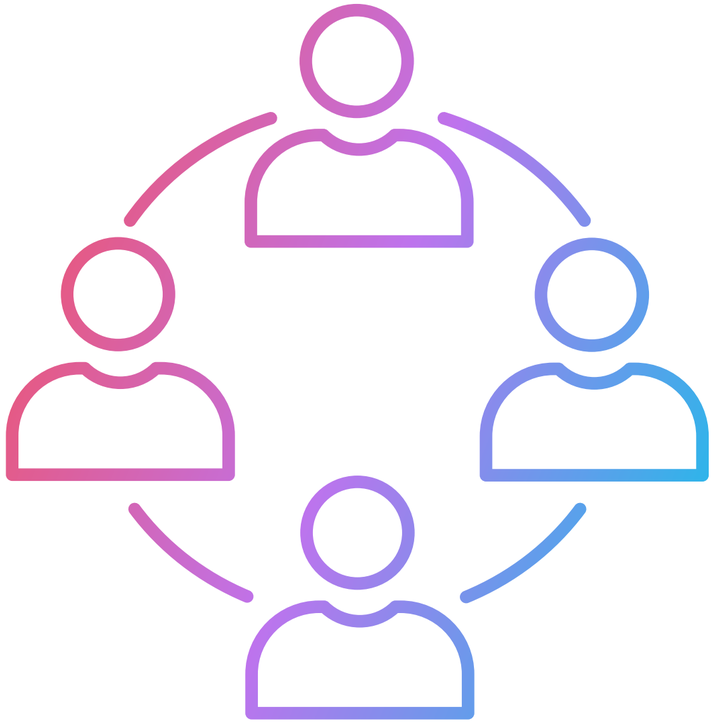
PERCEPTRON

Team



Team 1

1. Rohini Gudimetla
2. V Minmini

Team 2

3. Cheguri Teja Sree
4. P H Akanksha

Team 3

5. K B Amrutha Reddy
6. Naga Prasanth Nimmagadda

Team 4

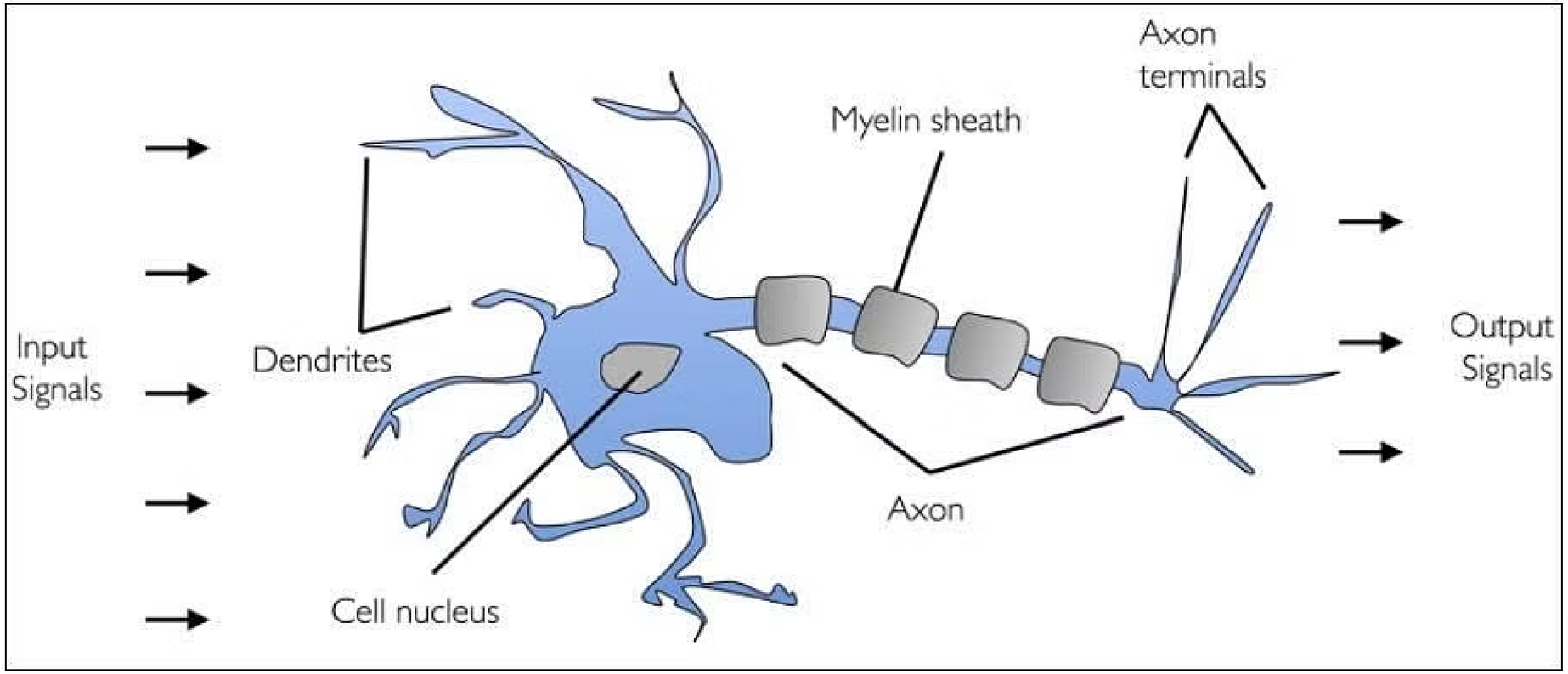
7. Poluru Venkata Koushik
8. Ruthvik Reddy Gaddam

Team 1

What is a Perceptron?

Let's understand a Biological Neuron first

- Dendrites are branches that receive information from other neurons.
- Cell nucleus or Soma processes the information received from dendrites.
- Axon is a cable that is used by neurons to send information.
- Synapse is the connection between an axon and other neuron dendrites

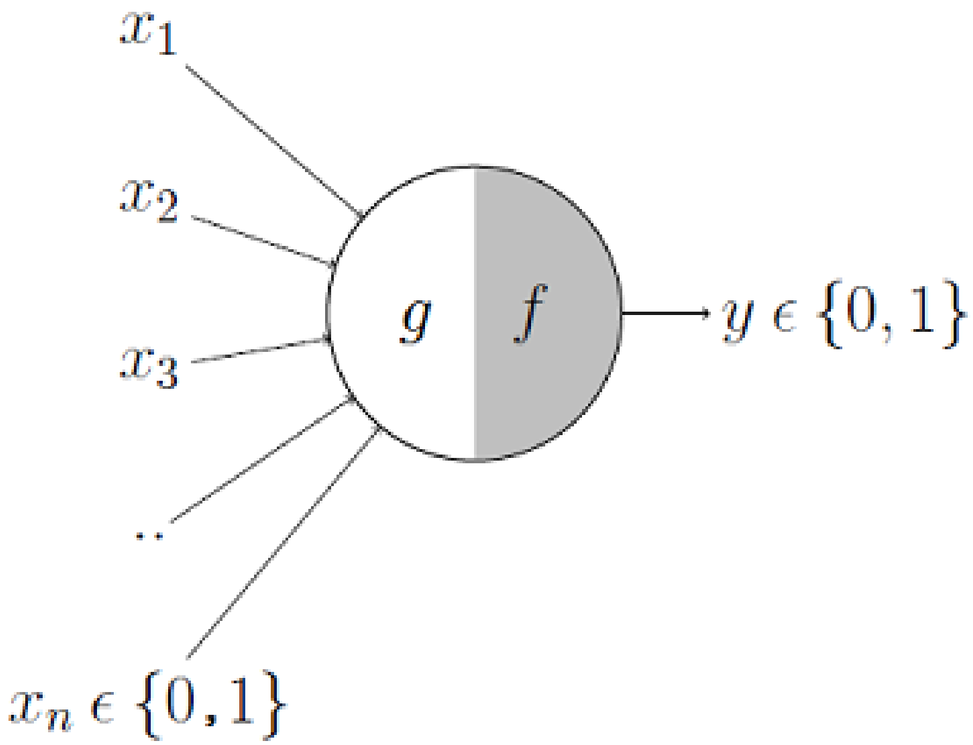


- Neurons in many regions of the brains are organized into layers
- Neuron in one layer receives inputs from adjacent layer
- Connections between layers are mostly in one direction
- Low level processing to higher level coordination

Biological Neuron	Artificial Neuron
Cell Nucleus (Soma)	Node
Dendrites	Input
Synapse	Weights or interconnections
Axon	Output

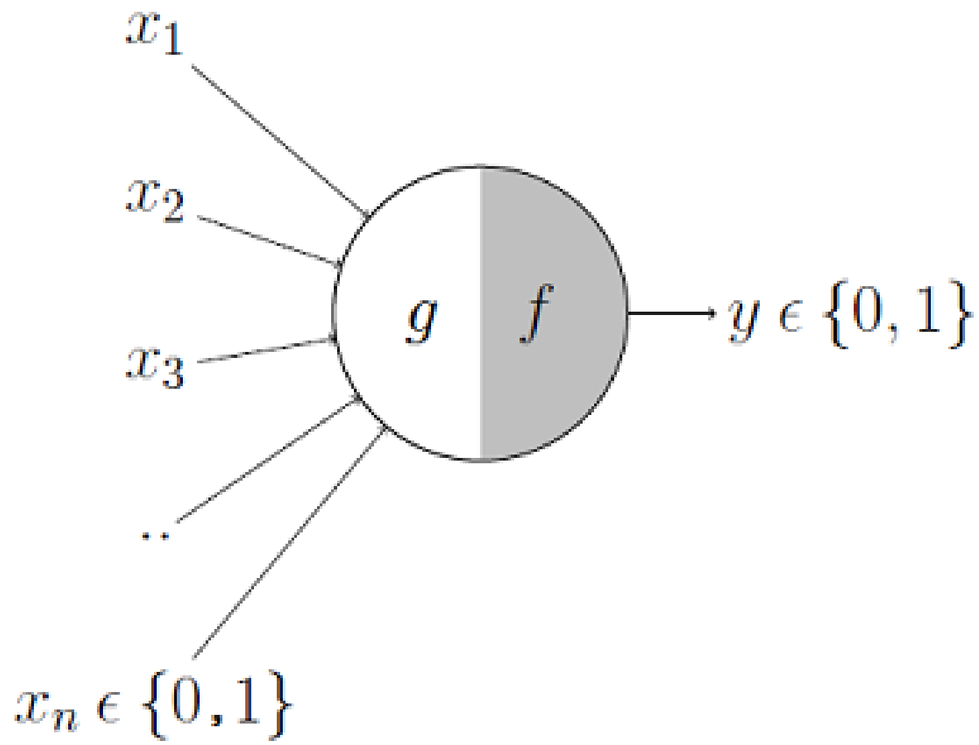
Rise of Artificial Neurons

- Researchers Warren McCullock and Walter Pitts published their first concept of simplified brain cell in 1943
- They described such a nerve cell as a simple logic gate with binary outputs.
- This was called McCullock-Pitts (MCP) neuron.



How Artificial Neurons work

- A neuron is a mathematical function modeled on the working of biological neurons
- It is an elementary unit in an artificial neural network
- One or more inputs are separately weighted
- Inputs are summed and passed through a nonlinear function to produce output

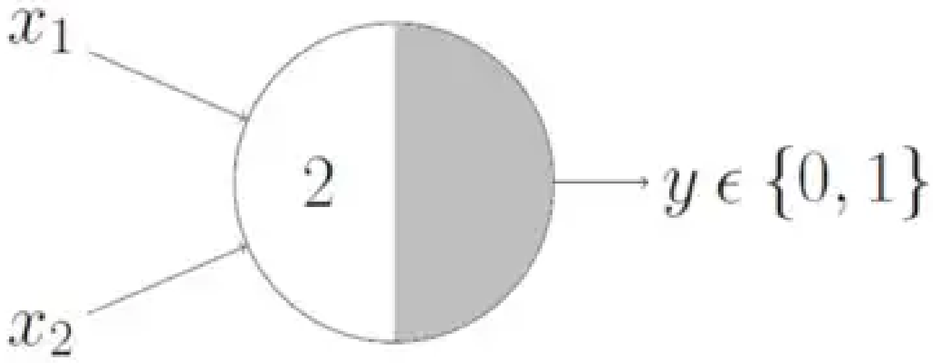


- g performs an aggregation and based on the aggregated value, f makes a decision

Implementation of McCulloch Pitts model for AND Gate

- Net input is yin=x1+x2
- Activation function =

$$\begin{cases} 1 & \text{if } y_{in} \geq 2 \\ 0 & \text{if } y_{in} < 2 \end{cases}$$



AND function

$$x_1 + x_2 = \sum_{i=1}^{2} x_i \geq 2$$

BUT WHAT IS A PERCEPTRON?!?!?!?

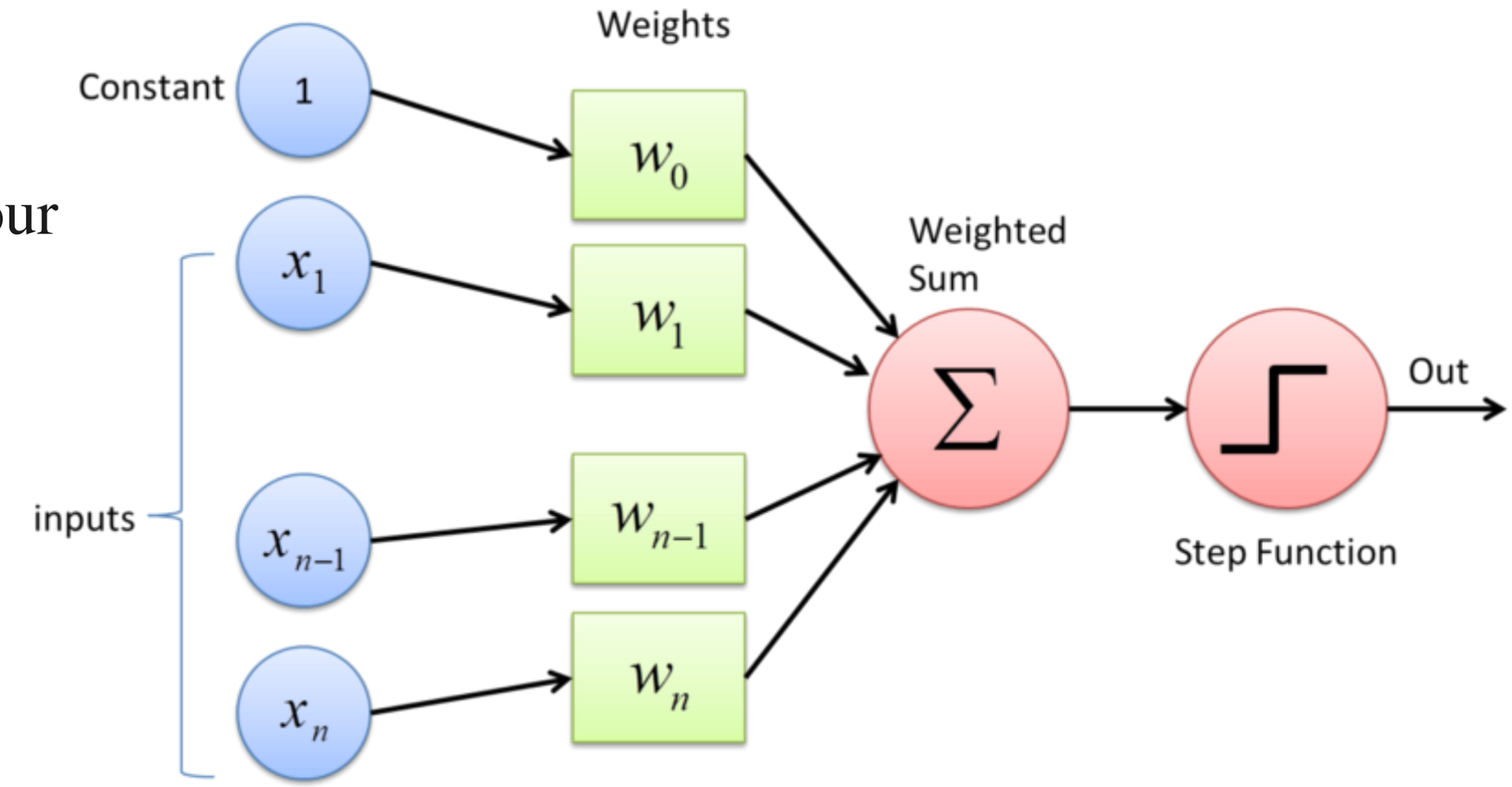
Perceptron

- Perceptron was introduced by Frank Rosenblatt in 1957.
- He proposed a Perceptron learning rule based on the original MCP neuron.
- An algorithm for supervised learning of binary classifiers.
- Enables neurons to learn and processes elements in the training set one at a time.

What does it consist of?

Perceptron is considered a single-layer neural link with four main parameters.

- Input Values
- Weights and bias
- Net sum
- Activation function



How does it work?

Frank Rosenblatt suggested this algorithm:

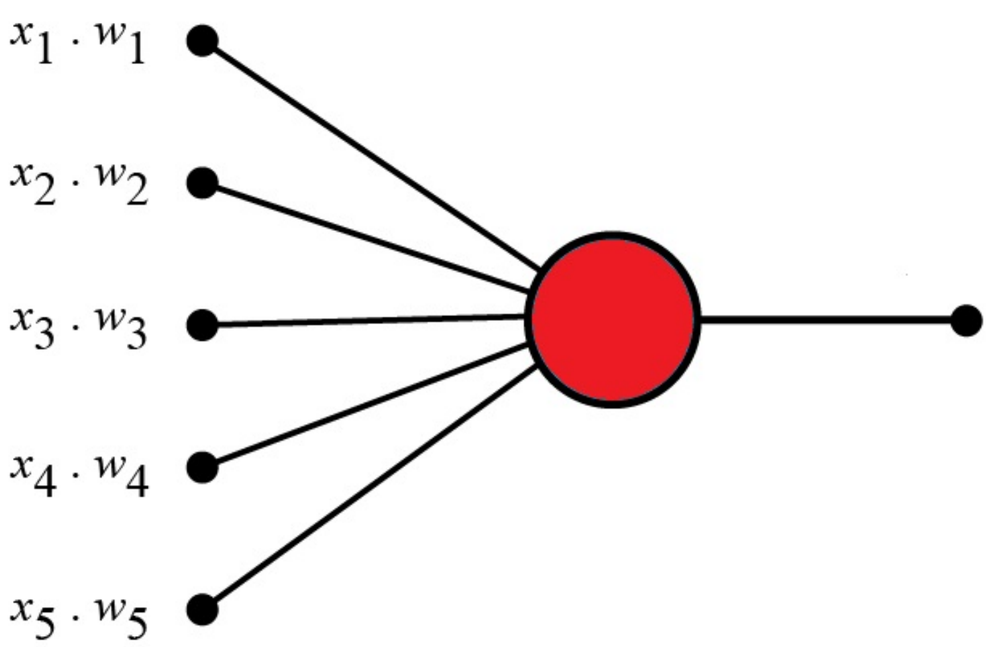
- Set a threshold value
- Multiply all inputs with its weights
- Sum all the results
- Activate the output

1. Set a threshold value

   Threshold = 0

2. Multiply all inputs with its weights:

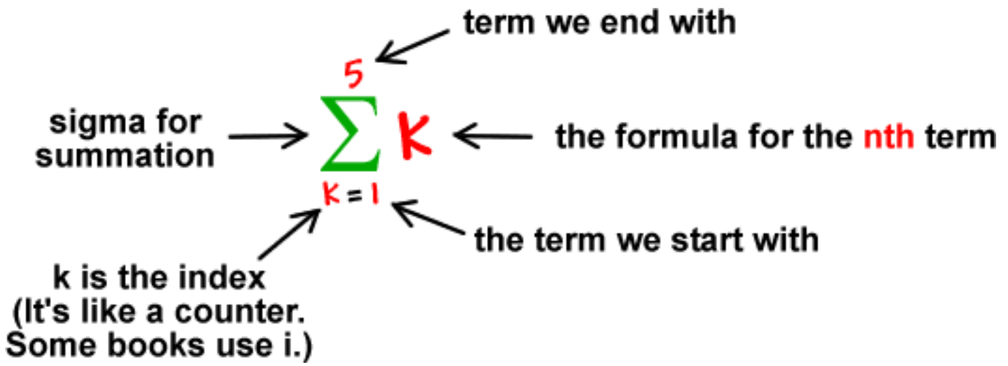
- x1 * w1 = 1 * 0.7 = 0.7
- x2 * w2 = 0 * 0.6 = 0
- x3 * w3 = 1 * 0.5 = 0.5
- x4 * w4 = 0 * 0.3 = 0
- x5 * w5 = 1 * 0.4 = 0.4



3. Sum all the results

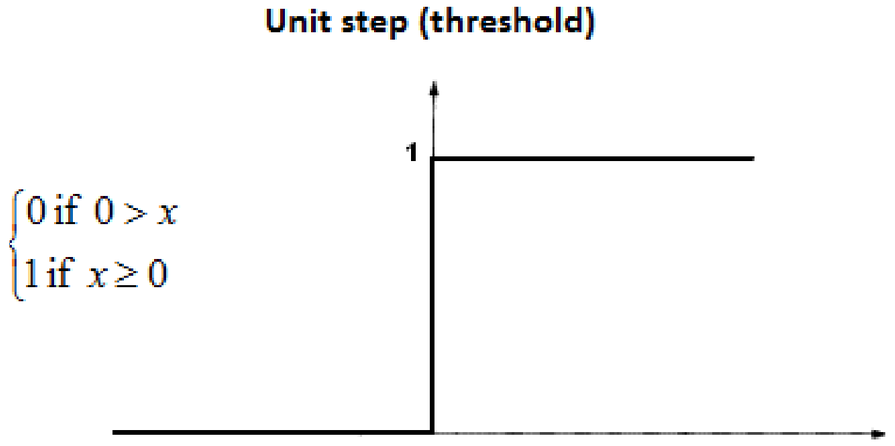
   0.7 + 0 + 0.5 + 0 + 0.4 = 1.6

(The Weighted Sum)



4. Activate the Output

   Return true if the sum > 0



Why weights, bias and activation function?

Why do we need Weights and Bias?

- Weights shows the strength of the particular node.
- A bias value allows you to shift the activation function curve up or down.

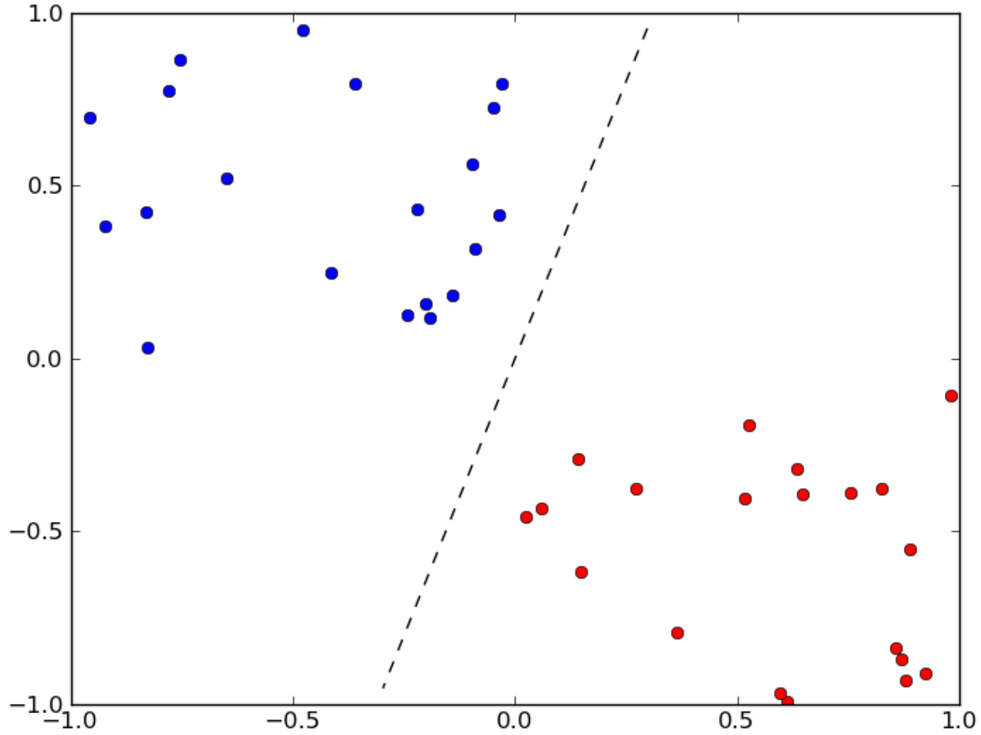
Why do we need Activation Function?

- The activation functions are used to map the input between the required values like (0, 1) or (-1, 1).

Where do we use Perceptrons?

Perceptron is usually used to classify and cluster information based on the specified parameters.

Therefore, it is also known as a Linear Binary Classifier.



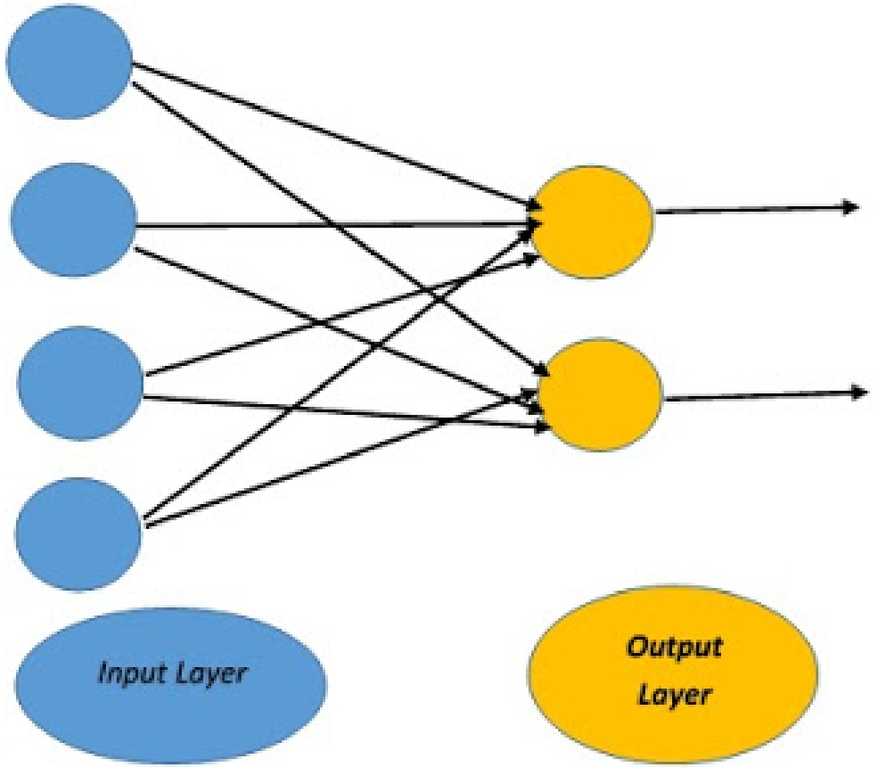
Team 2

Types of Perceptron Models

Single Layered Perceptron Model

- The first neural network model.
- Proposed in 1958 by Frank Rosenbluth
- Goal is to find a linear decision function measured by the weight vector w and the bias parameter b.
- It is necessary to comprehend artificial neural networks (ANNs).
- Often called perceptrons
- type of feed-forward neural network made up of input and output layers.

- Inputs provided are multi-dimensional
- Perceptrons are acyclic in nature
- The sum of the product of weights and the inputs is calculated in each node.
- The output can be represented in one or two values(0 or 1).



Advantages of Single-layered Neural Network

- These are easy to set up and train them as there is absence of hidden layers
- It has explicit links to statistical models.

Disadvantages of Single-layered Neural Network

- It can work better only for linearly separable data.
- It has low accuracy as compared to multi-layer neural network.

Algorithm

Step 1:- Initialize the weights and bias for easy calculation at them at 0. Also initialize learning rate α $(0 < \alpha \leq 1)$ for simplicity set α to 1.

Step 2:- Calculate output of the network. To do so, calculate net input first

$$y_{in} = b + \sum_{i=1}^{N} * w_i$$

$n \rightarrow$ no. of input neurons.
Apply activation function over the net input. Calculated to obtain the output.

$$Y = f(y_{in}) = \begin{cases} 1, & y_{in} > \theta \\ 0, & -\theta < f_{in} \leq \theta \\ -1, & y_{in} < -\theta \end{cases}$$

Step 3:- Weight and bias adjustment. Compare the value of the actual (calculated) and desired (target output).

if $y \neq t$, then

$$w_i(new) = w_i(old) + \alpha t x_i$$
$$b(new) = b(old) + \alpha t$$

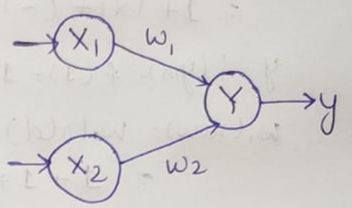
else, we have

$$w_i(new) = w_i(old)$$
$$b_i(new) = b(old)$$

Question 1:- Implement AND function using perceptron networks for bipolar inputs and targets.

0 → 1
1 → -1

x_1	x_2	t
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1



Initialize, $w_1 = w_2 = b = 0$ & $\theta = 0$, $\alpha = 1$

For 1st input :-

$x_1 = 1 = x_2 = t$

Calculate net input,

$y_{in} = b + w_1x_1 + w_2x_2$

$= 0 + 0 + 0 = 0$

Check with the activation function

$$y = f(y_{in}) = \begin{cases} 1, & y_{in} > 0 \\ 0, & y_{in} = 0 \\ -1, & y_{in} < 0 \end{cases}$$

$f(0) = 0 \neq t$, hence update the weights

$w_1(new) = w_1(old) + \alpha t x_1$

$= 0 + 1 * 1 * 1 = 1$

$w_2(new) = w_2(old) + \alpha t x_1$

$= 0 + 1 \times 1 \times 1 = 1$

$b(new) = b(old) + \alpha t$

$= 0 + 1 \times 1 = 1$

$\Delta w_1 = \alpha t x_1 = 1$
$\Delta w_2 = \alpha t x_2 = 1$
$\Delta b = \alpha t = 1$

For II input

$x_1 = 1, \; x_2 = -1, \; t = -1.$

$y_{in} = b + x_1w_1 + x_2w_2$

$= 1 + 1 \times 1 + (-1) \times 1 = 1$

$y = f(y_{in}) = f(1) = 1 \neq t$, so update

$w_1(new) = w_1(old) + \alpha t x_1$

$= 1 + 1 \times (-1)(1) = 0$

$w_2(new) = w_2(old) + \alpha t x_2$

$= 1 + (1)(-1)(-1) = 2$

$b(new) = b(old) + \alpha t$

$= 1 + 1 * -1 = 0$

$\Delta w_1 = -1, \; \Delta w_2 = -1, \; \Delta b = -1$

For III input,

$x_1 = -1, \; x_2 = 1, \; t = -1$

$y_{in} = b + w_1x_1 + w_2x_2$

$= 0 + (-1)(0) + (1)(2) = 2$

$y = f(y_{in}) = f(2) = 1 \neq t$, hence update weights.

$w_1(new) = w_1(old) + \alpha t x_1$

$= 0 + 1 \times (-1)(-1) = 1.$

$w_2(new) = w_2(old) + \alpha t x_2$

$= 2 + 1 \times -1 \times 1 = 1.$

$b(new) = b(old) + \alpha t$

$= 0 + (1)(-1) = -1$

$\Delta w_1 = 1,$
$\Delta w_2 = -1,$
$\Delta b = -1$

For IV input:-

$x_1 = -1,\ x_2 = -1,\ t = -1$

$y_{in} = b + w_1 x_1 + w_2 x_2$

$= -1 + (1)(-1) + (1)(-1) = -3$

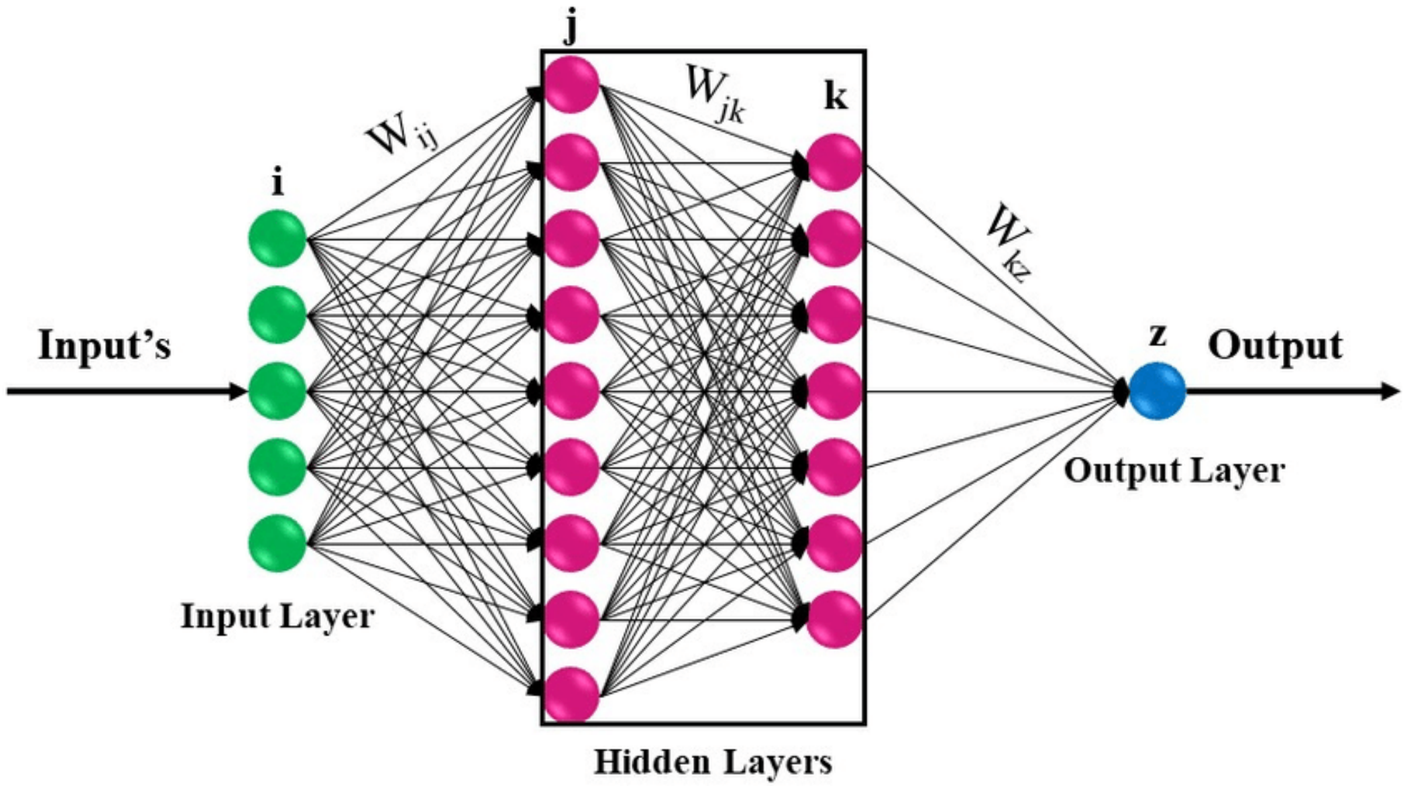
$y = f(y_{in}) = f(-3) = -1 = t$. Hence don't update weights.

$\Delta w_1 = 0,\ \Delta w_2 = 0,\ \Delta b = 0$

$w_1 = 1,\ w_2 = 1,\ b = -1$

x_1	x_2	t	y_{in}	y	Δw_1	Δw_2	Δb	w_1	w_2	b
								1	1	1
1	1	1	0	0	1	1	1			
1	−1	−1	1	1	−1	1	−1	0	2	0
1	1	−1	2	1	1	−1	−1	1	1	−1
−1	−1	−1	−3	−1	0	0	0	1	1	−1

Multi-layered Perceptron Model



Multi-layered Perceptron Model

- Similar to a single-layer perceptron model, the multi-layer perceptron model also implements the identical model structure but with more hidden layers.
- It is also known as backpropagation algorithm.
- It works in two stages:

1. Forward Stage
2. Backward Stage

Multi-layered Perceptron Model

Advantages:

- A multi-layered perceptron model can be used to solve complex non-linear problems.
- It works well with both small and large input data.
- It helps us to obtain quick predictions after the training.
- It helps to obtain the same accuracy ratio with large as well as small data.

Multi-layered Perceptron Model

Disadvantages:

- In Multi-layer perceptron, computations are difficult and time-consuming.
- In multi-layer Perceptron, it is difficult to predict how much the dependent variable affects each independent variable.
- The model functioning depends on the quality of the training.

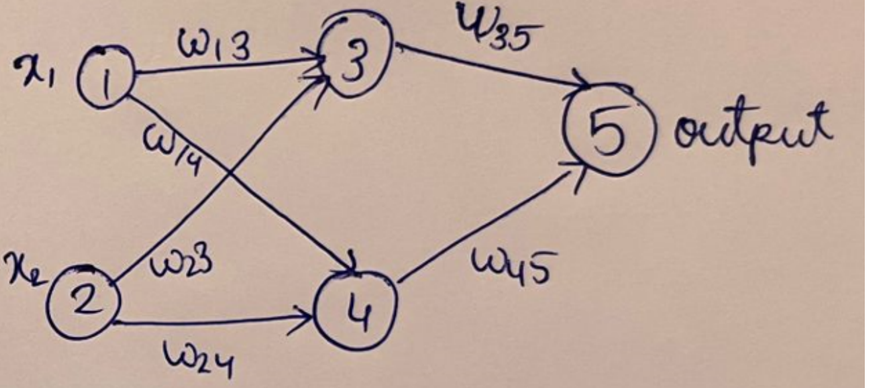
Example

$w_{12} = 2, \quad w_{35} = 2$

$w_{23} = -3, \quad w_{45} = -1$

$w_{14} = 1 \qquad w_{24} = 4$

$x_1 = 1, \quad x_2 = 0$



$$f(x) = \begin{cases} 1 & ; \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

① $V_3 = w_{13}x_1 + w_{23}x_2$

$= 2(1) + (-3)(0)$

$= 2$

$f(x) = 1$

② $V_4 = w_{14}x_1 + w_{24}x_2$

$= 1(1) + 4(0)$

$= 1$

$f(x) = 1$

③ $V_5 = w_{35} \times V_3 + V_{45} \times V_4$

$= 2(1) + (-1)(1)$

$= 1$

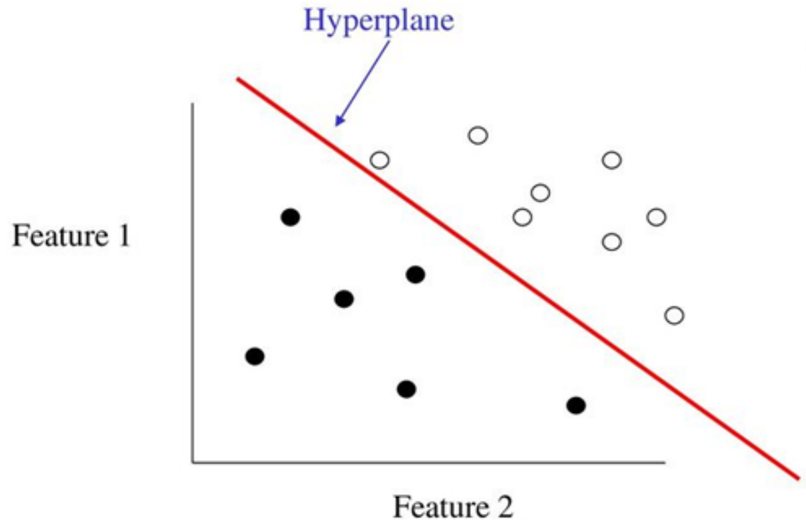
$f(x) = 1$

Team 3

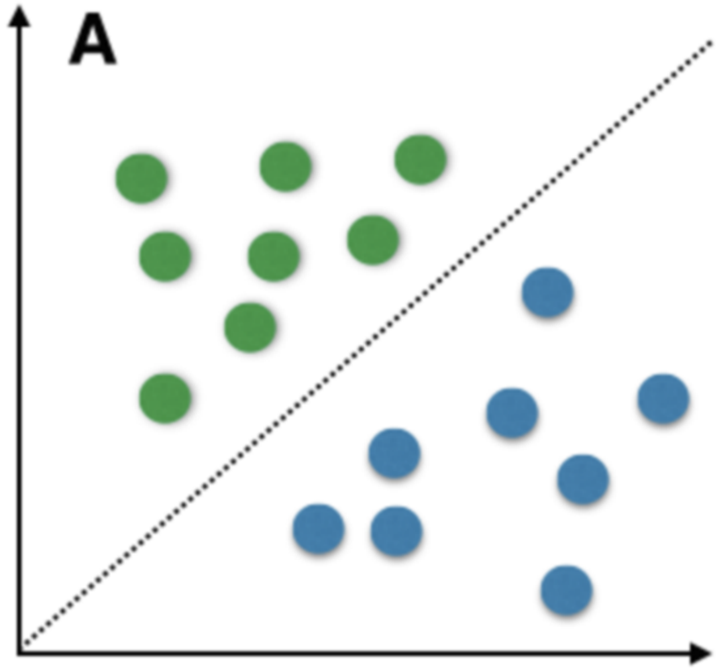
Introduction

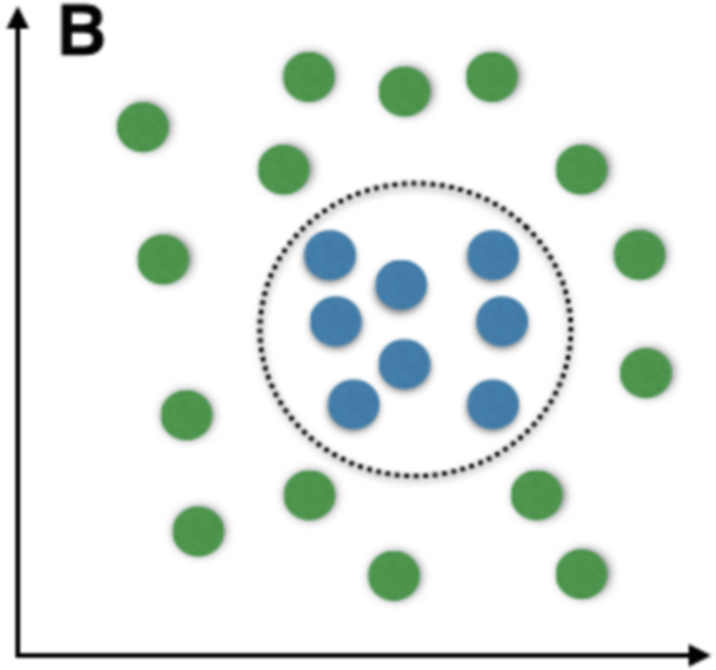
Linear Separability:

- Linear Separability is the concept wherein the separation of the input space into regions is based on whether the network response is positive or negative.
- A set of points in n-dimensional space are linearly separable if there is a hyper plane of n-1 dimensions that separates the sets.
- A perceptron can seperate data that is linearly separable.



Which one is Linearly Separable?





- Initialize the weights to zero or to small random values or to some initial guesses.
- Compute, $D = b + \sum x_i w_i$

  Where D - net input,

  b -bias,

  x_i - input from neuron i

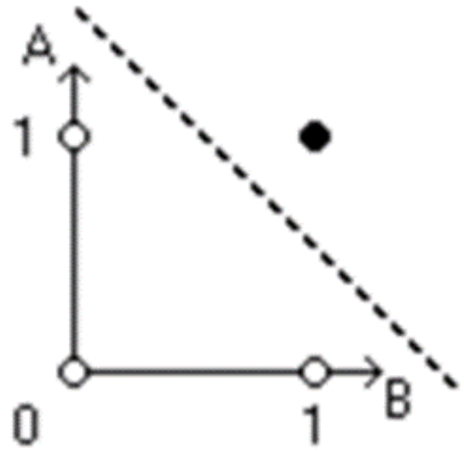
  w_i - weight from neuron i
- If D is not equal to threshold then update bias,weights.where ‘t’ is threshold and ‘ ’ is learning rate usually taken as 0 or 1.

$$b = b + \alpha t$$

$$w_i(new) = w_i(\text{old}) + \alpha t x_i$$

Linear Separability for AND gate using Binary Inputs

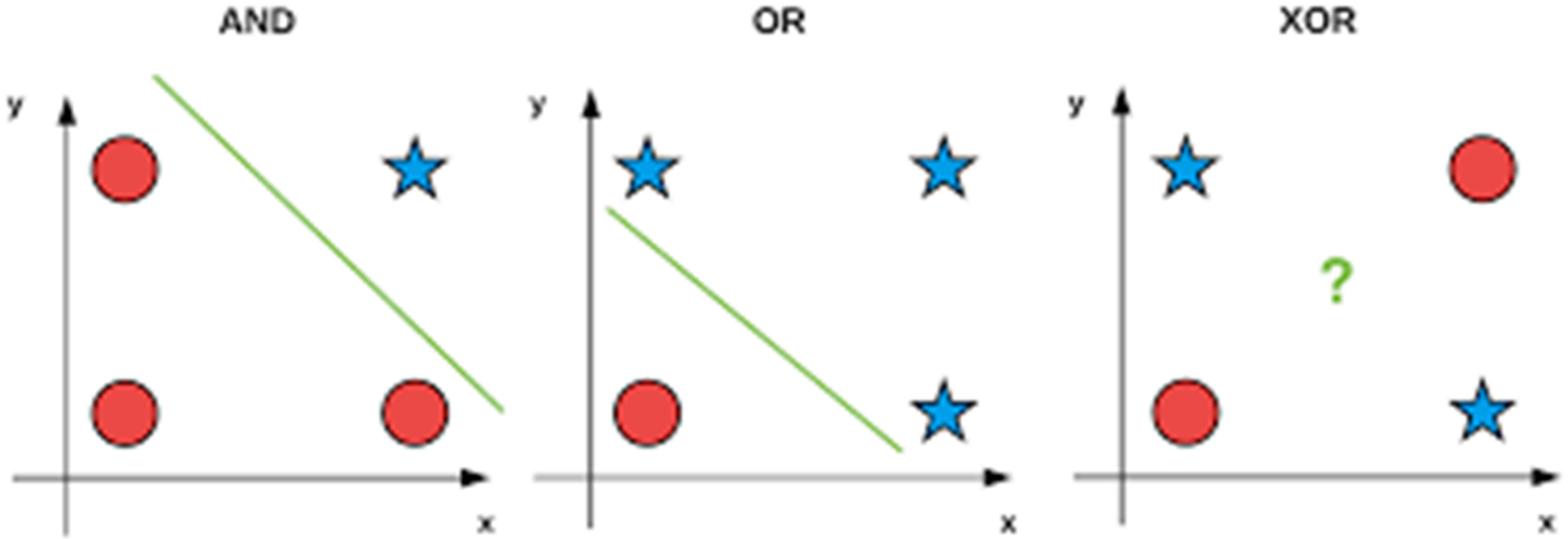
A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1



t	i	x1	x2	b	w1	w2	D	Error	New b	New w1	New w2
1	1	0	0	0	0	0	0	No	0	0	0
2	2	0	1	0	0	0	0	No	0	0	0
3	3	1	0	0	0	0	0	No	0	0	0
4	4	1	1	0	0	0	0	Yes	1	1	1
5	1	0	0	1	1	1	1	No	1	1	1
6	2	0	1	1	1	1	2	No	1	1	1
7	3	1	0	1	1	1	2	No	1	1	1
8	4	1	1	1	1	1	3	No	1	1	1

Consider if D>=3 , then sign of D is 1 else 0.

Separability for Gates



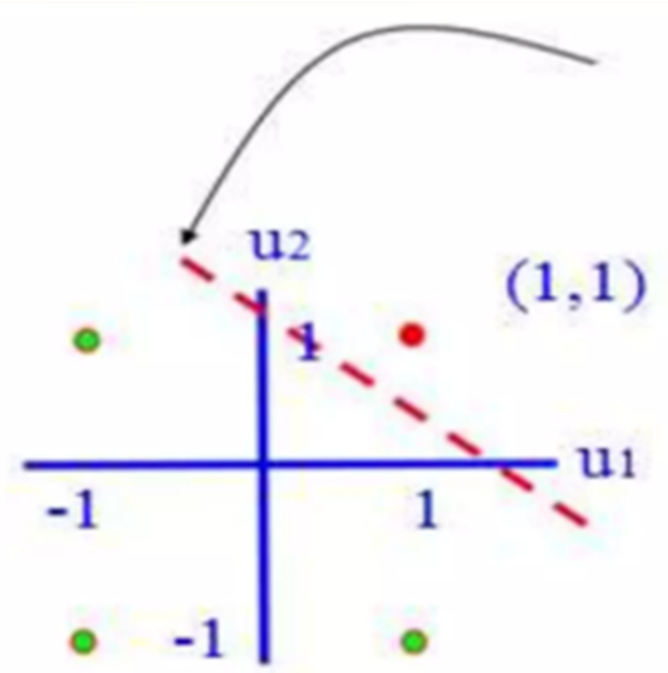
Bipolar Perceptron Model for AND gate

Binary Inputs

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Bipolar Inputs

A	B	Output
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1



t	i	x1	x2	b	w1	w2	D	Error	New b	New w1	New w2
1	1	-1	-1	0	0	0	0	No	0	0	0
2	2	-1	1	0	0	0	0	No	0	0	0
3	3	1	-1	0	0	0	0	No	0	0	0
4	4	1	1	0	0	0	0	Yes	1	1	1
5	1	-1	-1	1	1	1	-1	No	1	1	1
6	2	-1	1	1	1	1	1	Yes	0	2	0
7	3	1	-1	0		0	2	Yes	-1	1	1
8	4	1	1	-1	1	1	1	No	-1	1	1
9	1	-1	-1	-1	1	1	-1	No	-1	1	1
10	2	-1	1	-1	1	1	-1	No	-1	1	1
11	3	1	-1	-1	1	1	-1	No	-1	1	1
12	4	1	1	-1	1	1	1	No	-1	1	1

Consider if D>0 , then sign of D is 1 else -1

Limitations of Perceptron

- Only the boolean functions given below are linearly separable:

  AND

  OR

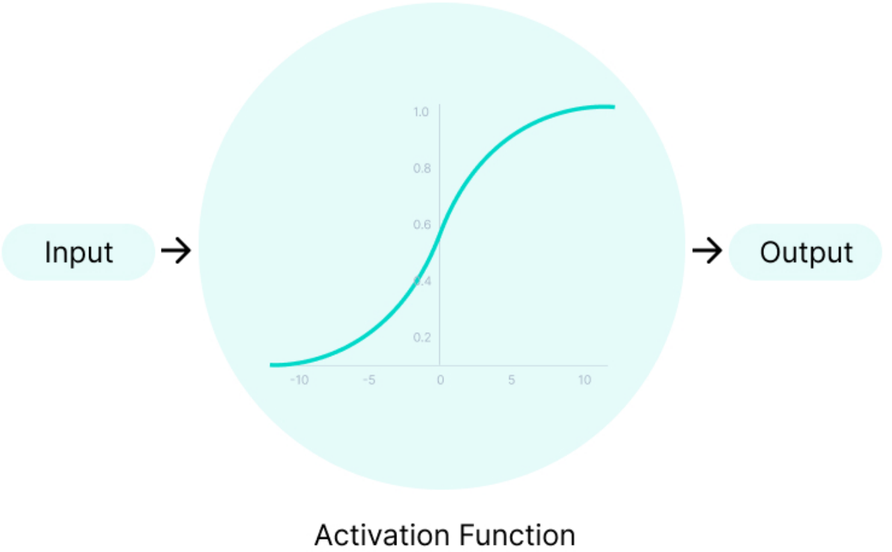
  COMPLEMENT
- It cannot model XOR function as it is non linearly separable.
- When the two classes are not linearly separable,it may be desirable to obtain a linear separator that minimizes the mean squared error.

Team 4

AGENDA

- Why do Neural Networks Need an activation function?
- 3 Types of Activation functions
- How to choose the right activation function?

WHAT IS ACTIVATION FUNCTION?



V7 Labs

- Decides whether neuron should be activated or not
- Role : Derive output from a set of input values fed to a node

WHY DO NEURAL NETWORKS NEED AN ACTIVATION FUNCTION?

- To add Non Linearity to neural network
- Without Activation Function:
  - Learning complex task becomes impossible
  - Model would be just a linear regression model

TYPES OF ACTIVATION FUNCTIONS

- Binary Step Function
- Linear Activation Function
- Non-Linear Activation Function

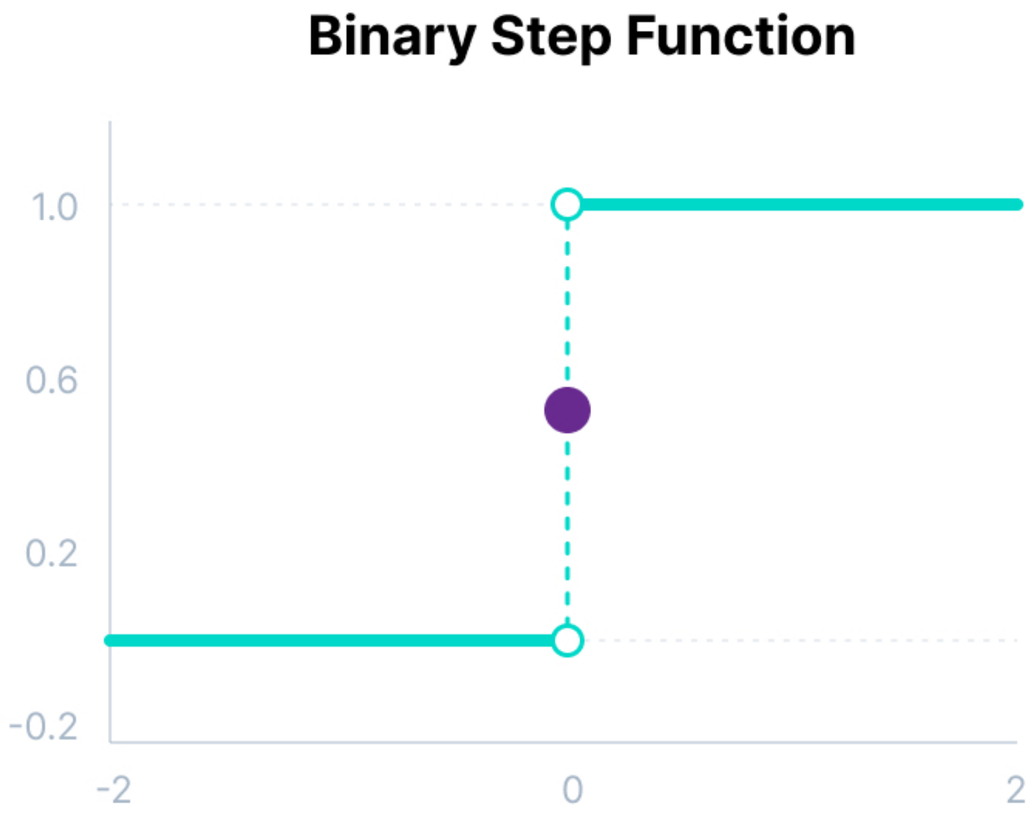
BINARY STEP FUNCTION

- Depends on a threshold value that decides whether a neuron should be activated or not

$$\textit{Binary step}$$

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

BINARY STEP FUNCTION



V7 Labs

BINARY STEP FUNCTION

- Disadvantages:
  - Cannot provide Multi Value Outputs
  - Gradient of step function is 0
    - Causes Hindrance in backpropogation process

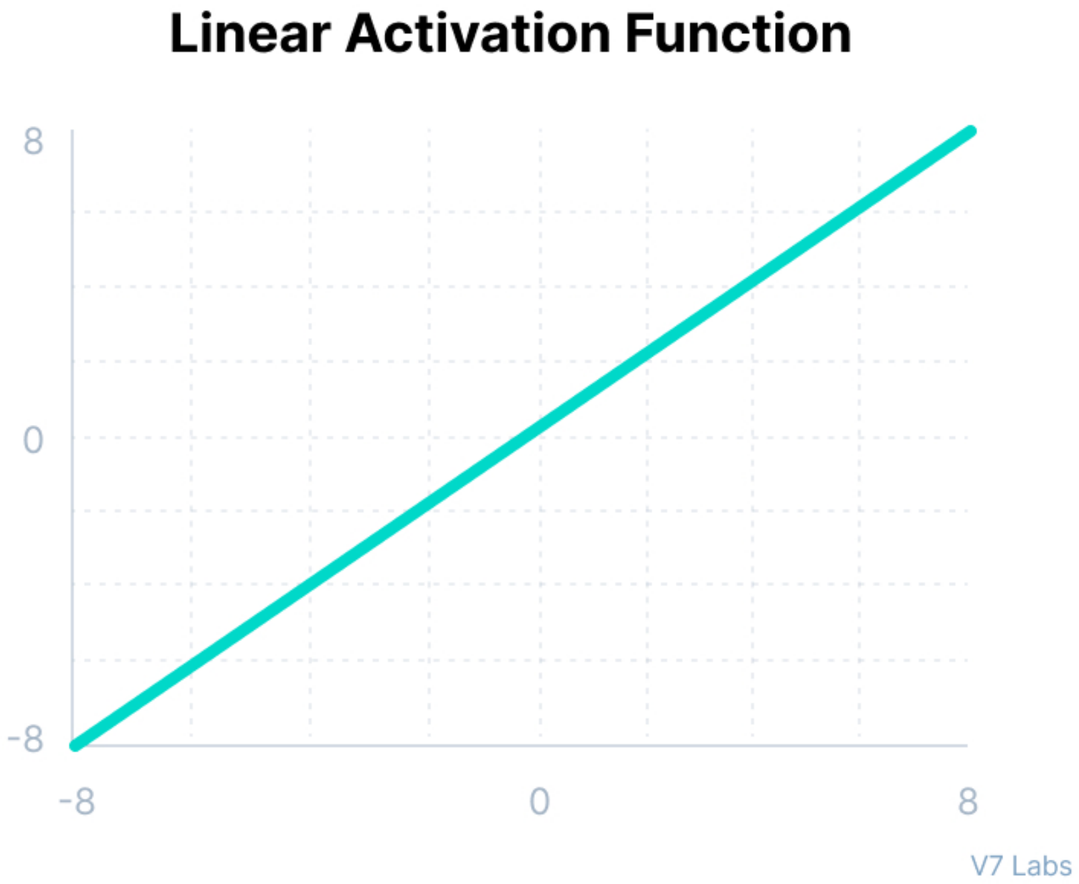
LINEAR ACTIVATION FUNCTION

- Identity function
- Simply spits out the value given

$$Linear$$

$$f(x) = x$$

LINEAR ACTIVATION FUNCTION



LINEAR ACTIVATION FUNCTION

- Disadvantages:
  - Not Possible to use backpropogation
  - All layers will collapse into one if a linear activation function is used

Simple Perceptron

Non Linear Activation Functions

Sigmoid	Tanh	Rectified Linear Unit (ReLU)	Leaky ReLU	Parametric ReLU
Exponential Linear Unit (ELU)	Softmax	Swish	Gaussian Error Linear Unit (GELU)	Scaled Exponential Linear Unit (SELU)

Sigmoid Activation Function

Sigmoid / Logistic

$$f(x) = \frac{1}{1 + e^{-x}}$$

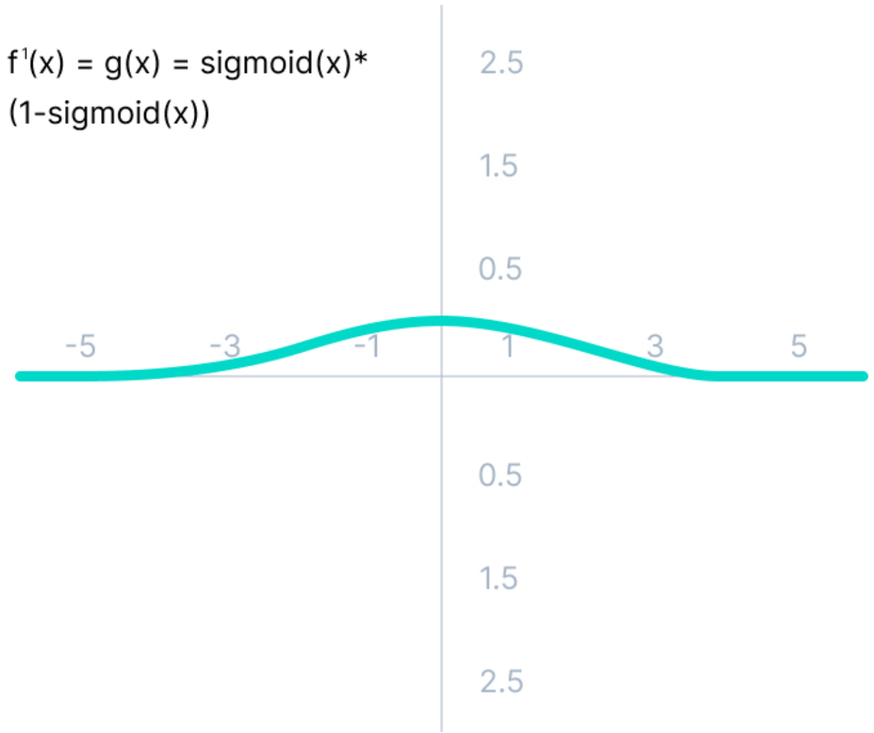
- The function takes any real value as input and outputs values in the range of 0 to 1
- More positive input has output closer to 1
- More negative input has output closer to 0

Advantages

- It is commonly used for models where we have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice because of its range
- The function is differentiable and provides a smooth gradient, i.e., preventing jumps in output values. This is represented by an S-shape of the sigmoid activation function

Disadvantages

- The derivative of the function is
  - **f'(x) = sigmoid(x)*(1-sigmoid(x))**
- As the gradient value approaches zero, the network ceases to learn and suffers from the Vanishing gradient problem
- The output of the function is not symmetric around zero.
  - So the output of all the neurons will be of the same sign
  - This makes the training of the neural network more difficult and unstable

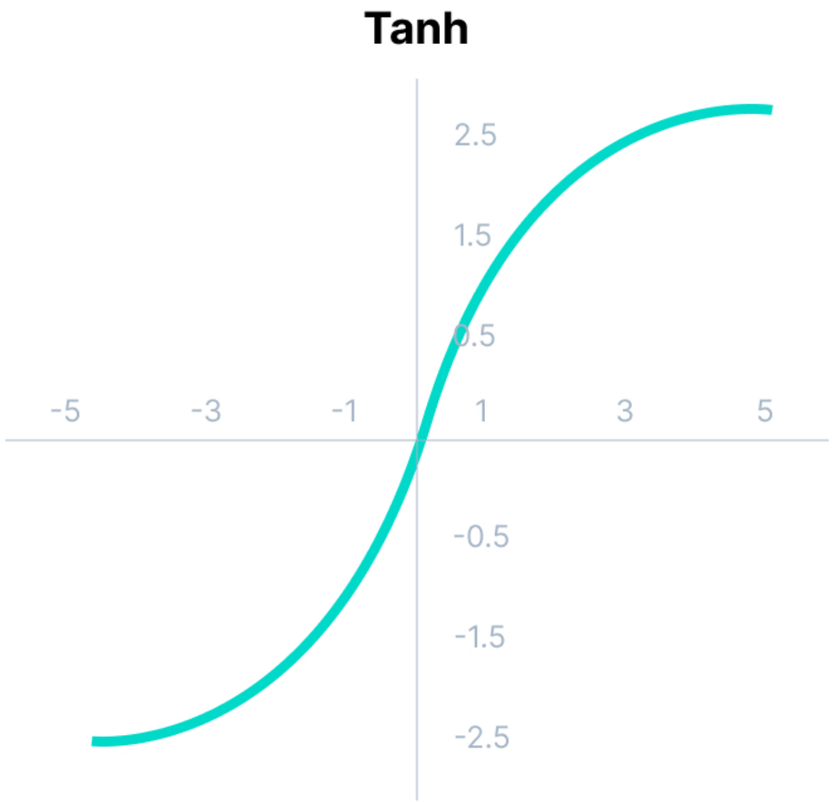


Tanh Activation Function

$$Tanh$$

$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$

- Tanh function is very similar to the sigmoid/logistic activation function
- It has the same S-shape with the difference in output range of -1 to 1

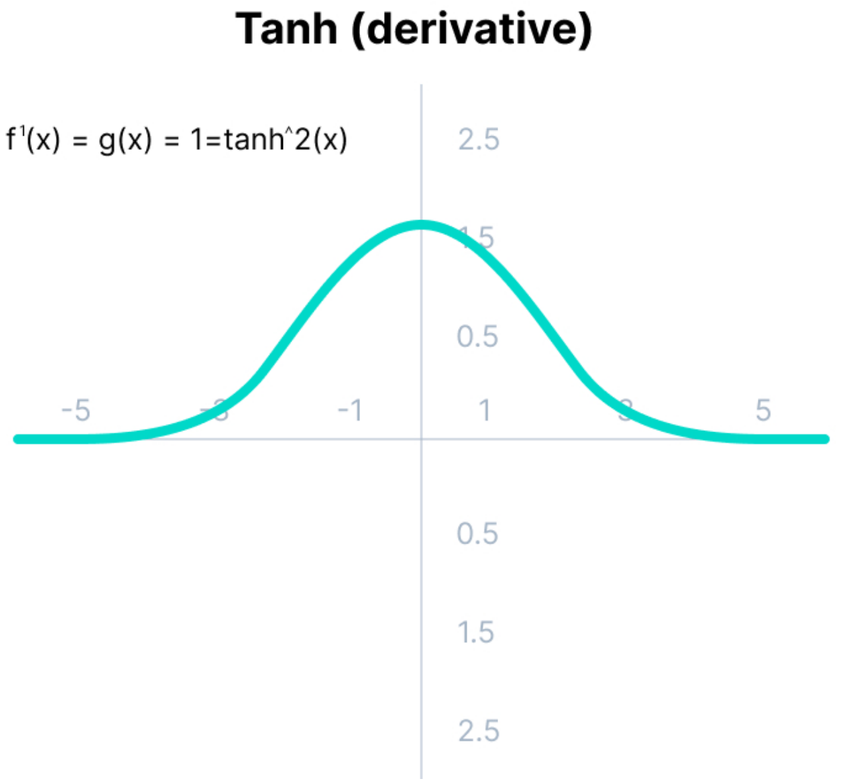


Advantages

- The output of the function is Zero centered
    - output values can easily be mapped as **strongly negative, neutral**, or **strongly positive**
- Usually used in hidden layers of a neural network as its values lie between -1 to 1
- The mean for the hidden layer comes out to be 0 or very close to it
- It helps in centering the data and makes learning for the next layer much easier

Disadvantages

- it also faces the problem of vanishing gradients similar to the sigmoid activation function. Plus the gradient of the tanh function is much steeper as compared to the sigmoid function

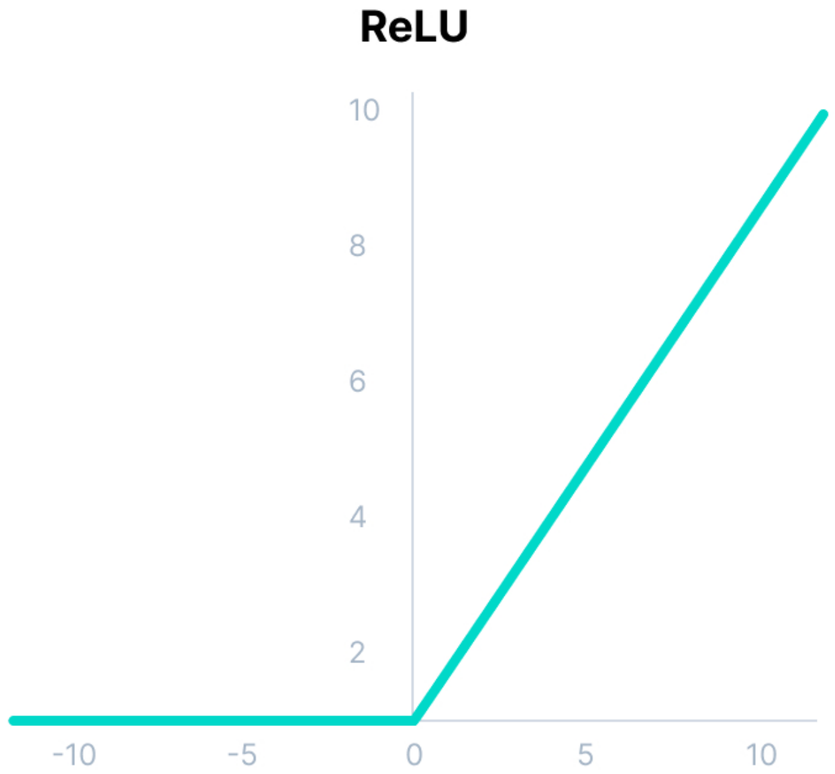


ReLU Activation Function

$$ReLU$$

$$f(x) = max\,(0, x)$$

- ReLU has a derivative function and allows for backpropagation while simultaneously making it computationally efficient
- It does not activate all the neurons at the same time
- The neurons will only be deactivated if the output of the linear transformation is less than 0

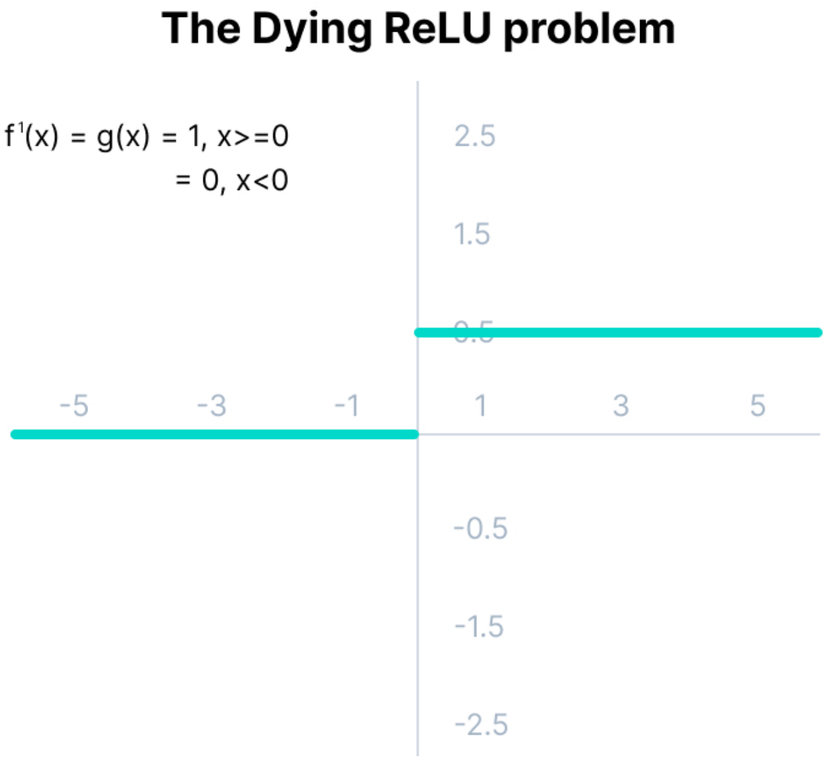


Advantages

- Since only a certain number of neurons are activated, the ReLU function is far more computationally efficient when compared to the sigmoid and tanh functions.
- ReLU accelerates the convergence of gradient descent towards the global minimum of the loss function due to its linear

Disadvantages

- The Dying ReLU problem
- The negative side of the graph makes the gradient value zero
- During the backpropagation process, the weights and biases for some neurons are not updated. This can create dead neurons which never get activated
- All the negative input values become zero immediately, which decreases the model's ability to fit or train from the data properly

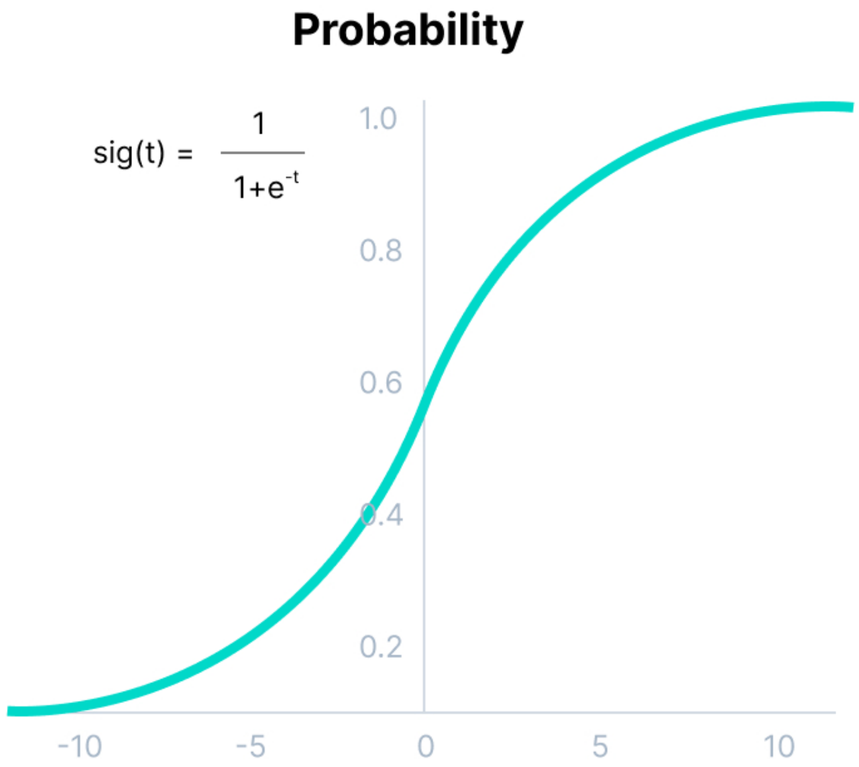


Softmax Activation Function

Softmax

$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

- The Softmax function is described as a combination of multiple sigmoids V7 Labs
- It calculates the relative probabilities
- It returns the probability of each class
- It is commonly used as an activation function for the last layer of the neural network in the case of multi-class classification



How to choose the right activation function?

- match your activation function for your output layer based on the type of predicted variable
- Rule of thumb - begin with using the ReLU activation function, then move over to other activation functions if ReLU doesn’t provide optimum results
- ReLU activation function should only be used in the hidden layers
- Sigmoid function should not be used in hidden layers as they make the model more susceptible to problems during training (due to vanishing gradients)

How to choose the right activation function?

- Rules for choosing an activation function for the output layer based on the type of prediction problem :
  - Regression - Linear Activation Function
  - Binary Classification—Sigmoid/Logistic Activation Function
  - Multiclass Classification—Softmax

- The activation function used in hidden layers is typically chosen based on the type of neural network architecture
  - Convolutional Neural Network (CNN): ReLU activation function
  - Recurrent Neural Network: Tanh and/or Sigmoid activation function

Implementation

Simple Perceptron

(Click on Links)

Thank you!