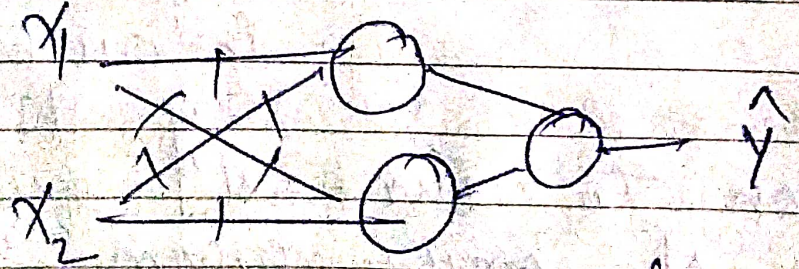
Optimizers in Neural Network

While performing any neural network model training we have always concern of the following things

1) Performance of NN
2) How to speed up the training?

3 techniques to speed up:
1) weight initialization
2) Batch Normalization
3) In some cases Activation fn.
4) optimizers.

What actually optimizers do?



4 2 2 1 (bias)

weights ↓ ↓ 9 params.

Bias weights

We need to find such a value from the 9 parameters that whatever the o/p model gives should be close to

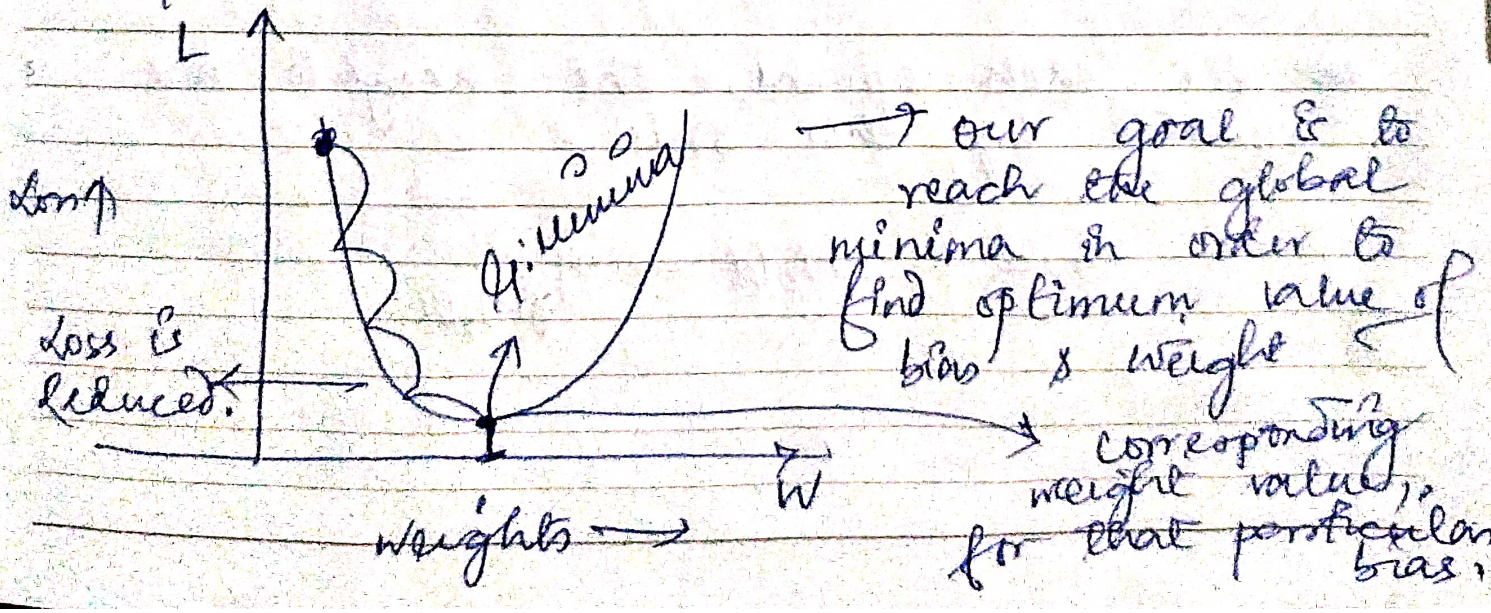
the original value.

→ Thus, we can conclude that the neural network is actually facing the optimization problem.
wherein,

1) we want to reduce the loss.
2) And to achieve this, the set values of weights & biases will be the optimum parameter of for our NN.

for this, we assign random value to weights & biases, & slowly & gradually while training, the optimum value of the weights & biases are reached.

If we draw the graph in 2-D, as for 1 parameter, → 1D graph is plotted.



→ our goal is to reach the global minima in order to find optimum value of bias & weight (corresponding weight value, for that particular bias).

→ So, to perform this task, we use Gradient Descent.

→ In DL, we use Gradient Descent.

Gradient Descent,

$$w_1 = w_0 - \eta \frac{\partial L}{\partial w}$$

w_1 → new weight, w_0 → old weight, η → learning rate

→ Type of Gradient Descent :-

(I) Batch GD
(II) Stochastic GD.
(III) Mini - Batch GD.

For each epochs, the weights are updated by this rule.

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

→ The main difference b/w the mentioned 3 GD, is that,

✓ How many datapoints, you want to see, before updating the weight?.

differentiation factor.

(i) for Batch GD, the whole datapoints are traversed. & weight is updated according to the epoch. For each epoch, weight updation is done, (for 10 epochs, 10 times the weight will be added).

(ii) for stochastic GD; the entire datapoints are traversed ~~but~~ & but difference lies, for each datapoints, there will be weight updation.

for eg, If there are 500 datapoints, then 500 × 10 (epochs)
≈ 5000 times weight updation takes place.

(iii) In mini-batch, we need to decide a batch for the available dataset & divide into batch for model training.

For example, for 500 datapoints, a batch of 100 datapoints may be used for training the model batchwise.

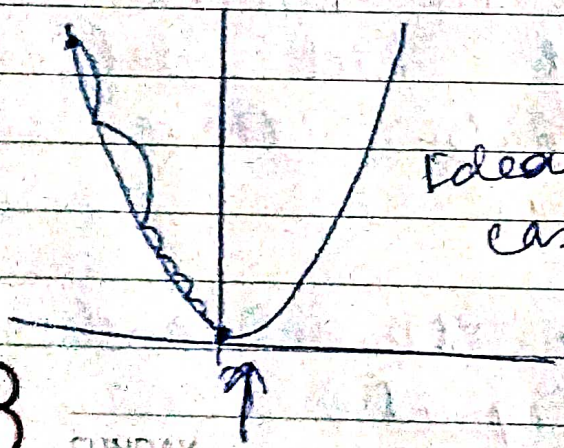
5 batch = 10×5 times

↑ epochs

↳ Challenges with previous optimizers:

1) Decide learning rate

$$w_n = w_0 - \eta \frac{\partial L}{\partial w_0}$$

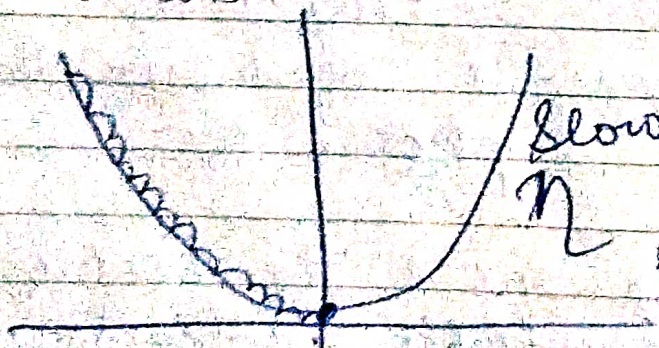


Ideal case

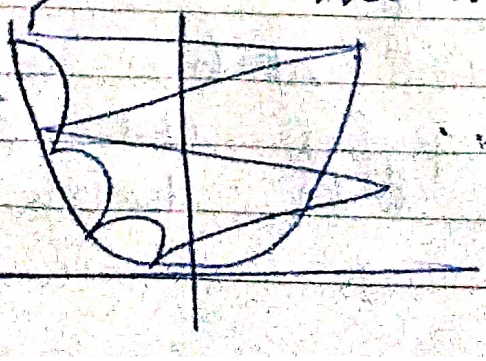
→ If we take η very less, convergence will be very slow.

→ Also, if we take η a bigger value, convergence will be very fast, may shoot the global minima.

08 SUNDAY



Slow η



bigger η

Soln: 2) Learning Rate scheduler:

→ pre-defined training model
problem of this, when to increase or
(after how decrease the learning as it is
many epochs) defined before the model training.
&

→ It means it doesn't depend on
any dataset. As if it is dependent
on dataset, to bring out proper
result would be challenging.

3) We need to minimize multiple
weights.
Ex: If there are two weights
w_1 & w_2, we cannot assign
separate learning rate for both the
& params, Gradient descent doesn't
provide that flexibility.

4) Local minima:

→ The GD graph is bit complex,
our target is to reach the global minima,
but we fall into the problem of
local minima, & get stuck & cannot
get out of that.
SGD has that ability to get out

as its movement is bit zig-zag, but for others it is quite natural possible, & work with sub-optimum minima. soln.

5) Saddle point:-

A point where slope of one (say) dimension moves in one direction & some other slope moves in other directions, (say down) this creates a place/region where, ~~the point has~~ the slope doesn't change, i.e, the slope remains constant: (diff=0), so, while updating, the Wold = Wnew, so, we get stuck.

For this reason, we need to use some different optimizers:-

1) Momentum
2) Adagrad
3) NAG
4) RMS Prop
5) Adam

→ Exponentially Weighted Moving Average.