BATCH GRADIENT DESCENT / STOCHASTIC GD / MINI BATCH GD

Depending upon the amount of data we feed in our neural network there are three types of Gradient Descent.

① Batch Gradient Descent:

Suppose we have 50 data in our dataset Batch Gradient Descent ma paila sabai 50 data ko corresponding ko $\hat{y}$ calculate garinxa tespachi loss calculate garinxa ani Ek choti parameter update garinxa.

Pseudo code:

epoch = 10
for i in range (epoch):
    $\hat{y} = np.dot(X, W) + b$ [X is whole row]
    ∴ In $\hat{y}$ we will have 50 values of predictions
    now, we calculate loss for 50 values
    ∴ loss = $\sum_{i=1}^{50} (y_i - \hat{y}_i)$
    After calculating loss we update parameters.

$$\therefore W_{new} = W_{old} - \alpha \frac{dl}{dW}$$

$$b_{new} = b_{old} - \alpha \frac{dl}{db}$$

After this we again repeat this process with new updated weights (W) and bias (b)

Paila sabai data(row) ko respective $\hat{y}$ predict garinxa ani (loss calculate garinxa sabai row ko sum garera) ani parameters update garinxa (this for 1st epoch)

feri arko epoch ma previous epoch ma aayeko updated parameters rakhera sabai row ko $\hat{y}$ calculate garinxa, loss calculate garinxa ani feri parameters update hunxa

(This process continues up to how many epochs we are giving) yesari nai loss minimize hudai janxa.

Note: Batch gradient Descent ma jati no of epoch diyeko xam teti choti matra weights and bias update hunxa.

like => epoch = 10
↓
10 choti weights ra bias update hunxa

② STohastic Gradient Descent

Suppose we have 50 data in our dataset. STohastic Gradient Descent ma paila randomly euta row select hunxa ani tesko $\hat{y}$ predict garinxa and tyo row ko hune loss calculate garinxa ani weight and bias update garinxa.

ani feri arko random row liginxa tesko ne $\hat{y}$ predict garinxa paila updated parameters use garera ani loss calculate gari weight and bias update garinxa.

yesari nai total row na sakinjel samma garinxa ani euta epoch sakinxa.

PSeudo code:

```
epoch = 10
for i in range (epoch):

    Suffle the dataset
    for j in range (x.shape[0]):
                    Total rows = 50

⇒ J ma random row aako xa
⇒ ŷ = calculate garinxa (feed forward garera) euta row ko
⇒ loss calculate garinxa (y - ŷ) euta point ko
⇒ w, b update garinxa
avg loss paint garinxa inner loop sakxe xi (for each epoch)
∴ Avg loss = ((ŷ1 - y1)² + (ŷ2 - y2)² + (ŷ30 - y30) + (ŷ50 - ŷ50) .....)
                        Total no of rows (50)
```

Note: STohastic Gradient ma (no of rows X epoch) times parameters update hunxa

like :-

epoch = 50

no of rows = 10

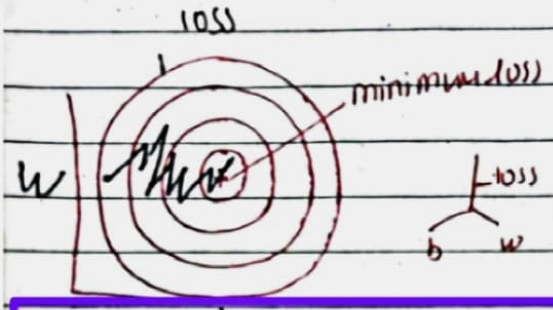
paila random 10 ota ^row bata ek ek ota row ligera weights update garinxa ani yo 50 epochs ko lagi garda 50x10 = 500 times update

③ Mini batch - Gradient Descent:

We already discussed about it during back propagation

Let's taks about properties and facts:

Stochastic



Batch

* which is faster to run (given same no of epoches)?
(give output)
=> Batch (Because it only updates the parameter as much the no of epochs given) but Stochastic updates (no of epoches x no of rows) times so, for Stochastic there is more computation to update weights so, Batch compiles(run) fast.

* which is faster to converge the loss (given same no of epoches)?
-) Stochastic (harek row lai ligera update garxa weight lai so, chadai yei pugxa convergente ma) batch ley sabai row lai ligera loss nikale update garxa yo bistari converge hunxa kina ki weights ne rapid way ma update hunna.

Stochastic gradient descent shows random behaviour while minimizing loss as can be seen from contour plot
yo random point bata suru hunxa ra randomly move garxa so, it prevents us from sitting in local minima. hami local minima ma na fasi global ma ne pugna sakxam as it starts from random point and move randomly
yesko demerit chai yo exactly global minimum ma pugdena tara closely pugxa as seen from Experiments in graphs.