Batch Gradient Descent

Agar 5 Epochs hai tou bss 5 Times Weights & Biases ko Update karenge

Weight Update utne baar hie hoga Jitne baar Epochs hai.

Stochastic Gradient Descent

SGD jo hai

Epoch = 10 hai

Epoch jitna hai utne baar 1 Loop chalega

Then uske andar 1 Loop chalega Jitne Rows honge utne baar

Basically Har Row hie yaha 10 baar update hoga

So if 50 Columns hai tou

*50*10 mtlb → 500 Times Loop chalega.*

In SGD Frequency of Weight Update is Higher, & Random shuffling bhi hota hai for removing bias.

Batch GD har Epoch mai saare ke saare data point *keliye 1 baar weight update karta hai*

Whereass

SGD Jo hai vo har Epoch ko Number of Row Times update krta hai

Basically jitna epoch hai utne baar har ek row update hoga

Questions

Which is Faster (Given some number of Epochs)?

→

Batch will be Faster in completing then SGD

As Batch will update weights the Epoch times

But SGD will Update weights based on

Epochs * Number of Rows.

```
(X_train,y_train,epochs=10,batch_size=320)
```

*Agar **batch_size mai == (No. of Rows)** diya*

Then it will do Batch GD.

*& If You give **batch_size = 1** then it will run SGD.*

Which is Faster to Convergence(most optimal solution tak kaun pochega sbhse pehle) given same number of Epochs ?

→

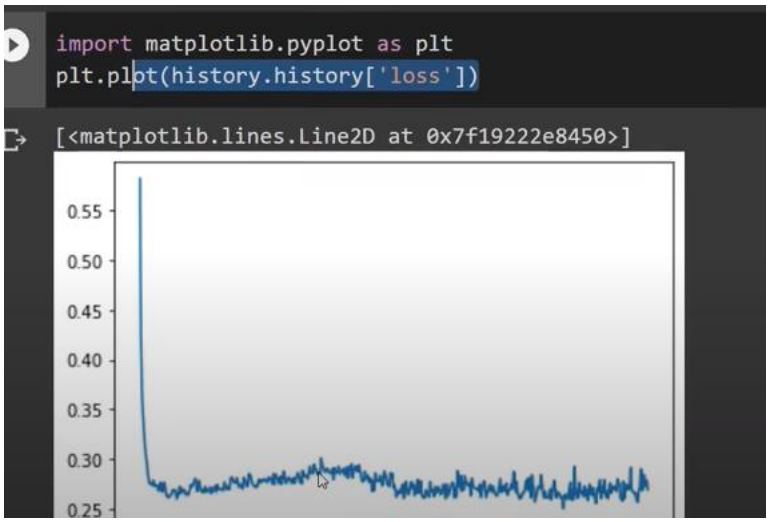
SGD will be faster to convergence, as it will Update Weights more number of time.

So SGD ko itne epochs ka zarrurat hie nahi padega converge karne jitna epochs Batch ko lagta hai.

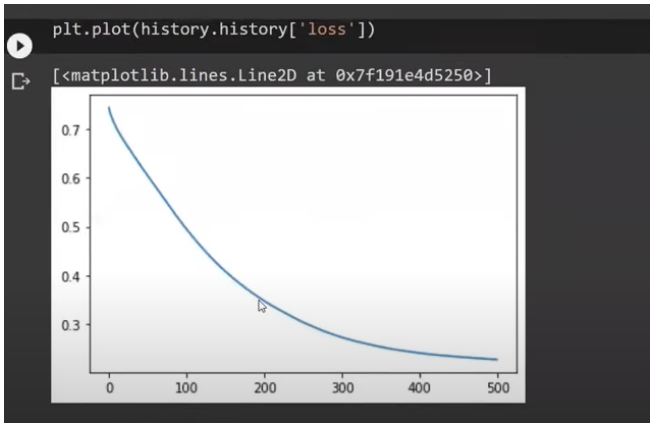
Time Kitna lagrha usme Batch faster hai

But Solution tak pochnemai SGD faster hai

The Difference of Instability at Time of Decreasing Loss

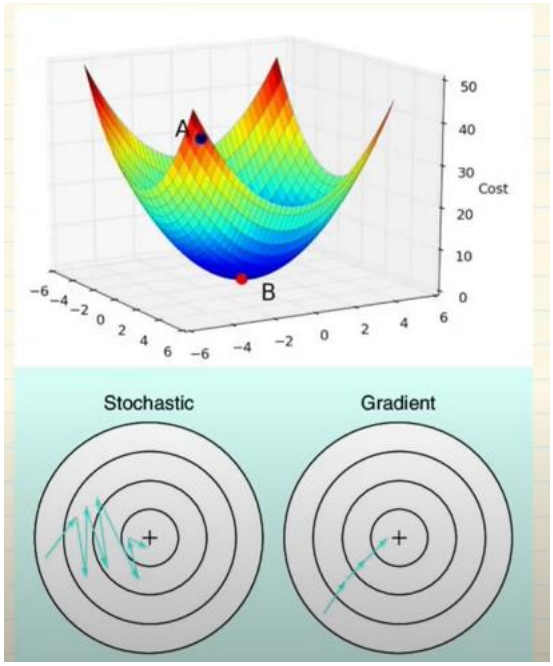


In SGD you can see there is Some instability at decreasing time of Loss Function.



While in Batch GD the decrease is very stable.

Let's understand why this happen like that



SGD ese chalega jaise sharab pii rkhi hai usne,

Jo Spiky behavior ya unstability jo hai SGD mai is it usefull?

Answer is Yes or No

1 Benefit & 1 Disadvantage

Benefit-> *SGD can move out of Local Minima & can reach Global Minima. Becoz of its Random Nature.*

Whereas Batch GD will get stucked in Local Minima.

Disadvantage:

Exact Solution nahi millta yaha, Yahape Approximate Solution milta hai.

Mini Batch GD

Isme Hum Jitne batches banayenge

Utne hie Number of Epochs baar update hoga

*Mini Batch Value * Epochs*

Hum Group banake run krte

Example:

Epoch is 10

& batch_size = 100

Rows are 500

So

Number_ of_Batches = Batch_size/Rows

Number_ of_Batches =500/100

Number_ of_Batches = 5

*Epochs * Number_ of_Batches*

*10 * 5 = 50 times chalega.*

In General Deep Learning mai zyaada tar time Mini Batch GD use krte hai.

Questions

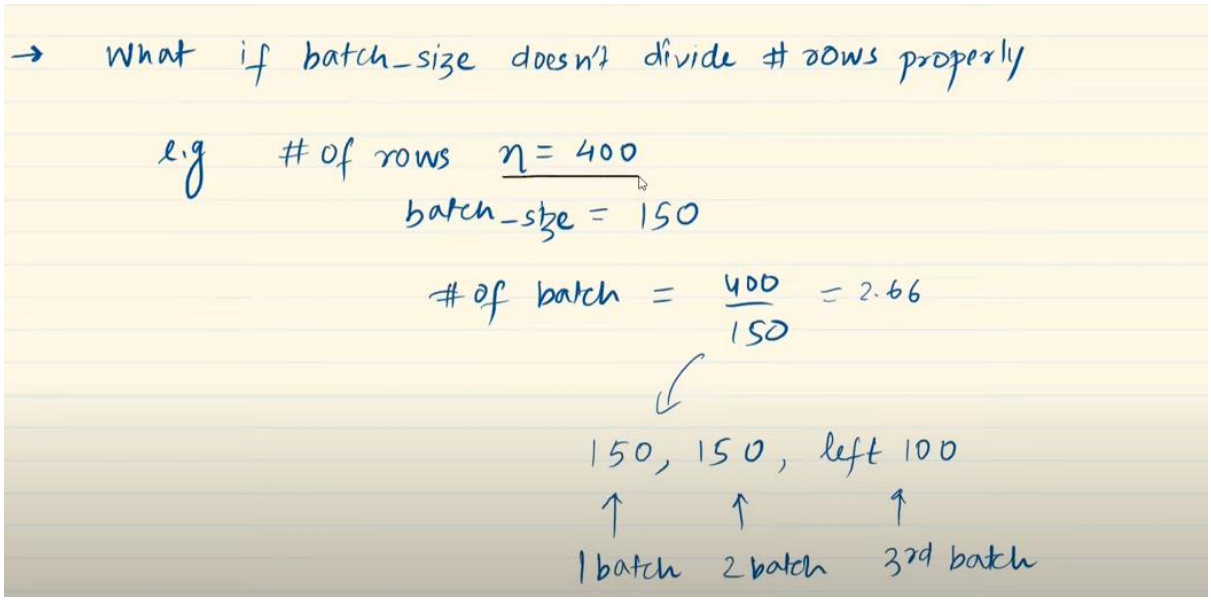
Q. Why Batch Size is provided in multiple of Two?

→

To use RAM Effectively,

Humara RAM is typically Designed in a way to handle Binary values, So basically Becoz of that

If we give Multiple of 2, then we make use of RAM Faster.



Now General Theory of All From Web!

BATCH GRADIENT DESCENT

Batch gradient descent, also called vanilla gradient descent, calculates the error for each example within the training dataset, but only after all training examples have been evaluated does the model get updated. This whole process is like a cycle and it's called a training epoch.

Some advantages of batch gradient descent are:

its computational efficiency: it produces a stable error gradient and a stable convergence.

Some disadvantages are that:

the stable error gradient can sometimes result in a state of convergence that isn't the best the model can achieve.

It also requires the entire training dataset to be in memory and available to the algorithm.

In batch gradient descent, the update of model parameters or weights is calculated using the entire training dataset at once. This means that for each step in the training process, the algorithm calculates the gradient of the cost function for the entire dataset.

This algorithm is often used when the training dataset is relatively small and can fit into memory comfortably. It's also suitable for less complex models where the cost of computing the gradient for the whole dataset is not prohibitively high. For example, consider a linear regression model where you're predicting housing prices based on features like size and location. If your dataset is small, you could use batch gradient descent to train your model, updating the weights based on the error calculated from the entire dataset.

It is a greedy approach where we have to sum over all examples for each update.

$$w = w - \alpha \nabla_w J(w) \tag{6}$$

Advantages of Batch GD:

a. Less noisy steps
b. produces stable GD convergence.
c. Computationally efficient as all resources aren't used for single sample but rather for all training samples

Disadvantages of Batch GD:

a. Additional memory might be needed.
b. It can take long to process large database.
c. Approximate gradients

STOCHASTIC GRADIENT DESCENT

SGD is particularly useful when dealing with large datasets that cannot fit into memory. It's also favored when training complex models, as it can handle the high computational cost more efficiently. Additionally, the stochastic nature of SGD can help the model escape local minima, potentially leading to better solutions. For example, in a deep learning model for image classification with a massive dataset, using SGD would be practical. Here, the model parameters are updated incrementally as each image (or a small batch of images) is processed, enabling the model to learn progressively without the need to load the entire dataset into memory.

In this, learning happens on every example:

$$w = w - \alpha \nabla_w J(x^i, y^i; w)$$

- Shuffle the training data set to avoid pre-existing order of examples
- Partition the training data set into m examples.

Advantages : —

a. Easy to fit in memory
b. Computationally fast
c. Efficient for large dataset

Disadvantages :-

a. Due to frequent updates steps taken towards minima are very noisy.
b. Noise can make it large to wait.
c. Frequent updates are computationally expensive.

MINI-BATCH GRADIENT DESCENT

Mini-batch gradient descent is the go-to method since it's a combination of the concepts of SGD and batch gradient descent. It simply splits the training dataset into small batches and performs an update for each of those batches. This creates a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent.

Common mini-batch sizes range between 50 and 256, but like any other machine learning technique, there is no clear rule because it varies for different applications. This is the go-to algorithm when training a neural network and it is the most common type of gradient descent within deep learning.

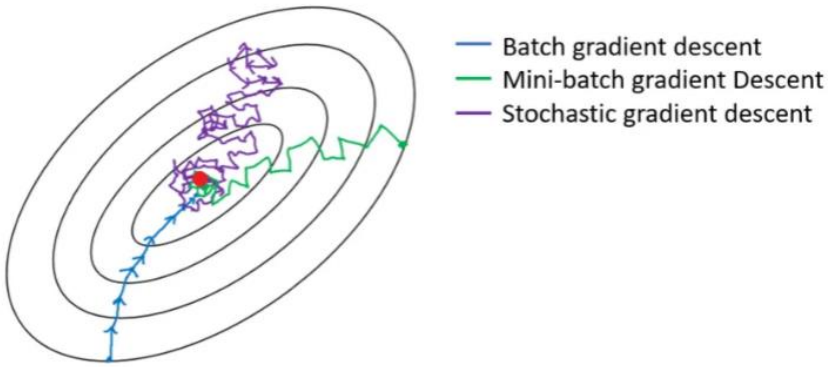
Instead of going over all examples, Mini-batch Gradient Descent sums up over lower number of examples based on the batch size.

It is sum of both Batch Gradient Descent and Stochastic Gradient Descent.

$$w = w - \alpha \nabla_w J(x^{[i:i+b]}, y^{[i:i+b]}; w) \qquad (7$$

Advantages :-

a. Easy fit in memory.
b. Computationally efficient.
c. Stable error go and convergence.



Strategy	no. of steps in each epoch
Batch GD	1
Stochastic GD	N
Mini Batch GD	N/B

N: Number of Rows

B: Number of Mini Batch Size