Date :- 01st February, 2023.

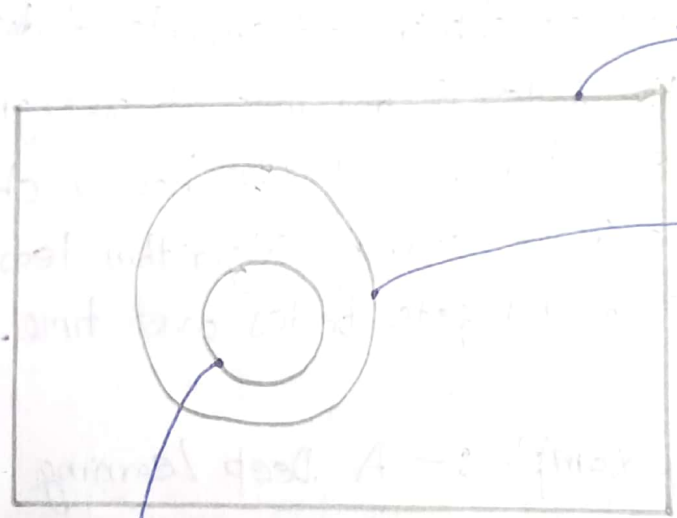
Deep Learning (Day-1)

Introduction to Deep Learning :-

Agenda :-

1) Deep Learning Introduction.
2) Type of Deep Learning.
3) Activation Function.

*) Graphical Representation of A.I vs ML vs DL.



Note :- Deep Learning is subset of ML.
Deep Learning is all about mimicing the Human Brain with the help of Multi-Layered Neural Network.

P.T.O

★) What is Deep Learning ?

⇒ Deep Learning is subset of ML.
Deep Learning is all about mimicing the Human Brain with the help of Multi-Layered Neural Network.

OR

Deep Learning is a type of Artificial intelligence (AI) that teaches Computers to Learn and make predictions based on data. It's like teaching a child to recognize different objects in the world, such as Dogs, Cats, Cars, etc.

Think of it as a digital brain that can identify Patterns and relationship in vast amounts of data. The more data it has, the better it becomes at recognizing patterns and making predictions. Just like how a child Learns from experience, a Deep Learning Algorithm learns from the data it is fed, and gets better over time.

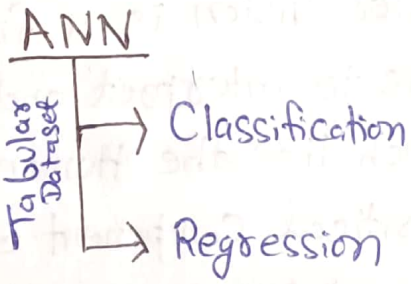
For example :- A Deep Learning Algorithm can be Used to identify objects in photos, translate Langvagues, or even drive a car without human intervention. The goal of Deep Learning is to help Computers make decisions and predictions that are as close to Human-Level intelligence as possible.

✱) In Deep Learning, Mainly there are Three Neural Network:-

1) Artificial Neural Network (ANN)
2) Convolutional Neural Network (CNN)
3) Recurrent Neural Network (RNN)

Now, in Details :-

1) Artificial Neural Network (ANN) :- AINN Stands for Artificial Neural Network, a type of Deep Learning Algorithm modelled after the Structure and Function of the human brain. It is used to Solve complex problems in areas such as image recognition, Speech-recognition, and natural Language processing by processing data, recognizing patterns, and Making Predictions.



2) Convolutional Neural Network (CNN) :- CNN Stands for Convolutional Neural Network, a type of Deep-Learning Algorithm used for Image and Video Recognition. It is designed to process data with a grid-like topology, such as an image, and is particularly useful for Image Classification tasks. CNNs use convolutional Layers to extract features from images and use pooling Layers to reduce the Spatial dimensions of the data. This allow the network to learn & recognize object, Shapes and textures in images.

Notes:— IN CNN, Input are :— Images and Video frames.

*) Some Example of CNN are :— RCNN, ,Maskid RCNN, Detectron, Yolo-V6, Yolo-V8, VGG-16, Resnet, etc.

Q) If CNN is also called Computer Vision ?

⇒ Yes, CNN (Convolutional Neural Network) is also Sometimes referred to as "Computer Vision". This is because it is a powerful tool for Computer Vision tasks such as Image Classification, Object-Detection and Segmentation.

Computer Vision is a field that aims to teach computers to interpret and understand visual information, much like the Human Visual System does. CNNs are a critical Component of Computer Vision, as they allow computers to recognize patterns and make predictions about the content of images and Videos.

In this sense, the term "Computer Vision" encompasses the entire field of using computers to process visual information, while "CNN" Specifically refers to one of the most powerful tools in that field.

3) Recurrent Neural Network (RNN) :- RNN Stands for Recurrent Neural Network, a type of deep Learning Algorithm designed for processing Sequential data, Such as Speech, text and time series data. RNNs have a "memory" component that allows them to retain information from previous steps in a Sequence, which enables them to make predictions based on context and capture patterns in sequential data. RNNs are commonly used in Natural Language Processing, Speech recognition and Time Series prediction.

Notes :- In RNN, Inputs data are :- Text Data and , Time-Series Data.

*) Some Example of RNN are :- LSTM RNN, RNN GRU, Bi-Directional LSTM RNN, Encoder-Decoder, Transformer, BERT, ,Attension Model.

*) Which Library is used for Deep Learning Model ?

⇒ TensorFlow and Pytorch are both open Source Software Library used for Deep Learning as well as Machine Learning.

*) TensorFlow was Developed by Google.

Notes :- Keras is a part of TensorFlow.

*) PyTorch was Developed by Facebook.

#) Why Deep Learning is Becoming Popular ?

#) 2005 → Facebook, orkut } ⟹ Social Media Platform

⇓

Data was getting Generated Exponentially.

Note :- (Data is generated in the forms of Images, Text, Document, Video, etc.)

⇓

Storing these Data Efficiently.

#) 2011 → Is a Craze of Big Data getting famous.

↳ Platform Like :-

↳ Hadoof

→ Structure Data
→ UnStructure Data.

⇓

#) 2011 - 2016 ⟹ Cloudera, Horton Work

(Eg :- AWS has its own Stored Frameworks like S3 Bucked.)

⇓

⇓

#) 2015 :—

Company thought, Can we use the Data to make the Product better, then

⇓

Then Data Science Concepts Comes as

Deep Learning.

Note :- (Geoffery Hinton is ⇒ Father of ANN)

*) Now Coming to Main Topic :— Why Deep-Learning is Becoming Popular ?

i) Hardware Requirements ⇒ GPU's ⇒ Nvidia GPU's

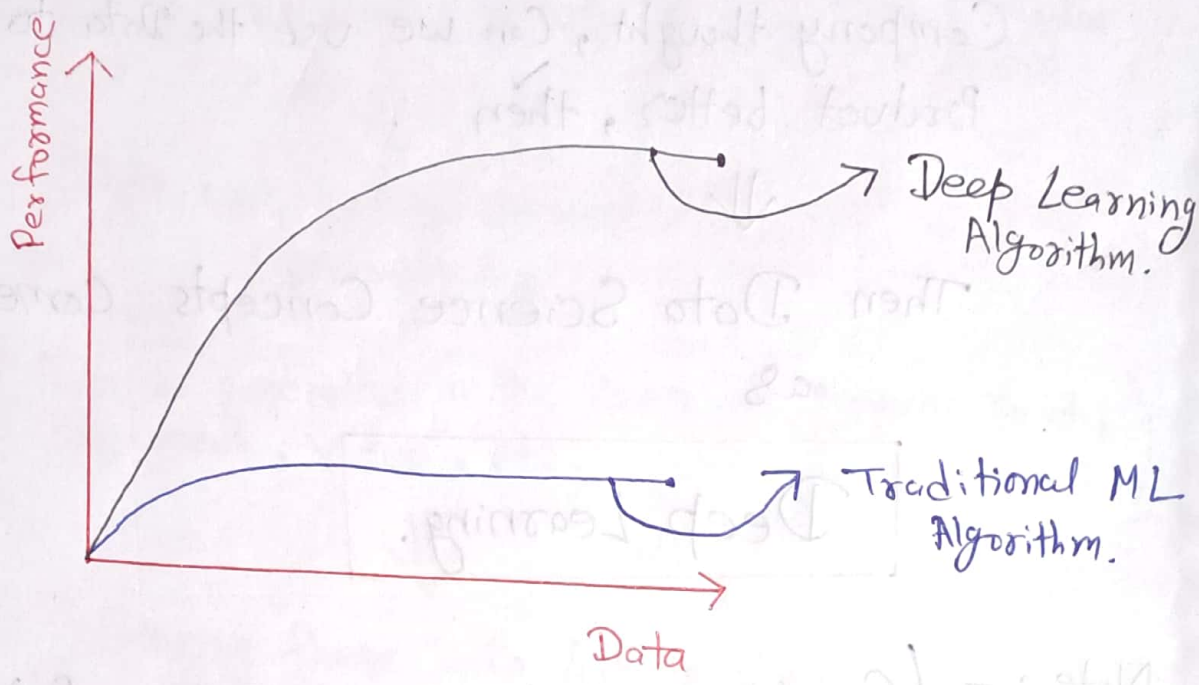
Notes:- (Due to Nvidia GPU's a price is Continously Decreasing)

for Example of GPU's are :—

ⓐ RTX 3080 GPU's
ⓑ Titan RTX GPU's.

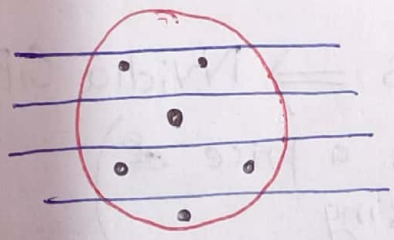
P.T.O

ii) Huge Amount of Data is Generated ⟹ Deep Learning Model Performs wells on Huge Amt. of Data.



iii) Deep Learning is been Used in Many Domains :—

a) Medical Domains ⟹ Prediction of Diseases, X-rays, Bone Crack, Lungs Disease.



Brain Scan ⟹ (Predict Tumors or Not)

↳ 3D Images

↳ Eg:- MRI Scan.

Main things we have Good Amts of Datas.

b) E-Commerce.

c) Retails

d) Logistics.

#) Now In Deep Learning :—

1) Perceptron → [Artificial Neural or Neural Network Unit]

⇓

[Perceptron is a Single Layered Neural Network]

OR

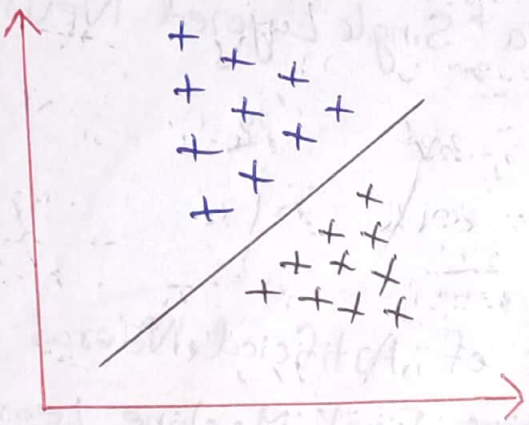
⇒ Perceptron is a type of Artificial Neural Network that was one of the ~~following~~ First Machine Learning Algorithm. It is a simple model that consists of a single layer of artificial neurons, each of which received input, processes it, and produces an output. The output of each neuron is binary (either 0 or 1) and represents the predicted class label. The model is trained by adjusting the weights of the inputs to each neuron until the output correctly classifies the input data.

Perceptron were developed in the 1950s and 1960s as a way to model how the human brain works. Although they are related simple and limited in their capabilities compared to modern deep learning models, they were a key step in the development of neural networks and Machine Learning.

P.T.O

Note :- (Perceptron is used to solve Binary Classification Problem.

Linear Separables.



a) I/P Layer
b) Hidden Layer
c) Weights
d) Activation function
e) Bias
f) O/P Layers.

Notes :-

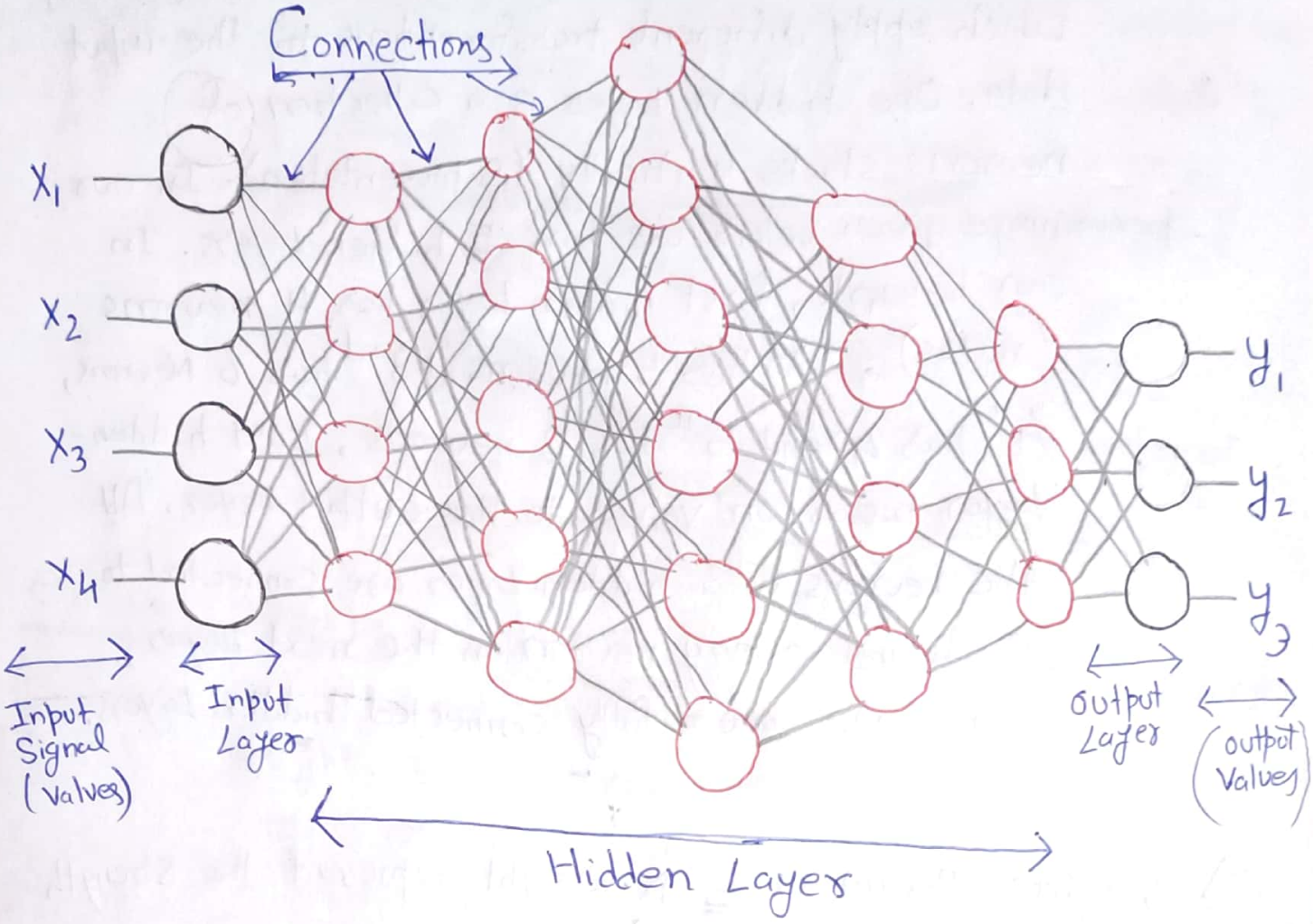
*) What is Neuron (Node) ?

⇒ Neuron (Node) is the basic unit of a neural Network. It gets certain number of inputs and a bias value. When a signal (value) arrives, it gets multiplied by a weight value. If a neuron has 4 input, it has 4 weight values which can be adjusted during training Time.

*) Now in Details :- (Please check Next Page Image for Details) :-

a) Input (I/P) Layer :- Input Layer is the first Layer in the neural Network. It takes input signals (value) and passes them to the next Layer. It doesn't apply any operations on the input signals (values) & has no weights & baises values associated. In our network we have 4 input signals x_1, x_2, x_3, x_4.

*) Basic Neural Network Layout :—



Notes :— What is connections ?

⇒ Connections connects one neuron in one layer to another neuron in other layer or the same layer. A connection always has a weight value associated with it. Goal of the training is to update this weight value to decrease the loss (error).

P.T.O

b) Hidden Layers :— Hidden Layers have neurons (nodes) which apply different transformations to the input data. One hidden Layer is a collection of neurons stacks vertically (Representation). In our previous image given below we have 5 hidden Layers. In our network, first hidden Layer has 4 neurons (nodes), 2nd has 5 neurons, 3rd has 6 Neurons, 4th has 4 and 5th has 3 neurons. Last hidden-Layer passes on values to the output Layer. All the neurons in a hidden Layer are connected to each and every neuron in the next Layer, hence we have a fully connected hidden Layers.

C) Weights (Parameters) :— A weight represent the Strength of the connection between units. If the weights from node 1 to node 2 has greater magnitude, it means that neuron 1 has greater influence over neuron 2. A weight brings down the important of the input value. Weights near Zero means Changing this input will not change the output. Negative weight mean increasing this input will decrease the output. A weight decides how much influence the input will have on the output.

P.T.O

d) Activation Function (Transfer Functions) :— Activation Functions are used to introduce non-linearity to neural Networks. It Squashes the values in a smaller range viz. a Sigmoid Activation function squashes values between a range 0 to 1. There are many Activation Functions used in Deep Learning industry and ReLU, SeLU and TanH are preferred over Sigmoid Activation functions.

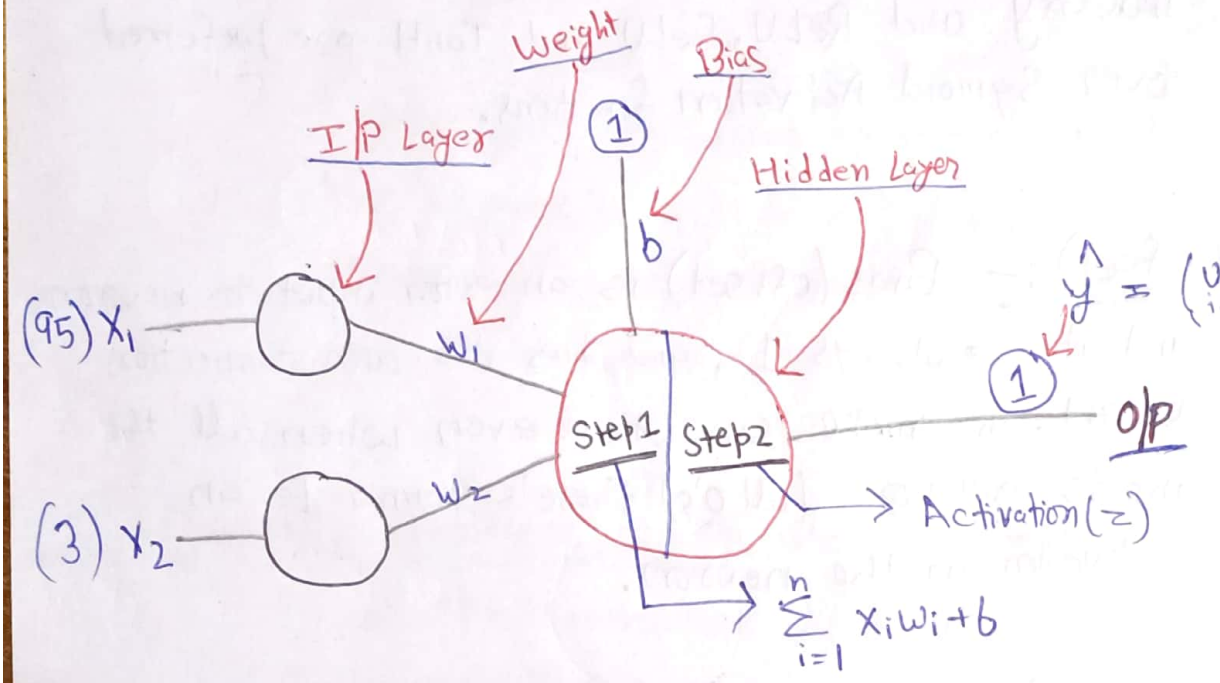
e) Bias (offset) :— Bias (offset) is an extra input to neurons and it is always 1, and has it's own connection weight. This makes sure that even when all the inputs are none (all 0's) there's gonna be an activation in the neuron.

f) Output Layer :— This Layer is the Last Layer in the network and receives input from the Last hidden Layer. With this Layer we can get desired Number of values and in a desired range. In this network we have 3 neurons in the output Layer and it outputs y_1, y_2, y_3.

Single Layered Neural Network.

Dataset

IQ	No. of Study hrs	O/P (Pass / Fails)
95	3	0
110	4	1
100	5	1



feed forward Neural Network (forward Propogation)

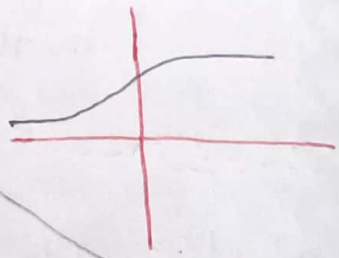
Step-1 :-

$Z = x_1 \cdot w_1 + x_2 \cdot w_2 + 1 \times b$ — Bias is as Intercept

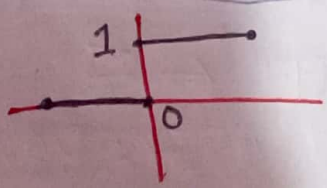
$z = \sum_{i=1}^{n} x_i w_i + b$ $(y = mx + c)$

$(y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_0)$

Sigmoid Functions



Step Functions :-



Here, Threshold = 0

Eg: $Z = 5 \Rightarrow 1$

$Z = -2 \Rightarrow 0$

0 or 1

#) Multi-Layered Perceptron Model (ANN)
(Artificial Neural Network)

⇒ A Multi-Layered Perceptron (MLP) Model is a feedforward Artificial neural Network (ANN) that generates a set of outputs from a set of inputs. An MLP is Characterized by several Layers of input nodes Connected as a directed graph between the input and output Layers. MLP uses back-propogation for training the network. MLP is a Deep Learning Method.

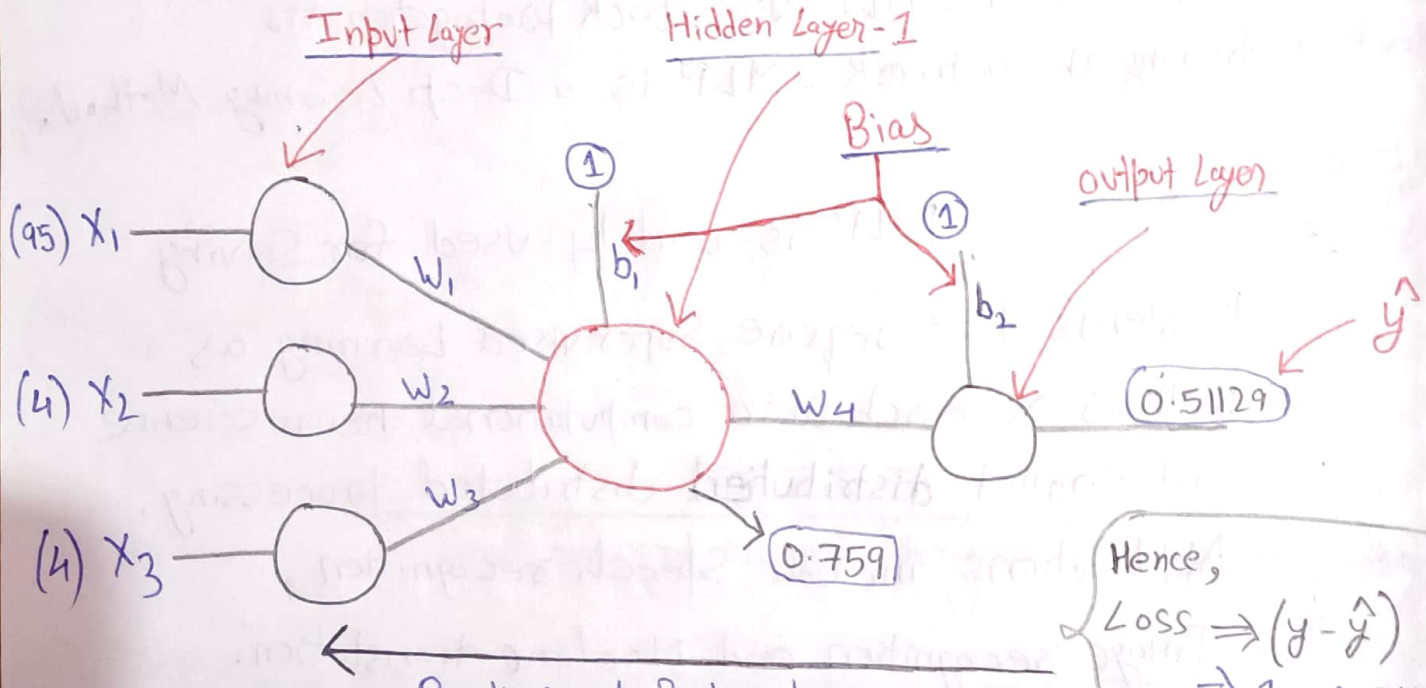
MLP is widely used for Solving Problems that require Supervised Learning as well as research into computational neuroscience and Parallel ~~distributed~~ distributed processing. Applications include Speech recognition, Image recognition and Machine translation.

a) Forward Propogation
b) Backward Propogation
c) Loss Function
d) Activation Function
e) Optimizers.

✱) Two - Layered Neural Network :-

Dataset (Binary Classification)

IQ	Study Hrs	Play hrs	O/P (Pass/Fail)
95	4	4	1
100	5	2	1
95	2	7	0



Hence,
Loss $\Rightarrow (y - \hat{y})$
$\Rightarrow 1 - 0.51129$
$\Rightarrow 0.49$ //

Notes:- Backward Propogation is used to update the weights, to minimized the Loss. //

✱) Hidden Layer - 1 :-

Note:-
Let Assumed (Initialized)
$\begin{bmatrix} W_1, W_2, W_3 \\ 0.01, 0.02, 0.03 \end{bmatrix}$ & Bias $[0.01]$

✱) Step-1 :- $95 * 0.01 + 4 * 0.02 + 4 * 0.03 + 0.01$

$z \Rightarrow 1.151$ //

✱) Step 2 :- Activation (z) $\Rightarrow$ O/P $\Rightarrow \frac{1}{1+e^{-(1.151)}} \Rightarrow 0.759$ //

Now, Let Assumed :-

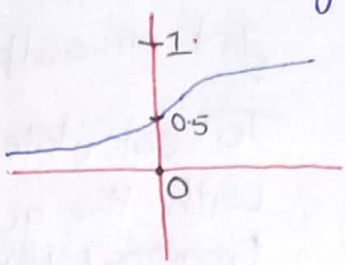
$W_4 = [0.02], 1$

&

$Bias_2 = 0.03$

Sigmoid

$y = \frac{1}{1+e^{-z}}$



✱) Hidden Layer-2 :-

* Step-1 :-

$Z \Rightarrow 0.759 * 0.02 + 1 \times 0.03$

$Z \Rightarrow 0.04518,$

Step-2 :-

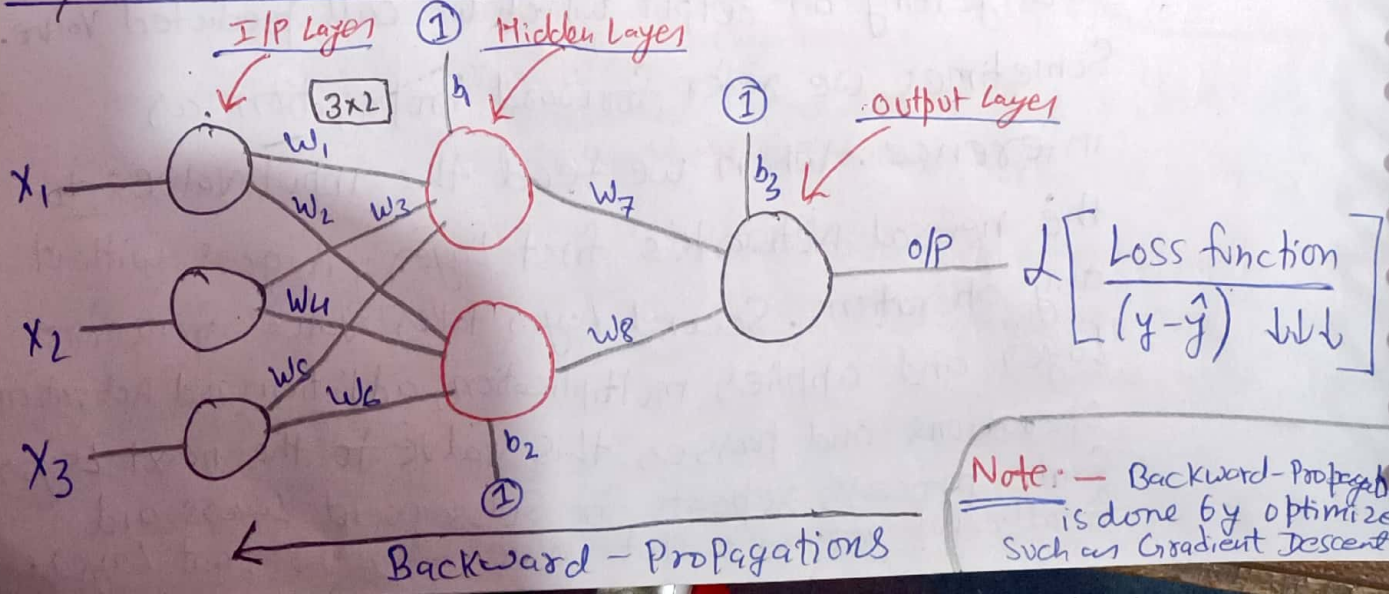
$O/P \Rightarrow \frac{1}{1+e^{-(0.04518)}} = 0.51129 \Longrightarrow$ Output Layers.

✱) Now, in Details :-

a) Forward Propagation :- Forward Propagation is a Process of feeding input values to the neural Network and getting an output which we call predicted Value. Sometimes we refer forward Propagation as inference. When we feed the input values to the neural Network's first Layer, it goes without any operations. Second Layer takes values from first Layer and applies multiplication, addition and Activation operations and passes this value to the next Layer. Same process repeats for Subsequent Layers and finally we get an output value from the Last Layer.

b) **Backward Propagation :-** After forward Propagation we get an output value which is the predicted value. To calculate error we compare the predicted value with the actual output value. We use a Loss function (mentioned below) to calculate the error value. Then we calculate the derivative of the error value with respect to each and every weight in the neural network. Back-Propagation uses chain Rule of Differential Calculus. In Chain rule first we calculate the derivatives of error value with respect to the weight values of the last layer. We call these derivatives, gradients and use these gradient values to calculate the gradient of the second last Layer. We repeat this process until we get gradients for each and every weight in our neural network. Then we substract this gradient value from the weight value to reduce the error value. In this way we move closer (descent) to the Local Minima (Means Minimum Loss).

*) **Backward Propagation And Weight Updation Formula :-**



Note :- Backward-Propagation is done by optimizers such as Gradient Descent

✱) Weight Updation Formula :-

$$\Rightarrow W_{7new} = W_{7old} - \eta \frac{\partial h}{\partial W_{7old}}$$

η → Learning Rate.

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

⇓

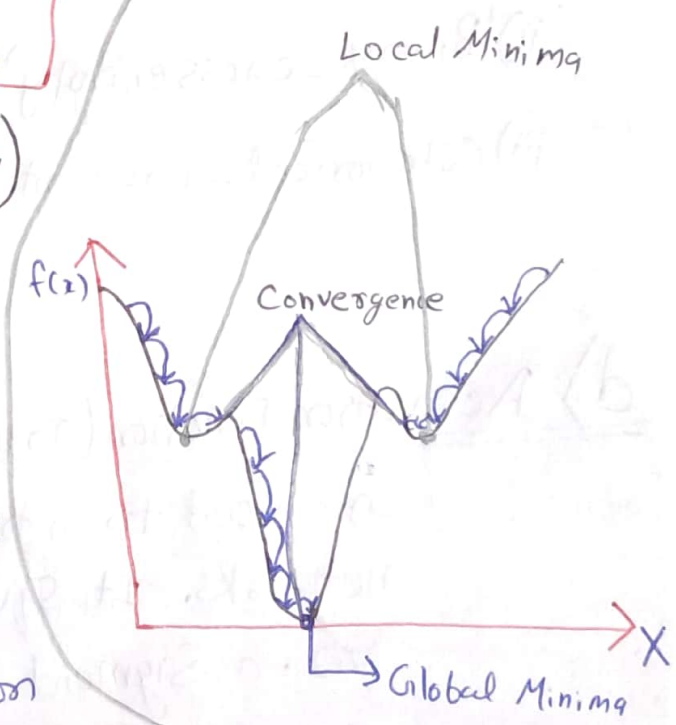
(Weight Updation Formula)

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

⇓

(Bias Updation Formula)

✱) Gradient Descent



Note :- What is Learning Rate :-

⇒ When we train neural network we usually use Gradient Descent to optimize the weights. At each iteration we use Back-Propagation to calculate the derivative of the Loss Function with respect to each weight and Substract it from that weight. Learning rate determined how quickly or how slowly you want to update your weight (parameters) value. Learning rate should be high enough So that it won't take ages to converge, and it should be low enough so that it finds the Local Minima.

P.T.O //

Note :- What is Convergence ?

⇒ Convergence is when as the iterations ~~proceed~~ proceed the output get closer and closer to a specific value.

c) Loss Functions (Cost Functions) :- The Loss Function Computes the error for a single training example. The Cost Function is the average of the Loss Functions of the entire training set.

i) 'mse' :- for Mean Squared Error.

ii) 'Binary_crossentropy' :- for binary logarithmic loss (logloss)

iii) 'Categorical_crossentropy' :- for Multi-Class Logarithmic Loss (LogLoss).

d) Activation Function (Transfer Function) :- Activation Functions are used to introduce non-linearity to neural-networks. It Squashes the values in a smaller range Viz. a Sigmoid activation function ~~Squashed~~ Squashes values between a range 0 to 1. There are many activation functions used in Deep Learning industry and ReLU, SeLU and TanH are preferred over Sigmoid activation function.

P.T.O

e) **Optimizers :—** The Optimizers is a Search technique, Which is used to update weights in the model.

i) SGD :— Stochastic Gradient Descent, with Support for Momentum.

ii) RMSprop :— Adaptive Learning rate Optimization method proposed by Geoff Hinton.

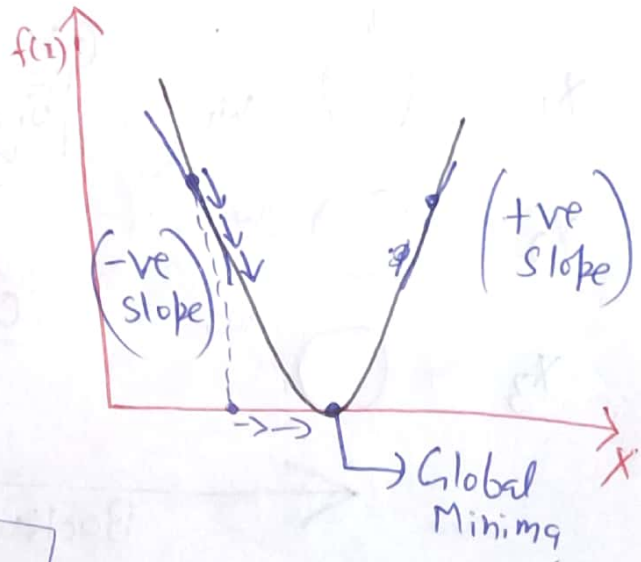
iii) Adam :— Adaptive Moment Estimation (Adam) that also used Adaptive Learning Rates.

Notes :— Optimizers is used to reduce the Loss Value.

$W_{new} = W_{old} - \eta\,(-ve)$

$W_{new} = W_{old} + (+ve)$

$W_{new} >> W_{old}$



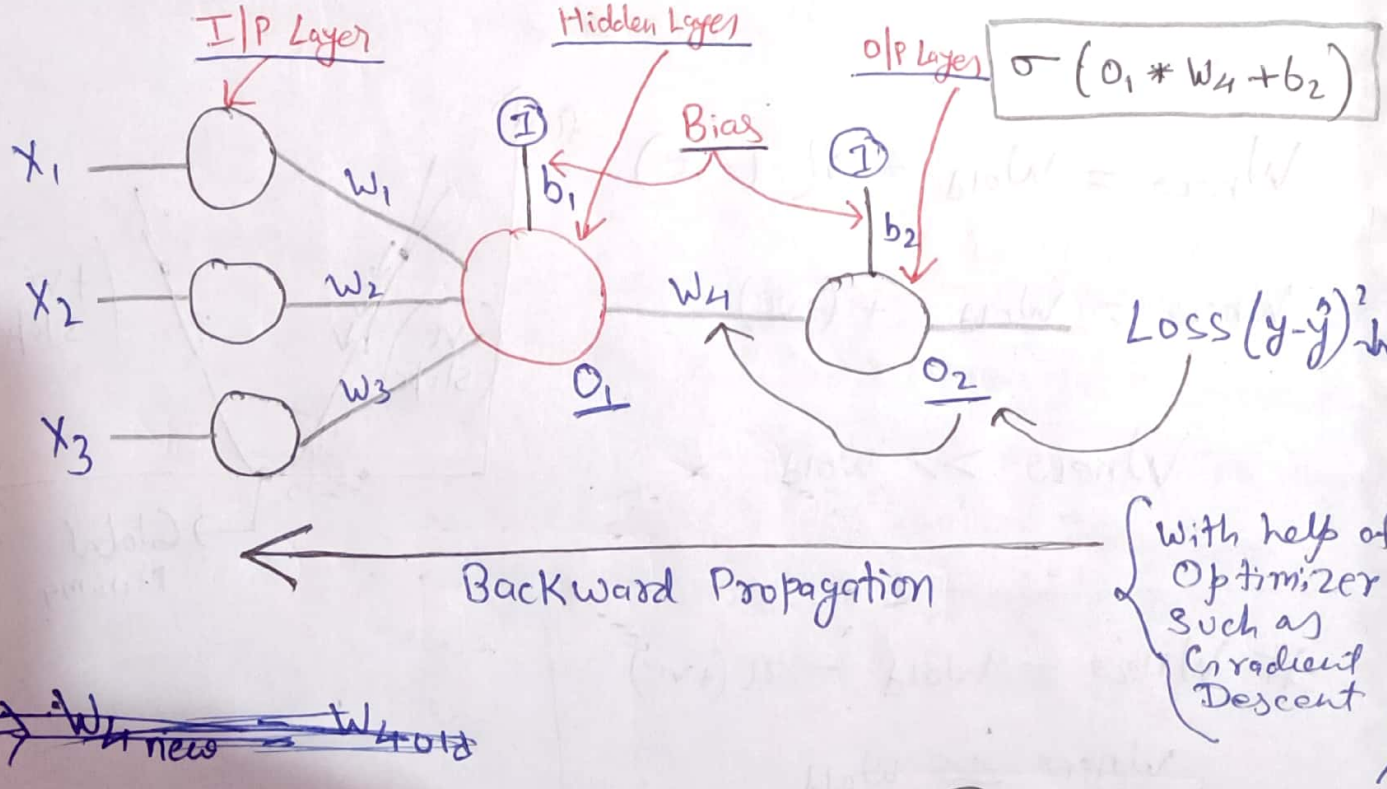
$W_{new} = W_{old} - \eta\,(+ve)$

$W_{new} << W_{old}$

Chain Rule of Derivative :—

⇒ The Chain Rule allows us to find the derivative of Composite Functions.

It is computed extensively by the back-propagation Algorithm, in order to train feedforward Neural Networks. By applying the Chain Rule in an efficient Manner while following a specific order of operations, the Back-Propagation Algorithm calculates the error gradient of the Loss-Function with respect to each weight of the Networks.



~~$W_{4\,new} = W_{4\,old}$~~

P.T.O

$$\Rightarrow W_{4\,new} = W_{4\,old} - \eta \boxed{\frac{\partial h}{\partial W_{old}}} \Rightarrow \underline{\text{Slope}}$$

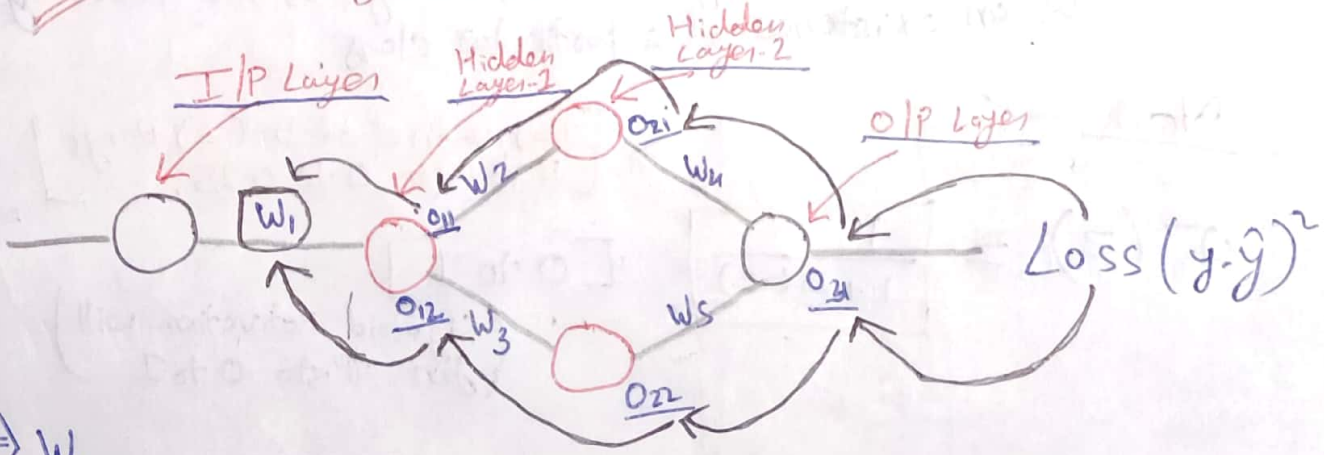
$$\boxed{\frac{\partial h}{\partial W_{4\,old}} = \frac{\partial h}{\partial O_2} * \frac{\partial O_2}{\partial W_{4\,old}}} \Rightarrow \text{Chain Rule of Derivative}$$

$$\Rightarrow W_{1\,new} = W_{1\,old} - \eta \frac{\partial h}{\partial W_{1,old}}.$$

$$\Rightarrow \frac{\partial h}{\partial W_{1,old}} = \frac{\partial h}{\partial O_2} * \frac{\partial O_2}{\partial O_1} * \frac{\partial O_1}{\partial W_{1,old}}$$

$$\Rightarrow \frac{\partial h}{\partial W_{1,old}} = \frac{\partial h}{\partial W_{1,old}}$$

☆ Assignment

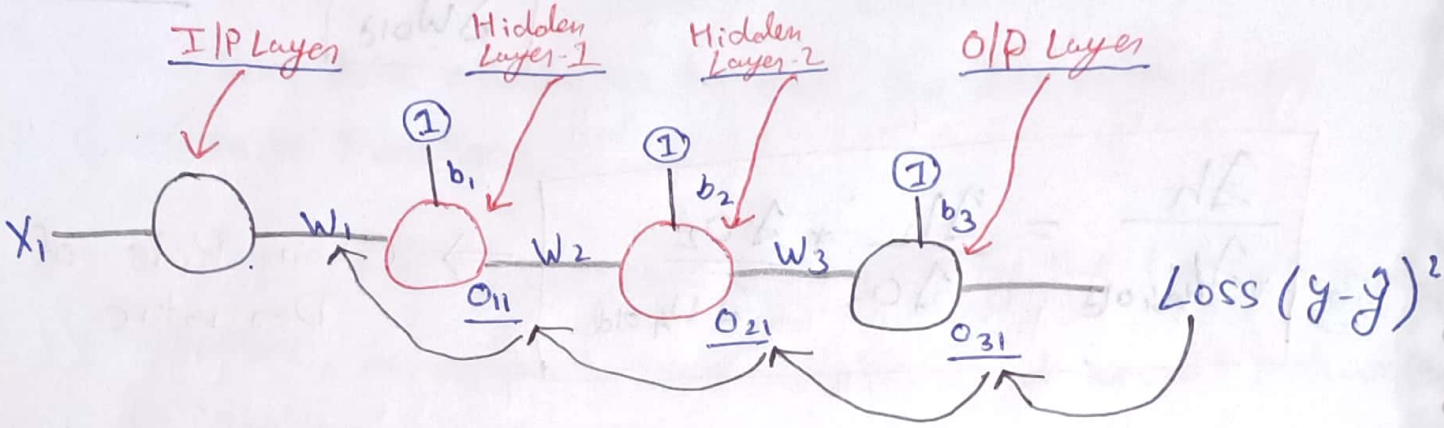


$$\Rightarrow W_{1\,new} = W_{1\,old} - \eta \boxed{\frac{\partial h}{\partial W_{1\,old}}}$$

$$\Rightarrow \frac{\partial h}{\partial W_{1old}} = \left[\frac{\partial h}{\partial O_{31}} * \frac{\partial O_{31}}{\partial O_{21}} * \frac{\partial O_{21}}{\partial O_{11}} * \frac{\partial O_{11}}{\partial O_{w,old}}\right] + \quad \leftarrow \text{Path-1}$$

$$\left[\frac{\partial h}{\partial O_{31}} * \frac{\partial O_{31}}{\partial O_{22}} * \frac{\partial O_{22}}{\partial O_{12}} * \frac{\partial O_{12}}{\partial O_{w,old}}\right] \leftarrow \text{Path-2}$$

✱) Vanishing Gradient Problem And Activation Functions :-



✱) Sigmoid Activation :- Sigmoid accepts a number as input and returns a number between 0 and 1. It's simple to use and has all the desirable ~~qvanh~~ qualities of Activation Functions :- Non-Linearity, Continous differentiation, monotonicity, and a set output range.

This is mainly used in binary Classification Problems. This Sigmoid function gives the Probality of an existence of a particular class.

Now

$$\sigma(z) = \boxed{\frac{1}{1+e^{(-z)}}} \quad [0 \text{ to } 1]$$

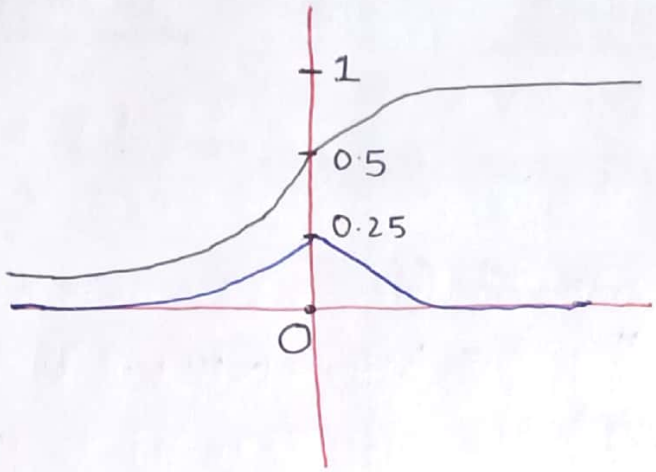
→ [Derivative of $\sigma(z)$ Range Between 0 to 0.25.]

(Sigmoid Activation will give o/p's to 0 to 1)

$$\boxed{0 \le \sigma(z) \le 1}$$

(Sigmoid Activation function)

P.T.O



★ $0 \le \sigma(z) \le 1$ ⇒ Sigmoid Activation Function

★ $0 \le \sigma(z) \le 0.25$ ⇒ Derivative of $\sigma(z)$

$$\Rightarrow w_{1\,new} = w_{1\,old} - \eta \frac{\partial h}{\partial w_{1\,old}}$$

$$\Rightarrow \frac{\partial h}{\partial w_{1\,old}} = \frac{\partial h}{\partial o_{31}} * \frac{\partial o_{31}}{\partial o_{21}} * \frac{\partial o_{21}}{\partial o_{11}} * \frac{\partial o_{11}}{\partial w_{1\,old}}$$

$$\Rightarrow o_{31} = \sigma(z)$$

$$\Rightarrow o_{31} = \sigma(w_3 * o_{21} + b_3)$$

$$\Rightarrow o_{31} = \sigma(z)$$

as, $z = w_3 * o_{21} + b_3$

$$\Rightarrow \frac{\partial o_{31}}{\partial o_{21}} = \frac{\partial \sigma(z)}{\partial o_{21}} = \frac{\partial \sigma(z)}{\partial z} * \frac{\partial z}{\partial o_{21}}$$ { Chain Rule }

$$\Downarrow$$

$$\frac{\partial (w_3 * o_{21} + b_3)}{\partial o_{21}} = w_3$$

$$\Rightarrow \frac{\partial o_{31}}{\partial o_{21}} = [0 - 0.25] * w_3 \Rightarrow$$ (we get small value)

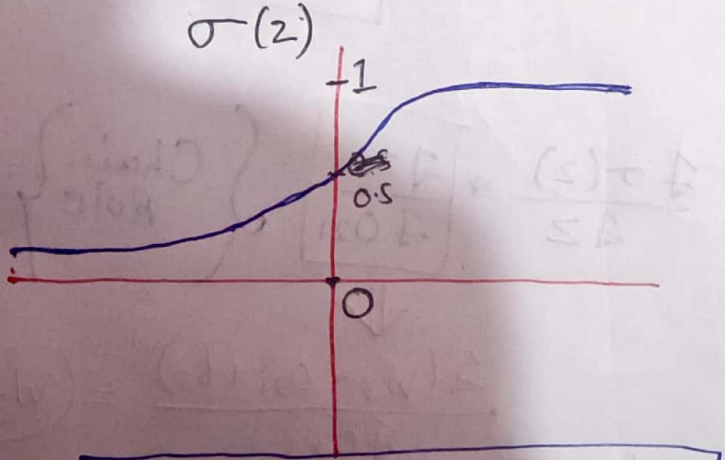
*) To Fix Previous Problem, we used other Activations functions :-

a) Tan-H
b) ReLU
c) Prelu
d) ELU
e) Softmax

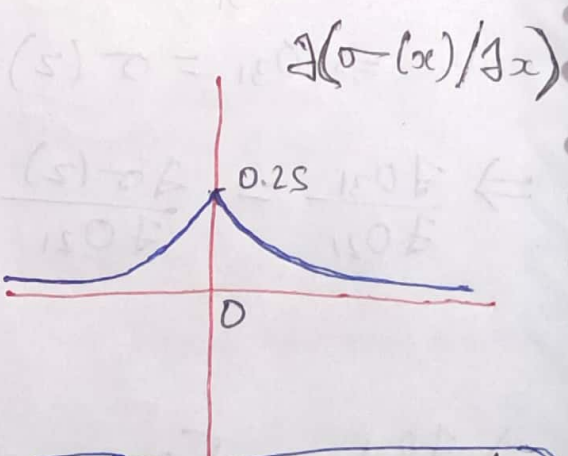
*) Sigmoid - Activation Function :-

⇒ Sigmoid accepts a number as input and returns a numbers between 0 and 1. It's simple to use and has all the desirable qualities of Activation Functions:- Non-Linearity, continuous.differentiation, monotonicity, and a set outcome range.

This is mainly used in binary Classification Problems. This Sigmoid Function gives the probability of an existence of a particolar class.



Sigmoid Activation function



Derivative Sigmoid Function

Advantage

a) Clear Prediction (0 to 1)

Disadvantage

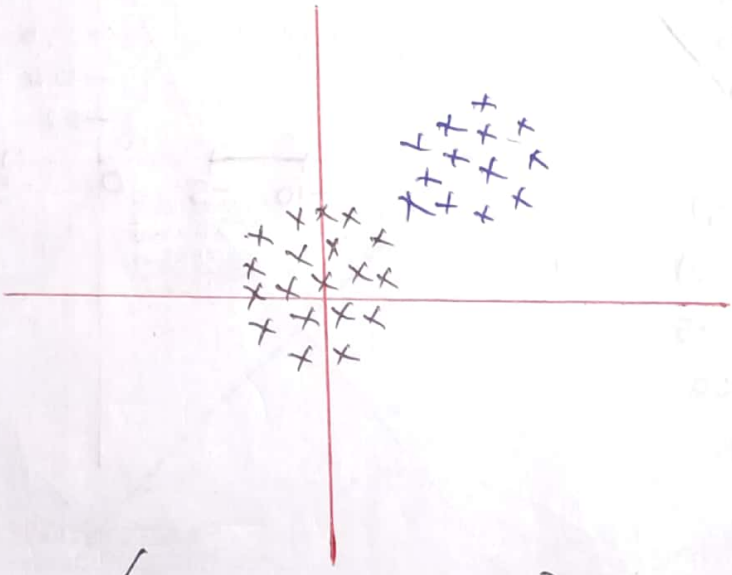
ⓐ Prone to Vanishing Gradient Problem.

ⓑ function output is not Zero Centered.

⇓

(Efficient Weight Updation)

A) Zero Centered

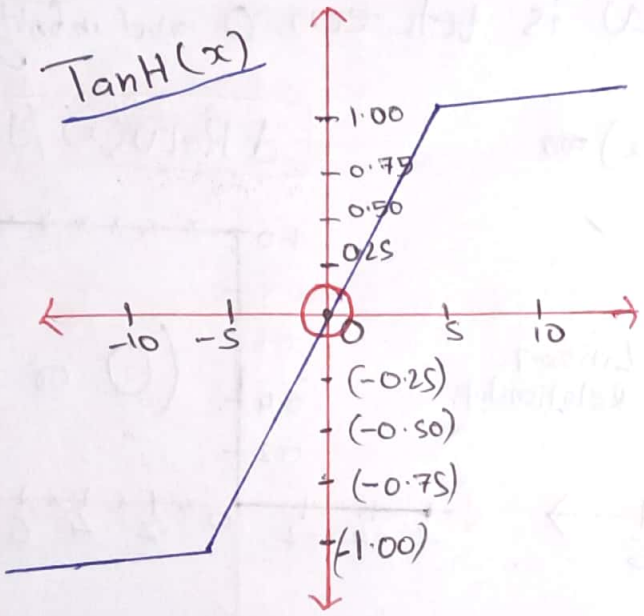


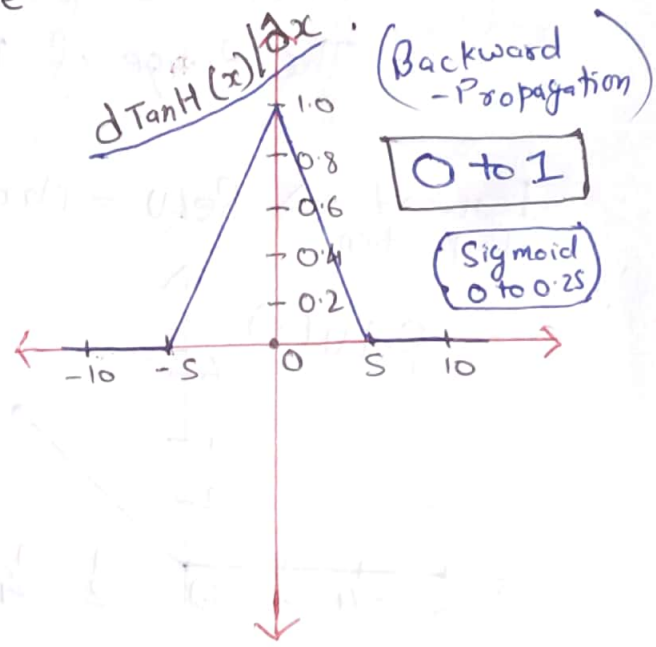
(Standardization)

a) TanH (Hyperbolic Tangent) :-

TanH Compress a real-valued Number to the range [-1 to 1]. It's non-Linear, But it's different from Sigmoid, and its output is zero-centered. The main Advantage of this is that the Negative inputs will be Mapped Strongly to the Negative and zero inputs will be mapped to almost zero in the graph of TanH.

Forward Propagation → $\text{TanH}\,x = \frac{e^{x} - e^{-x}}{e^{z} + e^{-z}} \Rightarrow -1 \text{ to } 1$





✱) Advantage

① TanH is Zero-Centered

DisAdvantage

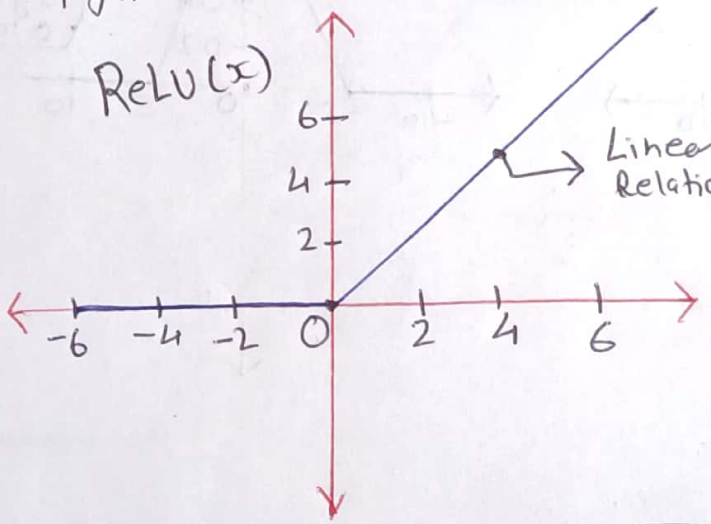
① Time Complexity more

② Vanishing Gradient Problem Still exists.

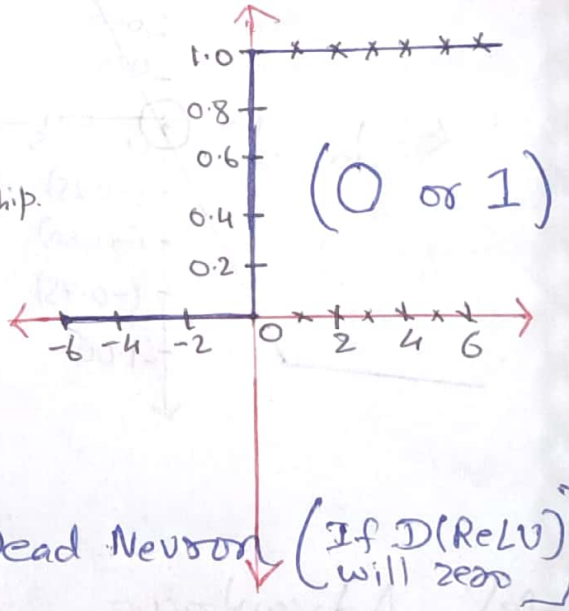
b) ReLU (Rectified Linear Unit) :— ReLU Stands

for Rectified Linear Unit and is one of the most commonly used Activation functions in the applications. It's Solved the problem of Vanishing Gradient because the Maximum value of the Gradient of ReLU function is one. It also Solved the problem of Saturating Neurons, Since the Slope is never zero for ReLU function. The Range of ReLU is between 0 and infinity.

Forward Propagation ⟹ ReLU = Max(0, x)

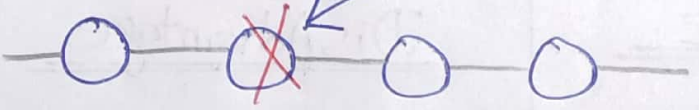


d ReLU(x)/dx



Dead Neuron (If D(ReLU) will zero

Eg :-



Advantage

① ReLU Solve the Vanishing Gradient Problem

② ReLU is faster than Sigmoid & TanH.

DisAdvantage

① If D(ReLU) will zero, then it Lead to Dead Neuron.

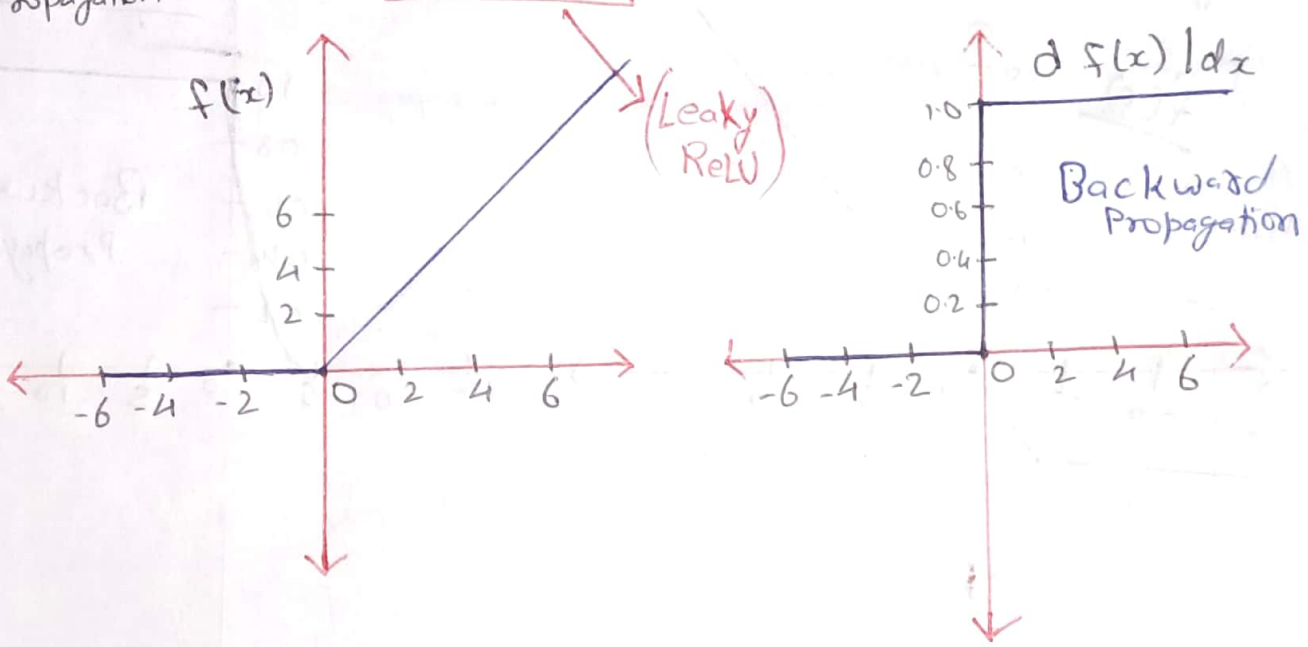
② ReLU is not a zero-centric Value.

c) Leaky ReLU or Parametric ReLU :— Leaky ReLU is an upgraded version of the ReLU Activation Function to Solve the dying ReLU problem, as it has a small positive slope in the negative area. But the Consistency of the benefit across tasks is presently ambiguous.

(Parametric ReLU)

(Hyper-Parameter)

forward Propagation $\Rightarrow f(x) = \max(0.01x, x) \rightarrow \max(\alpha x, x)$



✱) Advantage

(a) The Advantages of Leaky ReLU are the Same as that of ReLU.

(b) Leaky ReLU prevent the Dead Neuron.

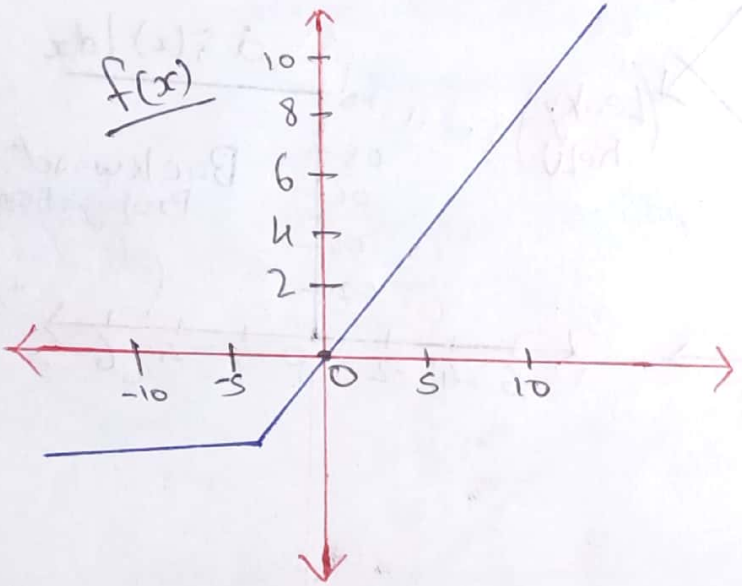
Dis Advantages

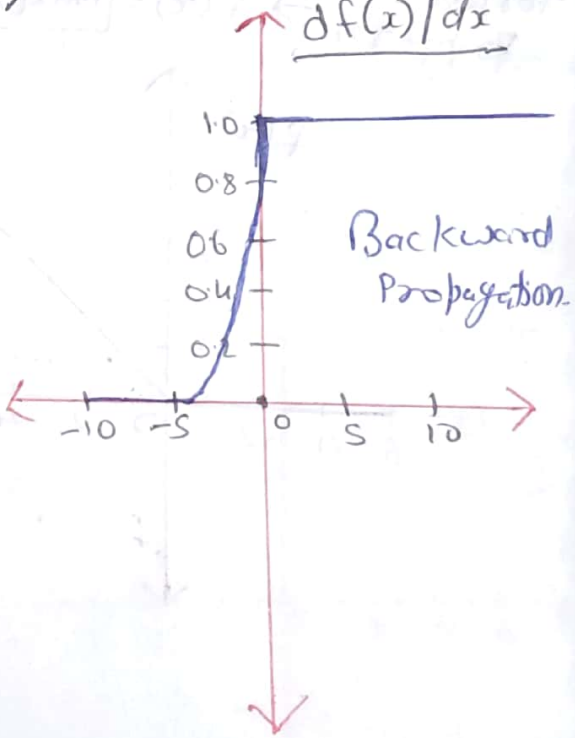
(a) The prediction may not be Steady for Negative input values.

d) ELU (Exponential Linear Units) :—

ELU is also one of the variations of ReLU which also solves the dead ReLU problem. ELU, just like Leaky ReLU also considers Negative Values by Introducing a new Alpha Parameter and Multiplying it will another equations.

Forward Propagation $\Rightarrow f(x) = \begin{cases} x, & \text{if } x > 0 \\ a(e^x - 1), & \text{otherwise} \end{cases}$





Advantage

(a) ELU is Zero-Centered.

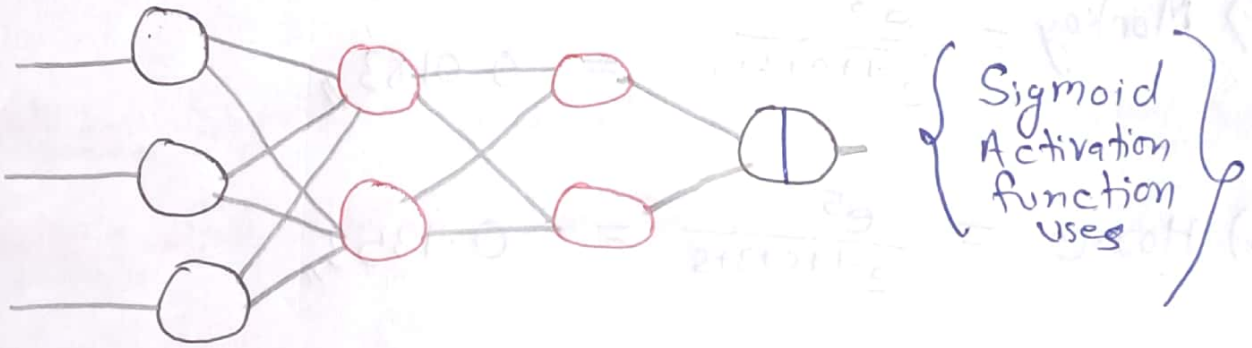
(b) ELU also preventing Dead Neuron.

DisAdvantage

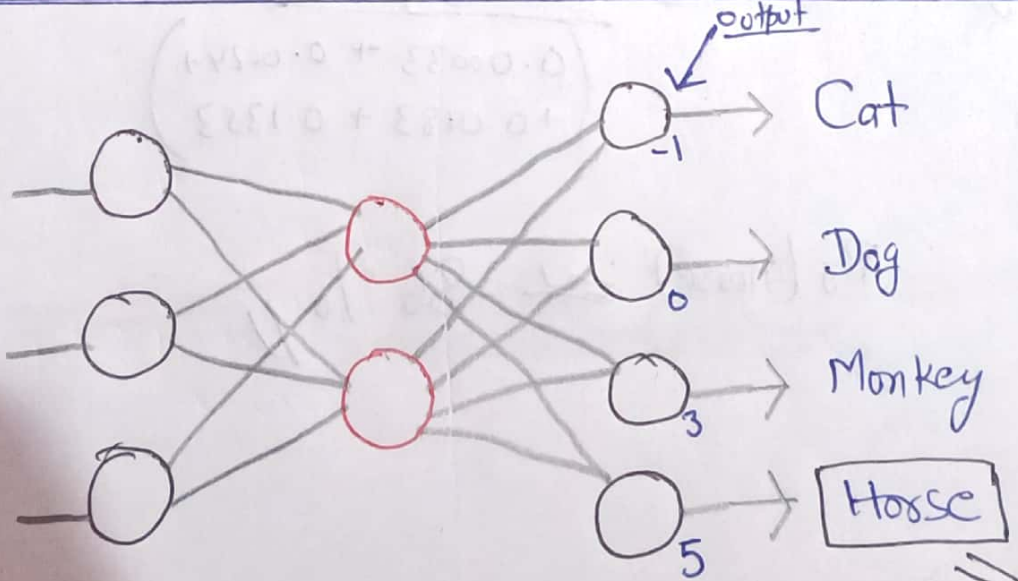
(a) ELU is Time Consuming.

e) Softmax Act :- A Combination of many Sigmoids is referred to as the Softmax function. It determines relative probability. Similar to the Sigmoid Activation function, the Softmax Function returns the Probability of each class/labels. In Multi-Class Classification, Softmax Activation function is most Commonly used for the Last Layer of the Neural Network.

* Binary Classification Problem :-



* Multi-Class Classification Problem



$\Rightarrow 86 \Rightarrow 0.86\%$
(Probability of Horse)

4) Softmax Activation Function :-

$$\text{Softmax} = \frac{e^{y_i}}{\sum_{K=0}^{n} e^{y_K}}$$

a) $\text{Cat} = \frac{e^{-1}}{e^{-1+0+3+5}} = 0.00033$ //

b) $\text{Dog} = \frac{e^{0}}{e^{-1+0+3+5}} = 0.0024$ //

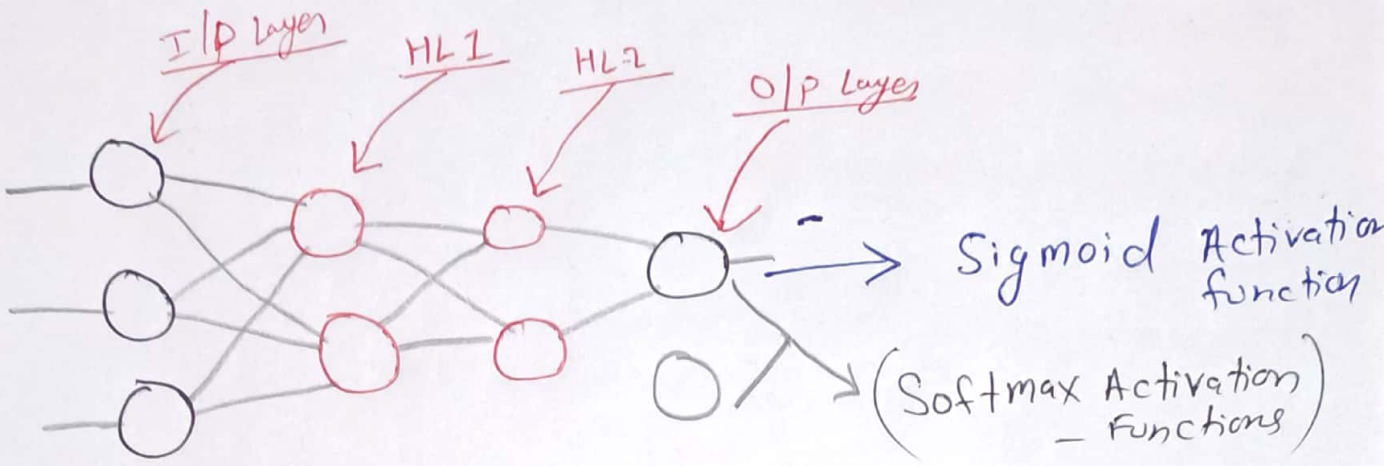
c) $\text{Monkey} = \frac{e^{3}}{e^{-1+0+3+5}} = 0.0183$ //

d) $\text{Horse} = \frac{e^{5}}{e^{-1+0+3+5}} = 0.1353$ //

(4) Porbability of Horse $= \frac{0.1353}{(0.00033 + 0.0024 + 0.0183 + 0.1353)}$

Pr (Horse) $\approx$ 86 % //

✱ Which Activation Function to use When :—



Note:- In Hidden Layer, we use ReLU and its variants (ReLU, PReLU, ELU)

Note:- Sigmoid function → Binary Classification Problem.

Note:- Softmax function → Multi-Class Classification Problem.

Note:- (Whenever Regression Problem → Linear Activation Function)

== Thank You ==