DEEP LEARNING
(CSE436)

BY: Nidhi S. Periwal,
Teaching Assistant,
COED, SVNIT, Surat

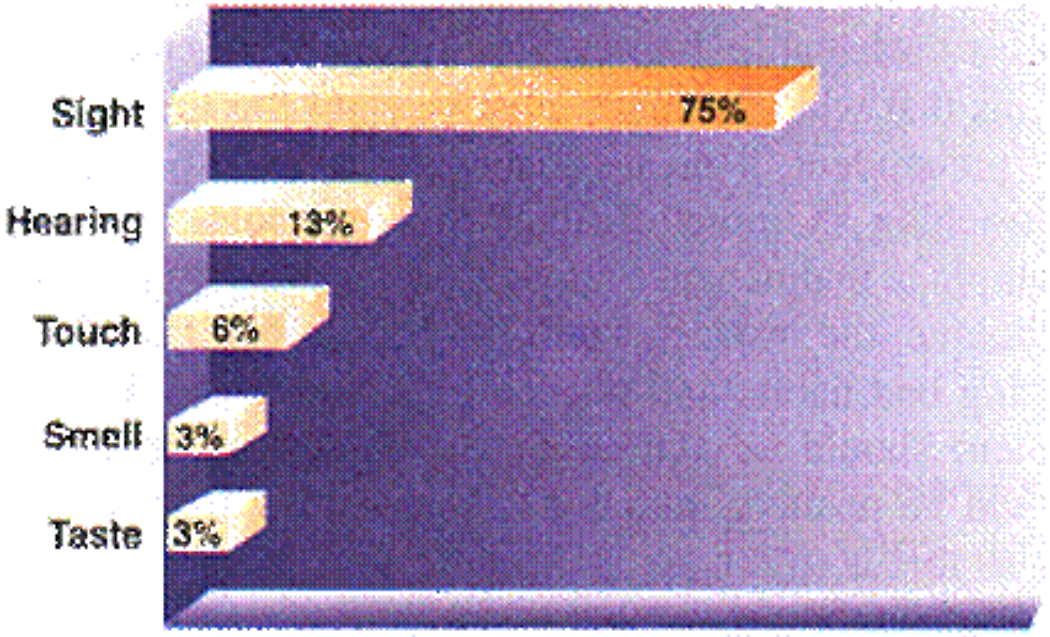
Introduction

Learning

- Learning is defined as “any relatively permanent change in behaviour that occurs as a result of experience and practice”
- Learning occurs most rapidly on a schedule of ***continuous*** **reinforcement**.
- Human learning **begins before birth and continues until death as a consequence of ongoing interactions between person and environment**.
- All learning comes from perceptions which are directed to brain by one or more of the 5 sense:
  - Sight , Hearing, Touch, Smell, Taste

Human Learning

- Learning occurs more rapidly when information is received from one or more senses.
- Learning is a key process in human behaviour.



Human Learning

Nature and Characteristics of Learning

- Learning is the change in behaviour.
- Change in behaviour caused by learning is relatively permanent.
- Learning is a continuous life long process.
- Learning is a universal process.
- Learning is purposive and goal directive.
- Learning involves reconstruction of experiences.
- Learning is the product of activity and environment.
- Learning is transferable from one situation to another.

Types of Learning

- **Motor learning**
  - Learning from day to day activities
  - Walking, running, skating, driving, climbing, etc.
- **Verbal learning**
  - Learning involves the language we speak, the communication devices we use.
  - Signs, pictures, symbols, words, figures, sounds, etc, are the tools used in such activities.
- **Concept learning**
  - Learning which requires higher order mental processes like thinking, reasoning, intelligence, etc. we learn different concepts from childhood.
  - Concept learning involves two processes, viz. abstraction and generalisation

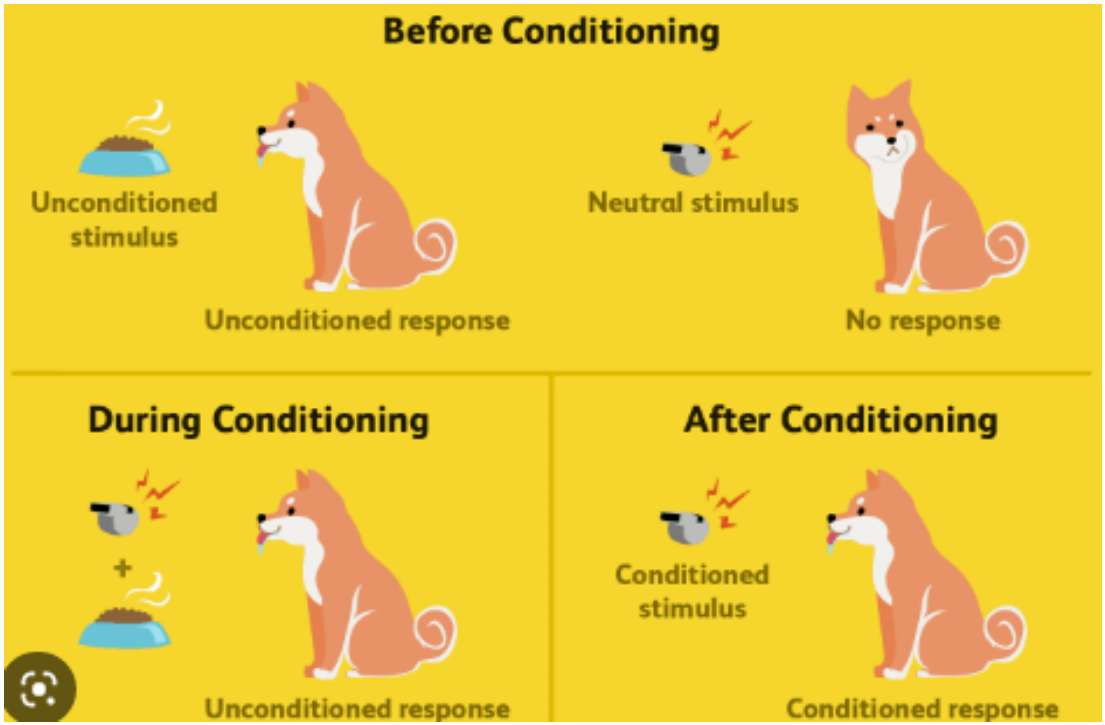
Types of Learning

- **Discrimination learning**
  - Learning to differentiate between stimuli and showing an appropriate response to these stimuli is called discrimination learning. Example, sound horns of different vehicles like bus, car, ambulance, etc.
- **Learning of principles**
  - Individuals learn certain principles related to science, mathematics, grammar, etc. in order to manage their work effectively.
  - Example: formulae, laws, associations, correlations, etc

Learning

Learning through association - Classical Conditioning

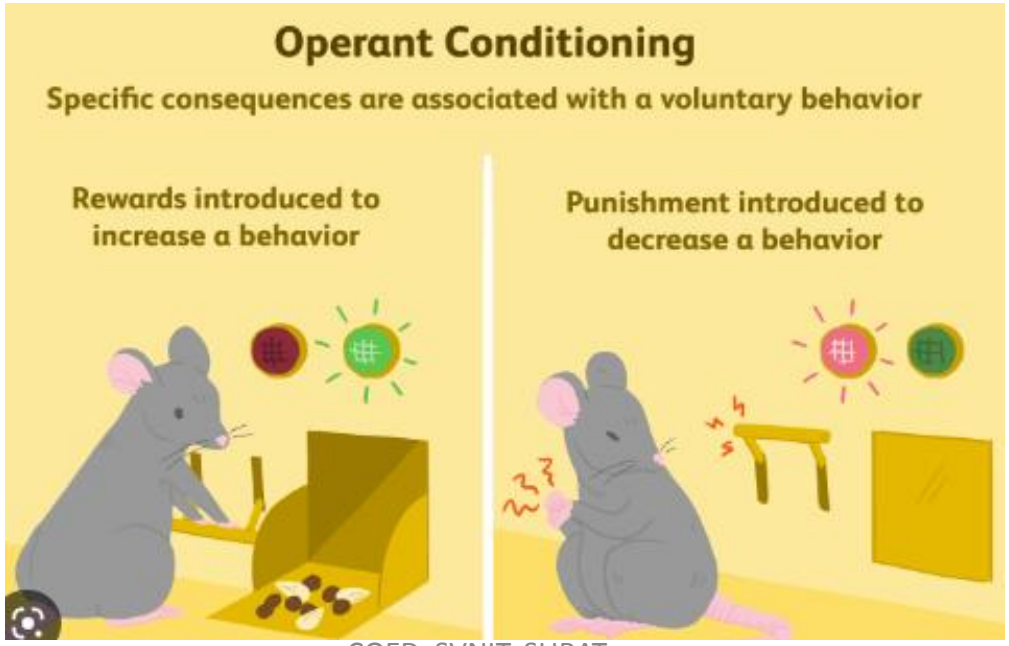
If a neutral stimulus (a stimulus that at first elicits no response) is paired with a stimulus that already evokes a reflex response, then eventually the new stimulus will by itself evoke a similar response.



Learning

Learning through consequences – Operant Conditioning

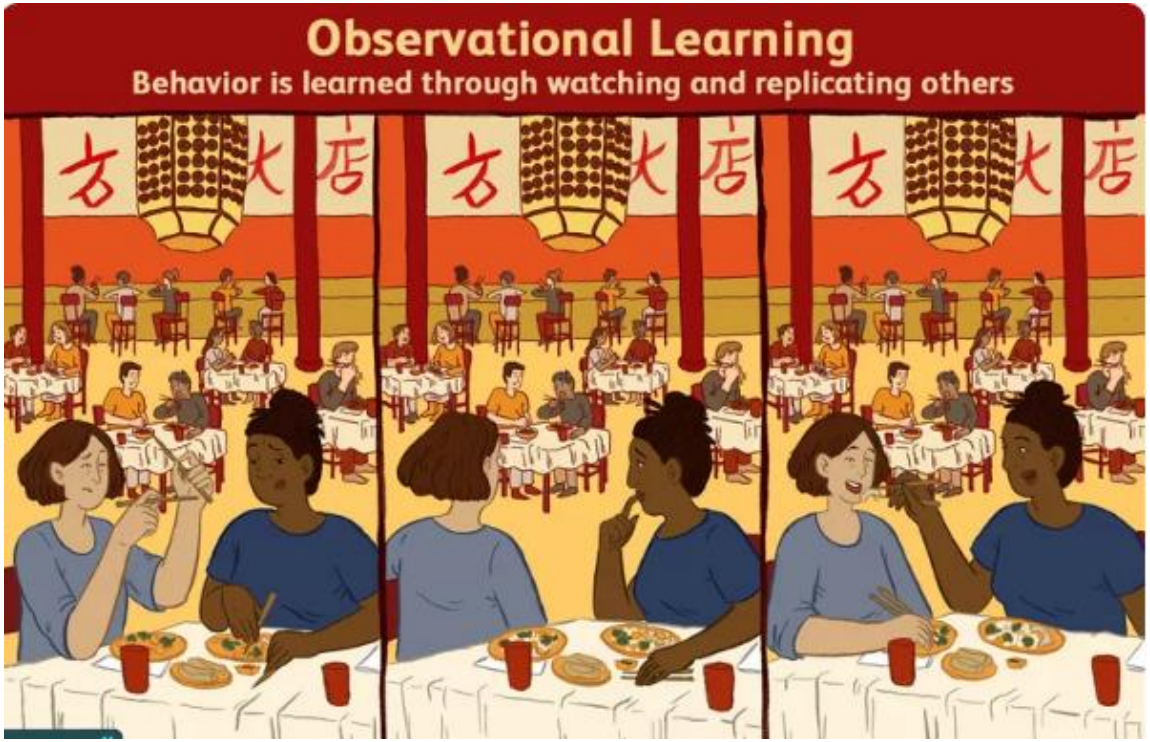
- The organism operates on its environment in some way; the behavior in which it engages are instrumental to achieving some outcome.



Learning

Learning through observation – Modeling/Observational Learning

- The process of learning by watching others, retaining the information, and then later replicating the behaviors that were observed.



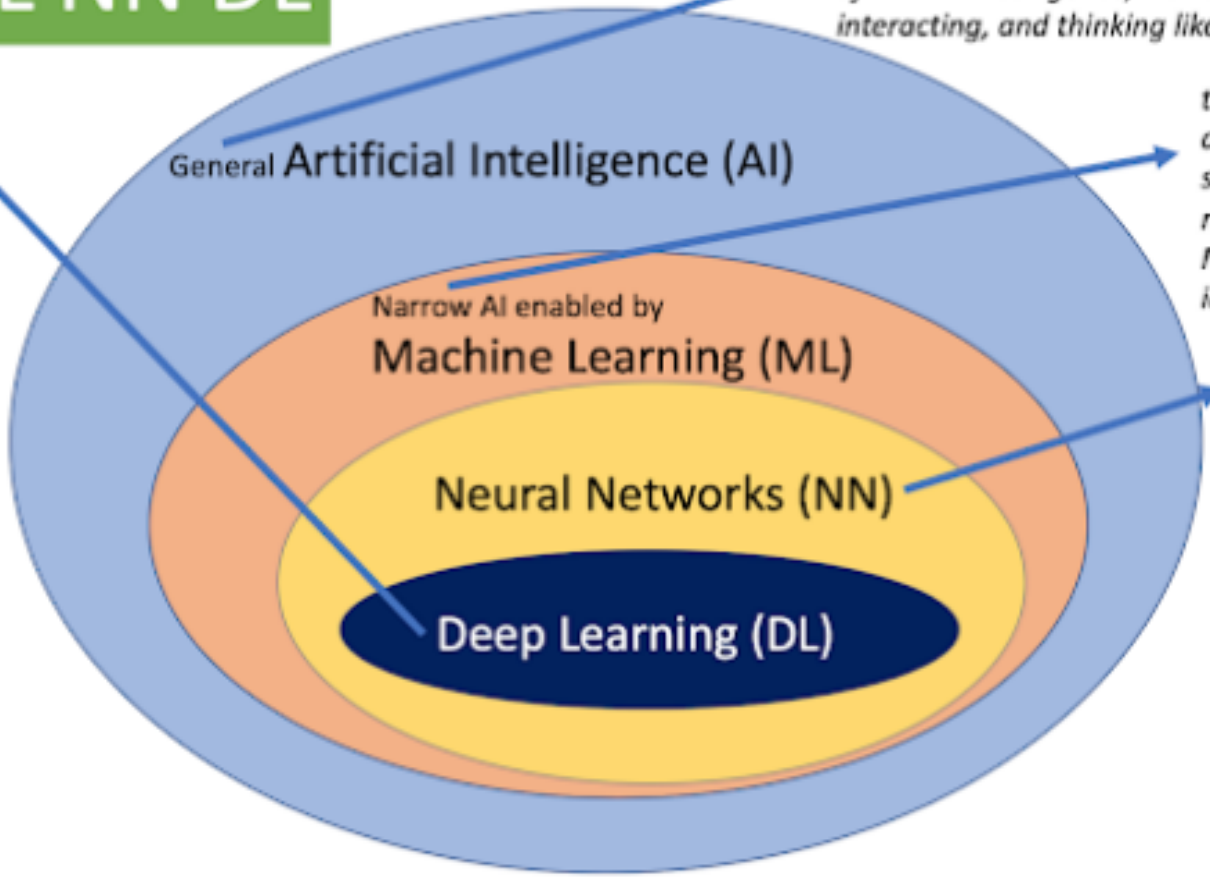
Humans Versus Computer

- Humans and computers are inherently suited to different types of tasks.
- Eg: Task 1- computing the cube root of a large number is very easy for a computer, but it is extremely difficult for humans.
- Task 2- Recognizing the objects in an image is a simple matter for a human, but has traditionally been very difficult for an automated learning algorithm.

Deep Learning

- Deep Learning is **an approach to learning where we can make a machine imitate the network of neurons in a human brain**
- Deep learning networks **learn** by **discovering intricate structures in the data they experience.** By building computational models that are composed of multiple processing layers, the networks can create multiple levels of abstraction to represent the data.
- In recent years that deep learning has shown an accuracy on some of the tasks that exceeds that of a human.

AI ML NN DL



computers possessing the same characteristics of human intelligence, including reasoning, interacting, and thinking like we do

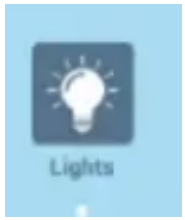
technologies that can accomplish specific tasks such as playing chess, recommending your next Netflix TV show, and identifying spam emails

neural networks are a specific group of algorithms used for machine learning that model data using graphs of Artificial Neurons. Those neurons are a mathematical model that "mimics approximately how a neuron in the brain works"

the word "deep" comes from the fact that DL algorithms are trained/run on deep neu ral networks. These are just neural networks with (usually) three or more "hidden" layers

ML	NN
Feature based	Example Based
Conscious Learning	Sub-conscious Learning
Eg: Fruit -> Shape, colour, texture	Eg: Different Pictures of fruits
Algorithms to parse data, learn from that data, and make informed decisions based on what it has learned	Structures algorithms in layers to create an “artificial neural network” that can learn and make intelligent decisions on its own

Application



ML

Dark

Switch On Lights

Input

Output

NN

Can't see

The light switch won't work

....

Switch on the Lights

Input

Output

Neural Network

- A neural network is a **massively parallel distributed processor** made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:
  - Knowledge is acquired by the network from its environment through a **learning process**
  - Interneuron connection strengths, known as **synaptic weights**, are used to store the acquired knowledge

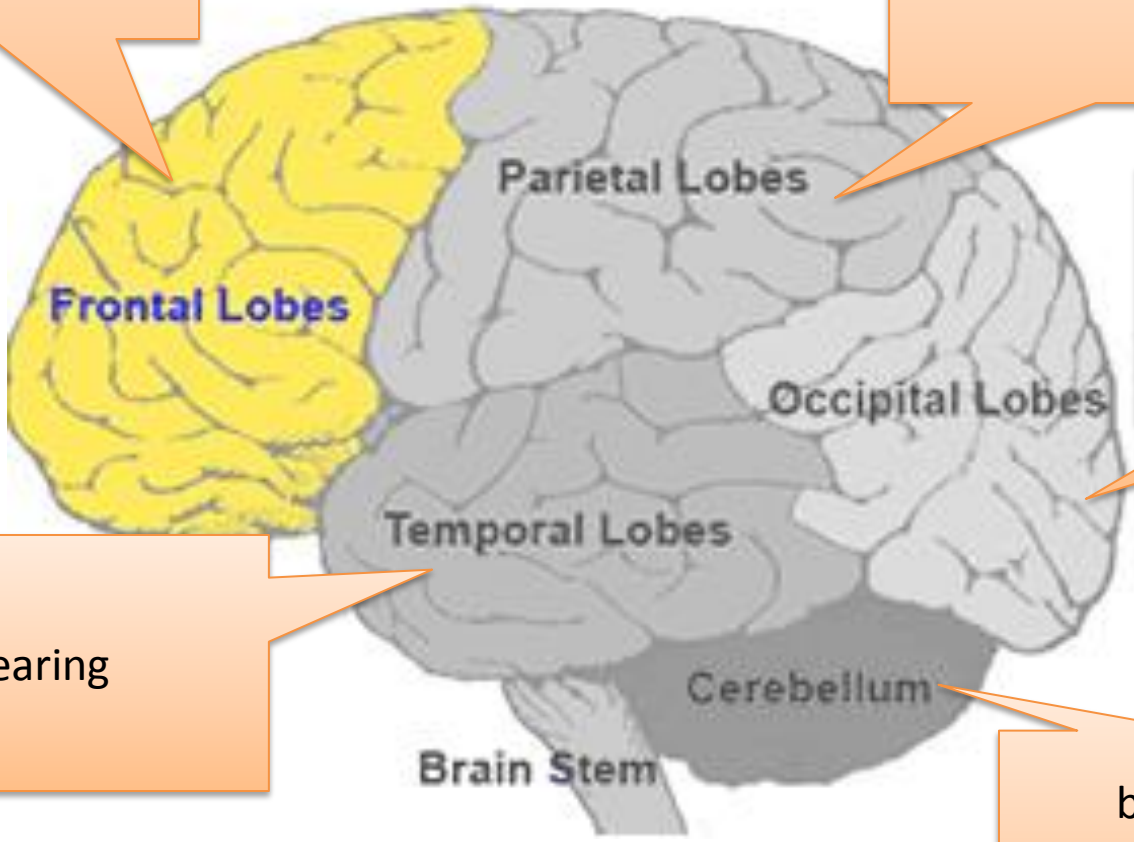
Neural Network

- *Neural network* is a machine that is designed to ***model*** the way in which the **brain performs a particular task or function of interest;** the network is usually implemented by using electronic components or is simulated in software on a digital computer.
- To achieve **good performance, neural networks employ a massive interconnection of simple computing cells referred to as "neurons" or "processing units."**
- The **procedure** used to perform the learning process is called a ***learning algorithm*****, the function of which is to modify the synaptic weights** of the network in an orderly fashion to attain a **desired design objective**.

Human Brain

For Head & eye movements. Also For Emotion, behaviour, speech

Basic Movements like touch



Vision

Hearing

balance and equilibrium and muscle tone.

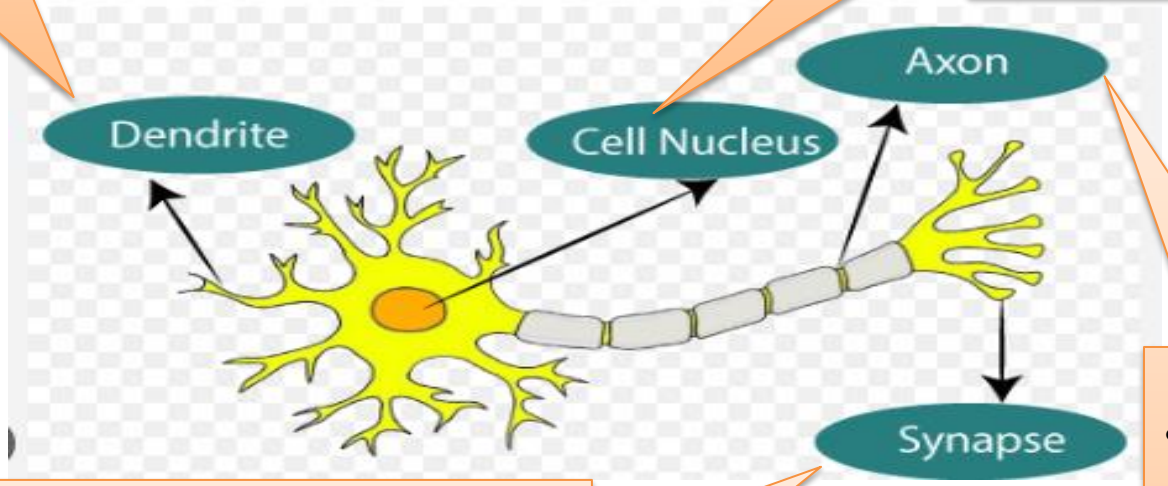
Biological Neuron

- Basic Structural as well as computational unit of brain.
- Able to receive, process, transmit information in form of signals.

Biological Neuron

Receive signals from neighbouring neuron and carry them towards cell body

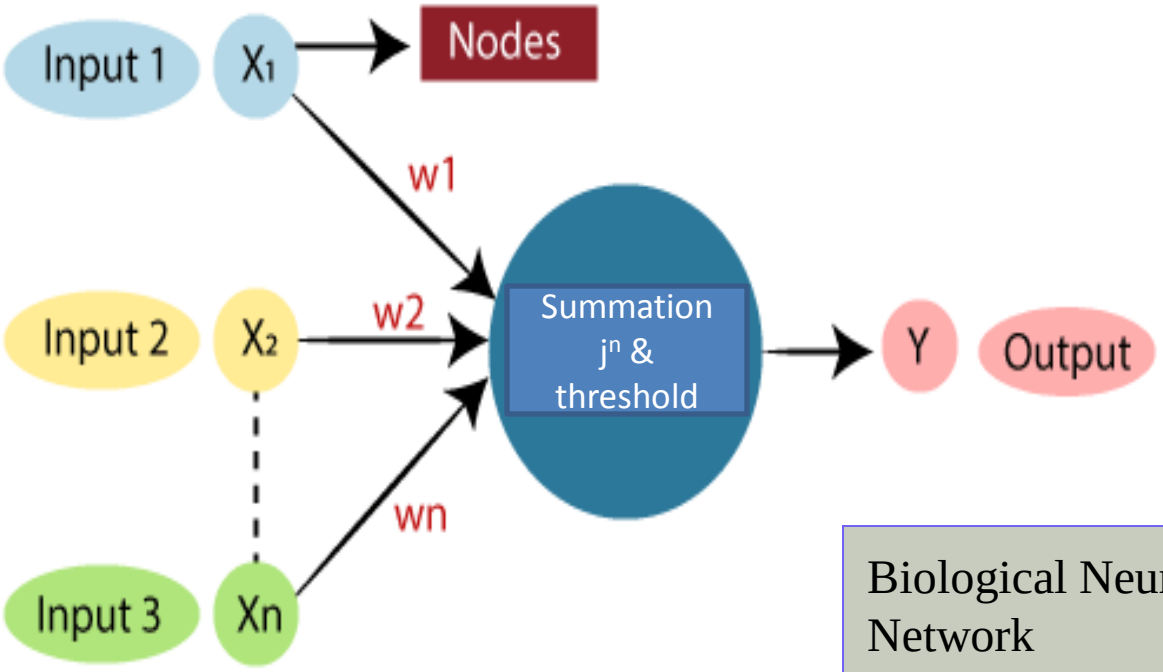
- Soma: Receive/ Accumulate signals from diff. dendrites.
- Fires-> when amt. of signal crosses threshold , sending a spike along its axon.



- Small Gap between that one neuron dendrite & adj. dendrites of neighbouring neuron.
- They are places where neurons connect and communicate with each other

- The output of a neuron it transmits the signal to other neurons

ANN



Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Synapse	Weights
Axon	Output

Applications

Application

- Consider a real estate problem, We need to predict whether a house (Flat) will get buyer or not. Need to make an ANN Model for same.

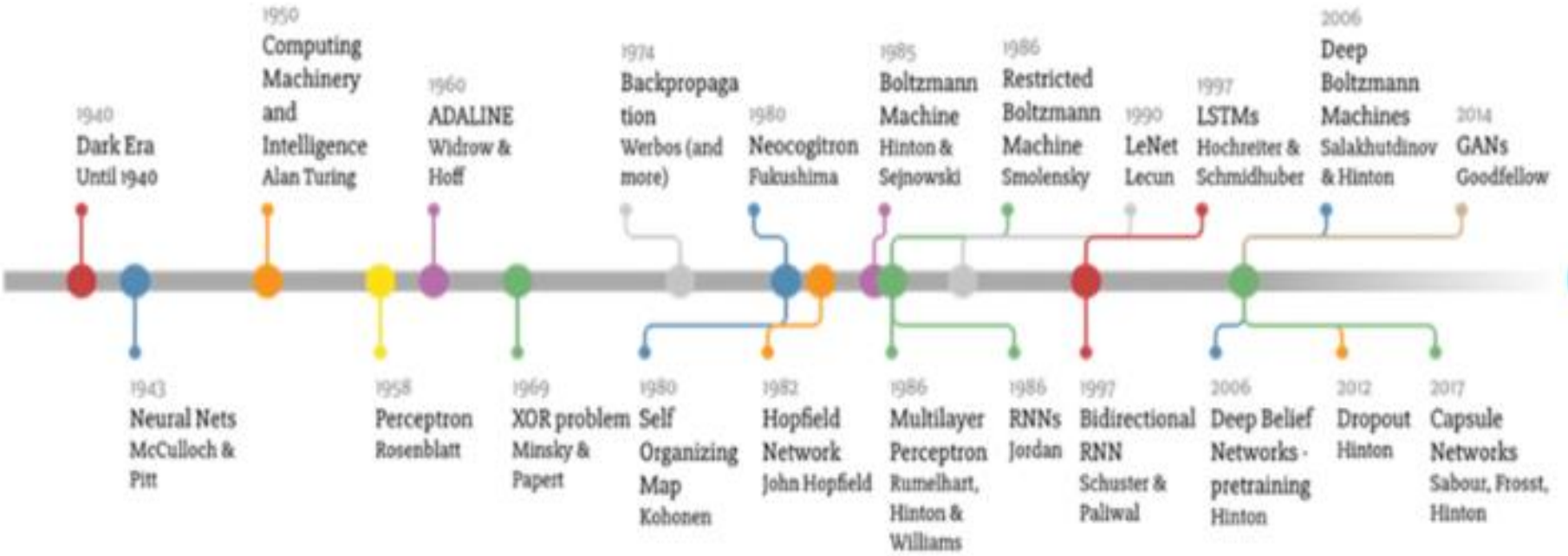
- Inputs: x_1, x_2 (Factors)
- Output: 0-> Will not get Buyer, 1-> Will get a buyer
- Goal: To learn Model f(x,w): where w= weight of vector, the value of which depends on relative influence of x_1, x_2

We need to find optimum value of weights

Human Learning References

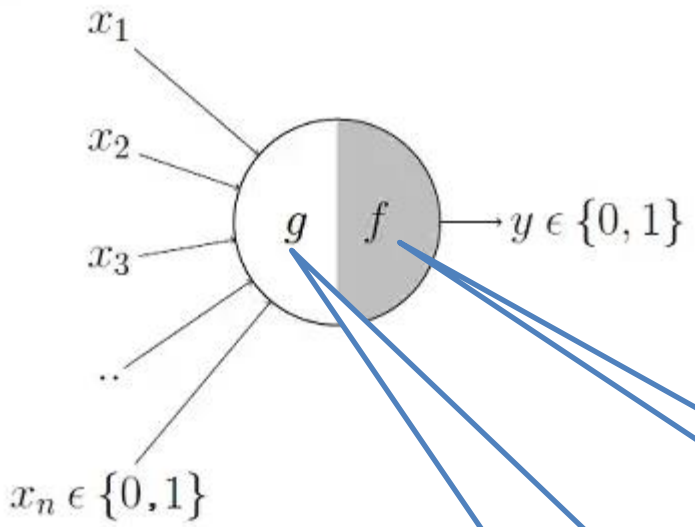
- https://specialeducationnotes.co.in/A3unit1.htm

Deep Learning Timeline



MuCulloch & Pitts Neuron

- The first computational model of a neuron was proposed by Warren MuCulloch (neuroscientist) and Walter Pitts (logician) in 1943.
- Idea of neural networks as **computing machines**.



$$g(x_1, x_2, x_3, ..., x_n) = g(\mathbf{x}) = \sum_{i=1}^{n} x_i$$

$$y = f(g(\mathbf{x})) = 1 \quad if \quad g(\mathbf{x}) \geq \theta$$
$$= 0 \quad if \quad g(\mathbf{x}) < \theta$$

g: Summation/Aggregation Function

f: Threshold -> for making decisions

MuCulloch & Pitts Neuron

- ***Excitatory*** *Signals are those that prompt* ***one neuron to share information with the next through an action potential****, while* ***inhibitory*** *signals* ***reduce*** *the probability that such a transfer will take place.*
- Inputs in Neuron can either be ***excitatory*** **or** ***inhibitory***.

Example

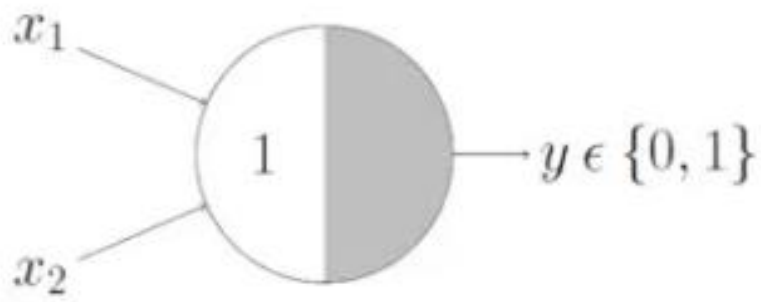
- Given the following weather situations, we want to predict John will carry Umbrella or Not using MP neuron Model. (Fact: John carries umbrella if it is sunny or raining)
  - Situation 1: It is not raining, nor it is sunny.
  - Situation 2: It is not raining, but it is sunny.
  - Situation 3: It is raining and it is not sunny.
  - Situation 4: It is raining as well as sunny.

Example

- Let weights and threshold be one.
- Raining and sunny be inputs
- Raining -> x_1
- Sunny -> x_2

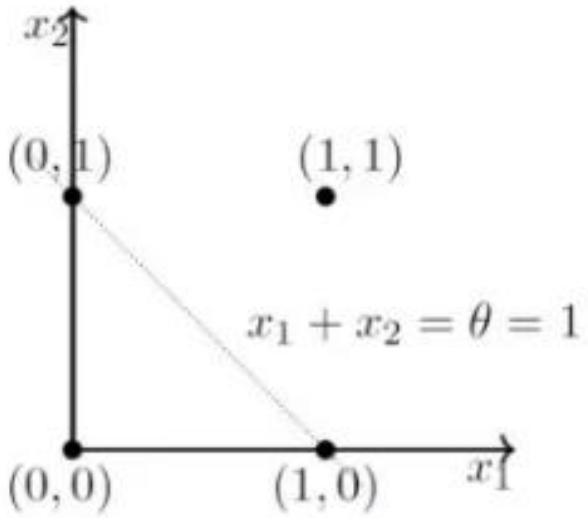
Situation	X_1	X_2	Y_{sum}	Y_{out}
1	0	0	0	0
2	0	1	1	1
3	1	0	1	1
4	1	1	2	1

Boolean Functions Using MP Neuron



OR function

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

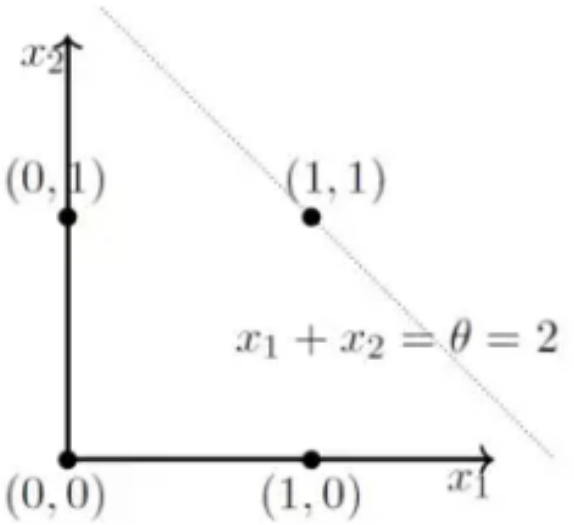


Boolean Functions Using MP Neuron

AND Function

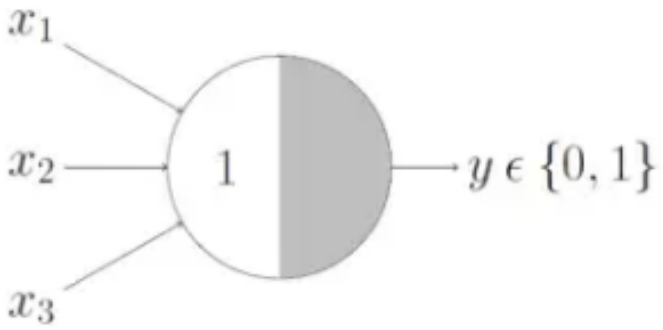
AND function

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



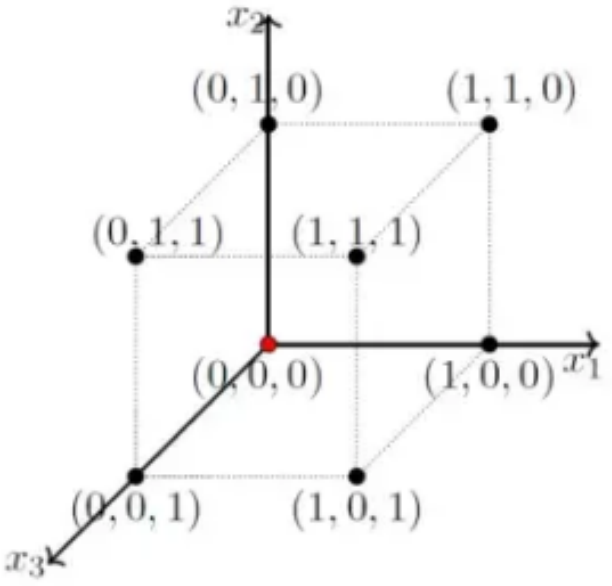
Boolean Functions Using MP Neuron

OR Function With 3 Inputs



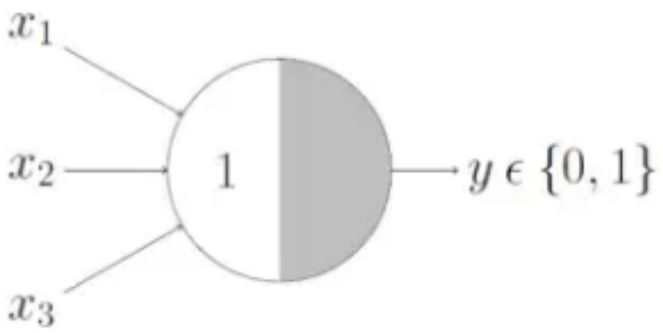
OR function

$$x_1 + x_2 + x_3 = \sum_{i=1}^{3} x_i \geq 1$$



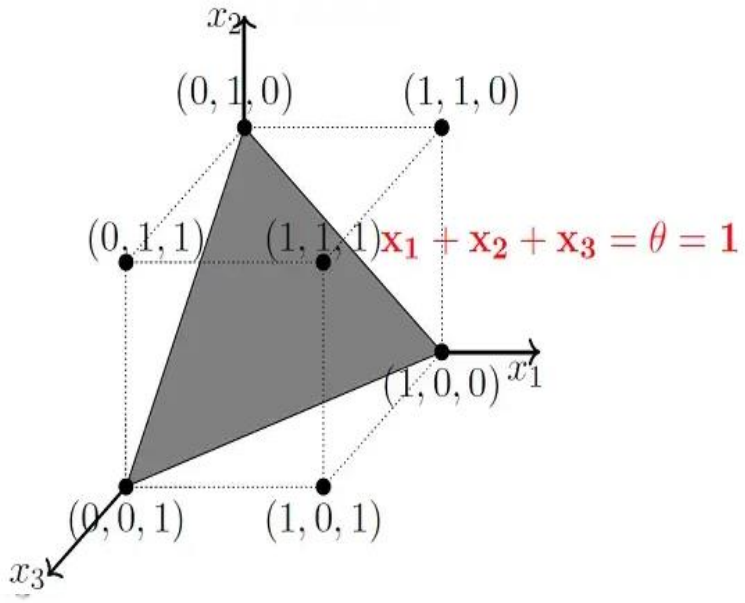
Boolean Functions Using MP Neuron

OR Function With 3 Inputs



OR function

$$x_1 + x_2 + x_3 = \sum_{i=1}^{3} x_i \geq 1$$



MuCulloch & Pitts Neuron Model

- Boolean Inputs
  - i.e. $X_1, X_2, \ldots X_n$ are always 0/1
- Inputs are not Weighted
  - i.e. Consider $w_1, w_2, \ldots w_n$ always as 1
- MP Neuron Model works linearly separable data.
- Model can represent Boolean function which are linearly separable, where a line/plane can be used to separate data.
- However, we can adjust threshold input to make the model fit our the dataset

Perceptron Learning Algorithm

$P \leftarrow$ *inputs with label* 1;
$N \leftarrow$ *inputs with label* 0;
Initialize $\mathbf{w} = [w_1, w_2, \ldots, w_N]$ randomly;
while !*convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P$ *and* $\sum_{i=0}^{n} w_i * x_i < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N$ *and* $\sum_{i=0}^{n} w_i * x_i \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

- Consider two vectors $\mathbf{w}$ and $\mathbf{x}$

$$\mathbf{w} = [w_0, w_1, w_2, ..., w_n]$$
$$\mathbf{x} = [1, x_1, x_2, ..., x_n]$$
$$\mathbf{w} \cdot \mathbf{x} = \mathbf{w}^{\mathbf{T}}\mathbf{x} = \sum_{i=0}^{n} w_i * x_i$$

- We can thus rewrite the perceptron rule as

$$y = 1 \quad if \quad \mathbf{w}^{\mathbf{T}}\mathbf{x} \geq 0$$
$$= 0 \quad if \quad \mathbf{w}^{\mathbf{T}}\mathbf{x} < 0$$

Perceptron Learning Algorithm

Proposition: If the sets P and N are finite and linearly separable, the perceptron learning algorithm updates the weight vector $\mathbf{w}_t$ a finite number of times. In other words: if the vectors in P and N are tested cyclically one after the other, a weight vector $\mathbf{w}_t$ is found after a finite number of steps t which can separate the two sets.

Perceptron Learning Algorithm

$P \leftarrow inputs \quad with \quad label \quad 1;$
$N \leftarrow inputs \quad with \quad label \quad 0;$
Initialize $\mathbf{w} = [w_1, w_2, \ldots, w_N]$ randomly;
while !*convergence* **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P \quad and \quad \sum_{i=0}^{n} w_i * x_i \leq 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N \quad and \quad \sum_{i=0}^{n} w_i * x > 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

$if\ f(x) > 0 \ \in P$
$if\ f(x) \leq 0 \ \in N$

X_1	X_2	Output
0	0	0
0	1	0
1	0	1
1	1	0

Perceptron Learning Algorithm

Implementation of following truth table using Perceptron.

X_1	X_2	Output
0	0	0
0	1	0
1	0	1
1	1	0

P $class_1$ i.e. 1 ; N $class_2$ i.e. 0
Let w=[0,-2]
Let **x** = (0,0)
Here, X belongs to N (check 2^{nd} condition for Algo.)
$x_1w_1 + x_2w_2 = 0+0= 0 > 0$ -> **Doesn't Satisfies**
End

Let w=[0,-2]
Let **x** = (0,1)
Here, X belongs to N (check 2^{nd} condition for Algo.)
$x_1w_1 + x_2w_2 = 0+ (-2)= -2 > 0$ -> **Doesn't Satisfies**
End

Let w=[0,-2]
Let **x** = (1,0)
Here, X belongs to P (check 1^{nd} condition for Algo.)
$x_1w_1 + x_2w_2 = 0+ 0= 0 \leq 0$ -> **Satisfies**
So w= w+x
[0,-2] + [1,0] = [1,-2] (Updated w)

Perceptron Learning Algorithm

Implementation of following truth table using Perceptron.

X_1	X_2	Output
0	0	0
0	1	0
1	0	1
1	1	0

P $class_1$ i.e. 1 ; N $class_2$ i.e. 0

Let w=[1,-2]

Let **x** = (0,0)

Here, X belongs to N (check 2^{nd} condition for Algo.)

$x_1w_1 + x_2w_2 = 0+0= 0 > 0$ -> **Doesn't Satisfies**

End

Let w=[1,-2]

Let **x** = (0,1)

Here, X belongs to N (check 2^{nd} condition for Algo.)

$x_1w_1 + x_2w_2 = 0+ (-2)= -2 > 0$ -> **Doesn't Satisfies**

End

Let w=[1,-2]

Let **x** = (1,0)

Here, X belongs to P (check 1^{nd} condition for Algo.)

$x_1w_1 + x_2w_2 = 1+ 0= 1 <= 0$ -> **Doesn't Satisfies**

End

Perceptron Learning Algorithm

Implementation of following truth table using Perceptron.

X_1	X_2	Output
0	0	0
0	1	0
1	0	1
1	1	0

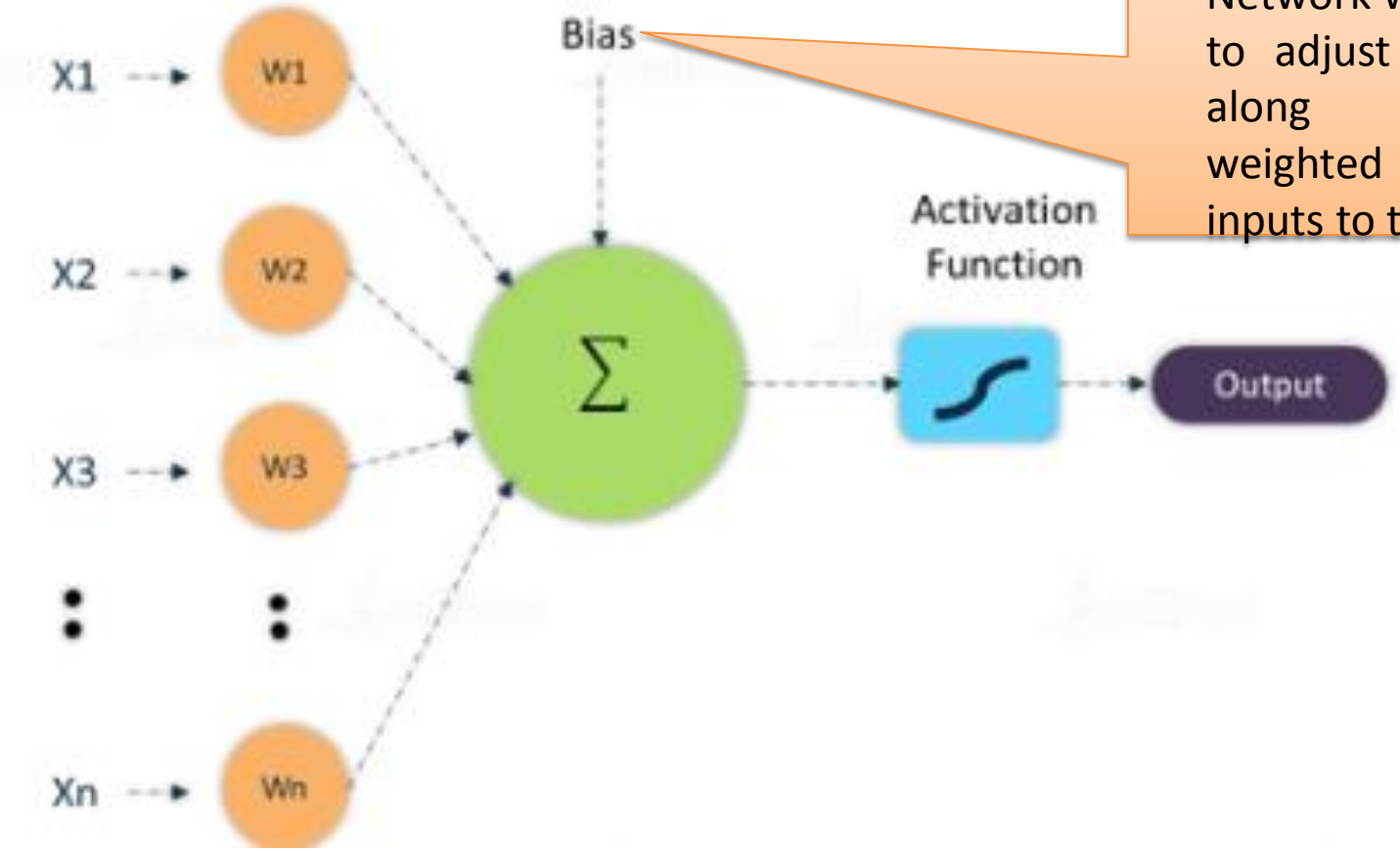
P $class_1$ i.e. 1 ; N $class_2$ i.e. 0
Let w=[1,-2]
Let **x** = (1,1)
Here, X belongs to N (check 2^{nd} condition for Algo.)
$x_1w_1 + x_2w_2$ = 1+-2= -1 > 0 -> **Doesn't Satisfies**
End

[1,-2] Final Weights

Rosenblatt's Perceptron

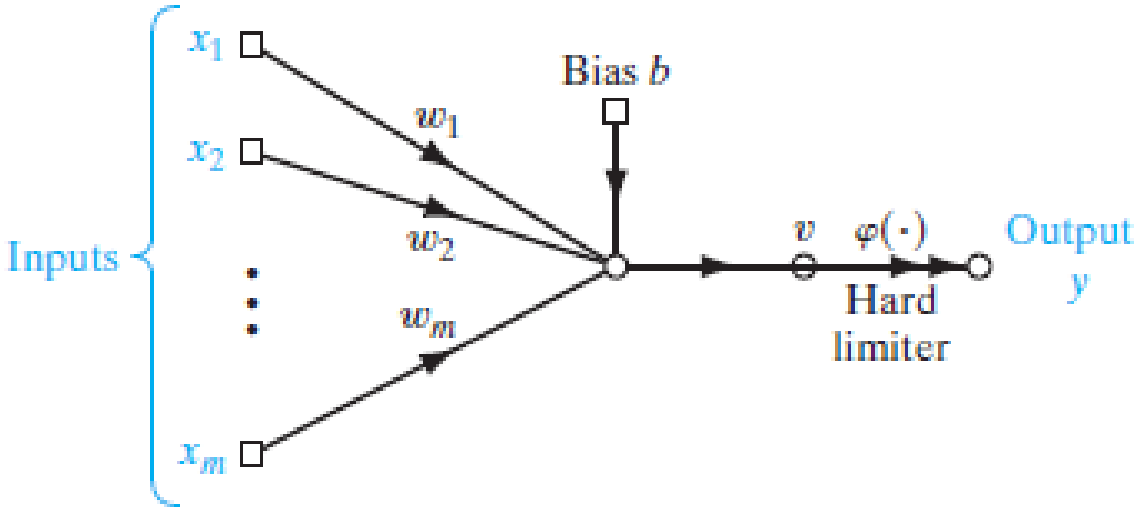
- Rosenblatt (1958) proposed perceptron as the first model supervised learning.
- The perceptron is the **simplest form** of a neural network used for the classification of patterns said to be *linearly separable.*
- It consists of a single neuron with **adjustable synaptic weights and bias.**
- Rosenblatt proved that if the patterns (vectors) used to train the perceptron are drawn from two linearly separable classes, then the perceptron algorithm converges and positions the decision surface in the form of a hyperplane between the two classes.

ANN



- It is an additional parameter in the Neural Network which is used to adjust the output along with the weighted sum of the inputs to the neuron

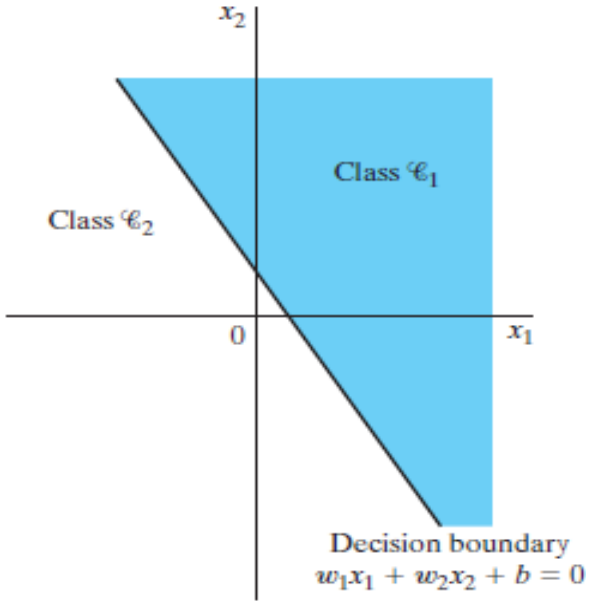
Rosenblatt's Perceptron



$$v = \sum_{i=1}^{m} w_i x_i + b$$

- The summing node of the neural model computes a linear combination of the inputs applied to its synapses, as well as incorporates an externally applied bias.
- The resulting sum, that is, the induced local field, is applied to a hard limiter.
- The neuron produces an output equal to 1 if the hard limiter input is positive, and -1 if it is negative.

Rosenblatt's Perceptron



- Fig shows decision boundary for a two-dimensional, two-class pattern-classification problem.
- The synaptic weights $w_1, w_2, \ldots, w_n$ of the perceptron can be adapted on an iteration by- iteration basis. For the adaptation, we may use an error-correction rule known as the perceptron convergence algorithm

Rosenblatt's Perceptron Model

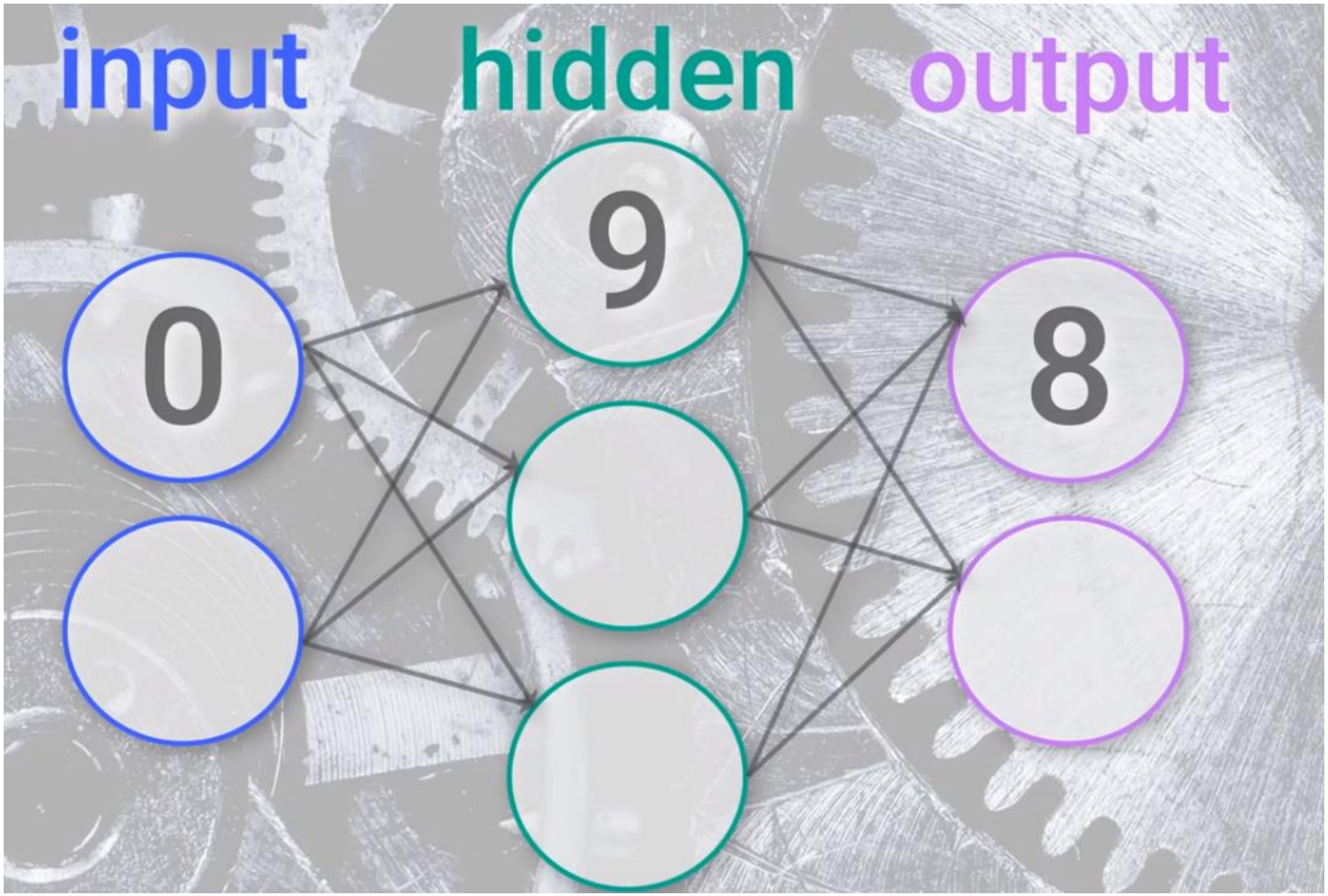
- Work on linearly separable data
- Numeric Weights (Weighted Input-i.e. All inputs are not equal)
- Real inputs (Not restricted to Boolean)

Bias

- Neural network bias can be defined as **the constant which is added to the product of features and weights**.
- It is used to offset the result. It helps the models to shift the activation function towards the positive or negative side
- It increases the flexibility of model
- Bias are learnable parameter.

Learnable Parameter

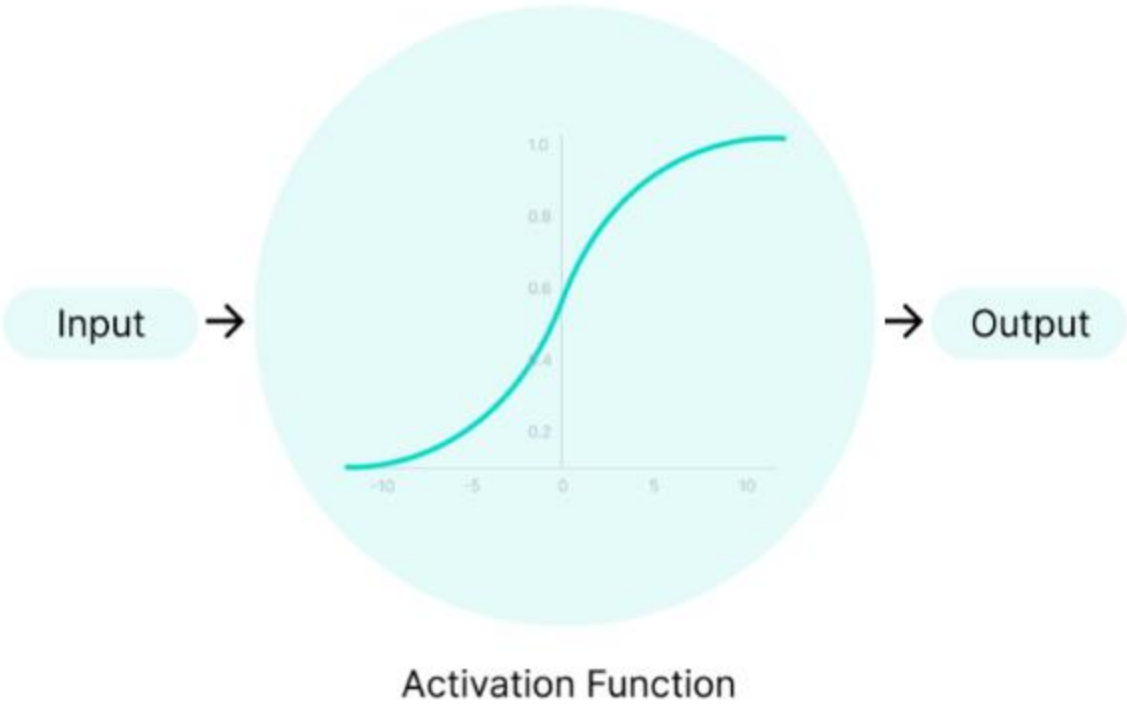
- Learnable parameter- a parameter which is learnt by network during training
- Weights and bias in each layer in NN
- **Each Layer : input* output +bias**


input
hidden
output
0
9
8

Activation Function

- **An Activation Function/Threshold function** decides whether a neuron should **be activated or not.**
- It will decide whether the neuron's input to the network is **important or not** in the process of prediction using simpler mathematical operations.
- **Activation Function** helps the neural network to use important information while **suppressing irrelevant data points.**
- The role of the Activation Function **is to derive output from a set of input values fed to a node** (or a layer)

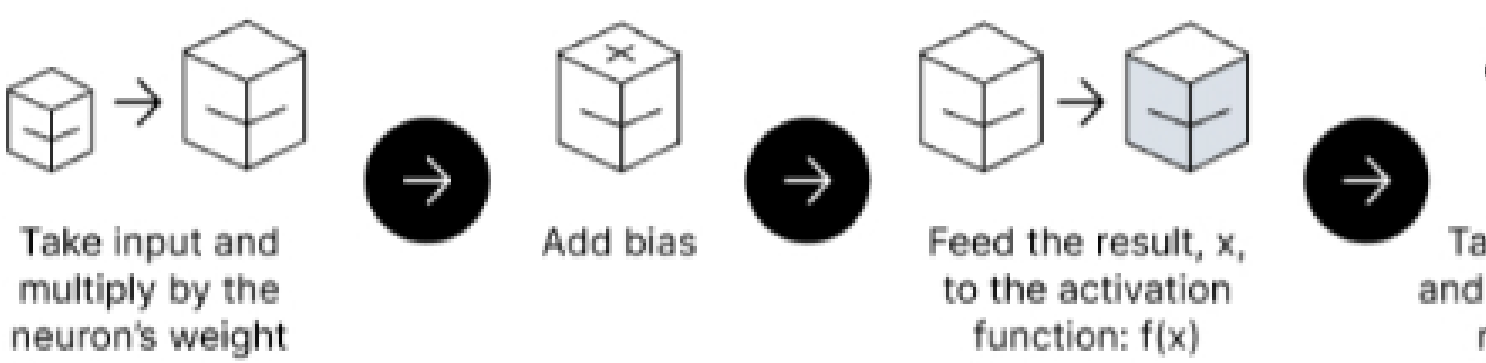
Activation Function



Activation Function

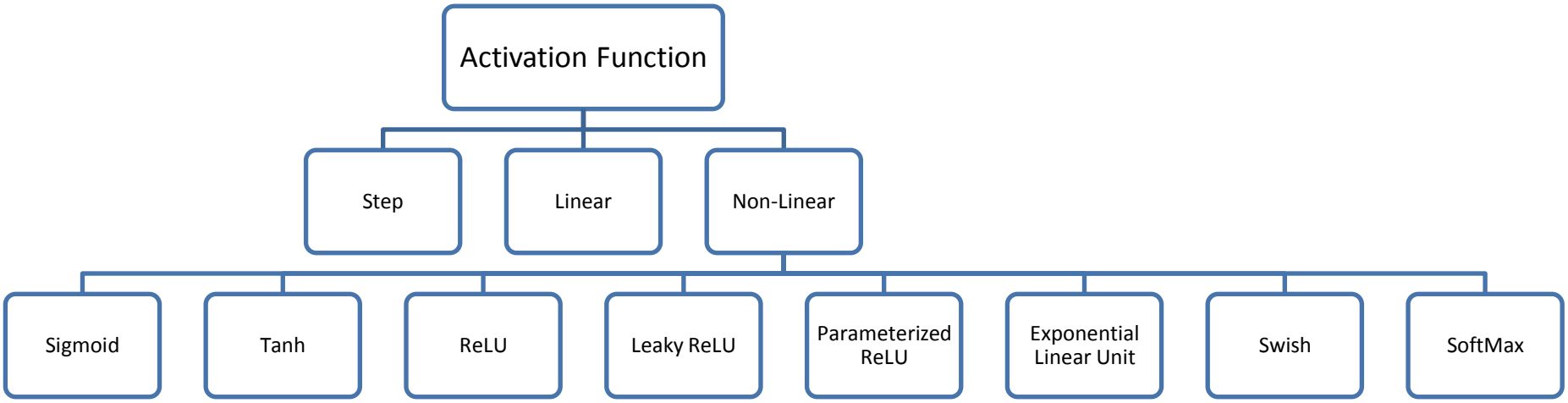
Activation Function

- In the feedforward propagation, the Activation Function is a mathematical “gate” in between the input feeding the current neuron and its output going to the next layer.



Activation Function

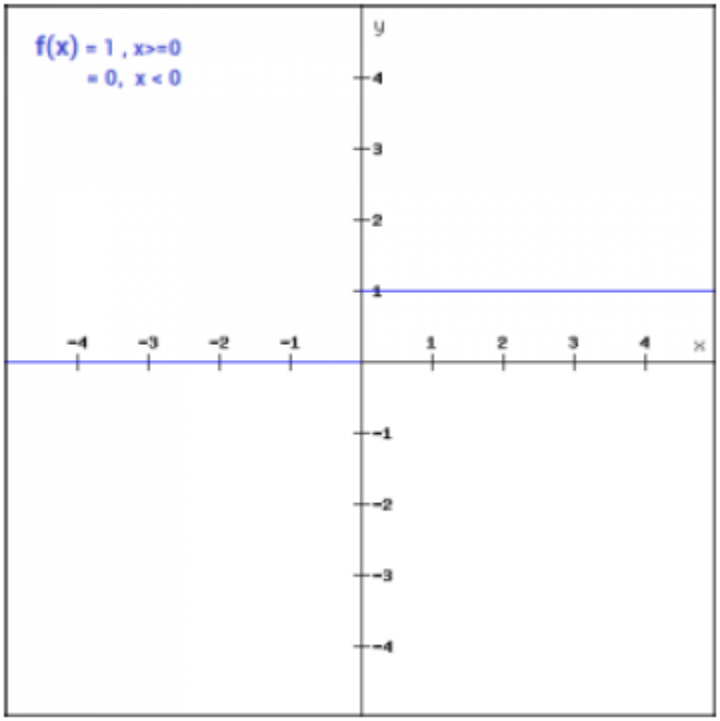
- Activation functions introduce an additional step at each layer during the **forward propagation**,
- Necessity :
  - If not used, Every neuron will only be performing a **linear transformation** on the inputs using the weights and biases.
  - Hence, our model would be just a **linear regression model.**
  - Used to enable a **limited amplitude** of the output of a neuron and enabling it in a **limited range** is known as **squashing functions. (**A squashing function squashes the amplitude of output signal into a finite value)


Activation Function
Step
Linear
Non-Linear
Sigmoid
Tanh
ReLU
Leaky ReLU
Parameterized ReLU
Exponential Linear Unit
Swish
SoftMax

Activation Function

Binary Step Function

- It depends on a threshold value that decides **whether a neuron should be activated or not**.
- The input fed to the activation function is compared to a certain **threshold**; if the input is greater than it, **then the neuron is activated, else it is deactivated, meaning that its output is not passed on to the next hidden layer.**
- It cannot provide **multi-value outputs—for** example, it cannot be used for multi-class classification problems.



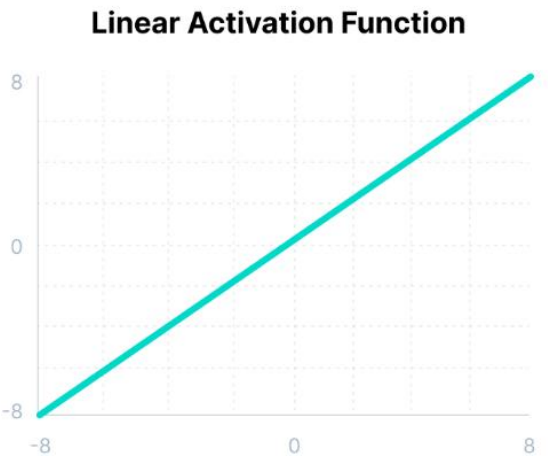
Binary step

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

Activation Function

Linear Activation Function

- It is directly proportional to the input.
- The final layer of the **Neural Network will be working as a linear function of the first layer**.
- **Does not** allow the model to create **complex mappings** between the network's inputs and outputs.
- Here the derivative of the function f(x) is equal to the **value of constant used**.
- There **isn't much benefit of using linear function because the neural network would not improve the error due to the same value of gradient** for every iteration.
- Linear functions are ideal where interpretability is required and for simple tasks.



$f(x) = ax + c$

Activation Function

Non-Linear Activation Functions

- They allow **backpropagation** because the derivative function would be related to the input and it's possible to go back and understand which **weights in the input neurons can provide a better prediction**.
- They allow the **stacking** of **multiple** layers of neurons as the output would now be a non-linear combination of input passed through multiple layers.

Activation Function

Sigmoid / Logistic Activation Function

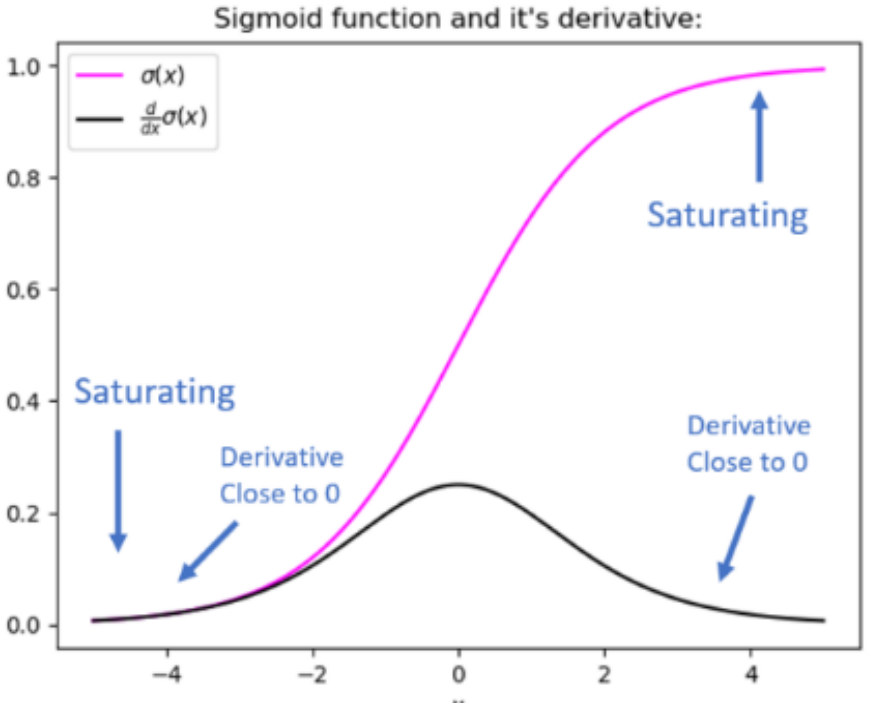
- This function takes any real value as input and outputs values in the range of 0 to 1.
- S-shape
- The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0
- Suitable for Binary Classification

$$\phi(z) = \frac{1}{1 + e^{-z}}$$

Activation Function

Sigmoid / Logistic Activation Function

- Sigmoids saturate and kill gradients.
- (See derivative in figure)Top and bottom level of sigmoid functions the curve changes slowly, if we calculate slope(gradients) it is zero.
- **Hence,** When the x value is small or big the slope is zero->then there is no learning, which leds to Slow convergence (**Vanishing gradients**).
- The output of the logistic 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



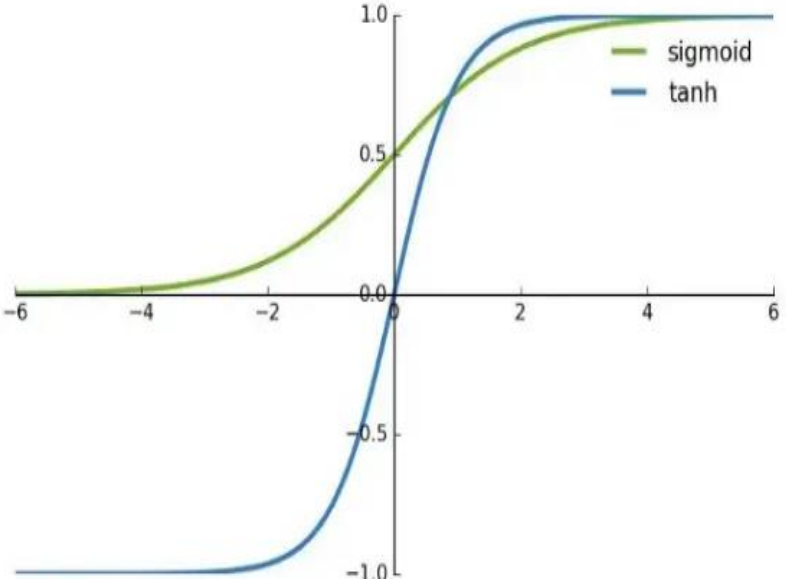
- **Gradients are used during training to update the network weights**
- **Vanishing gradients:** This occurs when the gradient is too small. As we move backwards during backpropagation, **the gradient continues to become smaller, causing the earlier layers in the network to learn more slowly than later layers.** When this happens, the weight parameters update until they become insignificant—i.e. 0—resulting in an algorithm that is no longer learning
- Eg: A large change in the input of the sigmoid function will cause a small change in the output.
- Hence, the derivative becomes small**. For shallow networks with only a few layers that use these activations, this isn't a big problem. However, when more layers are used, it can cause the gradient to be too small for training to work effectively**.

Activation Function

Tanh Activation Function

- Tanh function is very similar to the **sigmoid/logistic activation function,** and even has the **same S-shape with the difference in output range of -1 to 1.**
- In Tanh, the larger the input (more positive), the closer the output value will be to **1.0**, whereas the smaller the input (more negative), the closer the output will be to **-1.0.**
- It is bipolar in nature
- It is continuous activation function

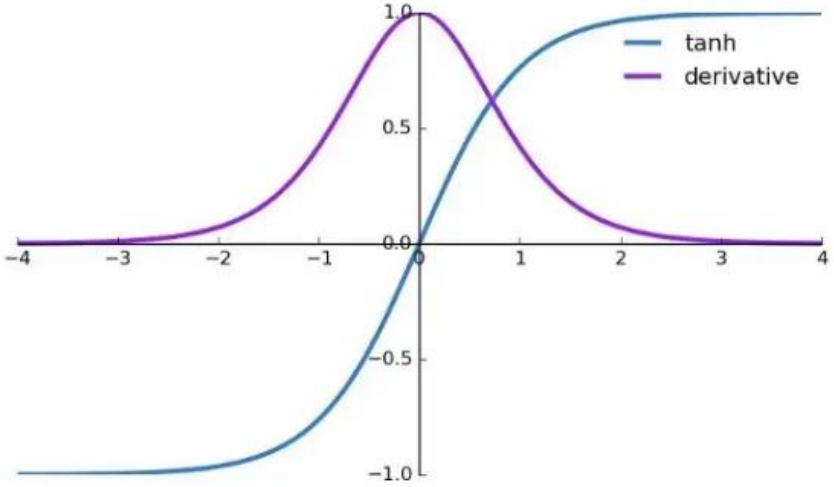
$$f(x) = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)}$$



Activation Function

Tanh Activation Function

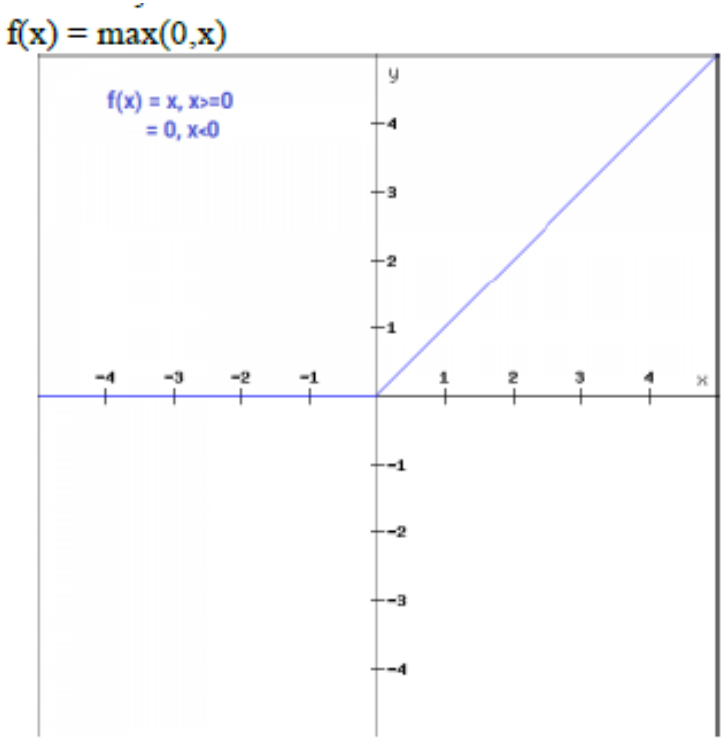
- Here, output is zero centered because its range in between -1 to 1 i.e -1 < output < 1. Hence, preferred more as compared to sigmoid.
- Usually used in **hidden layers** of a neural network as it's values lies between**-1 to 1** hence the mean for the hidden layer comes out be 0 or very close to it, hence helps in *centering the data* by bringing mean close to 0. This makes learning for the next layer much easier.
- As per derivative, it also suffers from **Vanishing gradients**



Activation Function

RELU Activation Function

- ReLU stands for rectified linear unit and is a non-linear activation function which is widely used in neural network in hidden layers.
- ***f(x) = max(0,x)***. It gives an output x if x is positive and 0 otherwise.
- Linear for x greater or equal to 0, non-linear otherwise.
- **Value Range :-** [0, inf)
- Referred as piece–wise linear or hinge function.

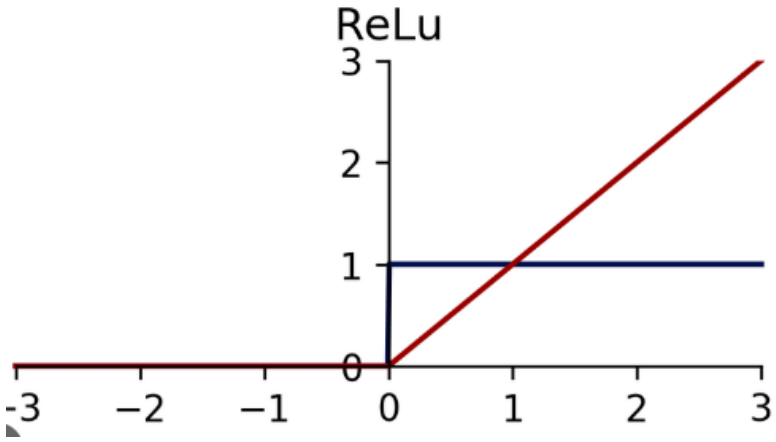


- *Monotonic means something that does not vary or change.*
- *Non-Monotonic means something which can vary according to the situation or condition.*

Activation Function

RELU Activation Function

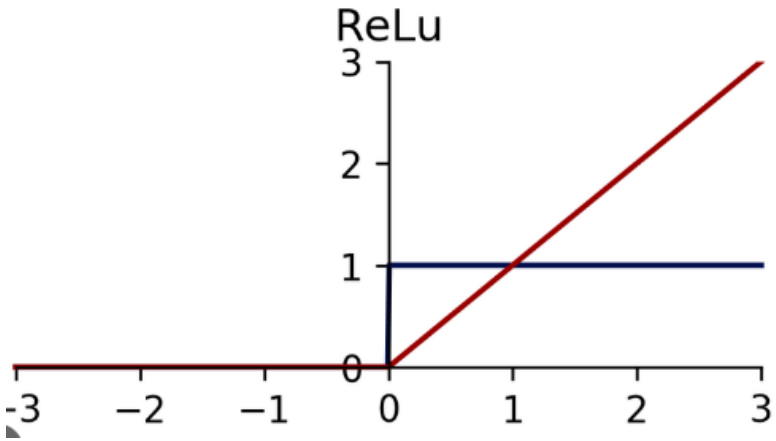
- The function and its derivative (in blue in figure) **both are monotonic.**
- ReLU has a derivative function and allows for **backpropagation** while simultaneously making it computationally efficient. It **converges faster** as compared to sigmoid and tanh.
- It involves simple calculation as compared to sigmoid and tanh, hence **Computation is faster.**



Activation Function

RELU Activation Function

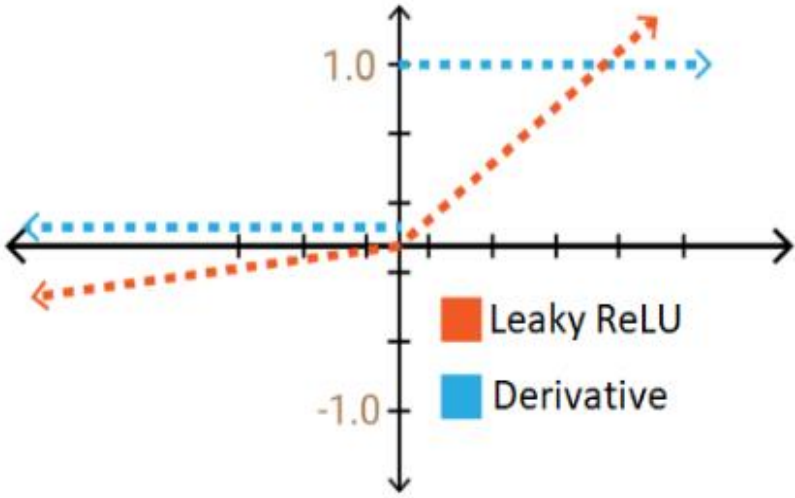
- **Issue**: **(Dying ReLU)** Any negative input given to the ReLU activation function turns the value into **zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately**. Hence, decreases the model's ability to fit or train from the data properly.
- **Moreover,** Some gradients can be fragile during training and can die. It can cause a **weight update which will makes it never activate on any data point again.** Simply saying that ReLu could result in **Dead Neurons.**



Activation Function

Leaky RELU Activation Function

- Leaky ReLU is **an improvised version of ReLU function where for negative values of x,** instead of defining the ReLU functions' value as zero, it is defined as **extremely small linear component of x.**
- **Range** of the Leaky ReLU is (-infinity to infinity)

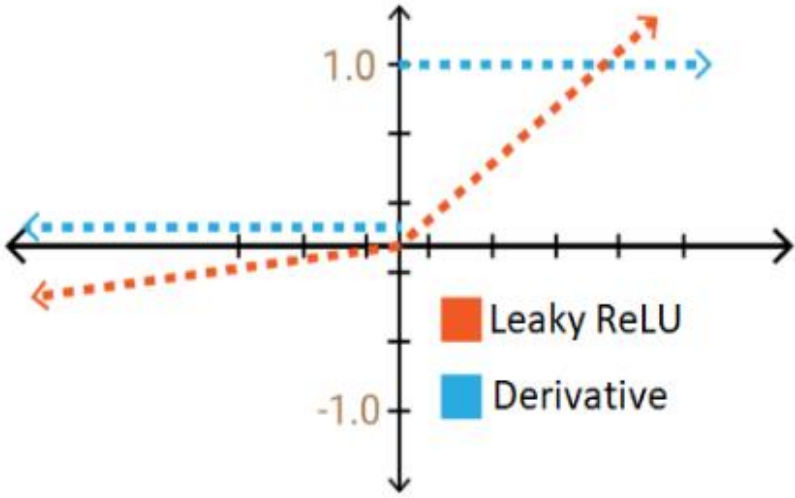


$$f(x) = 0.01x,\ x < 0$$
$$f(x) = x,\ x >= 0$$

Activation Function

Leaky RELU Activation Function

- **It** enables backpropagation, even for negative input values
- The gradient of the left side of the graph comes out to be a non-zero value. Therefore, we would **no longer encounter dead neurons in that region.**
- Limitations :
  - The predictions may not be consistent for negative input values.
  - The gradient for negative values is a small value that makes the learning of model parameters time-consuming



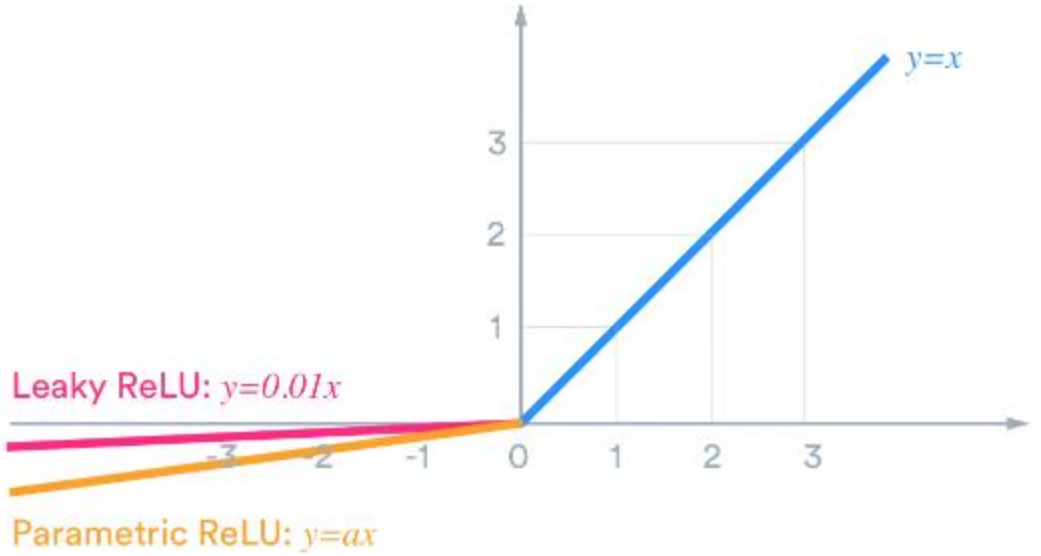
$f^1(x) = g(x) = 1, x>=0$

$= 0.01, x<0$

Activation Function

Parameterized RELU Activation Function

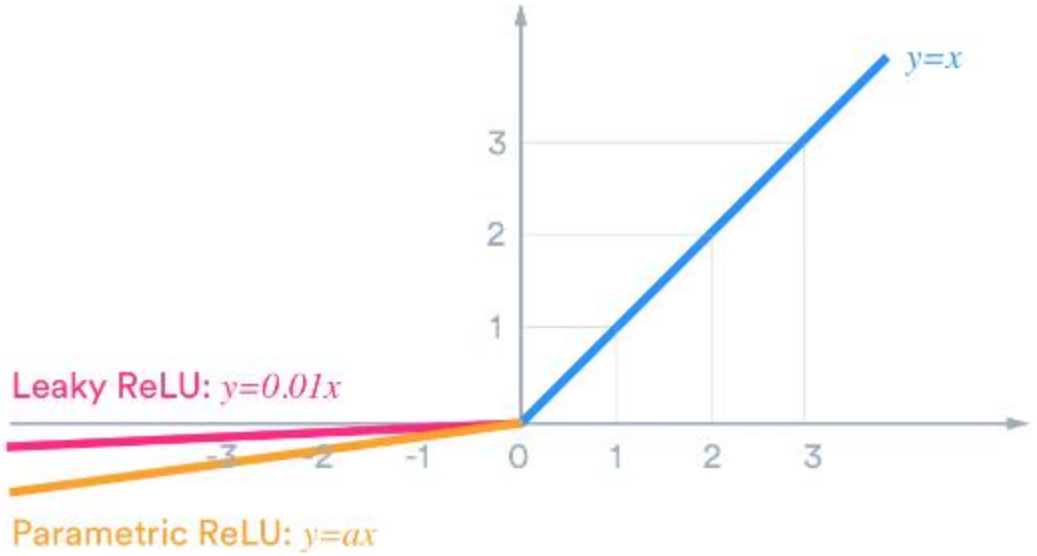
- This function provides the slope of the negative part of the function as an argument ***α***.
- When **a is not 0.01** then it is called **Parameterized ReLU**
- By performing backpropagation, the most appropriate value of *α* is learnt.



Activation Function

Parameterized RELU Activation Function

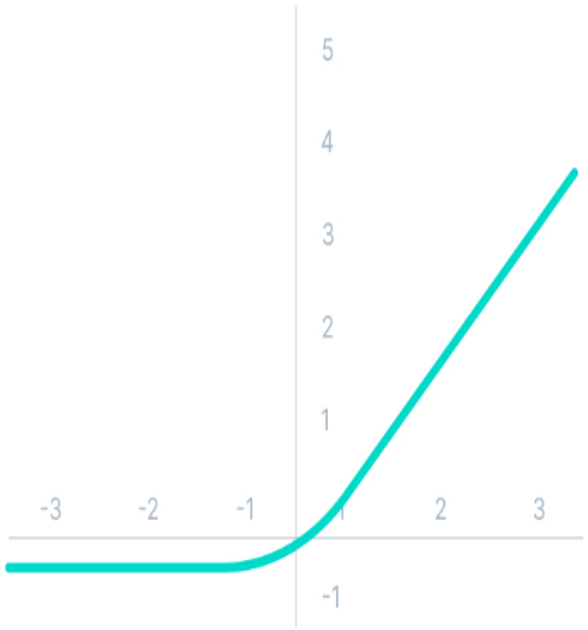
- The parameterized ReLU function is used when the **leaky ReLU function still fails at solving the problem of dead neurons,** and the relevant information is not successfully passed to the next layer.
- This function's limitation is that it may perform differently for different problems depending upon the value of slope parameter ***a***



Activation Function

ELU Activation Function

- Exponential Linear Unit or ELU is also a variant of Rectified Linear Unit. ELU introduces a parameter slope for the negative values of x. It uses a log curve for defining the negative values
- Avoids **dead ReLU problem by introducing log curve for negative values of input**. It helps the network nudge weights and biases in the **right direction.**
- Issue: It increases the computational time because of the exponential operation included



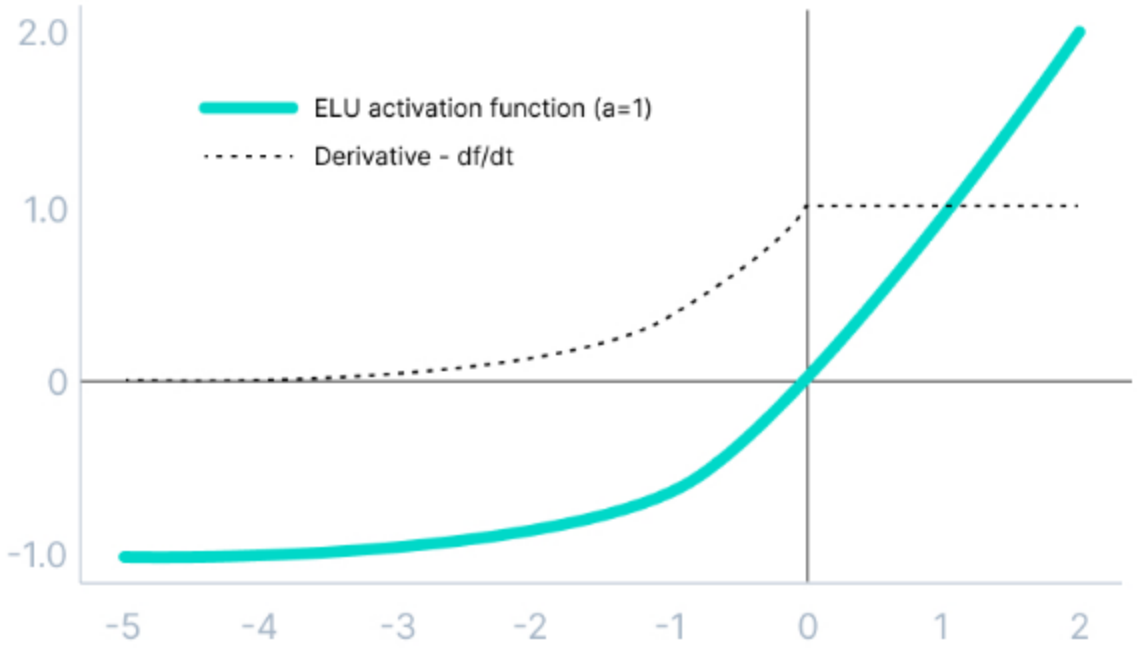
ELU

$$\begin{cases} x & for\ x \geqslant 0 \\ \alpha(e^x - 1) & for\ x < 0 \end{cases}$$

Activation Function

ELU Activation Function

- **Exploding Gradient Problem**

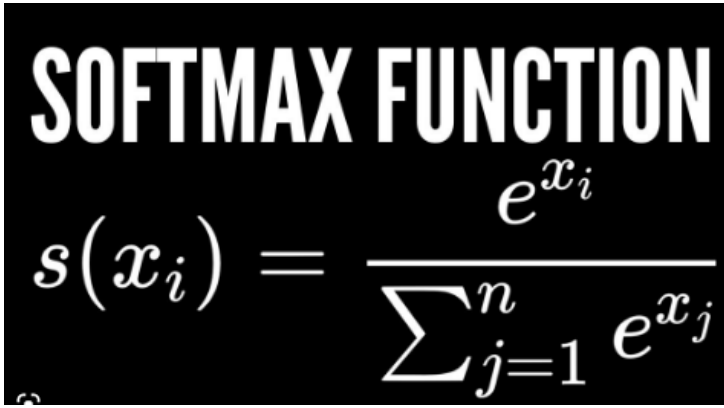


- Exploding gradients are **a problem when large error gradients accumulate and result in very large updates to neural network model weights during training**.
- An unstable network can result when there are exploding gradients, and the learning cannot be completed.
- The values of the weights can also become so large as to overflow.

Activation Function

Softmax

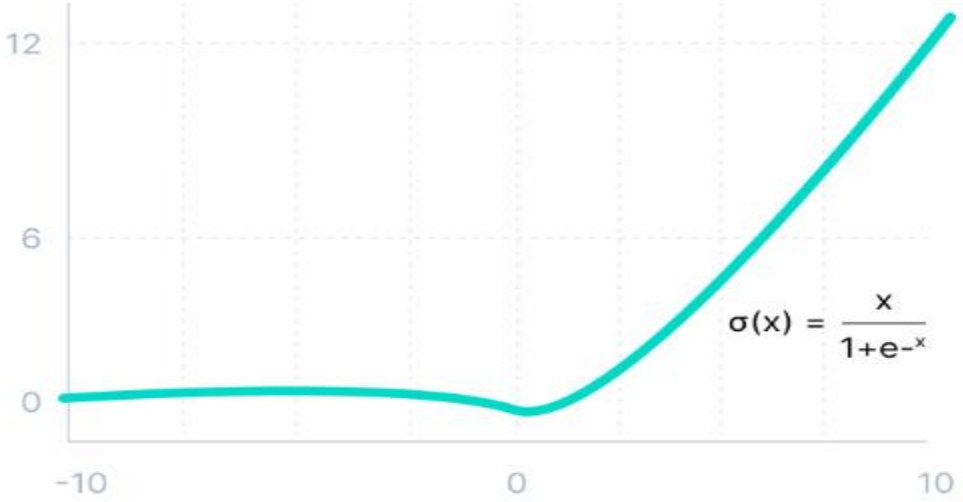
- It is most commonly used as an activation function for the last layer of the neural network in the case of **multi-class classification.**
- It **generates probability vector to indicate the probability of each of the classes**
- Softmax function is a combination of multiple sigmoid functions.
- The function, for every data point of all the **individual classes, returns the probability**.



Activation Function

Swish

- Swish consistently matches or outperforms ReLU activation function on deep networks applied to various challenging domains such as image classification, machine translation etc.
- This function is bounded below but unbounded above i.e. *Y* approaches to a constant value as *X* approaches negative infinity but *Y* approaches to infinity as *X* approaches infinity.



$$f(x) = x * \text{sigmoid}(x)$$
$$f(x) = x/(1 - e^{-x})$$

Activation Function

Swish

- Swish is a smooth function that means that it does not abruptly change direction like ReLU does near x = 0. Rather, it smoothly bends from 0 towards values < 0 and then upwards again.
- Small negative values were zeroed out in ReLU activation function. However, those negative values may still be relevant for capturing patterns underlying the data.
- **Large negative values are zeroed out for reasons of sparsity making it a win-win situation.**

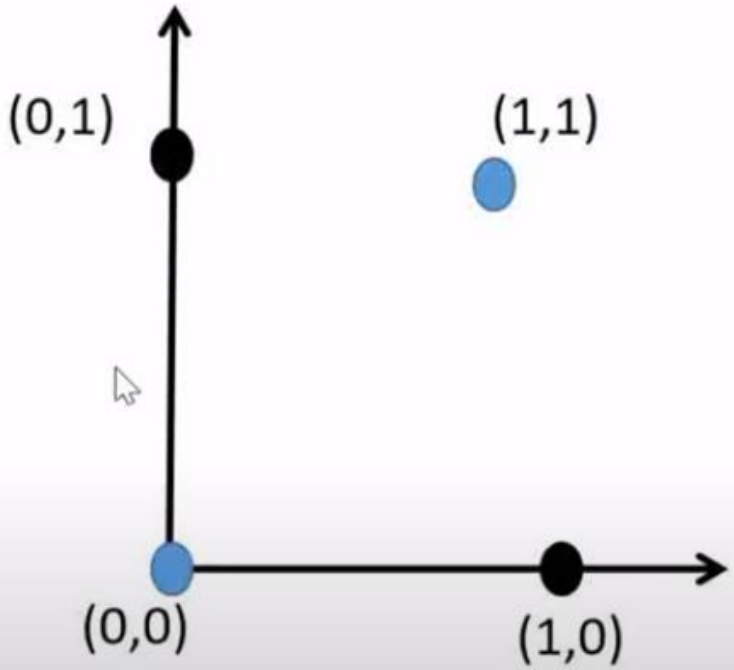
How to Use ??

- ReLU activation function should generally be used in the hidden layers.
- Sigmoid/Logistic and Tanh functions should not be used in hidden layers as they make the model more susceptible to problems during training (due to vanishing gradients).
- Swish function is used in neural networks having a depth greater than 40 layers.
- **Regression** - Linear Activation Function
- **Binary Classification**—Sigmoid/Logistic Activation Function
- **Multiclass Classification**—Softmax

Non-Linearly Separable

x1	x2	y
1	1	0
1	0	1
0	1	1
0	0	0

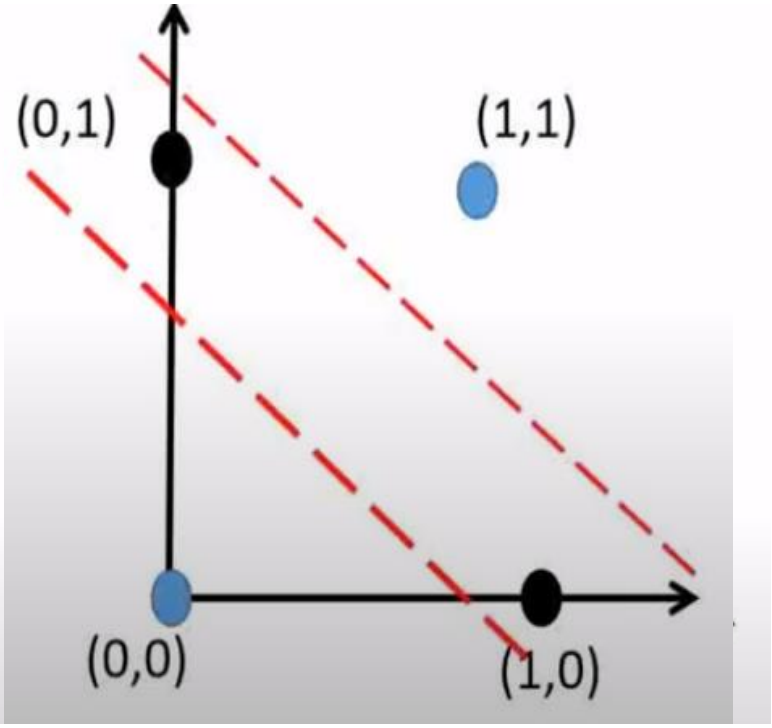
LOGIC
XOR



Non-Linearly Separable

x1	x2	y
1	1	0
1	0	1
0	1	1
0	0	0

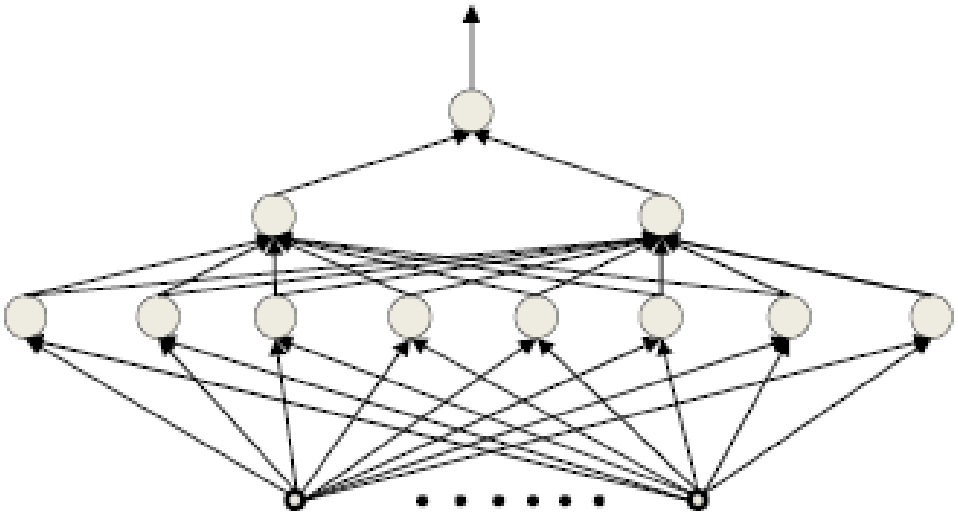
LOGIC XOR

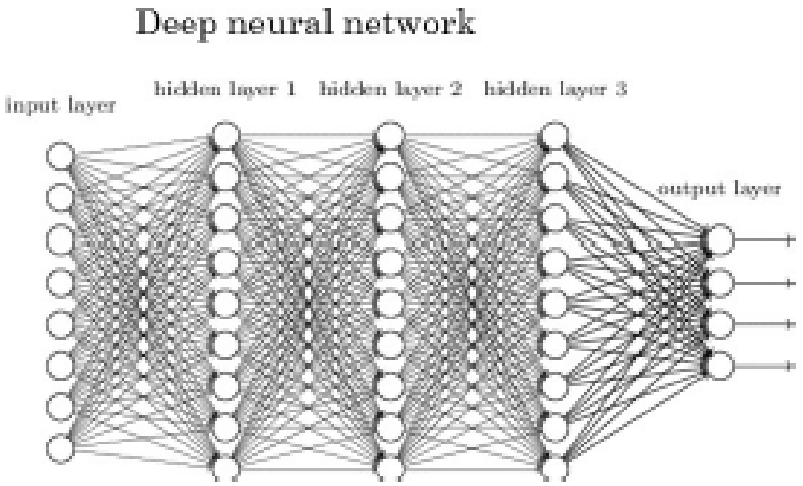


Multi-Layer Perceptrons (MLP)

- The XOR problem with neural networks can be solved by using Multi-Layer Perceptrons or a neural network architecture with an input layer, hidden layer, and output layer.
- So, during the forward propagation through the neural networks, the weights get updated to the corresponding layers and the XOR logic gets executed.

Multi-Layer Perceptrons (MLP)





- A network of perceptrons
  - Generally “layered”

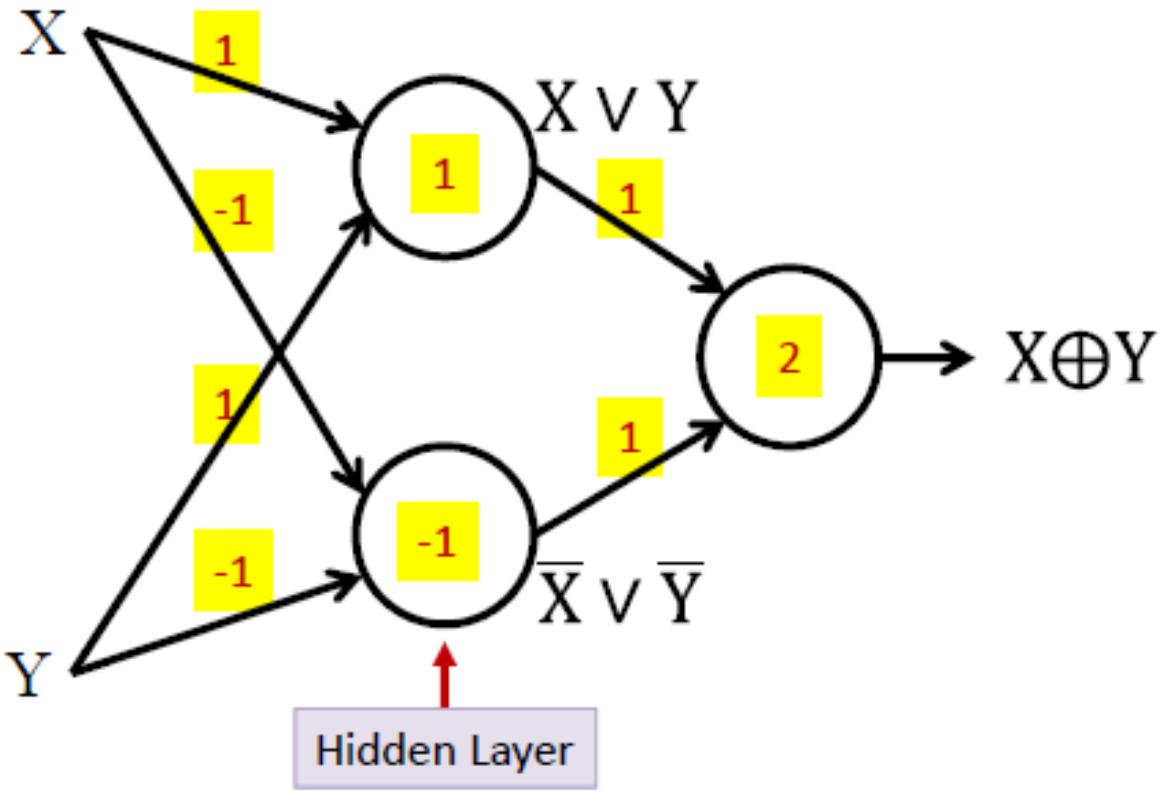
Multi-Layer Perceptrons (MLP)

Deep Structures

- In any directed network of computational elements with input source nodes and output sink nodes, "depth" is the length of the longest path from a source to a sink

- Left: Depth = 2. Right: Depth = 3

Multi-Layer Perceptrons (MLP)



MLPs can compute the XOR

- .....