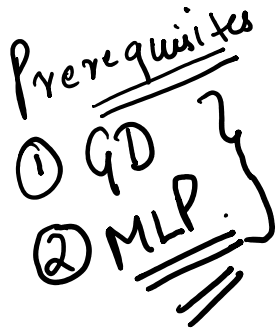


WHAT YOU WILL STUDY IN TODAY VIDEO ?

- What is Optimizer?
- Types of Optimizers
- Formula , Mathematical Working , Use Case of Each Optimizer
- Problems of Each Optimizer

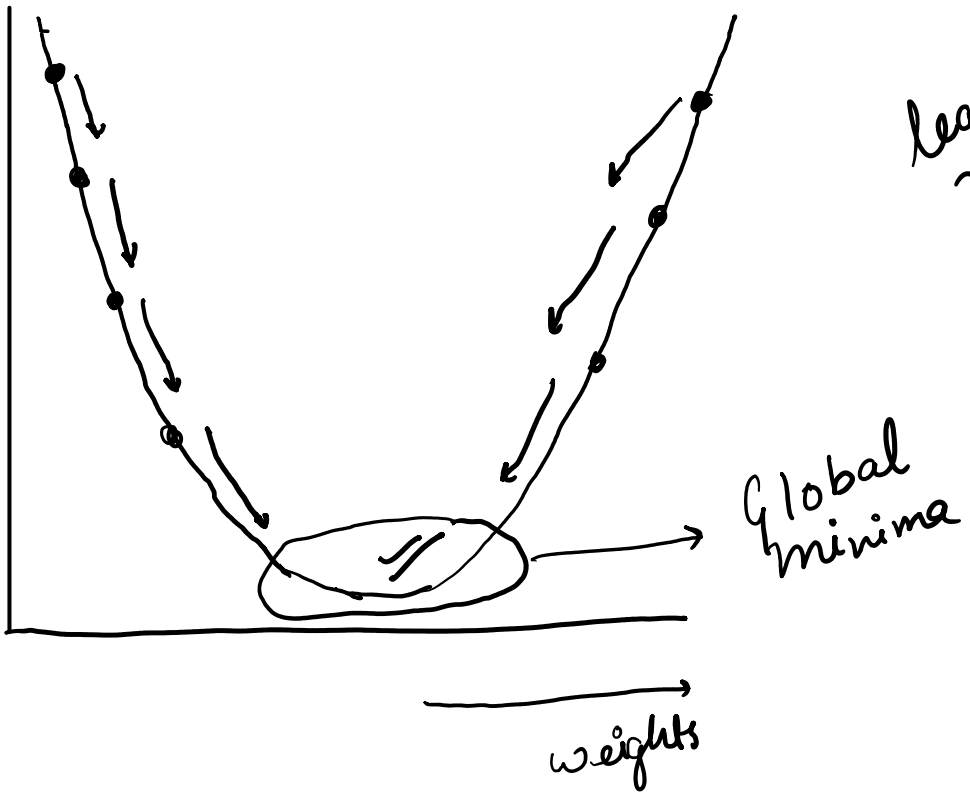
Introduction

algorithms → weight adjust / update

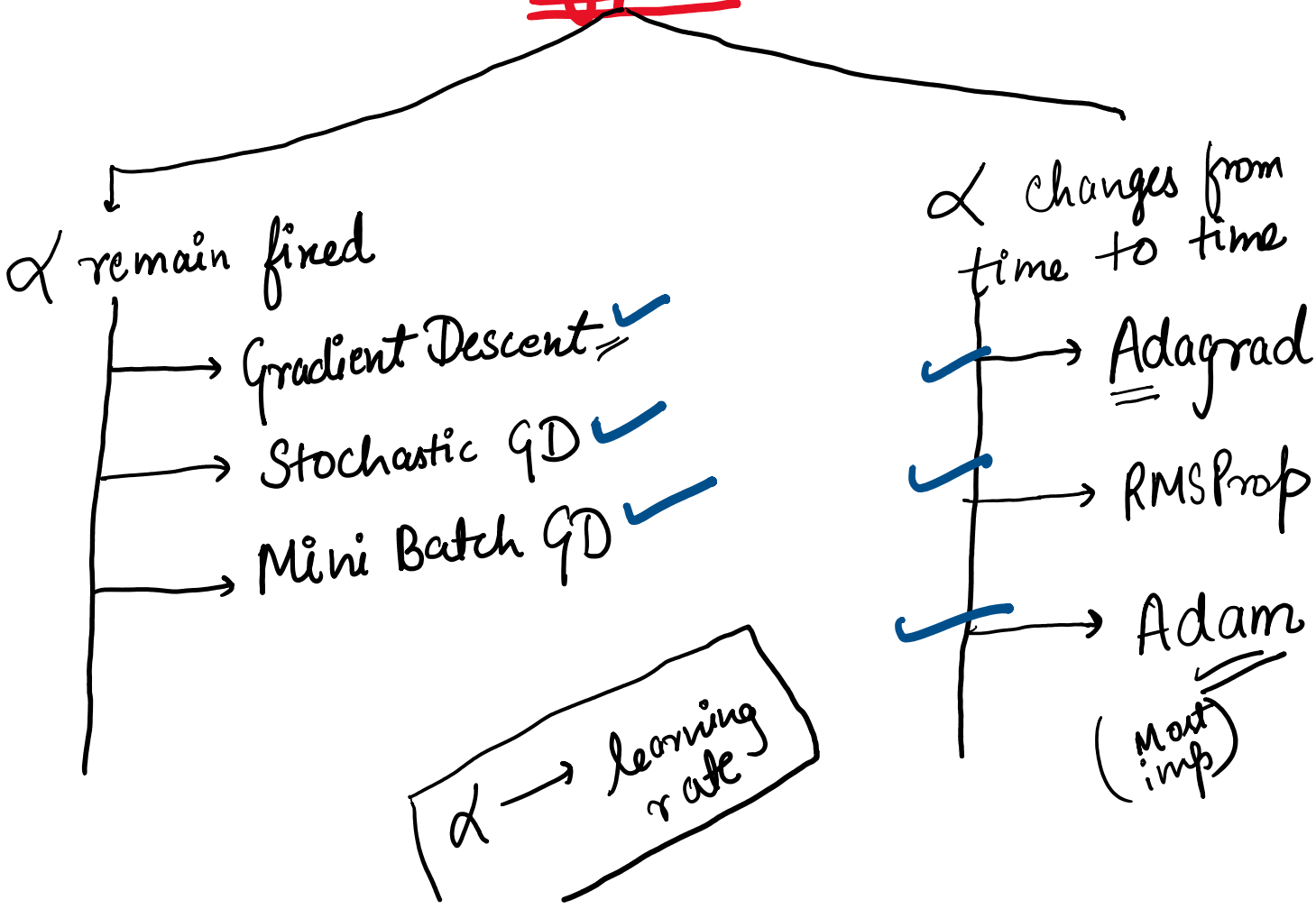
during backpropagation

Convergence

learning rate (α)



Types.


α remain fixed
Gradient Descent
Stochastic GD
Mini Batch GD
α changes from time to time
Adagrad
RMSProp
Adam
(most imp)
α → learning rate

1 GD → Gradient Descent

(Please watch GD video)

formula:

$$w_{new} = w_{old} - \alpha \cdot \frac{dL}{dw}$$

→ (loss gradient of whole dataset)

Working:

Compute loss gradient for entire dataset → Weight updation

Use → Dataset is small
→ So that they can fit in RAM

PC → 4GB RAM
Data → 500 MB
✓

PC → 4GB RAM
Data → 8GB
✗

Problem → Big dataset unable to work.

② Stochastic GD: ⟶ SGD:

formula:

$$w_{new} = w_{old} - \alpha \cdot \frac{dL}{dw}$$

loss gradient for each row of dataset

Working

Comput loss gd for 1st row ⟶ Weight update

2nd row loss gd

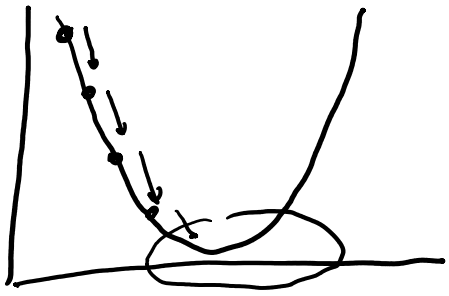
3rd row

Use ⟶ Good for large data
↳ No load on RAM

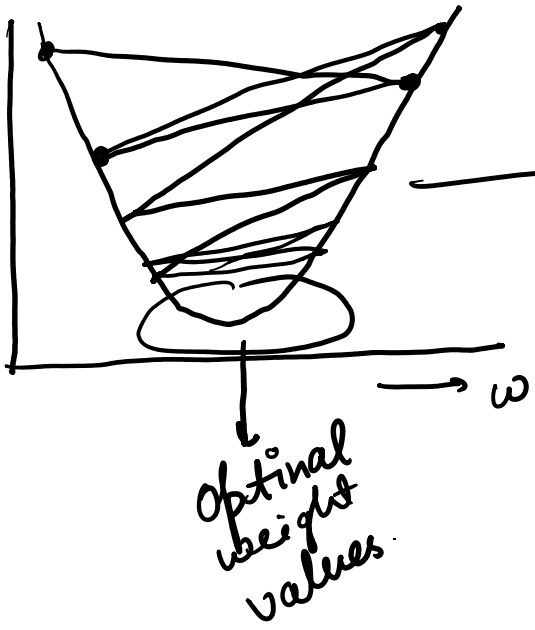
Problem ⟶
① Time ↑
② Zig Zag behaviour
↳ Unstable convergence
③ Overshooting Problem

GD ⟶ Slowly/steadily towards global minima

GD ⟶ slowly/ [illegible] ∪ minima



SGD ⟶



⟶ zig zag behaviour (slower convergence)

③ Mini Batch GD

m → batch size.

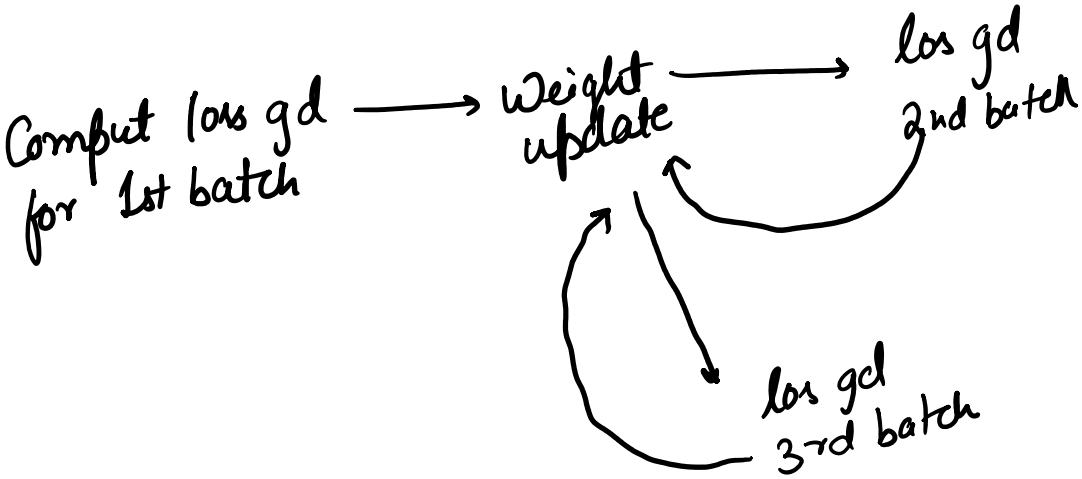
Formula

$$w_{new} = w_{old} - \alpha \cdot \frac{1}{m} \sum_{i=1}^{m} \frac{dL}{dw}$$

↳ loss gradient for a batch

Working

Data → 150 rows
Batch → 3 → 50 row each



Use → Most commonly used GD. of all

Efficient than GD

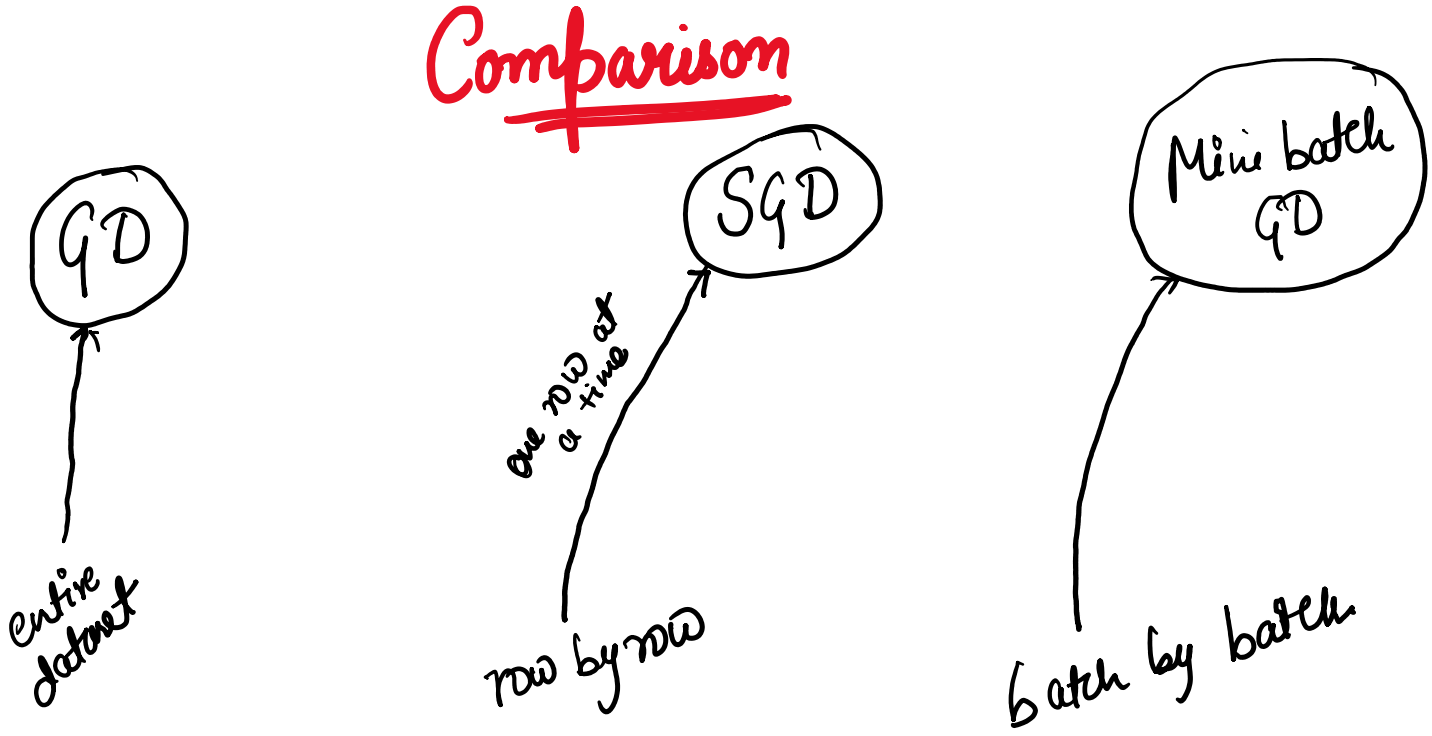
Fast than SGD.

Stable convergence

Converge.

Problem ⟶ find best value of m.

Comparison


GD
entire dataset
SGD
one row at a time
row by row
Mini batch GD
batch by batch

④ Adagrad → Adaptive gradient

Mathematical

3 step →
① Compute gradient
② Sum of squared gradient
③ Update weights

eg Given,

$w = 5$

$\alpha = 0.1$

$\epsilon = 10^{-8}$ → very small (prevent division by 0)

Loss fn $= w^2$

Derivative of loss fn ⇒ $g_t = 2w$

① Iteration 1 ($t=1$)

(a) $g_1 = 2w = 2 \times 5 = \underline{10}$

(b) $G_1 = (10)^2 = \underline{100}$

(c) $w_{new} = w_{old} - \frac{\alpha}{\sqrt{G_t + \epsilon}} \cdot g_t$ (learning rate for this iteration)

$= 5 - \frac{0.1}{\sqrt{100} + 10^{-8}} \cdot 10$

$= 5 \quad \sqrt{100} + 10^{-8}$

$\omega_{new} = 4.9$

② Iteration 2 (t=2)

① $g_2 = 2 \times \omega = 2 \times 4.9 = 9.8$

② $G_2 = 100 + (9.8)^2 = 196.04$

③ $\omega_{new} = 4.9 - \frac{0.1}{\sqrt{196.04} + 10^{-8}} \cdot 9.8$

$\omega_{new} = 4.83$

Do for iteration 3, 4 & so on

Working ① α decrease over time.

Iteration 1.

$\alpha = \frac{0.1}{10}$

$= 0.01$

iteration 2

$\alpha = \frac{0.1}{14}$

$= 0.00714$

decrease

decrease

② Large gradient = smaller weight update

Small gradient = larger weight update

⇒ gradient = large (g_t)

↓

G_t = large

↓

G_t is in denominator

↓

Overall learning rate is also small

↓

weight update is also small

$\frac{0.1}{100}$ → small

Use → Adjusting learning rate =

↳ No need to define lr

Use → Proje... → No need to define ...

→ NLP projects

→ Sparse data [1, 0, 0, 0, 0, 0]

Problem → Deep learning ✗ → Not used

→ Learning rate decay — Main

→ High Computation Cost.

fix → RMSProp

Adam

⑤ RMSProp.

→ (Root Mean Squared Propagation)

X factor → Exponential Weighted Moving Average.

Mathematical

Given: $w = 5$

$\alpha = 0.1$

$\epsilon = 10^{-8}$

$L = w^2$

$g_t = 2w$

$\beta = 0.9$ decay factor

$E|g_t^2| = 0$ → Moving average

game changer

Compute gd

Compute moving avg

Weight update

3 step

① Iteration 1 (t=1)

(a) $g_1 = 2 \times w = 2 \times 5 = \underline{10.}$

(b) $E|g_1^2| = (\beta \times \text{previous moving avg}) + (\alpha \times g_t^2)$

$= (0.9 \times 0) + (0.1 \times 100)$

$= \underline{10.}$

(c) $w_{new} = w_{old} - \frac{\alpha}{\sqrt{E|g_t^2| + \epsilon}} \times g_t$

ω_{new} old $\sqrt{E[g_t^2]}$

$$= 5 - \frac{0.1}{\sqrt{10} + 10^{-8}} \times 10$$

$$= 5 - \frac{0.1}{3.162} \times 10$$

$$\omega_{new} = 1.838$$

② Iteration 2 ($t = 2$)

(a) $g_2 = 2 \times \omega = 2 \times 1.838 = 3.676$

(b) $E[g_2^2] = (0.9 \times 10) + (0.1 \times 3.676^2)$

$= 9 + 1.352$

$= 10.352$

(c) $\omega_{new} = 1.838 - \frac{0.1}{\sqrt{10.352} + 10^{-8}} \times 3.676$

$\omega_{new} = 1.724$

So on...

- How better than Adagrad?

Adagrad → learning rate decay.

0.01 to 0.00714

0.00286

RMSProp → No learning rate decay

0.0316 to 0.03108

0.00052.

Exponential moving average

Use → Deep learning usable

Most popular till 2014 (Adam).

eg. Inception (Google)

eg. Incepti...
(Google)

Problem:

→ There is no momentum in RMSProp

(technique in which optimizer remember past gradients)

Adam has momentum

⑥ Adam.

Combination → Momentum + RMSProp.

Mathematical

Given. $w = 5$

$\alpha = 0.1$

$L = w^2$

$g_t = 2w$

$\epsilon = 10^{-8}$

decay factor $\beta_1 = 0.9$

$\beta_2 = 0.999$

To keep track of previous gradients

① Compute gradient

$g_1 = 2w$

$= 2 \times 5$

$= 10.$

② Compute m (first moment estimate)

$m_1 = \beta_1 m_{t-1} + (1-\beta_1) g_t$

$m_1 = (0.9 \times 0) + (1 - 0.9) \times 10$

$m_1 = 1$

$m_1 = 1$

③ Compute V (second moment estimate)

$$V_1 = \beta_2 V_{t-1} + (1-\beta_2) g_t^2$$

$$V_1 = (0.999 \times 0) + (1 - 0.999) \times 100$$

$V_1 = 0.1$

④ Bias Correction / Normalization $(\hat{m}, \hat{v})$

$$\hat{m} = \frac{m_1}{1-\beta_1} = \frac{1}{1-0.9} = \underline{\underline{10.}} \rightarrow \hat{m}$$

$$\hat{v} = \frac{V_1}{1-\beta_2} = \frac{0.1}{1-0.999} = \underline{\underline{100.}} \rightarrow \hat{v}$$

⑤ Weight updation

$$\omega_{new} = \omega_{old} - \frac{\alpha \hat{m}}{\sqrt{\hat{v}} + \epsilon}$$

$$= 1 \times 10$$

$$= 5 - \frac{0.1 \times 10}{\sqrt{100} + 10^{-8}}$$

$w_{new} = 4.9$

Do for iteration 2 so on...

Summary

1. Compute gradient
2. Compute m → weighted exponential moving avg of past gradient
3. Compute v → moving avg of squared gradients.

RMS prop:

Learning rate adjustment

- Larger gradient → smaller update
- Smaller gradient → larger update

4. Bias $\hat{m}$, $\hat{v}$ → Normalization

④ Bias Correction ⟶ $\hat{m}$, $\hat{v}$

Helps in avoiding slow convergence

⑤ Weight updation

Note ⟶ Variants of Adam

① AdamW ⟶ Adam with weight decay (most used)

② AdaBound

③ RAdam

Use ⟶ Perfect optimizer