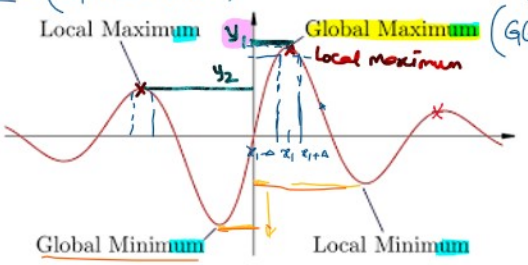
Types of GDA

03 August 2024 20:13

Ouk (LG: Local Gunda) Global Gunda (GG is known as DON)



minima → more than one point of minimum values
vs
minimum

© Story

career roadmap for a DON

Tea stall → ₹ → don't pay. → Beat up → stab → let him rot

Stochastic Gradient Descent (SGD) → an optimization algorithm

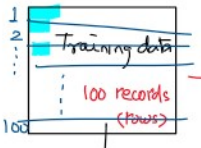
↳ refers to systems or processes that are random or probabilistic

LR $y = \beta_0 + \beta_1 x$

NN: weights & biases.

In the context of machine learning / AI, stochastic describes algorithms or models that incorporate randomness in their operation

SGD: SGD is an optimization algorithm used to minimize the loss function. To do so, SGD computes the gradient for a single randomly chosen training data at each iteration



Leveraging SGD, gradient is computed for each and every sample in the dataset → hence it updates every sample in the dataset

Working of SGD:

→ #epochs = 100

100 x 100 = 10,000.

a) Initialization

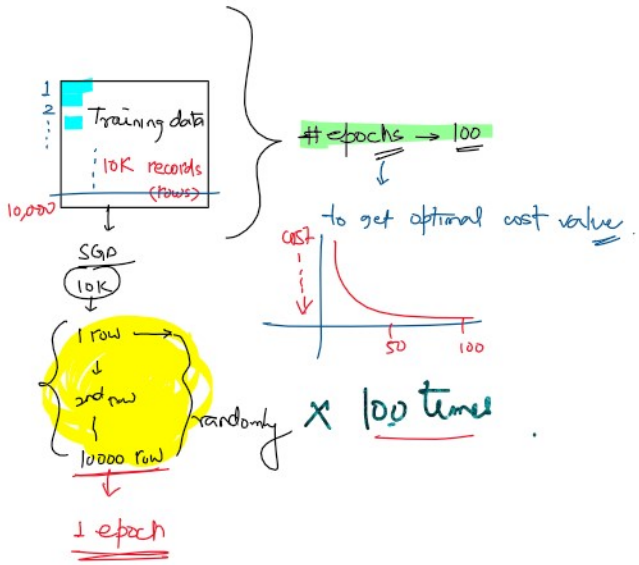
- Initialize the model parameters (weights and biases) with random values or zeroes.
- Set the learning rate (α or η)
  - set by modelers
  - α: (0.01 — 0.2) $\alpha = [0.01, 0.1, 0.15, 0.2]$
- Define the no of epochs (iteration over the entire dataset]

In SGD — training data : 10,000 records

epochs # 100.

How many times → model will update the weights & biases

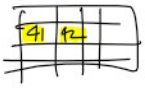
Ans: For SGD: $100 \times 10{,}000 = 1 \times 10^6 \Rightarrow 1$ million times.

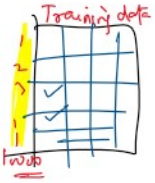


b) Iteration

- shuffle the training dataset to ensure the random sampling
- For randomly selected single training data point (row):
  - # Compute the gradient of the loss function
  - # Update the model parameters (weight & bias) using the computed gradient and learning rate

c) Updation rule :





For $j=0$

$$\theta_0 := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} [J(\theta_0, \theta_1)]$$

↓ Input → by modeler

Vanilla → No flavour
'Plain'

x 100 } time taking

[datapoint / raw record / training example]

Quality Testing
↓
Light bulb →  → 10,000 / day
↓
definite batch size
Fixed Limit → 100 → 10% → Test scripts → 10 bulbs → 1 bulb

Batch Gradient Descent (BGD)

Batch Gradient Descent (BGD) computes the gradient of the loss (cost) function w.r.t. the parameters using the entire training dataset

batch

In BGD, all the training data is taken into a consideration to take a single step

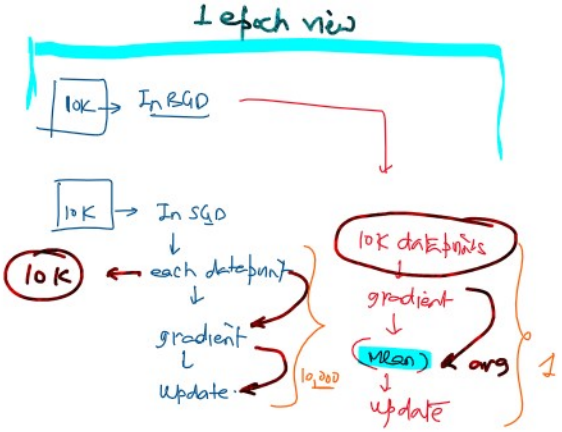
↓

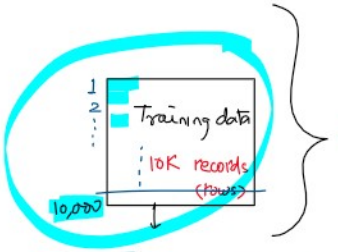
we would take the average of the gradients of all the training datapoints and use the mean gradient (one number) / epoch to update the model parameters

↓

Just one step of gradient descent in one epoch.

1 epoch view



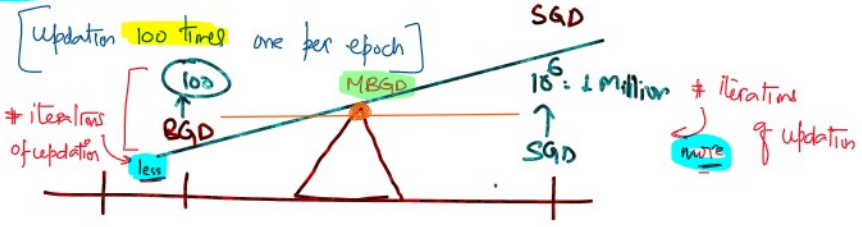


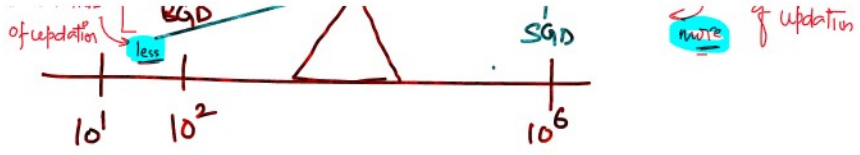
epochs → 100

↓

to get optimal cost value

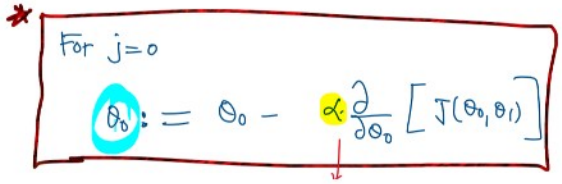
[Updation 100 times one per epoch]





Vanilla → BGD : Entire dataset

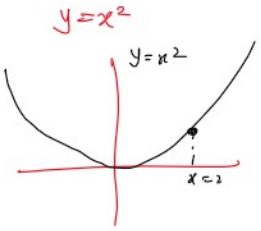
learning rate → Input for Updation Formula



$2 - 0.01 \times 0.2$

$2 - 0.002$

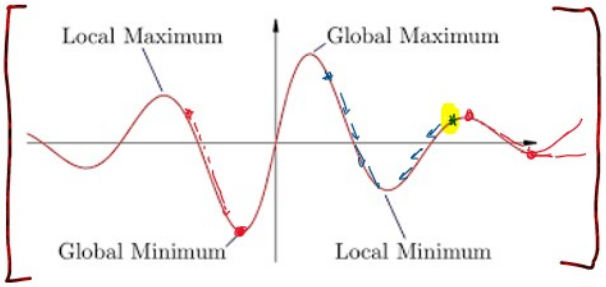
2-0.002=1.998



$\frac{dy}{dx} = 2x$

$\left.\frac{dy}{dx}\right|_{x=2} = 2 \times 2 = 4$.

Newton - Raphson's Method



Task: Understand the importance of initializing GDA.

Mini-Batch Gradient Descent (MBGD)

MBGD is a trade-off between Batch Gradient and stochastic Gradient descents.

Training data: 10,000 data records

batch_size = 32 data records.

No. of batches: 10000/32=312.5 ≈ 313 ⇒ 313 batches →

Given # epochs=100

100 records

batch_size = 5

5

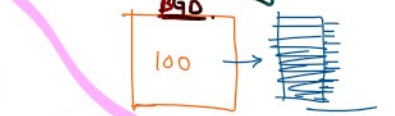
(one epoch)

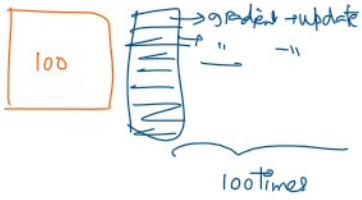
Q: For a dataset of 100 samples, if the batch size is 5, how many times updates will occur for 100 epochs.

Ans- #batches = 100/5 = 20 batches / epoch

20 batches MBGD.

20 × 100 = 2000 updates

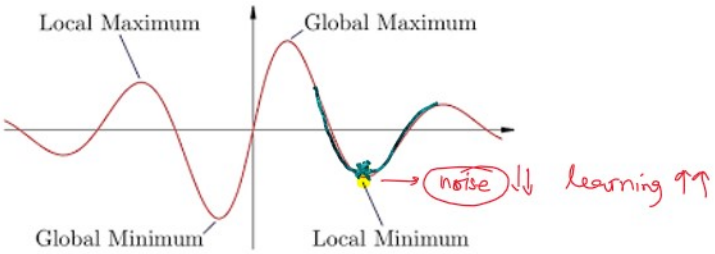




Mean Gradient → 1 time

steps

1. Pick a mini-batch (batch_size = 32)
2. Calculate the mean gradient of the mini-batch.
3. Update the model parameters (weights & biases) using mean gradient
4. Repeat the steps 1-3 for each epoch.

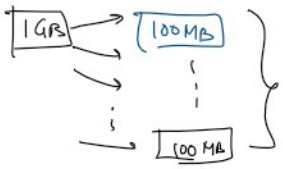


Comparison between SGD | BGD | MBGD

1) Computationally expensive

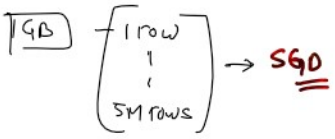
BGD ← 1GB : dataset → 5 million rows

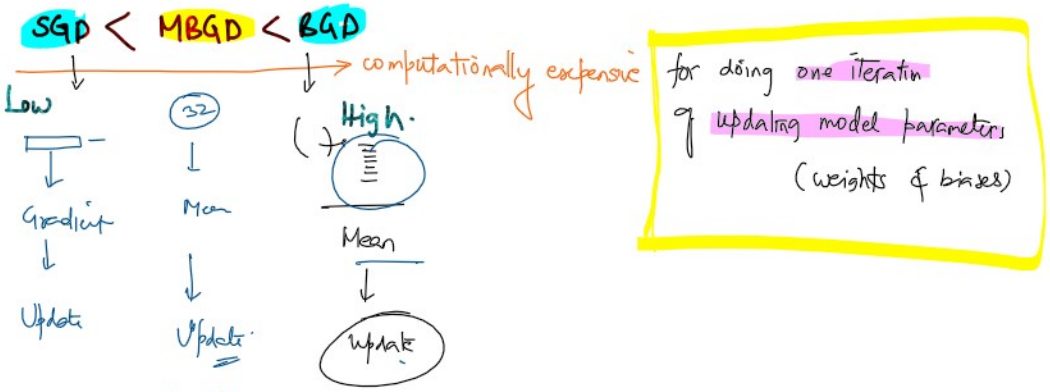
Group by → mean Gradient



MBGD

(same task)





No. of iterations performed in implementing 3 GDAs (refer the Google Colab notebook)

m = 100

epochs = 100

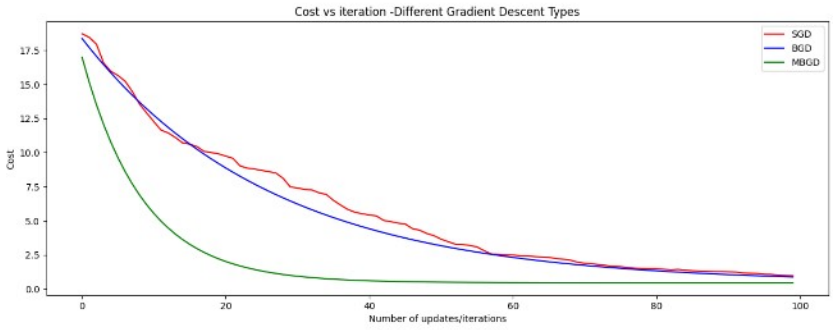
n_batch = $\left\lceil \frac{100}{32} \right\rceil$ = $\lceil 100/32 = 3.125 \rceil$ = 3

SGD # (training records * epochs) = m X epoch = 100 X 100 = 10,000

BGD # = epochs = 100

MBGD # = n_batch * epochs = 3 X 100 = 300

Aspect	SGD	BGD	MBGD
Update Frequency	Once per training example (every row)	Once per epoch (entire dataset/epoch)	Multiple times per epoch (batch_size)
Convergence Speed (default)	Fast	Slow	Fast (than BGD)
Memory Usage	Low (single example)	High (entire dataset)	Moderate (mini-batch)
Update Stability	Less stable (one row)	Very stable	Moderately stable
Computational Cost	Low	High (SGD)	Moderate
Scalability	Good for large datasets	Poor for large datasets	Good for large datasets
Noise in Updates	High (more noisy)	Low (less noisy)	Medium (less noisy than SGD)
Implementation Complexity	Simple	Simple	Slightly more complex



n_batch = 32

right batch size = ?

Hyper parameter tuning