

# MODULE - 1

## INTRODUCTION

Introduction: Biological Neuron- Artificial Neural Model-Types of activation functions-

Architecture: Feedforward and Feedback, Convex Sets, Convex Hull and Linear Separability, Non-Linear Separable Problem. XOR Problem, Multilayer Networks.

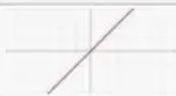
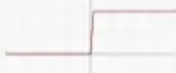




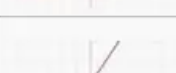
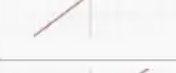

Learning: Learning Algorithms, Error correction and Gradient Descent Rules, Learning objective of TLNs, Perceptron Learning Algorithm, Perceptron Convergence Theorem.

### **1. Compare structure and function of biological neurons with artificial neurons. List the characteristics of NN.**

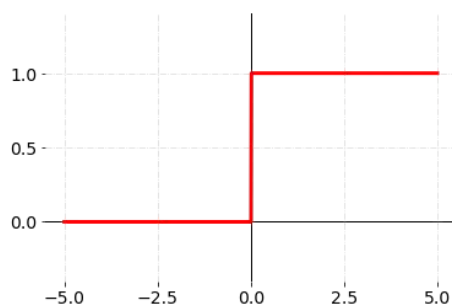
BIOLOGICAL NEURONS	ARTIFICIAL NEURONS
1. Major components : Axions, Dendrites, Synapses	1. Major components : nodes, layers, inputs , outputs, synaptic weights, bias
2. Information from other neurons are in the form of electrical pulses which flows through dendrites	2. Connections of the neurons are made up of layers and computation are in terms of activation functions from the features collected from input layers.
3. A synapse is able to increase or decrease the strength of connection where information is stored	3. The artificial signals can be changed by weight updation procedure.
4. Human brain works asynchronously	4. ANN works synchronously.
5. Eg: bats, human brain	5. Eq : neural network, ANN etc.
6. Biological neurons compute slowly (several ms per computation)	6. ANN compute fast (<1 nanosecond per computation )
7. Biological neurons have complicated topologies	7. ANN have often a tree structure

### **2. List activation or neural signal functions used in ANN.**

The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. The purpose of the activation function is to introduce non-linearity into the output of a neuron.

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) <sup>[2]</sup>		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) <sup>[3]</sup>		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

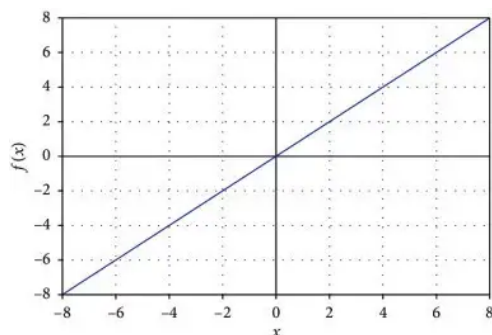
### Binary Threshold Signal Function:



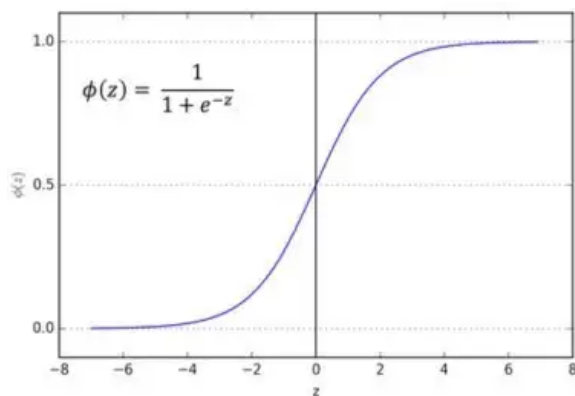
$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases}$$

### Linear function :

$$f(x) = x$$

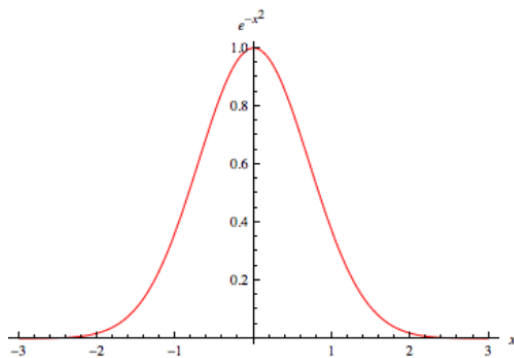


### Sigmoid function :



Sigmoid function (also known as logistic function) takes a probabilistic approach and the output ranges between 0–1. It normalizes the output of each neuron.

### Gaussian Signal Function:



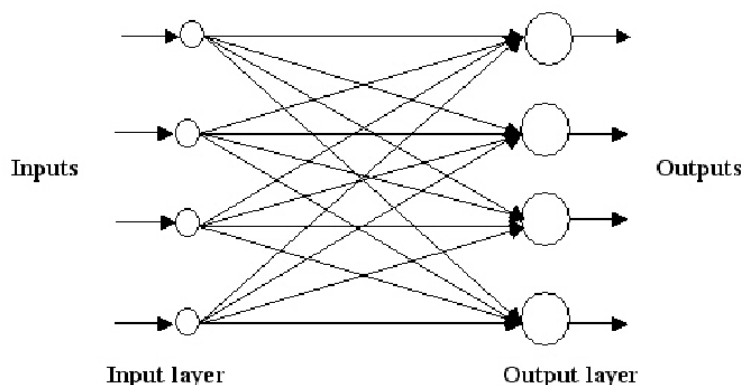
$$f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-(x-\mu)^2 / (2\sigma^2)},$$

## **Explain the different network architecture in NN.**

Three fundamental classes of network architectures.

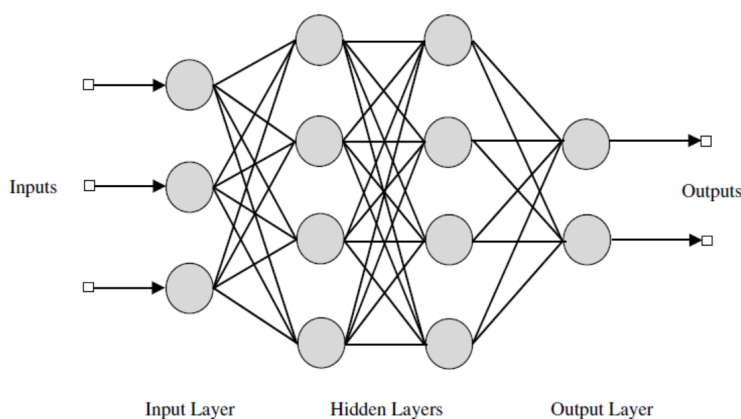
### 1. Single-Layer Feedforward Networks

- In a layered neural network, the neurons are organized in the form of layers.
- In the simplest form of a layered network, we have an input layer of source nodes that projects directly onto an output layer of neurons (computation nodes).
- “Single-layer” referring to the output layer of computation nodes (neurons).



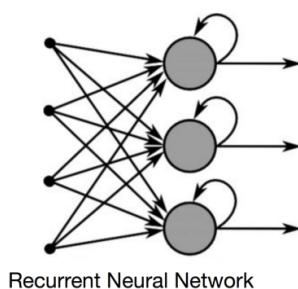
## 2. Multilayer Feedforward Networks

- The term “hidden” refers to the fact that this part of the neural network is not seen directly from either the input or output of the network.
- By adding one or more hidden layers, the network is enabled to extract higher-order statistics from its input.
- Example, a feedforward network with  $m$  source nodes,  $h_1$  neurons in the first hidden layer,  $h_2$  neurons in the second hidden layer, and  $q$  neurons in the output layer is referred to as an  $m-h_1-h_2-q$  network.
- The neural network in Figure is said to be fully connected in the sense that every node in each layer of the network is connected to every other node in the adjacent forward layer. If some of the communication links (synaptic connections) are missing from the network, we say that the network is partially connected.
- The function of hidden neurons is to intervene between the external input and the network output in some useful manner.
- Memory-less: output depends only on the present input.
- Possess no dynamics.
- Demonstrate powerful properties for function approximation and widespread applications in pattern classification.



## 3. Recurrent Networks

- A recurrent neural network distinguishes itself from a feedforward neural network in that it has at least one feedback loop.
- New state of the network is a function of the current input and the present state of the network.



**List the characteristics of NN.**

### 1. Nonlinearity :

- An artificial neuron can be linear or nonlinear.
- A neural network, made up of an interconnection of nonlinear neurons, is itself nonlinear.

### 2. Input–Output Mapping

- Supervised learning involves modification of the synaptic weights of a neural network by applying a set of labeled training examples.
- The network learns from the examples by constructing an input–output mapping for the problem at hand.

### 3. Adaptivity

- Neural networks have a built-in capability to adapt their synaptic weights to changes in the surrounding environment.
- A neural network trained to operate in a specific environment can be easily retrained to deal with minor changes in the operating environmental conditions. when it is operating in a nonstationary environment a neural network may be designed to change its synaptic weights in real time

### 4. Evidential Response

- In the context of pattern classification, a neural network can be designed to provide information not only about which particular pattern to select, but also about the confidence in the decision made.

### 5. Contextual Information

- Every neuron in the network is potentially affected by the global activity of all other neurons in the network. Consequently, contextual information is dealt with naturally by a neural network.

### 6. Fault Tolerance

- A neural network, implemented in hardware form, has the potential to be inherently fault tolerant, or capable of robust computation, in the sense that its performance degrades gracefully under adverse operating conditions.
- In principle, a neural network exhibits a graceful degradation in performance rather than catastrophic failure.
- For fault tolerance, it may be necessary to take corrective measures in designing the algorithm used to train the network.

### 7. VLSI Implementability

- Massively parallel nature of a neural network makes it potentially fast for the computation of certain tasks.
- The VLSI(very-large-scale-integrated) architecture is designed to capture complex behaviour in a hierarchical manner and hence suited for NN implementation.

### 8. Uniformity of Analysis and Design

- Neurons, in one form or another, represent an ingredient common to all neural networks.
- This commonality makes it possible to share theories and learning algorithms in different applications of neural networks.
- Modular networks can be built through a seamless integration of modules.

### 9. Computation Power

- Massively parallel distributed structure

## 10. Generalization

- Reasonable outputs for inputs not encountered during training (learning).

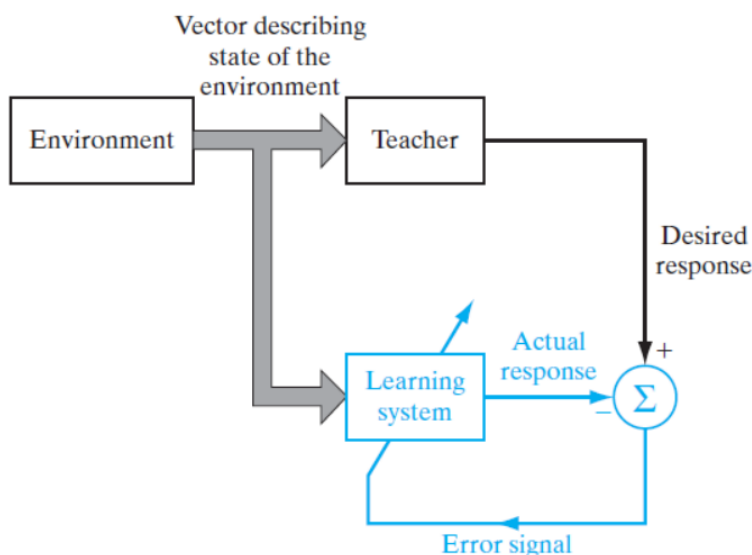
### **Explain the learning algorithms in NN.**

When a Neural Network system learns, it generates an internal model of the sampled data. These models are represented in terms of structural weight vectors.

Learning algorithms define an architecture-dependent procedure to encode information into weights to generate internal models. This learning proceeds by continuous modification of the connection strength (synaptic weights).

A system learns in 3 ways.

#### **Supervised Learning :**



We know that for any network with a set of discrete samples  $X_k$  presented to the system, it generates output as  $S_k$ . These input- output sample pairs are employed to train the network through a form of error correction learning or gradient descent weight adaptation.

For a given system, let's assume the environment to be the input- output samples and the existence of a teacher who has the knowledge of the environment . But the same knowledge is unknown to the network. Now, when both the teacher and the network are exposed to the training vector or the examples drawn from the same environment, by virtue of the built-in knowledge, the teacher is able to provide the neural network with a desired response for that training vector. The desired response,  $D_k$ , represents the “optimum” action to be performed by the neural network. This is how the network parameters are adjusted under the combined influence of the training vector and the error signal.

Supervised learning encodes a behavioristic pattern into the network by attempting to approximate the function that underlies the dataset. Here we want the system to generate the

output  $d_k$  in response to the input  $x_k$  and we say that the system has learnt the underlying map if the  $x_k$  close to the teacher's  $x_k$  elicits a response  $s_k$ , sufficiently close to  $s_k$  of the teacher.

The adjustment of weights is carried out iteratively in a step-by-step fashion with the aim of making the neural network emulate the teacher; the emulation is presumed to be optimum in some statistical sense. In this way, knowledge of the environment available to the teacher is transferred to the neural network through training and stored in the form of “fixed” synaptic weights representing long-term memory.

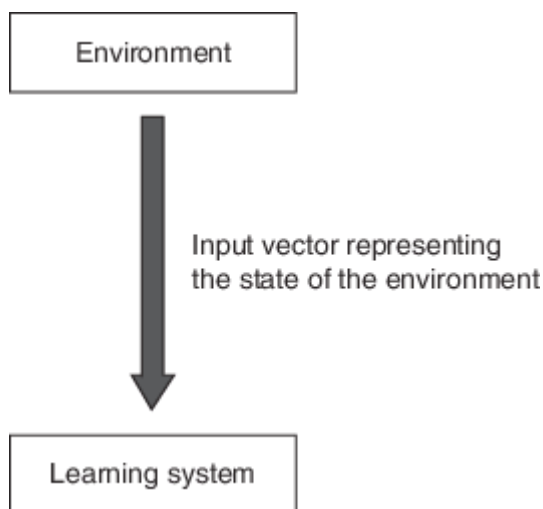
When this condition is reached, we may then disperse the teacher and let the neural network deal with the environment completely by itself.

#### KEY POINTS :

1. Given a discrete set of samples for learning in  $\{x_k, d_k\}$  drawn from the pattern space, the input-output sample pairs are employed to train the network through a form of error correction or gradient descent weight adaptation (LMS and BP).
2. The error measure is usually defined by the mean squared error (MSE).
3. The learning process is terminated when  $E$  (error) is sufficiently small or a failure criterion is met.

Examples would be classification, regression etc.

#### Unsupervised Learning :



Another way of learning is to simply provide the system with an input  $x_k$  and allow it to self-organize its parameters (or weights of the network) to generate internal models or prototypes of sample vectors. This self-organized learning does not require an external teacher or critic to oversee the learning process.

The network is required to learn and the free parameters (synaptic weights) of the network are optimized with respect to that measure.

Once the network has become tuned to the statistical regularities of the input data, the network develops the ability to form internal representations for encoding features of the new classes automatically.

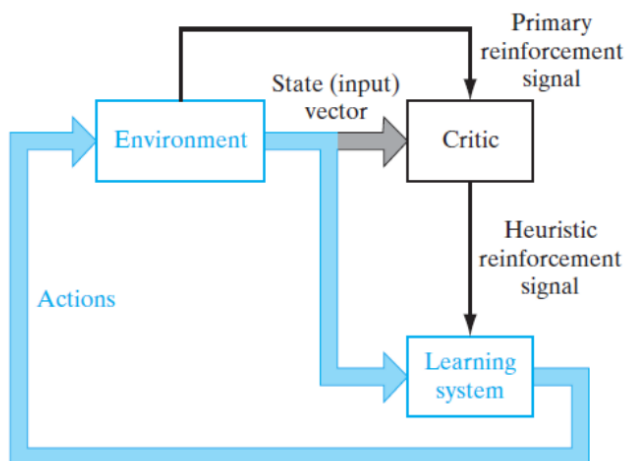
The system attempts to represent the entire data set by employing a small number of prototypical vectors- enough to allow the system to retain a desired level of discrimination between samples. As new samples continuously buffer the system the prototypes will be in a state of constant flux. This learning is often called adaptive vector quantization. A vector quantizer is *adaptive* if the codebook or the encoding rule is changed in time in order to match observed local statistics of the input sequence. This is exactly what happens in unsupervised learning.

#### KEY POINTS :

1. Given a set of data samples, we can identify well defined “clusters” where each cluster defines a class of vectors which are similar in some broad sense.
2. Clusters help establish classification structure within a dataset that has no categories defined in advance. Classes are derived from clusters by appropriate labelling .

So unsupervised learning schemes are mainly used for clustering vector quantization, feature extraction, signal encoding and data analysis.

#### Reinforcement Learning :



Reinforcement Learning is a class of computational algorithms that specifies how an artificial agent or real/simulated robot can learn to select actions in order to maximize the total expected reward. Reinforcement learning is a learning procedure that rewards the neural network for its good output result and punishes it for the bad output result.

There exists a critic that supplies only feedback about the success or failure of the actions. The critic that converts primary reinforcement signal received from the environment into a higher quality reinforcement signal called the heuristic reinforcement signal.



It provides the basis for the learning system to interact with the environment, thereby developing the ability to perform a prescribed task solely on the basis of the outcomes of its experience that result from the interaction.

#### KEY POINTS:

1. This type of learning is based on only the information as to whether or not the actual output is close to the estimate.
2. The system is designed to learn under delayed reinforcement.
3. It may turn out that certain actions taken earlier in that sequence of time steps are in fact the best determinants of overall system behavior. The function of the learning system is to discover these actions and feed them back to the environment.

Reinforcement learning is usually used in general purpose robotic control and artificial intelligence.

### **5. Explain the learning objective for TLN.**

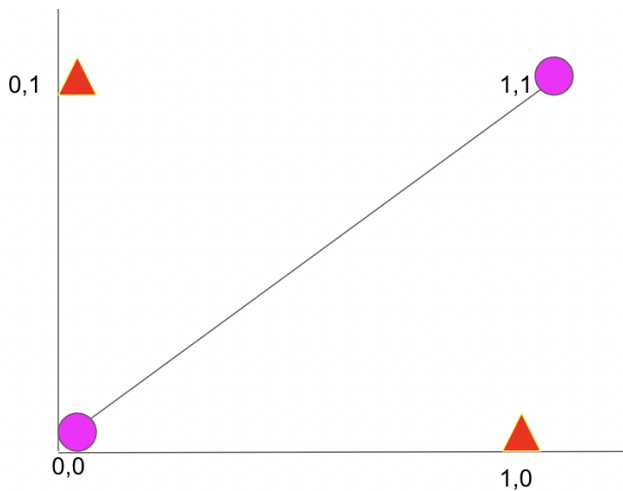
TLNs can be connected into multiple layers in a feedforward fashion. It is common to have more than one hidden layer when solving a classification problem especially when one has a number of disjoint regions in the pattern space that are associated into a single class. We make the following observations for TLNs with two hidden layers apart from the input and the output layer.

- Each neuron in the first hidden layer forms a hyperplane in the input pattern space.
- A neuron in the second hidden layer can form a hyper-region from the outputs of the first layer neurons by performing an AND operation on the hyperplanes. These neurons can thus approximate the boundaries between pattern classes.
- The output layer neurons can then combine disjoint pattern classes into decision regions made by the neurons in the second hidden layer by performing logical OR operations.

### **6. State and Explain XOR is non linearly separable. Also explain the implementation of XOR function using two layered network architecture.**

Linear separability of points is the ability to classify the data points in the hyperplane by avoiding the overlapping of the classes in the planes. Each of the classes should fall above or below the separating line and then they are termed as linearly separable data points. With respect to logical gates operations like AND or OR the outputs generated by this logic are linearly separable in the hyperplane.

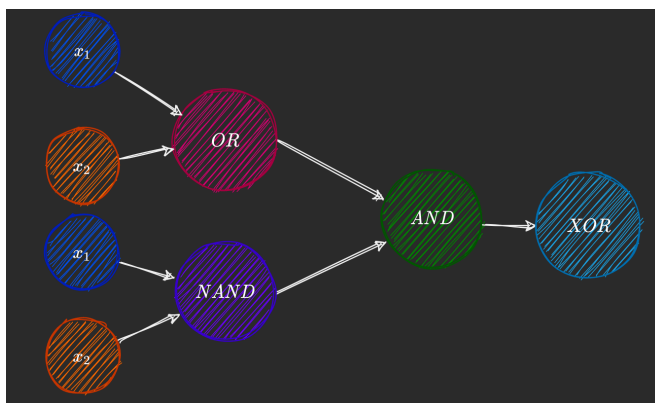
The data points of XOR appear to be as shown below.

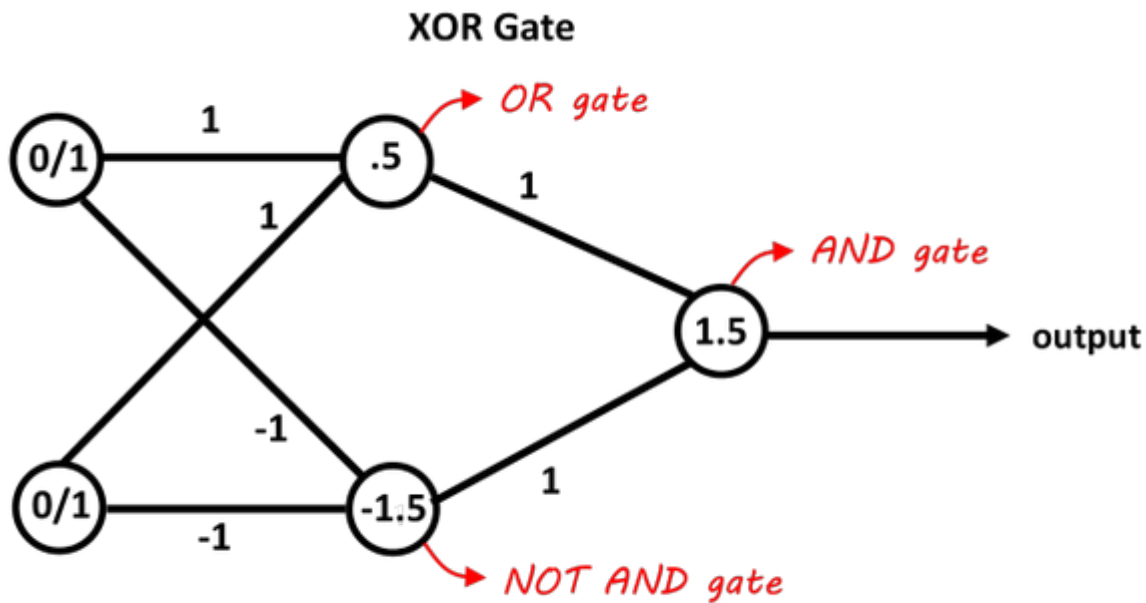


Linear separability is required in neural networks as basic operations of neural networks would be in N-dimensional space and the data points of the neural networks have to be linearly separable to eradicate the issues with wrong weight updation and wrong classifications. Linear separability of data is also considered as one of the prerequisites which help in the easy interpretation of input spaces into points whether the network is positive and negative and linearly separate the data points in the hyperplane.

Perceptrons are mainly termed as “linear classifiers” and can be used only for linear separable use cases and XOR is one of the logical operations which are not linearly separable as the data points will overlap the data points of the linear line or different classes occur on a single side of the linear line.

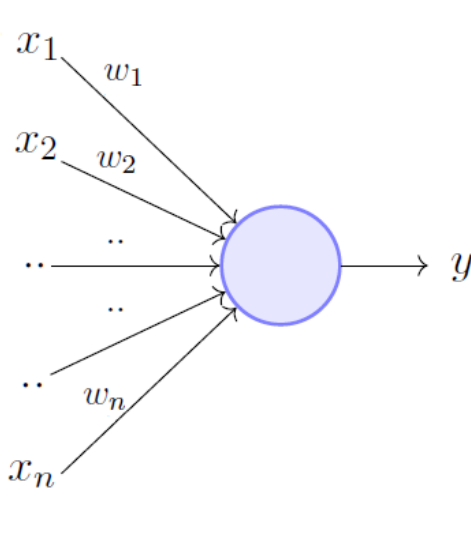
So we apply a two layer network approach by combining the OR and NAND gates.





(refer notes for calculations )

**7. State and prove Perceptron Convergence Theorem. What is the termination criteria in perceptron training if given samples are not linearly separable.**



$$y = 1 \quad \text{if } \sum_{i=1}^n w_i * x_i \geq \theta$$

$$= 0 \quad \text{if } \sum_{i=1}^n w_i * x_i < \theta$$

Rewriting the above,

$$y = 1 \quad \text{if } \sum_{i=1}^n w_i * x_i - \theta \geq 0$$

$$= 0 \quad \text{if } \sum_{i=1}^n w_i * x_i - \theta < 0$$

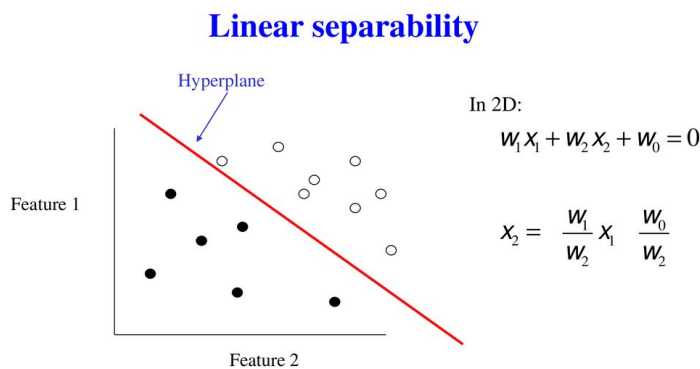
For a classifying machine, the perceptron error correction rule always converges to a solution for pattern sets that are linearly separable. This is proved by the Perceptron Convergence Theorem. The training problem for a TLN involves searching for a solution weight vector  $W_s$ . This can be done by presenting each pattern in  $X$  to the network and updating weights accordingly in response to some sequence of pattern presentations and continuing this process until for some index  $k_0$ , we have  $W_{k_0} = W_{k+1} = W_{k+2} = W_s$ .

To prove this theorem, we make a few assumptions.

Taking perceptron learning rule, observe that many patterns that are presented to the network from training sequence  $s_k$  will not cause any change of weights as they would correctly classified

in accordance to the present set of weights and deleting these patterns from  $S_k$  should have no effect on the training process since no weight perturbation happen due to the deleted patterns. So the reduced weight vector  $S_k$  has no repetitions and therefore terminates at the  $k_0$ th step if the resulting weight vector  $w_s$  satisfies when  $s_{k_0}$  terminates. Then the adaptation converges. So, given a linearly constrained training set  $X$ , initialize weight vector  $W_1$ . Let  $s_w$  be the weight vector sequence generated in response to the presentation of a training sequence  $s_x$  upon perceptron learning law. Then for some finite index  $k_0$  we have  $W_{k_0} = w_{k+1} = w_{k+2} = w_s =$  solution vector.

Perceptron Learning law :



A perceptron can separate data that is linearly separable.

Consider training set  $X = x_1, x_2, \dots, x_k$  with weights  $w = w_1, w_2, \dots, w_k$

For this network the net output  $v(n) = w(n) * x(n)$  and the actual output of the network  $y = f(v)$ .

Consider two classes  $c_1$  and  $c_2$  that are linearly separable.

The equation of the hyperplane or slope would be:

$$w_x X = 0$$

We now consider a hypothesis  $H_1$  belongs to  $C_1$  class and  $H_2$  belongs to  $C_2$  class.

Using binary threshold, the classification equation would be :

$$w_x X > 0 \text{ if } x \text{ belongs to } c_1$$

$$w_x X \leq 0 \text{ if } x \text{ belongs to } c_2.$$

Therefore the weight updation for a correct classification would be :

$$W_{k+1} = w_k \quad \text{if } w_x X > 0 \text{ and } x \text{ belongs to } C_1$$

$$W_{k+1} = w_k \quad \text{if } w_x X < 0 \text{ and } x \text{ belongs to } C_2$$

But in case of misclassification :

$$W_{k+1} = w_k - \eta X \quad \text{if } w_x X > 0 \text{ and } x \text{ belongs to } C_2$$

$$W_{k+1} = w_k + \eta X \quad \text{if } w_x X < 0 \text{ and } x \text{ belongs to } C_1$$

Perceptron convergence theorem:

Assume initial weight  $w_1 = 0$ , and  $\eta = 1$  for given two classes that are linearly separable.

The training examples  $X = x_1, x_2, \dots, x_k$  for  $n = 1, 2, 3, \dots, k$  belongs to a hypothesis  $H$

The corresponding weights  $W_{k+1} = w_1 + w_1 x_1 + w_2 x_2 + \dots, w_k x_k$

Since  $w_1 = 0$ , we get

$$W_{k+1} = x_1 + x_2 + \dots + x_k \quad (1)$$

A solution vector  $w_s$  exists such that  $x_i w_s > 0$  for every  $X_i$  belonging to  $X$

$$\text{Consider } \alpha = \min x_i w_s > 0 \quad (2)$$

Where each  $x_i > \alpha$

Taking inner product of  $w_s$  on both sides of equation (1) (2), we get :

$$W_s W_{k+1} = W_s x_1 + W_s x_2 + \dots + W_s x_k \geq n\alpha$$

Applying Cauchy Shwartz equation ,

$$\|W_s\|^2 \|W_{k+1}\|^2 \geq n^2 \alpha^2$$

$$\|W_{k+1}\|^2 \geq n^2 \alpha^2 / \|W_s\|^2 \quad (3)$$

Inequality states that the squared length of the weight vector grows at least quadratically with iteration  $k$ .

Now consider that for every  $j$ ,

$$W_{j+1} = W_j + X_j$$

We get squared euclidean normal on both sides, multiply  $w_{j+1}$  on LHS and  $w_j x_j$  on RHS

$$\|W_{j+1}\|^2 = \|W_j\|^2 + \|X_j\|^2 + 2 W_j X_j$$

So considering the case of incorrect classification,  $W_j X_j < 0$

$$\|W_{j+1}\|^2 \leq \|W_j\|^2 + \|X_j\|^2$$

$$\|W_{j+1}\|^2 - \|W_j\|^2 \leq \|X_j\|^2$$

Since  $w_1 = 0$ , terms of  $\|W_{j+1}\|^2$  gets cancelled out.

We get the final  $k$  iteration ,

$$\|W_{k+1}\|^2 \leq \sum_{j=1}^k \|X_j\|^2 \quad (4)$$

We define  $\beta$  as an upperbound, max of  $\|X_j\|^2$

$$\beta = \max \|X_j\|^2 \quad (5)$$

By comparing (4) and (5) we get result of inequality

$$\|W_{k+1}\|^2 \leq k \beta \quad (6)$$

This states that squared length of the weight vector can grow at most linearly with the iteration.

Clearly the bounds given by eqn (5) and (6) conflict for sufficiently large values of  $k$ . So we keep  $k$  no larger than some 'K' by combining the inequalities (3) and (6)

$$K\beta = K^2 \alpha^2 / \|W_s\|^2$$

$$K = \beta \|W_s\|^2 / \alpha^2$$

Therefore solution vector  $W_s$ , the fixed increment error correction procedure must terminate after at most  $K$  iterations and since every pattern in  $S_x$  occurs infinitely often in the training sequence, termination can occur only if a solution vector is found.