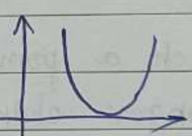
Machine Learning : Supervised Learning.

__/__/__

Gradient Descent :

1) Gradient Descent is used in machine learning to minimise the cost function.
$J(w,b)$: Cost function
we want $\min_{w,b} J(w,b)$

2) It minimises any function, not just cost function in linear regression.
If we have a cost function $J(w_1, w_2, w_3 \cdots w_n, b)$
our objective is to minimise J over $w_1, w_2, \ldots w_n, b$
we need to pick values for $w_1, w_2 \cdots w_n, b$.

3) Outline start with some initial guesses for w, b. Set them.
$w, b = 0, 0$
Keep changing w, b to reduce $J(w,b)$ everytime to reduce the cost j of w, b until hopefully j settles at or near a minimum.
Some functions J is always not 

4) This function is not a squared error cost function.
Steeptest descent.
The lowest point is called local minima. It helps to go downhill in the graph.

Mathematical Expressions.
Gradient descent Algorithm.

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

= Assignment operator.
= In Mathematics it is true assertions
== In python

α is the learning rate
It is a positive number.
between 0 and 1, 0.01 also

→ What alpha does is how big of a step you take downhill. If alpha is very large then that corresponds to a very aggresive gradient descent procedure where you are trying to take huge steps downhill.

→ If alpha is small we will take very small steps downhill.

$\frac{\partial}{\partial w} J(w,b)$: Derivative term of ~~alpha~~ cost function J.

b is assigned with the old value of b

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

Repeat until convergence.

We reach a point of local where the parameters w and b no longer change much with each additional step you take.

We will simultaneously update w and b. from old value.

$$\left.\begin{aligned} temp\_w &= w - \alpha \frac{\partial}{\partial w} J(w,b) \\ tmp\_b &= b - \alpha \frac{\partial}{\partial b} J(w,b) \end{aligned}\right\}$$ w, b before they got updated.

then copy value w = tmp_w, b = tmp_b.

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

where w goes into the derivative term $\frac{\partial}{\partial b} J(w,b)$

Incorrect Implementation.

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

we update $w = tmp\_w$

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

$$b = tmp\_b$$

Important: whenever we hear about gradient descent it is the simultaneous updates done on w, b. otherwise it will give incorrect results.

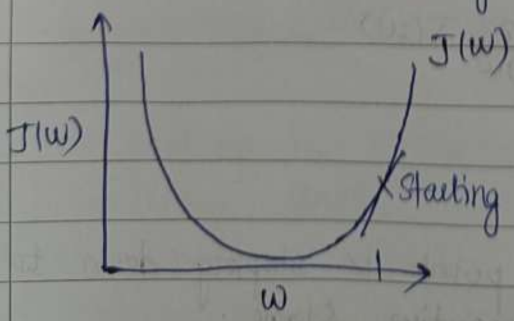
Gradient descent Intuition.

$$w = w - \alpha \frac{\partial}{\partial w} J(w,b)$$ α is the learning rate.

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

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

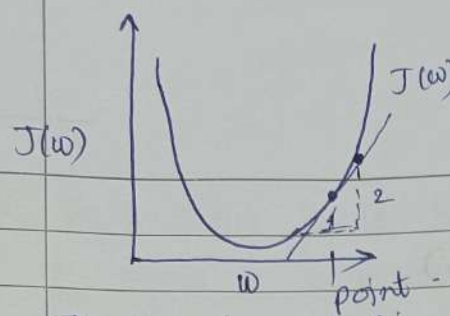
minimize the cost by adjusting parameter w $\min_{w} J(w)$



if $b = 0$

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

Draw a tangent line at the starting point curve. The slope of this line is the derivative of function j at this point



Case(i) on the Right side

__/__/__

If the tangent line is pointing up and to the right The slope is positive.

$\frac{\partial}{\partial w} J(w)$ should be greater than zero.

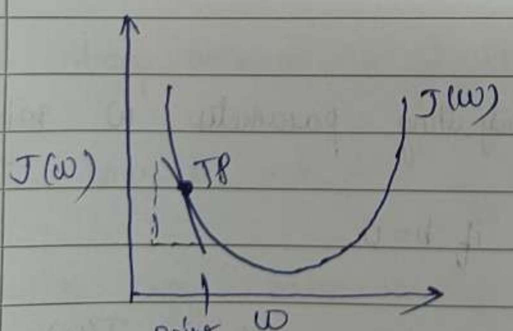
$w = w - \alpha$ (positive number)

The learning rate is always a positive number

We will end up with a new value for w, that's smaller.

On the graph, you are moving to the left, you are decreasing the value of w. You may notice that It is the right thing to do if your goal is to decrease the cost J because when we move towards the left on this curve; the cost j decreases. and you are getting closer to the minimum for j over here.

The Gradient descent seems to be doing the right thing



case (ii) on the left side

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

The Tangent line over point is sloping down to the right, has a negative slope.

The If we draw a triangle, slope is negative, height is negative. -2 and width is 1

_/_/_

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

where $\frac{\partial}{\partial w} J(w)$ is negative < 0.

$w = w - \alpha$ (negative number)
means adding a positive number, so we end up increasing w.

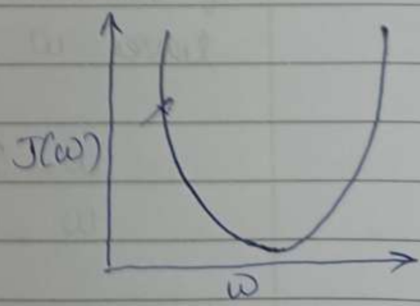
→ If we increase w, that means we are moving to the right side to the graph. J has decrease down to here.

→ It looks like gradient dessent is doing something reasonable. getting you closer to the minimum.

→ <u>Learning Rate α in Gradient Descent.</u>

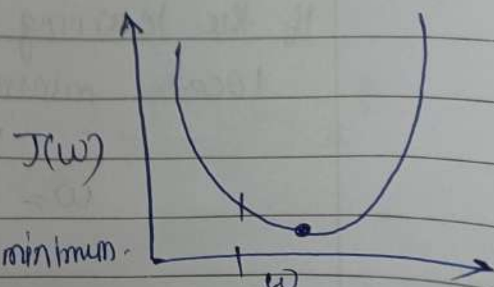
The choice of the learning rate alpha will have a huge impact of the efficiency of the implementation of gradient descent.

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



If learning rate α is small too small

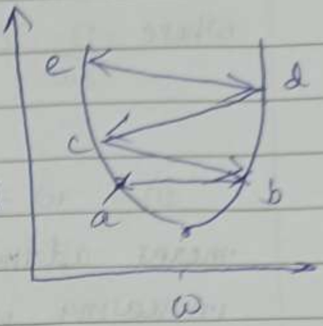
We end up in decreasing the cost J incredibly slowly to approach the minimum
Gradient decent is small.



if learning rate is too large

It is point which is very close to the minimum.

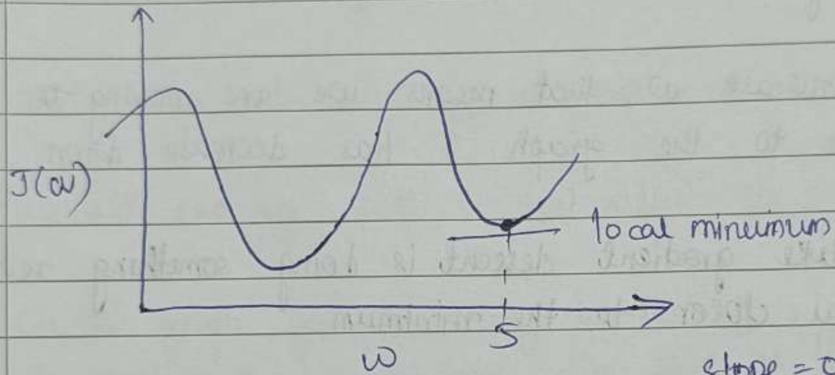
→ If the learning rate is too large, update w very giant step to be all the way over here.



→ It is changing from point a to b.

→ The gradient descent may overshoot $J(w)$ and may never reach the minimum.

→



slope = 0

derivative = 0

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

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

$$w = w$$

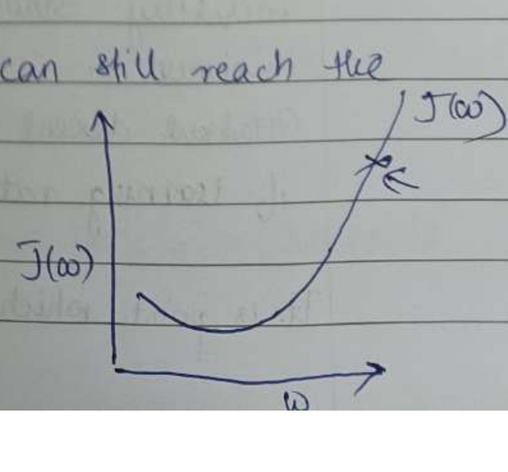
If we have already a local minimum gradient descent leaves w unchanged.

$$w = 5 - 0.1\,(0)$$

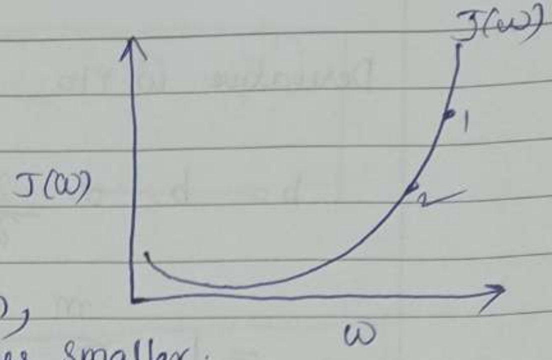
$$w = 5$$

If the learning rate is fixed, we can still reach the local minimum.

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



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



When we reach a local minimum,
the derivative automatically becomes smaller.
Gradient descent will automatically take smaller steps.
Update steps also becomes smaller.

can reach minimum without decreasing learning rate.

Gradient Descent for Linear Regression.

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

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

Gradient descent Algorithm.

→ repeat until convergence.

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

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

The derivative wr to w.

$$w = w - \alpha \frac{\partial}{\partial w} J \frac{d}{dm}(w,b) \rightarrow \frac{1}{m} \sum_{i=1}^{m} \left(f_{w,b}(x^{i} - y^{i}) x^{i} \right.$$

__/__/__

Derivative w.r.to b

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

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

Optional

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

where $f_{w,b}\, x^{i} = wx^{i} + b$

$$= \frac{\partial}{\partial w} \frac{1}{2m} \sum_{i=1}^{m} \left(wx^{(i)} + b - y^{(i)} \right)^2$$

$$= \frac{1}{\cancel{2}m} \sum_{i=1}^{m} \left(wx^{(i)} + b - y^{(i)} \right) \cancel{2}\, x^{(i)}$$

$$\boxed{\frac{\partial}{\partial w} J(w,b) = \frac{1}{m} \sum_{i=1}^{m} \left(wx^{(i)} + b - y^{(i)} \right) x^{(i)}}$$

Partial derivative w.r.to b.

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

$$= \frac{\partial}{\partial b} \frac{1}{2m} \sum_{i=1}^{m} \left(wx^{i} + b - y^{(i)} \right)^2$$

$$= \frac{1}{\cancel{2}m} \sum_{i=1}^{m} \left(wx^{(i)} + b - y^{(i)} \right) \cancel{2} = \frac{1}{m} \sum_{i=1}^{m} \left(wx^{i} + b - y^{i} \right)$$

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

Gradient descent Algorithm

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

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

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

where we update the derivative of , values of w and b simultaneously.

When using a squared error cost function with linear regression. The cost function doesnot and will never have a local minima. It has a single global minimum. because of the bowl shape.

The cost function is a convex function.

Training Linear Regression

Batch Gradient Descent: Each step of gradient descent uses all the traning examples.

m = total no of traning examples. for each update.