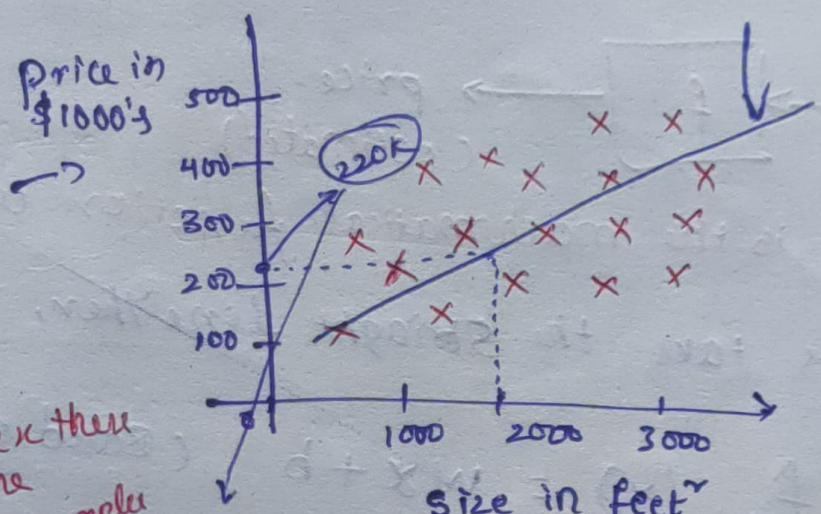
Linear Regⁿ Model part 1

Data table:

	(x) Size in feet²	(y) Price in $1000's
1 →	2104	440
2 →	1416	232
3 →	1534	315
	852	178
	⋮	⋮
47 →	3210	870

For ex there are 47 examples



Best fit line

we will find corresponding price for size of house using Regression.

This Dataset Used to train the model called "Training set"

x → "input" variable (feature)

y → "output" variable (target Variable)

x = 2104 y = 440

(x, y) → single training example like (2104, 440)

$(x^{(i)}, y^{(i)})$ → ith training example like 2nd baing example

$(x^{(2)}, y^{(2)})$ → (1416, 232)

i = 1, 2, 3 .....

Training Set → features and targets (Independent + Dependent cols)

↓

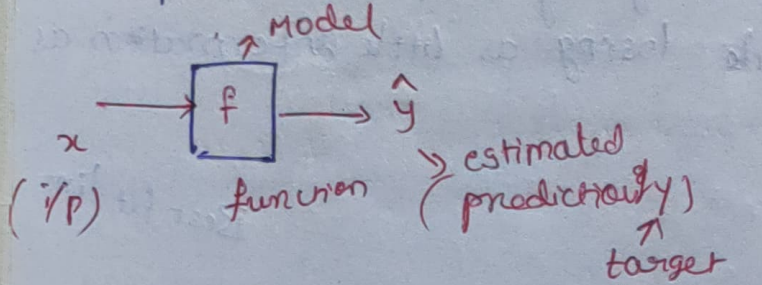
Learning Algorithm

↓ it will produce some function says 'f'

↓

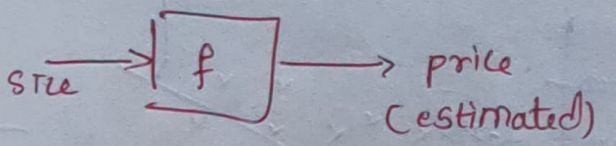
f → function called hypothesis

That function will generate new output for the test data



y → actual o/p in (traing set)

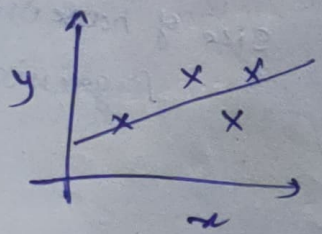
ŷ → predicted o/p (for test input)



What is the mathematical function we used in the 'f' to compute the output?

Let's take the Straight line then,

$f_{w,b}(x) = wx + b$ (Based on x we predict ŷ)



$f_{w,b}(x) = wx + b$

$f(x) = wx + b$

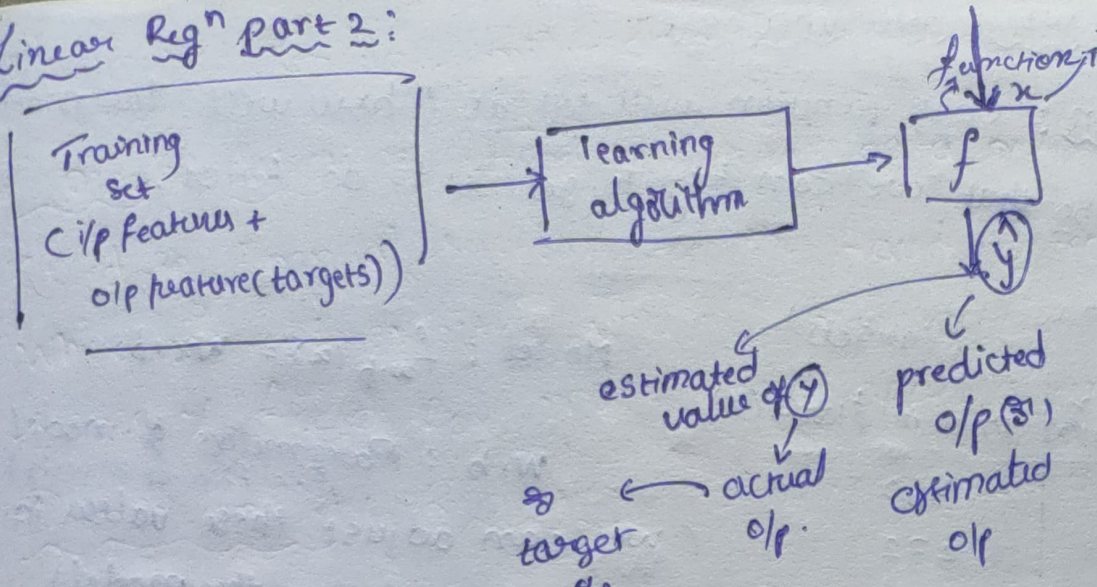
Linear Reg^n with one Variable (single feature x)

(Or)

Univariate Linear Reg^n

(Later we will how we can find price of house, depending on multiple feature like (size, area, bedroom, ...))

Linear Regⁿ Part 2:

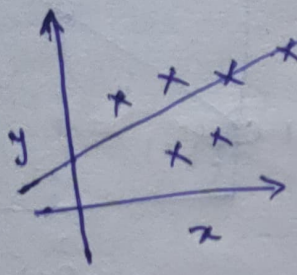


How to Represent f ? (Model) i/p feature i/p feature

(Mathematical eqⁿ) ⟶ $f_{w,b}(x) = wx + b$

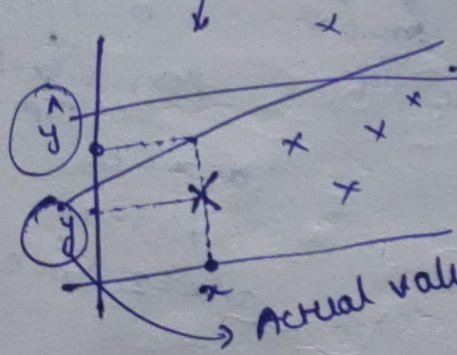
(or) $\hat{y}$ Based on w, b value the $\hat{y}$ values will be determined

$$f(x) = wx + b$$



straight line.

we predict value of y based on straight line



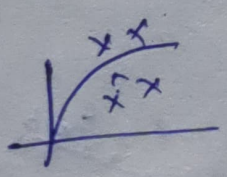
Single feature x ie size

(LINEAR REGⁿ WITH ONE VARIABLE)

Q) Why you choose straight line in Linear Regⁿ?

It's not like that we need to use some complex fⁿ also like curve

(Learn this Later)



(polynomial Regⁿ ; think

LINEAR REGN WITH ONE VARIABLE

Cost Function: Cost function will tell us "how well the model is doing so that we can try to get it to do better".

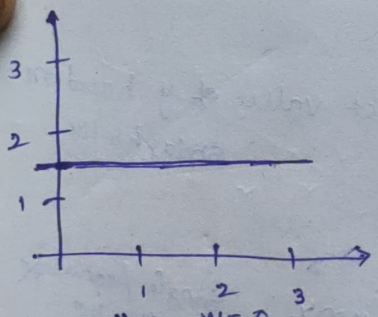
Training Set:

(features) Size in feet² (x)	(targets) price in $1000's (Y)
2104	460
1416	232
1534	315
852	178
⋮	⋮

Model : $f_{w,b}(x) = wx+b$

w, b : Parameters of model
(coefficient & weights)
(we can adjust these values in training to improve the model)

what w, b do? (Based on w, b we have diff f(x) then it generate diff line on the graph)



When $w=0$

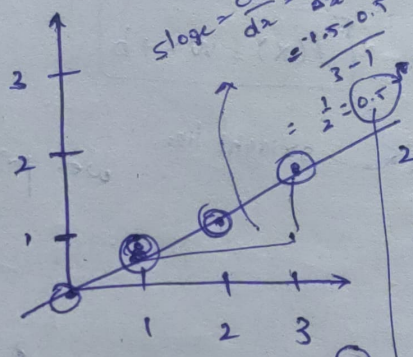
$b=1.5$

$f(x) = wx+b$

$= 0.x+b$

$\hat{y} = b = 1.5$ → constant

↑ y intercept



When $w=0.5$

$b=0$

$f(x) = 0.5x + b \leftarrow 0$

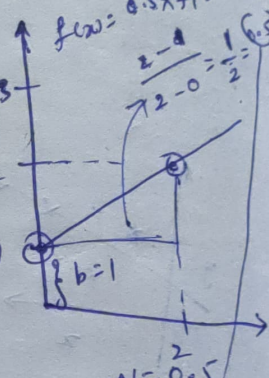
$f(x) = 0.5x$

if $x=0$ $f(x)=0$

$x=1$ $f(x)=0.5$

$x=2$ $f(x)=1$

$x=3$ $f(x)=1.5$



$w = 0.5$

$b=1$

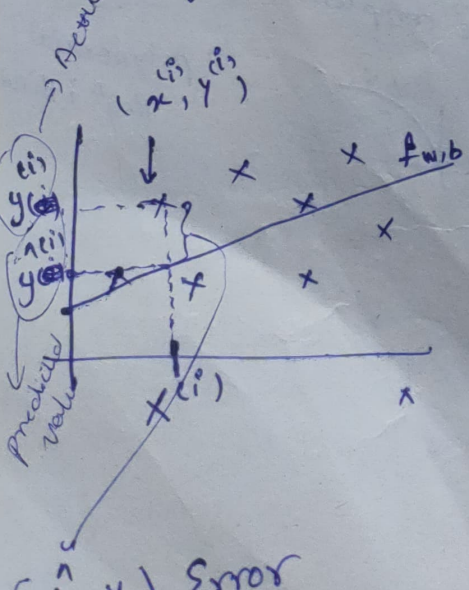
$f(x) = 0.5x+1$

slope

at $x=0$

$f(x)=1$

at $x=2$ $f(x)=2$



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

$\hat{y}^{(i)} = f_{w,b}(x^{(i)})$

$f_{w,b}(x^{(i)}) = wx^{(i)} + b$

$(\hat{y} - y)$ Error

To know how well a line fits the data?

→ This is know by Error b/w $\hat{y}$ & y (pred vs actual) for all pts whatever the line gives less error then that will be Best fit line.

cost f^n:

$m \rightarrow$ no. of training examples

$\sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2 \rightarrow$ Sum of squared errors for all training examples.

Suppose if we have more training examples then m gets so that the sum of squares of errors are also increases. So for that for large training set examples we take average like cost function as

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

some ML Engr use $\frac{1}{2m}$ instead of $\frac{1}{m}$ to make neater calculation.

we can use $\frac{1}{2m}$ or $\frac{1}{m}$ both are okay.

So, Finally

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

m = no. of training examples.

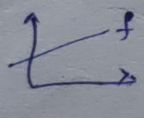
→ Squared error cost function.

$$J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (f_{w,b}(x^{(i)}) - y^{(i)})^2$$

Cost Function Intution:

If we are looking to fit a straight line to training data, then we have below,

model: $f_{w,b}(x) = wx + b$



parameters: w, b

cost f^n: $J(w,b) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$ (→ To know how best our line fit the data)

$f_{w,b}(x^{(i)})$

goal: $\min_{w,b} J(w,b)$ (whatever line gives less $J(w,b)$ value that will be the best fit line)

If you want to better visualize the cost function J, this work of a simplified version of Linear Reg^n model. We are going to use $f_w(x) = wx$

$f_w(x) = wx$ ← simplified model.

$b = 0$

Now we have only one parameter 'w' & cost f^n J

$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} (\underbrace{f_w(x^{(i)})}_{wx^{(i)}} - y^{(i)})^2$$

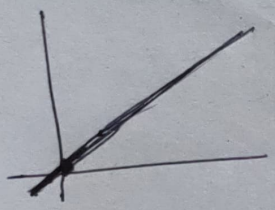
with this simplified model our goal is to find minimize value of J(w)

$\underset{w}{\text{minimize}}\ J(w)$ {we need to find value of w that minimize the value of J(w)}

if $b=0$, then $f_w(x) = wx \rightarrow$ lly $y = mx$

↓

straight line passing from origin



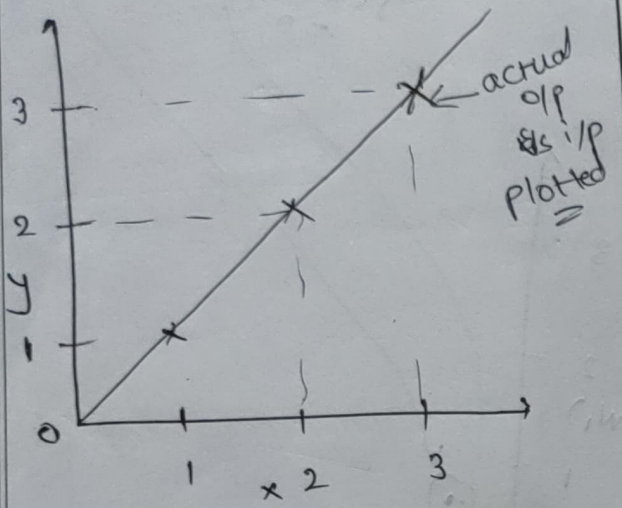
Let's see how cost function changes for different values of 'w'

$f_w(x)$	$J(w)$
$f_w(x) = wx$	$J(w) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$
For fixed w, function of x (input)	($\hat{y}^{(i)} \rightarrow wx^{(i)} \rightarrow f_w(x)$)
i.e the $f_w(x)$ or $\hat{y}$ is dep ends on 'x' (input)	(function of w)
(function of input x)	where 'w' controls the parameter slope of line defined by $f_w(x)$. so cost defines on J depends on parameter 'w'. (function of parameter w)

$f_w(x)$

(for fixed w, function of x)

↓ input



Let's say $w=1$

$f_w(x) = wx = x$

$f_w(1) = 1$ $\quad f_w(2) = 2$

Let's calculate J when $w=1$

$$J(1) = \frac{1}{2(3)} \sum_{i=1}^{3} (\hat{y}^{(i)} - y^{(i)})^2$$

$\hat{y}^{(i)} \rightarrow f_w(x)^{(i)} \rightarrow wx^{(i)}$

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

here the error is zero

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

$(1-1)^2 (1-1)^2 (1-1)^2$

$\boxed{J(1) = 0}$

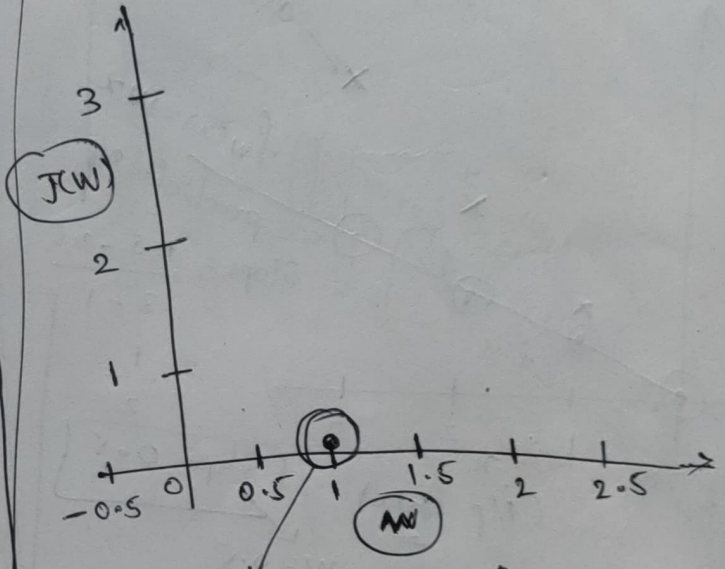
$J(w)$

(function of w)

↑ parameter

cost fn

As seen here at $w=1$, the $J=0$



at $w=1$ $J(w)=0$

$\boxed{J(1)=0}$

$f_w(x)$ | $J(w)$

Now take $w = 0.5$

$f_w(x) = 0.5x$

↓ slope = 0.5

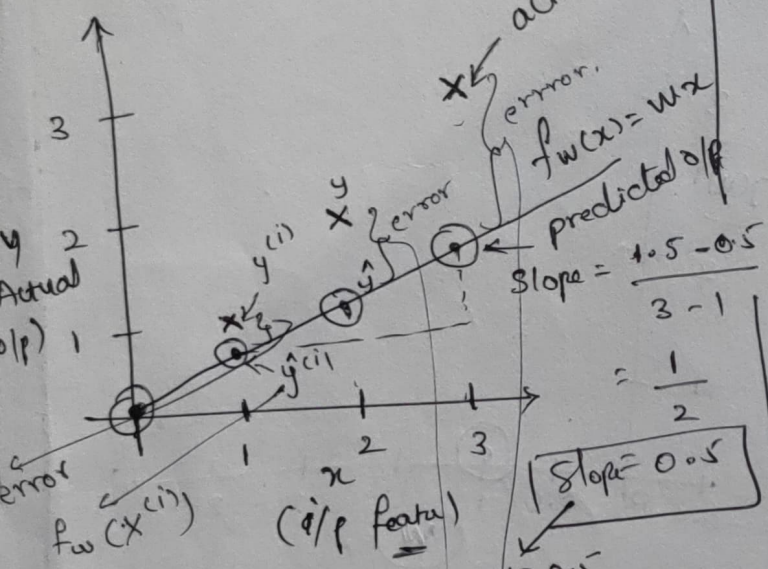
at $x = 1$ $f_w(x) = 0.5$

$x = 0$ $f_w(x) = 0$

$x = 2$ $f_w(x) = 1$

$x = 3$ $f_w(x) = 1.5$

P/L

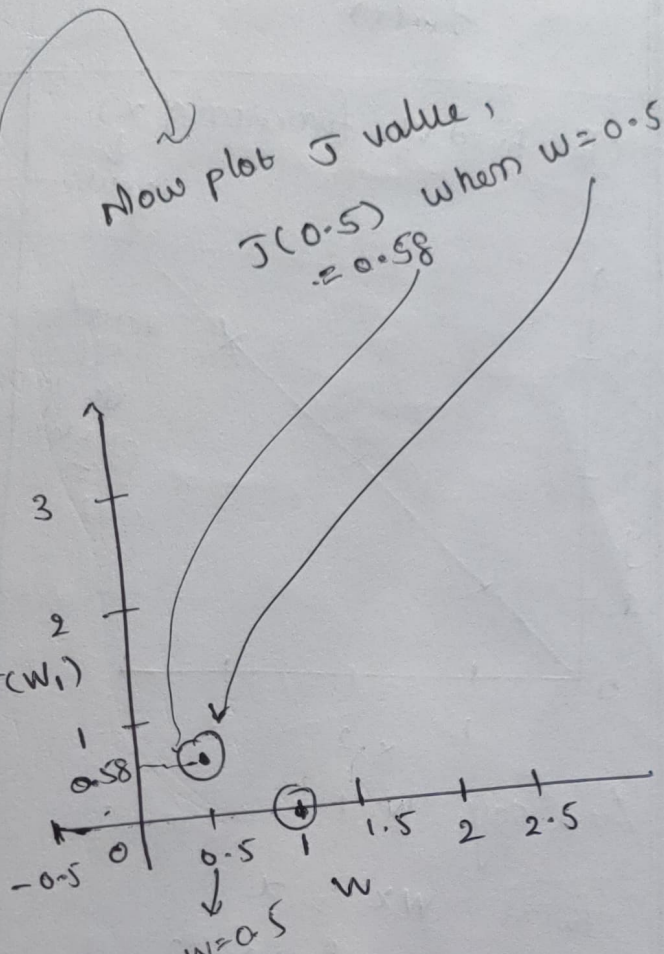


Slope $= \frac{1.5 - 0}{3 - 1}$

$= \frac{1}{2}$

Slope = 0.5

$w = 0.5$



$J(0.5)$

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

$\hat{y}^{(i)} = wx^{(i)}$

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

$= \frac{1}{2 \times 3} [3.5] = \frac{3.5}{6} = 0.58$

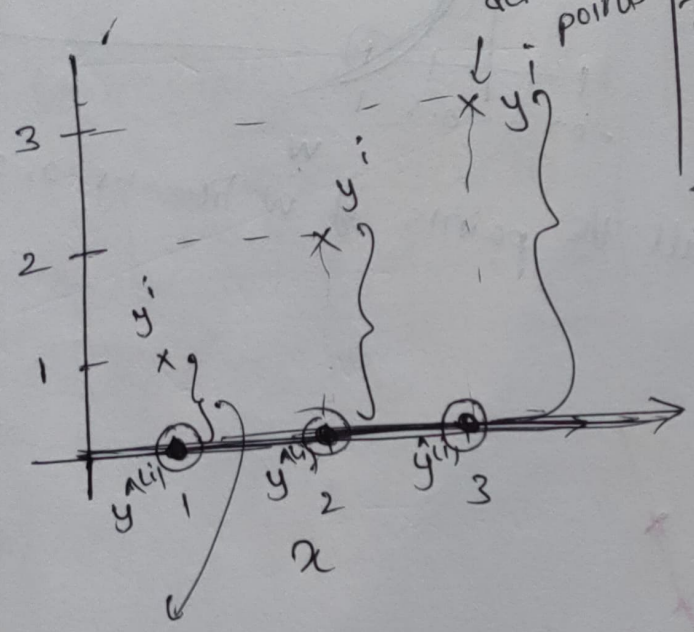
$f_w(x)$ | $J(w)$

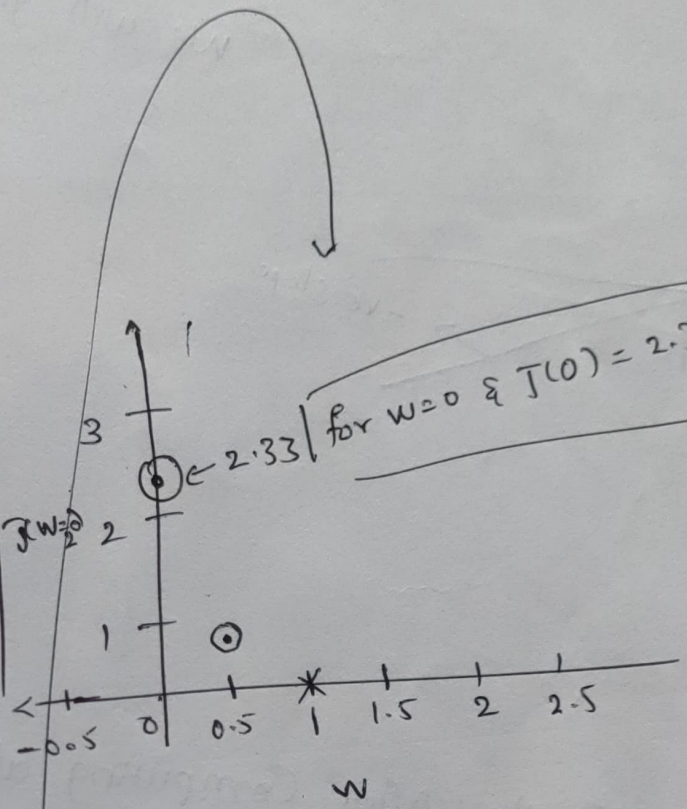
Take $w=0$

$f_w(x) = w.x$

$= 0$

for any value of x, the $f_w(x)$ is zero



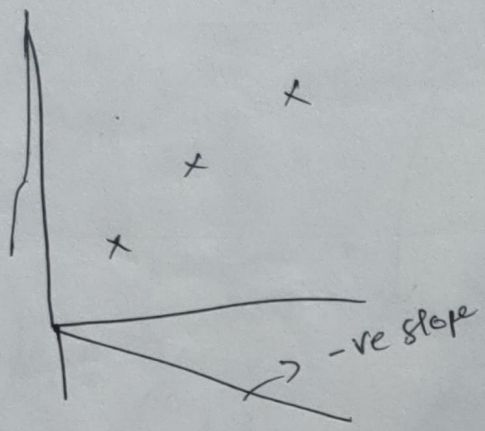


$$J(0) = \frac{1}{2\times 3}\left((0-1)^2+(0-2)^2+(0-3)^2\right)$$

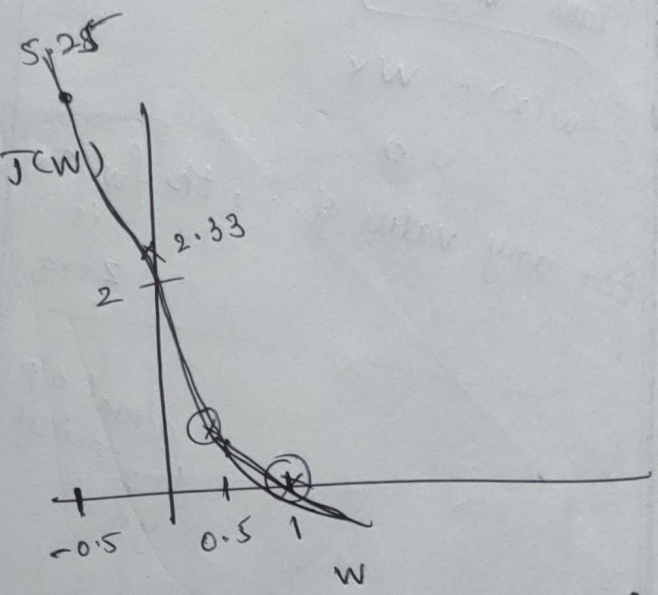
$$= \frac{1}{6}(14) = 2.33$$

If we Repeat the process for multiple points like

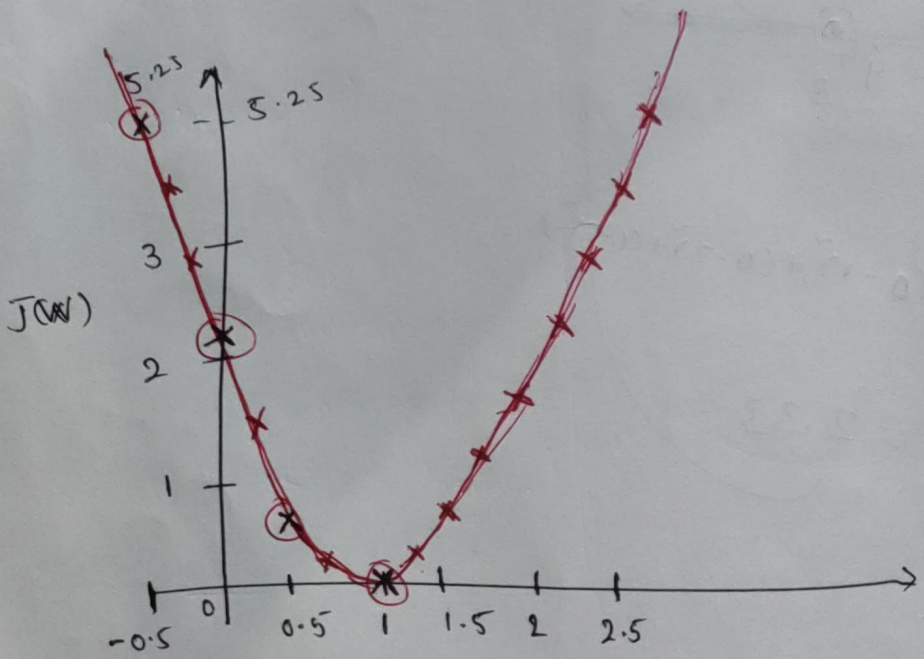
$w = -0.5$, then $f_w(x) = -0.5x$



We will got $J(w) = 5.25$ Something then



And Finally After Computing all the points of w like 1.5, 2, 2.5..... then we will got like below



(X) → we plotted prev.

So How the question is how can you choose value of w to Minimize J(w)?

We need value of w, where we have less J(w).

In this example, From graph at w=1, we have J(w) = 0

So, w=1 is the Best value for the Best fit Line.

Goal of Linear Regression

$$\underset{w}{\text{minimize}}\ J(w)$$

General Case:

$$\underset{w,b}{\text{minimize}}\ J(w,b)$$

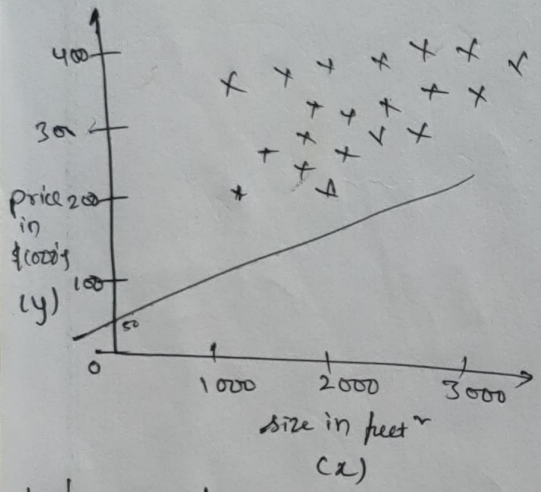
Visualizing the Cost function:

Till Now, we seen $f_w(x) = wx$ i,e at $b=0$

Now we will see when $b \neq 0$

i,e $f_{w,b}(x) = wx+b$

$f_{w,b}$	J
(function of x)	(function of w, b) ~ Both parameter



Let's take like

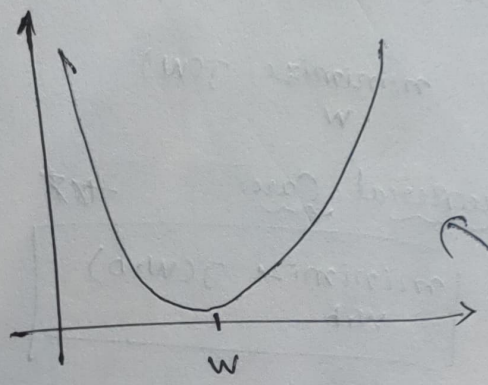
$f_{w,b}(x) = 0.06x + 50$

w → Slope

b → intercept

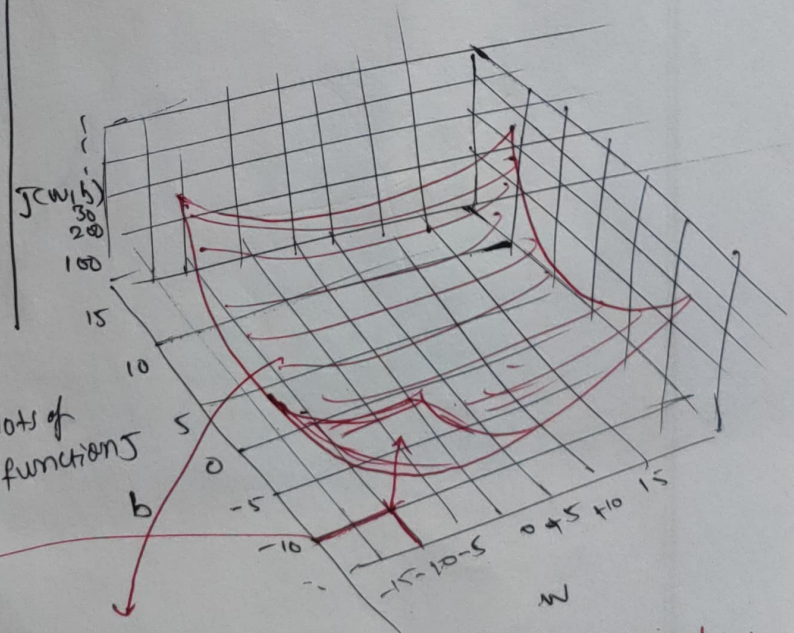
We know this Model is worst because it Underestimate the price of house

let's consider as Ex.



When $b=0$, J is function of w only then we got a ~~function~~ like above U-Curve

But Now we have two parameters (w, b)



Contour plots are best to visualize 3D plots of cost function J

at $b=-10$ & $w=-10$ the J is height of arrow

Now Also it is like U-shaped but in 3d shape

As we vary w & b we at diff values for $J(w,b)$

*What we really need is a efficient algorithm that you can write in code for automatically finding the values of parameters w & b they give you the Best fit Line that minimizes cost function J There is an algorithm for doing this called "GRADIENT DESCENT".

TRAIN THE MODEL WITH GRADIENT DESCENT:

GRADIENT DESCENT

Gradient descent not only used in Linear Reg^n But also used in some deep learning models.

Have some function $J(w,b)$ → it is cost f^n of Linear Reg^n.

Want to $\min_{w,b} J(w,b)$

Gradient descent is used to minimize any f^n (not only Linear Reg^n cost Reg^n)

" " is applied to more general functions, including other cost functions that work with models that have more than two parameters.

for Ex for instance we have cost f^n $J(w_1, w_2, w_3, \dots, w_n, b)$

Our objective is to ~~minimize~~ $\min_{w_1, w_2, w_3 \dots w_n, b} J(w_1, w_2, w_3 \dots, w_n, b)$

Outline:

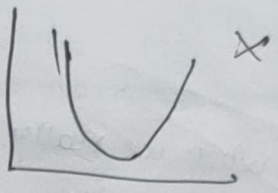
Minimize $J(w,b)$ w,b

1) Start with some w,b (like start with $w=0, b=0$) as initial guess

2) Keep changing w,b to reduce $J(w,b)$

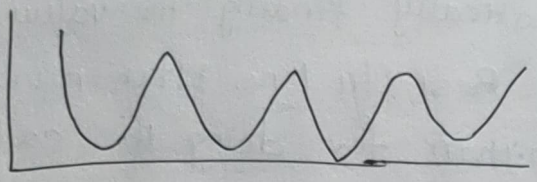
Until we settle at or near a minimum

One thing we need to note, the J is not always U-shaped like

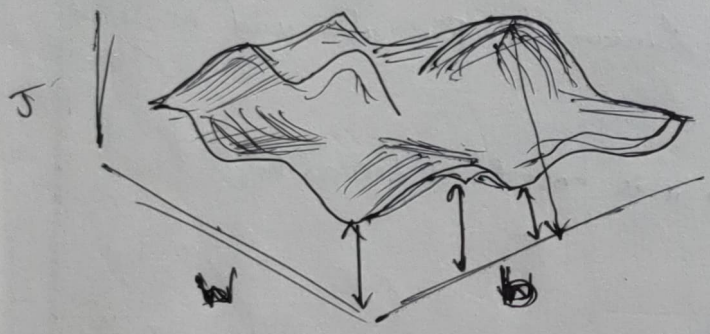


Sometimes we may have >1 minimum values

for ex:

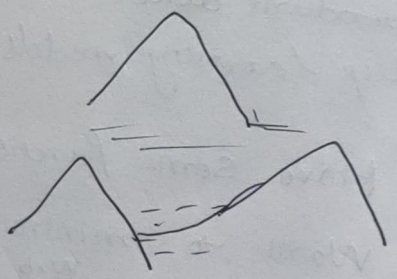


← Multiple low vals

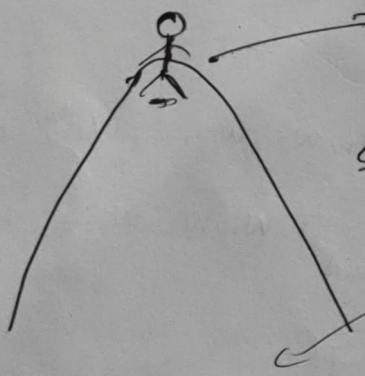


← 3D plot

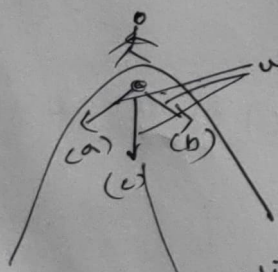
it is like Mountain



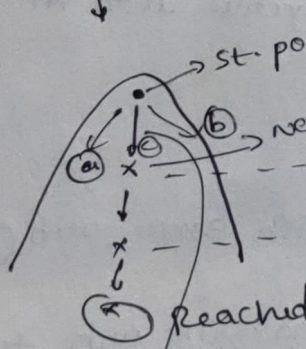
How Gradient descent Algo works?



if you are on the Mountain & you want to reach bottom of mountain Efficiently as possible, so, what we will do, we will look around & take one tiny step towards bottom.



we have 3 ways possible to take tiny step But we choose Middle one because Middle one is efficient (we can reach fastly if we select ~~2nd~~ one) Middle



st. position

new position

Reached → end

direction of Steepest descent.

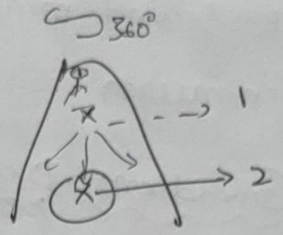
This will take (c) than a tiny little baby step you could have taken in other direction (a, b)

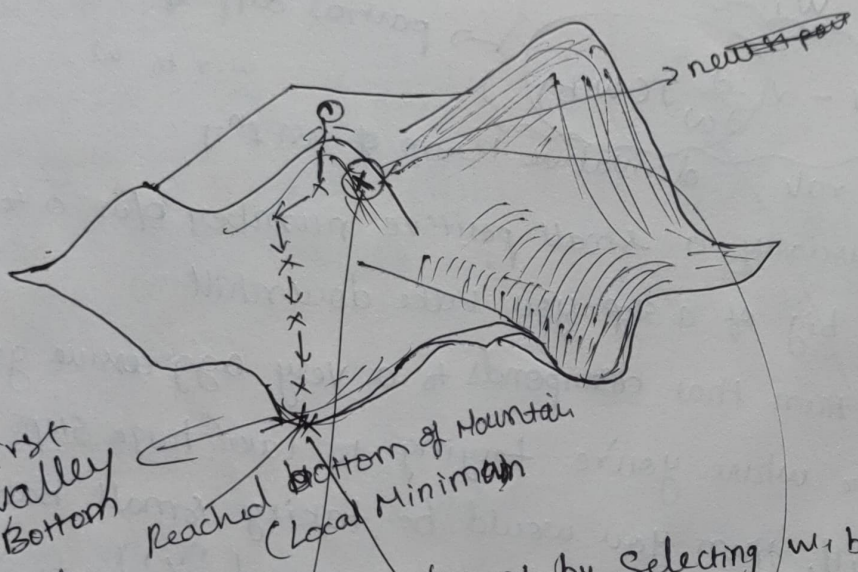
After taking one step, We need to repeat process

→ Need to see 360°

→ choose the Best path
(what rent little baby step inorder to move downhill.)

If you do that and take another step.... until end of Mountai



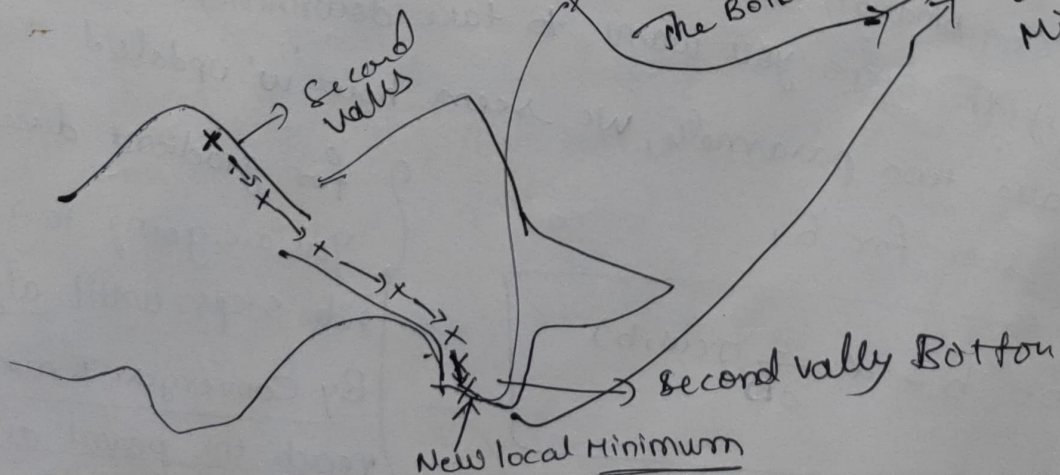


Remember,
We are choosen a starting pt by selecting w, b at starting. right.

Now imagine if you try gradient descent again, But this time you choose a different starting point by.

For this time we get new local Minimum

The Bottoms of Both valleys called Local Minima

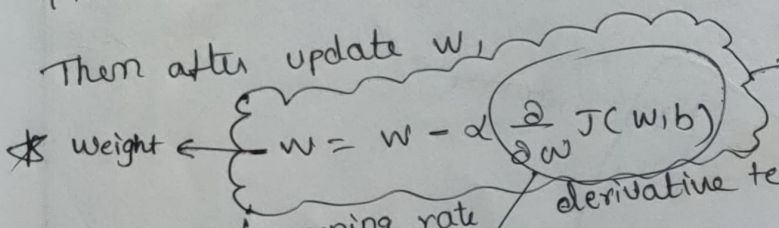


Because if you start going down the first valley, Gradient descent won't lead you to second valley, & the vice versa is also true. & Same is true if you started going down the second valley, you stay in that second minimum and not find your way into first local minimum

Implementing Gradient Descent Algorithm:

1) First take ~~one value of~~ we have taking the value for w right

Then after update w_1



Learning rate is usually a small positive number b/w 0 to 1 (say 0.01)

'α' controls how big of a step you take downhill

If 'α' is very large, then that corresponds to a very aggressive gradient descent procedure where you're toying to take huge steps downhill.

If 'α' is very small, then you would be taking small baby steps downhill. (We will see how to choose good 'α' later)

derivative part gives → In which direction you want to take your baby step.

In combination with learning rate alpha, it also determines the size of the steps you want to take downhill.

We have two parameters, we seen how 'w' updated

Let's see for b,

$$b = b - \alpha \cdot \frac{\partial}{\partial b} J(w,b)$$

bias

$$w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$

for gradient descent algorithm, you are going to repeat the side steps until algorithm converges.

By converges, I mean that you reach the point at a local minimum where the parameter w & b no longer change much with each additional step that you take.

Now, there's one more subtle detail about how to correctly in Semantic gradient descent, you are going to update two parameters w and b. This update takes place for both parameters, w and b. One important detail is that for gradient descent, you need to simultaneously update w and b (need to update Both parameters at the same time) (old w to New w & old b to New b)

code for update

$$tmp\_w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$

$$tmp\_b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

$$w = tmp\_w$$

$$b = tmp\_b$$

Correct:

simultaneous update ✓

(DO this one)

Incorrect

$$tmp\_w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$

$$w = tmp\_w$$

$$tmp\_b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

here w changing

$$b = tmp\_b$$

here b temp-b is diff & here temp-b value is also diff

so, this is incorrect notation

NOTE:

Gradient descent is an algo for finding values of parameters w and b that minimize the cost function J. What does this update statement do? (Assume α is small)

$$w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$

Ans) Updates parameter w by small amount.

GRADIENT DESCENT INTUITION: (Let's see how α & derivative & Multiplication of both is actually works)

Repeat until convergence {

$\alpha \rightarrow$ Learning rate (It control the how big of a step you take) when updating w & b

$$w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$

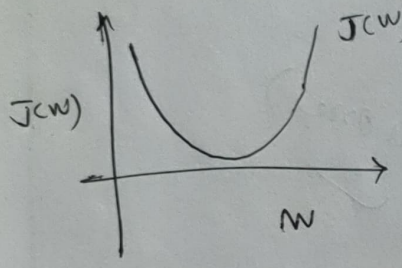
$$b = b - \alpha \frac{\partial}{\partial b} J(w,b)$$

Derivative part $\rightarrow$ It will

Let's see small example how updation is done & all

Let's take $J(w)$ instead of $J(w,b)$ $\rightarrow$ $w = w - \alpha \frac{\partial}{\partial w} J(w)$

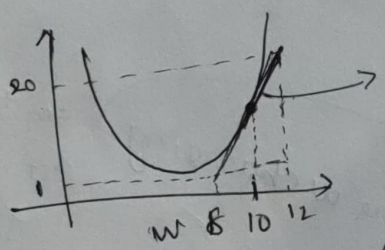
Our aim is to minimize $J(w)$ w



$\rightarrow$ it is like our prev example when we had temporarily set $b=0$

Now $\frac{\partial}{\partial w} J(w)$ means slope of tangent at all value of w.

Let's assume $w = 10$



$$\text{slop} = \frac{20-1}{12-8} = \frac{19}{4} = 4.75$$

slope = $\frac{\partial}{\partial w} J(w)$

+ve >0

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

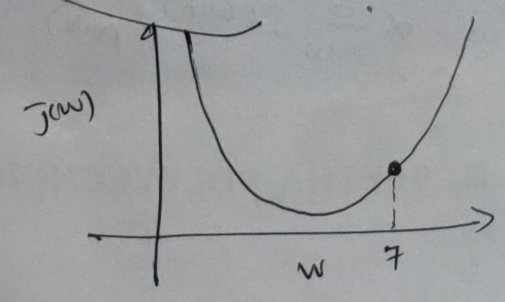
So, $w = w - \alpha$ (+ve number)

α is always +ve so,

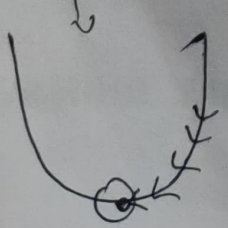
Finally, $w = w -$ (+ve number)

find w value is lower.

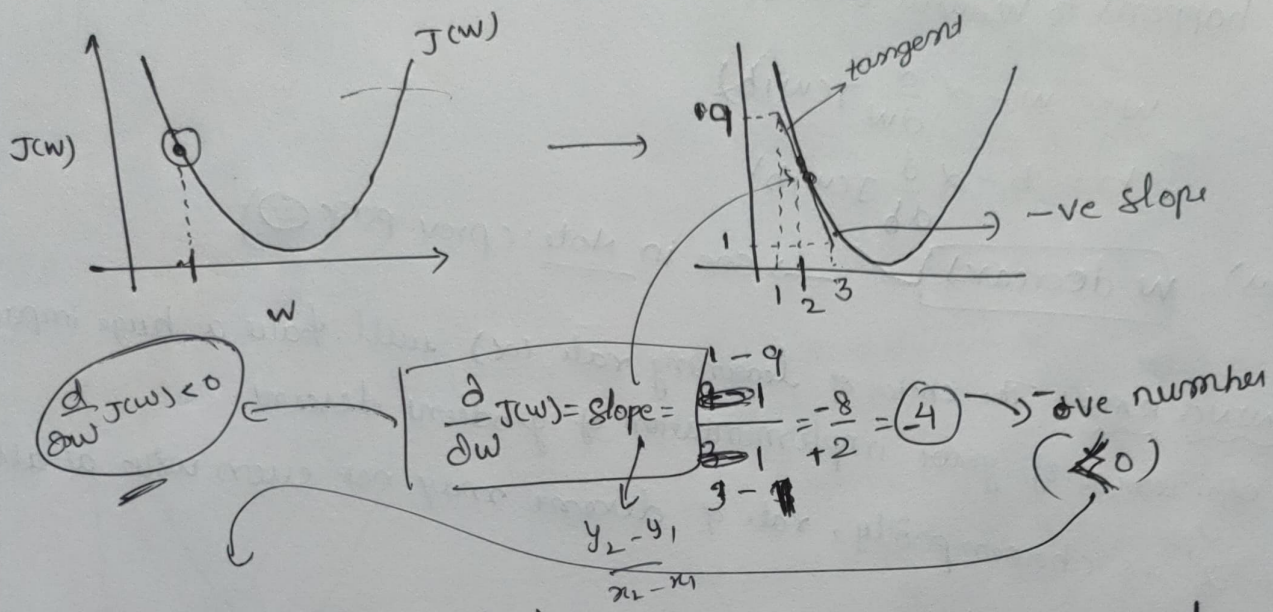
i.e in graph we are moving to left (say w=7)



So after moving to left again & again it will reached to Minimum point.



Let's take another point of w (this time at right)

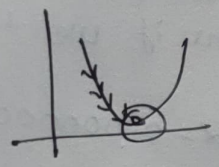


$w = w - \alpha$ (-ve number)

Then w value increases then it move towards right side (to know the Minimum point)

$w = w + +ve$

w ↑'es



NOTE :

If $\frac{d}{dw} J(w) > 0$ → i,e slope is (+ve) then w value will decreases

$\frac{d}{dw} J(w) < 0$ → i,e slope is (-ve) then w value Increases

Qns

1) Assume α is very small +ve number. When $\frac{d}{dw} J(w,b)$ is a +ve number (>0) → as in example in the upper part of slide down above, what happens to w after one update step.

$$w = w - \alpha \frac{d}{dw} J(w,b)$$

$$b = b - \alpha \frac{d}{db} J(w,b)$$

Ans) w decreases → see in Note: (prev page ☺)

LEARNING RATE: The choice of learning rate (α) will have a huge impact on the efficiency of your implementation of gradient descent

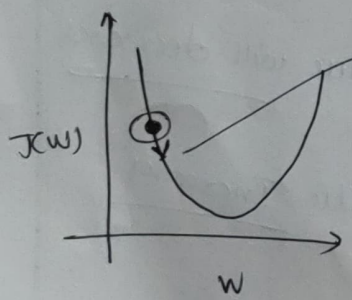
If α is choosen poorly, rate of descent may not even work at all

How to choose α?

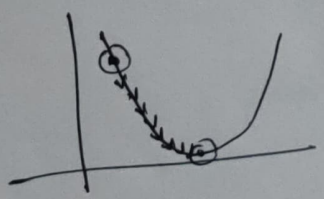
Let's take, $w = w - \alpha \frac{d}{dw} J(w)$

Let's see what happens if we take either too small (or) too high.

If α is too small: ($\alpha = 0.0000001$)



→ if we take α as small then the w is increased by very very small value that means we taken a litlly baby step.
So from that step we will repeat procss, we will take so many tiny steps to reach minimum point.



(Here we ~~did~~ The process is done like J is decreasing slowing upto it's minimum value by taking tiny steps)

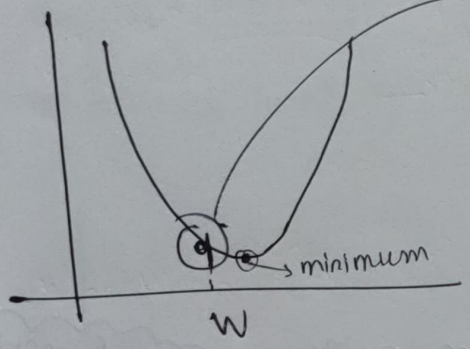
But we ~~need~~ notice that you're going to need a lot of steps to get minimum

To Summarize,

If 'α' is too small, Gradient descent will be basic, but it will be slow

↙ Because it take so many steps (it takes More time)

If α is too large:



Let's say we assume 'w' here. (It is pretty close to minimum)

But if α is high then,

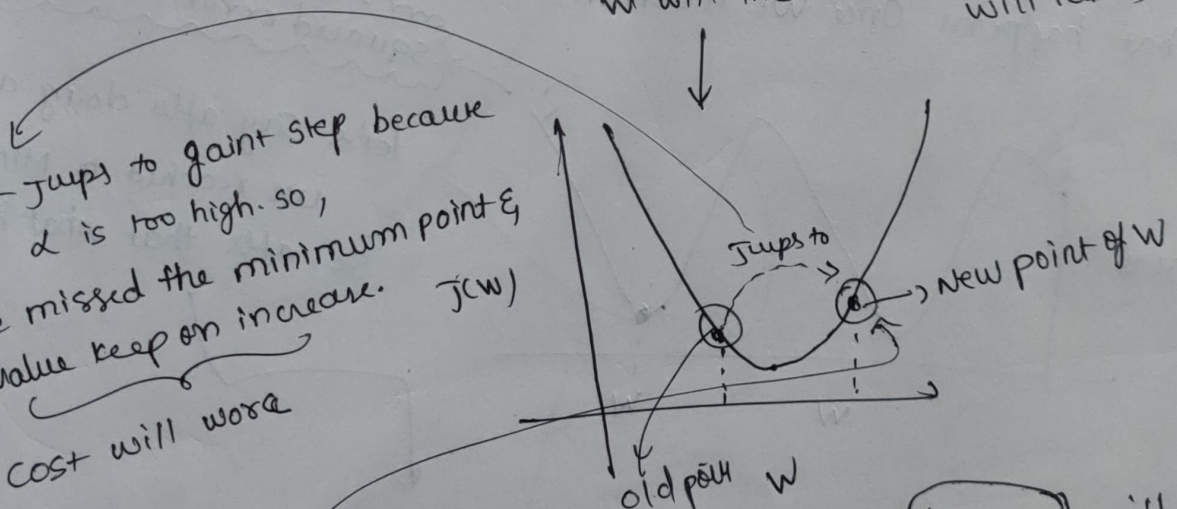
$w = w - \alpha(-ve)$ → -ve slope na for left side pt

↓ high

$w = w + \alpha$

↑ high

w will increase (a gaint step will taken)



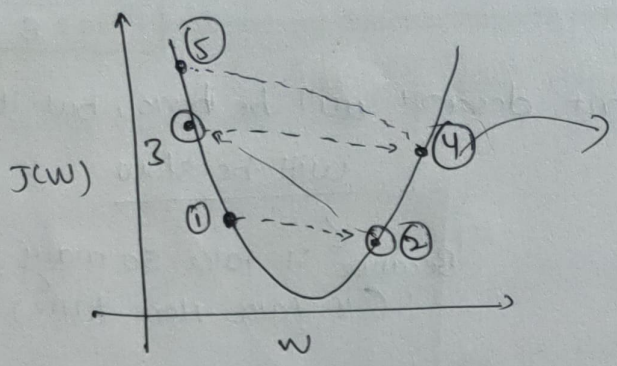
Then after reaching the derivative of $w = w - \alpha \frac{d}{dw} J(w)$ will (slope) be (+ve) then again $w = w - \alpha(+ve)$

↓ high

w = decrease by giant step again

Then new point will be here

If α is too high (too large) the steps will like



here we get +ve slope & α is also large then,

$w = w - \alpha$ (+ve)

+ve

+ve

w reduces very high (like w takes giant step to ⑤)

So, If α is too large,

Gradeint descent may

→ Overshoot, never reach minimum

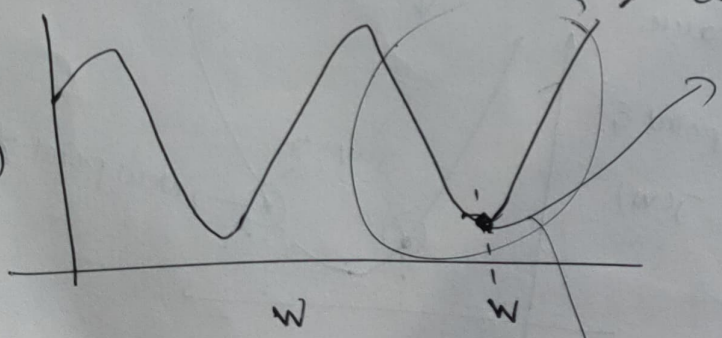
→ Fail to converge, diverge

IMPORTANT (Used for interview)

What happens Once We reach minimum point.

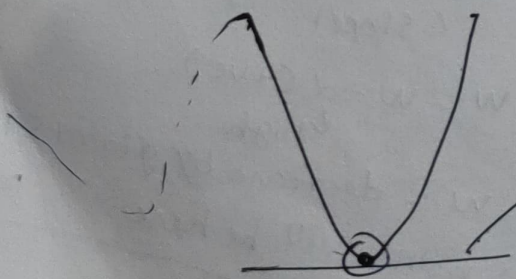
squared error cost function

Ex:



let's say after doing all steps we reached to Minimum

So, after that what happen?



slope = 0

0.

so, $w = w - \alpha \frac{d}{dw} J(w)$

$w = w - \alpha \cdot 0$ 0

$w = w$

So, w won't change

if we are at local minima, then we get slope = 0 then

$$w = w - \alpha \left(\frac{d}{dw} J(w)\right) \to 0$$

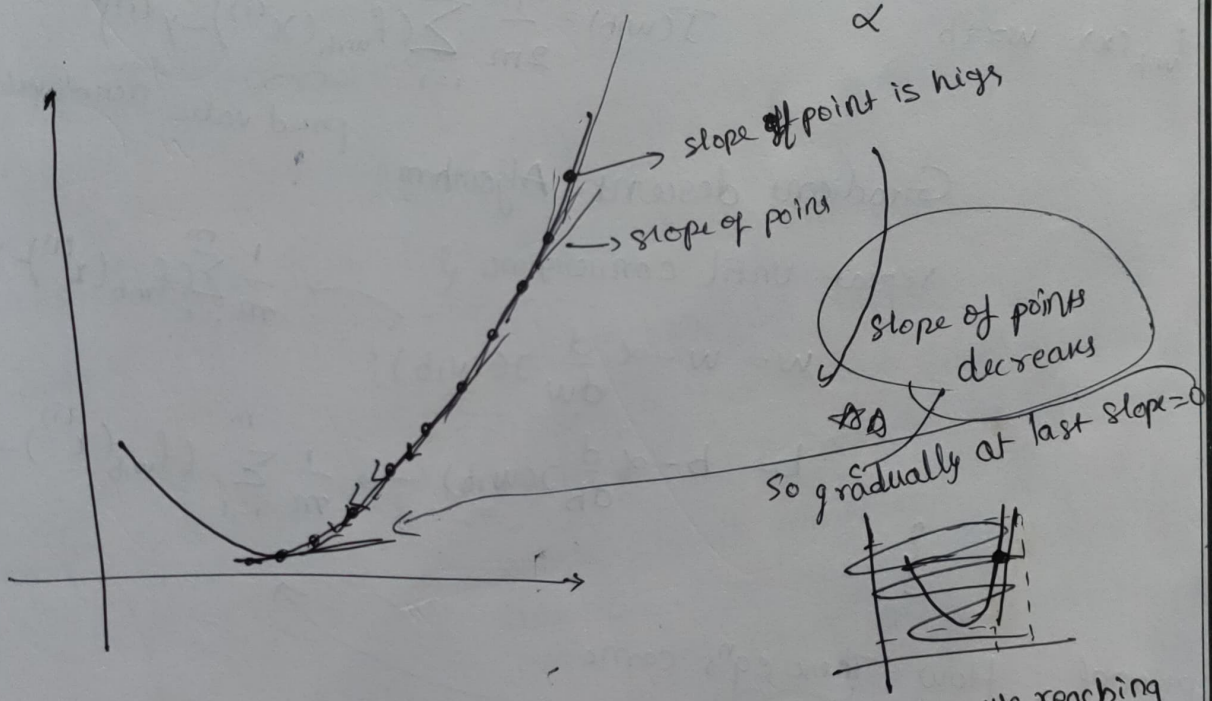
$$\therefore w - \alpha \cdot 0 = w.$$

The Gradient descent leaves 'w' unchanged when we are at minimum point. because at that point slope = 0 (horizontal line)

$w = w$ The w won't updated.

Q) Can reach local minimum with fixed learning rate?

α



$w = w - \alpha \frac{d}{dw} J(w)$ $\frac{d}{dw} J(w)$ getting Reduces when we reaching towards local minimum

Near local minimum,

→ Derivative becomes smaller.

→ Update steps become smaller.

Update steps smaller because (α is fixed)

α - fixed so, w depends on slope only

if slope ↓'es then w also ↓'es with small step.

So we can reach local minimum with fixed 'α'. ✓

GRADIENT DESCENT FOR LINEAR REGRESSION:-

(Here we use J as exact linear Reg^n cost function (& mean squared error cost funcn)

putting together Gradient descent with cost function will give you first learning algorithm, the Linear reg^n algorithm.

Linear Reg^n model

$$f_{w,b}(x) = wx+b$$

cost function

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$$

($f_{w,b}(x^{(i)})$: pred value, $y^{(i)}$: actual val)

Gradient descent Algorithm

repeat until convergence {

$$w = w - \alpha \frac{d}{dw} J(w,b) \longrightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})x^{(i)}$$

$$b = b - \alpha \frac{d}{db} J(w,b) \longrightarrow \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})$$

}

Optional: How these eq^{n}s came

$$\frac{d}{dw} J(w,b) = \frac{d}{dw}\frac{1}{2m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})^2$$ → partial derivative w.r.t. to derivative

$$= \frac{d}{dw}\frac{1}{2m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})^2$$

$$= \frac{1}{2m}\sum_{i=1}^{m} 2\cdot(wx^{(i)} + b - y^{(i)}) \frac{d}{dw}(wx^{(i)} + b - y^{(i)})$$

$$= \frac{1}{m}\sum_{i=1}^{m}(wx^{(i)} + b - y^{(i)})(x^{(i)} + 0 - 0)$$

$$= \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})$$

$\frac{d}{dx}(x+2+3)^2$

$2(x+2+3)\cdot\frac{d}{dx}(x+2+3)$

$2(x+2+3)(1+0+0)$

$2(x+5)$ ✓

Hy, $\frac{d}{db} J(w,b)$ also $\frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})$

↳ here we done partial diff w.r.to 'b' except b all will consider as constant.

Gradient descent algo for Linear Regn is

repeat until convergence {

$$w = w - \alpha \cdot \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)}) \cdot x^{i}$$

$$b = b - \alpha \cdot \frac{1}{m}\sum_{i=1}^{m}(f_{w,b}(x^{(i)}) - y^{(i)})$$

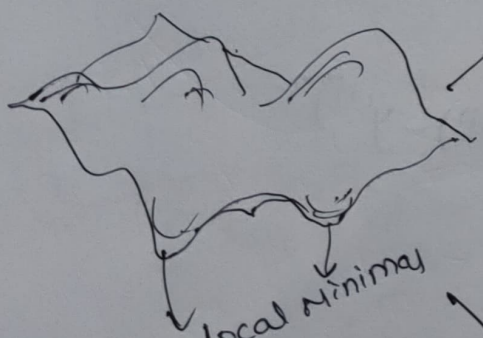
}

Need to update w, b simultaneously

$$f_{w,b}(x^{(i)}) = wx^{(i)} + b$$

Let's familiar with how gradient descent works

One the shock we saw with gradient descent is that it can lead to local Minimum instead of global minimum where global minimum means the point that has the lowest possible value for cost function J of all possible pts



>1 ~~Max~~ local minima (2 local Minima)

depending on where we initialize w, b, we can end up with diffent local minima. (we seen this earlier also)