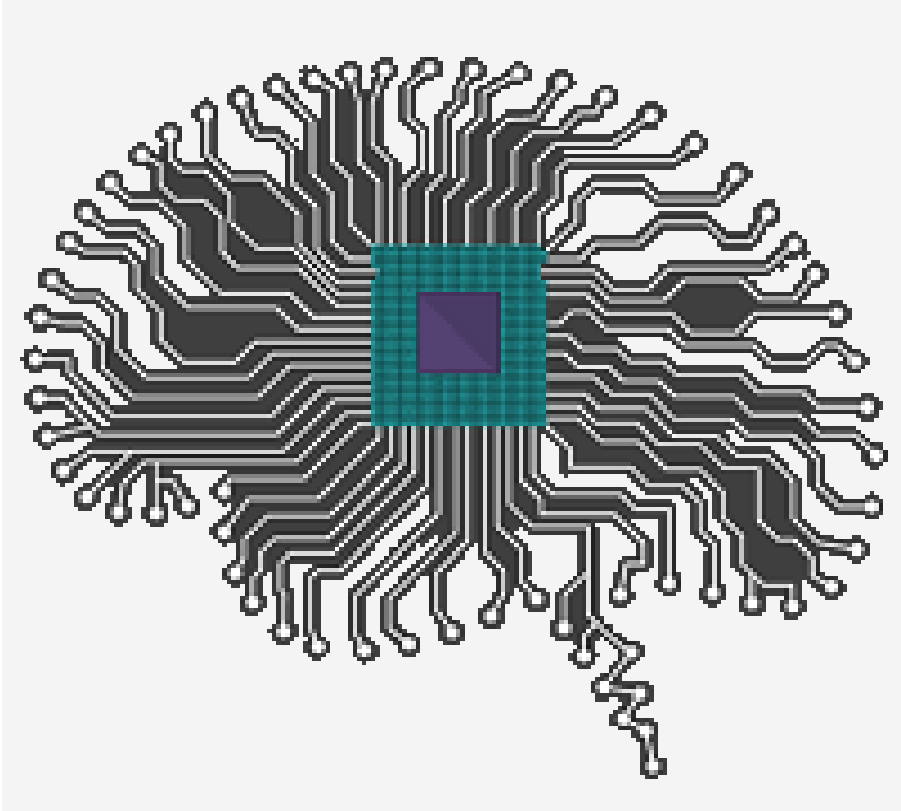


Artificial Neural Networks

By: Sajad Ahmadian

Kermanshah University of Technology

References

- Andrew W. Trask, Grokking Deep Learning, 2019.
- Charu C. Aggarwal, Neural Networks and Deep Learning, 2018.
- J. A. Anderson, An Introduction to Neural Networks, MIT Press, 1995.

Grading

- Seminar presentation (10%)
- Projects and exercises (15%)
- Midterm exam (25%)
- Final exam (50%)

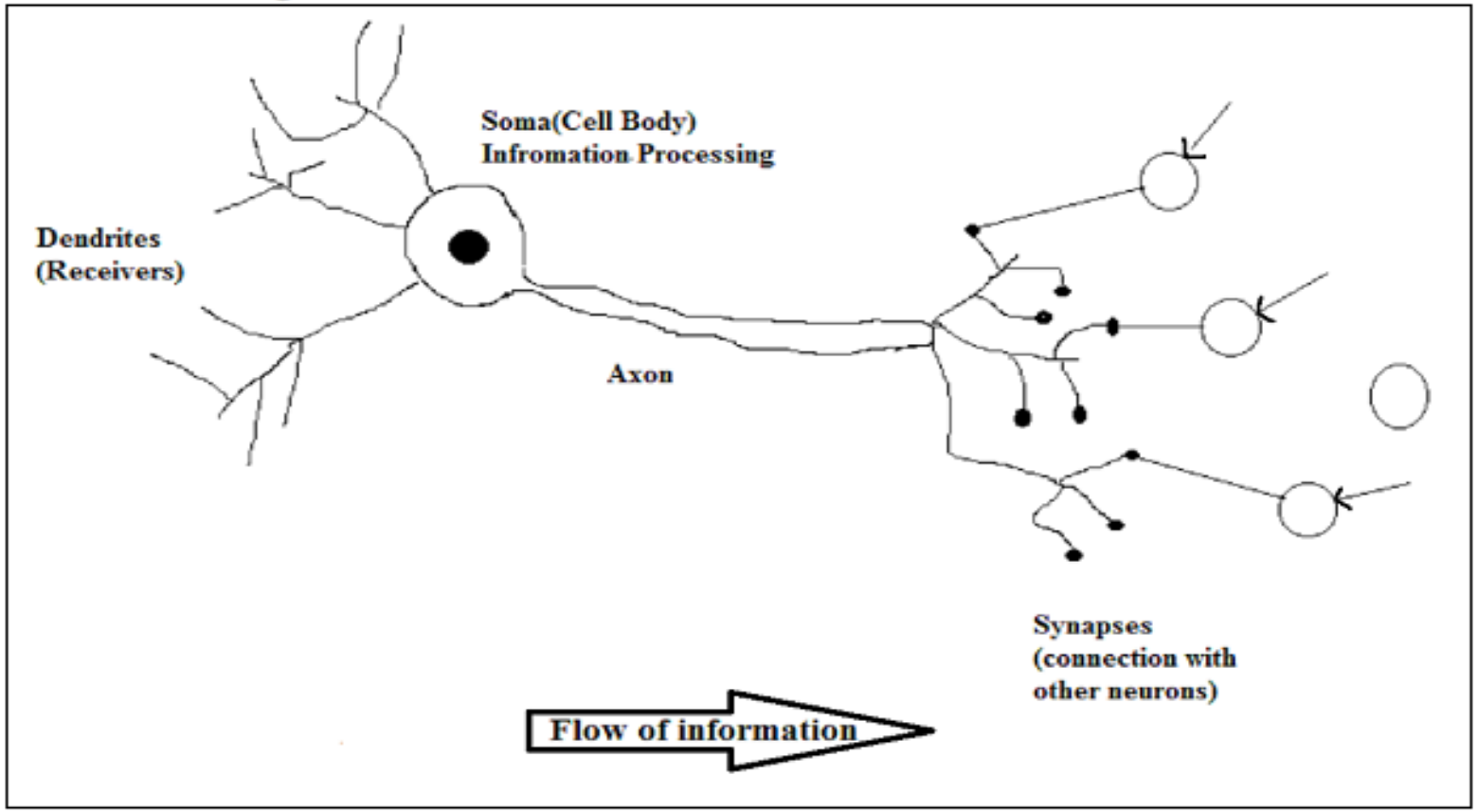
What is Artificial Neural Network?

- The term "Artificial Neural Network" is derived from Biological neural networks that develop the structure of a human brain.
- Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.
- The main objective is to develop a system to perform various computational tasks faster than the traditional systems. These tasks include pattern recognition and classification, approximation, optimization, and data clustering.

Biological Neuron

A nerve cell *neuron* is a special biological cell that processes information. According to an estimation, there are huge number of neurons, approximately 10^{11} with numerous interconnections, approximately 10^{15}.

Schematic Diagram



Working of a Biological Neuron

As shown in the above diagram, a typical neuron consists of the following four parts with the help of which we can explain its working −

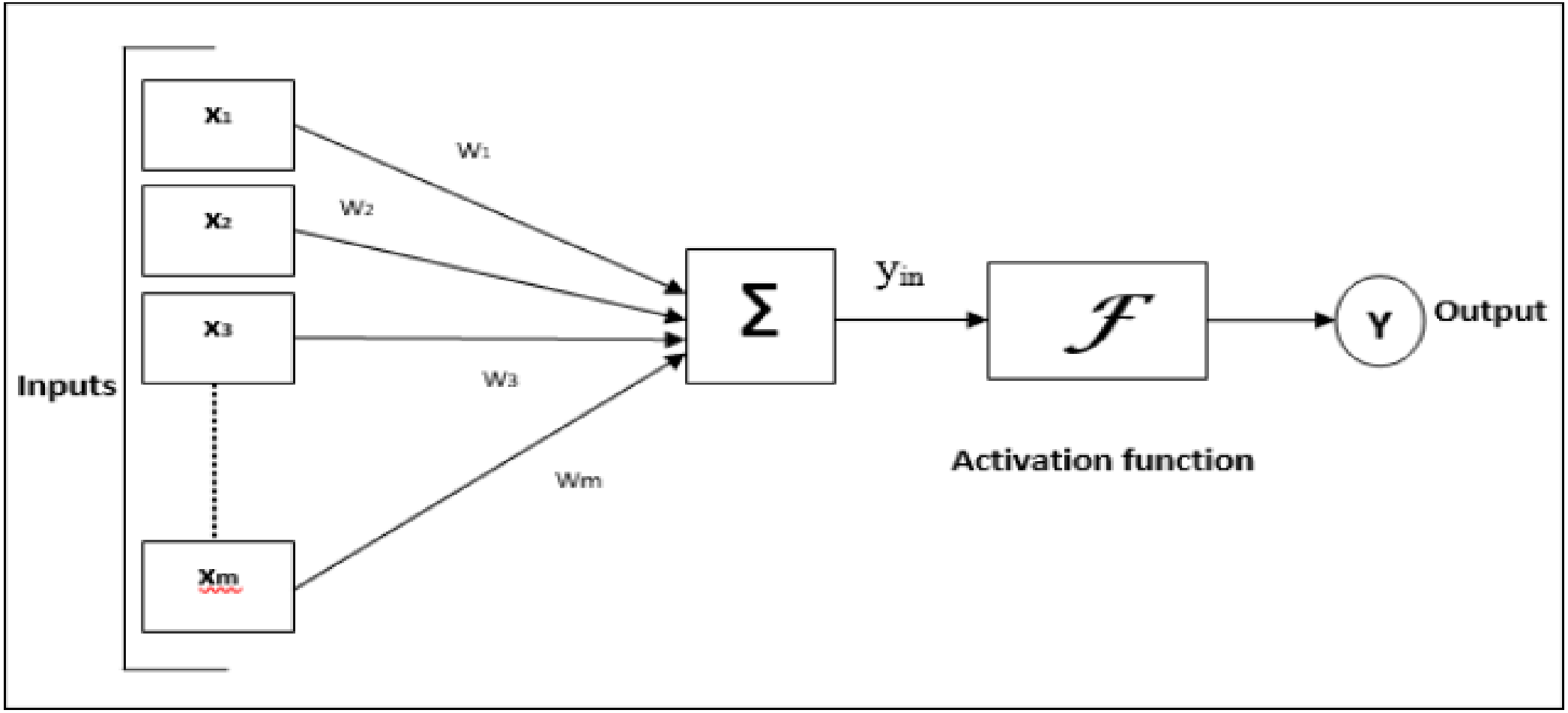
- **Dendrites** − They are tree-like branches, responsible for receiving the information from other neurons it is connected to. In other sense, we can say that they are like the ears of neuron.
- **Soma** − It is the cell body of the neuron and is responsible for processing of information, they have received from dendrites.
- **Axon** − It is just like a cable through which neurons send the information.
- **Synapses** − It is the connection between the axon and other neuron dendrites.

ANN versus BNN

Biological Neural Network BNN	Artificial Neural Network ANN
Soma	Node
Dendrites	Input
Synapse	Weights or Interconnections
Axon	Output

Model of Artificial Neural Network

The following diagram represents the general model of ANN followed by its processing.



Model of Artificial Neural Network

For the above general model of artificial neural network, the net input can be calculated as follows −

$$y_{in} = x_1.w_1 + x_2.w_2 + x_3.w_3 \ldots x_m.w_m$$

i.e., Net input $y_{in} = \sum_i^m x_i.w_i$

The output can be calculated by applying the activation function over the net input.

$$Y = F(y_{in})$$

Building Blocks

Processing of ANN depends upon the following three building blocks –

- Network Topology
- Adjustments of Weights or Learning
- Activation Functions

Network Topology

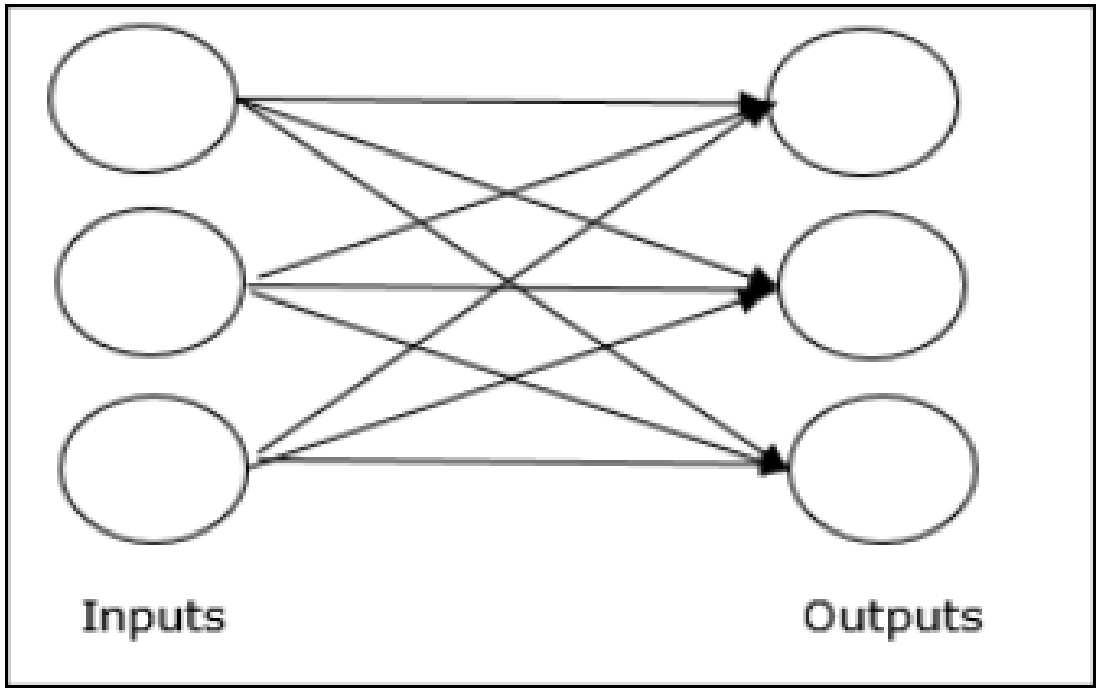
A network topology is the arrangement of a network along with its nodes and connecting lines. According to the topology, ANN can be classified as the following kinds –

Feedforward Network

It is a non-recurrent network having processing units/nodes in layers and all the nodes in a layer are connected with the nodes of the previous layers. The connection has different weights upon them. There is no feedback loop means the signal can only flow in one direction, from input to output. It may be divided into the following two types –

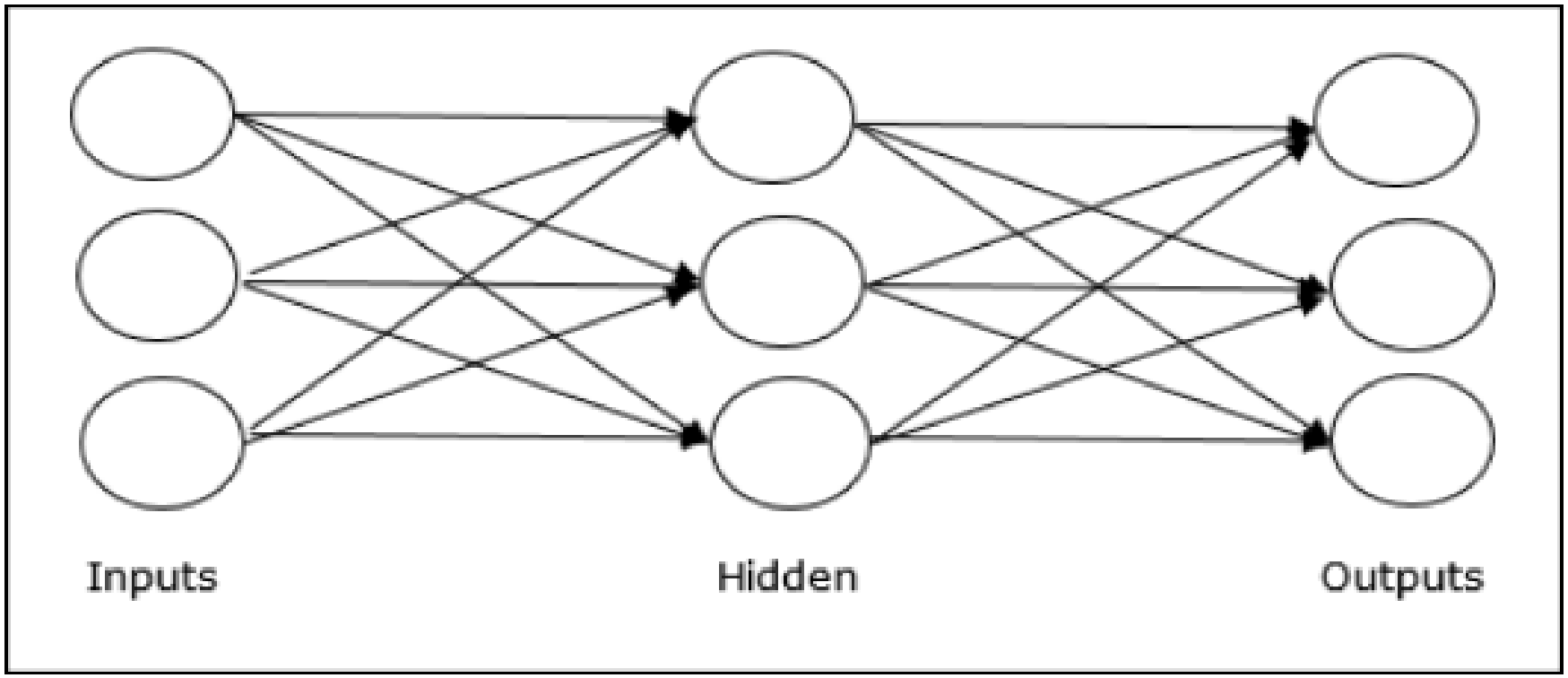
Single layer feedforward network

- **Single layer feedforward network** – The concept is of feedforward ANN having only one weighted layer. In other words, we can say the input layer is fully connected to the output layer.



Multilayer feedforward network

- **Multilayer feedforward network** – The concept is of feedforward ANN having more than one weighted layer. As this network has one or more layers between the input and the output layer, it is called hidden layers.

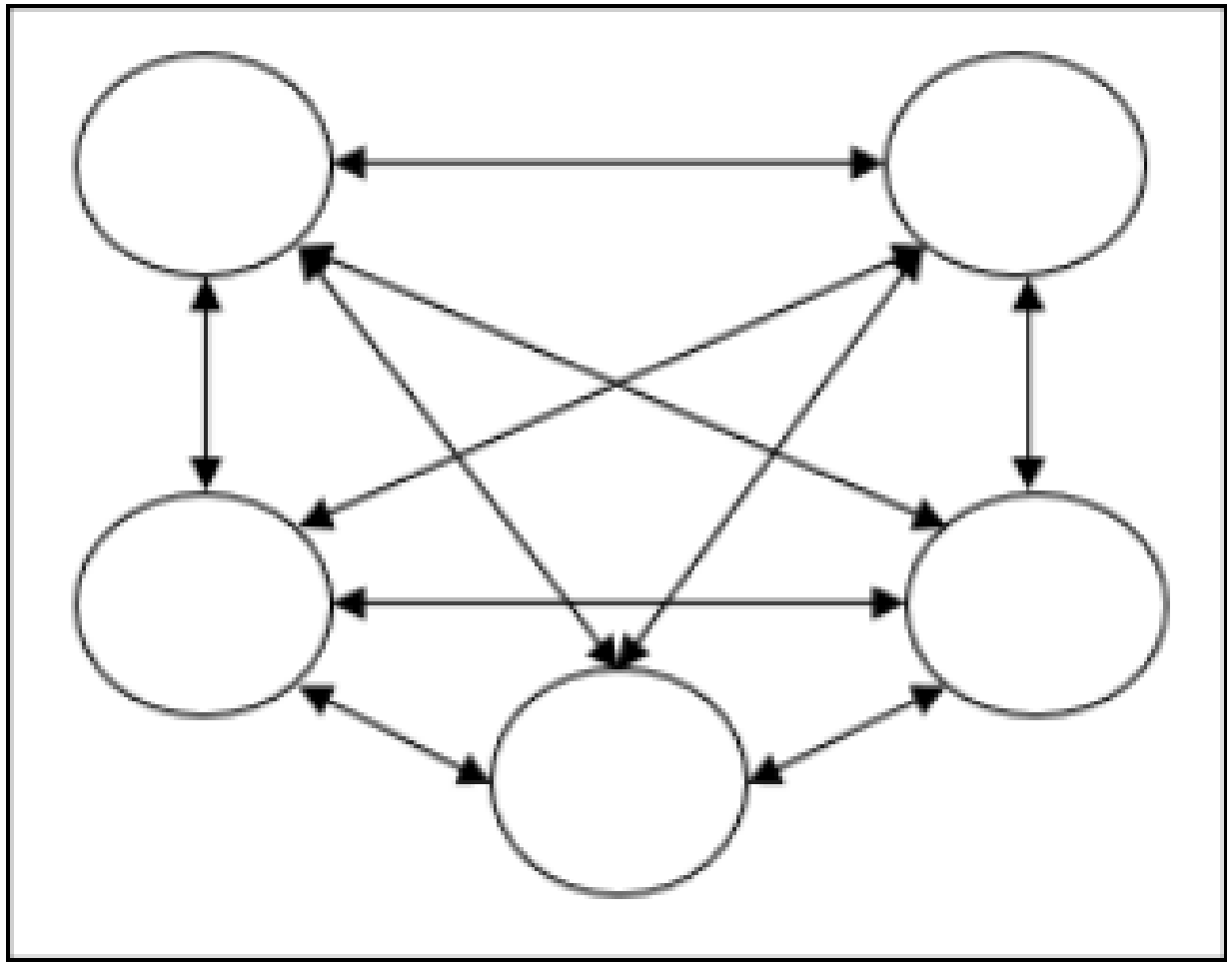


Feedback Network

As the name suggests, a feedback network has feedback paths, which means the signal can flow in both directions using loops. This makes it a non-linear dynamic system, which changes continuously until it reaches a state of equilibrium. It may be divided into the following types −

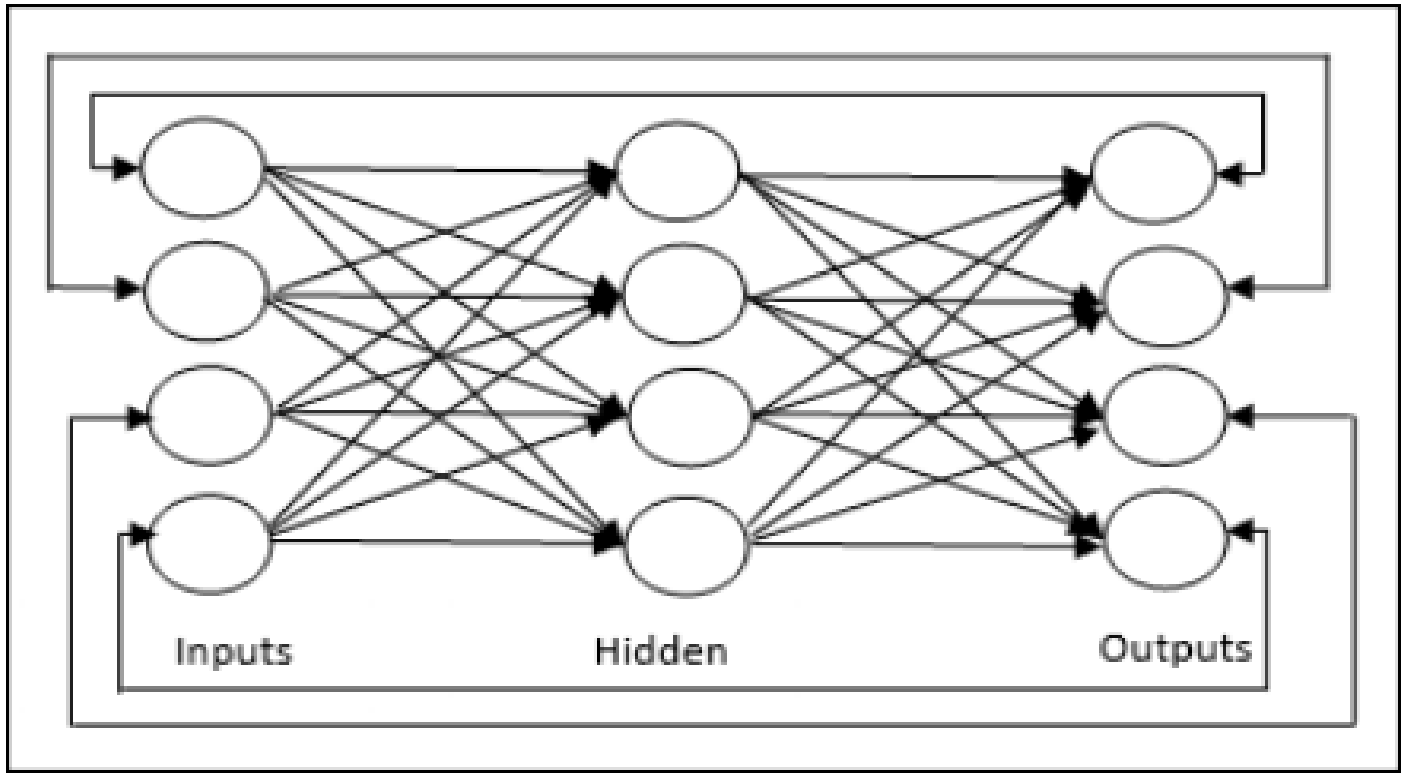
- **Recurrent networks** − They are feedback networks with closed loops. Following are the two types of recurrent networks.
- **Fully recurrent network** − It is the simplest neural network architecture because all nodes are connected to all other nodes and each node works as both input and output.

Feedback Network



Jordan network

- **Jordan network** – It is a closed loop network in which the output will go to the input again as feedback as shown in the following diagram.



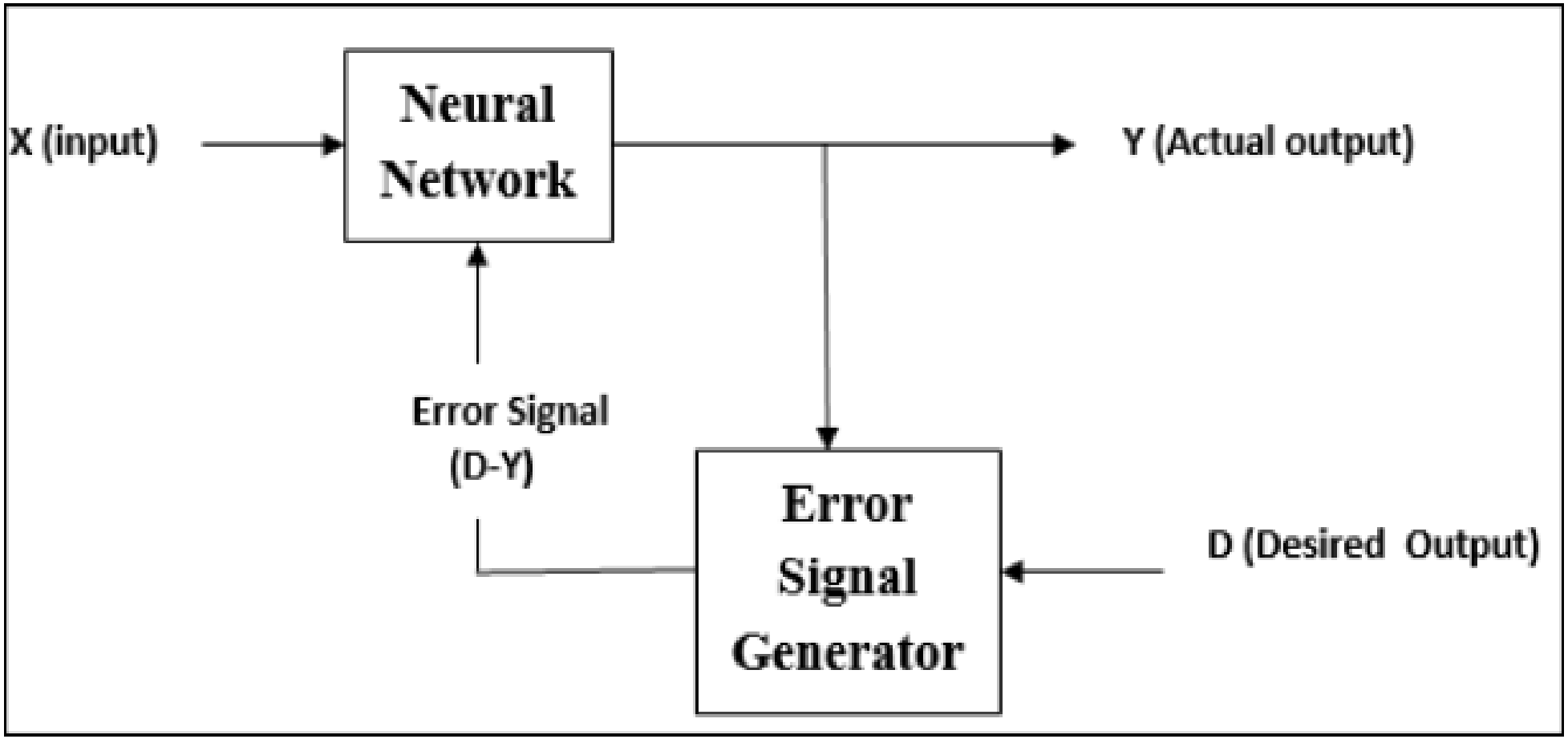
Adjustments of Weights or Learning

- Learning, in artificial neural network, is the method of modifying the weights of connections between the neurons of a specified network.
- Learning in ANN can be classified into three categories namely:
  - supervised learning
  - unsupervised learning
  - reinforcement learning

Supervised Learning

- This type of learning is done under the supervision of a teacher. This learning process is dependent.
- During the training of ANN under supervised learning:
  1. The input vector is presented to the network, which will give an output vector.
  2. This output vector is compared with the desired output vector.
  3. An error signal is generated, if there is a difference between the actual output and the desired output vector.
  4. On the basis of this error signal, the weights are adjusted until the actual output is matched with the desired output.

Supervised Learning

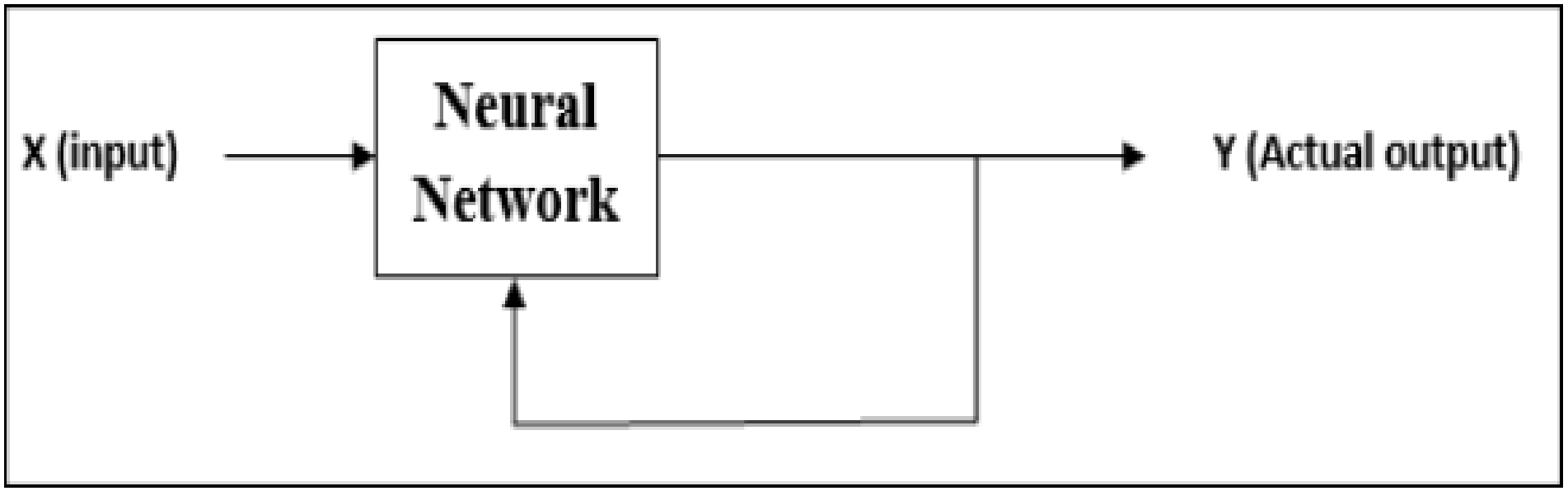


Unsupervised Learning

- This type of learning is done without the supervision of a teacher. This learning process is independent.
- During the training of ANN under unsupervised learning:
  1. The input vectors of similar type are combined to form clusters.
  2. When a new input pattern is applied, then the neural network gives an output response indicating the class to which the input pattern belongs.

Unsupervised Learning

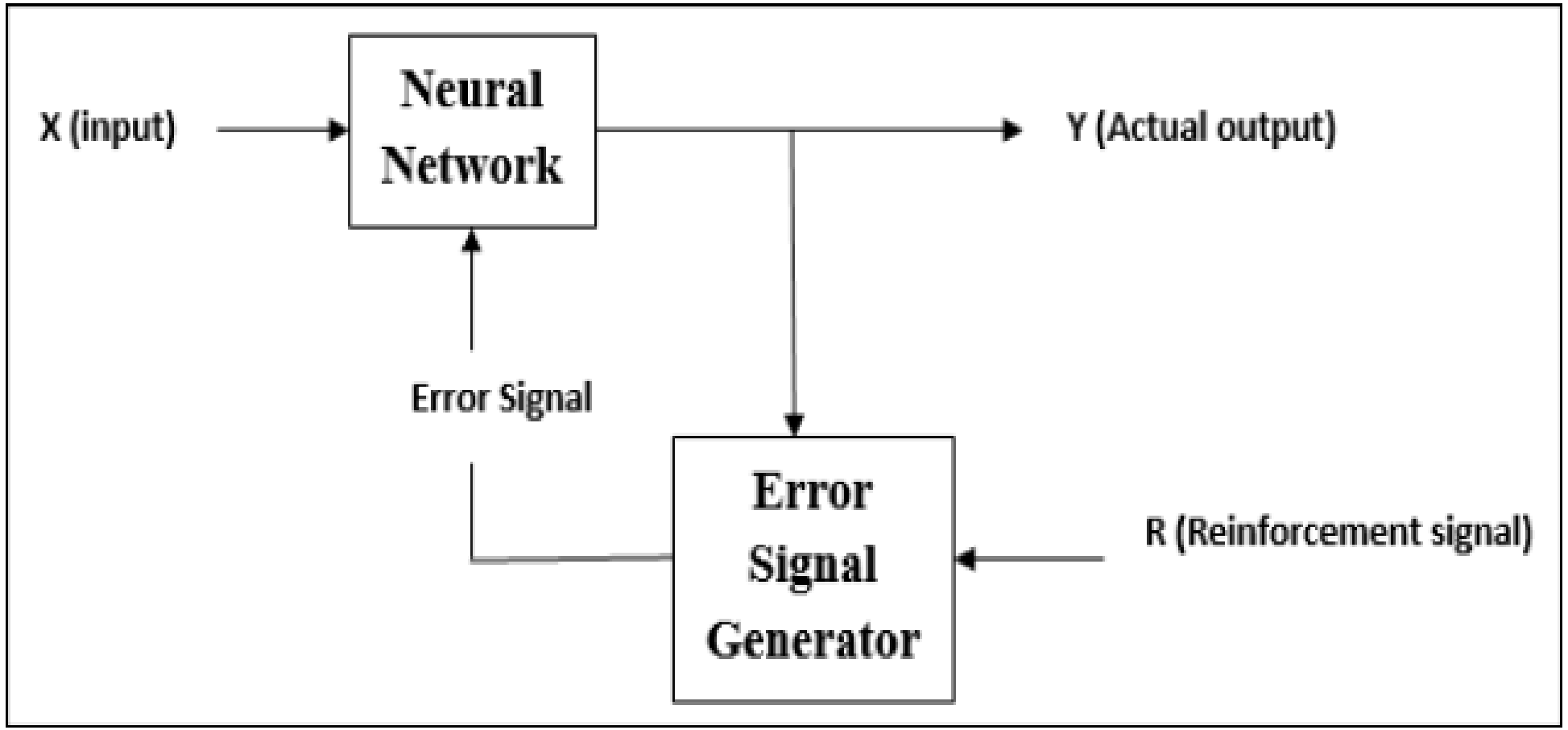
- There is no feedback from the environment as to what should be the desired output and if it is correct or incorrect.
- Hence, in this type of learning, the network itself must discover the patterns and features from the input data, and the relation for the input data over the output.



Reinforcement Learning

- This type of learning is used to reinforce or strengthen the network over some critic information. This learning process is similar to supervised learning, however we might have very less information.
- During the training of network under reinforcement learning:
  1. The network receives some feedback from the environment.
  2. This makes it somewhat similar to supervised learning. However, the feedback obtained here is evaluative not instructive, which means there is no teacher as in supervised learning.
  3. After receiving the feedback, the network performs adjustments of the weights to get better critic information in future.

Reinforcement Learning



Activation Functions

It may be defined as the extra force or effort applied over the input to obtain an exact output. In ANN, we can also apply activation functions over the input to get the exact output. Followings are some activation functions of interest −

Linear Activation Function

It is also called the identity function as it performs no input editing. It can be defined as −

$$F(x) = x$$

Activation Functions

Sigmoid Activation Function

It is of two type as follows −

- **Binary sigmoidal function** − This activation function performs input editing between 0 and 1. It is positive in nature. It is always bounded, which means its output cannot be less than 0 and more than 1. It is also strictly increasing in nature, which means more the input higher would be the output. It can be defined as

$$F(x) = sigm(x) = \frac{1}{1 + exp(-x)}$$

Activation Functions

- **Bipolar sigmoidal function** – This activation function performs input editing between -1 and 1. It can be positive or negative in nature. It is always bounded, which means its output cannot be less than -1 and more than 1. It is also strictly increasing in nature like sigmoid function. It can be defined as

$$F(x) = sigm(x) = \frac{2}{1 + exp(-x)} - 1 = \frac{1 - exp(x)}{1 + exp(x)}$$

Learning and Adaptation

- As stated earlier, ANN is completely inspired by the way biological nervous system, i.e. the human brain works.
- The most impressive characteristic of the human brain is to learn, hence the same feature is acquired by ANN.
- Basically, learning means to do and adapt the change in itself as and when there is a change in environment.
- ANN is a complex system or more precisely we can say that it is a complex adaptive system, which can change its internal structure based on the information passing through it.

Why Is learning in ANN Important?

- Being a complex adaptive system, learning in ANN implies that a processing unit is capable of changing its input/output behavior due to the change in environment.
- The importance of learning in ANN increases because of the fixed activation function as well as the input/output vector, when a particular network is constructed.
- Now to change the input/output behavior, we need to adjust the weights.

Classification

- It may be defined as the process of learning to distinguish the data of samples into different classes by finding common features between the samples of the same classes.
- For example, to perform training of ANN, we have some training samples with unique features, and to perform its testing we have some testing samples with other unique features.
- Classification is an example of supervised learning.

Neural Network Learning Rules

- We know that, during ANN learning, to change the input/output behavior, we need to adjust the weights.
- Hence, a method is required with the help of which the weights can be modified.
- These methods are called Learning rules, which are simply algorithms or equations.

Hebbian Learning Rule

This rule, one of the oldest and simplest, was introduced by Donald Hebb in his book *The Organization of Behavior* in 1949. It is a kind of feed-forward, unsupervised learning.

Basic Concept – This rule is based on a proposal given by Hebb, who wrote –

"When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased."

From the above postulate, we can conclude that the connections between two neurons might be strengthened if the neurons fire at the same time and might weaken if they fire at different times.

Hebbian Learning Rule

Mathematical Formulation – According to Hebbian learning rule, following is the formula to increase the weight of connection at every time step.

$$\Delta w_{ji}(t) = \alpha x_i(t).\, y_j(t)$$

Here, $\Delta w_{ji}(t)$ = increment by which the weight of connection increases at time step **t**

α = the positive and constant learning rate

$x_i(t)$ = the input value from pre-synaptic neuron at time step **t**

$y_i(t)$ = the output of pre-synaptic neuron at same time step **t**

Perceptron Learning Rule

This rule is an error correcting the supervised learning algorithm of single layer feedforward networks with linear activation function, introduced by Rosenblatt.

Basic Concept – As being supervised in nature, to calculate the error, there would be a comparison between the desired/target output and the actual output. If there is any difference found, then a change must be made to the weights of connection.

Perceptron Learning Rule

Mathematical Formulation – To explain its mathematical formulation, suppose we have 'n' number of finite input vectors, x n , along with its desired/target output vector t n , where n = 1 to N.

Now the output 'y' can be calculated, as explained earlier on the basis of the net input, and activation function being applied over that net input can be expressed as follows –

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

Where **θ** is threshold.

Perceptron Learning Rule

The updating of weight can be done in the following two cases −

Case I − when **t ≠ y**, then

$$w(new) = w(old) + tx$$

Case II − when **t = y**, then

No change in weight

Delta Learning Rule (Widrow–Hoff Rule)

It is introduced by Bernard Widrow and Marcian Hoff, also called Least Mean Square LMS method, to minimize the error over all training patterns. It is kind of supervised learning algorithm with having continuous activation function.

Basic Concept – The base of this rule is gradient-descent approach, which continues forever. Delta rule updates the synaptic weights so as to minimize the net input to the output unit and the target value.

Delta Learning Rule (Widrow–Hoff Rule)

Mathematical Formulation – To update the synaptic weights, delta rule is given by

$$\Delta w_i = \alpha \,.\, x_i \,.\, e_j$$

Here Δw_i = weight change for i^{th} pattern;

α = the positive and constant learning rate;

x_i = the input value from pre-synaptic neuron;

$e_j = (t - y_{in})$, the difference between the desired/target output and the actual output y_{in}

The above delta rule is for a single output unit only.

Delta Learning Rule (Widrow–Hoff Rule)

The updating of weight can be done in the following two cases –

Case-I – when **t ≠ y**, then

$$w(new) = w(old) + \Delta w$$

Case-II – when **t = y**, then

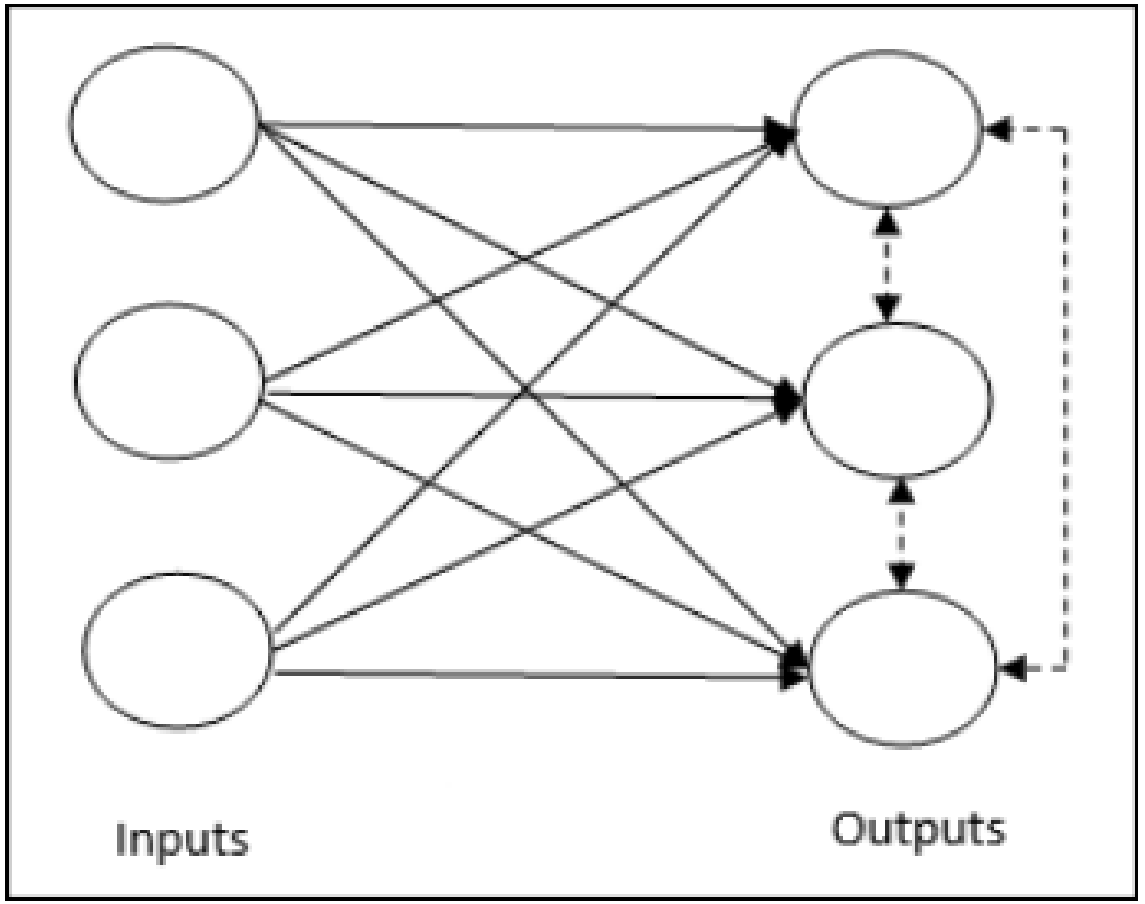
No change in weight

Competitive Learning Rule (Winner–takes–all)

It is concerned with unsupervised training in which the output nodes try to compete with each other to represent the input pattern. To understand this learning rule, we must understand the competitive network which is given as follows –

Basic Concept of Competitive Network – This network is just like a single layer feedforward network with feedback connection between outputs. The connections between outputs are inhibitory type, shown by dotted lines, which means the competitors never support themselves.

Competitive Learning Rule (Winner–takes–all)



Basic Concept of Competitive Learning Rule

- As said earlier, there will be a competition among the output nodes. Hence, the main concept is that during training, the output unit with the highest activation to a given input pattern, will be declared the winner.
- This rule is also called Winner-takes-all because only the winning neuron is updated and the rest of the neurons are left unchanged.

Basic Concept of Competitive Learning Rule

Mathematical formulation – Following are the three important factors for mathematical formulation of this learning rule –

- **Condition to be a winner** – Suppose if a neuron y_k wants to be the winner then there would be the following condition –

$$y_k = \begin{cases} 1 & if\ v_k > v_j\ for\ all\ j,\ j \neq k \\ 0 & otherwise \end{cases}$$

It means that if any neuron, say y_k , wants to win, then its induced local field $theoutputofsummationunit$, say v_k , must be the largest among all the other neurons in the network.

Basic Concept of Competitive Learning Rule

- **Condition of sum total of weight** – Another constraint over the competitive learning rule is, the sum total of weights to a particular output neuron is going to be 1. For example, if we consider neuron **k** then –

$$\sum_j w_{kj} = 1 \qquad for\ all\ k$$

Basic Concept of Competitive Learning Rule

- **Change of weight for winner** – If a neuron does not respond to the input pattern, then no learning takes place in that neuron. However, if a particular neuron wins, then the corresponding weights are adjusted as follows

$$\Delta w_{kj} = \begin{cases} -\alpha(x_j - w_{kj}), & if\ neuron\ k\ wins \\ 0, & if\ neuron\ k\ losses \end{cases}$$

Here α is the learning rate.

This clearly shows that we are favoring the winning neuron by adjusting its weight and if there is a neuron loss, then we need not bother to re-adjust its weight.

Outstar Learning Rule

This rule, introduced by Grossberg, is concerned with supervised learning because the desired outputs are known. It is also called Grossberg learning.

Basic Concept – This rule is applied over the neurons arranged in a layer. It is specially designed to produce a desired output **d** of the layer of **p** neurons.

Mathematical Formulation – The weight adjustments in this rule are computed as follows

$$\Delta w_j = \alpha (d - w_j)$$

Here **d** is the desired neuron output and α is the learning rate.