Subject Notes: Unit I

IT 701 Soft Computing

Syllabus:

Introduction to Neural Network: Concept, biological neural network, comparison of ANN with biological NN, evolution of artificial neural network, Basic models, Types of learning, linear separability, XOR problem, McCulloch-Pitts neuron model, Hebb rule.

Introduction to Neural Network: **Concept of Neural Network**

A neural network is a processing device, an algorithm or an actual hardware, whose design was inspired by the design and functioning of animal brains and components.

The computing world has a lot to gain from neural networks, also known as artificial neural networks or neural net.

The word neural network is referred to a network of biological neurons in the nervous system that process and transmit information. The neural network has the ability to learn by examples, which makes them very flexible and powerful.

Biological Neural Network

The human brain consists of a large number, more than a billion of neural cells that process information. Each cell works like a simple processor. The massive interaction between all cells and their parallel processing only makes the brain's abilities possible.

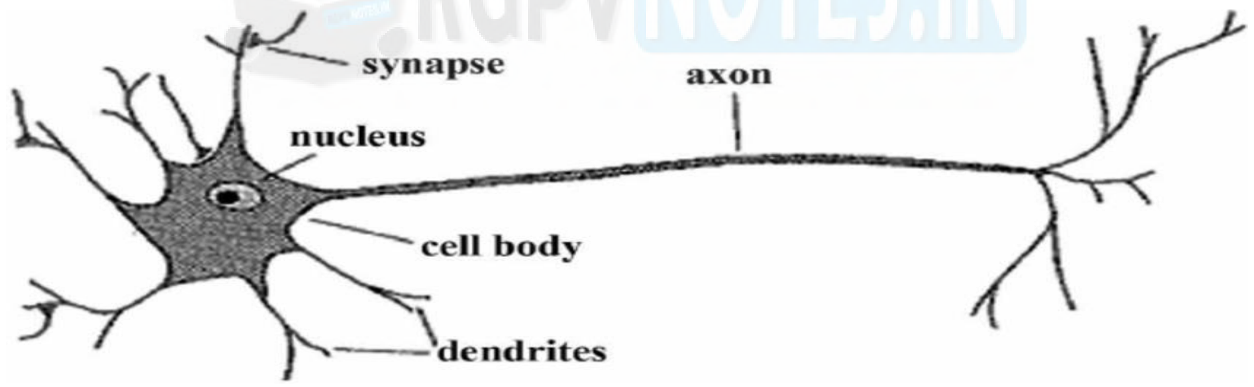

Figure 1.1: Structure of Biological Neuron

The biological neuron consists of three main parts:

Soma or Cell Body: where the cell nucleus is located

Dendrites: where the nerve is connected to the cell body

Axon: carries the impulses of the neuron

Information flow in neural cell

The input /output and the propagation of information is as give below:

- Dendrites receive activation from other neurons.
- Soma processes the incoming activations and converts them into output activations.

- Axons act as transmission lines to send activation to other neurons.
- Synapses the junctions allow signal transmission between the axons and dendrites.
- The process of transmission is by diffusion of chemicals called neuro-transmitters.

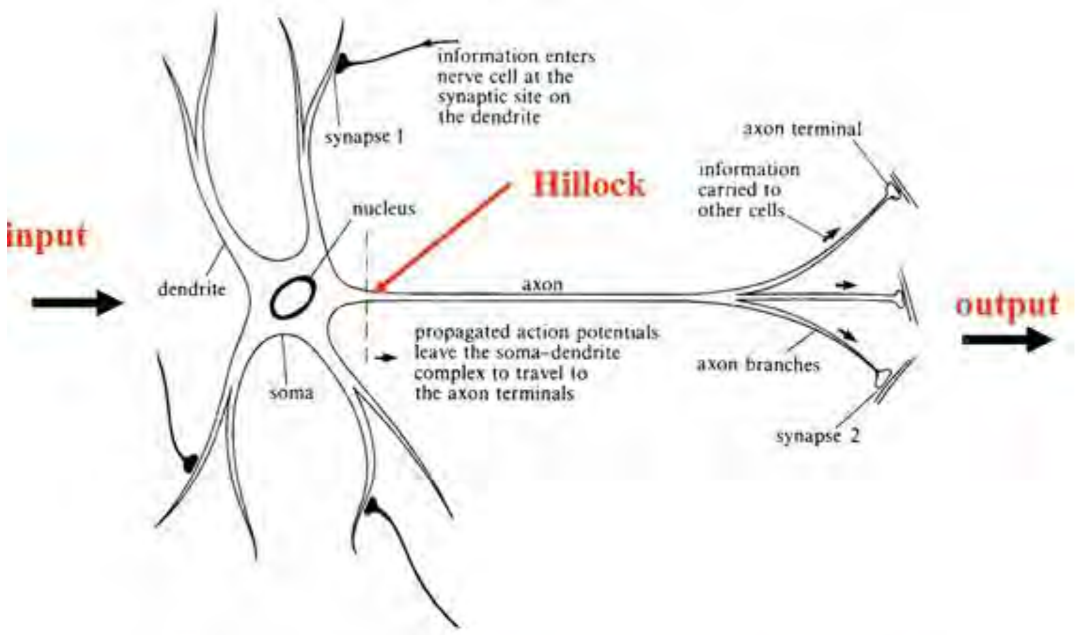
information enters nerve cell at the synaptic site on the dendrite
synapse 1
axon terminal
information carried to other cells
Hillock
nucleus
input
dendrite
axon
output
propagated action potentials leave the soma-dendrite complex to travel to the axon terminals
axon branches
soma
synapse 2

Figure 1.2: Information Flow in Neural Cell

Comparison between Biological Neuron and Artificial Neuron (Brain vs. Computer)

Particular	Brain	ANN
Speed	Few ms	Few nano second
Size & Complexity	10^{11} neurons & 10^{15} interconnections	Depends on designer
Storage Capacity	Stores information in its interconnection or in synapse. No loss of memory	Contiguous memory location. Loss of memory may happen sometimes
Tolerance	Has fault tolerance	No fault tolerance
Control Mechanism	Complicated involves chemicals in biological neuron	Simpler in ANN

Table 1.1: Brain vs ANN

Artificial Neural Network

An artificial neural network (ANN) may be defined as an information processing model that is inspired by the way biological nervous system, such as brain, process information. This model tries to replicate only the most basic functions of the brain. The key element of ANN is the novel structure of its information processing system.

An ANN is configured for a specific application, such as pattern recognition or data classification through a learning process.

Mathematical Model of Artificial Neuron

An artificial neuron is a mathematical function conceived as a simple model of a real (biological) neuron.

- A set of input connections brings in activations from other neurons.
- A processing unit sums the inputs, and then applies a non-linear activation function (i.e. squashing / transfer / threshold function).
- An output line transmits the result to other neurons.

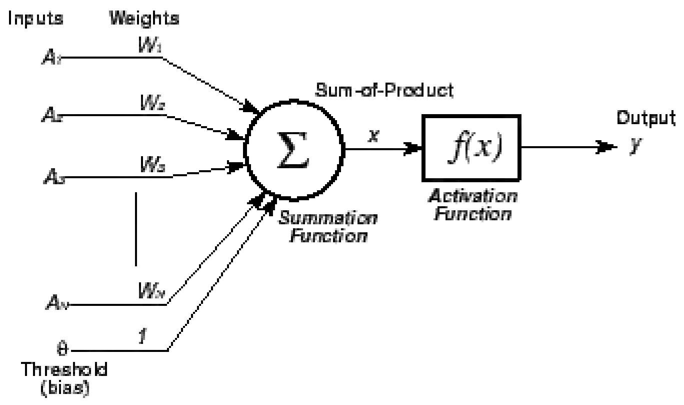

Figure 1.3: Model of Artificial Neuron

Terminology relationship between biological and artificial neurons

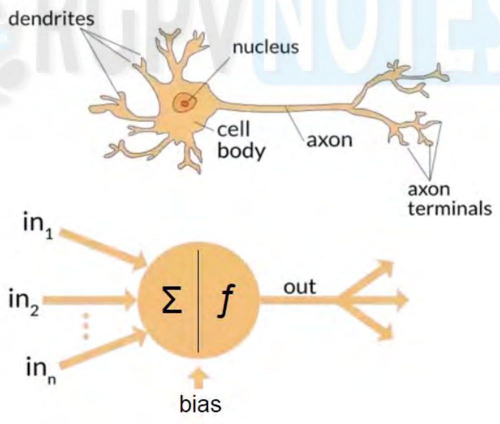

Figure 1.4: Terminology Relation

Evolution of Neural Networks

Year	**Neural Network**	**Description**
1943	McCulloch & Pitts Neuron	The arrangement of neurons in this case is a combination of logic functions. Unique feature is the concept of threshold
1949	Hebb Network	It is based upon the fact that if two neurons are found to be active simultaneously then the strength of the connection between them should be increased
1958	Perceptron	Here the weights on the connection path can be adjusted

1960	Adaline	Here the weights are adjusted to reduce the difference between the net input to the output unit and the desired output.
1972	Kohonen self-organizing feature map	The concept behind this network is that the inputs are clustered together to obtain a fired output neuron.
1982	Hopfield Network	This neural network is based on fixed weights. These nets can also act as associative memory nets.
1986	Back Propagation Network	This network is multi-layer with error being propagated backwards from the output units to be hidden units.
1988	Counter Propagation	This is similar to kohonen; here the learning occurs for all units in a particular layer, and there exists no competition among these units.
1987-90	Adaptive Resonance Theory (ART)	The ART network is designed for both binary inputs and analog valued inputs.
1988	Radial basis function	This resembles a back propagation network but the activation function used in a Gaussian function.
1988	Neo cognitron	This network is essential for character recognition.

Table 1.2: Evolution of Neural Network

McCulloch-Pitts Neuron Model

The McCulloch-Pitts model was an extremely simple artificial neuron. The inputs could be either a zero or a one. And the output was a zero or a one. And each input could be either excitatory or inhibitory. Now the whole point was to sum the inputs. If an input is one, and is excitatory in nature, it added one. If it was one, and was inhibitory, it subtracted one from the sum. This is done for all inputs, and a final sum is calculated. Now, if this final sum is less than some value (which you decide, say T), then the output is zero. Otherwise, the output is a one. Here is a graphical representation of the McCulloch-Pitts model

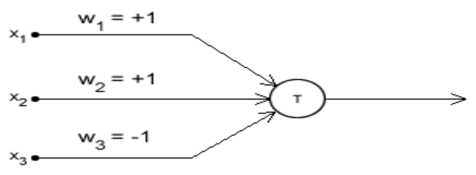

Figure 1.5: McCulloch-Pitts Neuron Model

The variables w_1, w_2 and w_3 indicate which input is excitatory, and which one is inhibitory. These are called "weights". So, in this model, if a weight is 1, it is an excitatory input. If it is -1, it is an inhibitory input.

x_1, x_2, and x_3 represent the inputs. There could be more (or less) inputs if required. And accordingly, there would be more 'w's to indicate if that particular input is excitatory or inhibitory.

Now, if you think about it, you can calculate the sum using the 'x's and 'w's... something like this:

$\text{sum} = x_1w_1 + x_2w_2 + x_3w_3 + \ldots$

This is what is called a 'weighted sum'.

Now that the sum has been calculated, we check if sum < T or not. If it is, then the output is made zero. Otherwise, it is made a one.

Learning

The main property of ANN is its capability to learn. Learning or training is a process by means of which a neural network adapts itself to a stimulus by making proper parameter adjustments, resulting in the production of desired response. Broadly there are two kinds of learning in ANN:

- Parameter Learning: It updates the connecting weights in a neural set.
- Structure Learning: It focuses on the change in network structure (which includes the number of processing elements as well as their connection types).

The above two types of learning can be performed simultaneously or separately. Apart from these two types, the learning in ANN can be generally classified into following categories:

- Supervised Learning
- Unsupervised Learning

Supervised Learning

Supervised learning is where you have input variables (x) and an output variable (Y) and you use an algorithm to learn the mapping function from the input to the output.

$$Y = f(X)$$

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data.

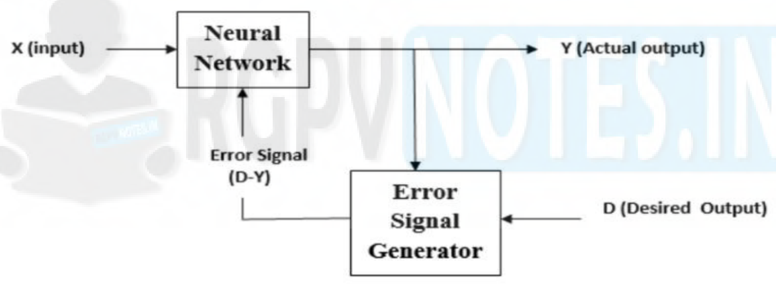

Figure 1.6: Supervised Learning

It is called supervised learning because the process of algorithm learning from the training dataset can be thought of as a teacher supervising the learning process. We know the correct answers; the algorithm iteratively makes predictions on the training data and is corrected by the teacher. Learning stops when the algorithm achieves an acceptable level of performance.

During training, the input vector is presented to the network, which results in an output vector. This output vector is the actual output vector. Then the actual output vector is compared with the desired (target) output vector. If there exists a difference between the two output vectors then an error signal is generated by the network. This error is used for adjustment of weights until the actual output matches the desired (target) output.
In this type of learning, a supervisor or teacher is required for error minimization. Hence, the network trained by this method is said to be using supervised training methodology. In supervised learning it is assumed that the correct “target” output values are known for each input pattern.

Unsupervised Learning

Unsupervised learning is where you only have input data (X) and no corresponding output variables.

The goal for unsupervised learning is to model the underlying structure or distribution in the data in order to learn more about the data.

In ANNs following unsupervised learning, the input vectors of similar type are grouped without the use of training data to specify how a member of each group looks or to which group a number belongs. In the training process, the network receives the input patterns and organizes these patterns to form clusters. When a new input pattern is applied, the neural network gives an output response indicating the class to which the input pattern belongs. If for an input, a pattern class cannot be found then a new class is generated.

These are called unsupervised learning because unlike supervised learning above there is no correct answer and there is no teacher. Algorithms are left to their own devises to discover and present the interesting structure in the data.

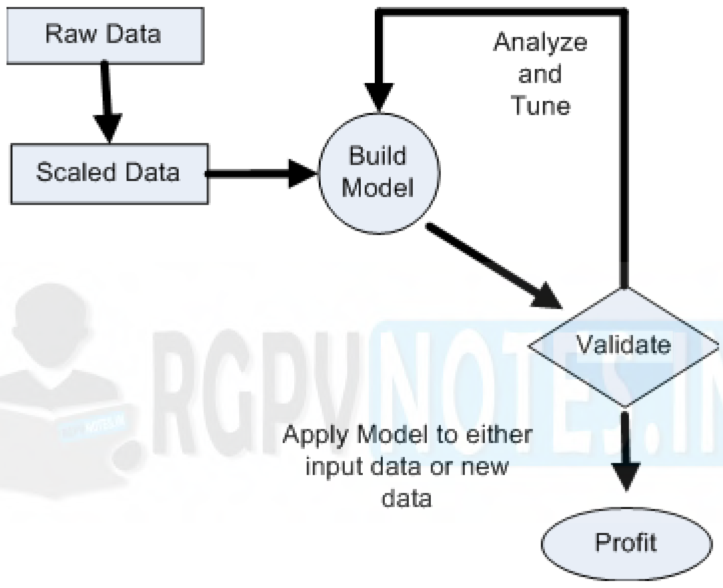

Figure 1.7: Unsupervised Learning

From the working of unsupervised learning it is clear that there is no feedback from the environment to inform what the outputs should be or whether the outputs are correct. In this case the network must itself discover patterns, regularities, features or categories from the input data and relations for the input data over the output.

Linear separability and XOR Problem

Linear separability is a powerful technique which is used to learn complicated concepts that are considerably more complicated than just hyperplane separation.

Let *f* be the XOR function which expects two binary inputs and generates a binary output .Let us try to represent the XOR function by means of an SLP with two input neurons i_1, i_2 and one output neuron:

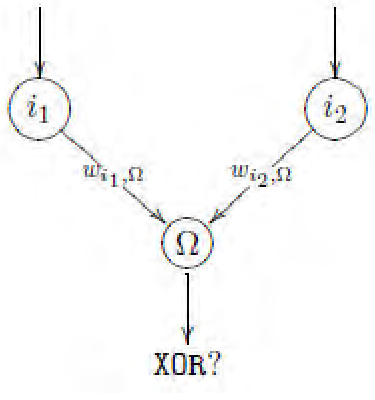

Figure 1.8: Single layer perceptron

Here we use the weighted sum as propagation function, a binary activation function with the threshold value and the identity as output function. Depending on i_1 and i_2 the output becomes the value 1 if the following holds:

$$\text{net}_\Omega = o_{i_1} w_{i_1,\Omega} + o_{i_2} w_{i_2,\Omega} \geq \Theta_\Omega \quad \text{----- (1)}$$

We assume a positive weight w_{i2}, Ω the inequality is then equivalent to:

$$o_{i_1} \geq \frac{1}{w_{i_1,\Omega}} (\Theta_\Omega - o_{i_2} w_{i_2,\Omega}) \quad \text{-------- (2)}$$

With a constant threshold value, the right part of in equation 2 is a straight line through a coordinate system defined by the possible outputs o_{i1} und o_{i2} of the input neurons i_1 and i_2.

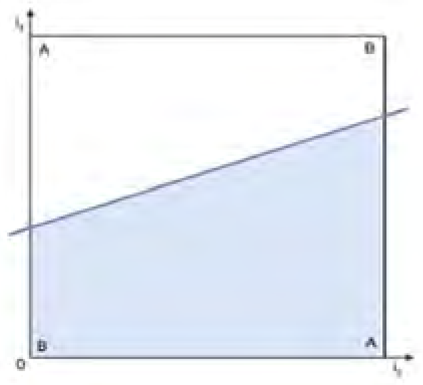

Figure 1.9: Linear separation of n=2 inputs of the input neuron i1 and i2 by 1-D line, A and B show the corner belonging to sets of XOR function that are to be separated

For a positive w_{i2},Ω the output neuron fires for input combinations lying above the generated straight line. For a negative w_{i2},Ω it would fire for all input combinations lying below the straight line. Note that only the four corners of the unit square are possible inputs because the XOR function only knows binary inputs.

In order to solve the XOR problem, we have to turn and move the straight line so that input set *A* = {(0*,* 0)*,* (1*,* 1)} is separated from input set *B* = {(0*,* 1)*,* (1*,* 0)} this is, obviously, impossible. Generally, the input parameters of *n* man input neurons can be represented in an *n* dimensional cube which is separated by an SLP through an (*n*−1)-dimensional hyper plane. Only sets that can be separated by such a hyper plane, i.e. which are ***linearly separable***, can be classified by an SLP.

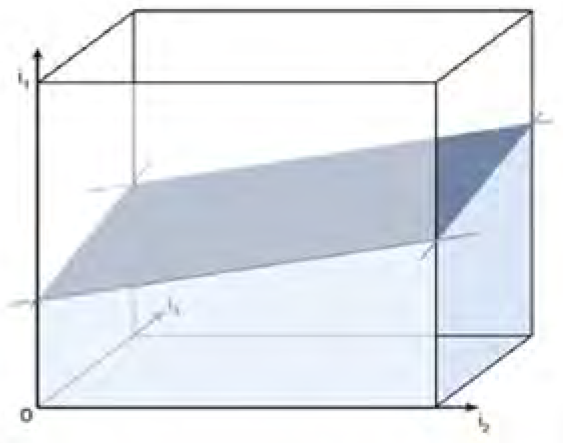

Figure 1.10: Linear separation of n=3 inputs of the input neuron i1, i2 and i3 by 2-D plane

Unfortunately, it seems that the percentage of the linearly separable problems rapidly decreases with increasing, which limits the functionality of the SLP. Additionally, tests for linear separability are difficult. Thus, for more difficult tasks with more inputs we need something more powerful than SLP. The XOR problem itself is one of these tasks, since a perceptron that is supposed to represent the XOR function already needs a hidden layer.

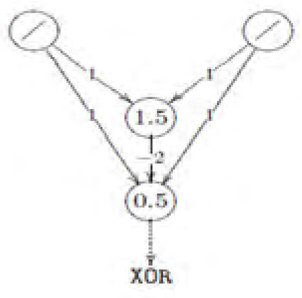

Figure 1.11: Neural network realizing the XOR Function

Activation Function

Let us assume a person is performing some task. To make the task more efficient and to obtain exact output, some force or activation may be given. This activation helps in achieving the exact

output. In the similar way, the activation function is applied over the net input to calculate the output of ANN.
It's just a thing (node) that you add to the output end of any neural network. It is also known as Transfer Function. It can also be attached in between two Neural Networks.

The information processing of a processing element can be viewed as consisting of two major parts: input and output. An integration function (f) is associated with the input of a processing element. This function serves to combine activation, information or evidence from an external source or other processing elements into a net input to the processing element.

Types of Activation Function:

- Sigmoid or Logistic
- Tanh — Hyperbolic tangent
- ReLu -Rectified linear units

Sigmoid Activation function:
It is an activation function of form f(x) = 1 / 1 + exp(-x). Its Range is between 0 and 1. It is a S — shaped curve. It is easy to understand and apply but it has major reasons which have made it fall out of popularity -

- Vanishing gradient problem
- Secondly, its output isn't zero centered. It makes the gradient updates go too far in different directions. 0 < output < 1, and it makes optimization harder.
- Sigmoids saturate and kill gradients.
- Sigmoids have slow convergence.

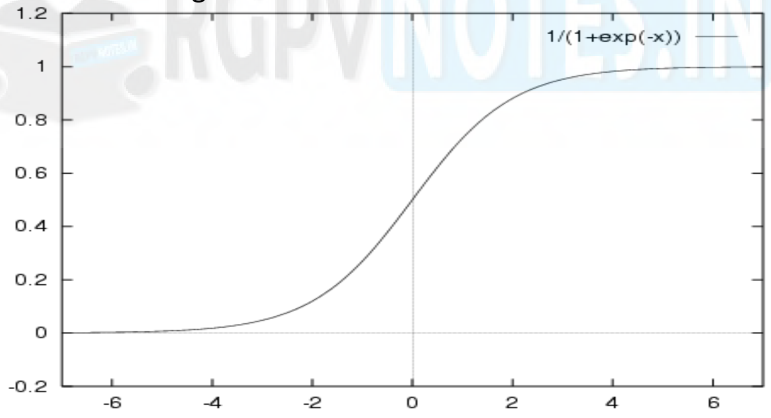

Figure 1.12: Sigmoid Activation Function

Hyperbolic Tangent Function- Tanh:
Its mathematical formula is f(x) = 1 — exp (-2x) / 1 + exp (-2x). Now its output is zero centered because its range in between -1 to 1 i.e -1 < output < 1. Hence optimization is easier in this method hence in practice it is always preferred over Sigmoid function. But still it suffers from Vanishing gradient problem.

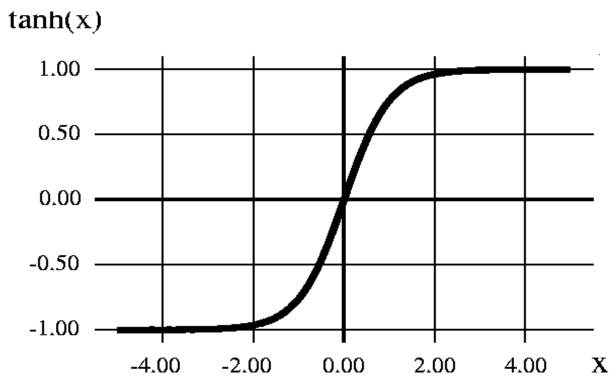

Figure 1.13: Hyperbolic Tangent Function

ReLu- Rectified Linear Units:

It has become very popular in the past couple of years. It was recently proved that it had 6 times improvement in convergence from Tanh function. It's just R(x) = max(0,x) i.e. if x < 0 , R(x) = 0 and if x >= 0 , R(x) = x. Hence as seeing the mathematical form of this function we can see that it is very simple and efficient. It avoids and rectifies vanishing gradient problem. Almost all deep learning Models use ReLu nowadays.

Models of ANN

Feed Forward Network

A feed forward neural network is an artificial neural network wherein connections between the units do not form a cycle. As such, it is different from recurrent neural networks.

The feed forward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes.

Single-Layer Feed Froward Network

The Single Layer Feed-forward Network consists of a single layer of weights, where the inputs are directly connected to the outputs, via a series of weights. The synaptic links carrying weights connect every input to every output, but no other way. The sum of the products of the weights and the inputs is calculated in each neuron node, and if the value is above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1).

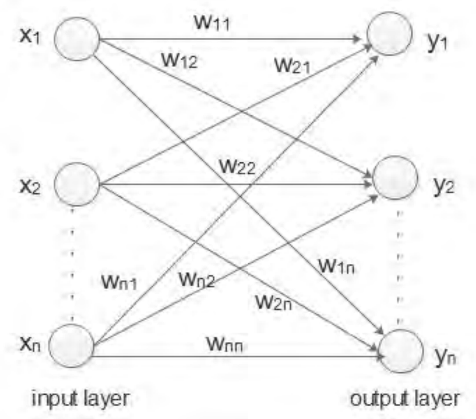

Figure 1.14: Single Layer Feed Forward Network

Multi-Layer Feed Froward Network

The name suggests, it consists of multiple layers. The architecture of this class of network, besides having the input and the output layers, also have one or more intermediary layers called hidden layers. The computational units of hidden layer are known as hidden neuron.

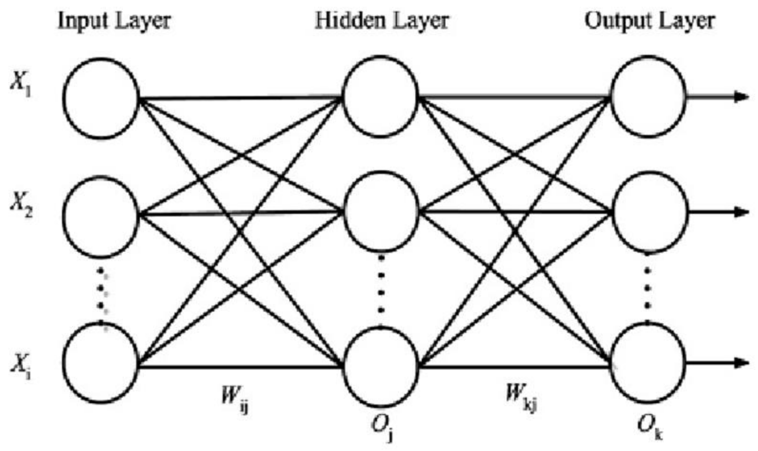

Figure 1.15: Multi-Layer Feed Forward Network

Learning Rule

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.

Learning rule or Learning process is a method or a mathematical logic. It improves the Artificial Neural Network's performance and applies this rule over the network. Thus learning rules updates the weights and bias levels of a network when a network simulates in a specific data environment.

Applying learning rule is an iterative process. It helps a neural network to learn from the existing conditions and improve its performance.

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.

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

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.

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.

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

Case-I − when $t \neq y$, then
$w(new)=w(old)+\Delta w$

Case-II − when $t = y$, then
No change in weight