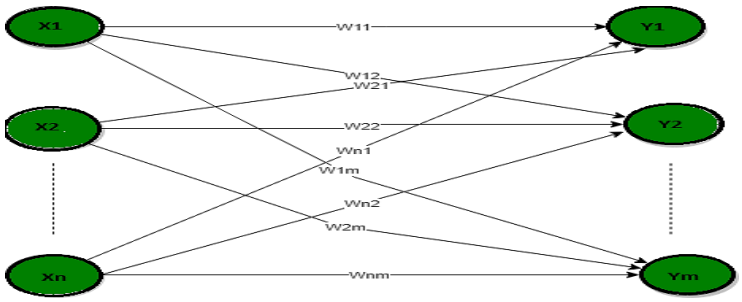
Unit II Artificial Neural Networks

01. WHAT ARE THE BASIC MODELS OF ARTIFICIAL NEURAL NETWORKS?

An artificial neural network (ANN) is a computational model ***inspired by the structure and function of the human brain,*** specifically ***designed to process and analyse complex data patterns.*** ANNs ***learn from examples*** and are ***configured for specific tasks, such as pattern recognition or data classification,*** through a learning process that involves ***adjusting the synaptic connections between neurons.***

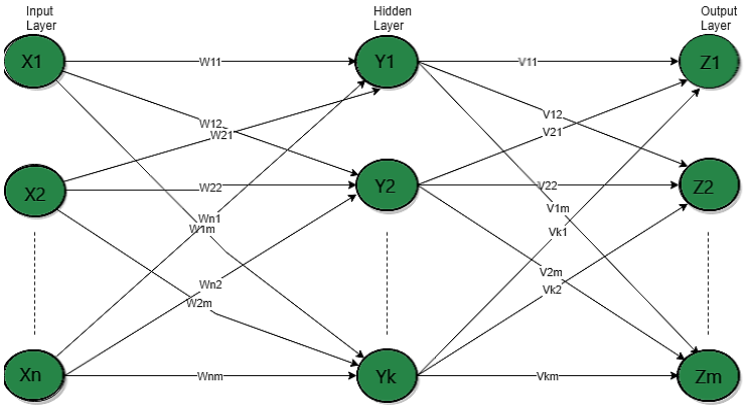
The basic models of artificial neural networks include:

1. **Single-layer feed-forward network:**



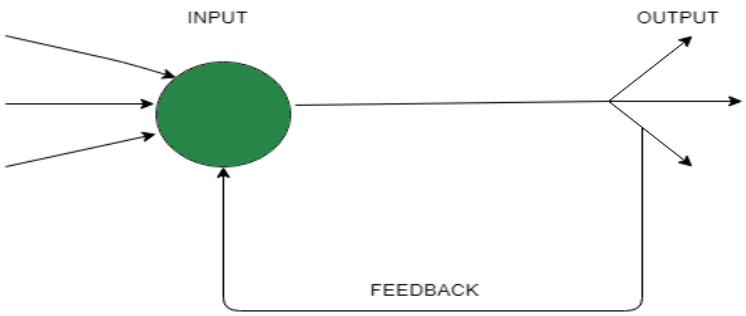
This is the simplest type of ANN architecture, ***consisting of only two layers - an input layer and an output layer.*** The input layer simply buffers the input signal, while the output layer computes the output signals ***based on different weights applied to input nodes.***

2. **Multilayer feed-forward network:**



This architecture ***includes one or more hidden layers,*** which are ***internal to the network and do not have direct contact with the external layer.*** The existence of hidden layers makes the network ***computationally stronger.*** Information flows through the input layer, through intermediate computations in the hidden layers, and finally determines the output.

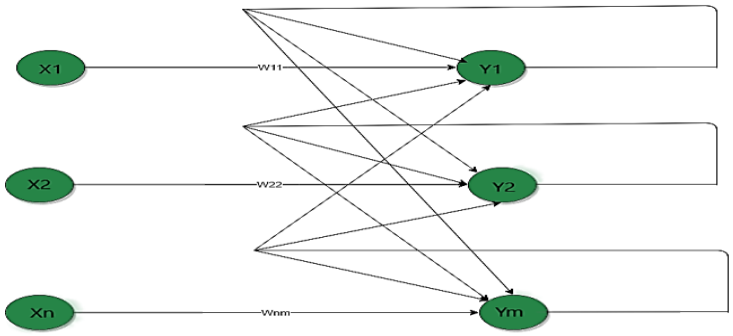
3. **Single node with its own feedback:**



Unit II Artificial Neural Networks

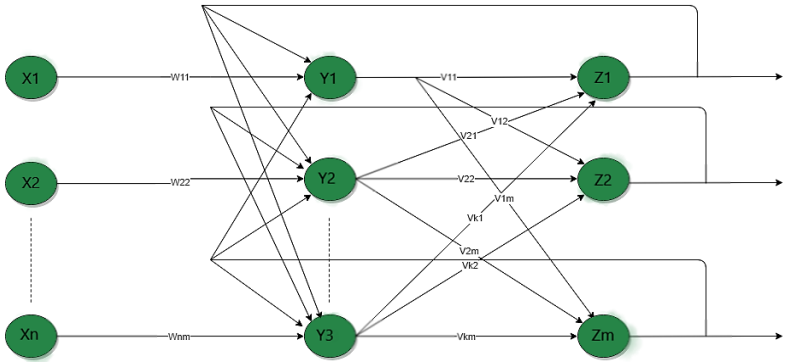
This type of network includes ***feedback connections,*** where ***outputs can be directed back as inputs to the same layer or preceding layer nodes. Recurrent neural networks*** are examples of feedback networks, where ***connections form closed loops.***

4. **Single-layer recurrent network:**



In this architecture, ***processing elements' outputs can be directed back to themselves or to other processing elements, forming a feedback connection.*** Recurrent neural networks (RNNs) are a class of artificial neural networks where ***connections between nodes form a directed graph*** along a sequence, allowing them to exhibit ***dynamic temporal behaviour.***

5. **Multilayer recurrent network:**

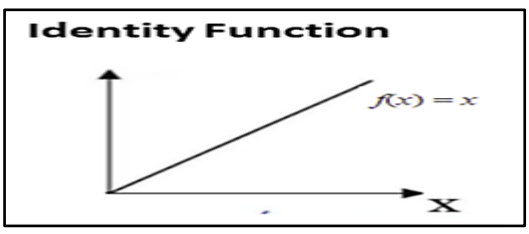


This type of network includes ***processing elements whose outputs can be directed to processing elements in the same layer and in the preceding layer, forming a multilayer recurrent network.*** These networks ***perform tasks for every element of a sequence,*** with outputs dependent on previous computations, and utilize a hidden state to capture information about the sequence.

02. ILLUSTRATE VARIOUS ACTIVATION FUNCTIONS IN ANN.

Activation functions are crucial components of artificial neural networks (ANNs) that ***introduce non-linearity into the network's decision-making process.*** They determine whether a neuron should be ***activated or not,*** based on the **weighted sum of its inputs/ net input.**

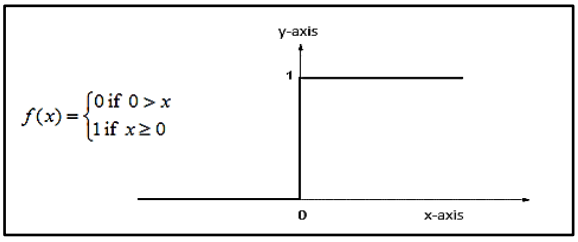
1. **Identity Function:**



Unit II Artificial Neural Networks

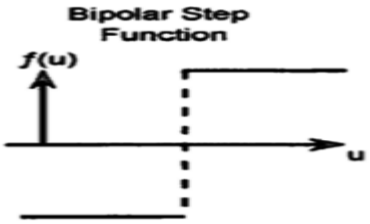
- **Definition:** Maps input to the same output value; ***linear operator.***
- **Equation:** *f(x) = a * x,* where ***a*** is a constant (e.g., f(x) = 2x)
- **Limitation:** Cannot handle complex scenarios due to ***lack of non-linearity.***

2. **Binary Step Function:**



- **Definition:** Outputs either 1 ***(activated)*** or 0 ***(not activated)*** based on a threshold.
- **Equation:** f(x) = 1 if x >= threshold, 0 otherwise.
- **Limitation:** ***Not differentiable at zero,*** leading to zero gradients. ***Suitable only for binary classification problems.***

3. **Bipolar Step Function:**

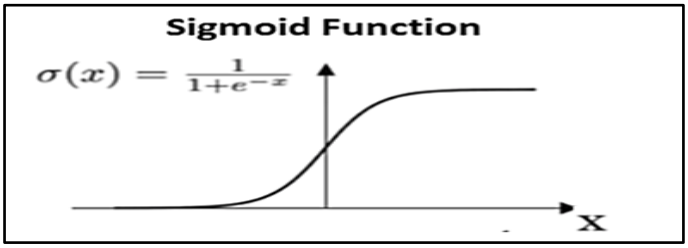


- **Definition:** ***Outputs +1 if above the threshold and -1 if below the threshold.***
- **Equation:**

$$f(x) = \begin{cases} 1, & \text{if } x \geq \text{threshold} \\ -1, & \text{otherwise} \end{cases}$$

- **Limitations:** Since it generates bipolar outputs, ***suitable only for single-layer networks.***

4. **Sigmoid Function:**



- **Definition:** S-shaped functions; ***introduces non-linearity.***
- **Types:**
  - **Binary Sigmoid Function (Logistic Function):** Outputs vary between 0 and 1.

- **Bipolar Sigmoid Function (Hyperbolic Tangent Function or Tanh):** Outputs vary between -1 and 1.

- **Equation (Binary Sigmoid):** $f(z)= 1/1+e^{-z}$
- **Equation (Bipolar Sigmoid):** $f(z) = \frac{2}{1+e^{-z}} - 1$
- **Advantages:** ***Differentiable, non-linear; suitable for multi-class classification tasks.***
- **Limitations:** ***Vanishing gradients; not zero-centric.***

Illustrating various activation functions in ANN ***helps understand their properties and how they contribute to the network's ability*** to ***learn complex patterns and make accurate predictions.*** Each activation function has its advantages and limitations, and ***choosing the right one depends on the specific task and dataset at hand.***

03. DEFINE THE FOLLOWING WITH RESPECT TO ARTIFICIAL NEURAL NETWORKS.

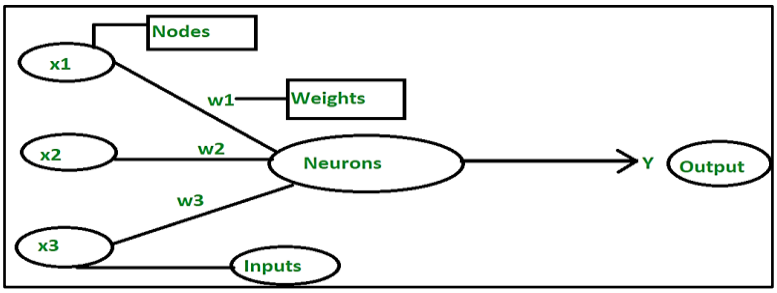
1. Weight

2. Bias

3. Threshold

4. Learning rate

An artificial neural network (ANN) is a computational model ***inspired by the structure and function of the human brain, specifically designed to process and analyse complex data patterns.*** Similar to the brain having neurons interlinked to each other, the ANN also has neurons that are linked to each other in various layers of the networks which are known as nodes.



Basic Architecture of ANN

1. **Weight:**

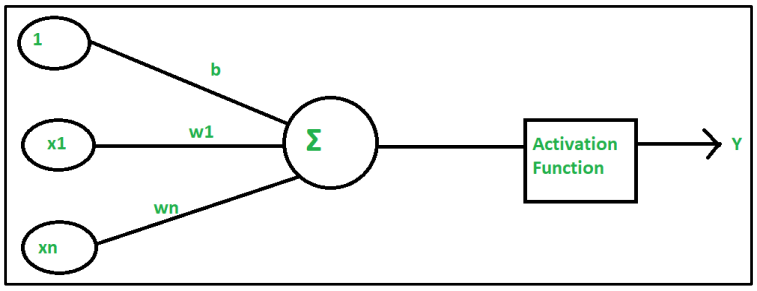
- **Definition:** In artificial neural networks, a weight represents the ***strength of the connection between two neurons.*** It ***determines the impact of the input on the output of a neuron.***
- **Role:** During the learning process, ***weights are adjusted to minimize the difference between the actual output and the desired output,*** allowing the network to learn from the input data.

Unit II Artificial Neural Networks

- **Representation:** Each connection in a neural network has an associated weight***, denoted by w.*** A ***larger weight means the connection has a stronger influence on the neuron's output.***
- **Weight Updating Formula:**

$$Weight(new) = Weight(old) + \alpha t x$$

2. **Bias:**



- **Definition:** Think of bias as an **adjustable constant** added to the weighted sum of inputs in a neuron. Bias provides the **flexibility** for a ***neuron to activate even if the weighted sum of inputs doesn't reach a certain threshold on its own.***
- **Role:** Bias helps the neural network to ***better fit the training data by shifting the activation function horizontally.`***
- **Representation:** Bias is typically ***denoted by b*** and added to the weighted sum of inputs before passing through the activation function.
- **Bias Adjustment Formula:**

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

3. **Threshold:**

- **Definition:** In the context of artificial neural networks, the threshold refers to a ***value used to define activation functions, such as the binary step function.*** It ***determines the point at which the neuron fires or activates.***
- **Role:** ***If the weighted sum of inputs (plus bias) exceeds the threshold,*** the ***neuron activates and produces an output;*** otherwise, it remains ***inactive***. Fires can be of ***excitatory or inhibitory.***
- **Representation:** Threshold is represented by a ***scalar value,*** typically ***denoted as θ.***

4. **Learning Rate:**

- **Definition:** Learning rate is a ***hyperparameter that controls the size of the update to the weights and bias*** during the training of a neural network. It ***determines how much the weights are adjusted in response to the error calculated during training.***
- **Role:** Learning rate ***affects the convergence speed and stability of the training process.*** A ***higher learning rate can lead to faster convergence*** but may ***risk***

overshooting the optimal solution, while a ***lower learning rate may result in slower convergence*** but ***more stable training.***

- **Representation:** Learning rate is denoted by ***α or η*** and is typically **set before training begins.**

04. DESIGN MCCULLOCH PITTS NEURON MODEL WITH EQUATIONS.

The McCulloch Pitts neuron model is one of the earliest artificial neural network models, proposed by ***Warren McCulloch*** and ***Walter Pitts in 1943.*** It ***consists of a single neuron with binary inputs and outputs,*** and ***activation function.*** The neuron ***computes the weighted sum of the inputs and compares it with a threshold value to produce the output.*** The equation for the output of the McCulloch Pitts neuron is:

$$y_{out} = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where ***xi are the inputs, wi are the weights,*** and ***θ is the threshold.***

The McCulloch Pitts neuron model can be used to implement simple logical functions, such as ***AND, OR, and NOT,*** by ***choosing appropriate values for the weights and the threshold.***

For example, to implement the logical AND:

let's calculate the output of the McCulloch-Pitts neuron model for the logical AND operation with the given ***weights, bias, and threshold.***

Given:

- **W1** for ***x1*** = 1
- **W2** for ***x2*** = 1
- **Bias** = -1.5
- **Threshold** = 0

We'll calculate the output for different input combinations of $x1$ and $x2$:

1. $x1=0, x2=0$:
   $z=(1\times0)+(1\times0)+(-1.5)=-1.5$
   Since z<0, the output is 0.

2. $x1=0, x2=1$:
   $z=(1\times0)+(1\times1)+(-1.5)=-0.5$
   Since z<0, the output is 0.

3. $x1=1, x2=0$:
   $z=(1\times1)+(1\times0)+(-1.5)=-0.5$
   Since z<0, the output is 0.

Unit II Artificial Neural Networks

4. $x1=1, x2=1$:

$z=(1\times1)+(1\times1)+(-1.5)=0.5$

Since z≥0, the output is 1.

Therefore, the McCulloch-Pitts neuron model correctly ***implements the logical AND operation, producing 1*** only when ***both inputs x1 and x2 are 1.***

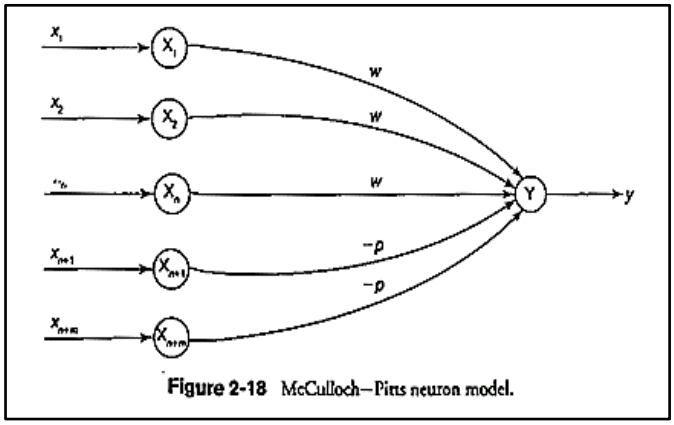


Figure 2-18 McCulloch–Pitts neuron model.

The McCulloch Pitts neuron model is a ***simple and elegant way to model the basic computation of a biological neuron,*** but it has some limitations. For instance, ***it cannot learn from data, as the weights and the threshold are fixed.*** It also ***cannot handle continuous or noisy inputs, as it only works with binary values.*** Moreover, it ***cannot represent more complex functions, such as XOR, with a single neuron.*** These limitations motivated the development of more advanced neural network models, such as the ***perceptron and the sigmoid neuron.***

05. EXPLAIN LINEAR SEPARABILITY.

Linear separability is a concept in machine learning that ***pertains to the ability of a linear model to separate data points of different classes with a hyperplane.***

In a neural network, linear separability means that it's possible to draw a ***straight line, plane, or hyperplane*** that ***separates the input vectors into two groups:*** one group with a ***positive response (+1)*** and another group with a ***negative response (-1).*** This separation is achieved by ***assigning appropriate weights (including bias) to the input units.***

The decision boundary is determined by the equation:

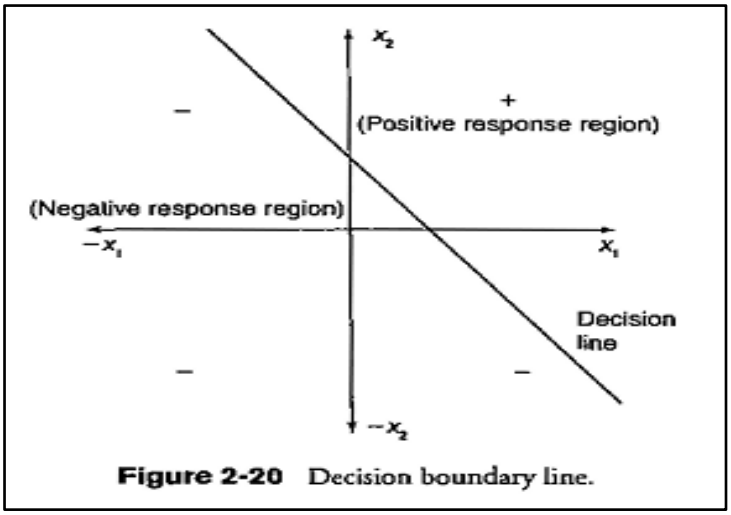


Figure 2-20 Decision boundary line.

Unit II Artificial Neural Networks

$$b + \sum_{i=1}^{n} x_i w_i = 0$$

where:

- b is the bias,
- xi are the input units,
- wi are the weights.

If the net input ***(yin) is positive,*** the output of the activation function is +1, and if yin is negative, the output is -1. ***The region where yin>0 and yin<0 is separated by the decision boundary.***

During training, the ***weights and bias are adjusted to make sure the network gives the right answers for the training examples.*** If the problem is ***"linearly separable,"*** it means there are some weights that can draw a clear line between the different classes in the data.

For example, consider a single-layer neural network. The net input for this network is given as:

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

The separating line between the values x1 and x2, so that the net gives a positive response on one side and negative response on the other side, is given as:

$$b + x_1 w_1 + x_2 w_2 = 0$$

If w1 is not equal to 0, then we get:

$$x_2 = -\frac{b}{w_1} - \frac{x_1}{w_2}$$

The requirement for the positive response of the net is:

$$x_1 w_1 + x_2 w_2 > b$$

During the training process, the values of w1, w2, and b are determined so that the net will produce a positive (correct) response for the training data.

06. EXPLAIN HEBB TRAINING ALGORITHMS.

The Hebbian learning rule, proposed by ***Donald Hebb in 1949,*** is a fundamental algorithm for ***updating weights and bias in neural networks.*** It operates as follows:

1. **Initial Weight Setup:** Start by initializing all weights to small values, typically around 0.
2. **Input Presentation:** Provide the network with an input pattern, specifying values for its input neurons.
3. **Output Calculation:** Let the network process the input and determine the resulting output.
4. **Weight Update:** Adjust weights based on the ***simultaneous activity of connected neurons:***
    - If ***both connected neurons are active*** (outputting 1), ***increase the weight.***

- If ***both are inactive*** (outputting 0), ***decrease the weight.***
- If ***only one neuron is active,*** the ***weight remains unchanged.***

5. **Iterative Process:** Repeat steps 2-4 for new input patterns, updating weights accordingly.

Weight Update Formula:

The weight update in Hebb's rule is computed as:
wi(new) = wi(old) + xi·y
where *wi* is the weight, *xi* is the input, and ***y is the neuron's output*** (learning signal).

Limitations:

- The ***Hebb rule is more suitable for real-valued data than binary data*** due to its inability to distinguish between certain conditions.
- ***Representation using bipolar data*** is advantageous to address this limitation.

This algorithm ***strengthens connections when connected neurons fire together and weakens them when they activate together but produce opposite outputs.***