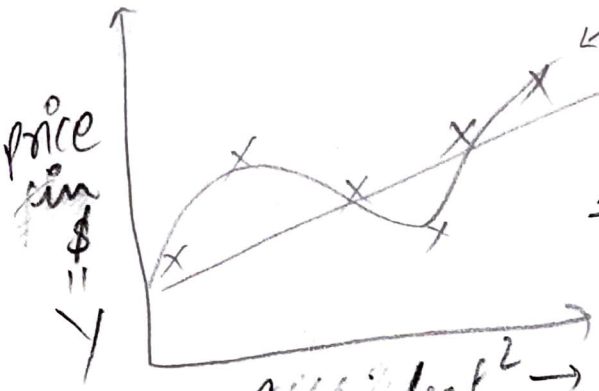


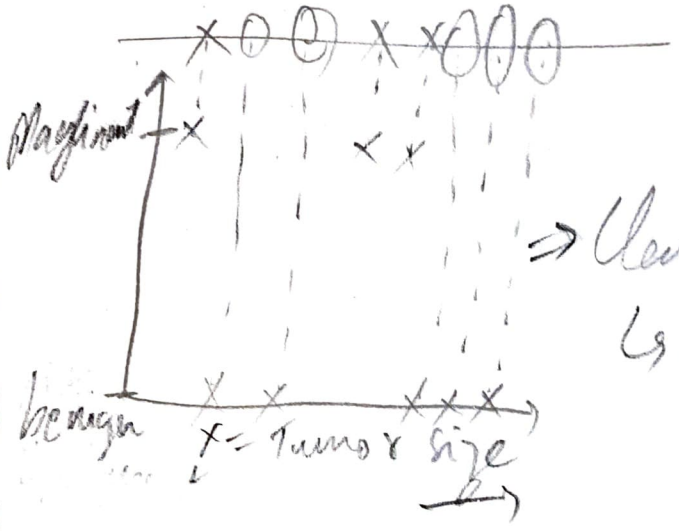
⇒ Regression

⇓

try to find prediction function mapping a given set of data with para ⇒ (x, y)

real & continuous ←

where ∀ X, Y ∈ ℝ



→ 1D classification

⇒ Classification

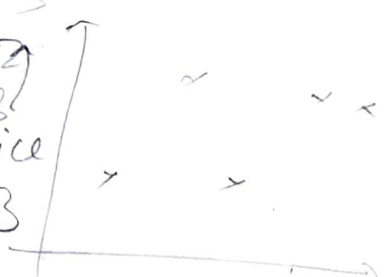
↳ Y → discrete

X → continuous

2D classification

Linear regression

Size (feet^2) x	Price ($1000) y	Bedrooms x_2
2104	400	3
1416	232	2
1534	315	3
852	178	3
⋮	⋮	⋮



Training Set

Learning Algo.

size → 'h' func. → estimated price

(how do you represent h?)

$h(x) = \theta_0 + \theta_1 x$ ← 2 parameters / 1 feature

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

x_1 = size (feet^2) ← 3 parameters / 2 features

x_2 = # bedrooms

hypothesis func.

$h(x) = \sum_{j=0}^{n} \theta_j x_j$ → for n parameters or $n-1$ features

$h(x) = \sum_{j=0}^{2} \theta_j x_j$

where: $x_0 = 1$

x = "inputs" / features θ = parameters

y = "output" / target variable

m = # training examples (# rows in table above)

(x, y) = training example

n = # features

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$ $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$

for n features there are $n+1$ parameters (θ)

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

$x_1^{(1)} = 2104$

$x_1^{(2)} = 1416$

Choose θ s.t. $h(x) \approx y$ for training examples

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

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

choose θ by

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

Choose θ s.t. $J(\theta)$ is minimized

Gradient descent

Start with θ (say $\theta = \vec{0}$)

Keep changing θ to reduce $J(\theta)$

$$\theta_j := \theta_j - \alpha \left(\frac{\partial}{\partial \theta_j} J(\theta) \right) \quad (j = 0, 1, 2, \ldots n)$$

$:=$ assignment operator → $a := a+1$ ↳ increment value of a by 1

α ↑ learning rate

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2$$

$$= 2 \cdot \frac{1}{2}(h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j}(h_\theta(x) - y)$$

$$= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j}(\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n - y)$$

for $j = 1 \Rightarrow (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_1}(\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n)$

$$\Rightarrow (h_\theta(x) - y) \cdot x_1$$

$\therefore$, $= (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j}(\theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_j x_j + \cdots + \theta_n x_n)$

$$= (h_\theta(x) - y)) \cdot x_j$$

$$\theta_j := \theta_j - \alpha(h_\theta(x) - (y)) \cdot x_j$$

(only one training example)

Repeat until convergence

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

for training example

$$\left[\frac{\partial}{\partial \theta_j}(J(\theta)) \right]$$

(for $j = 0, 1, \cdots n$)

a.k.a Batch gradient descent

Repeat {

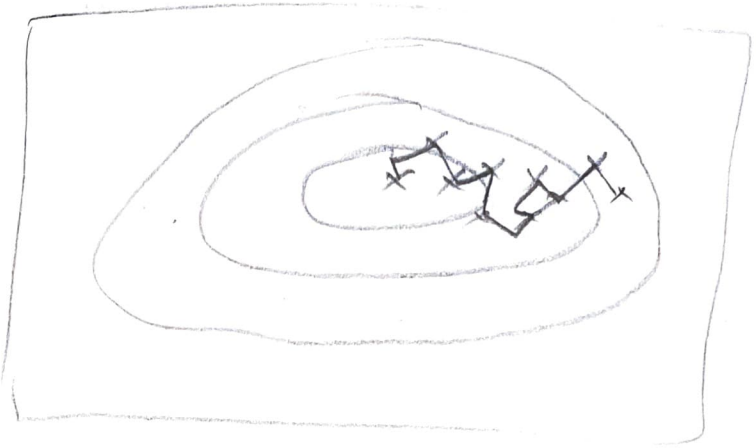
for every j

for $i = 1$ to m {

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

}

} $\Rightarrow$ Stochastic gradient descent



$\theta = (\dots)$

18008969999

$$\nabla_\theta J(\theta) = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \\ \frac{\partial J}{\partial \theta_2} \end{bmatrix}$$

$\theta \in \mathbb{R}^{n+1}$

$A \in \mathbb{R}^{2 \times 2}$ $\quad A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$

$f(A) : \mathbb{R}^{2\times 2} \longrightarrow \mathbb{R}$

$f(A) = A_{11} + A_{12}^2$

$f\left(\begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}\right) = 5 + 6^2 = 41$

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \frac{\partial f}{\partial A_{12}} \\ \frac{\partial f}{\partial A_{21}} & \frac{\partial f}{\partial A_{22}} \end{bmatrix}$$

$$\text{If } \nabla_A f(A) = \begin{bmatrix} 1 & 2A_{12} \\ 0 & 0 \end{bmatrix}$$

$\nabla_\theta J(\theta) \overset{\text{set}}{=} \vec{0}$

If A is a square. $(A \in \mathbb{R}^{n\times n})$

$\underbrace{\text{tr } A}_{\text{trace of } A}$ = sum of diagonal entries

$\text{tr } A = \text{tr } A^T$

$f(A) = \text{tr } AB$

$\nabla_A f(A) = B^T$

$\text{tr } AB = \text{tr } BA$

$\text{tr } ABC = \text{tr } CAB$

$$\nabla_A \operatorname{tr} AA^T C = CA + C^T A$$

$\left\{ \frac{d}{da} a^2 c = 2ac \right\}$ similar.

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

$$X\theta = \begin{bmatrix} -(x^{(1)})^T- \\ -(x^{(2)})^T- \\ \vdots \\ -(x^{(m)})^T- \end{bmatrix} \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$$

$$= \begin{bmatrix} (x^{(1)})^T\theta \\ (x^{(2)})^T\theta \\ \vdots \\ (x^{(m)})^T\theta \end{bmatrix} = \begin{bmatrix} h_\theta(x^{(1)}) \\ h_\theta(x^{(2)}) \\ \vdots \\ h_\theta(x^{(m)}) \end{bmatrix}$$

$$\vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$$J(\theta) = \frac{1}{2}(X\theta - y)^T(X\theta - y)$$

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

$$z^T z = \sum_i z^2$$

$$\Big\} (X\theta - y)^T(X\theta - y)$$

$$\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{2}(X\theta - y)^T(X\theta - y)$$

$$= \frac{1}{2}\nabla_\theta (\theta^T X^T - y^T)(X\theta - y)$$

$$= \frac{1}{2}\nabla_\theta (\theta^T X^T X\theta - yX^T\theta^T - y^T X\theta + y^T y)$$

$$\Leftarrow \frac{1}{2}\{(ax-b)(ax-b) = a^2x^2 - axb - bax + b^2\}$$

$$= \frac{1}{2}[X^T X\theta + X^T X\theta - X^T y - X^T y]$$

$$= X^T X\theta - X^T y \overset{\text{set}}{=} \vec{0}$$

$$X^T X\theta = X^T y$$

NORMAL EQN

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

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

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

1 2 3

i.e. $\theta_1 = 0 \Rightarrow h_\theta(x) = 0$

$$J(0) = \frac{1}{2\times 3} \left((0-1)^2 + (0-2)^2 + (0-3)^2 \right)$$

$$= \frac{1}{6}(1+4+9) = \frac{7}{3}$$

$$J(\theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\theta_1 x_i - y^{(i)})^2$$

---

gradient descent

$\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

(1) Start at $\theta_0 = 0$, $\theta_1 = 0$ (it can be any value though)

(2) Keep changing θ_0 & θ_1 to reduce $J(\theta_0, \theta_1)$

(3) until we reach minimum

$$\theta_j := \theta_j + \sqrt{\theta_0 \theta_1} \qquad \theta_0 = 1,\ \theta_1 = 2$$

$$\theta_0 := \theta_0 + \sqrt{\theta_0 \theta_1} = 1 + \sqrt{1 \times 2} = 1 + \sqrt{2}$$

$$\theta_1 := \theta_0 + \sqrt{\theta_0 \theta_1} = 2 + \sqrt{1 \times 2} = 2 + \sqrt{2}$$

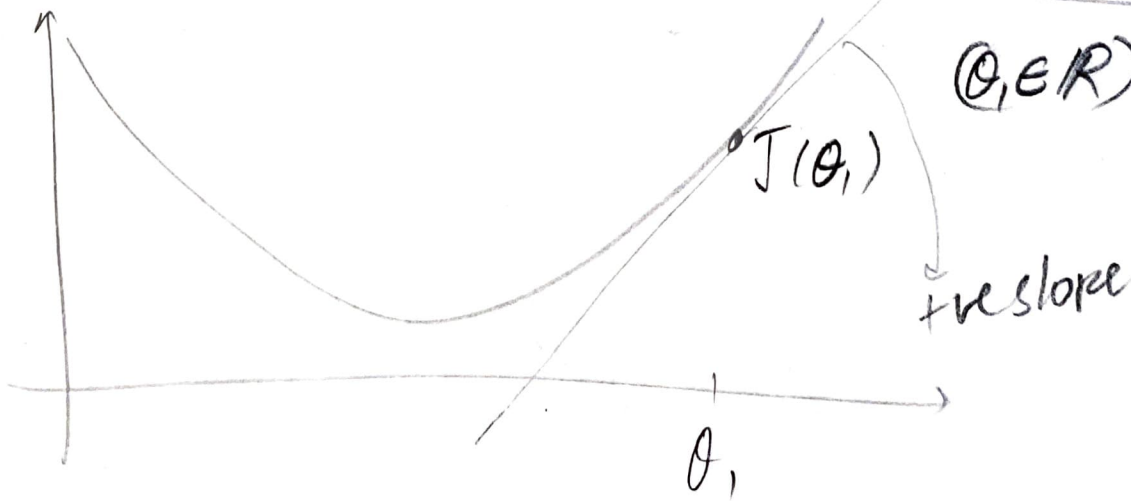
Correct Simultaneously update:

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

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

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

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

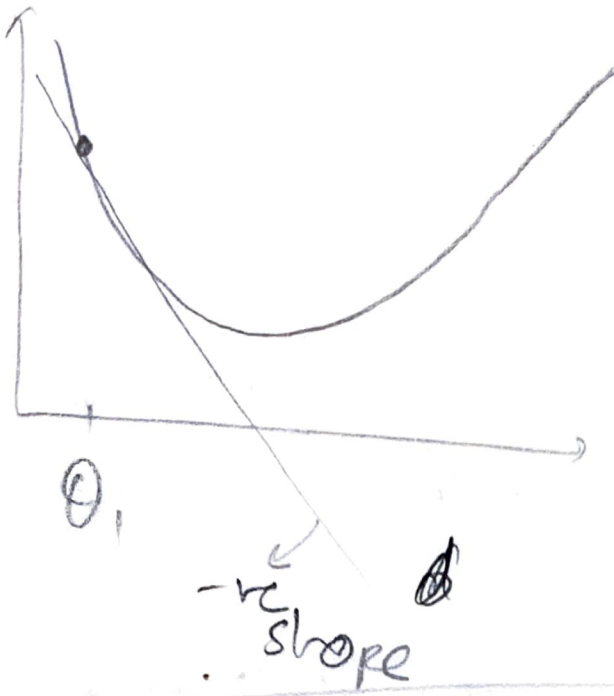


$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} (J(\theta_1))$$

$\theta_1 := \theta_1 - \alpha (+ve)$

$\rightarrow 0 (+ve) \rightarrow -ve$

$\rightarrow$ dec$^n \downarrow$



$\frac{d}{d\theta_1} J(\theta_1)$

$\theta_1 := \theta_1 - \alpha (-ve)$

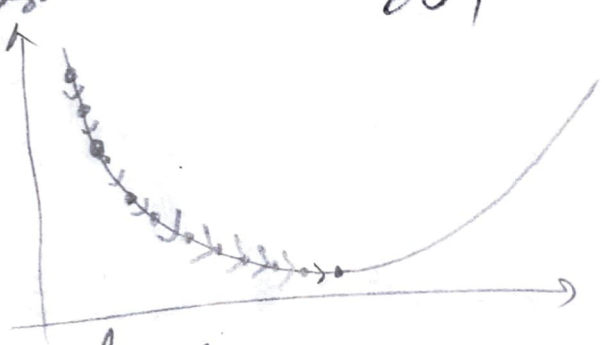
+ve

+ve

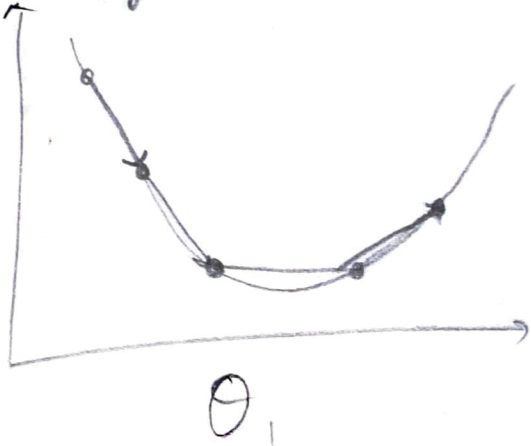
inc$^n \uparrow$

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

If α is small



If α is large



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

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

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

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

for $j = 0, \theta_0 \Rightarrow \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} h_\theta(x^{(i)}) - y^{(i)})$

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

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

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

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

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

for $j = 0, \frac{\partial}{\partial \theta_0} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \frac{\partial}{\partial \theta_0} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})$

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

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

Gradient descent for linear regression

repeat until convergence {

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

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

(Update θ_0 & θ_1 simultaneously)

$h_\theta(x) = \theta_0 = 0,\ \theta_1 = 1$ → $\theta_0 + \theta_1 x^{(i)} \equiv x^{(i)}$

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

$x^{(i)} - y^{(i)}$

x	y
3	2
1	2
0	1
4	3

$$J(0,1) = ? = \frac{1}{8}(1 + 1 + 1 + 1)$$

$$= \frac{1}{2}\begin{bmatrix}1 & 2 & 4\\ 5 & 3 & 2\end{bmatrix} + 2$$

$h_\theta(6) = \theta_0 + \theta_1 x$

$$\begin{matrix}2\\3\\\frac{7}{2}\end{matrix} - \begin{matrix}6\\3\\0\end{matrix}\begin{bmatrix}-4\\0\\7/2\end{bmatrix} = -2 + \frac{1}{2}\times\frac{3}{6}\begin{bmatrix}2 & 2 & 2\\ 2 & 2 & 2\end{bmatrix} = 1$$

$2(4-4)$

$-2(4-4)$

$+2($

$$\begin{matrix}11\\16\end{matrix}\quad\begin{matrix}16\\2\end{matrix}\quad\begin{bmatrix}3\\12\\6\end{bmatrix} - \begin{bmatrix}0\\0\\5\end{bmatrix} - \begin{bmatrix}1\\0\\2\\\frac{2}{3}\end{bmatrix}$$

$$\begin{bmatrix}2\\12\\\frac{31}{3}\end{bmatrix} \quad \begin{bmatrix}3\\12\\11\end{bmatrix} \quad \begin{bmatrix}1\\0\\2/3\end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 \\ 4 & 0 \\ 2 & 1 \end{bmatrix}_{3\times 2} \times \begin{bmatrix} 1 \\ 5 \end{bmatrix}_{2\times 1} = \begin{bmatrix} 16 \\ 4 \\ 7 \end{bmatrix}$$

$h_\theta(x) = -40 + \frac{x}{4}$

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix}_{4\times 2} \begin{bmatrix} -40 \\ 0.25 \end{bmatrix}_{2\times 1} = \begin{bmatrix} 486 \\ 314 \\ 384.5 \\ 173 \end{bmatrix}$$

526 − 40 = 486

$-40 + 526 =$

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

$1+6+2+5$

$0+9+0+4$

$-1-6+0+0$

$y = ax + b$

$Y = XP$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad P = \begin{bmatrix} a \\ b \end{bmatrix}$$

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

$y = -40 + \frac{x}{4}$

$\frac{x}{4} - y = 40$

$X = \begin{bmatrix} x \end{bmatrix}$ $B = \begin{bmatrix} 40 \end{bmatrix}$

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

$1+0+10=11$ $3+3+4$

$4+0+5=9$ $12+0+2$

$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 9 & 7 \\ 15 & 12 \end{bmatrix}$$

$0+9$ $\quad$ $1+6$ $\quad$ $2+10$

$0+15$

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

$$\begin{bmatrix} 1 & 3 \\ 2 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 7 \\ 12 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 2 & 3 \end{bmatrix} = \begin{bmatrix} 7 & 9 \\ a & b \\ c & d \end{bmatrix}$$

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 0 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 7 \\ 10 \\ 10 \end{bmatrix}$$

$a = 10$
$b = 12$
$c = 10$
$d = 15$

$2+8$ $\quad$ $6+10$

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \\ 6 & 5 \end{bmatrix} \begin{bmatrix} 0 \\ 3 \end{bmatrix} = \begin{bmatrix} 9 \\ 12 \\ 15 \end{bmatrix}$$

$0+9$ $\quad$ $0+12$

House Sizes
2104
1416
1534
852

3 competing hypotheses

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

$h_\theta(x) = 200 + 0.1x$

$h_\theta(x) = -150 + 0.4x$

$$\begin{bmatrix} 1 & 2104 \\ 1 & 1416 \\ 1 & 1534 \\ 1 & 852 \end{bmatrix} \times \begin{bmatrix} -40 & 200 & -150 \\ 0.25 & 0.1 & 0.4 \end{bmatrix}$$

$$= \begin{bmatrix} 486 & 410 & 692 \\ 314 & 342 & 416 \\ 344 & 353 & 464 \\ 173 & 285 & 191 \end{bmatrix}$$

---

identity matrix

[1] $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ $\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

informally, $\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$ $n \times n$

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

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

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

$1 \cdot 1 + 0 \cdot 3 + 0 \cdot 2 = 1$

$0 \cdot 1 + 1 \cdot 3 + 0 \cdot 2 = 3$

$0 \cdot 1 + 0 \cdot 3 + 1 \cdot 2 = 2$

Supervised learning

- the target variable is continuous → Regression
- target variable is discrete → Classification

① Linear Regression

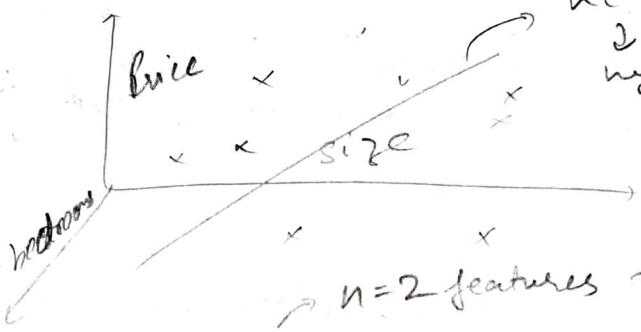
WORKFLOW

Feed training data → Learning Algo → h(x) func^n

m training examples

Hypothesis func^n

x_1 size (feet²)	y Price (1000$)	x_2 # bedrooms
2104	400	3
1600	330	3
2400	369	3
1416	232	2
3000	540	4
⋮	⋮	⋮



h(x) → hypothesis func^n → we need to find it

n = # features (n = 2)

m = # training examples (m = 47)

n = 2 features

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

$\theta_0, \theta_1, \theta_2$ → 3 parameters (or weights)

$x_1, x_2, \ldots$ → inputs (or features)

↳ x_1 = size (in feet²)

x_2 = # bedrooms

In general,

$$h(x) = \sum_{j=0}^{n} \theta_j x_j$$

where, $x_0 = 1$ (for n features)

and for n features there are n+1 parameters

Also, $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$ & $x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$

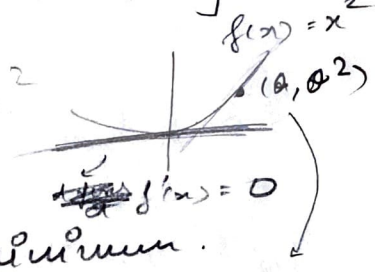
for n features there are n+1 parameters (θ)

$$h(x) = \sum_{j=0}^{n} \theta_j x_j = \theta^T x \qquad \theta = \begin{bmatrix} \theta_1 \\ \theta_m \end{bmatrix}$$

$(x^{(i)}, y^{(i)})$ → i^{th} training example

$$X = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_m \end{bmatrix}$$

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



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

↓ (cost func^n)

Task ⇒ Choose θ s.t. $J(\theta)$ is minimum.

$\frac{dx^2}{dx} = 2x$

$2x = 0$

$x = 0$

① LMS Algo (Gradient descent)

Starts with θ (say $\theta = \vec{0}$)

$\theta - 0.01 \times 2\theta$

$\theta - 0.02\theta$

$1.98\theta \cdot 001$

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

assignment operator

learning rate (size of step towards a minima)

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_\theta(x) - y)^2$$

$$= \frac{1}{2} \cdot 2 (h_\theta(x) - y) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{n} \theta_i x_i - y \right)$$

$$= (h_\theta(x) - y) \cdot (x_j)$$

$$\boxed{\theta_j := \theta_j - \alpha (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}}$$

Repeat this until convergence

(for $j = 0, 1, \ldots, n$) {

$$\theta_j := \theta_j - \alpha \sum_{i=0}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

}

Stochastic gradient descent

loop {

for $i = 1$ to $i = m$ {

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

(for $j = 0 \to n$)

}

}

Useful for large datasets (i.e large value of m)

Matrix derivatives

$f : \mathbb{R}^{m \times n} \mapsto \mathbb{R}$

gradient of $f(A)$ w.r.t A

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & & & A_{2n} \\ \vdots & & & \vdots \\ A_{m1} & A_{m2} & & A_{mn} \end{bmatrix}_{m \times n}$$

$$\nabla_A f(A) = \begin{bmatrix} \frac{\partial f}{\partial A_{11}} & \frac{\partial f}{\partial A_{12}} & \cdots & \frac{\partial f}{\partial A_{1n}} \\ \frac{\partial f}{\partial A_{21}} & \frac{\partial f}{\partial A_{22}} & & \frac{\partial f}{\partial A_{2n}} \\ \vdots & & & \vdots \\ \frac{\partial f}{\partial A_{m1}} & \frac{\partial f}{\partial A_{m2}} & & \frac{\partial f}{\partial A_{mn}} \end{bmatrix}_{m \times n}$$

derivative of f w.r.t A

$\Rightarrow (i, j)^{th}$ element $\frac{\partial f}{\partial A_{ij}}$

If A is a 2×2 matrix $\Rightarrow A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$

and func^n $f : \mathbb{R}^{2 \times 2} \to \mathbb{R}$

$$f(A) = \frac{3}{2} A_{11} + 5 A_{12}^2 + A_{21} A_{22}$$

$$\nabla_A f(A) = \begin{bmatrix} \frac{3}{2} & 10 A_{12} \\ A_{22} & A_{21} \end{bmatrix}$$

Trace of a matrix

$A \to n \times n$ matrix

$$tr\, A = \sum_{i=0}^{n} A_{ii} = A_{11} + A_{22} + \cdots + A_{nn}$$ (only for square matrix)

* If $a \in \mathbb{R}$ i.e. a is 1×1 matrix
then, $\boxed{tr(a) = a}$ proof (1)

If A & B are two matrices s.t. $\boxed{tr\,BA = tr\,AB}$

$\boxed{tr\,ABC = tr\,CAB = tr\,BCA}$

$\boxed{tr\,ABCD = tr\,DABC = tr\,CDAB = tr\,BCDA}$

$\boxed{tr\,A = tr\,A^T}$

$\boxed{tr\,(A+B) = tr\,A + tr\,B}$

$\boxed{tr\,aA = a\,tr\,A}$

$\boxed{\nabla_A tr\,AB = B^T}$

$\boxed{\nabla_{A^T} f(A) = (\nabla_A f(A))^T}$

$\boxed{\nabla_A ABA^TC = CAB + C^TAB^T}$

$\boxed{\nabla_A |A| = |A|(A^{-1})^T}$

$$X = \begin{bmatrix} -(x^{(1)})^T- \\ -(x^{(2)})^T- \\ \vdots \\ -(x^{(m)})^T- \end{bmatrix} \quad \vec{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

$$h_\theta(x^{(1)}) = (x^{(1)})^T\theta$$

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

$$= \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}$$

∵, $z^T z = \sum_i z_i^2$

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

$$= J(\theta)$$

∵, $\nabla_{A^T} \operatorname{tr} ABA^TC = B^TA^TC^T + BA^TC$

$$\nabla_\theta J(\theta) = \nabla_\theta \frac{1}{2}(X\theta - \vec{y})^T(X\theta - y)$$

$$= \frac{1}{2}\nabla_\theta(\theta^T X^T - \vec{y}^T)(X\theta - y)$$

$$= \frac{1}{2}\nabla_\theta(\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta + y^T y)$$

$$= \frac{1}{2}\nabla_\theta \operatorname{tr}(\theta^T X^T X\theta - \theta^T X^T y - y^T X\theta + y^T y)$$

$$= \frac{1}{2}\nabla_\theta(\operatorname{tr}\theta^T X^T X\theta - 2\operatorname{tr}\vec{y}^T X\theta) \quad ???$$

$$= \frac{1}{2}(X^T X\theta + X^T X\theta - 2X^T\vec{y})$$

$$= X^T X\theta - X^T y$$

$\nabla_\theta J(\theta) \overset{\text{set}}{=} \vec{0}$ for θ st $J(\theta)$ in min

$$\nabla_\theta J(\theta) = X^T X\theta - X^T y = 0$$

$$X^T X\theta - X^T y = 0$$

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

$3(3^{-1}) = 1$ $12(12^{-1}) = 1$ $0(0^{-1}) = 1$ ✗ WRONG (undefined)

$AA^{-1} = I = A^{-1}A$ only for m×m (or, square) matrices

I ↳ (A matrix) (inverse of A matrix)

$\begin{bmatrix} 3 & 4 \\ 2 & 16 \end{bmatrix}_{2\times2}$ (A) $\quad \begin{bmatrix} 0.4 & -0.1 \\ 0.05 & 0.075 \end{bmatrix}_{2\times2}$ (A^{-1}) $= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_{2\times2}$

$AA^{-1} =$

$A^{\top} = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 5 & 9 \end{bmatrix}_{m\times n}$ Transpose $A^{T} = \begin{bmatrix} 1 & 3 \\ 2 & 5 \\ 0 & 9 \end{bmatrix} = B$

2×3 $\quad n\times m$ $\quad 3\times2$

$A_{ij} = B_{ji}$

$A_{12} = 2 \rightarrow B_{21} = 2$ $\quad B_{32} = 9$ $\quad A_{23} = 9$

$\begin{bmatrix} 1 & 3 \\ 2 & 0 \\ 1 & 4 \end{bmatrix}^{T} \quad \begin{bmatrix} 1 & 2 \\ 0 & 1 \\ 3 & 4 \end{bmatrix}$

$A = \begin{bmatrix} 1 & 0 & 1 \\ 2 & -2 & -1 \\ 3 & 0 & 0 \end{bmatrix}$

$adj(A) = \begin{bmatrix} +0 & -3 & +6 \\ -0 & +3 & -0 \\ +2 & -3 & +2 \end{bmatrix}^{T}$

$adj(A) = \begin{bmatrix} 0 & -3 & 6 \\ 0 & -3 & 0 \\ 2 & 3 & -2 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 2 \\ -3 & -3 & 3 \\ 6 & 0 & -2 \end{bmatrix}$

$A^{-1} = \frac{1}{|A|} adj(A)$

$|A| = 1(0-0) - 0(\ ) + 1(0+6)$

$= 6$

$A^{-1} = \begin{bmatrix} 0 & 0 & 1/3 \\ -1/2 & -1/2 & 1/2 \\ 1 & 0 & -1/3 \end{bmatrix}$

$A^{-1} = \frac{1}{6}\begin{bmatrix} 0 & -3 & 6 \\ 0 & -3 & 0 \\ 2 & 3 & -2 \end{bmatrix}$

$A^{-1} = \begin{bmatrix} 0 & -1/2 & 1 \\ 0 & -1/2 & 0 \\ 1/3 & 1/2 & -1/3 \end{bmatrix}$

Linear Regression in multivariable

Size (feet²) x_1	# bedrooms x_2	# floors x_3	Age x_4	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
⋮	⋮	⋮	⋮	

this data (training set) has 4 features

n = # features

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

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

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix} \in \mathbb{R}^4 \quad (\mathbb{R}^n)$$

$x_3^{(2)} = 2$

$x_1^{(4)} = 852$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

(x_1 → size, x_2 → # bedrooms, x_3 → # floors, x_4 → Age)

Eg) $h_\theta(x) = 80 + 0.1 x_1 + 0.01 x_2 + 3 x_3 - 2 x_4$

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n \quad (\text{where } x_0^{(i)} = 1)$$

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

$\boxed{x_0 = 1}$

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

$$\theta^T = [\theta_0 \ \theta_1 \ \cdots \ \theta_n]_{1 \times (n+1)}$$

row matrix vector

$$\theta^T x = [\theta_0 \ \theta_1 \ \cdots \ \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$= \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

→ a.k.a multivariate linear regression.

$$\therefore h_\theta(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x$$

where, $x_0 = 1$

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

[cost function]

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

where, $x_0 = 1$

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

Gradient descent:

Repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \quad \text{(for every } j = 0, 1, \ldots, n\text{)}$$

[simultaneous update]

}

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

$$= \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \frac{\partial}{\partial \theta_j} \sum_{i=1}^{n} (h_\theta(x^{(i)}) - \dots y^{(i)})$$

$$= \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \times \frac{\partial}{\partial \theta_j} \left(\sum_{j=1}^{n} (\theta_j \cdot x_j^{(i)}) - y^{(i)} \right)$$

$$= \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \qquad \left(\frac{\partial}{\partial \theta_j} \left(\sum_{j=1}^{n} (\theta_j \cdot x_j^{(i)}) - y^{(i)} \right) = x_j^{(i)} \right)$$

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

Gradient descent for $n \geq 1$:

Repeat until convergence {

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

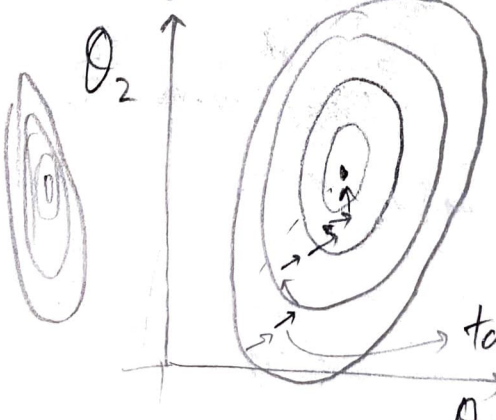
} (for every $j = 0, 1, 2, \dots, n$)

[simultaneous update]

Feature Scaling ⇒ Idea: Make sure features are on a similar scale.

Eg) x_1 = size (0-2000 feet^2)

x_2 = # bedrooms (1-5)

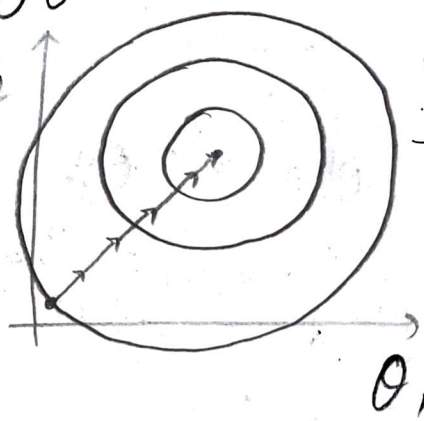


→ very skewed elliptical contours

→ taking a long time to converge or might not converge to minima

But scaling the feature properly by multiplying or dividing the inputs by an appropriate

$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$; $x_2 = \frac{\#\text{ bedrooms}}{5}$ $\quad 0 \le x_1 \le 1$ $\quad 0 \le x_2 \le 1$



$J(\theta)$ → contour of func^n is less skewed i.e., reaching to minima is easier

Get every feature into approx. a range. $\boxed{-1 \le x_i \le 1}$ where, $x_0 = 1$

Eg) $0 \le x_1 \le 3$ ✓ ; $-2 \le x_2 \le 0.5$ ✓ ; $-100 \le x_3 \le 100$ X

; $-0.0001 \le x_4 \le 0.0001$ X

Mean Normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Don't apply $x_0 = 1$)

Eg) $x_1 = \frac{\text{size} - 1000}{2000}$; $x_2 = \frac{\#\text{bedrooms} - 2}{5}$

(1000 ← avg. size; 2 ← avg # bedroom)

; $-0.5 \le x_1 \le 0.5$; $-0.5 \le x_2 \le 0.5$

generally,

$x_1 \rightarrow \frac{x_1 - \mu_1}{s_1}$ (μ_1 → avg. value of x_1 in training set; s_1 → range (max − min) or → Std deviation of x_1)

this is,

$x_2 \rightarrow \frac{x_2 - \mu_2}{s_2}$

$x_i = \frac{\text{Age} - 40}{20}$ $30 \le \text{Age} \le 50$

x_i ↓ input feature

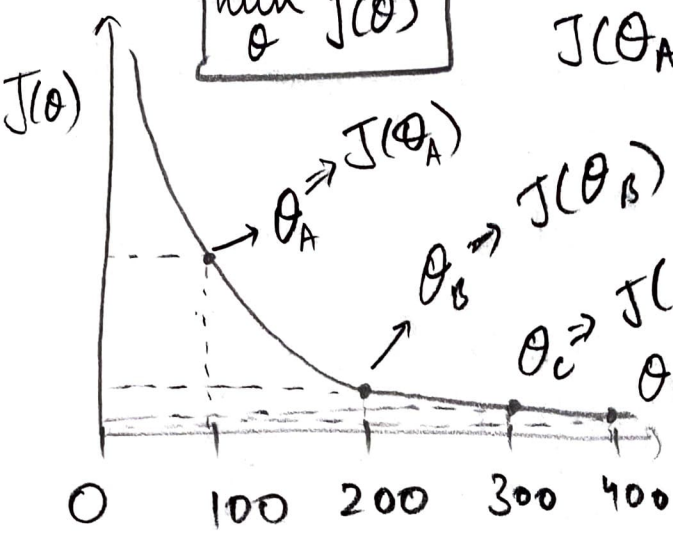
40 → avg Age

20 → range of Age

feature scaling can make gradient descent much faster

Debugging Gradient descent.

$\min_{\theta} J(\theta)$



$J(\theta_A) > J(\theta_B) > J(\theta_C) > J(\theta_D)$

$J(\theta)$ should dec↓ after every iteration

for this case $J(\theta)$ don't change significantly after 300 iterations

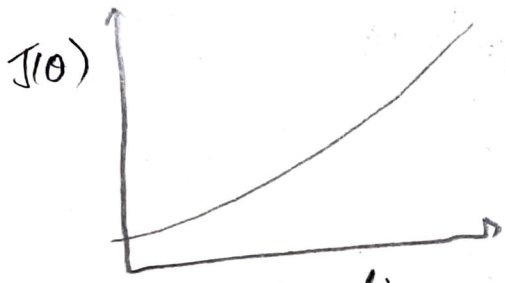
No. of iterations for gradient descent to get $\min_{\theta} J(\theta)$ can vary significantly

⇒ Eg of a automatic convergence test:

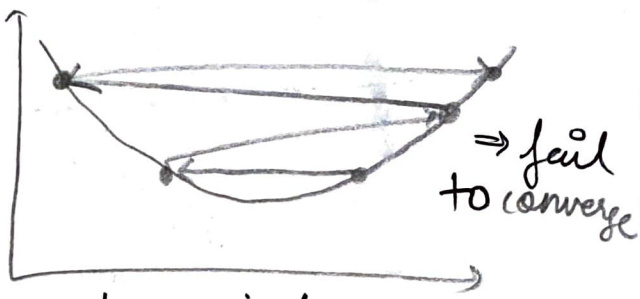
Declare convergence if $J(\theta)$ is less than 10^{-3} in one iterations

↳ this threshold is not definitive and plotting $J(\theta)$ v/s # iteration graph is better to declare convergence.

If $J(\theta)$ v/s # iteration looks like



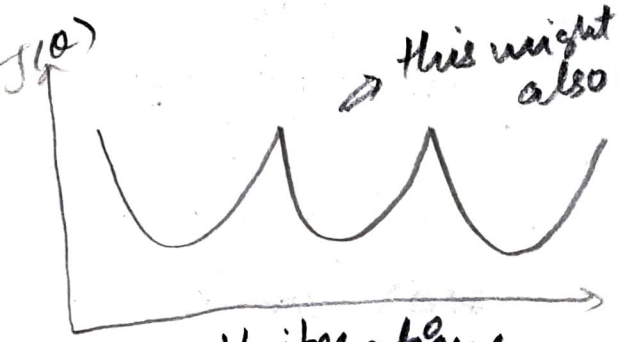
⇒ this means α (i.e learning rate is too large) and dec^{n} the value of α



⇒ fail to converge

this might also happen the code have bug(s)

When α is large



⇒ this might also happen if α is large

⇒ For sufficiently small α, $J(\theta)$ should decrease on every iteration (theorem)

⇒ but if α is too small then, can be slow to converge

Summary,

⇒ α is too small ⇒ slow convergence

⇒ α is too large ⇒ $J(\theta)$ may not converge with every iteration as it does decrease or convergence may happen but slowly.

(α = ..., 0.001, 0.01, 0.1, ..., 1 ... so on) ×3 → 0.003 ×3 → 0.03 ×3 ↗ try to choose like this in practice

Housing Prices Prediction

$h_\theta(x) = \theta_0 + \theta_1 \times \underbrace{\text{frontage}}_{x_1} + \theta_2 \times \underbrace{\text{depth}}_{x_2}$

↓ by linear regression

new feature frontage

↑

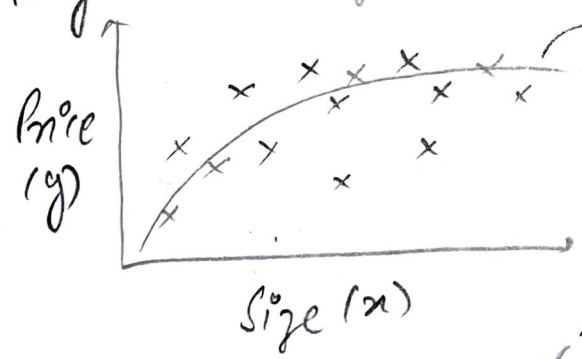
$\text{Area} = x = \text{frontage} \times \text{depth}$

$h_\theta(x) = \theta_0 + \theta_1 x = \theta_0 + \theta_1 \times \underbrace{\text{Area}}_{x}$

→ this new feature framed using two given features of training data set

∴, only two parameters instead of three

Polynomial Regression



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

→ but, this func^n moves down with inc^n↑ in x (size) which doesn't make sense

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

↓

this func^n inc^n↑ with inc^n↑ in size (x)

↳ which does make sense

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3$

$= \theta_0 + \theta_1 (\text{size}) + \theta_2 (\text{size})^2 + \theta_3 (\text{size})^3$

$x_1 = (\text{size})$; $x_2 = (\text{size})^2$; $x_3 = (\text{size})^3$

↳ 3 different features using single given feature (size → x)

POLYNOMIAL REGRESSION

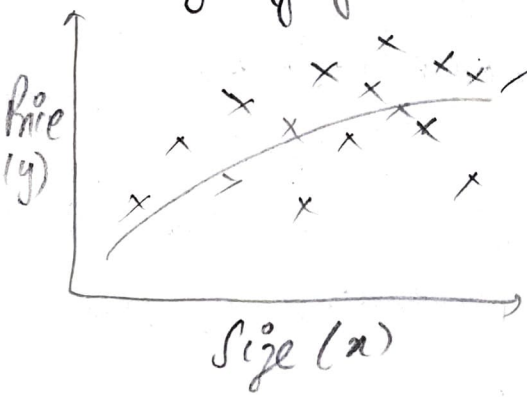
If, size : $1 \rightarrow 10^3$ $\quad \text{size}^3 = 1 \rightarrow 10^9$

then, $\text{size}^2 = 1 \rightarrow 10^6$

↳ quite large range

use feature scaling to improve gradient descent.

Choosing diff. features



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

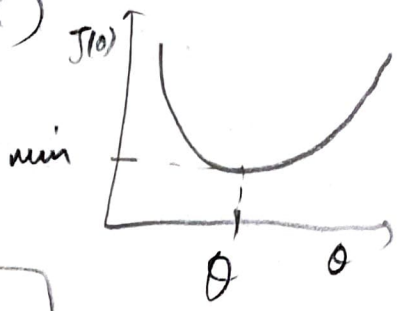
$= \theta_0 + \theta_1 (\text{size}) + \theta_2 \sqrt{(\text{size})}$

using diff set of features for same set of training data

NORMAL EQⁿ : Method to solve for θ analytically

Intuition : If $1D$ $(\theta \in \mathbb{R})$

$$J(\theta) = a\theta^2 + b\theta + c$$



$$\frac{d}{d\theta} J(\theta) = 2a\theta + b \overset{\text{set}}{=} 0$$

Solve for, $\boxed{\theta = \frac{-b}{2a}}$

$\theta \in \mathbb{R}^{n+1}$ $J(\theta_0, \theta_1, \theta_2, \cdots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

$\frac{\partial}{\partial \theta_j} J(\theta) \overset{\text{set}}{=} 0$ (for every j)

then solve for $\theta_0, \theta_1, \theta_2, \cdots, \theta_n$

Eg) $m = 4$

x_0	Size (feet²) x_1	# bedrooms x_2	# floors x_3	Age x_4	Price 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

$m = 4$

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}_{m \times (n+1)} \quad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}_m$$

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

normal eqⁿ

m examples $(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})$

, n features

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$$

$$X = \begin{bmatrix} \text{---} (x^{(1)})^T \text{---} \\ \text{---} (x^{(2)})^T \text{---} \\ \vdots \\ \text{---} (x^{(m)})^T \text{---} \end{bmatrix}_{m \times (n+1)}$$

(design matrix)

Eg) If $x^{(i)} = \begin{bmatrix} 1 \\ x_1^{(i)} \end{bmatrix}$ here, $n = 1$

$$X = \begin{bmatrix} 1 & x_1^{(1)} \\ 1 & x_1^{(2)} \\ \vdots & \vdots \\ 1 & x_1^{(m)} \end{bmatrix}_{m \times 2} \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}_{m}$$

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

Set $A = X^T X$, $A^{-1} = (X^T X)^{-1}$

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

not need for feature scaling

$\theta = (X^T X)^{-1} X^T y \Rightarrow$ s.t. $\min_\theta J(\theta)$

let m training examples, n features

Gradient Descent	Normal Equation
⇒ Need to choose 'α'	⇒ No need to choose 'α'
⇒ Needs many iterations	⇒ Don't need to iterate
⇒ works well even when n is large	⇒ Need to compute ** $(X^TX)^{-1}$ ⇒ Slow if n is very large
If $n \approx 10^6$ or larger	$n = 100, 1000, 10{,}000$ ⇒ might be slow (fast / reasonable)

** If $X \Rightarrow \mathbb{R}^{m \times n+1}$ then $X^T \Rightarrow \mathbb{R}^{n+1 \times m}$

Set, $X^TX = A \Rightarrow \mathbb{R}^{n+1 \times n+1}$ $O(n^3)$ ⇒ time complexity of inverting a matrix

$(n+1 \times m) \times (m \times n+1)$ — same

Square matrix

∴, for large value of n inverting a matrix is computationally expensive.

What if (X^TX) is non-invertable? (i.e. singular matrix / degenerate matrix)

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

↓ pseudo inverse

If X^TX is non-invertable

⇒ Redundant features exist (i.e linearly dependent)

Eg) $x_1 = \text{size (feet}^2\text{)}$; $x_2 = \text{size (m}^2\text{)}$

$x_1 = k x_2$ (linearly dependent)

↳ $x_1 = (3.28)^2 x_2$

→ If $x_1 = k x_2$ then $X^T X$ is non-invertable

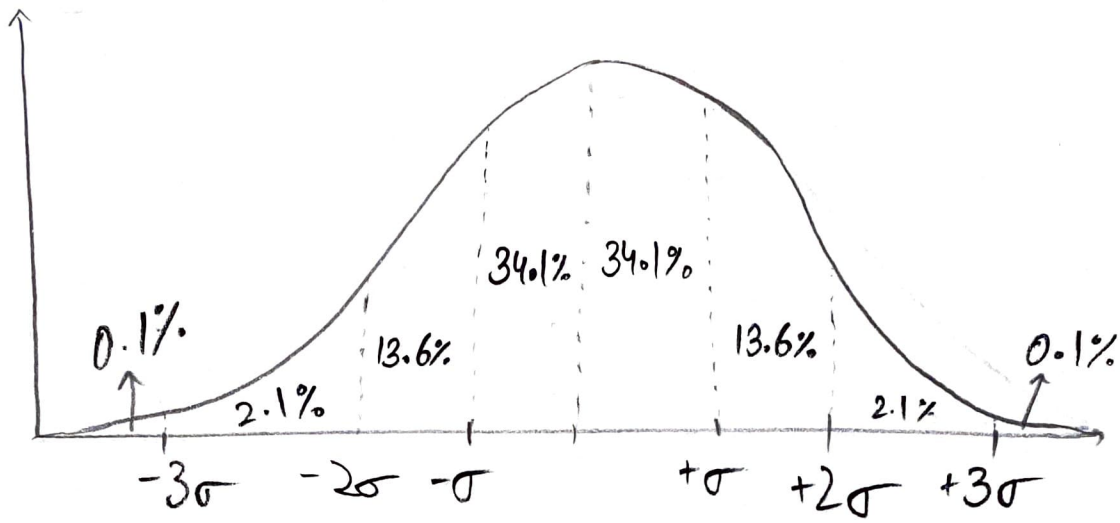
⇒ Too many features (Eg $m \leq n$) ⇒ $m = 10$, $n = 101$ ⇒ $\theta \in \mathbb{R}^{101}$

↳ Delete some features

Sometimes work, or use Regularization

but, not always ⇒ there is too little data

Standard deviation



$$\sigma = \sqrt{\frac{\Sigma(x-\bar{x})^2}{n-1}}$$

Data = 1, 2, 3, 4, 5

$$\bar{x} = \frac{1+2+3+4+5}{5} = \frac{15}{5}$$

$$= \boxed{3}$$

$$\Sigma(x-\bar{x})^2$$

$x=1 \Rightarrow (1-3)^2 = 4$ | $x=2 \Rightarrow (2-3)^2 = 1$ | $x=3 \Rightarrow (3-3)^2 = 0$ | $x=4 \Rightarrow (4-3)^2 = 1$ | $x=5 \Rightarrow (5-3)^2 = 4$

$$\Sigma(x-\bar{x})^2 = 4+1+0+1+4 = 10$$

$$\sigma = \sqrt{\frac{10}{5-1}} = \sqrt{\frac{10}{4_2}} = \sqrt{\frac{5}{2}} = 1.58$$

$$\sigma = \sqrt{\frac{((3.6)^2+(1.6)^2)\times 2+0}{4}}$$

$$= \sqrt{\frac{(12.96+2.56)\times 2+0}{4}} = 2.78$$

$$\bar{x} = \frac{0+2+4+5+7}{5}$$

$$= \frac{18}{5} = 3.6$$

Vectorization

① $h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j = \theta^T x$

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$ → theta(1), theta(2), theta(3)

$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$ → x(1), x(2), x(3)

Unvectorized implementation (MATLAB)

```
prediction = 0.0;
for j = 1:n+1
    prediction = prediction + theta(j) * x(j)
end;
```

Vectorized Implementation (MATLAB)

```
prediction = (theta)' * x;
```

Unvectorize Implementation (C++)

```
double prediction = 0.0;
for (int j = 0; j <= n; j++)
    prediction += theta[j] * x[j];
```

Vectorized Implementation (C++)

```
double prediction = theta.transpose() * x;
```

② Vectorization of gradient descent

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

(for all $j = 0, 1, \cdots, n$)

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

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

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

(for $j = 0, 1, 2$)

Vectorized Implementation:

$$\theta := \theta - \alpha \delta, \quad \theta \in \mathbb{R}^{n+1}, \ \alpha \in \mathbb{R}, \ \delta \in \mathbb{R}^{n+1}$$

where, $\delta = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$, $x^{(i)} \in \mathbb{R}^{n+1}$

$$\delta = \begin{bmatrix} \delta_0 \\ \delta_1 \\ \delta_2 \end{bmatrix}$$

$$\delta_0 = \frac{1}{m} \sum_{i=1}^{m} h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$(h_\theta(x^{(1)}) - y^{(1)}) x^{(1)} + (h_\theta(x^{(2)}) - y^{(2)}) x^{(2)} + (h_\theta(x^{(3)}) - y^{(3)}) x^{(3)}$$

$$X = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \end{bmatrix}$$

If $u(j) = 2v(j) + 5w(j)$ (for all j)

$u = 2V + 5W$ ↙ vectorization

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

CODE $\Rightarrow$ J = costFunction J(X, y, theata)

m = length(y)

predictions = X * theta $\Rightarrow \mathbb{R}^{m+1 \times 1}$

$\Downarrow$

$\mathbb{R}^{m \times n+1}$

$\Downarrow$

$\mathbb{R}^{m \times 1}$

sqrErrors = (predictions - y).^2;

$\Downarrow$ $\mathbb{R}^{m \times 1}$ $\Downarrow$ $\mathbb{R}^{m \times 1}$

$$\begin{bmatrix} h_\theta(x^{(1)}) \\ h_\theta(x^{(2)}) \\ \vdots \\ h_\theta(x^{(m)}) \end{bmatrix}$$

$$\begin{bmatrix} (h_\theta(x^{(1)}) - y^{(1)})^2 \\ (h_\theta(x^{(2)}) - y^{(2)})^2 \\ \vdots \\ (h_\theta(x^{(m)}) - y^{(m)})^2 \end{bmatrix} \Rightarrow \mathbb{R}^{m \times 1}$$

$$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$m \times 1$

$$\text{prediction} - y = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}$$

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

1 iteration of gradient descent

$$X = \begin{bmatrix} x_0^1 & x_1^1 \\ x_0^2 & x_1^2 \\ x_0^3 & x_1^3 \\ \vdots & \vdots \\ x_0^m & x_1^m \end{bmatrix}_{m \times 2} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}_{2 \times 1} \quad y = \begin{bmatrix} y^1 \\ y^2 \\ y^3 \\ \vdots \\ y^m \end{bmatrix}_{m \times 1}$$

$$\underbrace{\text{prediction} - y}_{\| \; X * \text{theta}} = \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}_{m \times 1}$$

$$X^T(\text{prediction} - y) = \begin{bmatrix} x_0^1 & x_0^2 & \cdots & x_0^m \\ x_1^1 & x_1^2 & \cdots & x_1^m \end{bmatrix}_{2 \times m} \times \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \\ h_\theta(x^{(2)}) - y^{(2)} \\ \vdots \\ h_\theta(x^{(m)}) - y^{(m)} \end{bmatrix}_{m \times 1}$$

$$= \begin{bmatrix} h_\theta(x^{(1)}) - y^{(1)} \cdot x_0^{(1)} + h_\theta(x^{(2)}) - y^{(2)} \cdot x_0^{(2)} + \cdots + h_\theta(x^{(m)}) - y^{(m)} \cdot x_0^{(m)} \\ h_\theta(x^{(1)}) - y^{(1)} \cdot x_1^{(1)} + \cdots + h_\theta(x^{(m)}) - y^{(m)} \cdot x_1^{(m)} \end{bmatrix}$$

$$\theta := \theta - \alpha * (X^T(\text{predictions} - y))$$

Feature Normalization

$$x_i \leftarrow \frac{x_i - \mu_i}{\sigma_i} \qquad \mu_i \rightarrow \text{mean}_i \qquad \sigma_i \rightarrow \text{std}_i$$

$$\mu = [\mu_1 \ \mu_2 \ \mu_3]_{1\times 3}$$

$$X = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} \\ \vdots & \vdots & \vdots \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} \end{bmatrix}_{m\times 3}$$

$$X - \mu = \begin{bmatrix} x_1^{(1)} - \mu_1 & x_2^{(1)} - \mu_2 & x_3^{(1)} - \mu_3 \\ \vdots & \vdots & \vdots \\ x_1^{(m)} - \mu_1 & x_2^{(m)} - \mu_2 & x_3^{(m)} - \mu_3 \end{bmatrix}$$

$$\sigma = [\sigma_1 \ \sigma_2 \ \sigma_3]$$

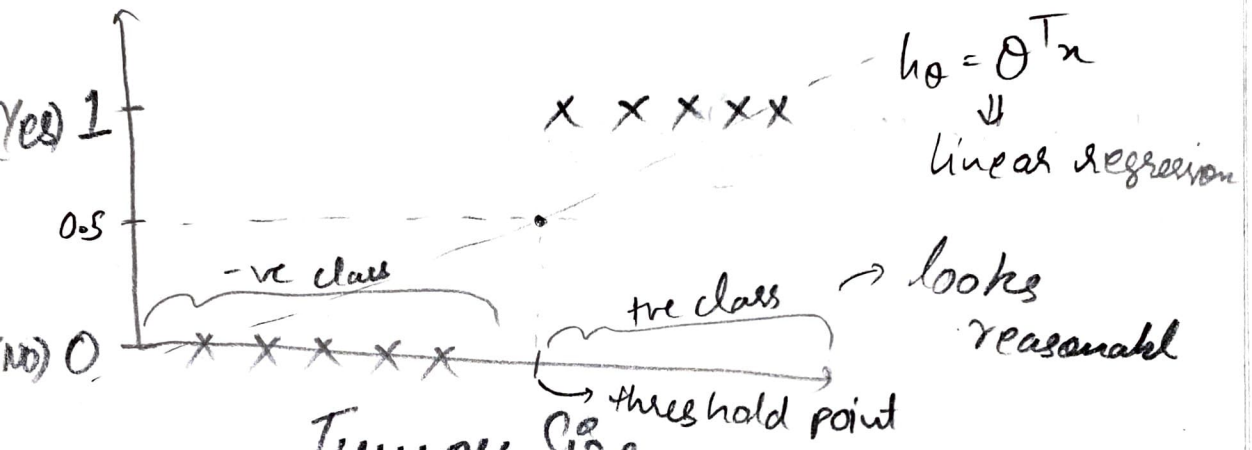
$$(X - \mu)./\sigma = \begin{bmatrix} \frac{x_1^{(1)} - \mu_1}{\sigma_1} & \frac{x_2^{(1)} - \mu_2}{\sigma_2} & \frac{x_3^{(1)} - \mu_3}{\sigma_3} \\ \vdots & \vdots & \vdots \\ \frac{x_1^{(m)} - \mu_1}{\sigma_1} & \frac{x_2^{(m)} - \mu_2}{\sigma_2} & \frac{x_3^{(m)} - \mu_3}{\sigma_3} \end{bmatrix}$$

Classification ⇒ Email ⇒ Spam / Not Spam
Online Transaction ⇒ Fraudulent (Y/N)
Tumor ⇒ Malignant / Benign?

Two class classification

$y \in \{0, 1\}$ → 0 ⇒ "Negative Class" (Eg, benign)
→ 1 ⇒ "Positive Class" (Eg malignant)

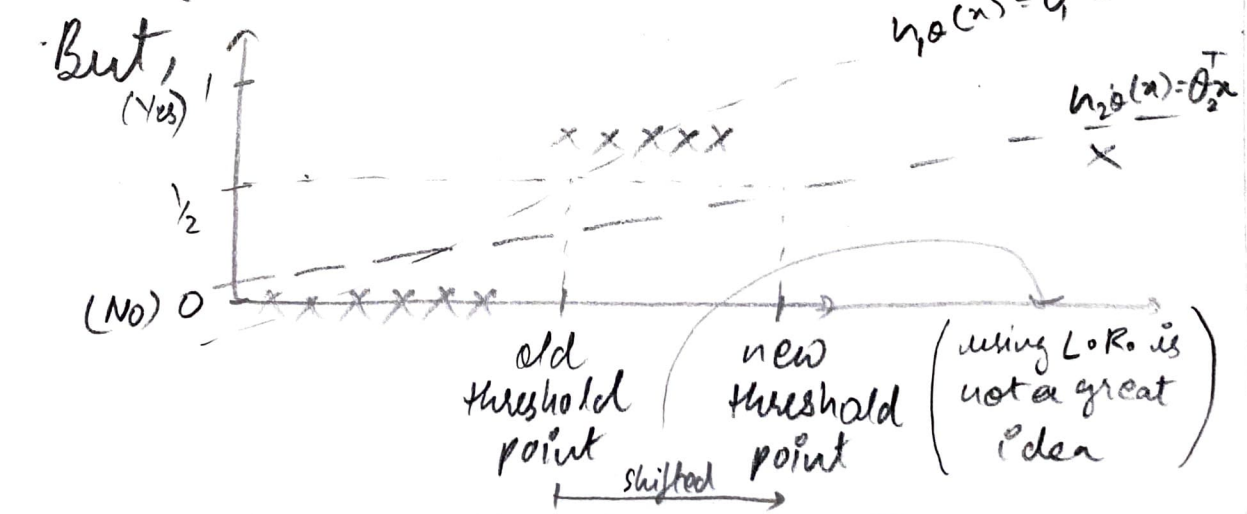
$y \in \{0, 1, 2, 3, 4, 5\}$ ⇒ Multi-class classification



Threshold classifier output $h_\theta(x)$ at 0.5:

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

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



for Classification ⇒ $y = 0$ or 1

$h_\theta(x)$ can be > 1 or < 0 ⇒ ∴, Linear Regression is not a great learning Algo for Classification problem

⇒ Logistic Regression ⇒ $0 \leq h_\theta(x) \leq 1$

↳ this is a better learning algo. for classification problems

---

Logistic Regression Model

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

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

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

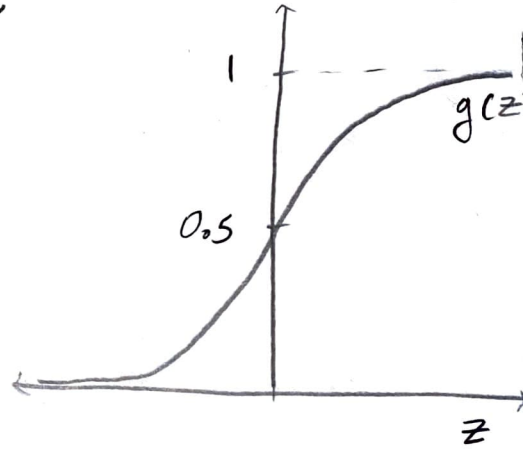
$z \in \mathbb{R}$

$g \in (0, 1)$

Sigmoid func^n OR Logistic func^n

$0 < g(z) < 1$

* we will use diff^n algo to fit parameters θ



Interpretation of Hypothesis Output

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

Eg) If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumor size} \end{bmatrix}$

$h_\theta(x) = 0.7$ → probability of $y = 1$ for given x is 0.7

↳ Tell patient that 70% chance of tumor being malignant

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

$y = \begin{matrix} \nearrow 0 \\ \searrow 1 \end{matrix}$ } Possible outcomes

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

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

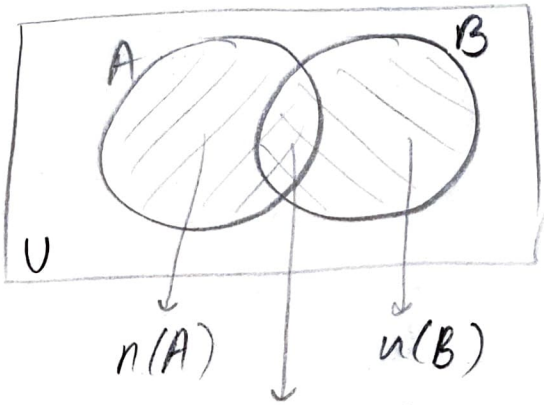
Tell the patient that 30% chance of tumor

OR

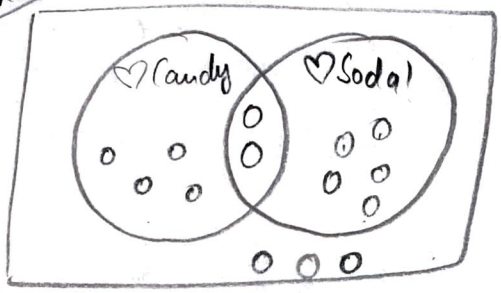
Probability of $y = 0$ for given x is 0.3

Conditional Probability

$$P(A|B) = \frac{n(A \cap B)}{n(B)} = \frac{n(A \cap B)/n(S)}{n(B)/n(S)} = \frac{P(A \cap B)}{P(S)}$$



Example



In Statland people like to eat candy & soda

	Loves Candy	Doesn't love Candy	
Loves Soda	2	5	Total loves soda = 7
Doesn't love Soda	4	3	
	Total Loves Candy = 4+2=6	8	→ # Doesn't love candy

$P(\text{Loves Candy}) = \frac{6}{14} = \frac{3}{7}$

$P(\text{Loves Soda}) = \frac{7}{14} = \frac{1}{2}$

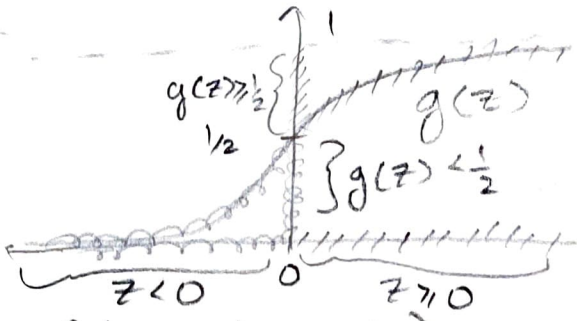
$P(\text{Loves Candy} | \text{loves Soda}) = \frac{2}{7}$

$P(\text{Love Soda} | \text{Doesn't love candy}) = \frac{5}{8}$

Decision Boundary

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

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



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

Suppose,

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

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

$\therefore$ $g(z) \geq 0.5$ when $z \geq 0$

$\therefore$ $h_\theta(x) = g(\theta^T x) \geq 0.5$ whenever $\theta^T x \geq 0$

Ily, $\because$ $g(z) < 0.5$ when $z < 0$

$\therefore$ $h_\theta(x) = g(\theta^T x) < 0.5$ whenever $\theta^T x < 0$

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

Let $\theta_0 = -3$; $\theta_1 = 1$, $\theta_2 = 1$

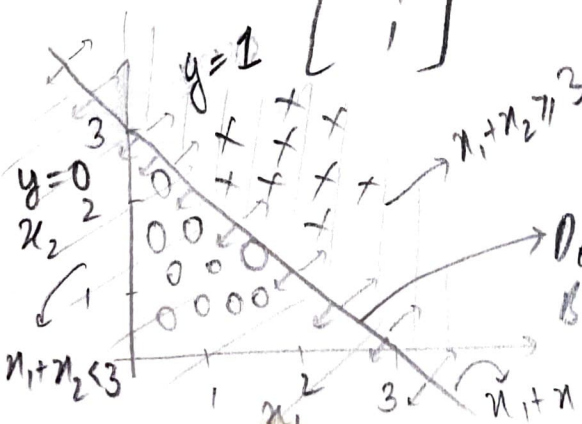
$\therefore$ $\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$

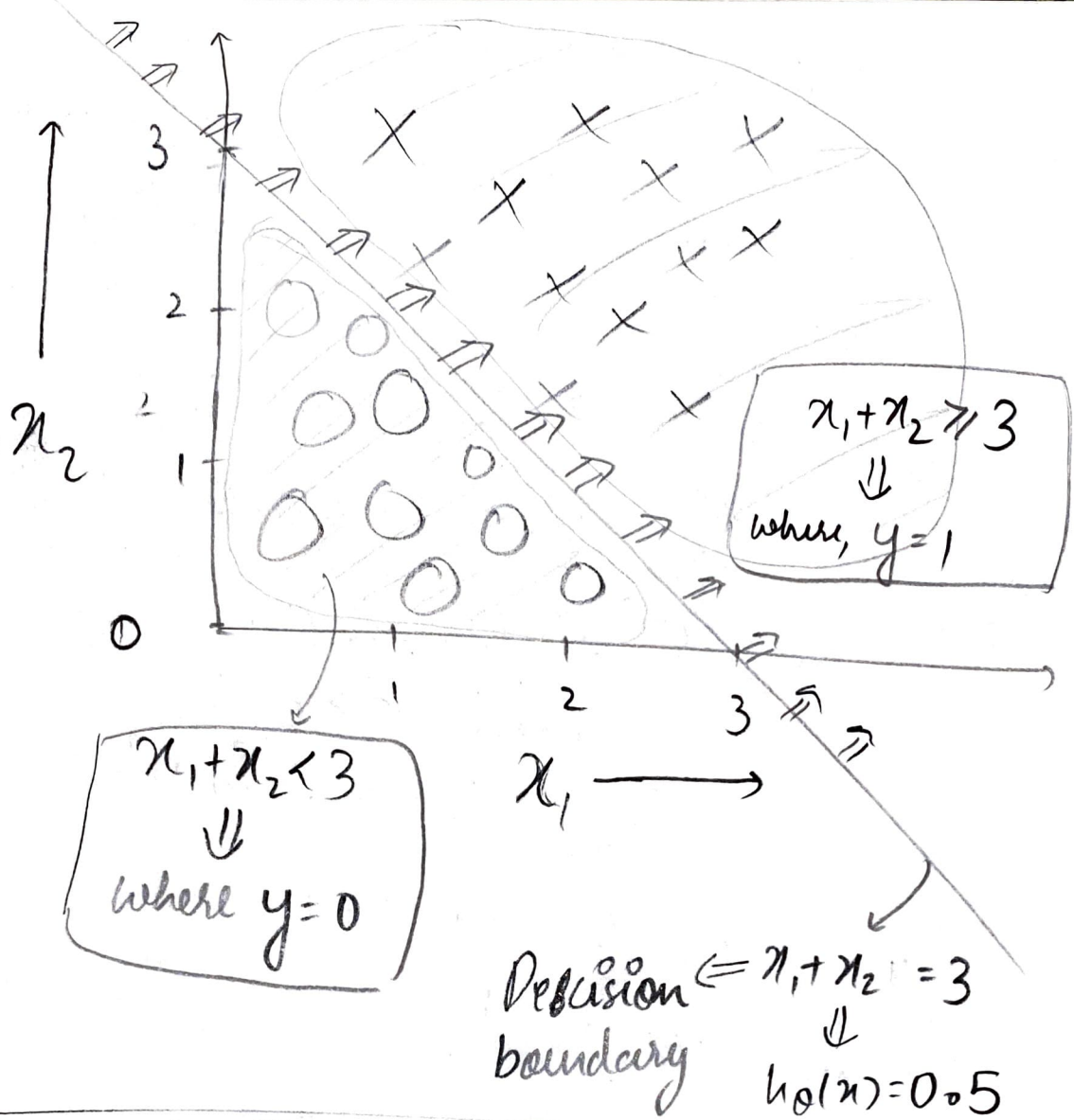
Predict $\Rightarrow y=1$ if $\underbrace{-3 + x_1 + x_2}_{\theta^T x} \geq 0$

$\Rightarrow x_1 + x_2 \geq 3$

$\Rightarrow y=0$ if $-3 + x_1 + x_2 < 0$

$\Rightarrow x_1 + x_2 < 3$





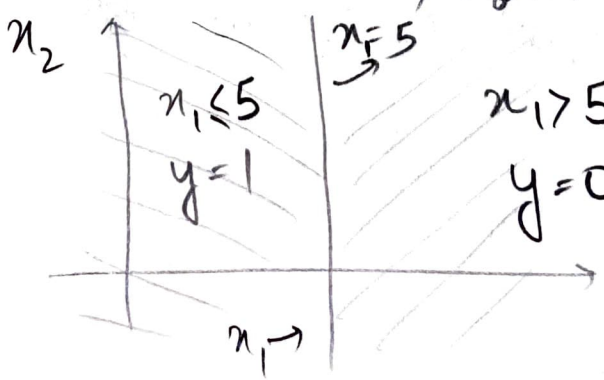
If $\theta = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix} \Rightarrow h_\theta(x) = g(5 - x_1)$

$h_\theta(x) \geq 0.5 \Rightarrow \theta^T x = 5 - x_1 \geq 0$

$\boxed{x_1 \leq 5}$

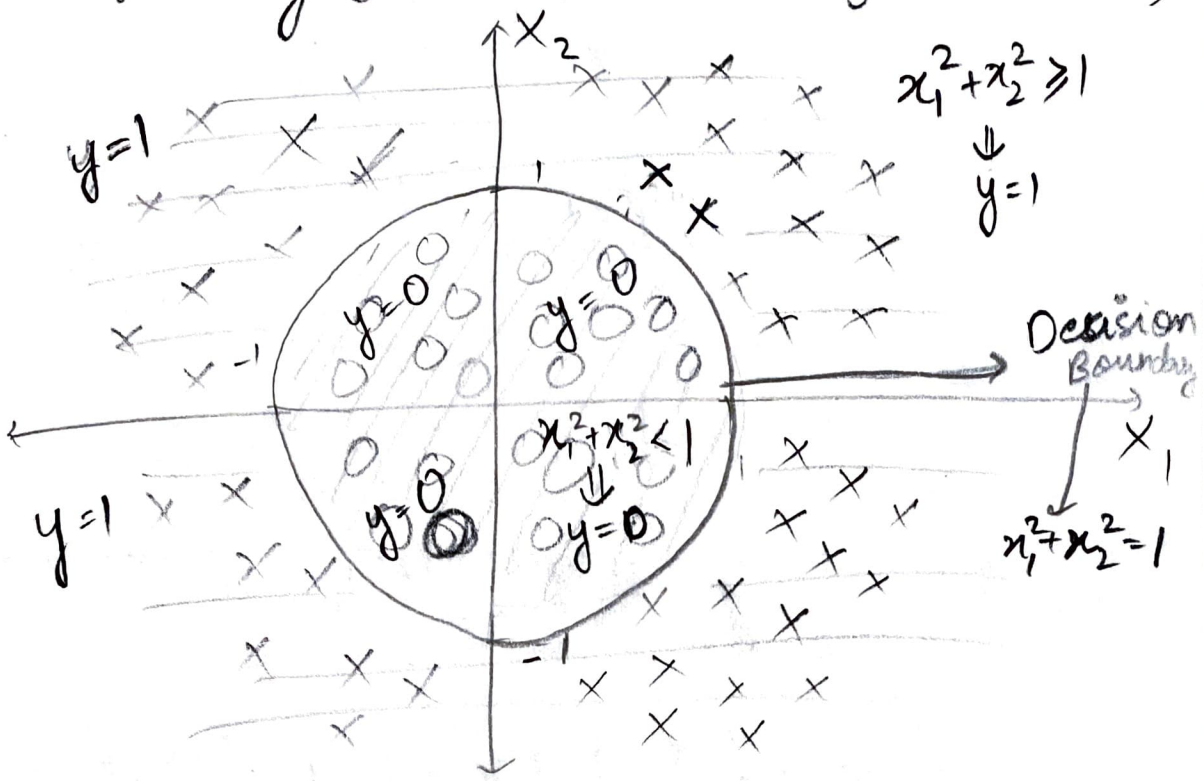
This is, $h_\theta(x) < 0.5 \Rightarrow 5 - x_1 < 0$

$\boxed{x_1 > 5}$



Non-linear decision boundaries

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



If $\theta_0 = -1, \theta_1 = 0, \theta_2 = 0, \theta_3 = 1, \theta_4 = 1$

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

Predict:

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

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

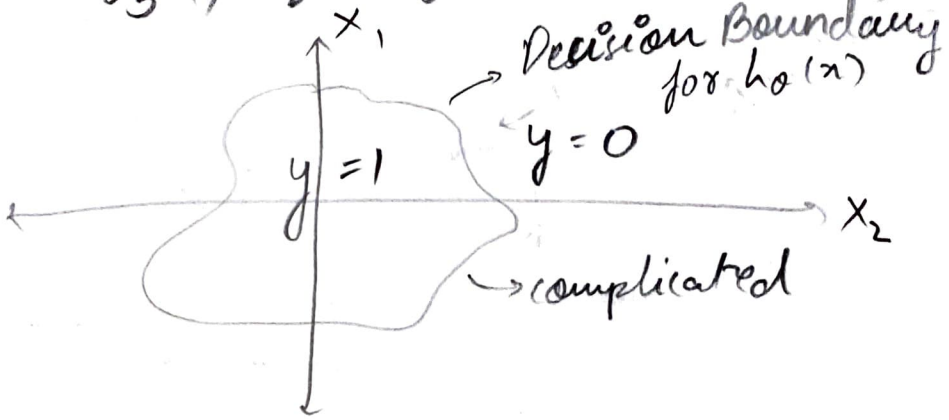
$x_1^2 + x_2^2 = 1$ is Eq^n of circle $((0,0); r = 1)$

② $y = 0 \Rightarrow -1 + x_1^2 + x_2^2 < 0$

$x_1^2 + x_2^2 < 1$

⊛ Decision Boundary is the property of parameters & $h_\theta(x)$ and not of training set

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1 x_2 + \cdots)$$



Training Set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

m examples: $x \in \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}_{n+1}$ $\boxed{x_0 = 1,\ y \in \{0, 1\}}$

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

Cost func^n in Linear Regression $\Rightarrow J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$

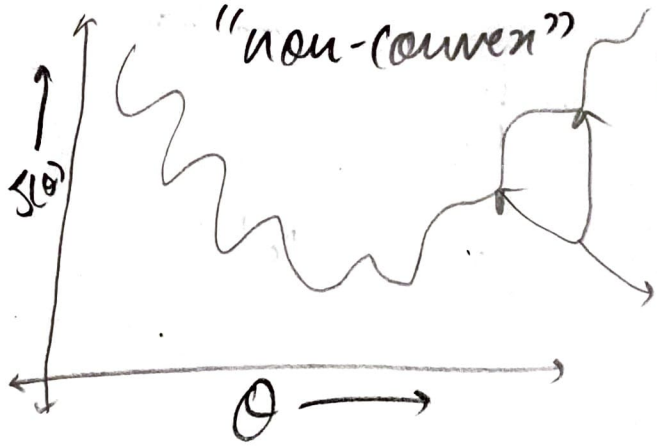
let, $\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} (h_\theta(x^{(i)}) - y^{(i)})^2$

OR

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

[for linear Regression]

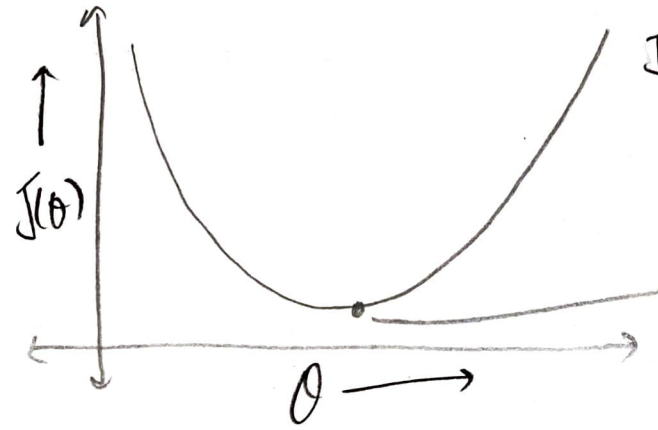
But for logistic Regression the above cost func^{n} won't work as it will return a non-convex func^{n}



J(θ) where $h_\theta(x) = g(\theta^T x)$

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

there are many minimals)



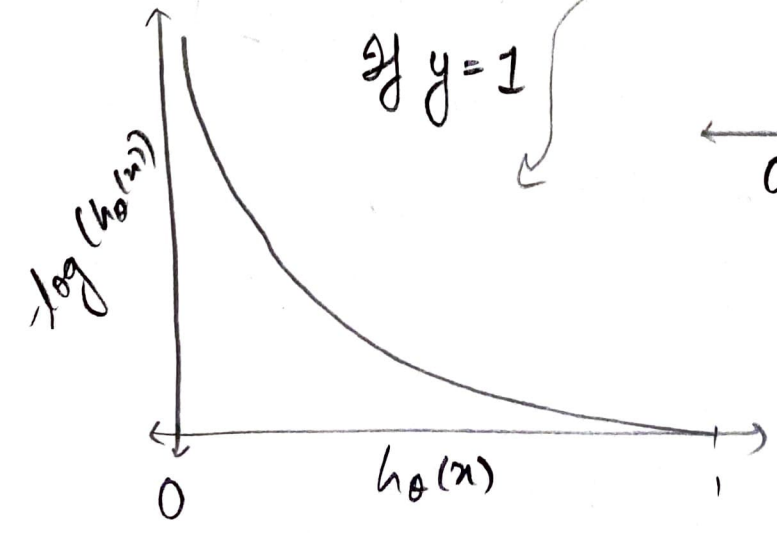
J(θ) where $h_\theta(x) = \theta^T x$

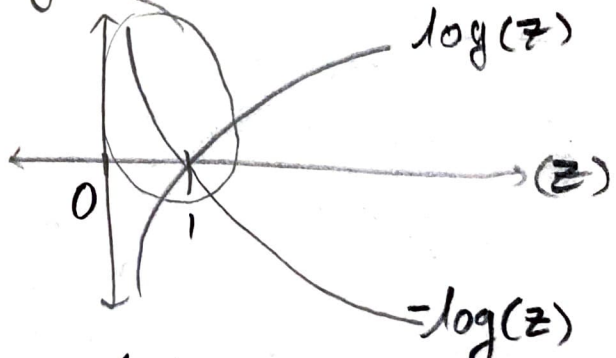
there is only 1 minima

Logistic Regression Cost function

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

If $y = 1$

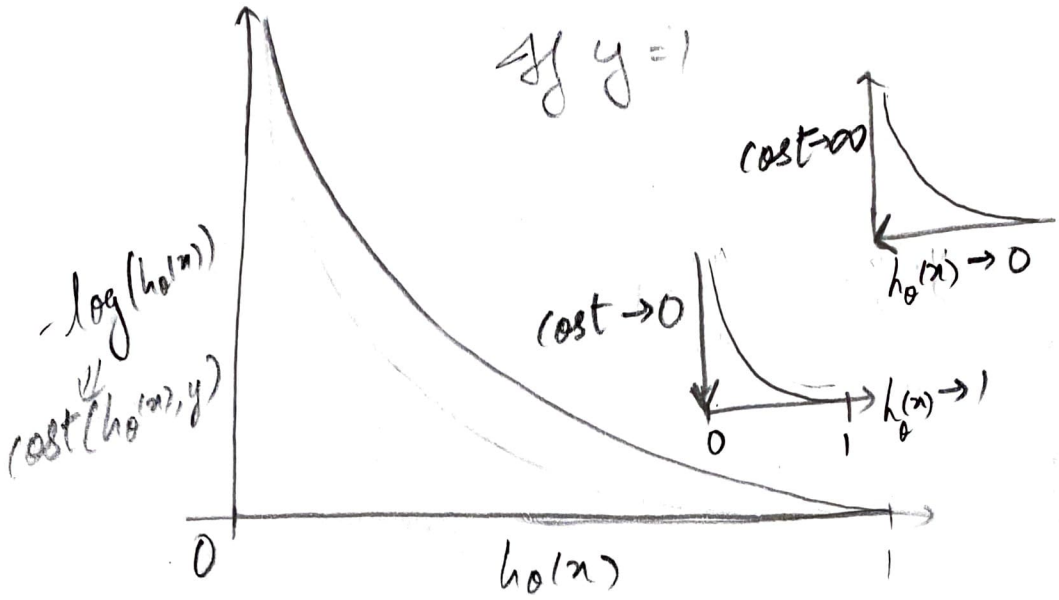




Cost = 0 if $y = 1, h_\theta(x) = 1$

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

Cost $\to \infty$

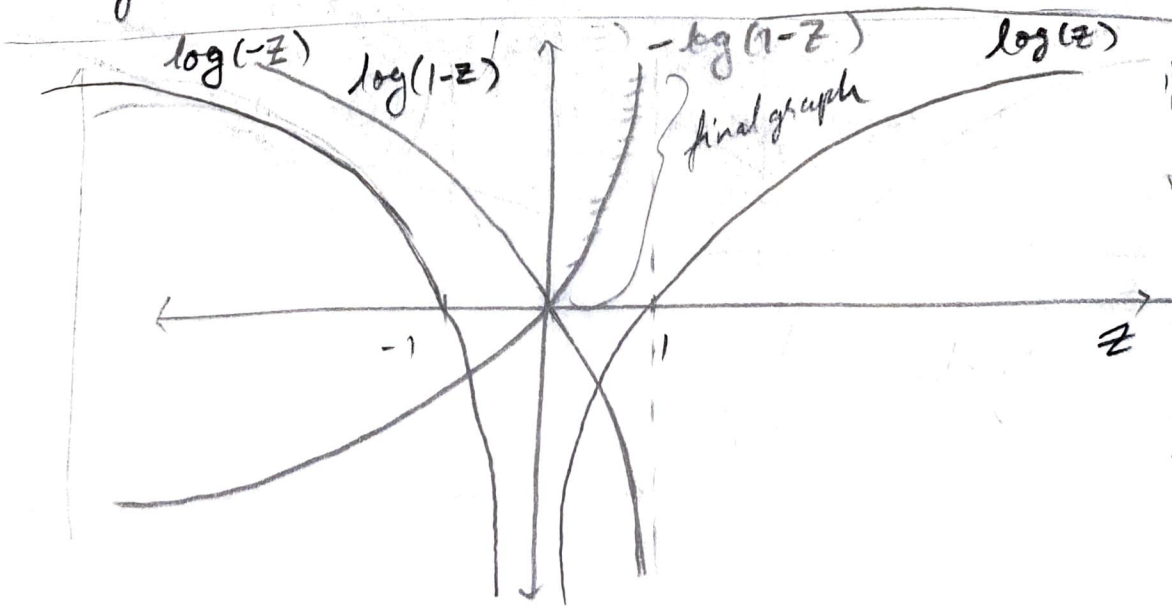


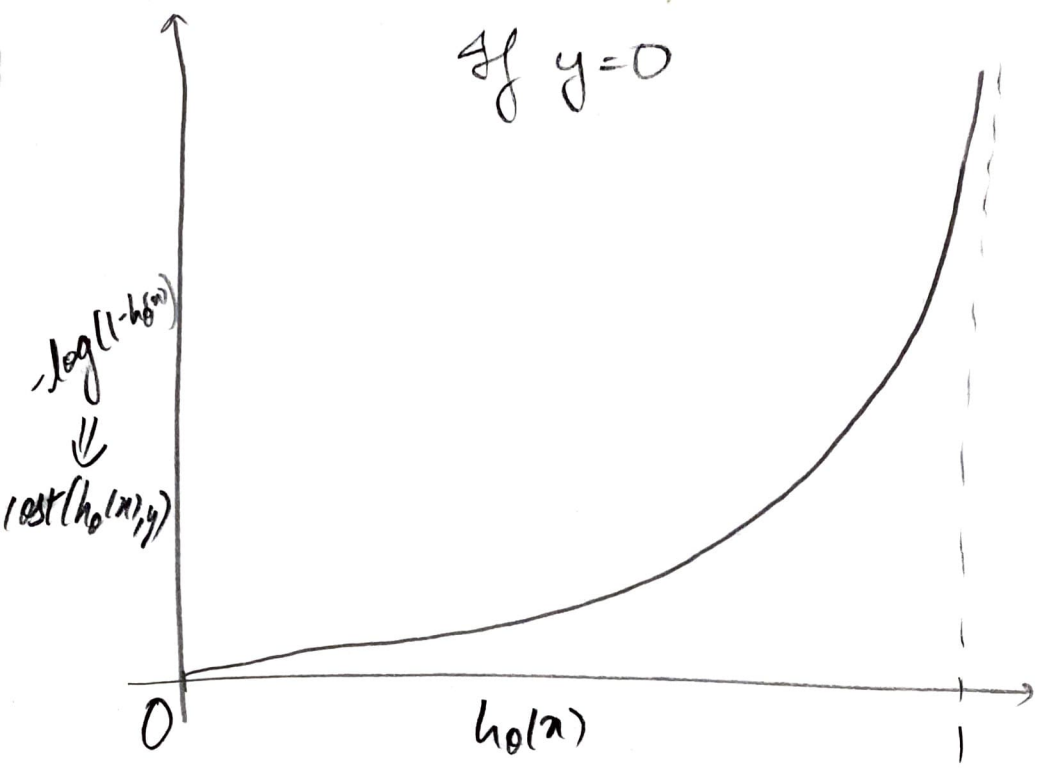
Cost = 0 if $y = 1, h_\theta(x) = 1$

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

$\text{Cost} \to \infty$

Captures intuition that if $h_\theta(x) = 0$, (i.e. predict $P(y = 1 | x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost





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

But as, $h_\theta(x) \to 1$

$\log y \to \infty$

Captures intuition that if $h_\theta(x) = 0$, (i.e. predict $P(y=0|x;\theta) = 1$), but $y = 0$, then we'll penalize learning algorithm by a very large cost

Logistic Regression cost function:

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

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

Note $\Rightarrow$ $y = 0$ or 1 always

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

If $y = 1 \Rightarrow Cost(h_\theta(x), y) = -\log(h_\theta(x)) - 0$

$\hookrightarrow$ $1 - y = 0$

If $y = 0 \Rightarrow Cost(h_\theta(x), y) = 0 - (1-0) \log(1 - h_\theta(x))$

$\hookrightarrow$ $y = 0$

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

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

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

To fit parameters: $\min_\theta J(\theta)$

To make a prediction: Output $h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$

given new x

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

Gradient descent for Logistic Regression.

We need $\min_\theta J(\theta)$:

Repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j}(J(\theta)) \quad (\text{for } j=0,1,2\cdots,n)$$

} (simultaneously update all θ_j)

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

$$\left. \cdots -(1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right)$$

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

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

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

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

Repeat {

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

(for $j=0,1,2,\cdots n$)

} (simultaneous update all θ_j)

★ Looks identical to Linear regression even though hypothesis func^n $\Rightarrow$ $h_\theta(x) = \theta^T x$ & $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

$\theta = \begin{bmatrix}\theta_0 \\ \theta_1 \\ \vdots \\ \theta_n\end{bmatrix}_{n+1}$ $x = \begin{bmatrix}x_0 \\ x_1 \\ \vdots \\ x_n\end{bmatrix}_{n+1}$

$h_\theta(x) = \theta^T x$: Linear Regression

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

Vectorization of Gradient Descent for Logistic Regression. ($\theta := \theta - \alpha\delta$)

$$X = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ x_0^{(2)} & & & & \\ \vdots & & & & \\ x_0^{(m)} & x_1^{(m)} & \cdots & & x_n^{(m)} \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \Rightarrow \mathbb{R}^{n+1 \times 1}$$

$$\hookrightarrow \mathbb{R}^{m \times n+1}$$

$$h_\theta(x) = \begin{bmatrix} h_\theta(x^{(1)}) \\ h_\theta(x^{(2)}) \\ \vdots \\ h_\theta(x^{(m)}) \end{bmatrix} \Rightarrow \mathbb{R}^m \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \Rightarrow \mathbb{R}^m$$

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

$$h_\theta(x) = X\theta = \begin{bmatrix} \theta_0 x_0^{(1)} & \theta_1 x_1^{(1)} & \cdots & \theta_n x_n^{(1)} \\ & & & \\ \theta_0 x_0^{(m)} & & & \theta_n x_n^{(m)} \end{bmatrix}$$

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

Code ⇒

```
theta = theta - (α/m) * X' * (X*theta - y)
```

Mathematical ⇒

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

Advanced Optimisation for Logistic Regression.

Given θ, we have code that can compute
→ $J(\theta)$
→ $\frac{\partial}{\partial \theta_j} J(\theta)$ (for $j = 0, 1, 2 \cdots, n$).

Optimization Algorithms:
→ Gradient descent (One we are using)
→ Conjugate gradient
→ BFGS
→ L-BFGS

Advantages:
→ No need to manually pick α
→ Often faster than gradient descent

Disadvantage: More Complex

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

for $\min_{\theta} J(\theta) \Rightarrow \theta_1 = 5, \theta_2 = 5$

$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$ $\left\{ \frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5) \right.$

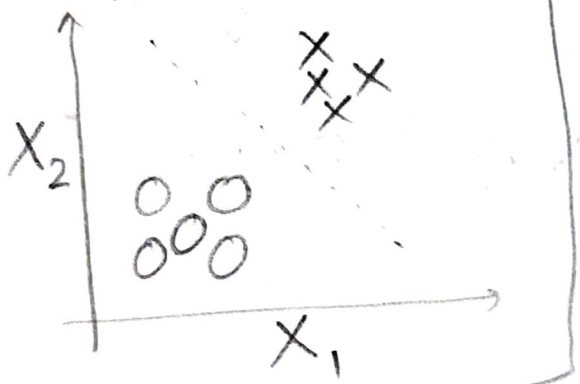
Logistic Regression for Multiclass classification

① Email foldering / tagging ⇒ Work, Friends, Family, Hobby
$y=1$ $y=2$ $y=3$ $y=4$

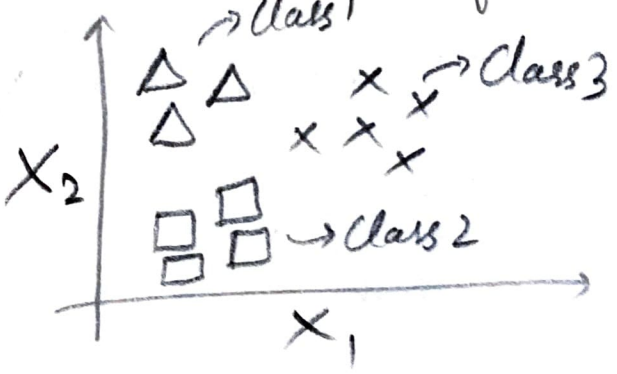
② Medical diagrams ⇒ Not ill, Cold, Flu
$y=1$ $y=2$ $y=3$

③ Weather ⇒ Sunny, Cloudy, Rain, Snow
$y=1$ $y=2$ $y=3$ $y=4$

Binary Classification

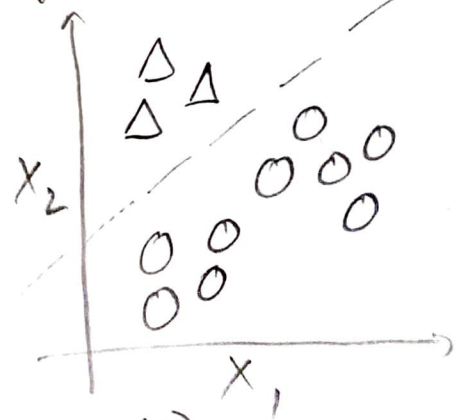


Multi-class Classification



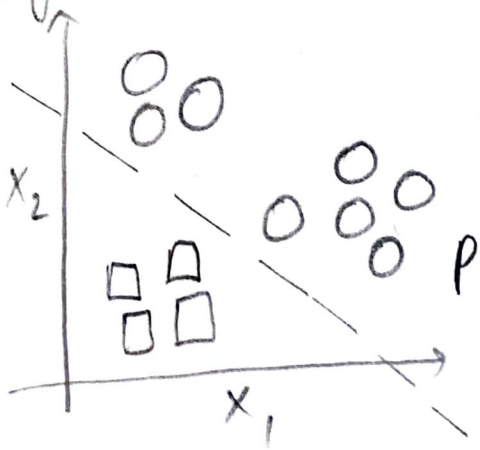
One v/s All (or, one v/s rest)

① for Class 1



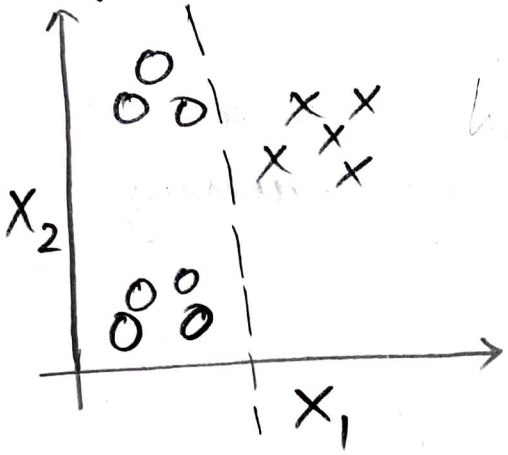
$h_\theta^{(1)}(x) = P(y=1|x;\theta)$

② for Class 2



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

③ for Class 3



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

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

$(i = 1, 2, 3)$

One v/s all

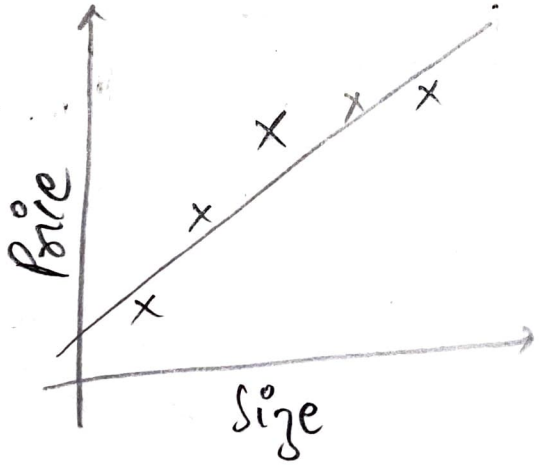
Train a logistic regression $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y=i$.

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

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

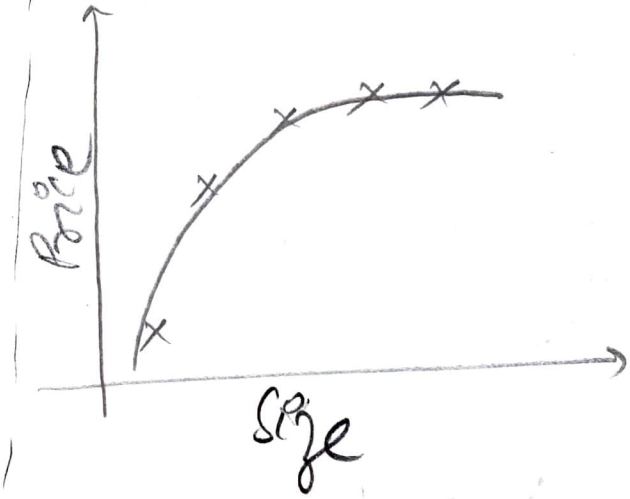
⇒ The problem of overfitting

Eg) Linear Regression (housing Prices)



$\theta_0 + \theta_1 x_1$

Underfit, High bias



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

Just Right

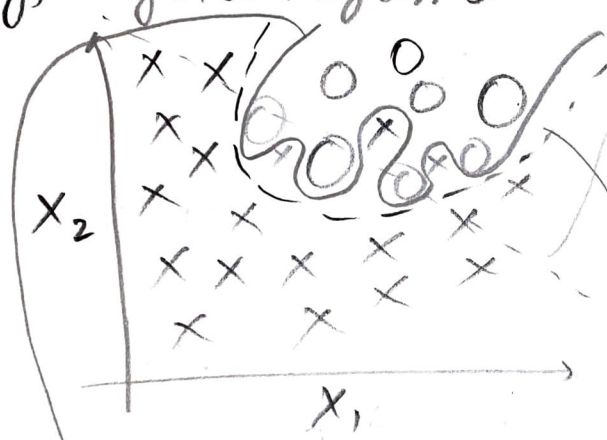
→ not a good hypo. func^n even though it is passing through all the training examples.

"Overfit" & "High Variance"

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

Overfitting ⇒ If we have too many features, the learned hypothesis may fit the training

eg) Logistic Regression



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

(Underfit)

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

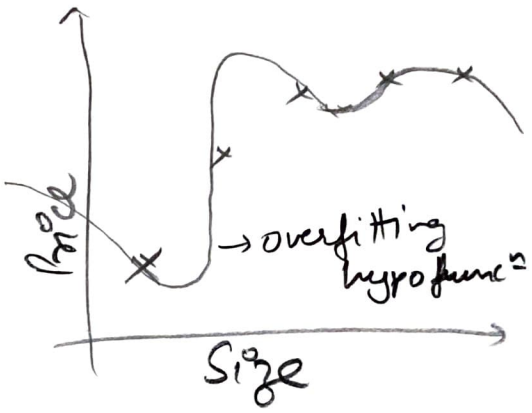
(Just right)

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \cdots)$

(Overfit)

Addressing Overfitting:

x_1 = size
x_2 = # bedrooms
x_3 = # floors
⋮
x_{100} = Area

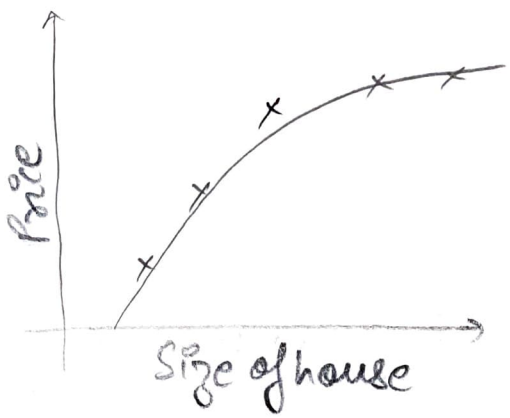


↳ too many features with less training data will make hypo func° overfitting

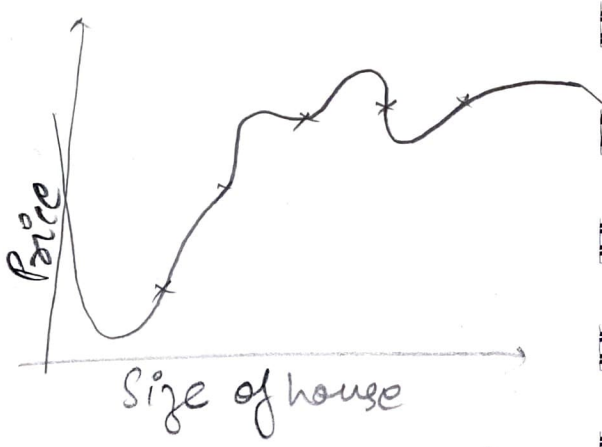
① Reducing # features.
↳ Manually select which features to keep
↳ Model selection algorithm

② Regularizations
↳ keep all the features, 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

Intuition



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



$\theta_0 + \theta_1 x_0 + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$.

Suppose we penalize and make θ_3 & θ_4 really small

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

↳ $\theta_3 \approx 0$ & $\theta_4 \approx 0 \Rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + 0 + 0$

Regularization

Small values for parameters $\theta_1, \theta_2 \cdots \theta_n$

↳ Simpler hypothesis

↳ Less prone to overfitting

eg) Housing:

↳ Features $\Rightarrow x_1, x_2, \cdots x_{100}$

↳ Parameters $\Rightarrow \theta_0, \theta_1, \theta_2, \cdots, \theta_{100}$

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

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

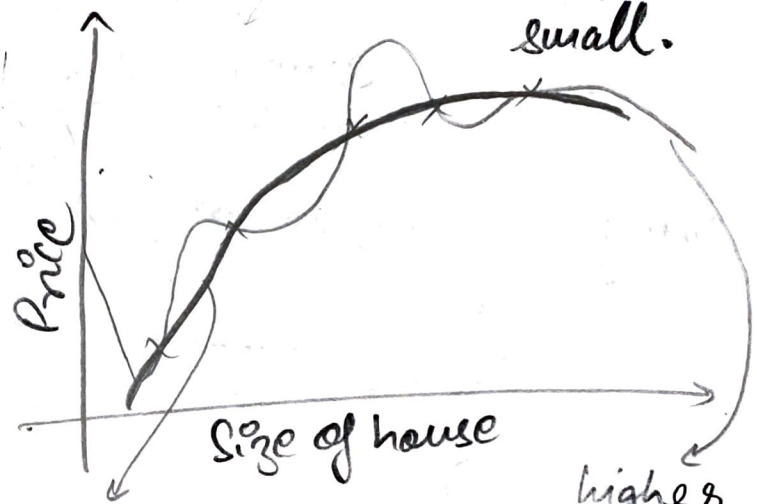
Regularization parameter

Regularization term

$\min_\theta J(\theta)$

this term helps fitting training data well.

this term helps in keeping the parameters small.



higher order hypo. func^n with regularization (smoother and better)

higher order hypo. func^n without regularization

If λ is very large,

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

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

for very large $\lambda \Rightarrow \theta_1 \approx 0, \theta_2 \approx 0 \cdots, \theta_n \approx 0$

$\therefore h_\theta(x) = \theta_0 \Rightarrow$ constant line $\parallel$ to x axis

↙ underfit or 'high bias'

Regularization linear regression

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

$\min_\theta J(\theta)$

Grad Desc.

Repeat {

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

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

}

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

$\left[1 - \alpha\frac{\lambda}{m} < 1\right]$ ↳ 0.99 Just little bit less than 1.

$$\theta_j := \theta_j\underbrace{\left(1 - \alpha\frac{\lambda}{m}\right)}_{0.99 \approx < 1} - \underbrace{\frac{\alpha}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}}_{\text{Original Grad-desc. formula without regularization}}$$

$0.99\,\theta_j$

Normal Eq: (with Regularization)

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix}_{m \times (n+1)} \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \Rightarrow \mathbb{R}^m$$

$\min_\theta J(\theta)$

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

Eg) $n = 2 \Rightarrow \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Non-invertibility (optional/advanced)

Suppose $m \leq n$

m: #examples, n: #features

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

$(X^T X)^{-1}$: non-invertible / singular

→ pinv: pseudo inverse

If $\lambda > 0$,

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \right)^{-1} X^T y$$

Regularized Logistic Regression

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

↓ new cost func^n with Regularization term

Grad Desc.

Repeat {

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

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

$\frac{\partial}{\partial \theta_j} J(\theta)$ ← new cost func^n

}

Adv. Optimisation (for new cost func^n)

function [jVal, gradient] = costFunction(theta)

jVal = [code to compute $J(\theta)$]; → new cost func^n

gradient(1) = [code to compute $\frac{\partial}{\partial \theta_0} J(\theta)$];

gradient(2) = [code to compute $\frac{\partial}{\partial \theta_1} J(\theta)$];

⋮

gradient(n+1) = [code to compute $\frac{\partial}{\partial \theta_n} J(\theta)$];

→ fminunc(@costFunction, ...)

↳ for applying