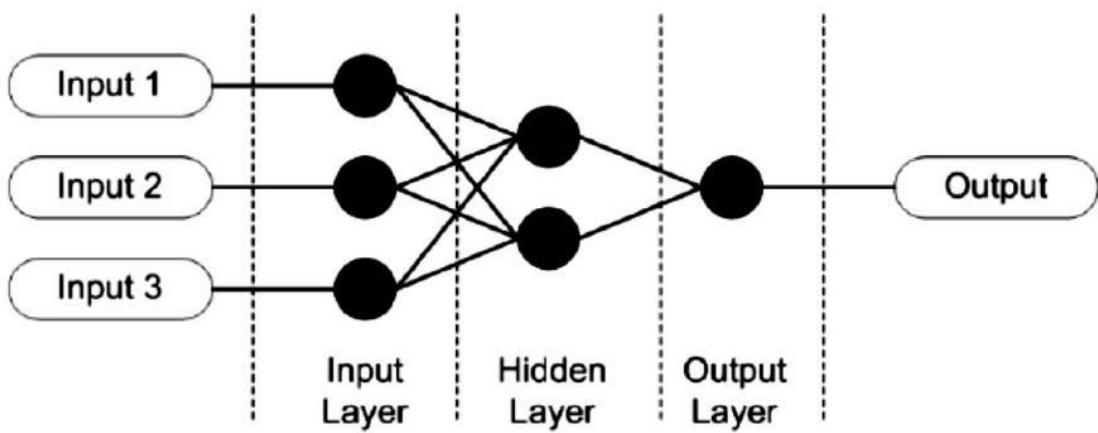
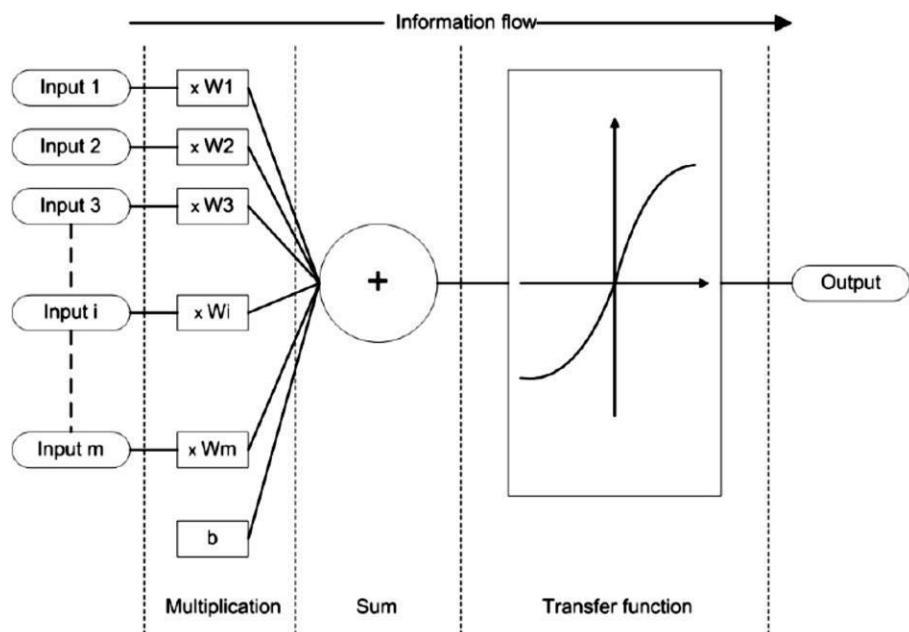


UNIT-1

NEURAL NETWORKS-1

WHAT IS ARTIFICIAL NEURAL NETWORK?

An Artificial Neural Network (ANN) is a mathematical model that tries to simulate the structure and functionalities of biological neural networks. Basic building block of every artificial neural network is artificial neuron, that is, a simple mathematical model (function). Such a model has three simple sets of rules: multiplication, summation and activation. At the entrance of artificial neuron the inputs are weighted what means that every input value is multiplied with individual weight. In the middle section of artificial neuron is sum function that sums all weighted inputs and bias. At the exit of artificial neuron the sum of previously weighted inputs and bias is passing through activation function that is also called transfer function.



Single neuron

BIOLOGICAL NEURON STRUCTURE AND FUNCTIONS.

A neuron, or nerve cell, is an electrically excitable cell that communicates with other cells via specialized connections called synapses. It is the main component of nervous tissue. Neurons are typically classified into three types based on their function. Sensory neurons respond to stimuli such as touch, sound, or light that affect the cells of the sensory organs, and they send signals to the spinal cord or brain. Motor neurons receive signals from the brain and spinal cord to control everything from muscle contractions to glandular output. Interneurons connect neurons to other neurons within the same region of the brain or spinal cord. A group of connected neurons is called a neural circuit.

A typical neuron consists of a cell body (soma), dendrites, and a single axon. The soma is usually compact. The axon and dendrites are filaments that extrude from it. Dendrites typically branch profusely and extend a few hundred micrometers from the soma. The axon leaves the soma at a swelling called the axon hillock, and travels for as far as 1 meter in humans or more in other species. It branches but usually maintains a constant diameter. At the farthest tip of the axon's branches are axon terminals, where the neuron can transmit a signal across the synapse to another cell. Neurons may lack dendrites or have no axon. The term neurite is used to describe either a dendrite or an axon, particularly when the cell is undifferentiated.

The soma is the body of the neuron. As it contains the nucleus, most protein synthesis occurs here. The nucleus can range from 3 to 18 micrometers in diameter.

The dendrites of a neuron are cellular extensions with many branches. This overall shape and structure is referred to metaphorically as a dendritic tree. This is where the majority of input to the neuron occurs via the dendritic spine.

The axon is a finer, cable-like projection that can extend tens, hundreds, or even tens of thousands of times the diameter of the soma in length. The axon primarily carries nerve signals away from the soma, and carries some types of information back to it. Many neurons have only one axon, but this axon may—and usually will—undergo extensive branching, enabling communication with many target cells. The part of the axon where it emerges from the soma is called the axon hillock. Besides being an anatomical structure, the axon hillock also has the greatest density of voltage-dependent sodium channels. This makes it the most easily excited part of the neuron and the spike initiation zone for the axon. In electrophysiological terms, it has the most negative threshold potential.

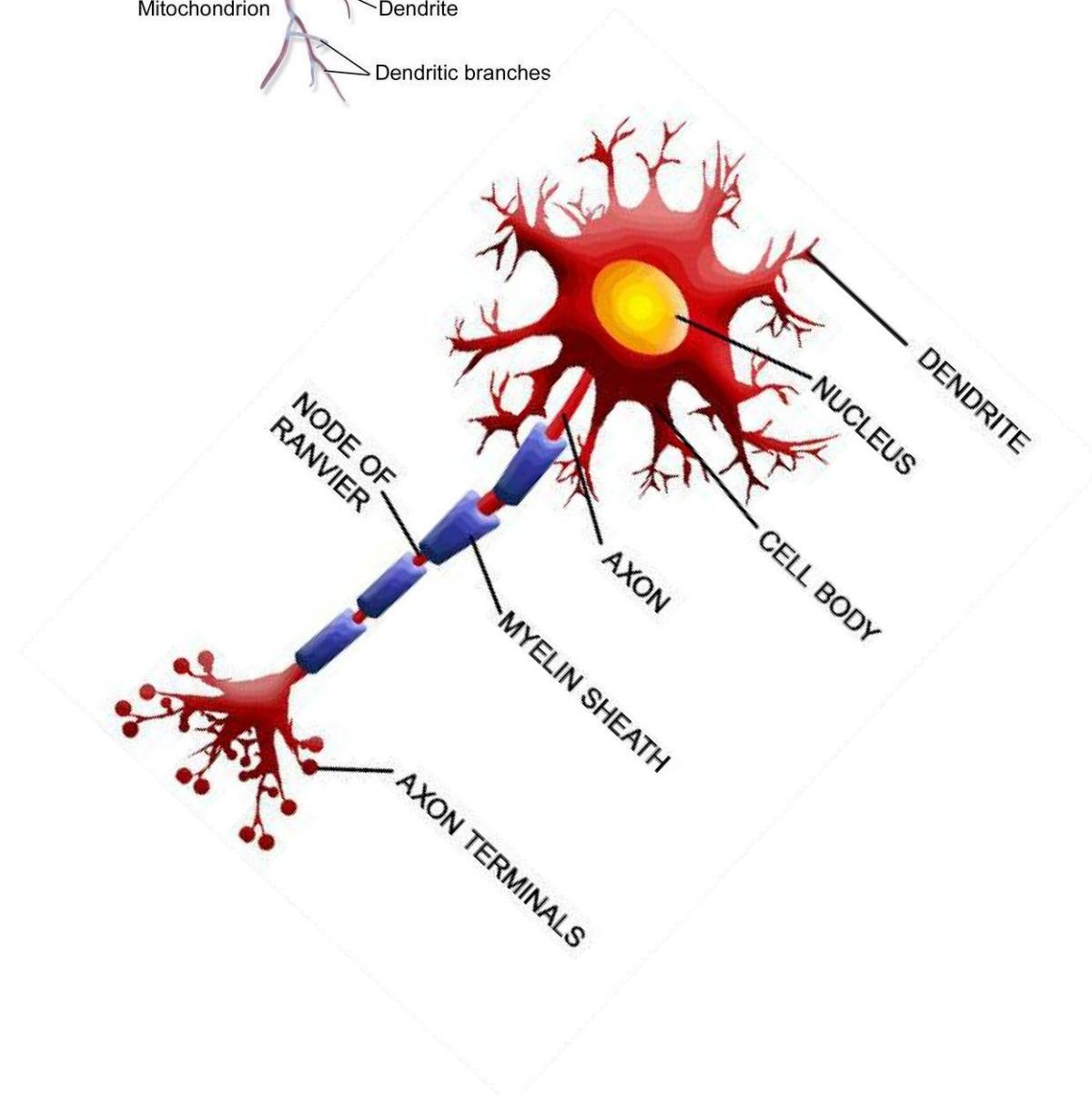
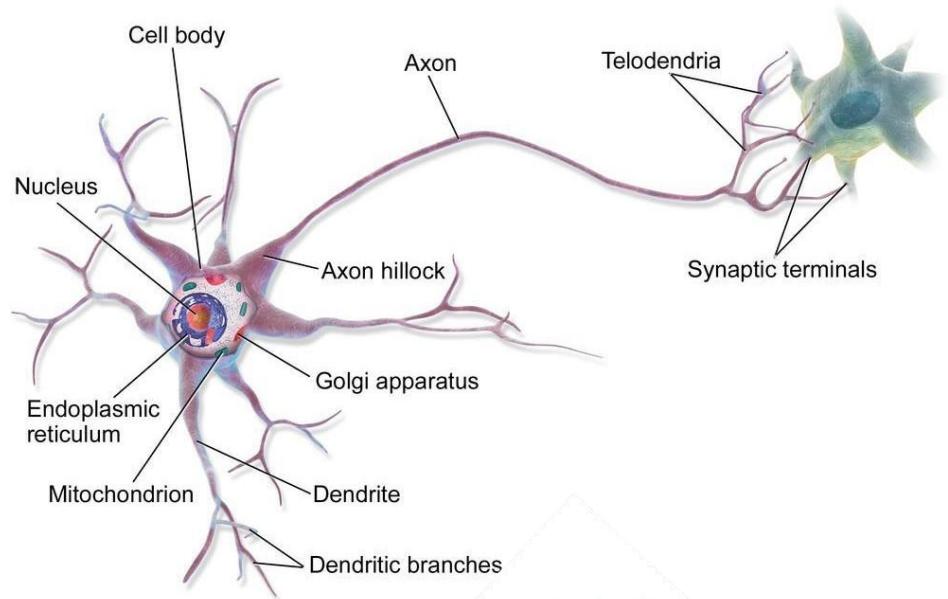
While the axon and axon hillock are generally involved in information outflow, this region can also receive input from other neurons.

The axon terminal is found at the end of the axon farthest from the soma and contains synapses. Synaptic boutons are specialized structures where neurotransmitter chemicals are released to communicate with target neurons. In addition to synaptic boutons at the axon terminal, a neuron may have en passant boutons, which are located along the length of the axon.

Most neurons receive signals via the dendrites and soma and send out signals down the axon. At the majority of synapses, signals cross from the axon of one neuron to a dendrite of another. However, synapses can connect an axon to another axon or a dendrite to another dendrite. The signaling process is partly electrical and partly chemical. Neurons are electrically excitable, due to maintenance of voltage gradients across their membranes. If the voltage changes by a large amount over a short interval, the neuron generates an all-or-nothing electrochemical pulse called an action potential. This potential travels

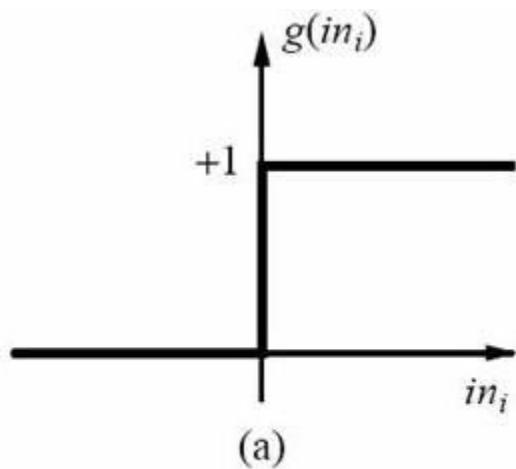
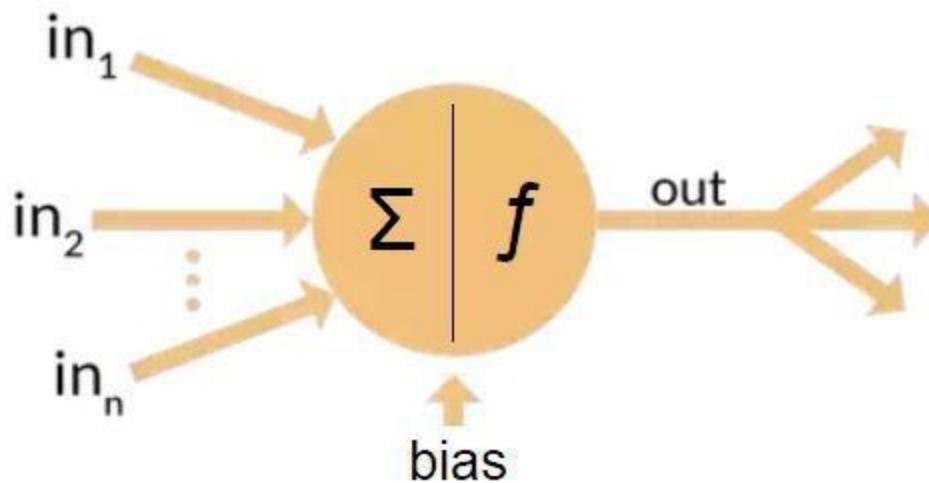
rapidly along the axon, and activates synaptic connections as it reaches them. Synaptic signals may be excitatory or inhibitory, increasing or reducing the net voltage that reaches the soma.

In most cases, neurons are generated by neural stem cells during brain development and childhood. Neurogenesis largely ceases during adulthood in most areas of the brain. However, strong evidence supports generation of substantial numbers of new neurons in the hippocampus and olfactory bulb.

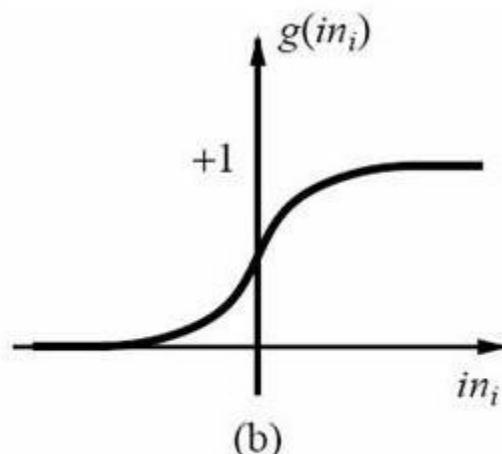


STRUCTURE AND FUNCTIONS OF ARTIFICIAL NEURON.

An artificial neuron is a mathematical function conceived as a model of biological neurons, a neural network. Artificial neurons are elementary units in an artificial neural network. The artificial neuron receives one or more inputs (representing excitatory postsynaptic potentials and inhibitory postsynaptic potentials at neural dendrites) and sums them to produce an output (or activation, representing a neuron's action potential which is transmitted along its axon). Usually each input is separately weighted, and the sum is passed through a non-linear function known as an activation function or transfer function. The transfer functions usually have a sigmoid shape, but they may also take the form of other non-linear functions, piecewise linear functions, or step functions. They are also often monotonically increasing, continuous, differentiable and bounded. The thresholding function has inspired building logic gates referred to as threshold logic; applicable to building logic circuits resembling brain processing. For example, new devices such as memristors have been extensively used to develop such logic in recent times.



step function



sigmoid function

STATE THE MAJOR DIFFERENCES BETWEEN BIOLOGICAL AND ARTIFICIANEURAL NETWORKS

- 1. Size:** Our brain contains about 86 billion neurons and more than a 100 synapses (connections). The number of “neurons” in artificial networks is much less than that.
- 2. Signal transport and processing:** The human brain works asynchronously, ANNs work synchronously.
- 3. Processing speed:** Single biological neurons are slow, while standard neurons in ANNs are fast.
- 4. Topology:** Biological neural networks have complicated topologies, while ANNs are often in a tree structure.
- 5. Speed:** certain biological neurons can fire around 200 times a second on average. Signals travel at different speeds depending on the type of the nerve impulse, ranging from 0.61 m/s up to 119 m/s. Signal travel speeds also vary from person to person depending on their sex, age, height, temperature, medical condition, lack of sleep etc. Information in artificial neurons is carried over by the continuous, floating point number values of synaptic weights. There are no refractory periods for artificial neural networks (periods while it is impossible to send another action potential, due to the sodium channels being lock shut) and artificial neurons do not experience “fatigue”: they are functions that can be calculated as many times and as fast as the computer architecture would allow.
- 6. Fault-tolerance:** biological neuron networks due to their topology are also fault-tolerant. Artificial neural networks are not modeled for fault tolerance or self regeneration (similarly to fatigue, these ideas are not applicable to matrix operations), though recovery is possible by saving the current state (weight values) of the model and continuing the training from that save state.
- 7. Power consumption:** the brain consumes about 20% of all the human body's energy — despite it's large cut, an adult brain operates on about 20 watts (barely enough to dimly light a bulb) being extremely efficient. Taking into account how humans can still operate for a while, when only given some c-vitamin rich lemon juice and beef tallow, this is quite remarkable. For benchmark: a single Nvidia GeForce Titan X GPU runs on 250 watts alone, and requires a power supply. Our machines are way less efficient than biological systems. Computers also generate a lot of heat when used, with consumer GPUs operating safely between 50–80°Celsius instead of 36.5–37.5 °C.
- 8. Learning:** we still do not understand how brains learn, or how redundant connections store and recall information. By learning, we are building on information that is already stored in the brain. Our knowledge deepens by repetition and during sleep, and tasks that once required a focus can be executed automatically once mastered. Artificial neural networks in the other hand, have a predefined model, where no further neurons or connections can be added or removed. Only the weights of the connections (and biases representing thresholds) can change during training. The networks start with random weight values and will slowly try to reach a point where further changes in the weights would no longer improve performance. Biological networks usually don't stop / start learning. ANNs have different fitting (train) and prediction (evaluate) phases.
- 9. Field of application:** ANNs are specialized. They can perform one task. They might be perfect at playing chess, but they fail at playing go (or vice versa). Biological neural networks can learn completely new tasks.
- 10. Training algorithm:** ANNs use Gradient Descent for learning. Human brains use something different (but we don't know what).

BRIEFLY EXPLAIN THE BASIC BUILDING BLOCKS OF ARTIFICIAL NEURAL NETWORKS.

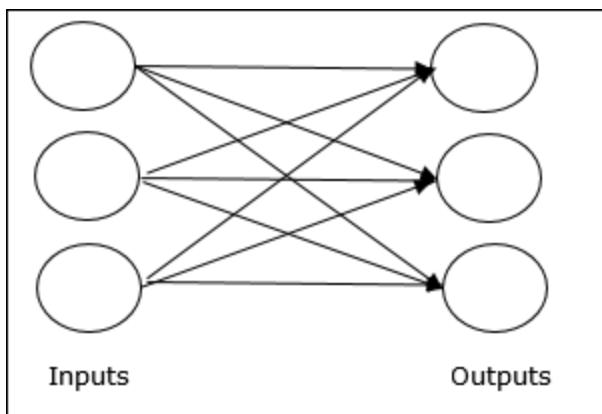
Processing of ANN depends upon the following three building blocks:

1. Network Topology
2. Adjustments of Weights or Learning
3. Activation Functions

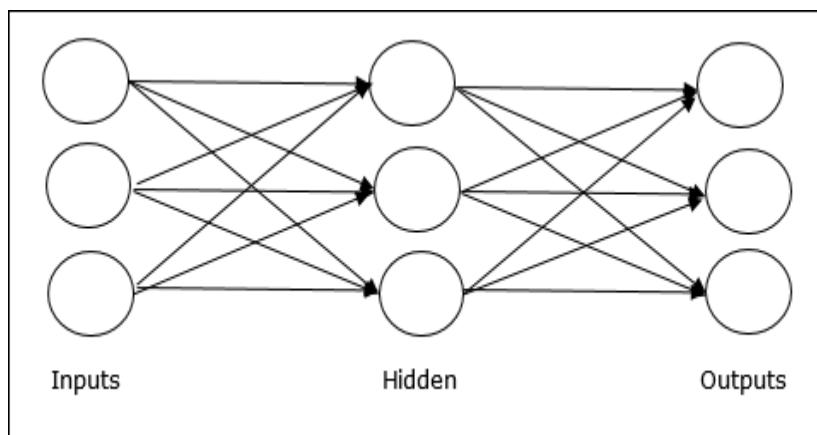
1. Network Topology: A network topology is the arrangement of a network along with its nodes and connecting lines. According to the topology, ANN can be classified as the following kinds:

A. Feed forward 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 feed forward network:** The concept is of feed forward ANN having only one weighted layer. In other words, we can say the input layer is fully connected to the output layer.

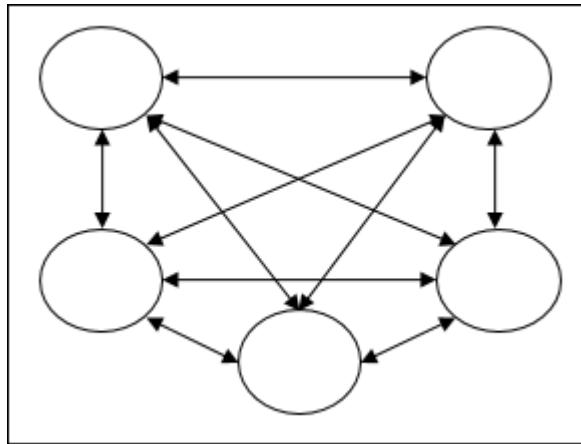


- **Multilayer feed forward network:** The concept is of feed forward ANN having more than one weighted layer. As this network has one or more layers between the input and the output layer, it is called hidden layers.

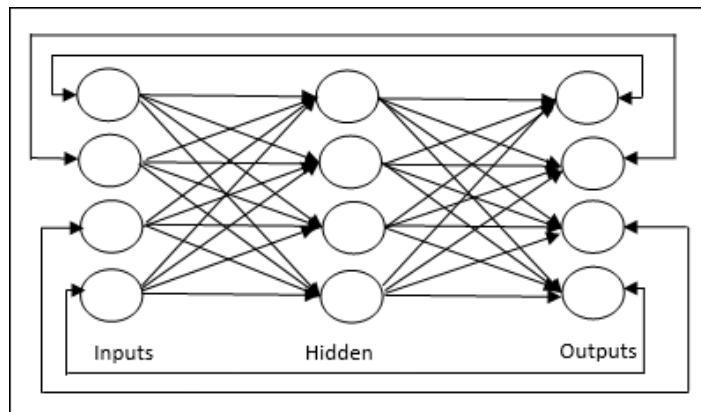


B. Feedback Network: As the name suggests, a feedback network has feedback paths, which means the signal can flow in both directions using loops. This makes it a non-linear dynamic system, which changes continuously until it reaches a state of equilibrium. It may be divided into the following types:

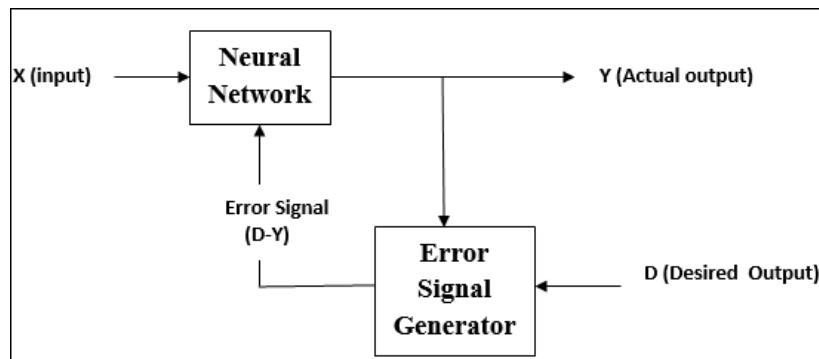
- **Recurrent networks:** They are feedback networks with closed loops. Following are the two types of recurrent networks.
- **Fully recurrent network:** It is the simplest neural network architecture because all nodes are connected to all other nodes and each node works as both input and output.



- **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.

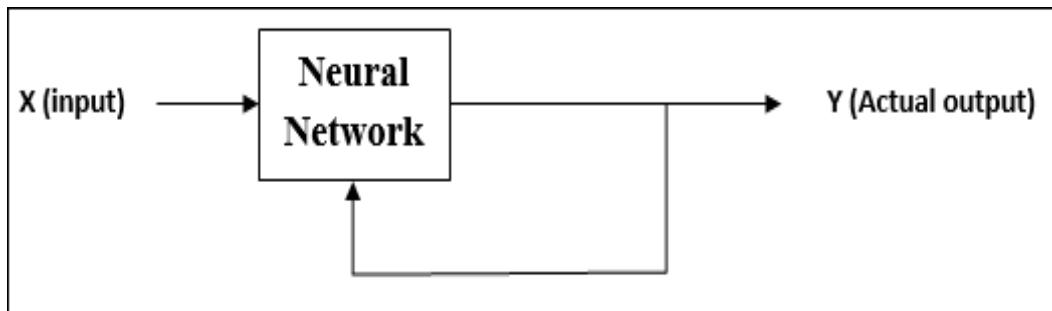


2. Adjustments of Weights or Learning: Learning, in artificial neural network, is the method of modifying the weights of connections between the neurons of a specified network. Learning in ANN can be classified into three categories namely supervised learning, unsupervised learning, and reinforcement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

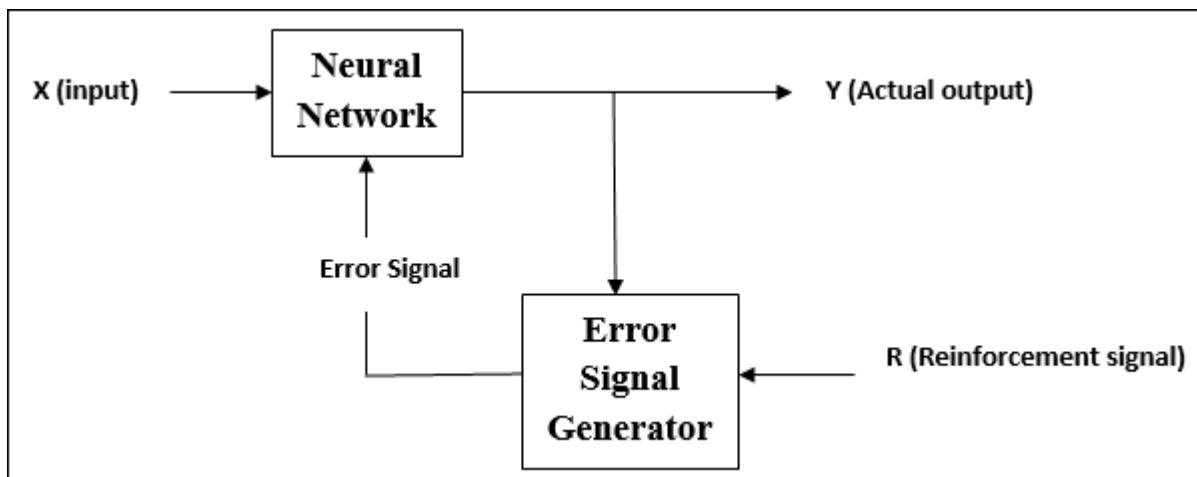


Supervised Learning: As the name suggests, this type of learning is done under the supervision of a teacher. This learning process is dependent. During the training of ANN under supervised learning, the input vector is presented to the network, which will give an output vector. This output vector is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

Unsupervised Learning: As the name suggests, 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, the input vectors of similar type are combined to form clusters. When a new input pattern is applied, then the neural network gives an output response indicating the class to which the input pattern belongs. 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: As the name suggests, 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, the network receives some feedback from the environment. 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. After receiving the feedback, the network performs adjustments of the weights to get better critic information in future.



3. **Activation Functions:** An activation function is a mathematical equation that determines the output of each element (perceptron or neuron) in the neural network. It takes in the input from each neuron and transforms it into an output, usually between one and zero or between -1 and one. 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:

i) Linear Activation Function: It is also called the identity function as it performs no input editing. It can be defined as: $F(x) = x$

ii) 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

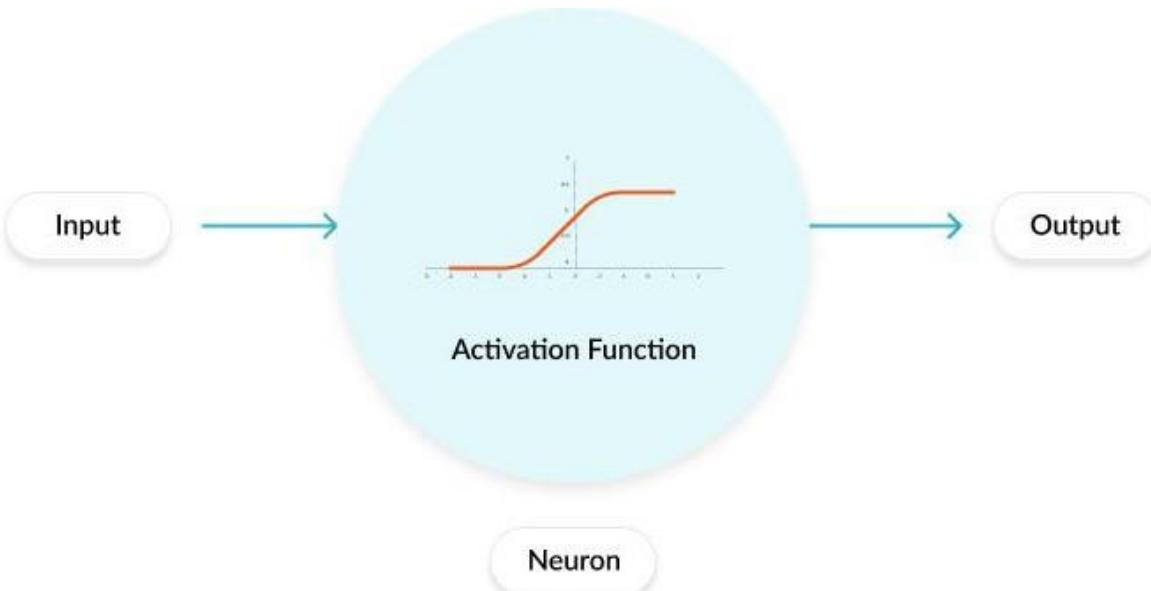
$$F(x) = \text{sigm}(x) = \frac{1}{1 + \exp(-x)}$$

- **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

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

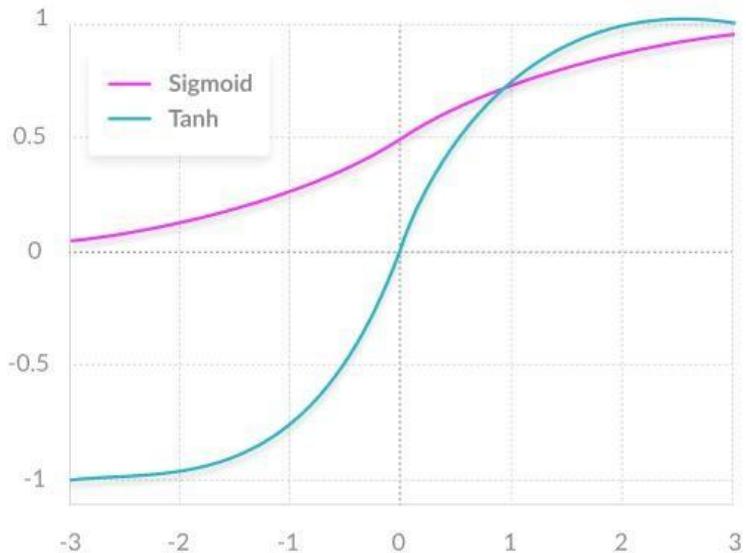
WHAT IS A NEURAL NETWORK ACTIVATION FUNCTION?

In a neural network, inputs, which are typically real values, are fed into the neurons in the network. Each neuron has a weight, and the inputs are multiplied by the weight and fed into the activation function. Each neuron's output is the input of the neurons in the next layer of the network, and so the inputs cascade through multiple activation functions until eventually, the output layer generates a prediction. Neural networks rely on nonlinear activation functions—the derivative of the activation function helps the network learn through the backpropagation process.



SOME COMMON ACTIVATION FUNCTIONS INCLUDE THE FOLLOWING:

1. **The sigmoid function** has a smooth gradient and outputs values between zero and one. For very high or low values of the input parameters, the network can be very slow to reach a prediction, called the *vanishing gradient* problem.
2. **The TanH function** is zero-centered making it easier to model inputs that are strongly negative, strongly positive or neutral.
3. **The ReLu function** is highly computationally efficient but is not able to process inputs that approach zero or negative.
4. **The Leaky ReLu function** has a small positive slope in its negative area, enabling it to process zero or negative values.
5. **The Parametric ReLu function** allows the negative slope to be learned, performing backpropagation to learn the most effective slope for zero and negative input values.
6. **Softmax** is a special activation function used for output neurons. It normalizes outputs for each class between 0 and 1, and returns the probability that the input belongs to a specific class.
7. **Swish** is a new activation function discovered by Google researchers. It performs better than ReLu with a similar level of computational efficiency.



Two common neural network activation functions - Sigmoid and Tanh

APPLICATIONS OF ANN

1. Data Mining: Discovery of meaningful patterns (knowledge) from large volumes of data.
2. Expert Systems: A computer program for decision making that simulates thought process of a human expert.
3. Fuzzy Logic: Theory of approximate reasoning.
4. Artificial Life: Evolutionary Computation, Swarm Intelligence.
5. Artificial Immune System: A computer program based on the biological immune system.
6. Medical: At the moment, the research is mostly on modelling parts of the human body and recognizing diseases from various scans (e.g. cardiograms, CAT scans, ultrasonic scans, etc.). Neural networks are ideal in recognizing diseases using scans since there is no need to provide a specific algorithm on how to identify the disease. Neural networks learn by example so the details of how to recognize the disease are not needed. What is needed is a set of examples that are representative of all the variations of the disease. The quantity of examples is not as important as the 'quantity'. The examples need to be selected very carefully if the system is to perform reliably and efficiently.

7. Computer Science: Researchers in quest of artificial intelligence have created spin offs like dynamic programming, object oriented programming, symbolic programming, intelligent storage management systems and many more such tools. The primary goal of creating an artificial intelligence still remains a distant dream but people are getting an idea of the ultimate path, which could lead to it.
8. Aviation: Airlines use expert systems in planes to monitor atmospheric conditions and system status. The plane can be put on autopilot once a course is set for the destination.
9. Weather Forecast: Neural networks are used for predicting weather conditions. Previous data is fed to a neural network, which learns the pattern and uses that knowledge to predict weather patterns.
10. Neural Networks in business: Business is a diverted field with several general areas of specialization such as accounting or financial analysis. Almost any neural network application would fit into one business area or financial analysis.
11. There is some potential for using neural networks for business purposes, including resource allocation and scheduling.
12. There is also a strong potential for using neural networks for database mining, which is, searching for patterns implicit within the explicitly stored information in databases. Most of the funded work in this area is classified as proprietary. Thus, it is not possible to report on the full extent of the work going on. Most work is applying neural networks, such as the Hopfield-Tank network for optimization and scheduling.
13. Marketing: There is a marketing application which has been integrated with a neural network system. The Airline Marketing Tactician (a trademark abbreviated as AMT) is a computer system made of various intelligent technologies including expert systems. A feed forward neural network is integrated with the AMT and was trained using back-propagation to assist the marketing control of airline seat allocations. The adaptive neural approach was amenable to rule expression. Additionally, the application's environment changed rapidly and constantly, which required a continuously adaptive solution.
14. Credit Evaluation: The HNC company, founded by Robert Hecht-Nielsen, has developed several neural network applications. One of them is the Credit Scoring system which increases the profitability of the existing model up to 27%. The HNC neural systems were also applied to mortgage screening. A neural network automated mortgage insurance under writing system was developed by the Nestor Company. This system was trained with 5048 applications of which 2597 were certified. The data related to property and borrower qualifications. In a conservative mode the system agreed on the under writers on 97% of the cases. In the liberal model the system agreed 84% of the cases. This system run on an Apollo DN3000 and used 250K memory while processing a case file in approximately 1 sec.

ADVANTAGES OF ANN

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time.
3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. Pattern recognition: is a powerful technique for harnessing the information in the data and generalizing about it. Neural nets learn to recognize the patterns which exist in the data set.
5. The system is developed through learning rather than programming.. Neural nets teach themselves the patterns in the data freeing the analyst for more interesting work.

6. Neural networks are flexible in a changing environment. Although neural networks may take some time to learn a sudden drastic change they are excellent at adapting to constantly changing information.
7. Neural networks can build informative models whenever conventional approaches fail. Because neural networks can handle very complex interactions they can easily model data which is too difficult to model with traditional approaches such as inferential statistics or programming logic.
8. Performance of neural networks is at least as good as classical statistical modelling, and better on most problems. The neural networks build models that are more reflective of the structure of the data in significantly less time.

LIMITATIONS OF ANN

In this technological era everything has Merits and some Demerits in others words there is a Limitation with every system which makes this ANN technology weak in some points. The various Limitations of ANN are:-

- 1) ANN is not a daily life general purpose problem solver.
- 2) There is no structured methodology available in ANN.
- 3) There is no single standardized paradigm for ANN development.
- 4) The Output Quality of an ANN may be unpredictable.
- 5) Many ANN Systems does not describe how they solve problems.
- 6) Black box Nature
- 7) Greater computational burden.
- 8) Proneness to over fitting.
- 9) Empirical nature of model development.

ARTIFICIAL NEURAL NETWORK CONCEPTS/TERMINOLOGY

Here is a glossary of basic terms you should be familiar with before learning the details of neural networks.

Inputs: Source data fed into the neural network, with the goal of making a decision or prediction about the data. Inputs to a neural network are typically a set of real values; each value is fed into one of the neurons in the input layer.

Training Set: A set of inputs for which the correct outputs are known, used to train the neural network.

Outputs : Neural networks generate their predictions in the form of a set of real values or boolean decisions. Each output value is generated by one of the neurons in the output layer.

Neuron/perceptron: The basic unit of the neural network. Accepts an input and generates a prediction.

Each neuron accepts part of the input and passes it through the activation function. Common activation functions are sigmoid, TanH and ReLu. Activation functions help generate output values within an acceptable range, and their non-linear form is crucial for training the network.

Weight Space: Each neuron is given a numeric weight. The weights, together with the activation function, define each neuron's output. Neural networks are trained by fine-tuning weights, to discover the optimal set of weights that generates the most accurate prediction.

Forward Pass: The forward pass takes the inputs, passes them through the network and allows each neuron to react to a fraction of the input. Neurons generate their outputs and pass them on to the next layer, until eventually the network generates an output.

Error Function: Defines how far the actual output of the current model is from the correct output. When training the model, the objective is to minimize the error function and bring output as close as possible to the correct value.

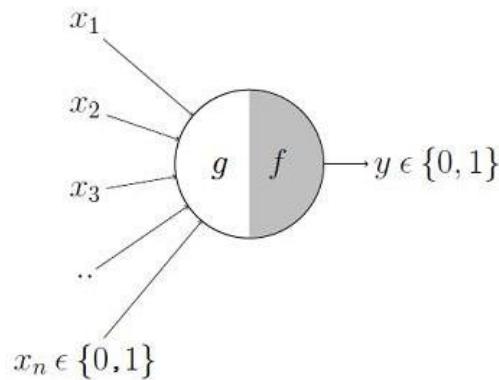
Backpropagation: In order to discover the optimal weights for the neurons, we perform a backward pass, moving back from the network's prediction to the neurons that generated that prediction. This is called backpropagation. Backpropagation tracks the derivatives of the activation functions in each successive neuron, to find weights that bring the loss function to a minimum, which will generate the best prediction. This is a mathematical process called *gradient descent*.

Bias and Variance: When training neural networks, like in other machine learning techniques, we try to balance between bias and variance. Bias measures how well the model fits the training set—able to correctly predict the known outputs of the training examples. Variance measures how well the model works with unknown inputs that were not available during training. Another meaning of bias is a “bias neuron” which is used in every layer of the neural network. The bias neuron holds the number 1, and makes it possible to move the activation function up, down, left and right on the number graph.

Hyperparameters: A hyper parameter is a setting that affects the structure or operation of the neural network. In real deep learning projects, tuning hyper parameters is the primary way to build a network that provides accurate predictions for a certain problem. Common hyper parameters include the number of hidden layers, the activation function, and how many times (epochs) training should be repeated.

McCULLOCH-PITTS MODEL

In 1943 two electrical engineers, Warren McCulloch and Walter Pitts, published the first paper describing what we would call a neural network.



It may be divided into 2 parts. The first part, g takes an input, performs an aggregation and based on the aggregated value the second part, f makes a decision. Let us suppose that I want to predict my own decision, whether to watch a random football game or not on TV. The inputs are all boolean i.e., $\{0,1\}$ and my output variable is also boolean $\{0: \text{Will watch it}, 1: \text{Won't watch it}\}$.

So, X_1 could be ‘is Indian Premier League On’ (I like Premier League more)

X_2 could be ‘is it a knockout game (I tend to care less about the league level matches)

X_3 could be ‘is Not Home’ (Can’t watch it when I’m in College. Can I?)

X_4 could be ‘is my favorite team playing’ and so on.

These inputs can either be excitatory or inhibitory. Inhibitory inputs are those that have maximum effect on the decision making irrespective of other inputs i.e., if X_3 is 1 (not home) then my output will always be 0 i.e., the neuron will never fire, so X_3 is an inhibitory input. Excitatory inputs are NOT the ones that will make the neuron fire on their own but they might fire it when combined together. Formally, this is what is going on:

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

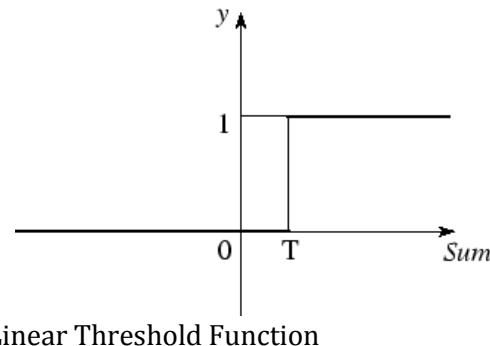
$$\begin{aligned} y &= f(g(\mathbf{x})) = 1 \quad \text{if} \quad g(\mathbf{x}) \geq \theta \\ &= 0 \quad \text{if} \quad g(\mathbf{x}) < \theta \end{aligned}$$

We can see that $g(\mathbf{x})$ is just doing a sum of the inputs — a simple aggregation. And theta here is called thresholding parameter. For example, if I always watch the game when the sum turns out to be 2 or more, the theta is 2 here. This is called the Thresholding Logic.

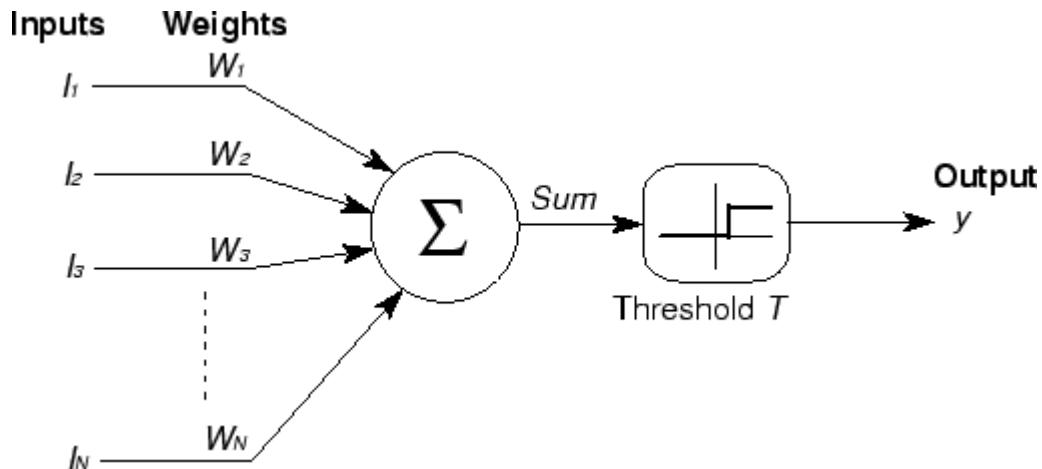
The McCulloch-Pitts neural model is also known as linear threshold gate. It is a neuron of a set of inputs $I_1, I_2, I_3, \dots, I_m$ and one output ' y '. The linear threshold gate simply classifies the set of inputs into two different classes. Thus the output y is binary. Such a function can be described mathematically using these equations:

$$\text{Sum} = \sum_{i=1}^N I_i W_i, \quad y = f(\text{Sum}).$$

Where, $W_1, W_2, W_3, \dots, W_m$ are weight values normalized in the range of either $(0, 1)$ or $(-1, 1)$ and associated with each input line, Sum is the weighted sum, and T is a threshold constant. The function f is a linear step function at threshold T as shown in figure 2.3. The symbolic representation of the linear threshold gate is shown in figure below.



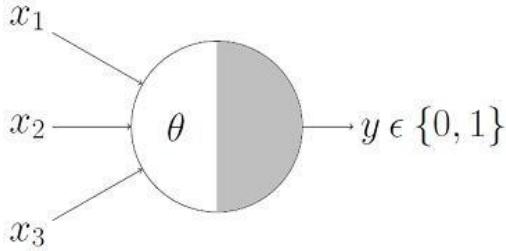
Linear Threshold Function



Symbolic Illustration of Linear Threshold Gate

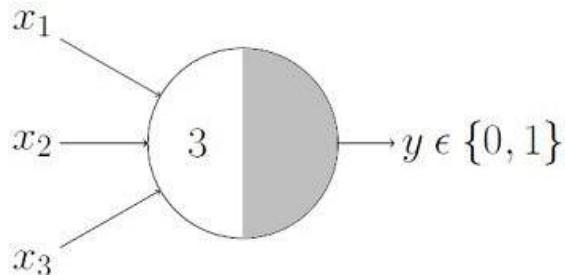
BOOLEAN FUNCTIONS USING McCULLOCH-PITTS NEURON

In any Boolean function, all inputs are Boolean and the output is also Boolean. So essentially, the neuron is just trying to learn a Boolean function.



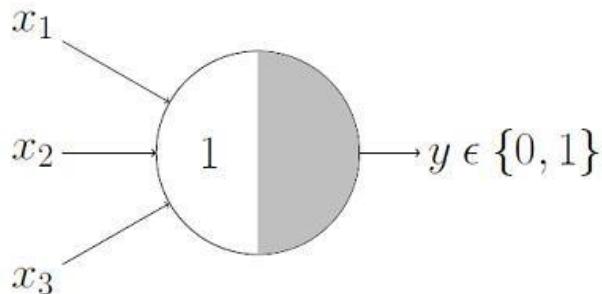
This representation just denotes that, for the boolean inputs x_1 , x_2 and x_3 if the $g(x)$ i.e., sum $\geq \theta$, the neuron will fire otherwise, it won't.

AND Function



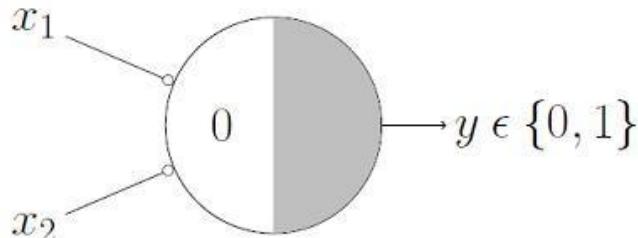
An AND function neuron would only fire when ALL the inputs are ON i.e., $g(x) \geq 3$ here.

OR Function



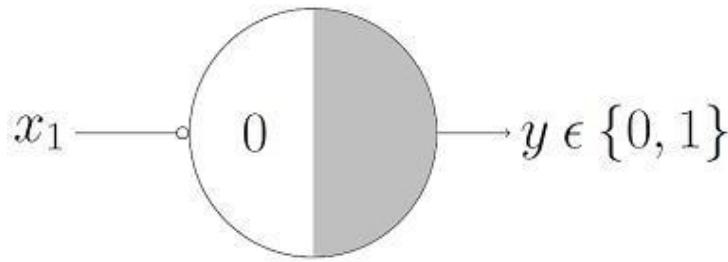
For an OR function neuron would fire if ANY of the inputs is ON i.e., $g(x) \geq 1$ here.

NOR Function



For a NOR neuron to fire, we want ALL the inputs to be 0 so the thresholding parameter should also be 0 and we take them all as inhibitory input.

NOT Function



For a NOT neuron, 1 outputs 0 and 0 outputs 1. So we take the input as an inhibitory input and set the thresholding parameter to 0.

We can summarize these rules with the McCullough-Pitts output rule as:

The McCulloch-Pitts model of a neuron is simple yet has substantial computing potential. It also has a precise mathematical definition. However, this model is so simplistic that it only generates a binary output and also the weight and threshold values are fixed. The neural computing algorithm has diverse features for various applications. Thus, we need to obtain the neural model with more flexible computational features.

WHAT ARE THE LEARNING RULES IN ANN?

Learning rule is a method or a mathematical logic. It helps a Neural Network to learn from the existing conditions and improve its performance. 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.

The different learning rules in the Neural network are:

1. Hebbian learning rule – It identifies, how to modify the weights of nodes of a network.
 2. Perceptron learning rule – Network starts its learning by assigning a random value to each weight.
 3. Delta learning rule – Modification in synaptic weight of a node is equal to the multiplication of error and the input.
 4. Correlation learning rule – The correlation rule is the supervised learning.
 5. Outstar learning rule – We can use it when it assumes that nodes or neurons in a network arranged in a layer.
- 1. Hebbian Learning Rule:** The Hebbian rule was the first learning rule. In 1949 Donald Hebb developed it as learning algorithm of the unsupervised neural network. We can use it to identify how to improve the weights of nodes of a network. The Hebb learning rule assumes that – If two neighbor neurons activated and deactivated at the same time, then the weight connecting these neurons should increase. At the start, values of all weights are set to zero. This learning rule can be used for both soft- and hard-activation functions. Since desired responses of neurons are not used in the learning procedure, this is the unsupervised learning rule. The absolute values of the weights are usually proportional to the learning time, which is undesired.

$$W_{ij} = x_i^* x_j$$

Mathematical Formula of Hebb Learning Rule.

- 2. Perceptron Learning Rule:** Each connection in a neural network has an associated weight, which changes in the course of learning. According to it, an example of supervised learning, the network starts its learning by assigning a random value to each weight. Calculate the output value on the basis of a set of records for which we can know the expected output value. This is the learning sample that indicates the entire definition. As a result, it is called a learning sample. The network then compares the calculated output value with the expected value. Next calculates an error function ϵ , which can be the sum of squares of the errors occurring for each individual in the learning sample which can be computed as:

$$\sum_i \sum_j (E_{ij} - O_{ij})^2$$

Mathematical Formula of Perceptron Learning Rule

Perform the first summation on the individuals of the learning set, and perform the second summation on the output units. E_{ij} and O_{ij} are the expected and obtained values of the j^{th} unit for the i^{th} individual. The network then adjusts the weights of the different units, checking each time to see if the error function has increased or decreased. As in a conventional regression, this is a matter of solving a problem of least squares. Since assigning the weights of nodes according to users, it is an example of supervised learning.

- 3. Delta Learning Rule:** Developed by Widrow and Hoff, the delta rule, is one of the most common learning rules. It depends on supervised learning. This rule states that the modification in synaptic weight of a node is equal to the multiplication of error and the input. In Mathematical form the delta rule is as follows:

$$\Delta w = \eta (t - y) x_i$$

Mathematical Formula of Delta Learning Rule

For a given input vector, compare the output vector is the correct answer. If the difference is zero, no learning takes place; otherwise, adjusts its weights to reduce this difference. The change in weight from u_i to u_j is: $\Delta w_{ij} = r * a_i * e_j$. where r is the learning rate, a_i represents the activation of u_i and e_j is the difference between the expected output and the actual output of u_j . If the set of input patterns form an independent set then learn arbitrary associations using the delta rule.

It has been seen that for networks with linear activation functions and with no hidden units. The error squared vs. the weight graph is a paraboloid in n-space. Since the proportionality constant is negative, the graph of such a function is concave upward and has the least value. The vertex of this paraboloid represents the point where it reduces the error. The weight vector corresponding to this point is then the ideal weight vector. We can use the delta learning rule with both single output unit and several output units. While applying the delta rule assume that the error can be directly measured. The aim of applying the delta rule is to reduce the difference between the actual and expected output that is the error.

- 4. Correlation Learning Rule:** The correlation learning rule based on a similar principle as the Hebbian learning rule. It assumes that weights between responding neurons should be more positive, and weights between neurons with opposite reaction should be more negative. Contrary to the Hebbian rule, the correlation rule is the supervised learning, instead of an actual. The response, o_j , the desired response, d_j , uses for the weight-change calculation. In Mathematical form the correlation learning rule is as follows:

$$\Delta w_{ij} = \eta x_i d_j$$

Mathematical Formula of Correlation Learning Rule

Where d_j is the desired value of output signal. This training algorithm usually starts with the initialization of weights to zero. Since assigning the desired weight by users, the correlation learning rule is an example of supervised learning.

5. Out Star Learning Rule: We use the Out Star Learning Rule when we assume that nodes or neurons in a network arranged in a layer. Here the weights connected to a certain node should be equal to the desired outputs for the neurons connected through those weights. The out start rule produces the desired response t for the layer of n nodes. Apply this type of learning for all nodes in a particular layer. Update the weights for nodes are as in Kohonen neural networks. In Mathematical form, express the out star learning as follows:

$$W_{jk} = \begin{cases} \eta(y_k - W_{jk}) & \text{if node } j \text{ wins the competition} \\ 0 & \text{if node } j \text{ losses the competition} \end{cases}$$

Mathematical Formula of Out Star Learning Rule

This is a supervised training procedure because desired outputs must be known.

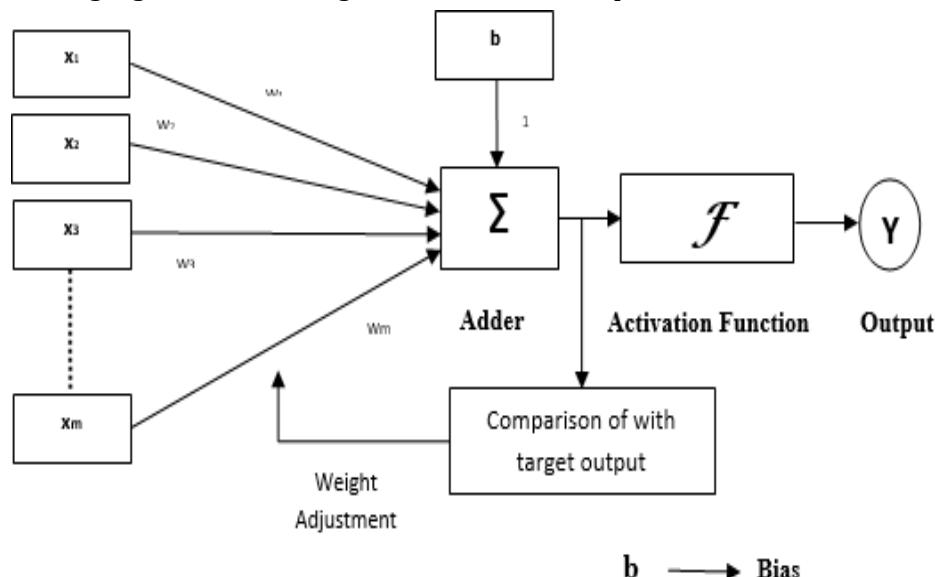
BRIEFLY EXPLAIN THE ADALINE MODEL OF ANN.

ADALINE (Adaptive Linear Neuron or later Adaptive Linear Element) is an early single-layer artificial neural network and the name of the physical device that implemented this network. The network uses memistors. It was developed by Professor Bernard Widrow and his graduate student Ted Hoff at Stanford University in 1960. It is based on the McCulloch-Pitts neuron. It consists of a weight, a bias and a summation function. The difference between Adaline and the standard (McCulloch-Pitts) perceptron is that in the learning phase, the weights are adjusted according to the weighted sum of the inputs (the net). In the standard perceptron, the net is passed to the activation (transfer) function and the function's output is used for adjusting the weights. Some important points about Adaline are as follows:

- It uses bipolar activation function.
- It uses delta rule for training to minimize the Mean-Squared Error (MSE) between the actual output and the desired/target output.
- The weights and the bias are adjustable.

Architecture of ADALINE network: The basic structure of Adaline is similar to perceptron having an extra feedback loop with the help of which the actual output is compared with the desired/target output. After comparison on the basis of training algorithm, the weights and bias will be updated.

Architecture of ADALINE: The basic structure of Adaline is similar to perceptron having an extra feedback loop with the help of which the actual output is compared with the desired/target output. After comparison on the basis of training algorithm, the weights and bias will be updated.



Training Algorithm of ADALINE:

Step 1 – Initialize the following to start the training:

- Weights
- Bias
- Learning rate α

For easy calculation and simplicity, weights and bias must be set equal to 0 and the learning rate must be set equal to 1.

Step 2 – Continue step 3-8 when the stopping condition is not true.

Step 3 – Continue step 4-6 for every bipolar training pair $s : t$.

Step 4 – Activate each input unit as follows:

$$x_i = s_i \quad (i=1 \text{ to } n)$$

Step 5 – Obtain the net input with the following relation:

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

Here 'b' is bias and 'n' is the total number of input neurons.

Step 6 – Apply the following activation function to obtain the final output:

$$f(y_{in}) = \begin{cases} 1, & \text{if } y_{in} \geq 0 \\ -1, & \text{if } y_{in} < 0 \end{cases}$$

Step 7 – Adjust the weight and bias as follows :

$$\text{Case 1 - if } y \neq t \text{ then, } w_i(\text{new}) = w_i(\text{old}) + \alpha(t - y_{in})x_i$$

$$b(\text{new}) = b(\text{old}) + \alpha(t - y_{in})$$

$$\text{Case 2 - if } y = t \text{ then, } w_i(\text{new}) = w_i(\text{old})$$

$$b(\text{new}) = b(\text{old})$$

Here 'y' is the actual output and 't' is the desired/target output. $(t - y_{in})$ is the computed error.

Step 8 – Test for the stopping condition, which will happen when there is no change in weight or the highest weight change occurred during training is smaller than the specified tolerance.

EXPLAIN MULTIPLE ADAPTIVE LINEAR NEURONS (MADALINE).

Madaline which stands for Multiple Adaptive Linear Neuron, is a network which consists of many Adalines in parallel. It will have a single output unit. Three different training algorithms for MADALINE networks called Rule I, Rule II and Rule III have been suggested, which cannot be learned using backpropagation. The first of these dates back to 1962 and cannot adapt the weights of the hidden-output connection.[10] The second training algorithm improved on Rule I and was described in 1988.[8] The third "Rule" applied to a modified network with sigmoid activations instead of signum; it was later found to be equivalent to backpropagation. The Rule II training algorithm is based on a principle called "minimal disturbance". It proceeds by looping over training examples, then for each example, it:

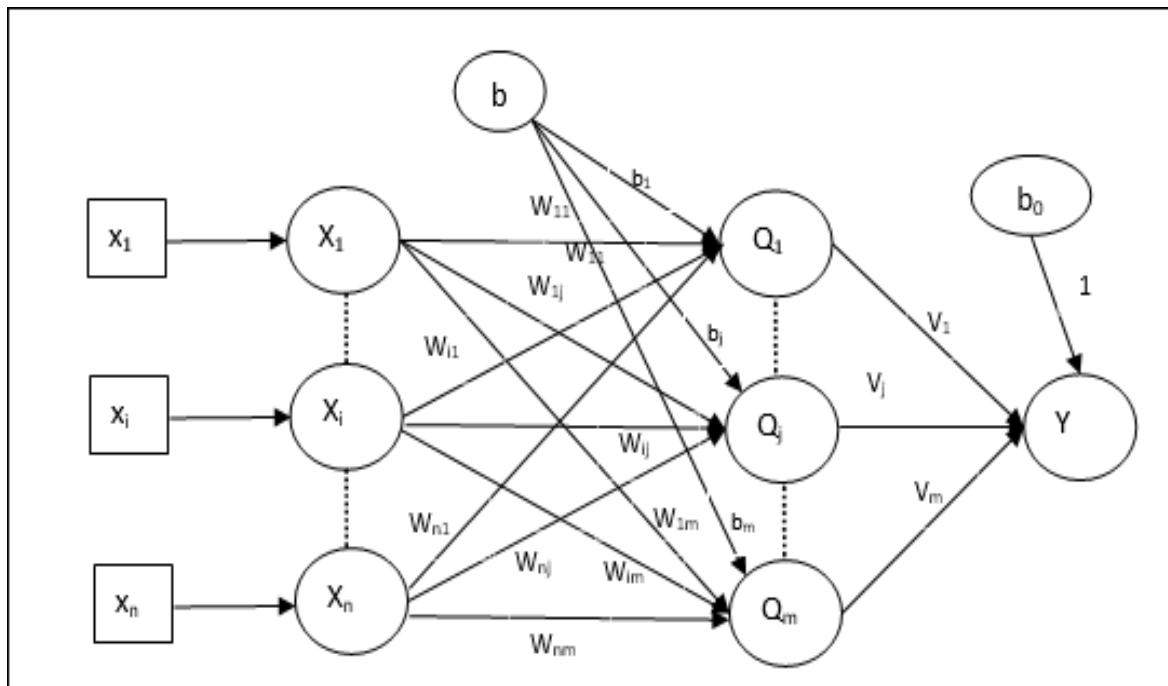
- finds the hidden layer unit (ADALINE classifier) with the lowest confidence in its prediction, tentatively flips the sign of the unit,
- accepts or rejects the change based on whether the network's error is reduced,
- stops when the error is zero.

Some important points about Madaline are as follows:

- It is just like a multilayer perceptron, where Adaline will act as a hidden unit between the input and the Madaline layer.
- The weights and the bias between the input and Adaline layers, as in we see in the Adaline architecture, are adjustable.
- The Adaline and Madaline layers have fixed weights and bias of 1.
- Training can be done with the help of Delta rule.

BRIEFLY EXPLAIN THE ARCHITECTURE OF MADALINE

MADALINE (Many ADALINE) is a three-layer (input, hidden, output), fully connected, feed-forward artificial neural network architecture for classification that uses ADALINE units in its hidden and output layers, i.e. its activation function is the sign function. The three-layer network uses memistors. The architecture of Madaline consists of “n” neurons of the input layer, “m” neurons of the Adaline layer, and 1 neuron of the Madaline layer. The Adaline layer can be considered as the hidden layer as it is between the input layer and the output layer, i.e. the Madaline layer.



Training Algorithm of MADALINE

By now we know that only the weights and bias between the input and the Adaline layer are to be adjusted, and the weights and bias between the Adaline and the Madaline layer are fixed.

Step 1 – Initialize the following to start the training:

- Weights
- Bias
- Learning rate α

For easy calculation and simplicity, weights and bias must be set equal to 0 and the learning rate must be set equal to 1.

Step 2 – Continue step 3-8 when the stopping condition is not true.

Step 3 – Continue step 4-6 for every bipolar training pair s:t.

Step 4 – Activate each input unit as follows:

$$x_i = s_i \quad (I = 1 \text{ to } n)$$

Step 5 – Obtain the net input at each hidden layer, i.e. the Adaline layer with the following relation:

$$O_{inj} = b_j + \sum_i x_i w_{ij} \quad j = 1 \text{ to } m$$

Here 'b' is bias and 'n' is the total number of input neurons.

Step 6 – Apply the following activation function to obtain the final output at the Adaline and the Madaline

Layer: $f(y_{in}) = \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases}$

Output at the hidden Adaline unit $Q_j = f(Q_{inj})$

Final output of the network $y = f(y_{in})$
i.e.

$$y_{inj} = b_0 + \sum_{j=1}^m Q_j v_j$$

Step 7 – Calculate the error and adjust the weights as follows –

Case 1 – if $y \neq t$ and $t = 1$ then,

$$\begin{aligned} w_{ij}(\text{new}) &= w_{ij}(\text{old}) + \alpha(1 - Q_{inj})x_i \\ b_j(\text{new}) &= b_j(\text{old}) + \alpha(1 - Q_{inj}) \end{aligned}$$

In this case, the weights would be updated on Q_j where the net input is close to 0 because $t = 1$.

Case 2 – if $y \neq t$ and $t = -1$ then,

$$\begin{aligned} w_{ik}(\text{new}) &= w_{ik}(\text{old}) + \alpha(-1 - Q_{ink})x_i \\ b_k(\text{new}) &= b_k(\text{old}) + \alpha(-1 - Q_{ink}) \end{aligned}$$

In this case, the weights would be updated on Q_k where the net input is positive because $t = -1$.
Here 'y' is the actual output and 't' is the desired/target output.

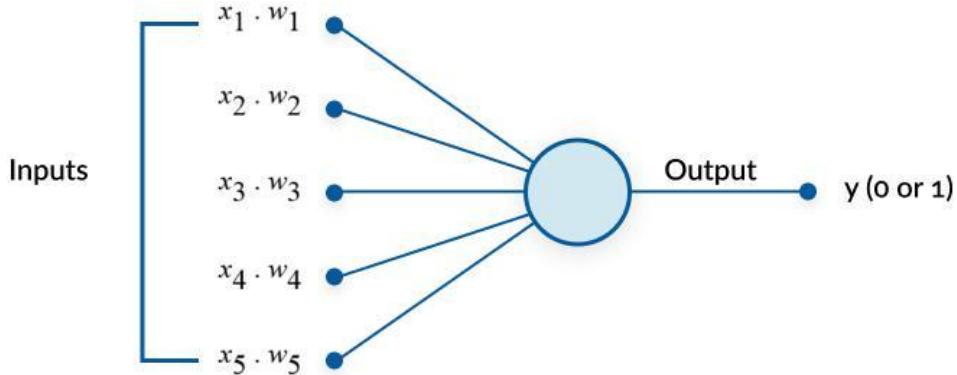
Case 3 – if $y = t$, then there would be no change in weights.

Step 8 – Test for the stopping condition, which will happen when there is no change in weight or the highest weight change occurred during training is smaller than the specified tolerance.

WHAT IS A PERCEPTRON?

A perceptron is a binary classification algorithm modeled after the functioning of the human brain—it was intended to emulate the neuron. The perceptron, while it has a simple structure, has the ability to learn a

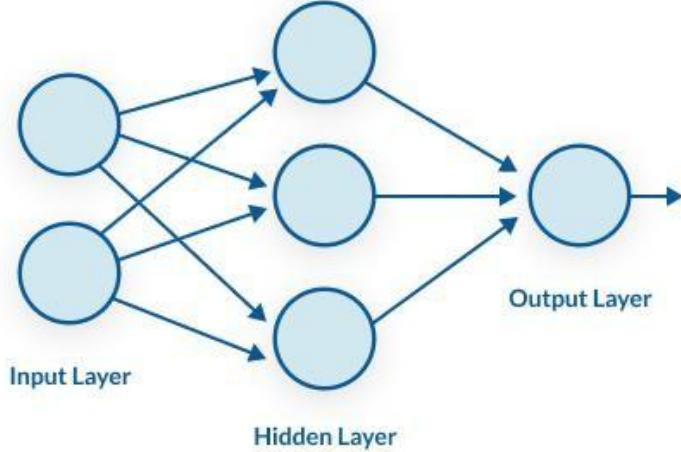
Perceptron Input And Output



What is Multilayer Perceptron?

A multilayer perceptron (MLP) is a group of perceptrons, organized in multiple layers, that can accurately answer complex questions. Each perceptron in the first layer (on the left) sends signals to all the perceptrons in the second layer, and so on. An MLP contains an input layer, at least one hidden layer, and an output layer.

Perceptron Input And Output



The perceptron learns as follows:

1. Takes the inputs which are fed into the perceptrons in the input layer, multiplies them by their weights, and computes the sum.
2. Adds the number one, multiplied by a “bias weight”. This is a technical step that makes it possible to move the output function of each perceptron (the activation function) up, down, left and right on the number graph.
3. Feeds the sum through the activation function—in a simple perceptron system, the activation function is a step function.
4. The result of the step function is the output.