Day - 1

* What is Machine Learning ?

① Field of Programming that gives machine the ability to learn without being explicitly Programmed

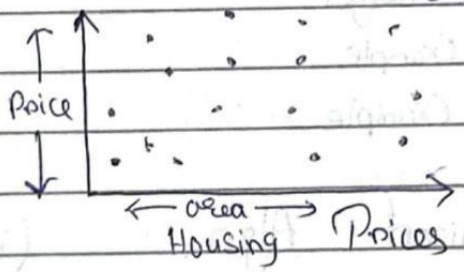
② A Computer Program is said to learn from exp E w.r.t some task T and Perf. Parameter P. If it's Performance on T as measured by P improved with experience E.

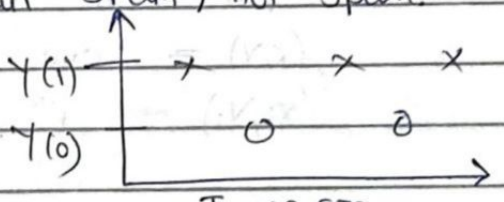
Supervised learning

Here the available dataset is such that it consists of correct answers. and task of this algo to Produce much more right / correct answers.

Regression :- Predicting continuous valued o/p
e.x. House Prices

Classification :- Predicting discrete valued o/p
e.x. Email Spam / not Spam.





Tumor classification

Unsupervised Learning

This type allows us to approach toward solution with very little or no idea about the solution. We Can derive Structures from data where we don't necessarily know effect of Variable.

NOTE :-
Unsupervised learning don't have f/b on Pred Result.

Clustering :- Collection of 1000000 balls, we have to find out way to cluster them on the basis of various Parameter.

Non-clustering = Finding structure in Chaotic env.

* Linear Regression with one Variable

⇒ Model Representation

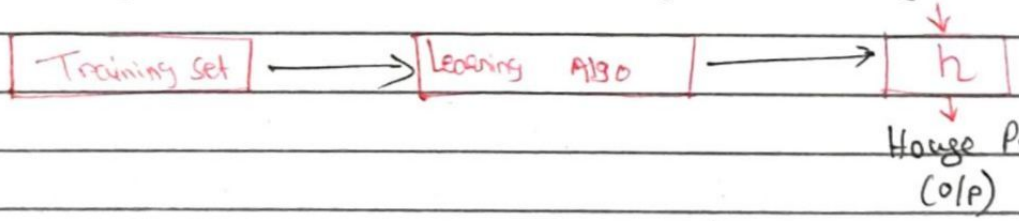
Training set of Housing Prices

Size in feet^2 (x)	Price (Y)
2104	460
1416	232
1534	315

Notations :-

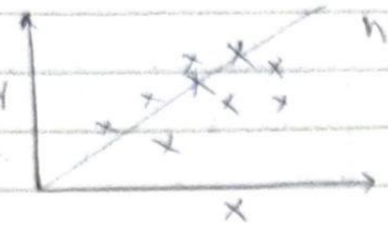
- m = No. of training examples
- X's = i/p Variable / feature
- Y's = o/p Variable / Target
- (x, Y) = one training example
- (x_i, Y_i) = i^{th} training example

⇒ working on Supervised Algo.



Hypothesis Matches X's with Y's.

⇒ Representation of hypothesis.



Here hypothesis is fitting linearly, so it's called linear Regression with one Variable.

⇒ Cost function

* Here $h(x_i)$ Represents the Predicted response Value for i^{th} obs.
* θ_1 — slope ; θ_0 = Y-intercept

We want minimum Value of θ_1 and θ_0 So we will implement Cost

i.e.

Squared cost fun ← $J(\theta_0, \theta_1) = \frac{1}{2m}\left(h_\theta(x_i) - y^{(i)}\right)^2$

$h_\theta(x_i)$ ← Predicted O/P ; $y^{(i)}$ → actual O/P

here,

$$\theta_1 = \frac{S_{xy}}{S_{xx}} = \frac{\sum_{i=1}^{m}(x_i-\bar{x})(y_i-\bar{y})}{\sum_{i=1}^{m}(x_i-\bar{x})^2} = \frac{\sum_{i=1}^{m} x_i y_i - m\bar{x}\bar{y}}{\sum_{i=1}^{m} x_i^2 - n(\bar{x})^2}$$

$$\theta_0 = \bar{y} - \theta_1 \bar{x}$$

⇒ Gradient Descent

Repeat until convergence {

$$\text{temp}\_\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h(x_i) - y^{(i)}\right)$$

$$\text{temp}\_\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h(x_i) - y^{(i)}\right).x^{(i)}$$

}

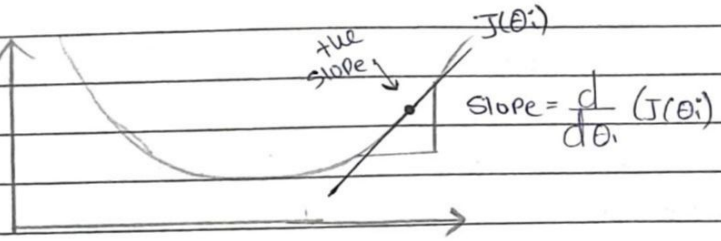
$\theta_0 = \text{temp}\_\theta_0$ and $\theta_1 = \text{temp}\_\theta_1$

⇒ Gradient Descent intution

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

- α: learning Rate / step size
- $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$: Derivative

Case I:-



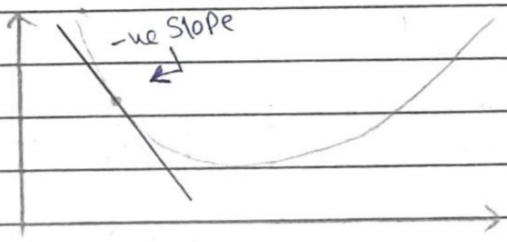
Here to Reach minima theta should decreage so,

$\theta_j := \theta_j - \alpha \underbrace{\frac{d}{\partial \theta_j} J(\theta_i)}_{\geq 0}$

derivative Part must be greater that zero / Positive

i.e $\theta_j := \theta_j - \alpha$ (Positive Num) [θ_j will ↓]

Case II:-

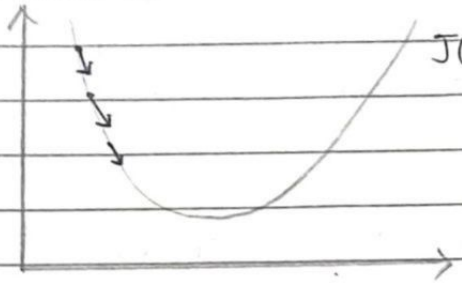


$\theta_j := \theta_j - \underbrace{\alpha \frac{\partial}{\partial \theta_j} J(\theta_j)}_{\leq 0}$

here θ_j have to decreage to Reach ~~Glob~~ minima.

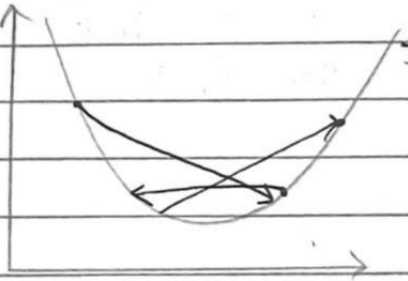
⇒ What α Does?

Case 1 :- α is very Small



If α is very Small Gread clescent will be Slow

Case 2 :- α is very large



If α is too large Gread descent will overshoot local mining

* Linear Regression with Multiple Variables

Sample Data:-

Size (feet^2) X_1	No. of bedroom X_2	No. of floor X_3	Age of home X_4	Price (Y)

Notation

n = No. of features

$x^{(i)}$ = i^{th} training ex

$x_j^{(i)}$ = Value of feature j in i^{th} training ex

Hypothesis:-

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

here

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

So,

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

$= [\theta_0\ \theta_1 \dots \theta_n]$ — Parameter Vect

$\begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$ — Feature vect.

* Gradient descent for LR multiple Variable

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

Parameter $= \theta_0, \theta_1 \dots \theta_n$

Cost fun =

$$J(\theta_0, \theta_1, \theta_2 \dots \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Gradient Descent

Repeat {

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

}

for $n = 1$	for $n > 1$
Repeat {	Repeat {
$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$	$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x_j^i$
$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i) x^i$	}
}	

let $n = 2$ as $x_0 = 1$

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

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

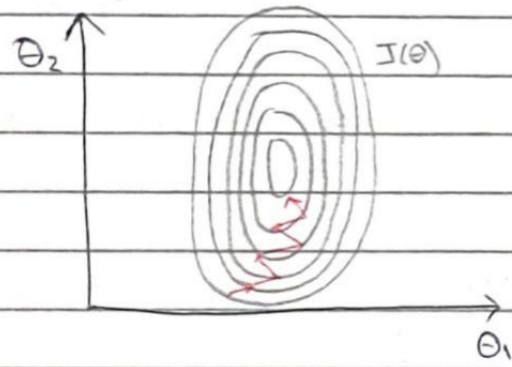
$$\rightarrow \quad \theta_2 := \theta_2 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^i) - y^i)\, x_2^{(i)}$$

* Gradient Descent Feature Scaling

Feature Scaling :- Making sure that features are on similar scale

E.g. x_1 = Size of House $(0 - 2000)$ feet^2

x_2 = No. of bedrooms $(1-5)$

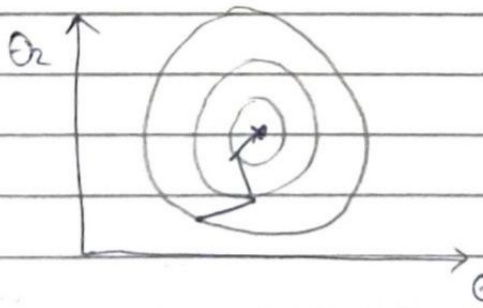


$\Longrightarrow$ This will take long time to Reach global minima

So, we Scale feature

i.e $x_1 = \dfrac{\text{Size (feet}^2\text{)}}{2000} \Rightarrow 0 \le x_1 \le 1$

$x_2 = \dfrac{\text{No. of bedrooms}}{5} \Rightarrow 0 \le x_2 \le 1$



$\Rightarrow$ we can Reach local Minima faster

Feature Scaling :-

- Get every feature into approx a $-1 \leq x_i \leq 1$ Range

E.g. $0 \leq x_i \leq 3$ ✓ OK

$-2 \leq x_2 \leq 0.5$ ✓ OK

$-100 \leq x_3 \leq 100$ ✗ Not OK } Not too large, Not too small

$-0.001 \leq x_4 \leq 0.001$ ✗ Not OK }

→ Generally ⇒ $\boxed{-3 \leq x_i \leq 3}$

Mean Normalization :-

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

i.e $x_1 = \frac{\text{Size} - 1000}{2000} \Rightarrow -0.5 \leq x_1 \leq 0.5$

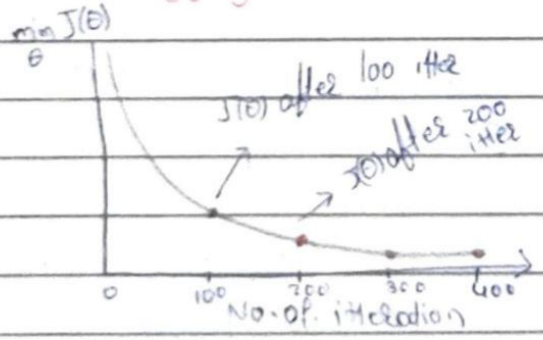
$x_2 = \frac{\text{bedroom} - 2}{5} \Rightarrow -0.5 \leq x_2 \leq 0.5$

Generally,

$x_i \longleftarrow x_i - \mu_i$ — Avg value of x (training set)

s_i — Range (max - min) / std. dev

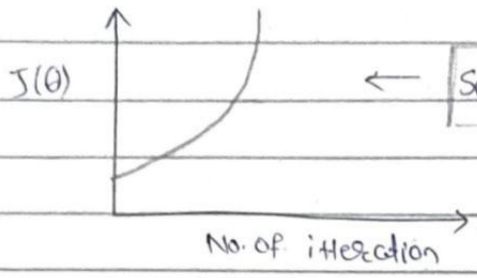
Debugging Grad descent



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

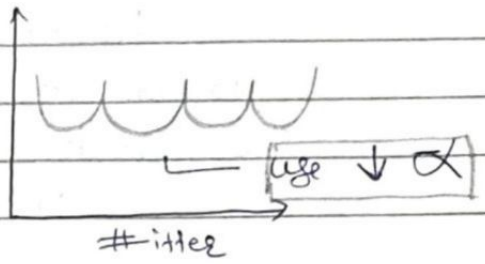
"Debugging" :- Making sure that Grad descent working Properly.

Case I



← Sol^n :- ↓ α

Case II

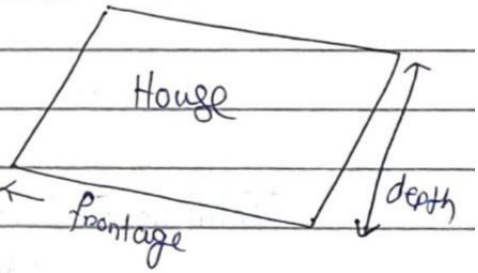


→ our α should be suffi small so that J(θ) will decrease after every itteration

⇒ if α is too small then, Grad descent can be slow to Converge.

* Features and Polynomial Regression

House Price Prediction.



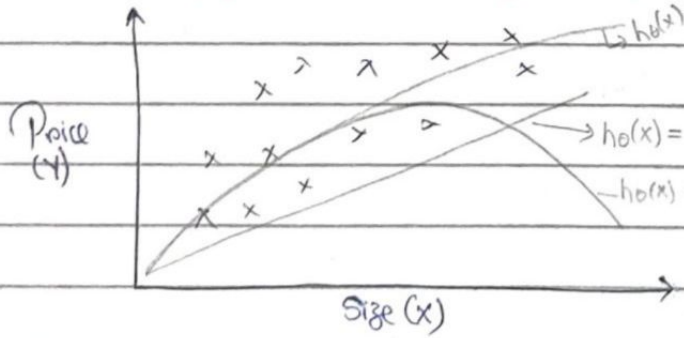
$h_\theta(x) = \theta_0 + \theta_1(\text{frontage}) + \theta_2(\text{depth})$

- we are given with two features
  X_1 = frontage
  X_2 = depth
- but we can Create our own feature
  i.e X_3 = frontage * depth = land area

Now,

$$h_\theta(x) = \theta_0 + \theta_1(\text{land area})$$ — New hypo

* Polynomial Regression



- we can see lineal model Can't fit here ⇒ $h_\theta(x) = \theta_0 + \theta_1 x$
- we won't fit Quadratic model as it will come down
  ↳ $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
- we are going to fit Cubic model

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

Now, $h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2 + \theta_3(size)^3$ — (1)

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

to map eq. (1) and eq (2)

we can do,

$x_1 = (size)$ $\quad x_2 = (size)^2$

$x_3 = (size)^3$

size : 1 – 1000 ; $size^2$: 1 – 100000 , $size^3$:. 1 – 10^x

So our features are having varied range

So here we will apply

FEATURE SCALING

Other choise,

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

* Normal Equation

Normal Eq :- Meth to solve θ analytically

Intution :- If 1D ($\theta \in R$)

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

To minimize $J(\theta) \Rightarrow \frac{d}{d\theta} J(\theta) \underset{Set}{=} 0$ – Solve for θ

Actually,

$\theta \in R^{n+1}$ $\quad J(\theta_0, \theta_1, \dots \theta_m) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

$\frac{\partial}{\partial \theta_j} J(\theta) = \dots \underset{Set}{=} 0$ for every j

Solve for $\theta_0, \theta_1, \dots \theta_n$.

P.T.O

Ex.. $m = 4$

x_0	Size (feet²) (x_1)	# bedroom (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

Now,

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

m = no. of exam/sample
n = features

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

In General

m examples :- $[(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}) \dots (x^{(m)}, y^{(m)})]$; n features

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

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

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

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

- $(X^T X)^{-1}$ is inverse of $X^T X$.

∴ Octave : pinv (x'*x)*x'*y

= Set $A = X^T X$

then $\theta = A^{-1} X^T y$.

# Gradient Descent		Normal Equation	
Disadv	= Need to choose α	Adv	= No need of α
	= Need many iter		= Don't need to iterate
Advan	= Work well even when n is large	Dis Adv	= Need to compute $(X^T X)^{-1}$
			= Slow if n is very large

Use Gradient Descent ← beyond this — Can go till $n = 10000$

★ Normal eqn (Noninvertibility)

What if $(X^T X)$ is non-invertible (singular)

In octave we have Pinv and inv to calculate inverse

Pinv ⟹ Pseudo Inverse = although the $X^T X$ is singular it will calculate inv

inv ⟹ Real Inv = No ans if $X^T X$ is singular

When $X^T X$ will be non-invertible?

① Redundant features (linearly dependent)

Deleting Redundant feature will solve this problem

let $x_1 = \text{Size (feet)}^2$ $\quad 1m = 3.28$ feet

$x_2 = \text{Size (m)}^2$

$\therefore x_1 = (3.28)^2 x_2$

= if x_1 and x_2 are linearly dependent then $X^T X$ is non-invertible

② Too many features (e.g. $m \leq n$)

m = training ex.

n = features

— Delete some feature / Regularization

↳ This will solve non-invertibility