Q) What is machine learning?

→ field of study that gives computers the ability to learn without being explicitly programmed.

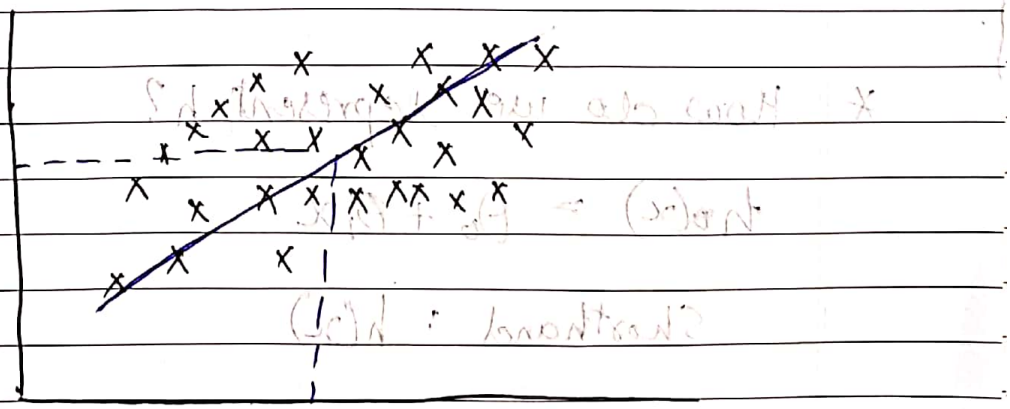
→ A computer program is said to learn from Experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with Experience E.

Machine learning Algorithms:

→ Supervised learning (features & labels)
→ Unsupervised learning. (only features)

Others: Reinforcement learning, Recommender System.

* Linear regression with one variable
→ model Representation.

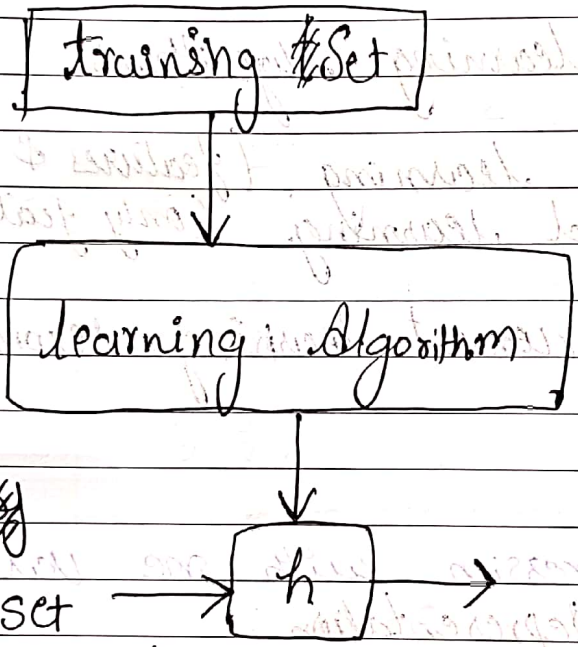


Notation:

m = Number of training Example
X's = "input" Variable/Feature
Y's = "Output" Variable / "target" Variable

(x, y) - one training Example.

$(x^{(i)}, y^{(i)})$ - i^{th} training Example



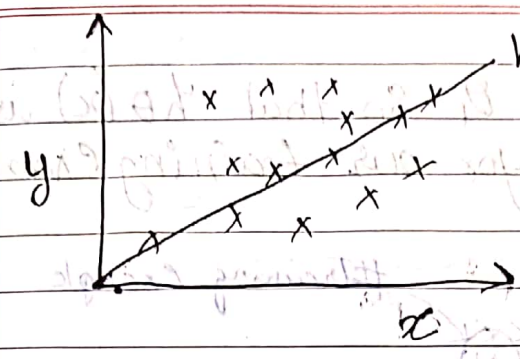
Regression → Continuous Values

Classification → Discrete Values

* How do we represent h?

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

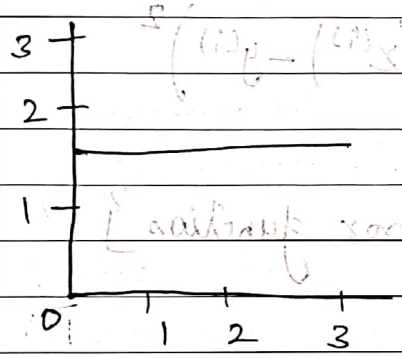
Shorthand : $h(x)$



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

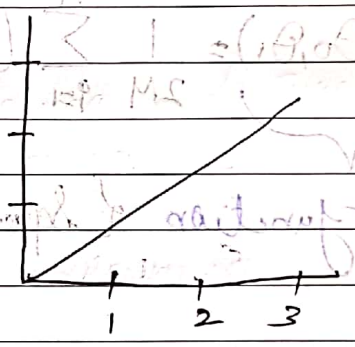
θ_i's : parameters.

#



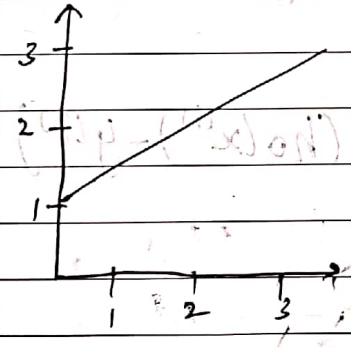
$\theta_0 = 1.5$

$\theta_1 = 0$



$\theta_0 = 0$

$\theta_1 = 0.5$



$\theta_0 = 1$

$\theta_1 = 0.5$

idea : choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training example (x, y)

training example

$$\text{Minimize}_{\theta_0, \theta_1} \quad \frac{1}{2M} \sum_{i=1}^{M} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

$h_\theta(x^{(i)})$: real output.

$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x^{(i)}$: estimated output

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

cost function { Square error function }

Cost function.

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

parameters : θ_0, θ_1

Cost function :

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

Goal : $\underset{\theta_0, \theta_1}{\text{minimum}} \; J(\theta_0, \theta_1)$

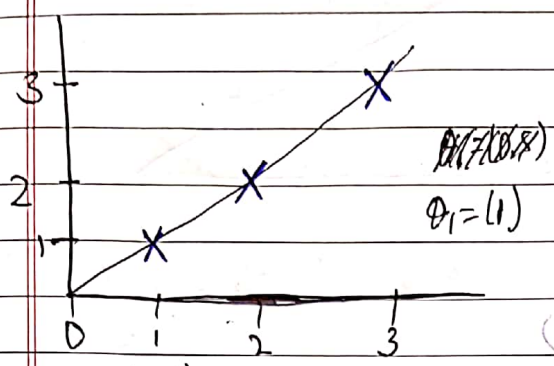
Simplified

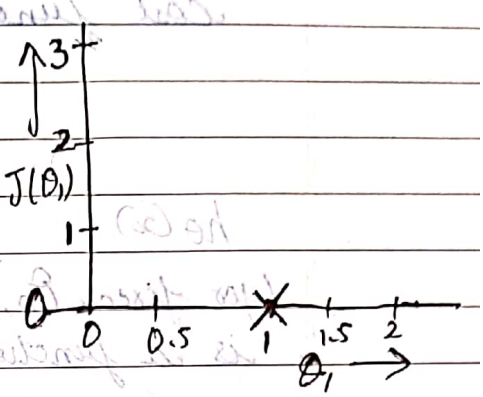
$h_\theta(x) = \theta_1 x \quad ; \quad \theta_0 = 0$

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

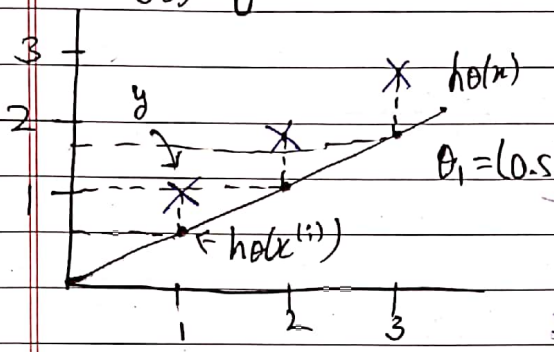
*enlivo
Date: __/__/____
Page: _____

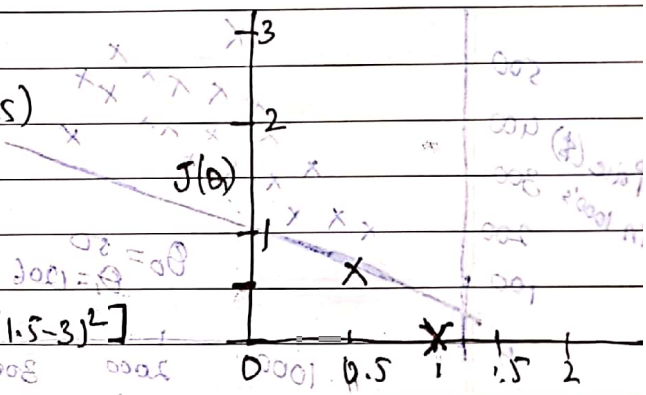
$h_\theta(x)$ $J(\theta_1)$





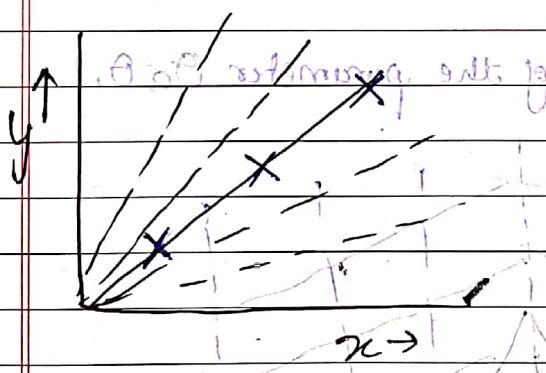
$J(1) = 0$

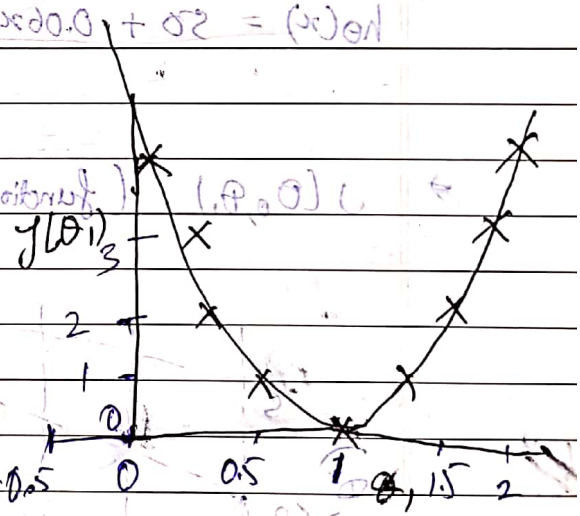




$$J(0.5) = \frac{1}{2M}\left[(0.5-1)^2+(1-2)^2+(1.5-3)^2\right]$$

$$= \frac{1}{2\times 3}(3.5) = 0.58.$$



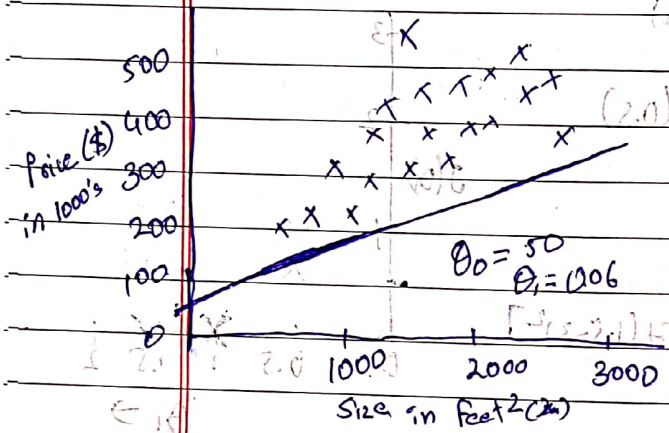


~~linear~~

⇒ Better the ^regression line minimum the cost function.

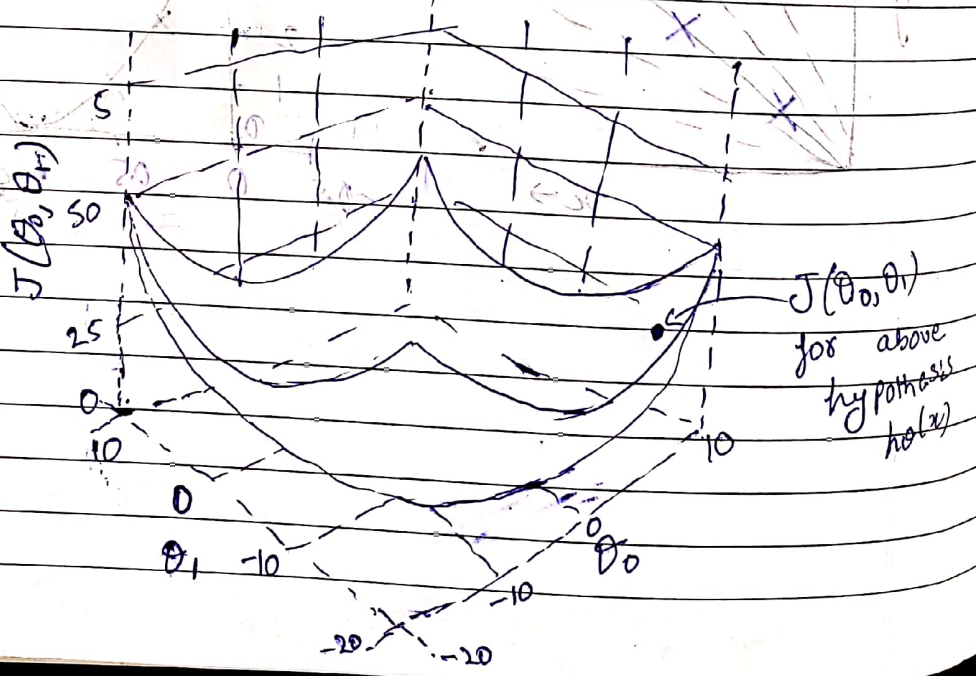
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



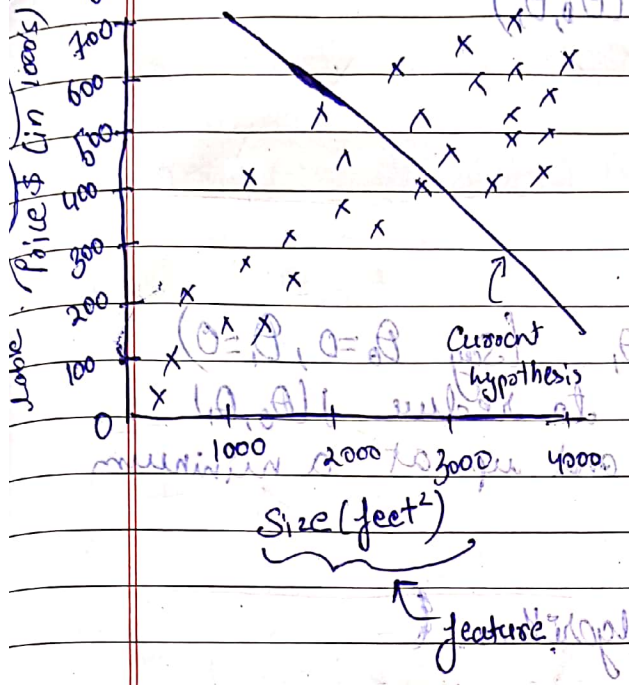
$h_\theta(x) = 50 + 0.06x.$

⇒ $J(\theta_0, \theta_1)$ (function of the parameter θ_0, θ_1)



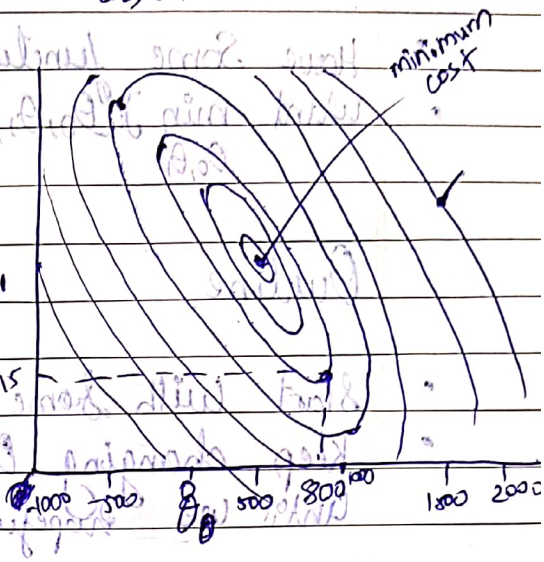
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x.)



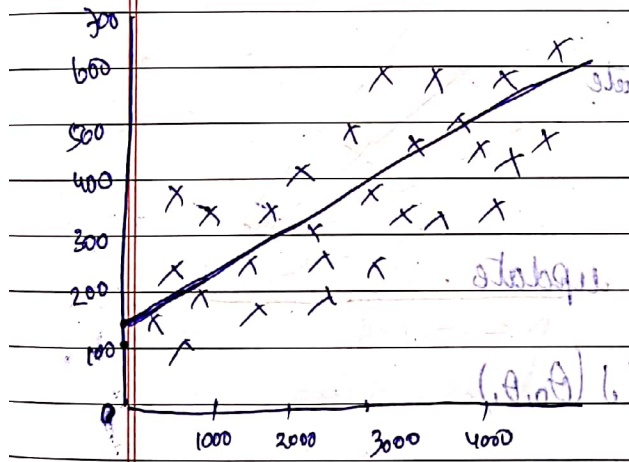
$J(\theta_0, \theta_1)$

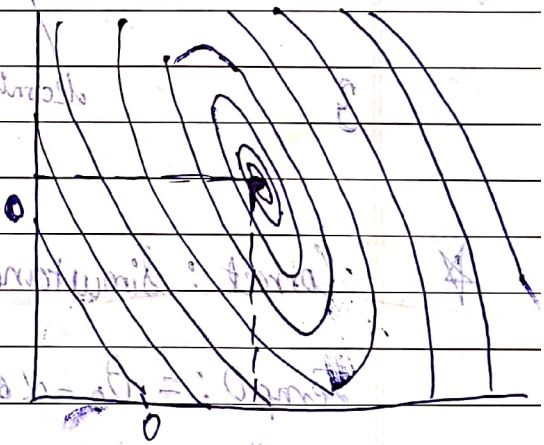
(function of the parameter θ_0, θ_1)



$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)





Gradient descent (with one variable)

- Have some function $J(\theta_0, \theta_1)$
- Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline

- Start with some θ_0, θ_1 (Say $\theta_0 = 0, \theta_1 = 0$)
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum

Gradient descent algorithm

Repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$

}

α : learning rate

Correct : Simultaneous update.

$$\text{temp}0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\text{temp}1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := \text{temp}0$$

$$\theta_1 := \text{temp}1$$

*enlivo
Date : ___/___/___
Page: ___

★ Incorrect :

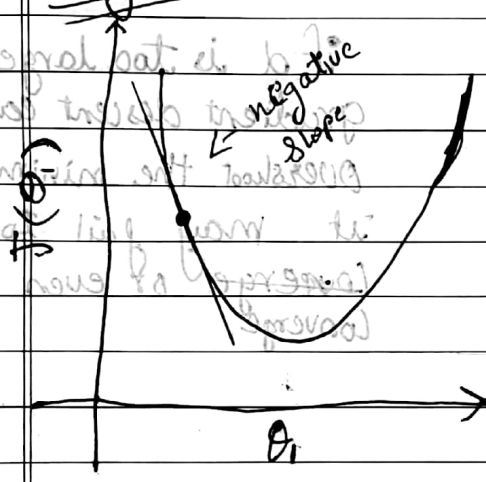
$temp0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

~~$temp0 = \theta_0$~~ $\theta_0 := temp0$

$temp1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

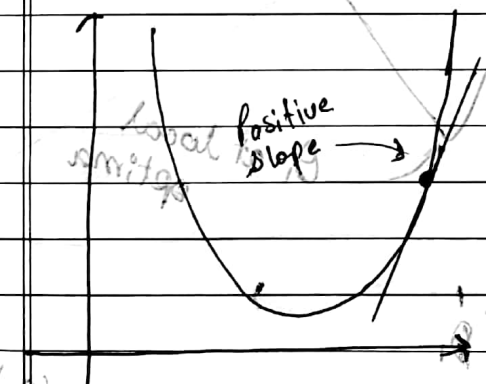
$\theta_1 := temp1$

★ Why minus (-) in $\left[\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)\right]$



$\frac{\partial}{\partial \theta_1} J(\theta_1) \leq 0$

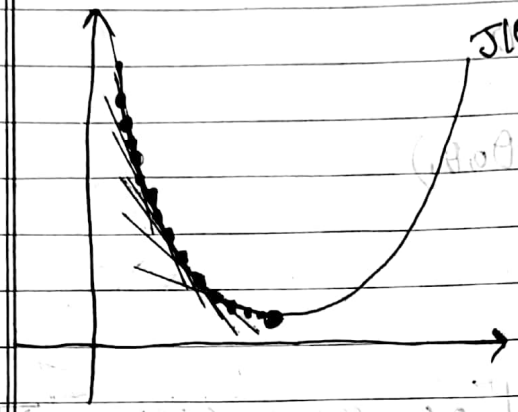
$\theta_1 := \theta_1 - \alpha$ (negative number)



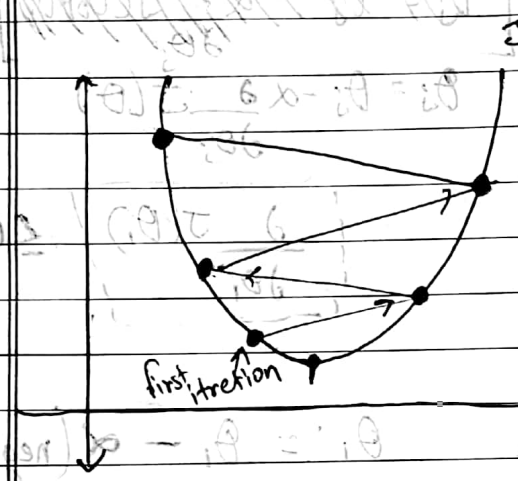
$\frac{\partial}{\partial \theta_1} J(\theta_1) \geq 0$

$\theta_1 := \theta_1 - \alpha$ (positive number)

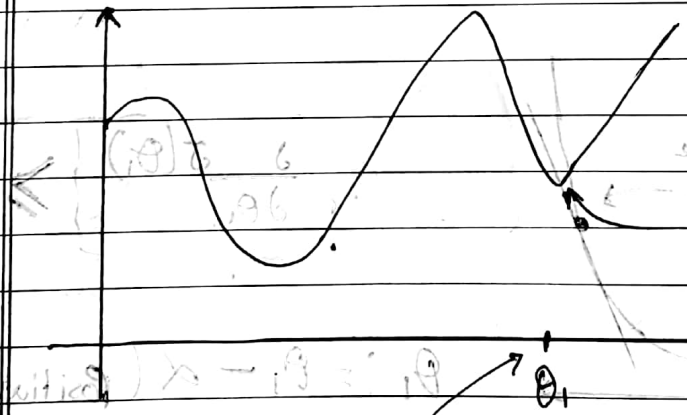
$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$



if α is too small, gradient descent can be slow.



if α is too large, gradient descent can overshoot the minimum, it may fail to ~~converge~~ converge or even diverge



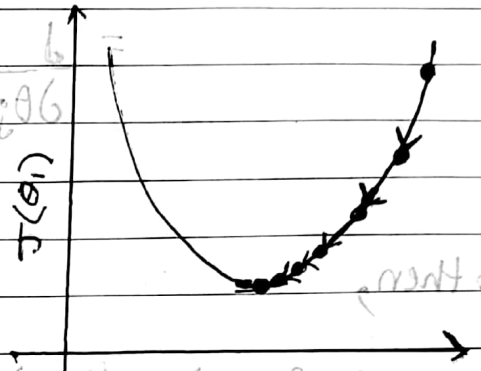
θ_1 at local optima

Current value of θ_1

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

★ Gradient descent can ~~be~~ converge to a local ~~minimum~~ minimum, even with the learning rate α fixed.

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



As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrese α over time.

bigger the slope $\left(\frac{\partial}{\partial \theta_j} J(\theta)\right)$ larger the step. $\left(\alpha \frac{\partial}{\partial \theta_j} J(\theta)\right)$

★ linear regression model (one varible)

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

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

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

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

then,

$$j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} 2 \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right) \cdot \frac{\partial}{\partial \theta_0} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)$$

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

$$= \frac{1}{m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right) \cdot (1 + 0 - 0)$$

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

$j = 1$ (feature)

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

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

$$= \frac{1}{m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right) \cdot \left(0 + x^{(i)} - 0 \right)$$

$$= \frac{1}{m}\sum_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)}) \cdot x^{(i)}$$

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

★ Repeat until convergence {

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

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

} update θ_0 and θ_1 simultaneously

★ "Batch"

→ Each step of gradient descent uses all the training examples.

MATRICES AND VECTORS.

Matrices : Rectangular arrays of numbers.

Dimension of matrix : number of rows × number of columns.

Example: $\begin{bmatrix} 1402 & 123 \\ 321 & 145 \\ 402 & 2759 \\ 102 & 279 \end{bmatrix}$ 4×2 matrix

$\mathbb{R}^{4\times 2}$

$$A = \begin{bmatrix} 1402 & 8102 \\ 2047 & 1112 \\ 9234 & 378 \\ 071 & 9279 \end{bmatrix}$$

A_{ij} = "i,j entry" in the i^{th} row, j^{th} column.

$A_{22} = 1112$, $A_{12} = 8102$, $A_{41} = 071$

★ Vector : An $n \times 1$ matrix

$$y = \begin{bmatrix} 460 \\ 320 \\ 123 \\ 145 \end{bmatrix} \in \mathbb{R}^4$$

← 4-Dimensional Vector.

$y_i = i^{th}$ element

$y_1 = 460$, $y_2 = 320$, $y_4 = 145$.

⇒ 1-indexed vs 0-indexed

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} \qquad y = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

⇒ We use A, B, C, X for matrices

and, a, b, x, y for vectors.

*enlivo
Date : ___/___/___
Page: ___

✱ matrix addition:

$$\begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 0.5 \\ 2 & 5 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 5 & 0.5 \\ 4 & 10 \\ 3 & 2 \end{bmatrix}$$

$3 \times 2 \qquad 3 \times 2 \qquad 3 \times 2$

$$\begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} + \begin{bmatrix} 4 & 0.5 \\ 2 & 5 \end{bmatrix} = \text{error} \quad [\text{dimensions are not same}]$$

$3 \times 2 \qquad 2 \times 2$

✱ Scalar Multiplication.

$$3 \times \begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 0 \\ 6 & 15 \\ 9 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 5 \\ 3 & 1 \end{bmatrix} \times 3$$

✱ Scalar division.

$$\begin{bmatrix} 4 & 0 \\ 6 & 3 \end{bmatrix} / 4 = \frac{1}{4} \begin{bmatrix} 4 & 0 \\ 6 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 3/2 & 3/4 \end{bmatrix}$$

✱ Combination of Operands

$$\begin{bmatrix} 1 \\ 4 \\ 2 \end{bmatrix} \times 3 + \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} - \begin{bmatrix} 3 \\ 0 \\ 2 \end{bmatrix} / 3$$

$3 \times 1 \qquad 3 \times 1 \qquad 3 \times 1$

$$= \begin{bmatrix} 3 \\ 12 \\ 6 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 5 \end{bmatrix} - \begin{bmatrix} 1 \\ 0 \\ 2/3 \end{bmatrix} = \begin{bmatrix} 2 \\ 12 \\ 10\ 1/3 \end{bmatrix}$$

3×1 matrix
3-dimensional vector

*enlivo
Date: / /
Page:

☆ Matrix-Vector multiplication

$$\begin{bmatrix} 1 & 3 \\ 4 & 0 \\ 2 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 5 \end{bmatrix} = \begin{bmatrix} (1\times1)+(3\times5) \\ (4\times1)+(0\times5) \\ (2\times1)+(1\times5) \end{bmatrix} = \begin{bmatrix} 16 \\ 4 \\ 7 \end{bmatrix}$$

(3×2) (2×1) → 3×1 matrix

Same, means possible.

dimension of resultant matrix

$$\underset{A}{\begin{bmatrix} \quad \\ \quad \\ \quad \end{bmatrix}} \times \underset{x}{\begin{bmatrix} \quad \\ \quad \end{bmatrix}} = \underset{y}{\begin{bmatrix} \quad \\ \quad \\ \quad \end{bmatrix}}$$

m×n matrix (m rows, n columns) | n×1 (n-dimensional) vector | m-dimensional Vector

To get y_i, multiply A's i^{th} row with elements of vector x, and add them up.

Example:

$$\begin{bmatrix} 1 & 2 & 1 & 5 \\ 0 & 3 & 0 & 4 \\ -1 & -2 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 14 \\ 13 \\ -7 \end{bmatrix}$$

3×4 4×1 3×1

$$\begin{bmatrix} 1\times1 + 2\times3 + 1\times2 + 5\times1 \\ 0\times1 + 3\times3 + 0\times2 + 4\times1 \\ -1\times1 + (-2)\times3 + 0\times2 + 0\times1 \end{bmatrix} = \begin{bmatrix} 14 \\ 13 \\ -7 \end{bmatrix}$$

House Sizes (x)

2104
1416
1534
852

$h_\theta(x) = -40 + 0.25x$

$\theta = \begin{bmatrix} -40 \\ 0.25 \end{bmatrix}$

for θ_0 for θ_1

$$x = \begin{bmatrix} 2104 \\ 1416 \\ 1534 \\ 852 \end{bmatrix} \quad \begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \times \begin{bmatrix} -40 \\ 0.25 \end{bmatrix} = \begin{bmatrix} (-40\times1)+(0.25\times2104) \\ \\ \\ \end{bmatrix}$$

4×2 (2×1) = 4×1

Prediction = Data matrix * parameters (θ)

☆ Matrix - Matrix multiplication

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 3 \\ 0 & 1 \\ 5 & 2 \end{bmatrix} = \begin{bmatrix} 11 & 10 \\ 9 & 14 \end{bmatrix}$$

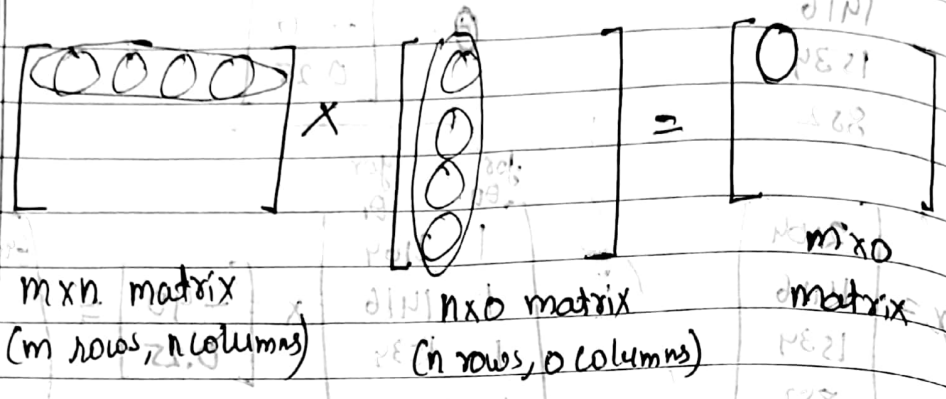
(2×3) (3×2) → 2×2

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix} = \begin{bmatrix} 11 \\ 9 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 10 \\ 14 \end{bmatrix}$$

Details:

A x B = C



m x n matrix (m rows, n columns)

n x o matrix (n rows, o columns)

m x o matrix

The i^{th} column of the matrix C is obtained by multiplying A with the i^{th} column of B. (for $i = 1, 2, \ldots o$)

★ House Sizes (x)

2104
1416
1534
852

Have 3 competing hypotheses:

1. $h_\theta(x) = -40 + 0.25x$
2. $h_\theta(x) = 200 + 0.1x$
3. $h_\theta(x) = -150 + 0.4x$

Matrix for θ_0, for θ_1

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \times \begin{bmatrix} -40 & 200 & -150 \\ 0.25 & 0.1 & 0.4 \end{bmatrix} = \begin{bmatrix} 486 & 410 & 692 \\ 314 & 342 & 416 \\ 344 & 353 & 464 \\ 173 & 285 & 191 \end{bmatrix}$$

Prediction of 1st h_θ | Prediction of 2nd h_θ | 3rd h_θ

Date: ___/___/_____
Page: _____

Matrix multiplication properties

★ Let A and B be matrices. Then in general
$A \times B \neq B \times A$ (not commutative)

E.g. $\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$

Or, $\begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \end{bmatrix}$

$A_{m \times n} \times B_{n \times m}$

$A \times B$ is $m \times m$

$B \times A$ is $n \times n$

★ Let A, B and C be matrices.
if $(A \times B) \times C = A \times (B \times C)$ (Associative)

★ Identity Matrix

Denoted I (or $I_{n \times n}$)

Example:

$\begin{bmatrix} 1 \end{bmatrix}$ 1×1

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ 2×2

$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ 3×3

$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ 4×4

for any matrix A,

$A \cdot I = I \cdot A = A$

$m \times n \quad n \times n \quad m \times m \quad m \times n \quad m \times n$

note:

$A \times B \neq B \times A$

$AI = IA$

*enlivo
Date: __/__/__
Page:

☆ Inverse and transpose

Matrix Inverse

→ Not all numbers have an inverse.

→ if A is an m×m matrix, and if it has an inverse,

→ $A(A^{-1}) = (A^{-1})A = I$

→ $A = \mathbb{R}^{m \times m}$ (Square matrix) because I is square matrix.

→ matrices that don't have an inverse are "Singular" or "degenerate".

Matrix Transpose

Example:

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 5 & 9 \end{bmatrix} \qquad A^T = A' = \begin{bmatrix} 1 & 3 \\ 2 & 5 \\ 0 & 9 \end{bmatrix}$$

Let A be an m×n matrix, and let $B = A^T$. Then B is an n×m matrix, and

$$B_{ij} = A_{ji}$$

$\Rightarrow B_{12} = A_{21} = 2$

$B_{32} = 9 \qquad A_{23} = 9.$

☆ LINEAR REGRESSION WITH MULTIPLE VARIABLES (Features)

Size (feet^2) (x_1)	No. of bedrooms (x_2)	No. of floors (x_3)	Age of home (years) (x_4)	price ($1000) ($y$)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

m

Notation:

m = number of training Examples.

n = number of features.

$x^{(i)}$ = input (features) of i^{th} training Example.

$x_j^{(i)}$ = value of feature j in i^{th} training example.

Hypothesis:

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

For convenience of notation, define $x_0^{(i)} = 1$

*enlivo
Date: __/__/__
Page:

For convenience of notation, define $x_0^{(i)} = 1$

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$$

then, $\theta^T = [\theta_0, \theta_1, \theta_2 \ldots \theta_n] * X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

$$\boxed{h_\theta(x) = \theta^T X} \leftarrow \text{Hypothesis.}$$

☆ Gradient descent for multiple Variables

Cost function:

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

Gradient descent:

Repeat {

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

} (simultaneously update for every $j = 0, \ldots n$)

*enlivo

Date: ___/___/___

Page: ___

Repeat {

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

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

(Simultaneously update θ_j for $j = 0, \ldots n$)

}

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

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

$$\theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_2^{(i)}$$

$$\vdots$$

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

---

for easy understanding.

$$(h_\theta(x^{(i)}) - y^{(i)}) = \begin{bmatrix} \\ \\ \\ \\ \\ \end{bmatrix}_{m}$$

$$x^{(i)} = \begin{matrix} x_0 & x_1 & x_2 & x_3 \end{matrix} \begin{bmatrix} x^1 \\ x^2 \\ x^3 \\ x^4 \\ x^5 \end{bmatrix}_{m} \quad m \times (n+1)$$

$$\theta = \begin{bmatrix} \\ \\ \\ \end{bmatrix}_{(n+1)} \quad (n+1) \times 1$$

$m \times 1$

$(h_\theta(x^{(i)}) - y^{(i)})$	$x^{(i)}$	θ
$m \times 1$	$m \times (n+1)$	$(n+1) \times 1$

⇒ Vectorized Gradient descent.

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

θ_j → $(n+1) \times 1$

$$\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)} \rightarrow (x_j^{(i)})^T * (h_\theta(x^{(i)}) - y^{(i)}) \rightarrow (n+1) \times 1$$

$$\therefore \Rightarrow \theta = \theta - \alpha \frac{1}{m} \left[(x^{(i)})^T * (h_\theta(x^{(i)}) - y^{(i)}) \right]$$

★ Feature Scaling.

→ Get every feature into approximately a $-1 \leq x_i \leq 1$ range.

→ Examples for Ranges:

maximum range ↳ -3 to 3 ✓

minimum range ↳ $-1/3$ to $1/3$ ✓

*enlivo
Date : ___/___/_____
Page : _____

☆

→ idea : Make Sure feature are on a similar Scale.

☆ Mean Normalization (example of feature scaling)

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean

(Do not apply to $x_0 = 1$)

where μ_i = average value of x_i feature

Now,

$$x_i = \frac{x_i - \mu_i}{S_i}, \text{ where } S_i = \text{max} - \text{min}$$

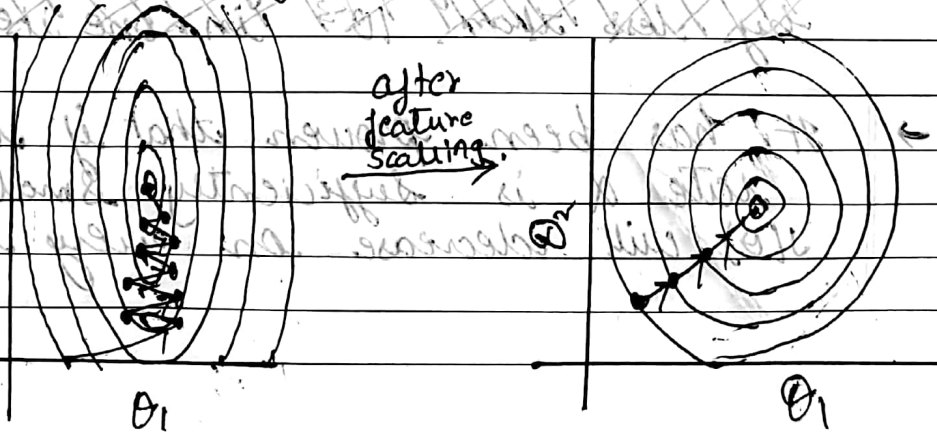
S_i → helps to make range small.
e.g. $-3 \le x < 3$, $-1 \le x \le 1$, etc.

max → maximum value of feature x_i

min → minimum value of feature x_i

☆ Feature Scaling on two feature data.

→ (Ex. How cost function J on gradient descent look like)



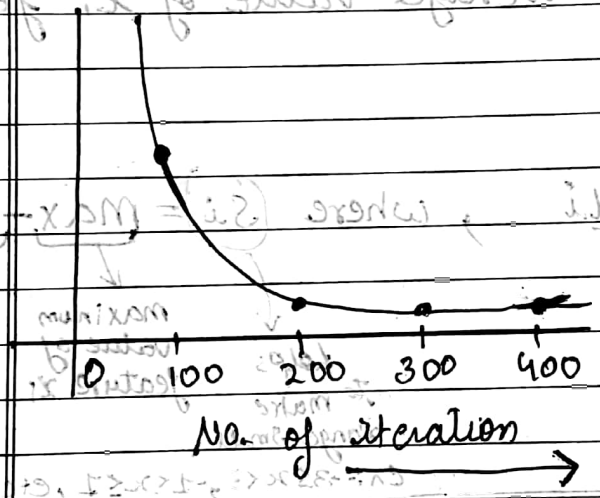
→ Debugging in Gradient descent :

↳ How to make sure gradient descent is working correctly.

↳ How to choose learning rate α.

Making sure gradient descent is working correctly

→ min $J(\theta)$

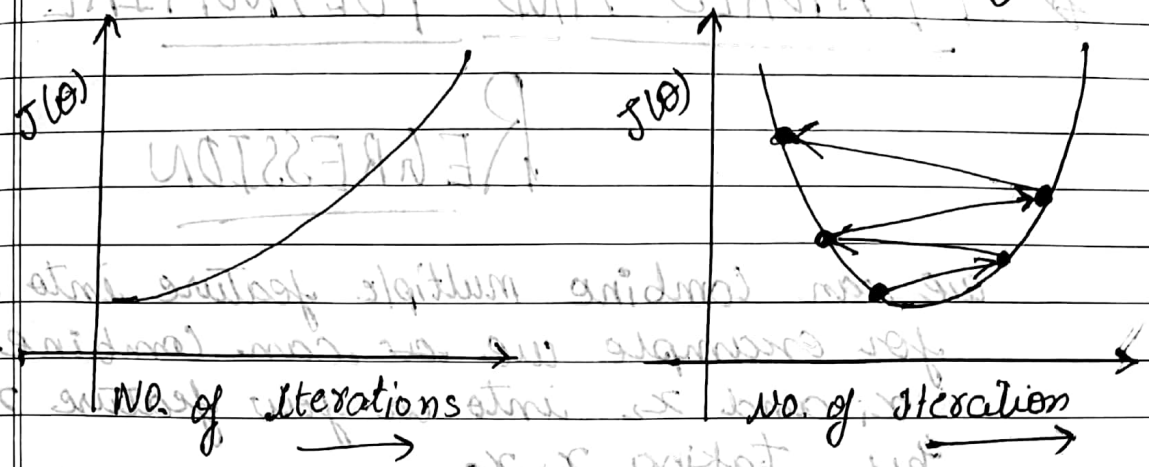


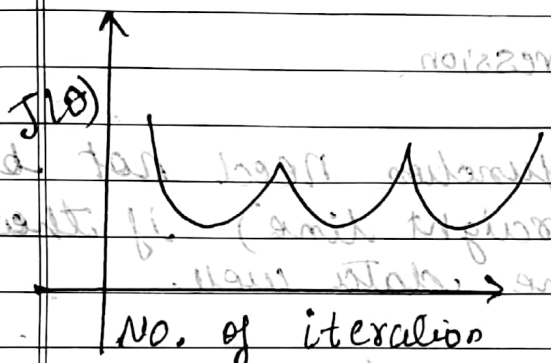
– $J(\theta)$ should decrease after every iteration

→ ~~Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration.~~

→ It has been proven that if learning rate α is sufficiently small, then $J(\theta)$ will decrease on every iteration

if Gradient descent ~~is~~ not working. i.e.





→ then use smaller α.

→ But if α is too small, gradient descent can be slow to converge.

Summary:

- if α is too small: slow convergence.
- if α is too large: J(θ) may not decrease on every iteration; may not converge

To choose α, try

..., 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1, ...

FEATURES AND POLYNOMIAL REGRESSION

we can **combine** multiple feature into one. for example we can combine x_1 and x_2 into a new feature x_3 by taking $x_1 \cdot x_2$

→ Polynominal Regression.

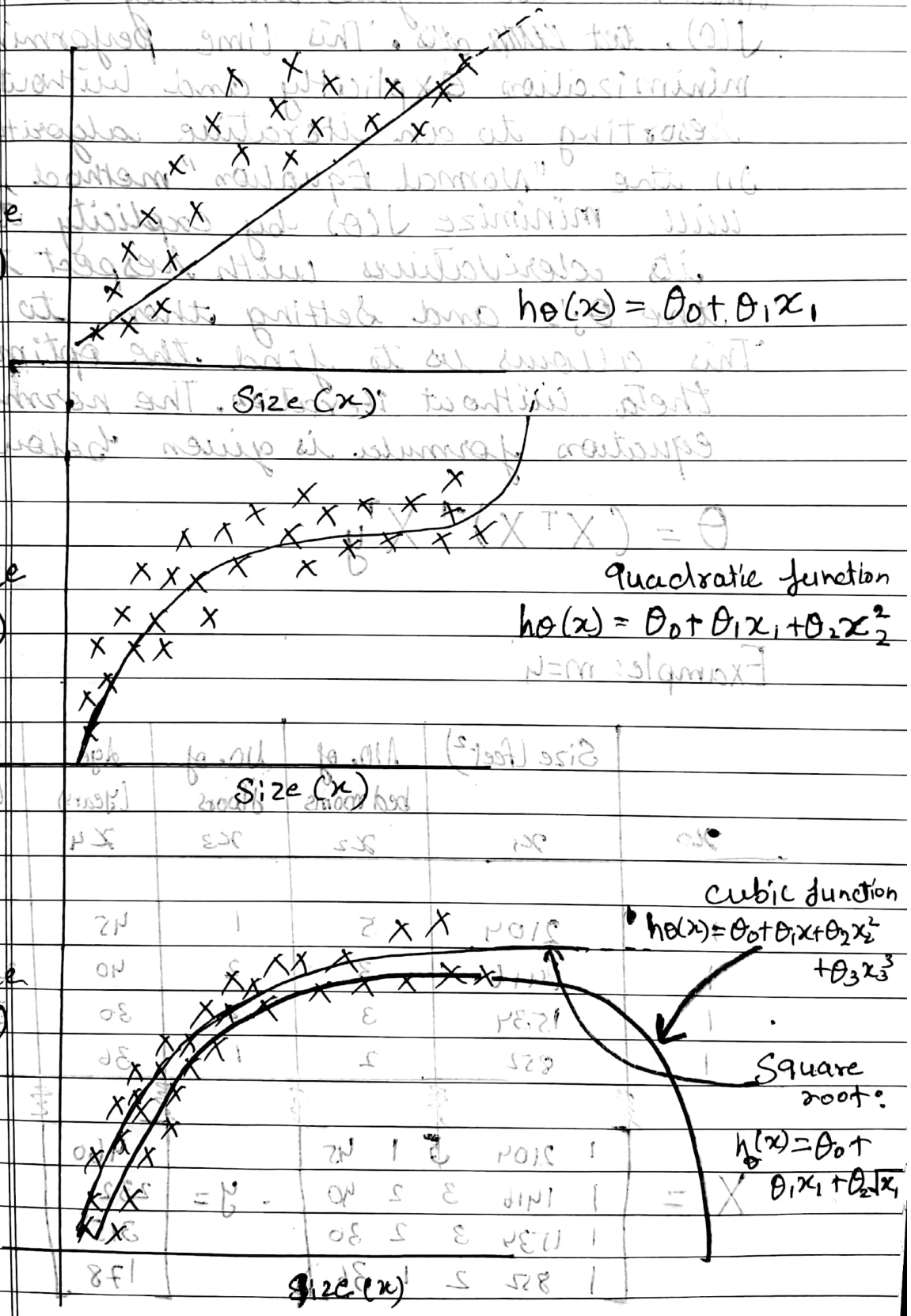
our hypothesis function need not be linear (a straight line) if that does not fit the data well.

we can **change the behaviour or curve** of our hypothesis function by making it a quadratic cubic or square root function (or any other form)

for example, if our hypothesis function is $h_\theta(x) = \theta_0 + \theta_1 x_1$ then we can create additional features based on x_1, to get the quadratic function $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2$ or the cubic function $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^3$

To make it a Square root function, we could do: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 \sqrt{x_1}$

One important thing to keep in mind is, if your feature this way then feature scaling becomes very important.



☆ NORMAL EQUATION

Gradient descent gives one way to minimizing $J(\theta)$. This time performing the minimization explicitly and without resorting to an iterative algorithm. In the "Normal Equation" method, we will minimize $J(\theta)$ by explicity taking its derivatives with respect to the θ_j's and setting them to zero This allows us to find the optimum theta without iteration. The normal equation formula is given below.

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

Example: m=4

	Size (feet^2)	No. of bedrooms	No. of floors	Age (years)	Price ($1000)
x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \quad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

There is **no need** to do feature scaling with the normal equation.

Gradient descent (GD) V.S Normal Equation

Gradient descent (GD)	Normal Equation
→ Need to choose alpha (α)	→ No Need to choose alpha (α)
→ Needs many iterations	→ No Need to iterate
→ $O(kn^2)$	→ $O(n^3)$, need to calculate inverse of X^TX
→ Work well when n is large	→ Slow if n is very large

Tip: if ~~n~~ $n \geq 10{,}000$? then GD : else Normal Equation

★ What if X^TX is non-invertible?

- Redundant feature (linear dependent)

E.g. x_1 = size in feet^2, x_2 = size in m^2 } ~~does not~~ 1m = 3.28 feet; does not required both features. One can do Job.

- Too many features (e.g. $m \leq n$)

→ Delete some feature, or use Regularization.

Octave: pinv(X'*X)*(X'*y)

*** Linear Regression : Continuous Value.
Logistic Regression : discrete Value

LOGISTIC REGRESSION

CLASSIFICATION

To attempt classification, one method is to use linear regression and map all prediction greater than 0.5 as a 1 and all less than 0.5 as a 0. However this method does not work well because classification is not actually a linear function.

classification.

Email : Spam / Not Spam
Online Transactions : fraudulent (Yes / No)?
Tumar : Malignant / Benign?

$y \in \{0, 1\}$
- 0 : "Negative class" (eg; benign tumar)
- 1 : "Positive class" (eg., malignant tumar)

*** y could have more than 2 class
i.e. $y = \{0, 1, 2\}$ or $y = \{0, 1, 2, 3\}$ etc.

where $y \in \{0, 1\}$ called binary classification.

Hypothesis Representation

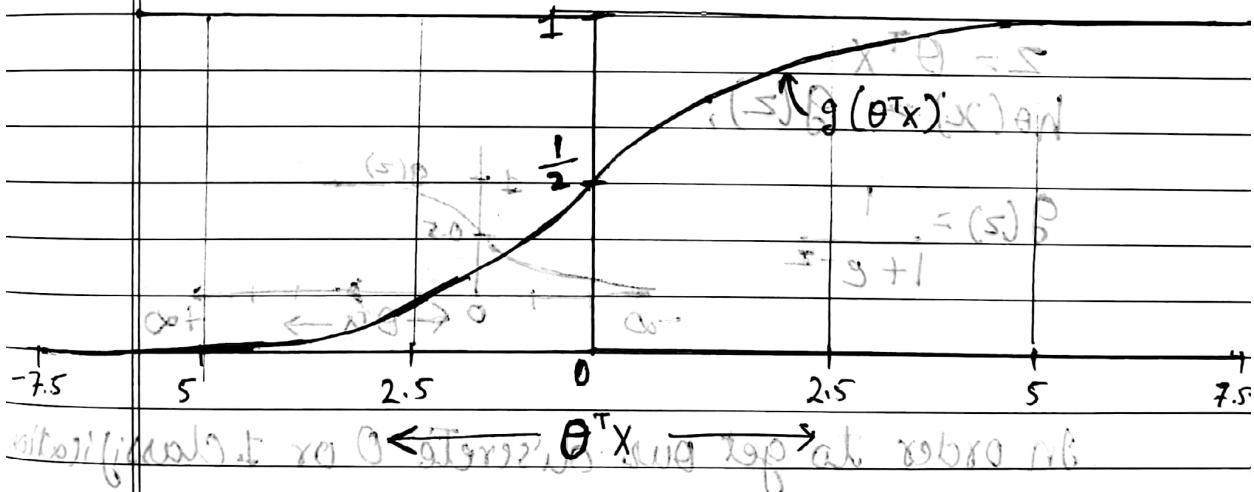
We could approach the classification problem ignoring the fact that y is discrete-valued, and our old linear regression algo. to try to predict y given x. However, it is easy to construct example where this method peforms very poorly. Intuitively, it also doesn't make sense for $h_\theta(x)$ to take value larger than 1 or smaller then 0 i.e $h_\theta(x) > 1$ or < 0 when we know that $y \in \{1, 0\}$. To fix this, our new hypotheses $h_\theta(x)$ to satisfy $0 \leq h_\theta(x) \leq 1$

→ This can be done by plugging $\theta^T x$ into the "logistic function" or "sigmoid function".

$$h_\theta(x) = g(\theta^T x) \quad \text{— hypothesis}$$

$$z = \theta^T x$$

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



The function $g(\theta^T x)$ or $g(z)$, $z = \theta^T x$, shows. maps any real number to the (0, 1) interval

★ Example to interpret Hypothesis output

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

if $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$h_\theta(x) = 0.7$

Tell patient that 70% change of tumor being malignant.

i.e if probability that it is 1 is 70%, then the probability that it is 0 is 30%

$P(y=0) + P(y=1) = 1$

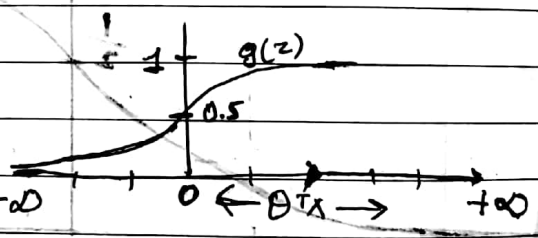
or $P(y=0) = 1 - P(y=1)$

★ DECISION BOUNDARY

$z = \theta^T x$

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

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



In order to get our discrete 0 or 1 classification

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

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

if $z \geq 0$
then $g(z) \geq 0.5$: when Input is greater than or equal to 0. Its output is greater than or equal to 0.5

Remember

$z = 0 \; ; \; e^0 = 1 \Rightarrow g(z) = 1/2$

$z = \infty \; ; \; e^{-\infty} = 0 \Rightarrow g(z) = 1$

$z = -\infty \; ; \; e^{-(-\infty)} = \infty \Rightarrow g(z) = 0$

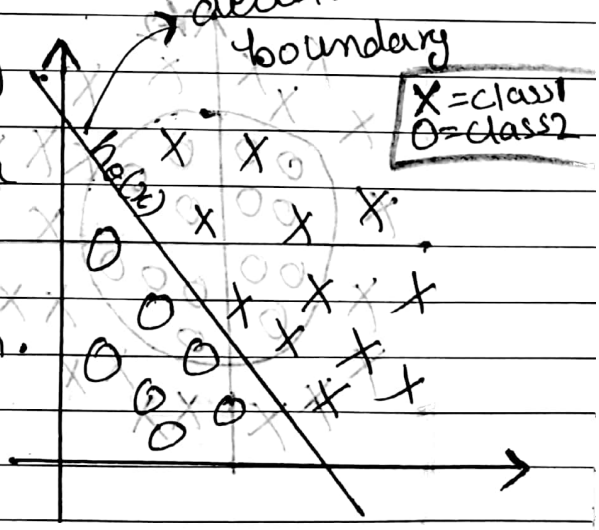
So, if our input to g is $\theta^T X$,

$h_\theta(x) = g(\theta^T X) \geq 0.5$
when $\theta^T X \geq 0$

from these Statment we can now say;

$\theta^T X \geq 0 \Rightarrow y = 1$
$\theta^T X < 0 \Rightarrow y = 0$

→ The decision Boundary is the line that separates the area where $y=0$ and where $y=1$, it is created by our hypothesis function.



Example:

let, $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$

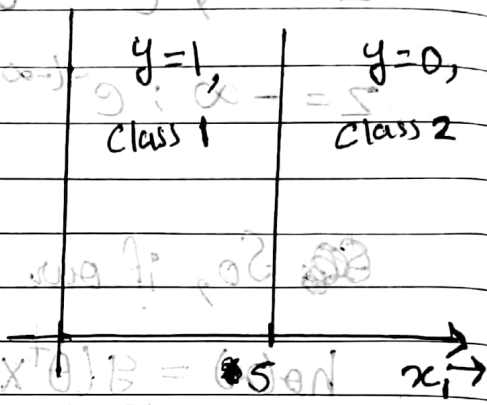
For $y=1$ i.e $\theta^T x \geq 0$

$\therefore \quad 5 + [(-1)x_1 + (0)x_2] \geq 0$

$5 - x_1 \geq 0$

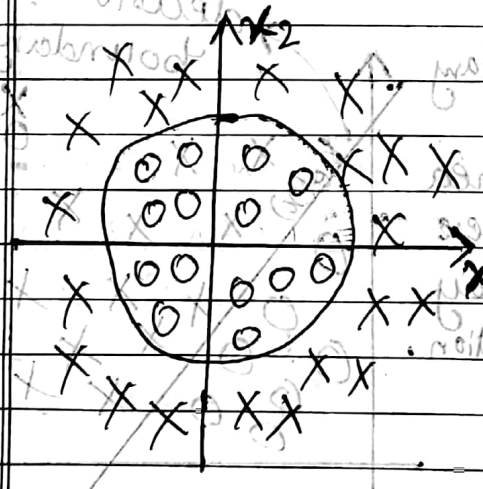
$-x_1 \geq -5$

$x_1 \leq 5$



☆ Again, the input to the Sigmoid function $g(z)$ (eg: $\theta^T x$) does not need to be linear, and could be function that describes a circle (e.g. $z = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2$) or any shape to fit data

Example: (Non-linear decision boundaries)



$z = \theta_0 + \theta_1 x_1^2 + \theta_2 x_2^2$

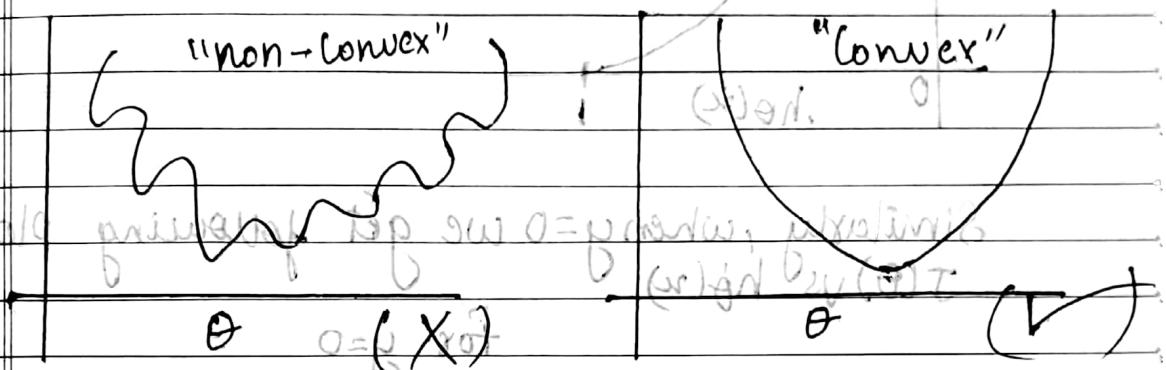
where $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \\ 1 \end{bmatrix}$

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

$\Rightarrow x_1^2 + x_2^2 \geq 1$

☆ Cost FUNCTION

we cant use the same function that we use for linear regression because the logistic function will cause the output to be wavy, causing many local optima, it will not be convex function.



Logistic regression cost function.

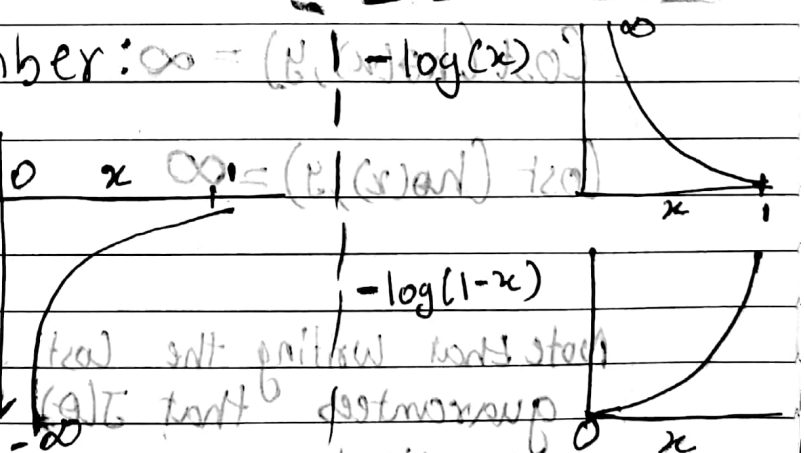
$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}\left(h_\theta(x^{(i)}), y^{(i)}\right)$$

$$\text{Cost}\left(h_\theta(x^{(i)}), y^{(i)}\right) = -\log\left(h_\theta(x)\right) \quad \text{if } y=1$$

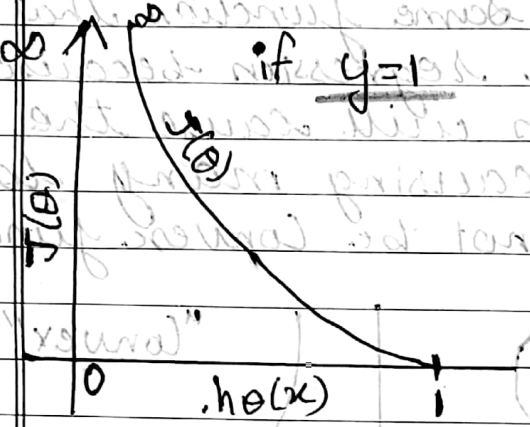
$$\text{Cost}\left(h_\theta(x^{(i)}), y^{(i)}\right) = -\log\left(1-h_\theta(x)\right) \quad \text{if } y=0$$

*** Remember: $-\log(x)$

$\log(x)$

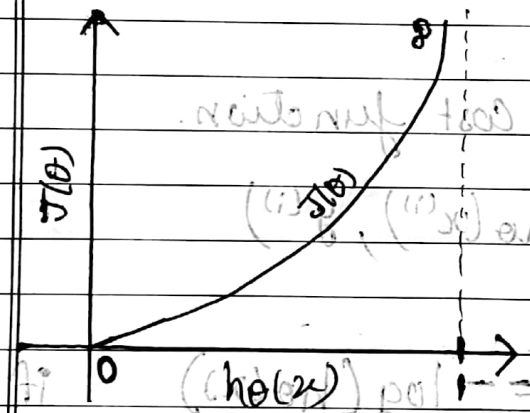


When $y=1$ we get the following plot for $J(\theta)$ vs $h_\theta(x)$



Similarly, when $y=0$ we get following plot for $J(\theta)$ vs $h_\theta(x)$

for $y=0$



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

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

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

Note that writing the cost function in this way guarantees that $J(\theta)$ is convex for logistic regression.

✱ Simplified Cost function.

We can compress our cost functions two conditional cases into one case.

Logistic Regression cost function.

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

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

Note: $y = 0$ or 1 always.

$$\therefore Cost\left(h_\theta(x), y\right) = -y \log\left(h_\theta(x)\right) - (1-y) \log\left(1-h_\theta(x)\right)$$

Notice that,

if $y=1$:

$$Cost\left(h_\theta(x), 1\right) = -(1) \log\left(h_\theta(x)\right) - (1-1) \log\left(1-h_\theta(x)\right)$$
$$Cost\left(h_\theta(x), y\right) = -\log\left(h_\theta(x)\right)$$

if $y=0$:

$$Cost\left(h_\theta(x), 0\right) = -(0) \log\left(h_\theta(x)\right) - (1-0) \log\left(1-h_\theta(x)\right)$$
$$= -\log\left(1-h_\theta(x)\right)$$

therefore,

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

☆ Gradient Descent

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

Want $\min_\theta J(\theta)$:

Algorithm:

Repeat {

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

} (Simultaneously update all θ_j).

here,

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

~~Notice / Hint / this~~

∴ Repeat {

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

}

Notice that this algorithm is identical to the one we used in linear Regression.

algorithm is identical, but defination is different (partial derivation of linear regression cost function and partial derivation of logistic regression cost function is totally different).

★ Vectorized implemention of:

1) Cost function.

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

2) Gradient descent

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

★ ADVANCED OPTIMIZATION

optimization algorithms.

- → Gradient descent
- → Conjugate gradient
- → BFGS
- → L-BFGS

→ are more sophisticated, faster ways to optimize θ that can be used instead of gradient descent.

→ But are more complex

This page shows how to use in-build Advanced optimization function on octave. ("fminunc()")

*enlivo
Date: __/__/__
Page:

Octave provides highly optimized library. ("fminunc()")

We first need to provide a function the evaluates the following two function for a given input value θ.

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

We can write a single functions that return both of these:

```
function [jVal, gradient] = CostFunction (theta)

    jVal = [... code to compute J(θ)];
    gradient = [... code to compute ∂J(θ)/∂θj ]

end
```

Now, we can use "fminunc()" optimization algo. along with "optimset()" function that creates an object containing the options we want to send to "fminunc()".

```
options = optimset('GradObj', 'on', 'MaxIter', 100);
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] = ...
    fminunc(@costFunction, initialTheta, options);
```

We give our cost function, our inital vector of Theta and "options" object to "fminunc()".

☆ MULTI-CLASS CLASSIFICATION:

ONE-VS-ALL

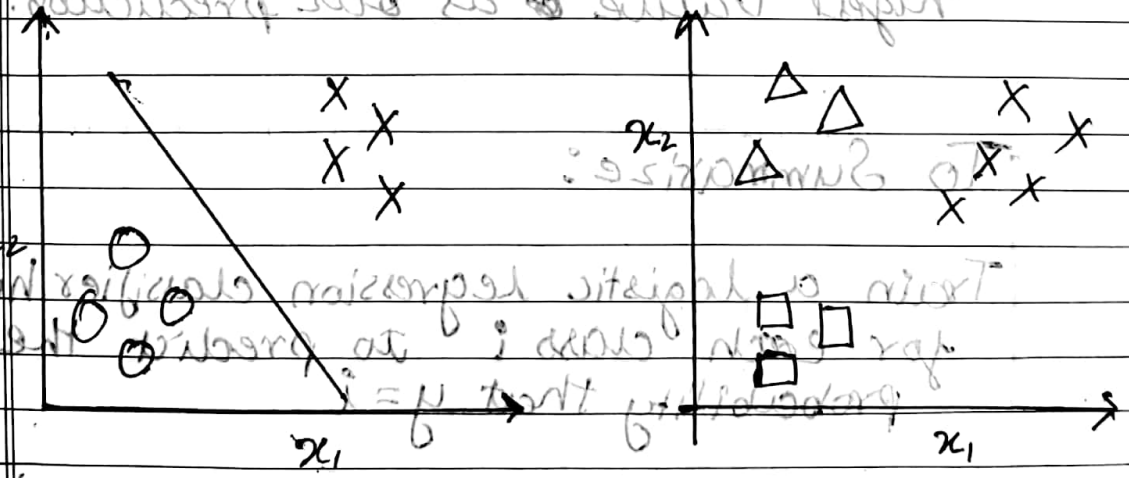
☆ multiclass classification.

→ Email foldering / tagging : work / Friends / family / Hobby.
$y \in \{0, 1, 2, 3\}$.

→ Medical diagram : Not ill, Cold, Flu
$y \in \{0, 1, 2\}$.

→ Weather : Sunny, Cloudy, Rain, Snow
$y \in \{0, 1, 2, 3\}$.

Binary classification. Multi-class Classification



Let $y \in \{0, 1, \ldots, n\}$

$h_\theta^{(0)}(x) = P(y=0 \mid x; \theta)$

$h_\theta^{(1)}(x) = P(y=1 \mid x; \theta)$

$h_\theta^{(2)}(x) = P(y=2 \mid x; \theta)$

$\vdots$

$h_\theta^{(n)}(x) = P(y=n \mid x; \theta)$

$\text{prediction} = \max_i \left(h_\theta^{(i)}(x) \right)$

We are basically choosing one class and then lumping all the others into single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returns the highest value as our prediction.

To Summarize:

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

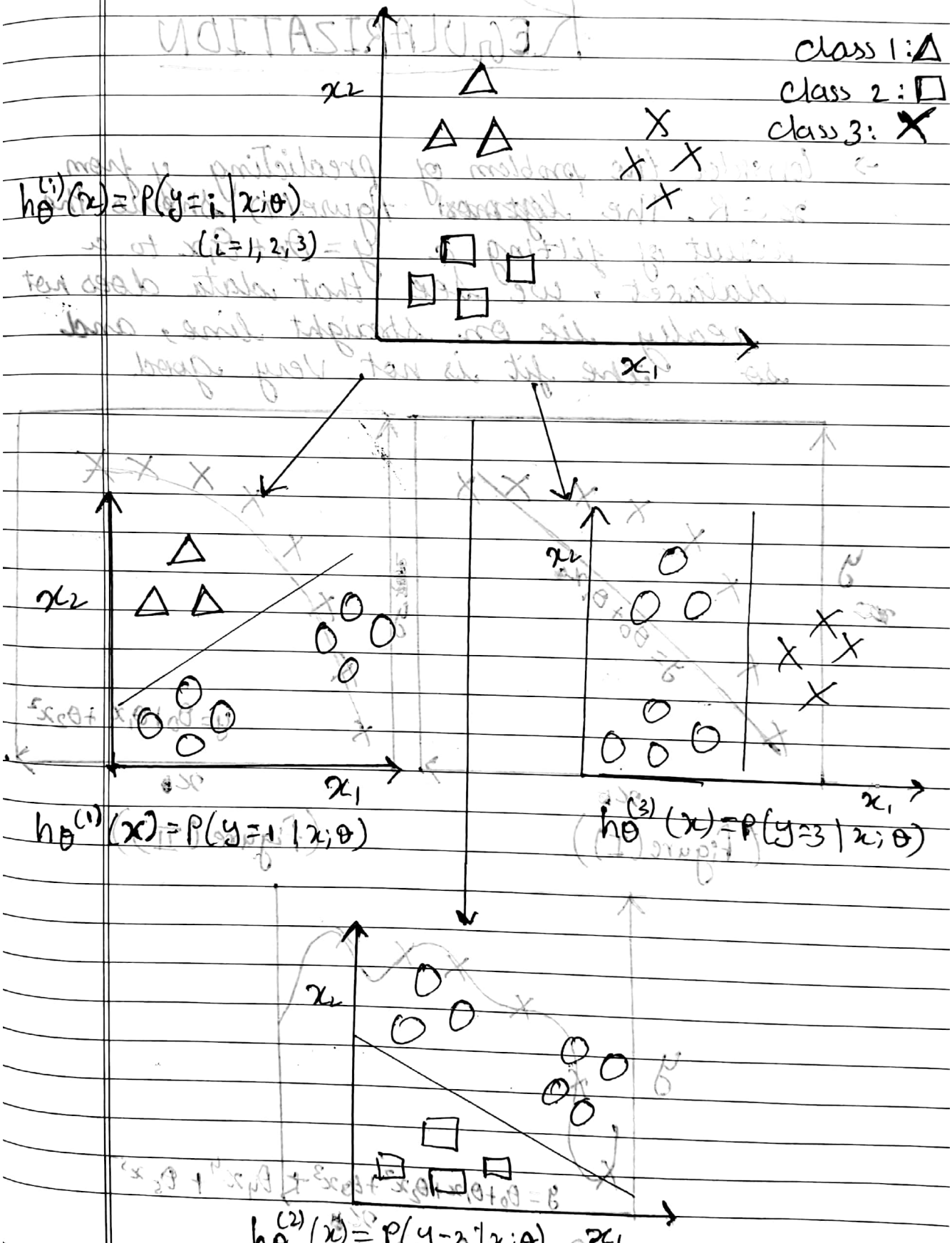
On a new input x to make a prediction, pick the class i that maximizes

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

*enlivo
Date : ____/____/____
Page : _____

Example:

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

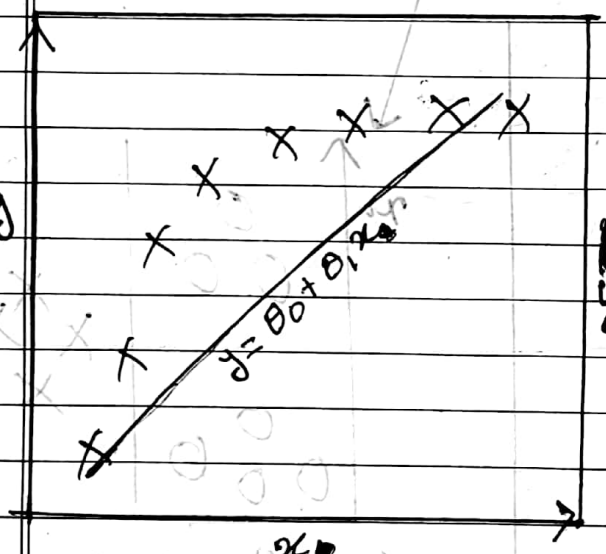


*enlivo
Date: __/__/____
Page: _____

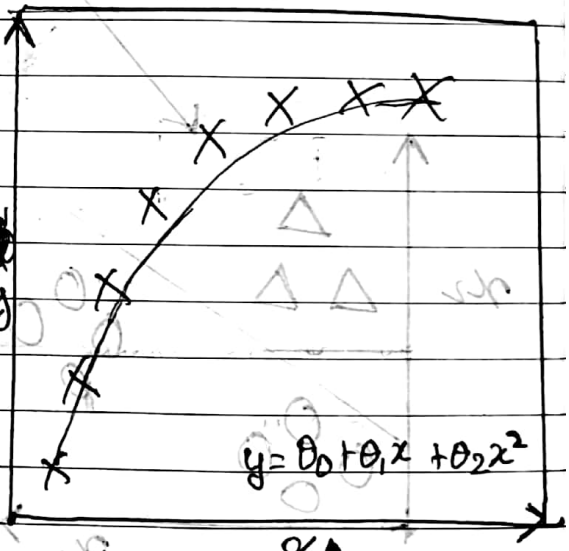
✡ THE PROBLEM OF OVERFITTING:

REGULARIZATION

→ Consider the problem of predicting y from $x \in R$. The figure (I) shows the result of fitting a $y = \theta_0 + \theta_1 x$ to a dataset. we see that data does not really lie on straight line, and so the fit is not very good.



(figure (I))



(figure (II))

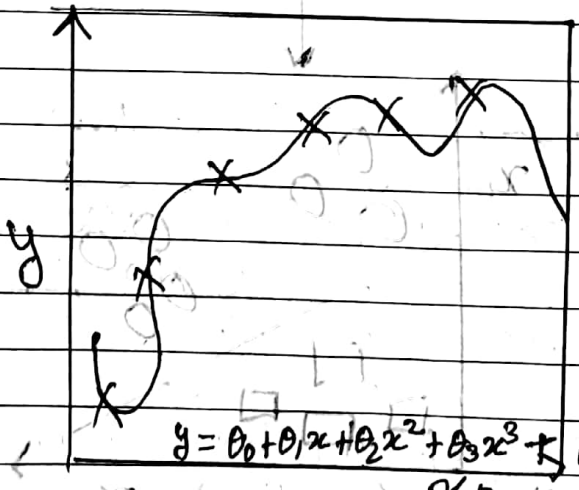


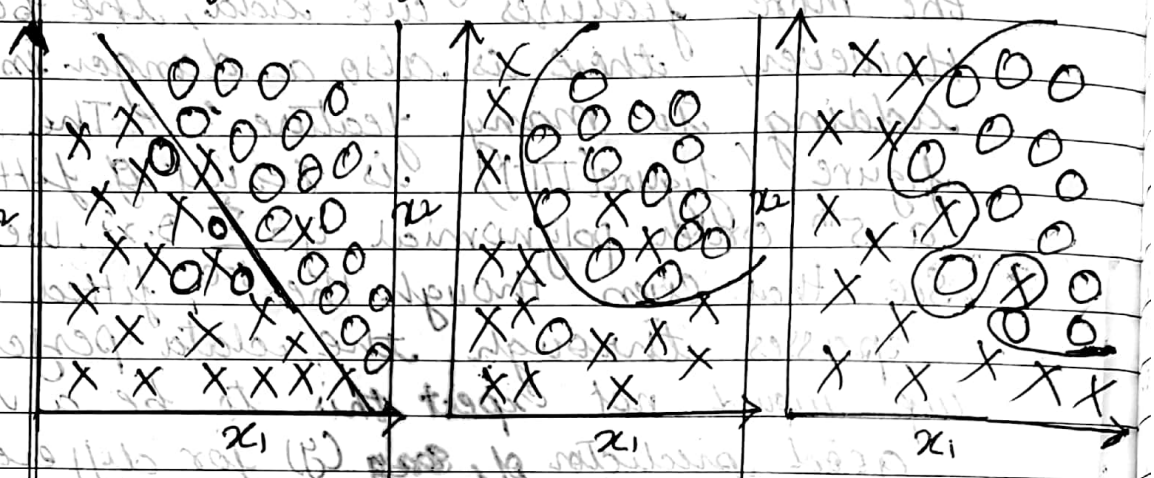
figure (III)

instead, if we had added an extra feature x^2, and fit $y = \theta_0 + \theta_1 x + \theta_2 x^2$, then we obtain a slightly better fit to the data (figure (II)). Naively, it might seem that the more features we add, the better. However, there is also a danger in adding too many features: The figure (figure (III)) is result of fitting a 5th order polynomial $y = \sum_{j=0}^{5} \theta_j x^j$. We see that even though the fitted curve passes through the data perfectly we would not expect this to be a very good predictor of, say (y) for different (x). Without formally defining what these terms mean, we'll say the figure (I) shows an instance of **underfitting** — in which the data clearly shows structure not captured by the model — and the figure (III) is an example of **Overfitting**

Underfitting, or high bias, is when the form of our hypothesis function h maps poorly to the trend of the data. It is usually caused by a function that is too simple or uses too few features. **Overfitting or high variance**, is caused by a hypothesis function that fits the available data but does not generalize well to predict new data. It is usually caused by a complicated function that creates a lot of unnecessary curves and angles unrelated to the data.

This terminology is applied to both linear and logistic regression.

→ Logistic regression:



→ $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$	$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$	$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$
→ Underfit	→ Just right	→ Overfit.

☆ Addressing Overfitting.

1. Reduce number of features.
   - manually select which feature to keep.
   - Model Selection algorithm
2. Regularization.
   - Keep all the feature, but reduce magnitude/values of parameters θ_j.
   - Works well when we have a lot of features, each of which contributes a bit to predicting y.

Cost function

if we have overfitting from our hypothesis function, we can reduce the weight that some of the terms in our function carry by increasing their cost.

Say we wanted to make the following function more quadratic.

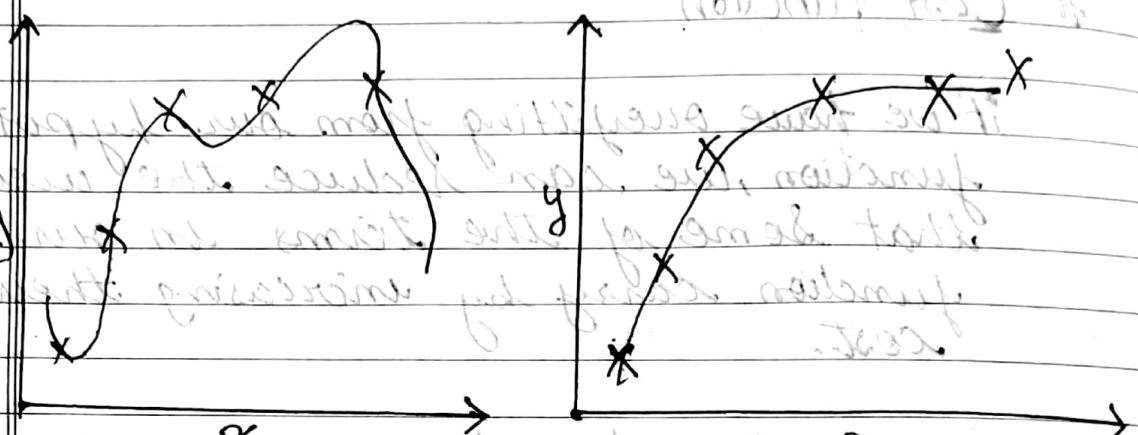
$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

we'll want to eliminate the influence of $\theta_3 x^3$ and $\theta_4 x^4$. Without actually getting rid of these features or changing the form of our hypothesis, we can instead modify our cost function.

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

We've added two extra terms at the end to inflate the cost of θ_3 and θ_4. Now, in order for cost function to get close to zero, we will have to reduce the value of θ_3 and θ_4 to near zero. This will in turn greatly reduce the value of $\theta_3 x^3$ and $\theta_4 x^4$ in our hypothesis function. As a result, we see that new hypothesis looks like a quadratic function but fits the data better due to the extra small terms $\theta_3 x^3$ and $\theta_4 x^4$.

Intuition



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

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

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

we penalize and make θ_3, θ_4, really small.

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

→ above ~~Intitution~~ Intuition shows how we can penalize θ and make value of θ very small ≈ 0, In order to fit ~~hypothesis~~ hypothesis on data ~~perfect~~ just right.

We could also regularize all of our ~~data~~ theta parameters in a single summation as:

Cost function

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

The λ, or lambda, is the Regularization parameter, it determines how much the costs of our theta parameters are inflated.

*enlivo
Date: ___/___/_____
Page: _____

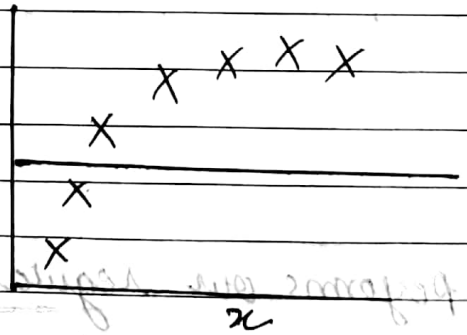
Small value for parameters $\theta_0, \theta_1, \ldots, \theta_n$
- "Simpler" hypothesis
- Less likely to overfitting.

Using this cost function with extra summation, we can smooth the output of our hypothesis function to reduce overfitting. if lambda is chosen to be too large, it may smooth out the function too much and cause underfitting.

In regularized linear regression, we choose θ to minimize:

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

if λ is set to an extremely large value, (say $\lambda = 10^{10}$)
- Algorithm work fine; setting λ to be very large can't hurt it
- Algorithm fails to eliminate overfitting.
- Algorithm results in underfitting. (fails to fit even training data well).
- Gradient descent will fail to converge.



θ_j will be $\theta_j \approx 0$, $(j = 1, 2, \ldots n)$ except θ_0 and cause underfit.

$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$
$\theta_1 \approx 0 \quad \theta_2 \approx 0 \quad \theta_3 \approx 0 \quad \theta_4 \approx 0$

☆ Regularized Linear Regression

→ Cost function

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

$j \in \{1, 2, 3 \ldots n\}$, θ_0 is excluded (The bias term)

$\min_\theta J(\theta)$

→ Gradient Descent

We will modify our gradient descent function to seperate out θ_0 from the rest of the parameters because we do not want to penalize θ_0.

Repeat {

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

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

$j \in \{1, 2, \ldots n\}$

}

The term $\frac{\lambda}{m}\theta_j$ peforms our regularization. with