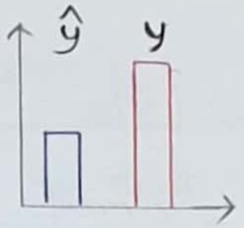
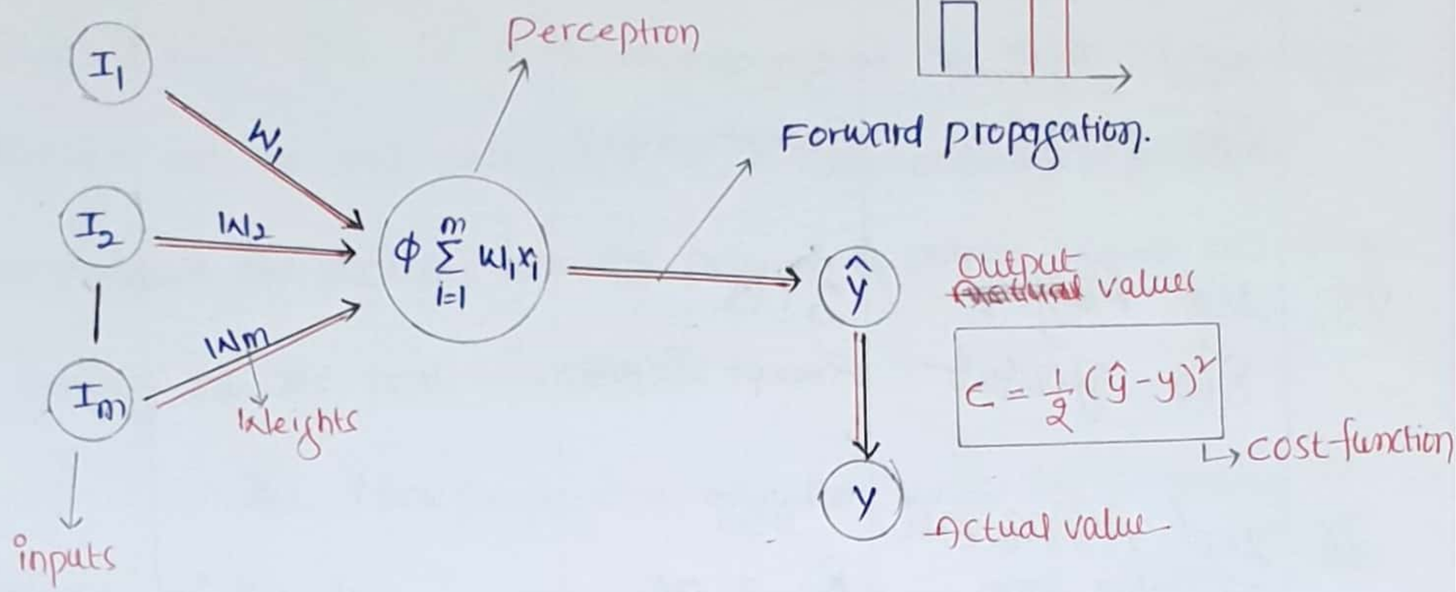
Forward propagation :-





→ Feed features to the Input layers

→ Caliculate the weights of an each neuron

→ predict the output

→ finally Compare the actual and predicted output.

Back propagation :- procedure —

i) caliculate the cost-function for predicted output in the forward propagation.

ii) back propagate (Gradient descent) and adjust the weights of the layers so that Cost-function is minimized.

iii) predict the output for all Observations. & caliculate the 'c'

iv) Adjust the weights so that Cost-function is minimized.

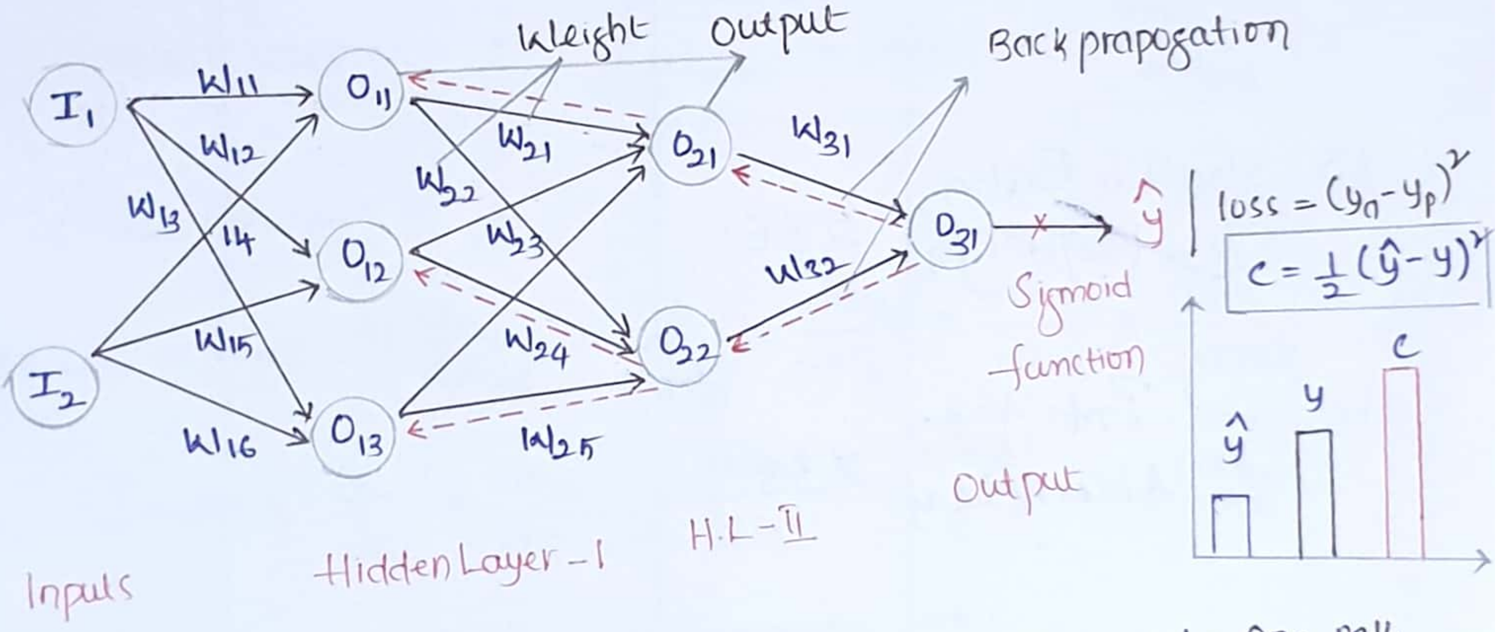
v) Again predict the output with new adjusted weights and repeate the procedure untill the Cost-function (c) is minimized.

Scanned with CamScanner

Back propagation :- [Backword prapogation]

It is a technique Used to train a certain classes of neural networks, it is essentially a principle that allows the machine learning program to adjust it self according to looking at its past function.

The back prapogation algorithm works by Computing the gradient of the loss function with respect to each weight by the chain rule, computing the gradient One layer at a time, iterating backword from the last layer to avoid redundant caliculations of Intermediate terms in the chain rule.



Note :- Back prapogation at a time Considering only One path if we observe from output we have two paths initially it takes one path ($O_{31} \rightarrow O_{21} \rightarrow O_{11}$) then other ($O_{31} \rightarrow O_{22} \rightarrow O_{12}$) like that.

Scanned with CamScanner

Activation functions & i/c forms :-

1) Sigmoid → $y = \frac{1}{1+e^{-x}}$ 2) Tanh → $y = \tanh(x)$

3) Step function → $y = \begin{cases} 0, & x < n \\ 1, & x \geq n \end{cases}$

4) Soft plus → $y = \log(1+e^{x})$

5) ReLU → $y = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$

6) Soft Sign → $y = \frac{x}{(1+|x|)}$

7) ELU → $y = \begin{cases} \alpha(e^{x}-1), & x < 0 \\ x, & x \geq 0 \end{cases}$

8) log of Sigmoid → $y = \log\left(\frac{1}{1+e^{-x}}\right)$

9) Swish → $y = \frac{x}{1+e^{-x}}$

10) Sinc → $y = \frac{\sin(x)}{x}$

11) Leaky Relu → $y = \max(0.1x, x)$

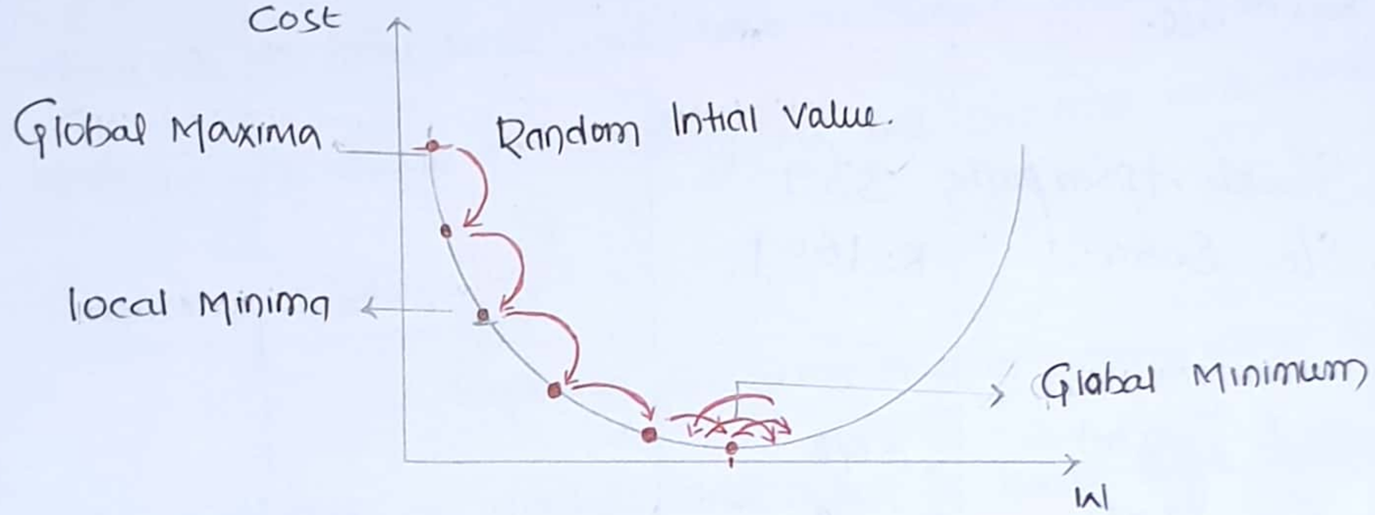
12) Mish → $y = x(\tanh(\text{soft plus}(x)))$

Scanned with CamScanner

Gradient descent :-

GD is an optimization algorithm for finding a local minimum of a differentiable function, it is used in machine learning to find a values of a functions parameters (Coefficents) that minimize a cost function as far as possible.

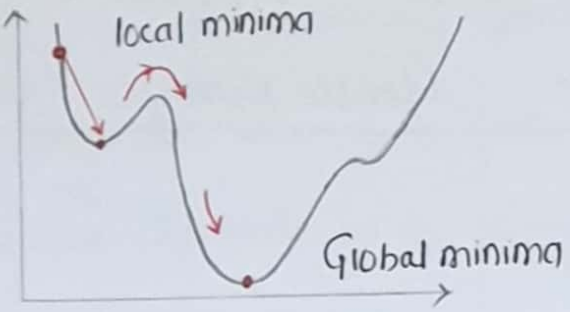
as well this algorithm works in neural networks, Training data helps these models learn over time and the cost function with in gradient descent specially acts as a barometer, guiding its accuracy with each itaration of parameter updates.



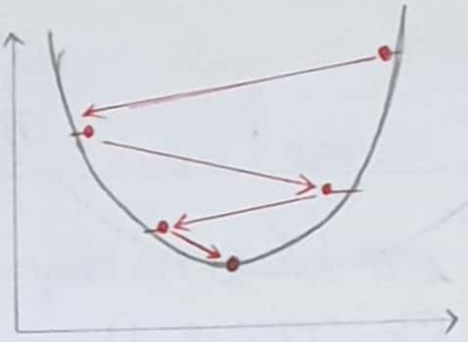
This jumps may happen if the learning rate is very high.

Vanishing Gradient descent :- it is deffect (cause) in Gradient descent, it refers to the fact that in a feed forward network (FFN). the back propagated error Signal typically decreases or increases exponantially. as a function of the distance from the final layer.

Scanned with CamScanner



[Vanishing G.D.]



[Exploding Gradient]

In G.D. vanishing G.D. is a deffect we have to overcome that cause of local value, when we iterate the value, then generated new value. reaches the minimum point then after no improvement in output, the point thinks to be the Global value reaches but that is a local value, we have to reach global value. this deffect is called as vanishing G.D., but our task is to overcome it.

Exploding Gradient :-

Exploding Gradients are a problem where large error gradients accumulate, and result in heavy large updates to neural network model weights during training. this has the effect of our model being. unstable and unable to learn from training data.

By Normalization we explod the Gradients (stop).

A common Solution to exploding Gradients is to change the error derivative before propagating it backward through the network and using it to update the weights.

Scanned with CamScanner

Note :- In Vanishing Gradient, the weights change are negligible and hence the learning is poor.

We can identify by the change in loss for every iteration.

The Exploding Gradient makes an exponential increment in Weights / higher jump in weight values, which Impacts the learning and loss are infinet or losses are very very high.

Exploding Gradient can be identified as nan in loss value.

Important Notes on GD :-

→ In Optimization, the main aim is to find weights that reduce loss.

→ Gradient is caliculated by optimizing function.

→ Gradient is the change is loss with change in weights.

→ The weights are modified according to the caliculated gradients.

→ Same process is repeated untill minima is reached.

Learning Rate and Momentum :-

Learning Rate is a hyper parameter to what extant newly aquired weights overrides the existing weights, lies between 0 and 1.

Momentum is used to decide the weights on nodes from previous iterations, it helps in improving training speed and also avoiding local minimas.

Scanned with CamScanner

Types of Gradient Descent :-

They are mainly classified as 3 categories. are -

1. Batch Gradient Descent
2. Stochastic Gradient Descent
3. Mini - batch Gradient Descent.

Batch Gradient Descent :-

In this we will caliculate the cost-function for all observations and update the weights to minimize the cost-function this is called Batch Gradient Descent.

Stochastic Gradient Descent :-

Here we caliculate the cost-function for each observation and updates the weights to minimize the cost-function this is called as Stochastic Gradient descent.

it takes the dataset in random and negative with FWP and BP for each epochs (iterations), the main disadvantage is it make the result more bias.

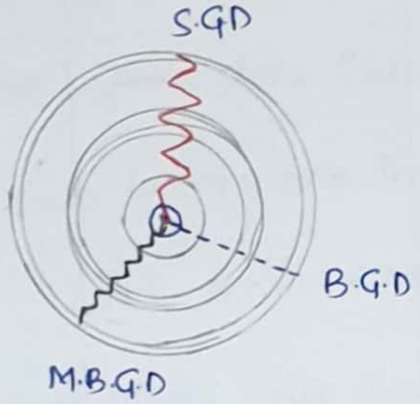
Mini - Batch Gradient Descent :-

In this we will first divide the data in to mini batches (5 or 8 observations for batch) then caliculate the cost-function for each mini observation and update the weights to minimize the cost-function.

Scanned with CamScanner

This phenominun is called as "Mini-batch G.D."

It navigates based upon steps per epochs | batch size of the G.D.



⇒

Momentum → Improving the noise from S.G.D, Mini batch SGD | from G.D.

$$\boxed{W_{new} = W_{old} - \eta \frac{dL}{dW_{t-1}}}$$

Exponential Weighted Average

$t_1, t_2, t_3 \ldots \quad W_{new} = W_{old} - \eta \frac{dL}{dw_{t-1}}$

$a_1 \; a_2 \; a_3$

$$= (1-\beta) * a_2$$

$$= \beta * a_1 + (1-\beta) * a_2$$

$$[\beta = 0.1] = 0.1 * a_1 + (1-0.1) * a_2$$

$$= 0.1 * a_1 + 0.9 * a_2$$

Major differences in G.D :-

Stochastic G.D.	Batch G.D.
1) Local minima problem is canbe resolved	1) may endup local minima.
2) fast	2) SLOW
3) Less Deterministic	3) More Deterministic

Scanned with CamScanner

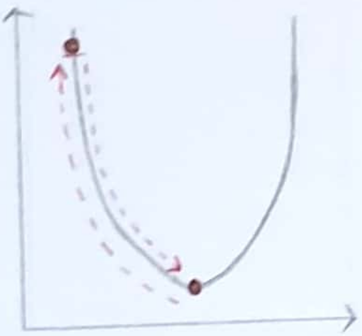
Adaptive Gradient Descent (Adagrad) :-

It is an algorithm for gradient based optimization it performs smaller updates as result, it is well suited when dealing with spares data (NLP or Image recognization) each parameter has its own learning rate that improves performance of problems.

It is an extention of Gradient decent optimization algorithm that allows step size in each dimention used by the optimization algorithm to be automatically adapted based on the gradients seen for the variables (partial derivatives).

In adagrade, the learning rate is not fixed and it is subjected to change based upon the loss.

$$\boxed{W_{new} = W_{old} - \eta' \frac{dL}{dW_{t-1}}}$$



$$\eta' = \frac{\eta \text{ (intial learning rate)}}{(\alpha_t + e)}$$

↳ constant

$$\alpha_t = \sum_{i=1}^{t} \left(\frac{dL}{dW_t}\right)^2$$

↳ $t = W_{11}$, W_{12}, W_{13}

$$\boxed{W_{new} \simeq W_{old}}$$

Scanned with CamScanner

RMSprop Optimization :-

It is an gradient based Optimization technique Used in training neural networks. this normalization balances the Step size (Momentum) decreasing the Step for large gradients to avoid exploding and increasing the Step for Small Gradients to avoid vanishing.

RMSprop Stands for root mean Square prop., which can also accelarate gradient descent, it uses the Same Concept of the exponentially weighted average of gradient descent with momentum but the differences is parameter update.

$$\boxed{W_{new} = W_{old} - \eta'' \frac{dL}{dw_{old}}}$$

$$\eta'' = \frac{\eta}{\sqrt{Sd_{wt} + \varepsilon}}$$

$$Sd_{wt} = \beta \cdot Sd_{wt-1} + (1-\beta)\left(\frac{dL}{dw_t}\right)^2$$

↳ previous

Adam optimization :- Adam is a replacement optimization algo for Stochastic GD. for training deep learning models. Adam Combines the best properties of the Adagrad and RMSprop algorithms to provide an optimization algo that can handle spares gradients.

Scanned with CamScanner

Adam is best among the adaptive optimizers in most of the cases., Good with Spare data.

Adam optimizer well Suited for large data sets and its Computationally efficient, there are few distadvantages with it as the adam optimizer tends to coverage faster, but Other algorithms like the Stochastic GD. focus on the data points and generalize in a better manner.

$$\boxed{W_{new} = W_{old} - \eta''' \frac{dL}{dW_{old}}}$$

where η''' — momentum, RMSprop.

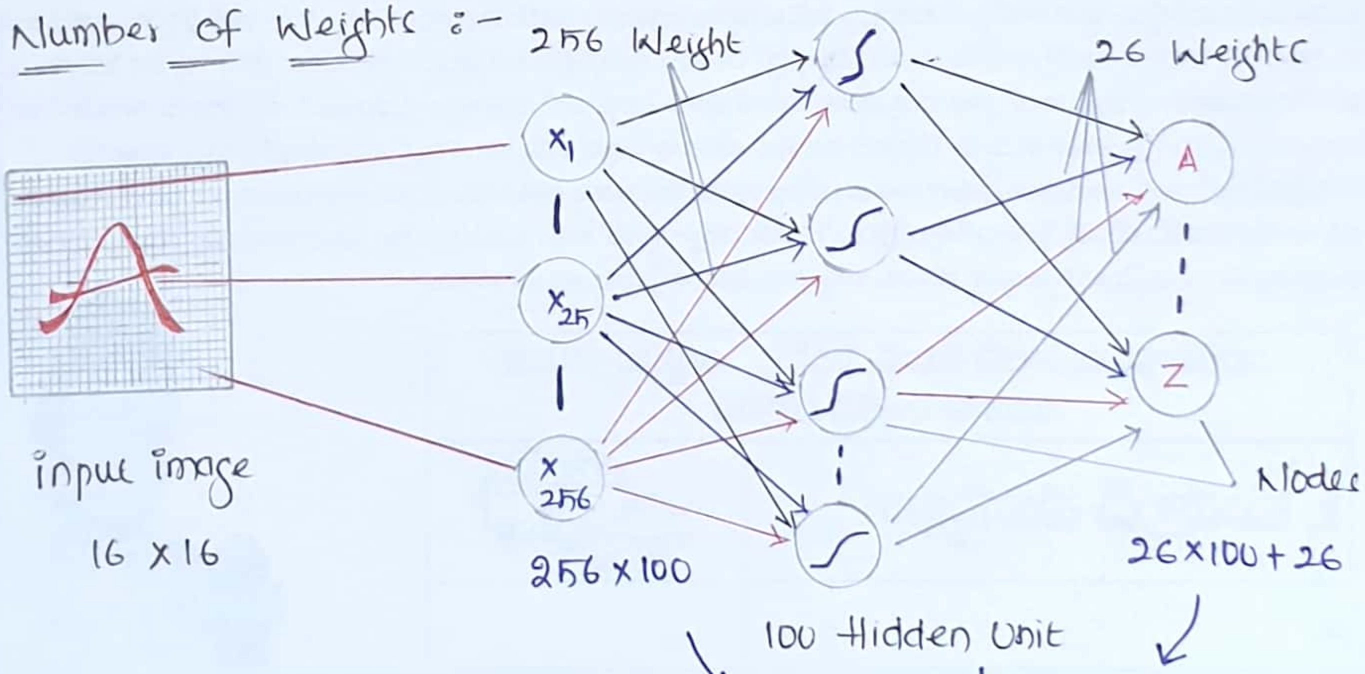
$$W_{new} = W_{old\,t-1} - \frac{\eta * V_{dw}}{\sqrt{S_{dw} + \varepsilon}}$$

V_{dw} → momentum

$\sqrt{S_{dw} + \varepsilon}$ → RMSprop.

Momentum :- $\boxed{V_{dw} = \beta V_{dw} + (1-\beta)\frac{dL}{dw}}$

Number of Weights :-



$= 25600 + 100 + 2600 + 26$

$= 25326$