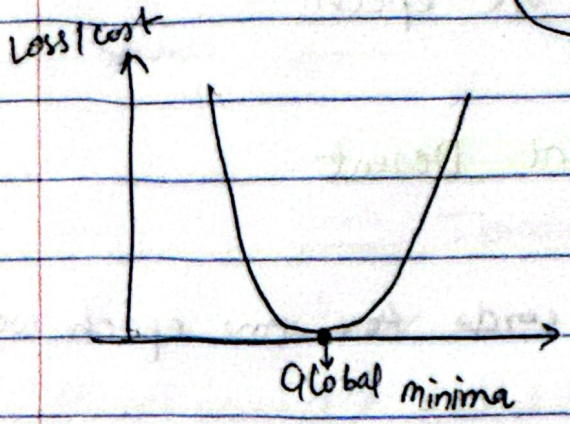
OPTIMIZERS

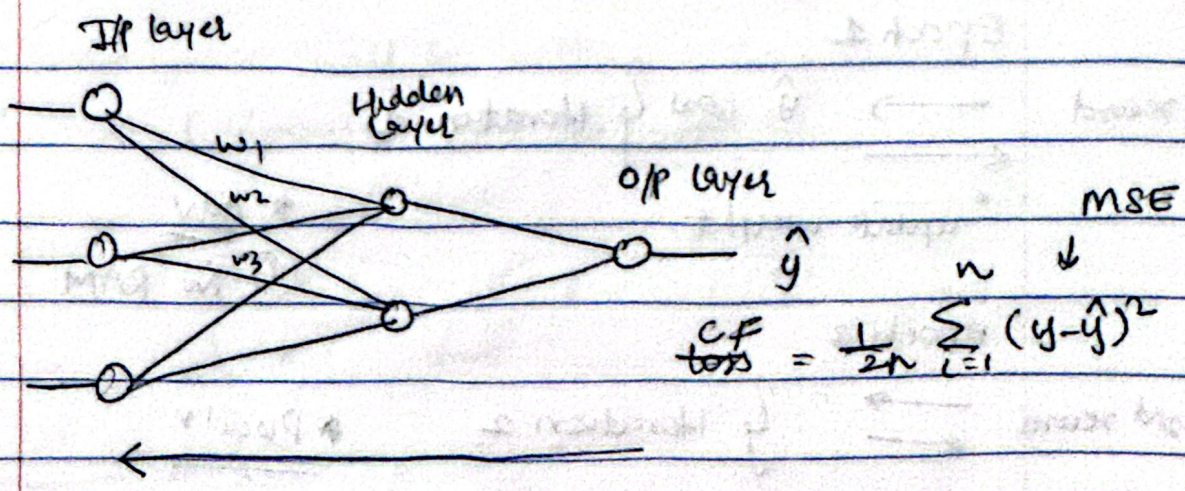
① Gradient Descent

Weight updatn formula.

$$W_{new} = W_{old} - \eta \frac{\partial L}{\partial W_{old}}$$

learning Rate





to minimize loss funct^n we use optimizers.

We send all records in gradient descent
we sent 1 record in epoch is stochastic
we sent in 1000 batch in mini Batch

→ Epoch $\overset{100000}{\rightleftarrows}$ } 1 Epoch

One forward & backward propagation is 1 Epoch.

* disadv of gradient

① Resource extensive
(You'll require lots of ram if you have to include 100000 records in one epoch.

② Stochastic Gradient Desent

For eg. we 1m records for one epoch we pass one

Epoch 1

1 record → ŷ loss } Iteration 1

← update weight

Epoch 2

2nd record → } Iteration 2

⋮

Million Iteration

* Adv
① No RAM

* Disadv
① Convergence will be very slow
② Time complexity High

(3) Mini Batch Stochastic Gradient Descent (SGD)

- Set a batch size eg. batch size = 1000

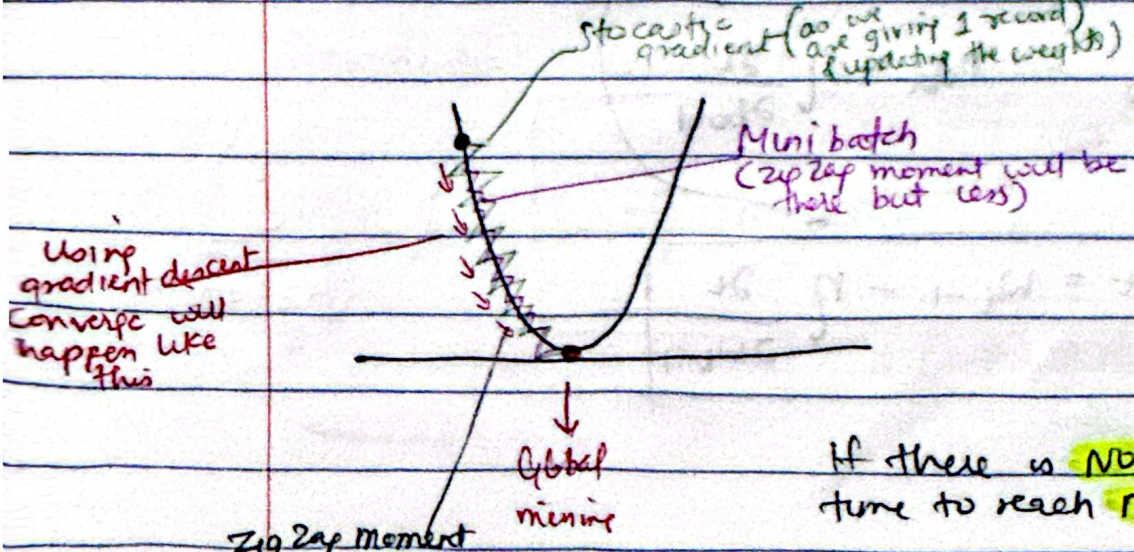
Epoch 1 ⟶ ⟵ } Iteration 1 (1000 records / 1000)

1000 records ⟶ ⟵ } Iteration 2

⋮

⟶ ⟵ Iteration 1000

- Not Resource Intensive
- Convergence will be better
- Time Complexity will improve.



If there is NOISE it will take time to reach minima

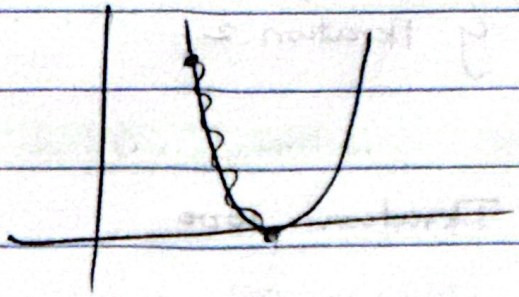
* How do we remove the noise?

We use concept as **momentum**

④ Mini Batch

SGD with momentum

It will smotten the journey while coming to minim



- Exponential Weighted Avg (moving Average)

Weight updat^n

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w_{old}}$$

$$bias_{new} = b_{old} - \eta \frac{\partial L}{\partial b_{old}}$$

t = new
t-1 = old

$$\boxed{w_t = w_{t-1} - \eta \frac{\partial L}{\partial w_{t-1}}}$$

t = time

t_1 t_2 t_3 t_4 ... t_n

a_1 a_2 a_3 a_4 ... a_n

$\beta \rightarrow$ Hyperparameter

$\beta = 0$ to 1

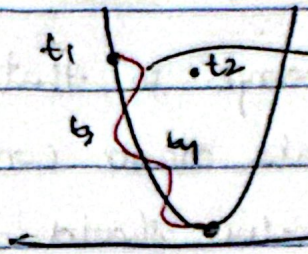
0.95

$V_{t1} = a_1$

$V_{t2} = \beta * V_{t1} + (1-\beta) * a_2$

$= (0.95) * V_{t1} + (0.05) * a_2$

weight values



Adv

Removing Noise & Smoothing curve

Quicker Convergence Mini Batch

$V_{t3} = \beta * V_{t2} + (1-\beta) * a_3$

→ Were do we apply this?

We apply it in $\frac{\partial L}{\partial w_{t-1}}$

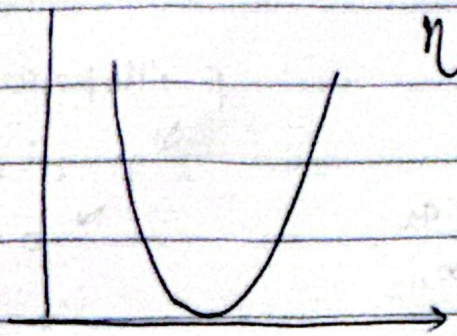
Exponential weighted Avg

~~$V_{dw} = \beta \times V_{dw_{\beta-1}}$~~

$W_t = W_{t-1} - \eta (V_{dw})$ →

$$V_{dw_t} = \beta * V_{dw_{\beta-1}} + (1-\beta) * \frac{dL}{\partial w_{t-1}}$$

(5) Adagrad → Adaptive Gradient Desent



η = learning Rate

↳ helps to maintain speed of convergence.

→ Can we somehow change so that if the learning rate is high once its starts coming close to global minima its value should decrease?

How can we do that

$$w_t = w_{t-1} - \eta \frac{dL}{\partial w_{t-1}}$$

Here we replace η with something else as we want to make it adaptive

$$w_t = w_{t-1} - \eta \frac{\partial L}{\partial w_{t-1}}$$

(1) → eta

$$\eta' = \frac{\eta}{\sqrt{\alpha_t + \epsilon}}$$

small no to avoid dw by 0

current timestamp

$$\alpha_t = \sum_{i=1}^{t} \left(\frac{\partial L}{\partial w_t}\right)^2$$

This value will increase so we have Σ (sumation)

& this α_t is in denominator for $\eta' = \frac{\eta}{\sqrt{\alpha_t + \epsilon}}$

so the value of η' eqn will decrease

$\Longrightarrow$ By this we are bringing Adaptiveness in the learning rate. So by this the learning rate wont be fixed it will be decreasing as we move towards global minima.

Disadv

if the value of ~~η' more~~ α_t increases to much there will be negligible change of η' so for that we have next optimizer.

We have to make sure α_t doesn't reach huge ~~value~~ valve.

(6) Adadelta & RMS Prop.

$$\eta' = \frac{\eta}{\sqrt{sdw + \epsilon}}$$

we apply exponential weighted avg.

$$sdw_t = \beta\, sdw_{t-1} + (1-\beta)\left(\frac{\partial L}{\partial w_{t-1}}\right)^2$$

if $\beta = 0.95$

$$sdw_t = (0.95)\, sdw_{t-1} + (0.05)\left(\frac{\partial L}{\partial w_{t-1}}\right)^2$$

this will control & not make equation bump up or value to go up.

as a result

$$\eta' = \frac{\eta}{\sqrt{sdw + \epsilon}}$$

there will be a close decrease in the value.

Smoothing, learning Rate becomes Adaptive

Smoothing eqn will have $\boxed{Vdw}$ as in SGD with momentum but here in Adadelta Smoothing wont happen //

there is no Vdw here.

Now thus Vdw & Sdw should be combined together.

V. V. Imp (7)

Adam optimizers (Best optimizer)

(momentum + RMS Prop (Adaptive learning Rate)

Vdw | Vdb (bias) Sdw | Sdb (bias)

Now my weight updatn eqn will be,

$$w_t = w_{t-1} - \eta' Vdw$$

for Bias updatn

$$b_t = w_{b_{t-1}} - \eta' Vdb.$$

$$\eta' = \frac{\eta}{\sqrt{Sdw + \epsilon}}$$

$$Vdb_t = \beta * Vd_{bt-1} + (1-\beta)\frac{\partial L}{\partial b_{t-1}}$$

$$Vdw_t = \beta * Vdw_{t-1} + (1-\beta) * \frac{dL}{\partial w_{t-1}}$$