$Gradient\ Descent\ algorithm$

$Gradients:\ \ difference$

$x_1\ and\ \ x_2\ = x_2 - x_1 =\ \Delta x\ =\ the\ change\ in\ x$

$y_1\ and\ \ y_2\ = y_2 - y_1 =\ \Delta y\ =\ the\ change\ in\ y$

$(x_1, x_2)\ and\ \left(y_1, y_2\right)\ \ the\ slope:$

$$slope\ = \frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x} = \frac{the\ rate\ of\ change\ in\ y}{the\ rate\ of\ change\ in\ x} = \frac{dy}{dx}\ =\ m\ =\ tan(\theta)$$

$tan(0) =\ 0\ \ \ No\ slope$

$tan(45) =\ 1\ \ \ Littile\ slope$

$tan(90) =\ inf\ \ \ max\ slope\ \ ,\ imagine\ you\ have\ a\ bike\ \ you\ cant\ drive\ vertical$

$Imagine\ you\ are\ trying\ to\ find\ the\ minimum\ point\ of\ \ y\ = x^2$

$y\ = x^2\ \ \ graph$

$I\ assume\ that\ \ at\ \ x\ =\ 4\ \ ,\ y\ = x^2\ \ might\ have\ minimum\ point$

$No\ it\ is\ wrong$

$why\ ?:\ \ \ My\ omkar\ sir\ told\ if\ any\ equation\ has\ minmum\ point\ ,$

$\qquad the\ slop\ of\ the\ equation\ at\ particular\ point\ =\ 0$

$y\ = x^2\ \ \ slope$

$\frac{dy}{dx} = y'\ =\ 2x\ \ =\ 2\ *\ 4\ =\ 8$

$jr\ bikas\ understood\ \ x = 4\ \ is\ not\ a\ minimum\ point$

$x = 4\ +\ \ or\ \ 4\ -\ \ \ ====\ \ >uncle\ zubair$

$For\ the\ first\ assumption:\ \ y = x^{2}\ \ slope$

$\frac{dy}{dx} = y^{'} = 2x\ = 2 * 4 = 8\ \ \ (+\ ve)$

$x_{new} = x_{old} -$

$Goal:\ \ Need\ to\ find\ minmum\ point$

$Rule:\ \ at\ minmum\ point\ slope\ of\ the\ equation\ = 0$

$Case - 1:\ \ \ assume\ \ x = 4\ \ is\ Minimu\ point$

$It\ is\ wrong:\ \ \ y^{'} = 2x = 2 * 4 = 8 != 0$

$Case - 2:\ \ x_{new} = x_{old}\ \ +\ \ \ or\ \ \ -$

$In\ case - 1\ \ slope\ value\ \ is\ (+\ ve)\ \ decrease\ the\ values$

$slope\ value\ \ is\ (-\ ve)\ \ increase\ the\ value$

$Case - 3:$

$x_{new} = 4\ \ -\ \ \ slope\ value$

$x_{new} = 4\ \ -\ \ 0.2 *\ 8 = 4 - 1.6 = 2.4$

$slope\ = y^{'} = 2 * x = 2 * 2.4 = 4.8$

$x_{new} = x_{old} -\ \ 0.2 *\ slope$

$x_{new} = 2.4 - 0.2 * 4.8 = 2.4\ - 0.96\ = 1.44$

$slope\ = y^{'} = 2 * x = 2 * 1.44 = 2.88$

$x_{new} = 1.44 - \ 0.2 * 2.88 = 0.864$

$4, \ ==== \ > 2.4 === \ > 1.44 === \ > 0.8 ===== \ > 0$

$x_{new} = x_{old}$

$x_{new} = x_{old} - \ 0.2 * slope$

$Iteration - 1: \quad x = 4$

$calculate\ slope\ at\ this\ point = y^{'} = 2 * x = \ 2 * 4 = 8$

$x_{new} = 4 - \ 0.2 * 8 = 4 - 1.6 = 2.4$

$Iteration - 2: \quad x = 2.4$

$calculate\ slope\ at\ this\ point = y^{'} = 2 * 2.4 = 4.8$

$x_{new} = 2.4 - \ 0.2 * 4.8 = 2.4 - 0.96 = 1.44$

$Iteration - 2: \quad x = 1.44$

$calculate\ slope\ at\ this\ point = y^{'} = 2 * 1.44 = 2.88$

$x_{new} = 1.44 - \ 0.2 * 2.88 = 1.44 - 0.576 = 0.86$

$x_{new} = x_{old} - \ learning\ rate * slope$

$\alpha = learning\ rate = (0, 1)$

$Neural\ network: \ Will\ find\ the\ error$

$Goal: \ Need\ to\ Minimise\ the\ cost\ function(J)\ by\ providing\ the\ suitable\ weights$

$Minimise\ the\ cost\ function = slope\ of\ the\ cost\ function$

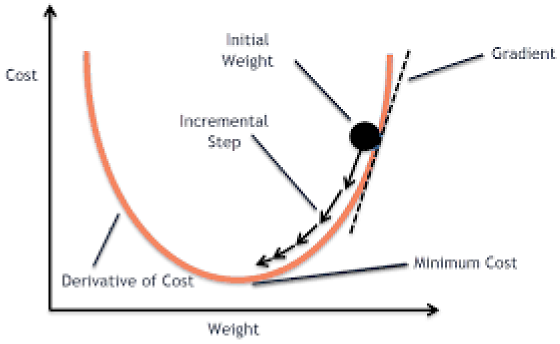
$$\frac{dJ}{dw} = 0$$

$$w_{new} = w_{old} - \ learning\ rate\ * \frac{dJ}{dw_{at=w_{old}}}$$

$$b_{new} = b_{old} - \ learning\ rate\ * \frac{dJ}{db_{at=b_{old}}}$$

$$w_{new} = w_{old} - \ \alpha\ * \frac{dJ}{dw_{at=w_{old}}}$$

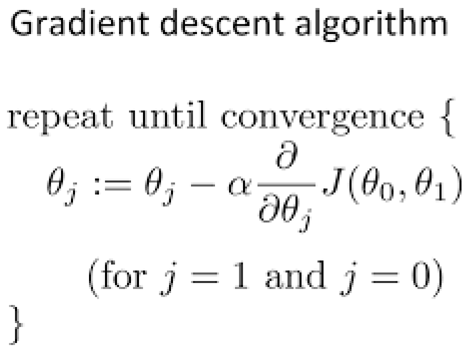
$$\frac{dJ}{dw_{at=w_{old}}} = 0\ then\ w_{new} = w_{old}$$



Sir to what we call learning rate i didn't get network issue plz sir just explain again

Learning rate :

$$w_{new} = w_{old} - \ 0.001\ * \frac{dJ}{dw_{at=w_{old}}}$$



Gradient descent algorithm

```
repeat until convergence {
    $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$
        (for $j = 1$ and $j = 0$)
}
```

Convergence: *reach the minmum point*

1) *Batch Gradient descent*

Imagine you have an image size is 28*x*28

$H1$: 128N

$H2$: 128N

$H3$: 128N

O: 10N

$I\ (784) - H1(128) - H2(128) - H3(128) - O(10N)$: *Output*

Calculate the error

Go back by update the weights

Forward Prpogation + *Backward prpogation* : 1*Min*

100 *Images are there* ===== *at a time*

Im asking to 100 *students*, *go down and come up* :

1 *training iteration is completed* = *epoch*

100 *members go down and come up* : 1 *epoch* : *accuracy* *mean square error*

I will get error afer 100 *mins*

I will get a chance to do next iteration after 100 *mins*

I will get a chance to update my weights after 100mins

Analogy: pass X_{test} to DT.predict(X_{test})

one epoch will not give desired answer:

starting we intilaise weights randomly

at least 10 epochs we required

For the entire process: $100 * 10 = 1000min$

2) Mini batch Grdaient descent

100 members divided into 10 batches

Mini batch Gradient descent

first 10 mebers will go : $F + B$

will calculate 10 members avg error, we try to update the weights

after 10 mins only, we started to update the weights

Next bactch 10 members use the new updated weights

will calculate the error of 10 m, update the weights

after 10 batch Forward + Back = 1 epoch

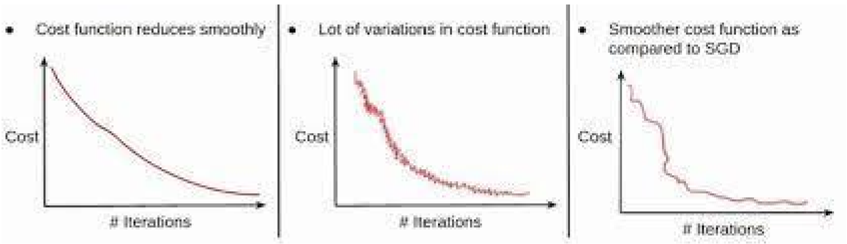
3159/3159
1/3159
2/3159
3159/3159

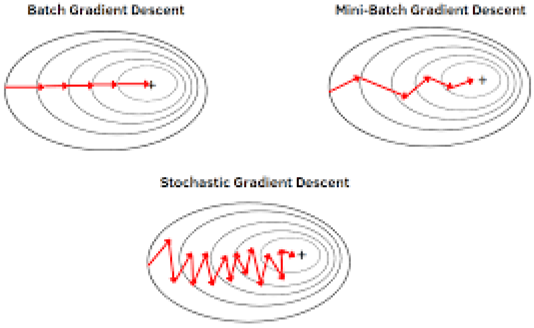
3rd batch pepole

3) Stochastic gradient descent:
every time one input will go and will calculate the error and it come back by update the weights

Error Graphs

For every epoch error will be decrease





Saturday 9 to 10 exam : Statistics

10 to 1 : NLP class

10 to 1 : NLP class

Exponential weighted averages