

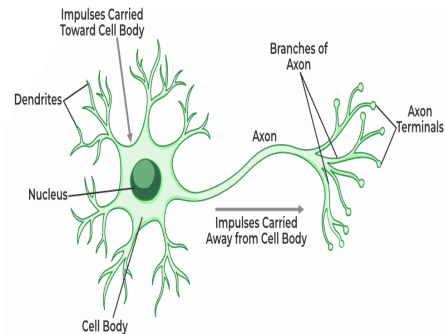


# Course: Machine Learning Techniques (CSE3008) - Module3

Dr.Mohana S D,  
Assistant Professor,  
(Course In-charge - CSE3008)  
School of Computer Science and Engineering & Information Science,  
Presidency University Bengaluru.



## Perceptron Learning



# Perceptron Learning

## Biological neuron has three basic functionality

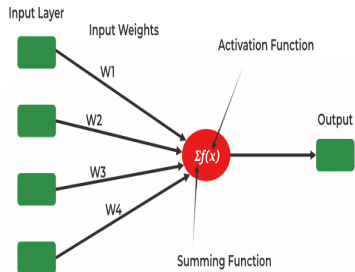
- Receive signal from outside.
- Process the signal and enhance whether we need to send information or not.
- Communicate the signal to the target cell which can be another neuron or gland.

# Perceptron Learning

## Biological neuron has three basic functionality

- It is one of the oldest and first introduced neural networks.
- It was proposed by Frank Rosenblatt in 1958.
- Perceptron is also known as an artificial neural network.
- Perceptron is mainly used to compute the logical gate like AND, OR, and NOR which has binary input and binary output.

## Perceptron Learning

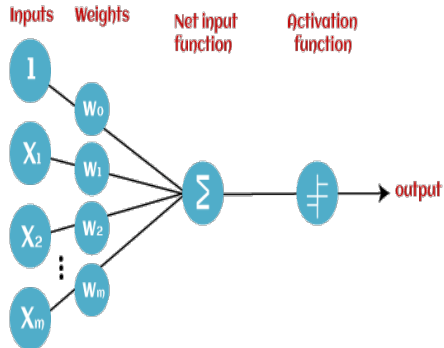


# Perceptron Learning

- Perceptron is considered as a single-layer neural network that consists of four main parameters named input values (Input nodes), weights and Bias, net sum, and an activation function.
- The perceptron model begins with the multiplication of all input values and their weights, then adds these values together to create the weighted sum.
- Then this weighted sum is applied to the activation function 'f' to obtain the desired output. This activation function is also known as the step function and is represented by 'f'.



## Perceptron Learning



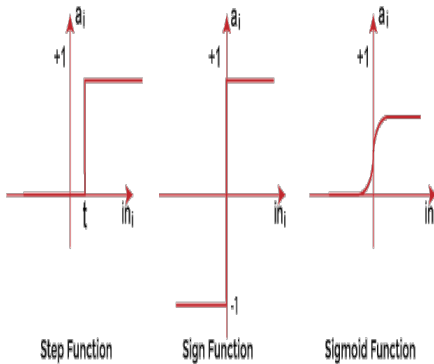
# Perceptron Learning

- Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence..
- Input Nodes or Input Layer: This is the primary component of Perceptron which accepts the initial data into the system for further processing. Each input node contains a real numerical value.
- Weight and Bias: Weight parameter represents the strength of the connection between units and Bias can be considered as the line of intercept in a linear equation.

# Perceptron Learning

- Activation Function: Activation Function can be considered primarily as a step function.
- Types of Activation functions: Sign function, Step function, and Sigmoid function

## Perceptron Learning



# Perceptron Learning

Based on the layers, Perceptron models are divided into two types. These are as follows:

- Single-layer Perceptron Model
- Multi-layer Perceptron model

# Perceptron Learning

## Single Layer Perceptron Model:

- A single-layered perceptron model consists feed-forward network and also includes a threshold transfer function inside the model.
- The main objective of the single-layer perceptron model is to analyze the linearly separable objects with binary outcomes.

# Perceptron Learning

## Multi-Layered Perceptron Model:

The multi-layer perceptron model is also known as the Backpropagation algorithm, which executes in two stages as follows:

- Forward Stage: Activation functions start from the input layer in the forward stage and terminate on the output layer.
- Backward Stage: In the backward stage, weight and bias values are modified as per the model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.

# Perceptron Learning

## Perceptron Function:

Perceptron function " $f(x)$ " can be achieved as output by multiplying the input ' $x$ ' with the learned weight coefficient ' $w$ '.

$$f(x)=1;$$

$$\text{if } w.x+b > 0$$

$$\text{otherwise, } f(x)=0$$

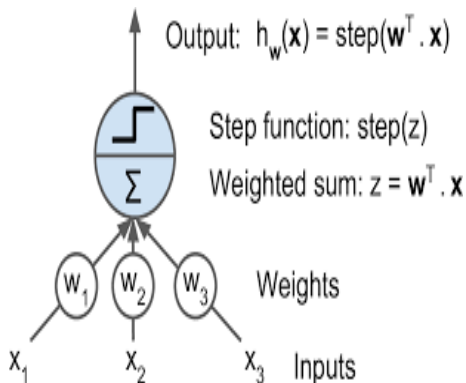
' $w$ ' represents real-valued weights vector ' $b$ ' represents the bias ' $x$ ' represents a vector of input  $x$  values.



# Perceptron Learning

- Artificial neuron, also called linear threshold unit(LTU),by McCulloch and Pitts,1943: with one or more numeric inputs,it produces a weighted sum of them, applies an activation function,and outputs the result.
- Common activation functions: step function and sigmoid function

## Linear Threshold Unit(LTU)



# Linear Threshold Unit(LTU)

- Linear threshold units (LTUs) are similar to perceptrons but use a linear activation function instead of a step function.
- This means that the output of an LTU is a continuous value between 0 and 1, rather than a binary 0 or 1.
- LTUs are used in multi-layer neural networks, where the output of one LTU is used as input to another LTU.
- Linear Threshold Units (LTUs) are a type of artificial neuron that compute a weighted sum of their inputs and apply a threshold function to produce an output.
- The threshold function is a step function that maps the weighted sum to either 0 or 1, depending on whether the sum is above or below a certain threshold value.

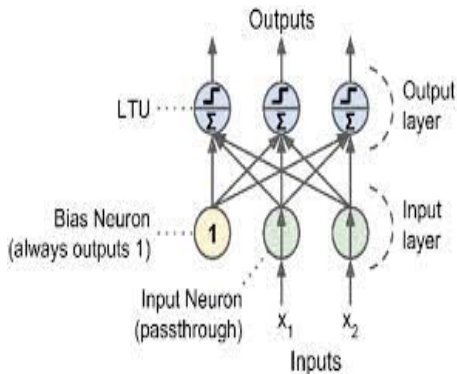
# Linear Threshold Unit(LTU)

- More specifically, an LTU takes a set of input values,  $x_1, x_2, \dots, x_n$ , and computes a weighted sum of the form  $w_1x_1 + w_2x_2 + \dots + w_nx_n$ .
- Here, the weights  $w_1, w_2, \dots, w_n$  are learned parameters that determine the strength of the inputs in influencing the output.
- The LTU then applies a threshold function, which is usually a step function, to the weighted sum.
- If the weighted sum is greater than or equal to the threshold value, the output is 1, otherwise the output is 0.

# Linear Threshold Unit(LTU)

- The threshold value is also a learned parameter that determines how sensitive the LTU is to the inputs.
- For example, if the threshold value is very low, the LTU will be more likely to output 1, even if the weighted sum is not very large. On the other hand, if the threshold value is very high, the LTU will only output 1 if the weighted sum is very large.
- LTUs can be used as building blocks for more complex neural networks, such as multi-layer perceptrons.
- In a multi-layer perceptron, the output of one layer of LTUs serves as input to the next layer, and the weights and threshold values are adjusted during training using a technique called backpropagation.

## Linear Threshold Unit(LTU)



# Logical computations with Perceptrons

## Logical computations with Perceptrons:

Perceptrons are simple computational units in neural networks that can perform binary classification tasks. They take in multiple inputs and produce a single output. The output is determined by a weighted sum of the inputs, which is passed through an activation function.

- The activation function of a perceptron is typically a step function, which produces a binary output of 1 or 0 depending on whether the weighted sum of the inputs is above or below a threshold value.
- This makes perceptrons useful for solving problems where the input features can be cleanly separated into two classes.

# Logical computations with Perceptrons

To perform logical computations with perceptrons, we can use them to create logical gates such as AND, OR, and NOT gates:

AND gate: An AND gate takes two binary inputs and produces a binary output that is 1 if both inputs are 1, and 0 otherwise. We can implement an AND gate with two perceptrons, each with two inputs and a threshold of 2. The weights on each input can be set to 1, so that the output is 1 only when both inputs are 1.



# Logical computations with Perceptrons

To perform logical computations with perceptrons, we can use them to create logical gates such as AND, OR, and NOT gates:

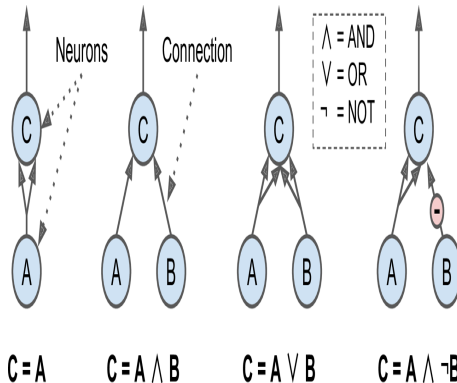
OR gate: An OR gate takes two binary inputs and produces a binary output that is 1 if either input is 1, and 0 otherwise. We can implement an OR gate with two perceptrons, each with two inputs and a threshold of 1. The weights on each input can be set to 1, so that the output is 1 if either input is 1.

# Logical computations with Perceptrons

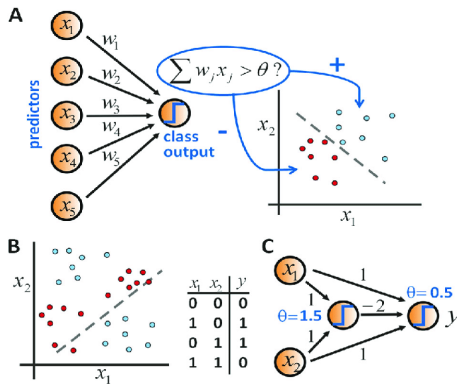
To perform logical computations with perceptrons, we can use them to create logical gates such as AND, OR, and NOT gates:

By combining these basic logical gates, we can build more complex logical circuits using perceptrons. For example, we can implement an XOR gate (which outputs 1 if either input is 1, but not both) using a combination of AND, OR, and NOT gates.

## Logical computations with Perceptrons



# Logical computations with Perceptrons









# Logical computations with Perceptrons

To perform logical computations with perceptrons, we can use them to create logical gates such as AND, OR, and NOT gates:

**NOT gate:** A NOT gate takes a single binary input and produces a binary output that is the opposite of the input (i.e., 1 if the input is 0, and 0 if the input is 1). We can implement a NOT gate with a single perceptron with one input and a threshold of 0.5. The weight on the input can be set to -1, so that the output is 1 if the input is 0, and 0 if the input is 1.

By combining these basic logical gates, we can build more complex logical circuits using perceptrons. For example, we can implement an XOR gate (which outputs 1 if either input is 1, but not both) using a combination of AND, OR, and NOT gates.

# Activation Functions

Commonly Used Activation Functions			Range
1. Step function: $f(z) = \begin{cases} 0 & z < 0 \\ 1 & z \geq 0 \end{cases}$	①		$\{0, 1\}$
2. Signum function: $f(z) = \begin{cases} -1 & z < 0 \\ 0 & z = 0 \\ 1 & z > 0 \end{cases}$	②		$\{-1, 1\}$
3. Linear function: $f(z) = z$	③		$(-\infty, \infty)$
4. ReLU function: $f(z) = \begin{cases} 0 & z < 0 \\ z & z \geq 0 \end{cases}$	④		$(0, \infty)$
5. Sigmoid function: $f(z) = \frac{e^z}{1+e^z}$	⑤		$(0, 1)$
6. Hyperbolic tan: $\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	⑥		$(-1, 1)$

by Dr. Pankaj Kumar Porwal (BTech - IT Mumbai, PhD - Cornell University) : Principal, Techno India NIR Institute of Technology, Udaipur

# Multi-layer Perceptrons (MLPs)

## Multi-layer Perceptrons (MLPs):

Multi-layer Perceptrons (MLPs) are a type of Deep Neural Network (DNN) that consist of an input layer, several hidden layers, and an output layer. MLPs are highly effective in supervised learning tasks and can approximate any function in a continuous domain. The activation functions used in MLPs are sigmoid, tanh, ReLU, and leaky ReLU.

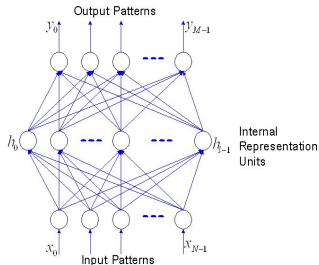
The Backpropagation algorithm is used to train the MLPs by efficiently computing the derivative with respect to each parameter. However, training MLPs can be computationally expensive and require a large amount of data. Techniques such as regularization, dropout, and batch normalization are used to improve their performance.

# Multi-layer Perceptrons (MLPs)

## Multi-layer Perceptrons (MLPs):

Regularization helps to prevent overfitting, dropout helps to reduce over-reliance on specific neurons, and batch normalization helps to improve the generalization performance of MLPs.

## Multi-Layer Perceptron Model





# Multi-layer Perceptrons (MLPs)

## Multi-layer Perceptrons (MLPs):

It is a type of neural network that consists of multiple layers of interconnected neurons. MLPs are trained using the Backpropagation algorithm, which is a supervised learning algorithm that uses Gradient Descent to minimize the error between the predicted and actual output.

During training, the Backpropagation algorithm calculates the error between the predicted and actual output and then propagates it back through the network to adjust the weights of the connections between the neurons. This process is repeated multiple times until the error is minimized.

# Multi-layer Perceptrons (MLPs)

## Multi-layer Perceptrons (MLPs):

MLPs are used in a wide range of applications, including image classification, speech recognition, and natural language processing. However, they can be computationally expensive and require a large amount of data to train effectively.

To improve the performance of MLPs, various techniques such as regularization, dropout, and batch normalization can be used. These techniques help to prevent overfitting, reduce the impact of noisy data, and improve the convergence rate of the algorithm.

# Multi-layer Perceptrons (MLPs)

## Backpropagation:

Backpropagation is a widely used algorithm for training multilayer neural networks. It works by computing the gradient of the loss function with respect to each weight in the network, and then using this gradient to update the weights using the gradient descent algorithm.

The basic idea behind backpropagation is to propagate the error backwards through the network, from the output layer to the input layer. At each layer, the error is multiplied by the derivative of the activation function with respect to the input, which gives the gradient of the error with respect to the weights and biases at that layer.

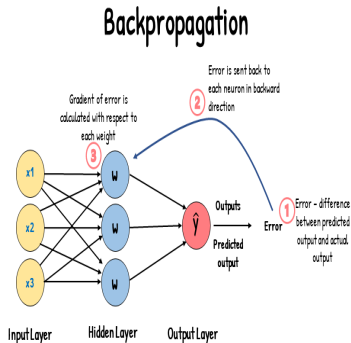
# Multi-layer Perceptrons (MLPs)

## Backpropagation:

Once the gradients have been computed, they can be used to update the weights and biases using a learning rate and the gradient descent update rule. This process is repeated for each input in the training set until the network converges to a minimum of the loss function.

Backpropagation is a powerful algorithm for training neural networks, but it can suffer from overfitting and slow convergence. Techniques such as regularization, dropout, and batch normalization can be used to mitigate these issues and improve the performance of the network.

# Backpropagation



## 18870backprop2