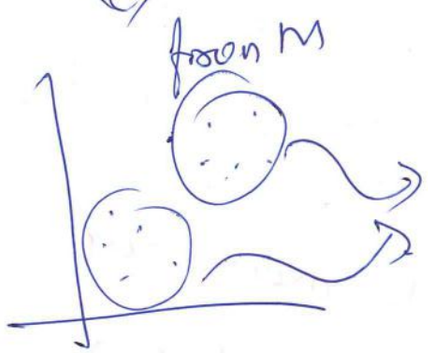
→ Neural Network works for most of these setting but may be slower to train.

Unsupervised learning

Unsupervised Learning (No Labeling as in supervised

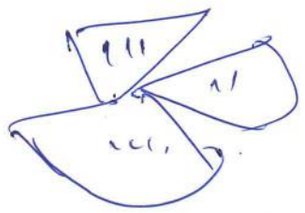
Introduction :-

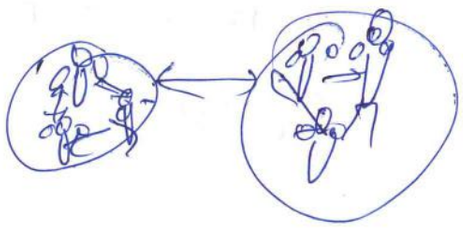




Clustering algorithm.

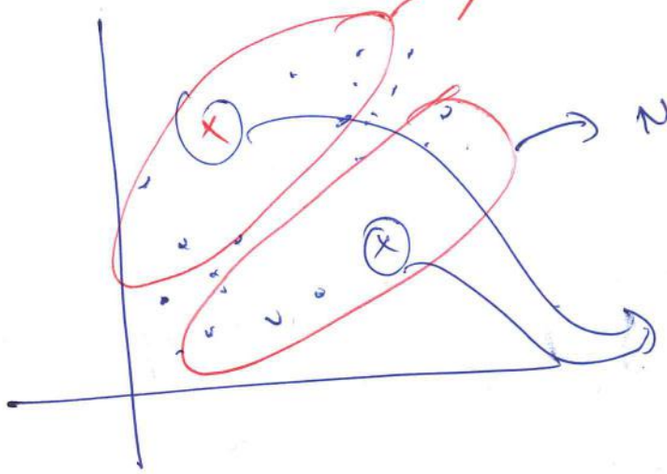
App of clustering





Social network.

Clustering : K-means Algorithm.



→ Each point is compared to cluster centroid

→ depends on their nearer centroid

→ More iterations of k-means are done, to find k-number of clusters.

K-means algorithm.

Input:

- K (number of clusters)
- Training set $\{x^{(1)}, x^{(2)}, x^{(3)} \ldots x^{(m)}\}$

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

K-means Algorithm:

Randomly initialize K cluster centroids $\mu_1, \mu_2, \mu_3 \ldots \mu_K$

Repeat {

Cluster assignment step

for $i = 1$ to m

$c^{(i)}$ = index (from 1 to K) of cluster centroid closest to $x^{(i)}$ ← min $\|x^{(i)} - \mu_k\|^2$ where $\mu_1, \mu_2 \ldots \mu_K$ are fixed

for $k = 1$ to K

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

}

Move centroid

min J w.r.t $\mu_1 \ldots \mu_K$

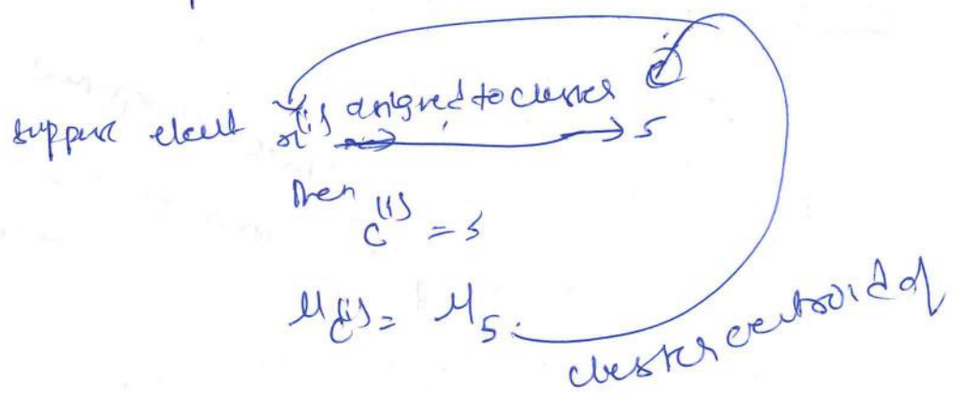
assigning/reassigning centroids to update centroids

K-Means optimization objective

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

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

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



Optimization objective

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

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

distance between centroid and object has to be minimized so that

they can be almost classified same.

K-Mean Initialization

see privious for~~Rooster~~

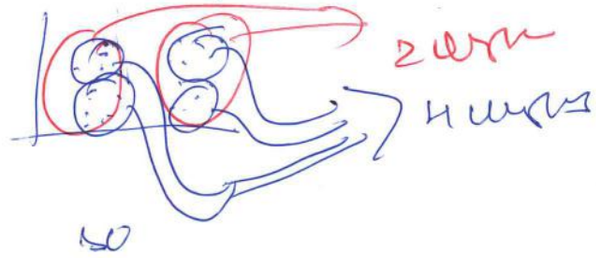
random initialization

→ should $K < m$.

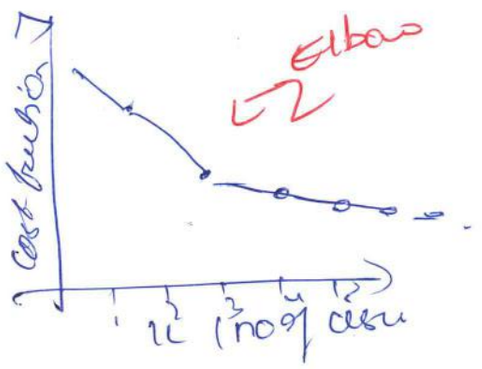
→ Randomly pick K training examples

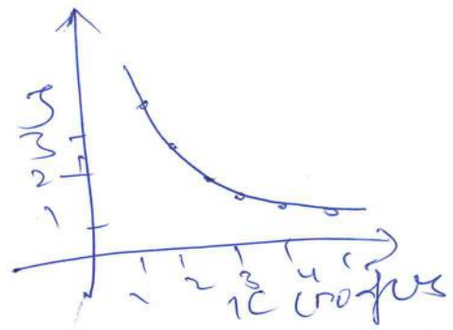
→ set $\mu_1, \ldots \mu_K$ equal to these K examples

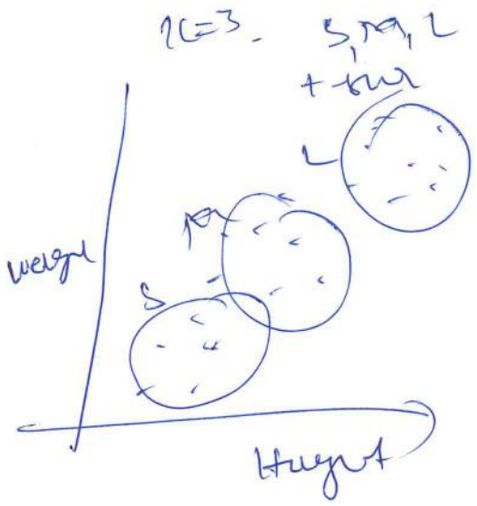
Choose no number of clusters

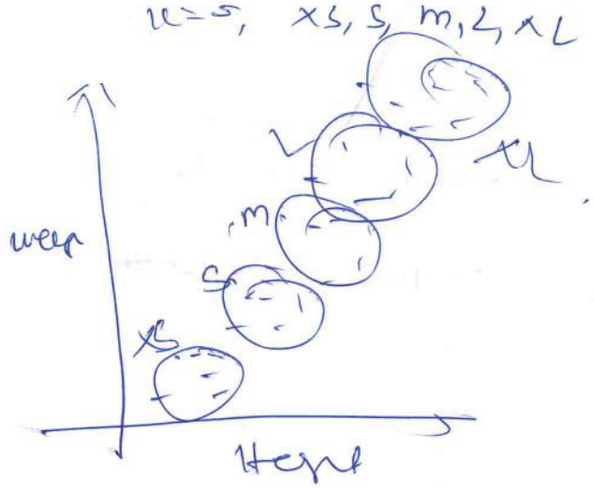


Elbow method

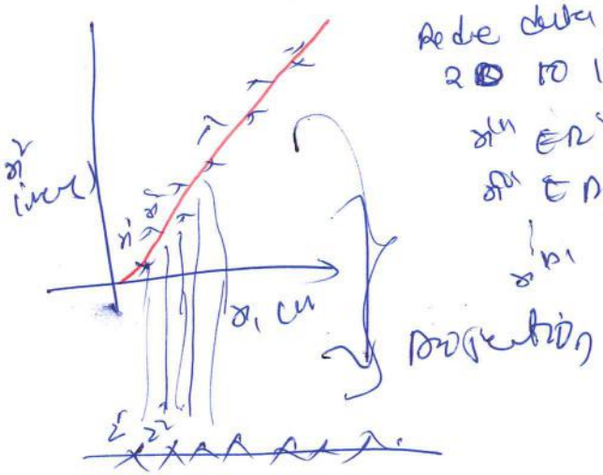








Data compression



Reduce data from 2D to 1D

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

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

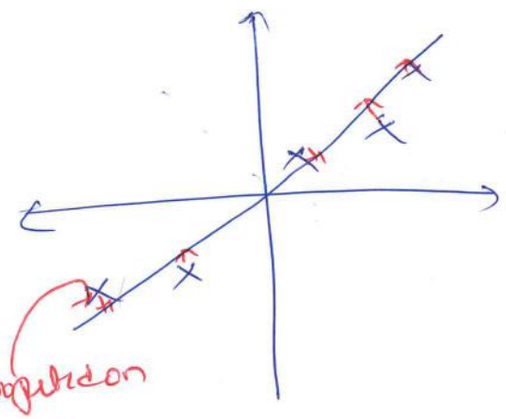
$\vdots$

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

DIMENSIONALITY REDUCTION:

PRINCIPAL COMPONENT ~~Algorithm~~ Analysis.

PCA: Formulation



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

→ low dimensional

→ PCA = does it minimum of sum of squared projected distance

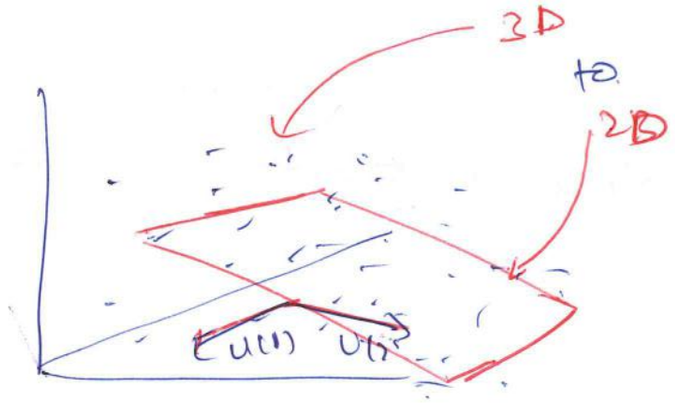
⇒

PCA :- Reduce 2D-to-1D with min projection..

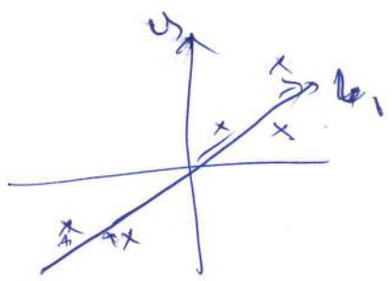
Find direction (a vector $u^{(1)} \in \mathbb{R}^n$).

→ Generalization

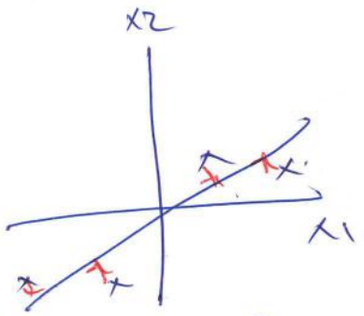
Reduce n-dimension to k-dimensions: find k-vectors $u^{(1)}, u^{(2)}, \ldots u^{(k)}$ onto which to project the data. so as to minimize the projection error.



→ PCA is NOT Linear regression



Linear regression we just consider x to predict y knowing y o/p.



PCA
→ minimizes the projection distance
→ 2D x_1, x_2 to Single 1D

PCA Algorithm

Data Preprocessing: Training set: $x^{(1)}, \ldots, x^{(m)}$

Scaling/mean Normalization:

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

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

If different features on different scales (e.g. x_1 = size of house, x_2 = number of bedrooms), scale features to have comparable ranges of values.

* → Reduce data from n-D to 1-D

Compute "covariance matrix"

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

→ Compute eigenvectors of Σ:

$[U, S, V] = \text{SVD}(\text{Sigma})$

Singular Value Decomposition

$$U_2 = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \cdots & u^{(n)} \\ | & | & & | \end{bmatrix}$$

$K \in \mathbb{R}^n \rightarrow \mathbb{R}^k$

take $x \in \mathbb{R}^n \Rightarrow$ to $z \in \mathbb{R}^k$

$U_{reduce} = [u^{(1)}\ u^{(2)} \dots u^{(k)}]$ — $n \times k$

$U_{ordered}$

Mathematical proof / Beyond the scope

$(U_{reduce})^T x^{(i)} = \begin{bmatrix} - u^{(1)T} - \\ - u^{(2)T} - \\ \dots \\ - u^{(k)T} - \end{bmatrix} x^{(i)}$ — $(n \times 1)$

$k \times 1$

Summary of PCA Algorithm

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

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

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

$z = U_{reduce}' * x;$

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

$Sigma = \frac{1}{m} * X' * X$

Reconstruction from compressed representation

$z = U_{reduce}^T x$

$x_{approx} = U_{reduce} \cdot z^{(1)}$

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

Choosing 'k' (No. of PCA)

Avg Square error : $\frac{1}{m} \sum_{i=1}^{m} (x^{(i)} - x^{(i)}_{approx})$

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

* Choose 'k' to be smallest value so that

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

→ 99% variance retained

Algorithm:- (le Choice)

Try PCA with $k=1$

Compute $U_{reduce}, z^{(1)}, z^{(2)}, \ldots z^{(m)}$

$x^{(1)}_{approx} \ldots x^{(m)}_{approx}$

Choose if

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

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

$S = \begin{bmatrix} S_{11} & 0 & 0 \\ 0 & S_{22} & 0 \\ 0 & 0 & S_{nn} \end{bmatrix}$

for que k

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

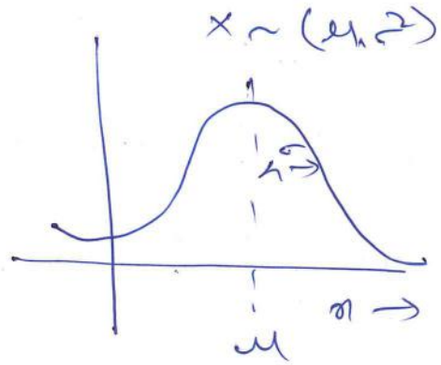
$k=3$

Density Estimation:

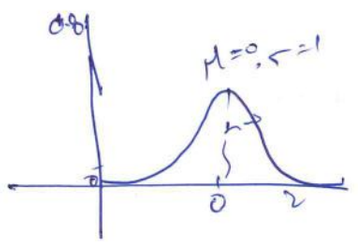
Gaussian distribution

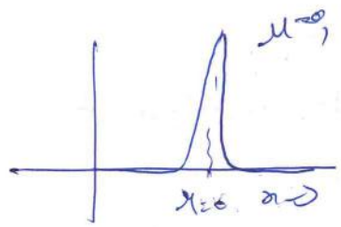
$x \in \mathbb{R}$, x is distributed with mean μ, variance σ^2

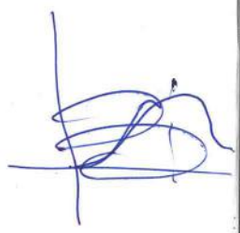
$x \sim (\mu, \sigma^2)$

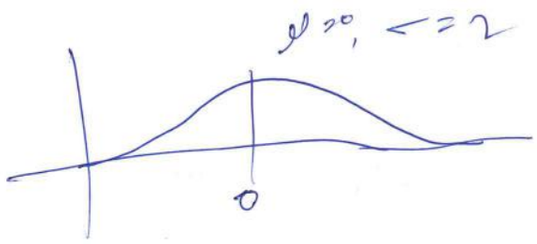


$$p(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$









Training set : $\{x^{(1)}, \ldots x^{(m)}\}$

Each example $x \in \mathbb{R}^n$

$x_i \sim N(\mu_i, \sigma_i^2)$

$P(x)$ = probability $= P(x_1; \mu_1, \sigma_1^2) \cdot P(x_2; \mu_2, \sigma_2^2) \cdots P(x_n; \mu_n, \sigma_n^2)$

Every of them has different Gaussian parameters

Anomaly detection algorithm (Assume Gaussian distribution even fault is different)

1. Choose features x_i, that In general responsible for fault.

2. Fit parameters $\mu_1, \mu_2, \ldots \mu_n, \sigma_1^2, \sigma_2^2, \ldots \sigma_n^2$

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

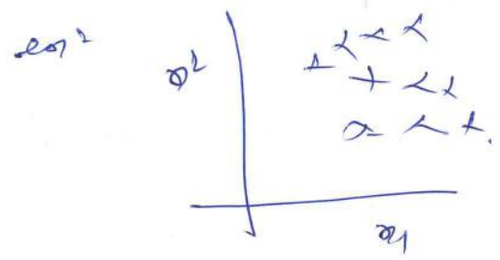
$$\sigma_j^2 = \frac{1}{m} \sum_{i=1}^{m} (x_j^{(i)} - \mu_j)^2$$

3. Given new example x, compute $P(x)$

$$P(x) = \prod_{j=1}^{n} P(x_j; \mu_j, \sigma_j^2)$$

$$= \prod_{j=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_j} \exp\left(-\frac{(x_j - \mu_j)^2}{2\sigma_j^2}\right)$$

Anomaly if $P(x) < \varepsilon$



PREDECTING MOVIE RATINGS:

Recommendation system. — Recommending products based on previous reviews for different type of films for a particular user.

n_u = no of users

n =

Problem Formulation :-

→ $r(i,j)$ = 1, if user 'j' has rated movie 'i'
(0, otherwise)

→ $y(i,j)$ = rating by user 'j' for movie 'i'
(if defined : if $r(i,j)=1$)

→ $\theta^{(j)}$ = parameter vector for user j

→ $x^{(i)}$ = feature vector for movie i

→ For user j, movie i, ~~predicted rating~~ : $(\theta^{(j)})^T(x^{(i)})$
Predicted rating : $(\theta^{(j)})^T(x^{(i)})$

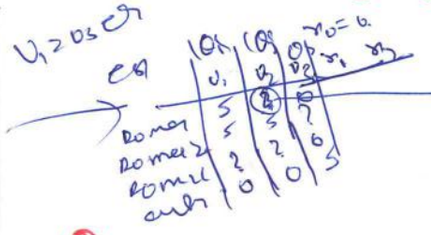
→ $m^{(j)}$ = no. of movies rated by user j

To learn $\theta^{(j)}$:

$$\min_{\theta(j)} \frac{1}{2m^{(i)}} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)} \right)^2 + \frac{\lambda}{2m^{(i)}} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

constant

constant



Optimization problem:

To learn $\theta^{(j)}$ (parameters for User j):

specific user

$$\min_{\theta(j)} \frac{1}{2} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

To learn $\theta^{(1)}, \theta^{(2)}, \ldots \theta^{(n_u)}$:

for all users

$$\min_{\theta^{(1)} \ldots \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} (\theta_k^{(j)})^2$$

So Finally:

Optimization algorithm:

$$\min_{\theta^{(1)},\dots,\theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1} (\theta_k^{(j)})^2$$

$$J(\theta_1, \dots, \theta^{(n_u)})$$

Gradient descent Update.

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)}\right) x_k^{(i)} \quad \text{for } k=0$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T (x^{(i)}) - y^{(i,j)}\right) x_k^{(i)} + \lambda \theta_k^{(j)}$$

(for $k \neq 0$)

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} x_0^1 \\ x_0^2 \end{bmatrix} = 0.$$

x_2

x_0

$(\theta^1)^T x^1 \simeq$

$$\begin{bmatrix} 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = 0; \qquad \begin{bmatrix} 0 & 3 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = 1.5$$

$$\begin{bmatrix} 0 & 5 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = 2.5$$

$3x_1 = 1.5$

$x_1 = \frac{1.5}{3}$

Collaborative Filtering

Given $x^{(1)}, x^{(2)}, \dots x^{(n_m)}$ (and movie ratings)

can estimate $\theta^1, \theta^2, \dots \theta^{(n_u)}$

Given $\theta^{(1)}, \dots \theta^{(n_u)}$

can estimate $x^{(1)}, \dots x^{(n_m)}$

Guess $\theta \rightarrow x \rightarrow \theta \rightarrow x \rightarrow \theta \rightarrow$

Collaborative Filtering optimization objective

Given $x^{(1)}, \ldots x^{(n_m)}$, estimate $\theta^{(1)}, \ldots \theta^{(n_u)}$

$$\min_{\theta^{(1)}, \ldots \theta^{(n_u)}} \frac{1}{2} \sum_{j=1}^{n_u} \sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(\theta_k^{(j)} \right)^2$$

Given $\theta^{(1)}, \ldots \theta^{(n_u)}$, estimate $x^{(1)}, \ldots x^{(n_m)}$:

$$\min_{x^{(1)}, \ldots x^{(n_m)}} \frac{1}{2} \sum_{i=1}^{n_m} \sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)} \right)^2$$

→ Minimize both simultaneously.

$$J(x^{(1)}, \ldots x^{(n_m)}, \theta^{(1)}, \theta^{(2)}, \ldots \theta^{(n_u)})$$

$$= \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_m} \sum_{k=1}^{n} \left(x_k^{(i)} \right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \left(\theta_k^{(j)} \right)^2$$

$$\min_{\substack{x^{(1)}, \ldots x^{(n_m)} \\ \theta^{(1)}, \ldots \theta^{(n_u)}}} J(x^{(1)}, \ldots x^{(n_m)}, \theta^{(1)}, \ldots \theta^{(n_u)})$$

Collaborative filtering algorithm

1. Initialize $x^{(1)}, \ldots x^{(n_m)}, \theta^{(1)}, \ldots \theta^{(n_u)}$ to small random values.
2. Minimize $J(x^{(1)}, \ldots x^{(n_m)}, \theta^{(1)}, \ldots \theta^{(n_u)})$ using gradient descent (or an advanced optimization algorithm). E.g. for every $j = 1, \ldots n_u$ $i = 1, \ldots n_m$

$$x_k^{(i)} := x_k^{(i)} - \alpha \left(\sum_{j:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) \theta_k^{(j)} + \lambda x_k^{(i)} \right)$$

$$\theta_k^{(j)} := \theta_k^{(j)} - \alpha \left(\sum_{i:r(i,j)=1} \left((\theta^{(j)})^T x^{(i)} - y^{(i,j)} \right) x_k^{(i)} + \lambda \theta_k^{(j)} \right)$$

$\frac{d}{d x_k^{(i)}} J$

Low Rank matrix factorization

$$Y = \begin{bmatrix} 5 & 5 & 0 & 0 \\ 5 & ? & ? & 0 \\ ? & 4 & 0 & ? \\ 0 & 0 & 5 & 0 \end{bmatrix}$$

Predicted ratings

$$\begin{bmatrix} (\theta^{(1)})^T x^{(1)} & (\theta^{(2)})^T x^{(1)} & \cdots & (\theta^{(n_u)})^T x^{(1)} \\ (\theta^{(1)})^T x^{(2)} & & & \\ \vdots & & & \\ (\theta^{(1)})^T x^{(n_m)} & \cdots & & (\theta^{(n_u)})^T x^{(n_m)} \end{bmatrix}$$

$$X = \begin{bmatrix} — (x^{(1)})^T — \\ — (x^{(2)})^T — \\ \vdots \\ — x^{(m)T} — \end{bmatrix} \qquad \Theta = \begin{bmatrix} — \theta^{(1)T} — \\ — \theta^{(2)T} — \\ \vdots \\ — \theta^{(n_u)T} — \\ — 0 — \end{bmatrix}$$

(10)

$$\begin{bmatrix} \theta^{(1)T} x^{(1)} & \cdots & \theta^{(n_u)T} x^{(1)} \\ \vdots & & \\ \theta^{(1)T} x^{(n_m)} & \cdots & \theta^{(n_u)T} x^{(n_m)} \end{bmatrix} = X\,\Theta^T$$

Low Rank Matrix factorization

SUMMARY OF COURSE

— Andrew Ng

Main topics

ML Tools

① Supervised Learning ($x^{(i)}, y^{(i)}$)
- Linear regression, logistic regression, neural Networks, SVMs

② Unsupervised Learning
- K-means, PCA, Anomaly detection.

③ Special applications / special topics
- Recommendation system, large scale machine learning.

(10) Advice on building a machine learning system
- Debugging learning. Bias/variance, regularization; deciding what to work on next; evaluation of learning algorithm. learning curve, error analysis, ceiling analysis.

ML Applying tools

M2

Course

Completed