A Presentation on

I. PERCEPTRON

REPRESENTATION & ISSUES

CLASSIFICATION

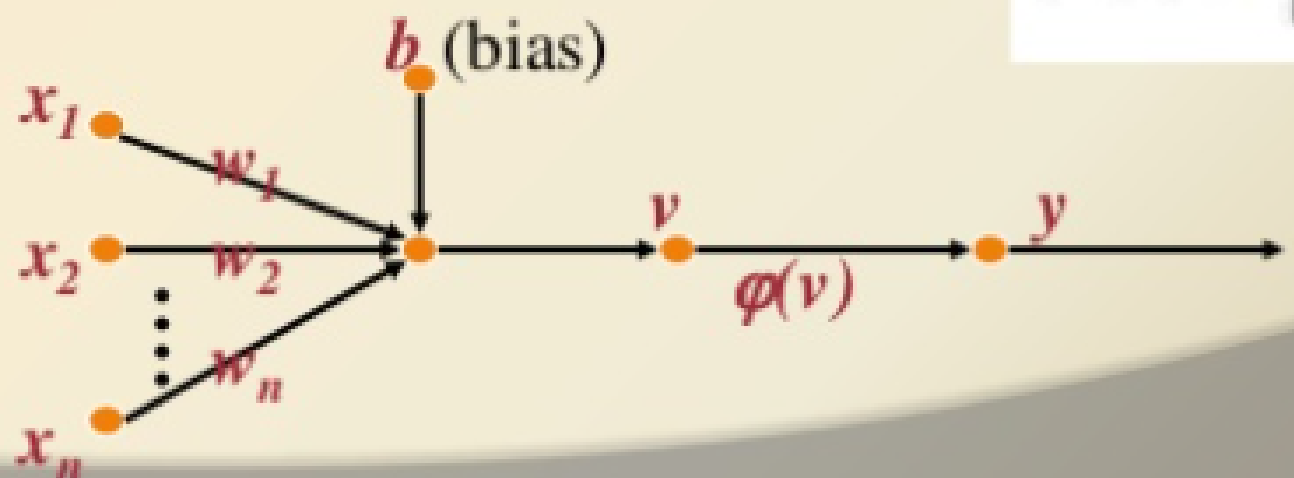
LEARNING

II. LINEAR SEPARABILITY

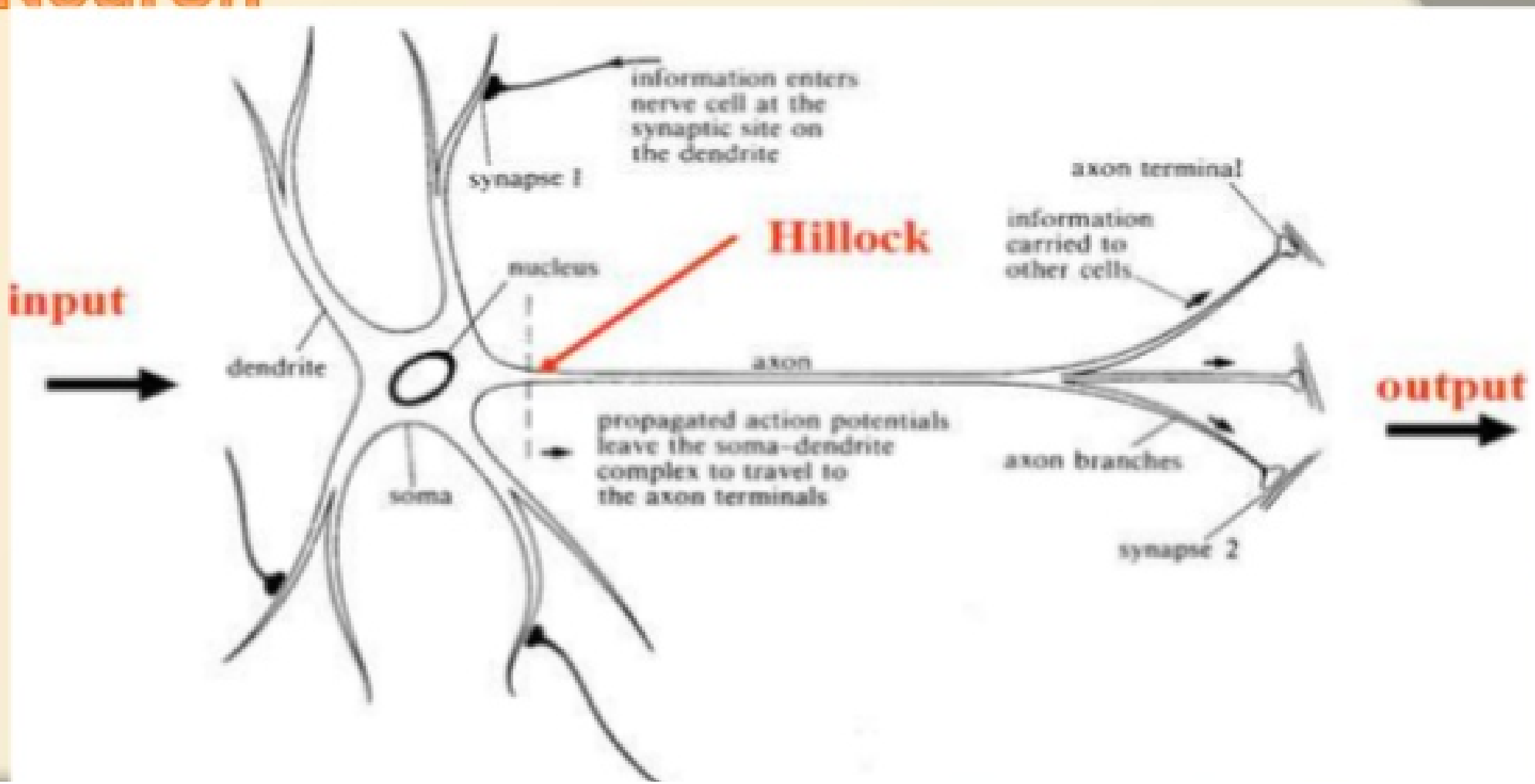
Introduction of Perceptron

- The perceptron was first proposed by Rosenblatt (1958) is a simple neuron that is used to classify its input into one of two categories.
- A perceptron is a single processing unit of a neural network. A perceptron uses a **step function** that returns +1 if weighted sum of its input ≥ 0 and -1 otherwise.

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



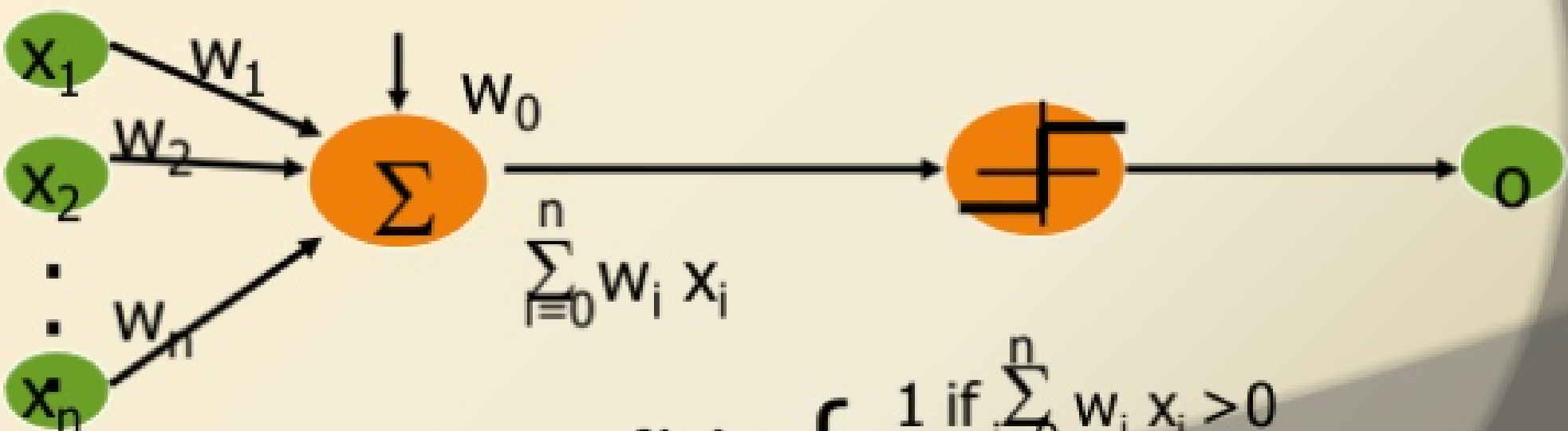
Perceptron in terms of a Biological Neuron



- While in actual neurons the dendrite receives electrical signals from the axons of other neurons, in the perceptron these electrical signals are represented as numerical values. At the synapses between the dendrite and axons, electrical signals are modulated in various amounts. This is also modeled in the perceptron by multiplying each input value by a value called the weight.
- An actual neuron fires an output signal only when the total strength of the input signals exceed a certain threshold. We model this phenomenon in a perceptron by calculating the weighted sum of the inputs to represent the total strength of the input signals, and applying a step function on the sum to determine its output. As in biological neural networks, this output is fed to other perceptrons.

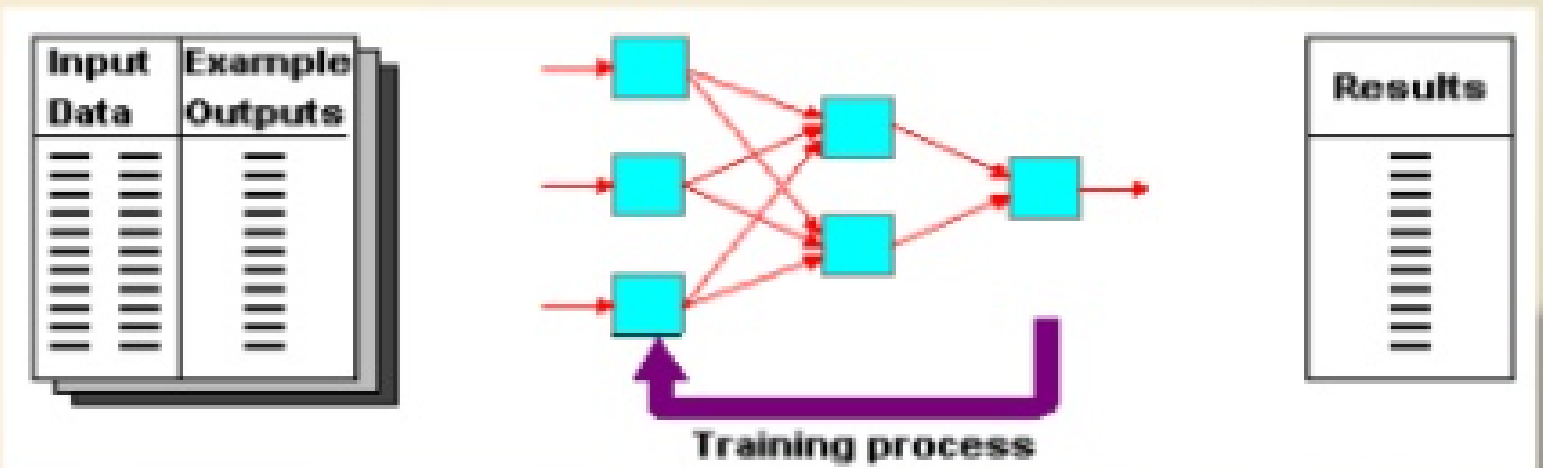
Linear Threshold Unit (LTU)

- Perceptron can be defined as a single artificial neuron that computes its weighted input with the help of the threshold activation function or step function.
- It is also called as a TLU (Threshold Logical Unit).



Definition of Supervised Learning Network

Supervised learning is used when we have a set of training data.This training data consists of some input data that is connected with some correct output values. The output values are often referred to as target values. This training data is used by learning algorithms like back propagation or genetic algorithms.

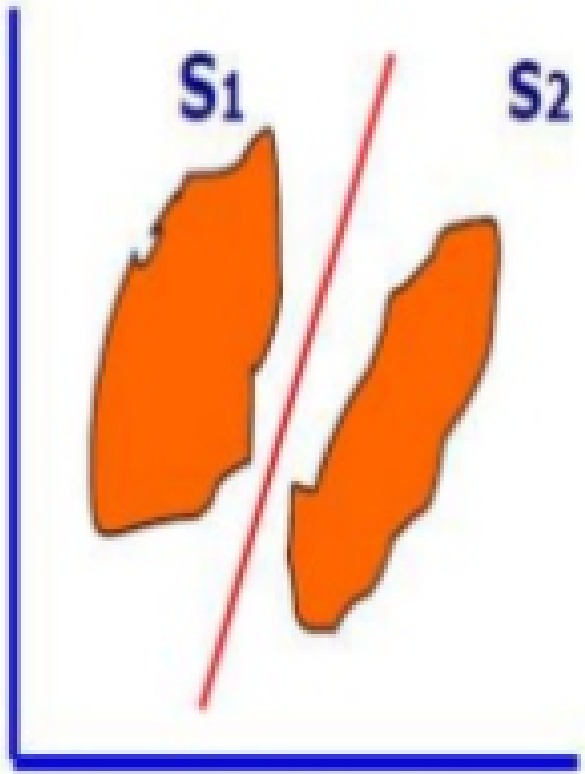


What can a Perceptron do ?

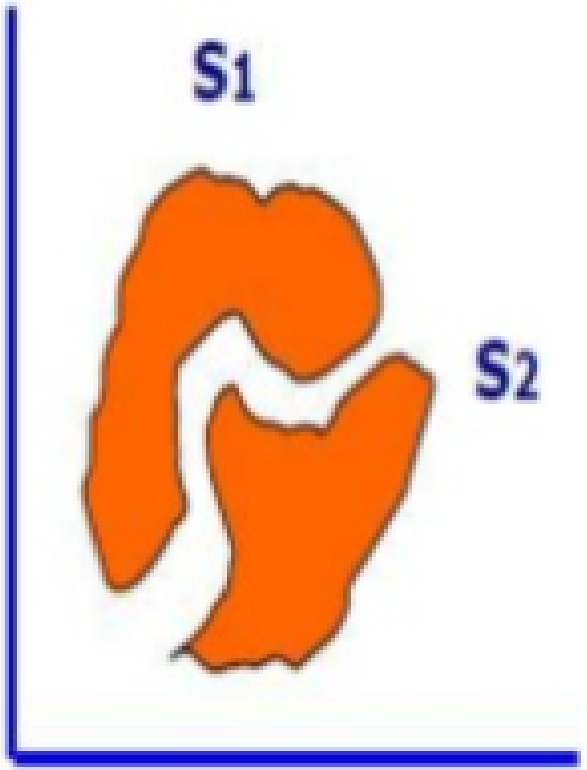
- In machine learning, the **perceptron** is an algorithm for supervised classification of an input into one of several possible non-binary outputs.
- Perceptron can be defined as a single artificial neuron that computes its weighted input with the help of the threshold activation function or step function.
- The Perceptron is used for binary Classification.
- The Perceptron can only model linearly separable classes.
- First train a perceptron for a classification task.
  - Find suitable weights in such a way that the training examples are correctly classified.
  - Geometrically try to find a hyper-plane that separates the examples of the two classes.

Linear Separability:

- Linear separability is the concept wherein the separation of the input space into regions is based on whether the network response is positiv or negative.
- When the two classes are not linearly separable, it may be desirable to obtain a linear separator that minimizes the mean squared error.
- Definition : Sets of points in 2-D space are linearly separable if the sets can be separated by a straight line.
- Generalizing, a set of points in n-dimensional space are linearly separable if there is a hyper plane of (n-1) dimensions separates the sets.

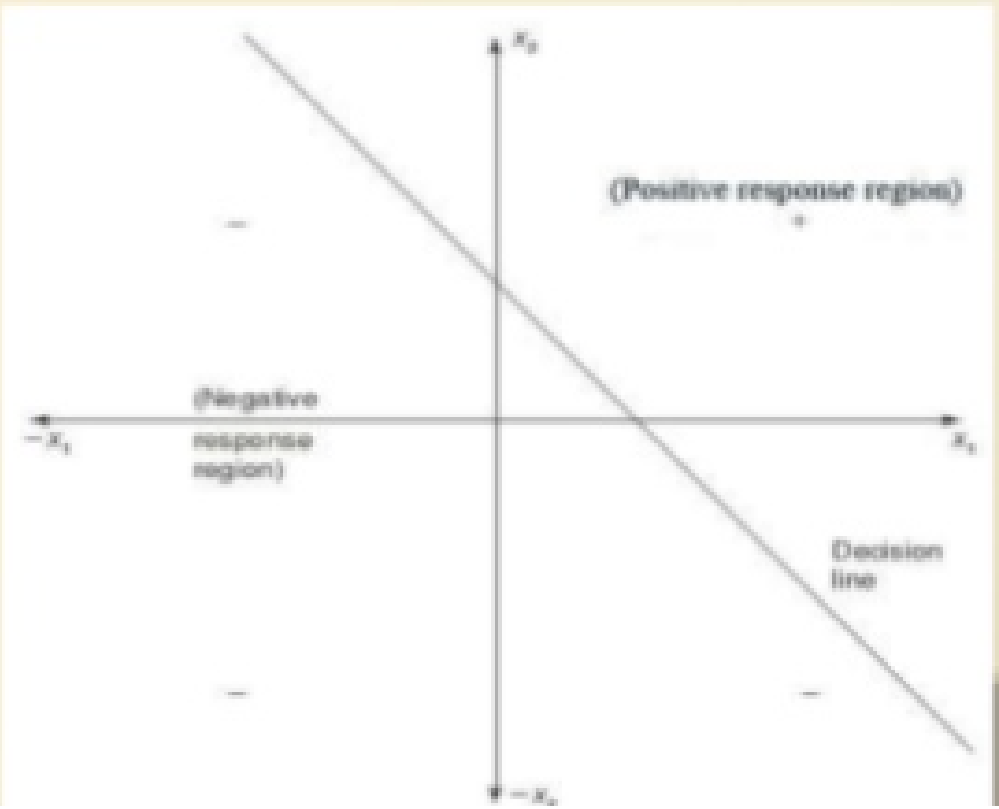


(a) Linearly separable patterns



(b) Not Linearly separable patterns

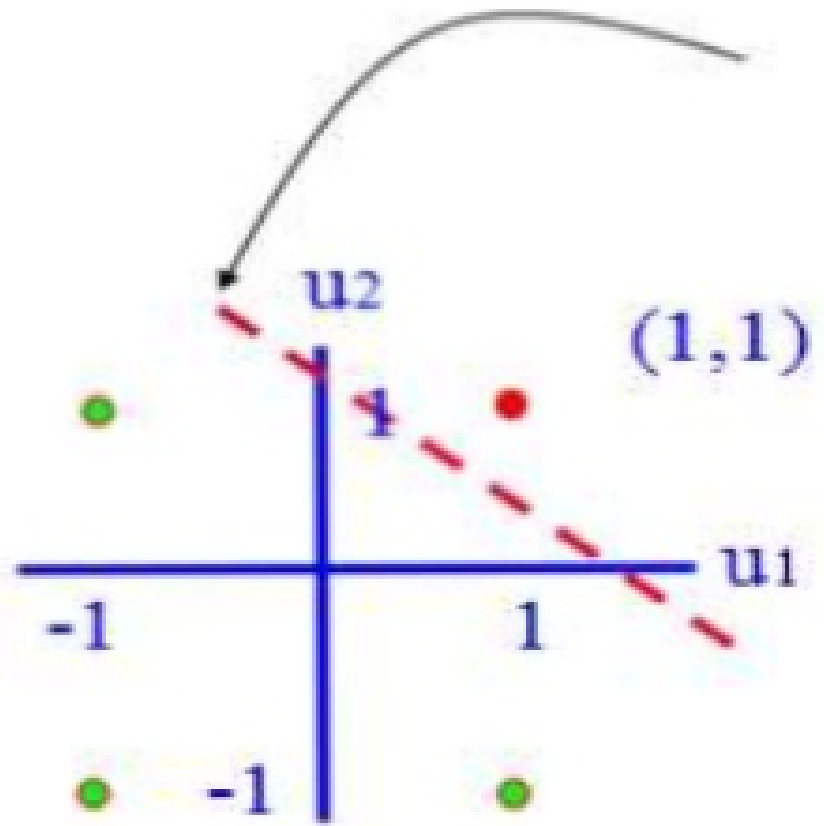
- Consider a network having positive response in the first quadrant and negative response in all other quadrants (AND function) with either binary or bipolar data, then the decision line is drawn separating the positive response region from the negative response region.

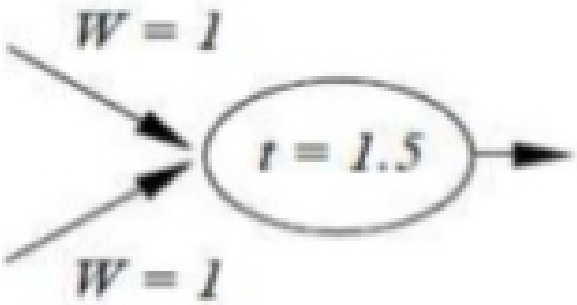


AND Gate is linearly Separable

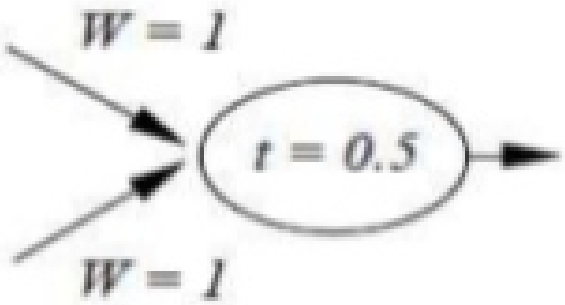
:-

u1	u2	AND
-1	-1	-1
1	-1	-1
-1	1	-1
1	1	1



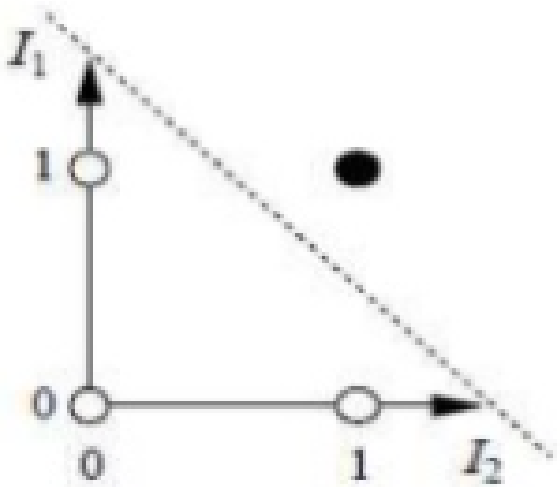


AND

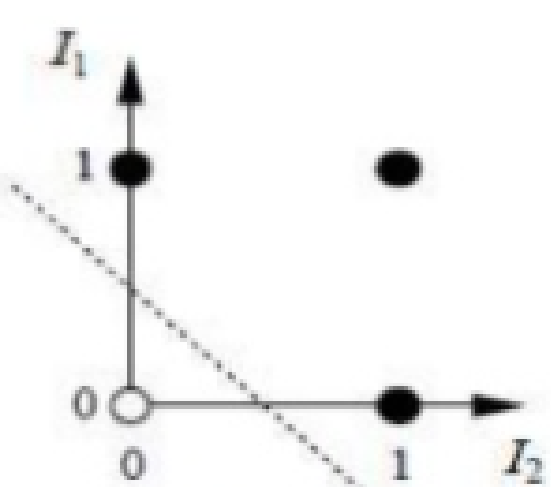


OR

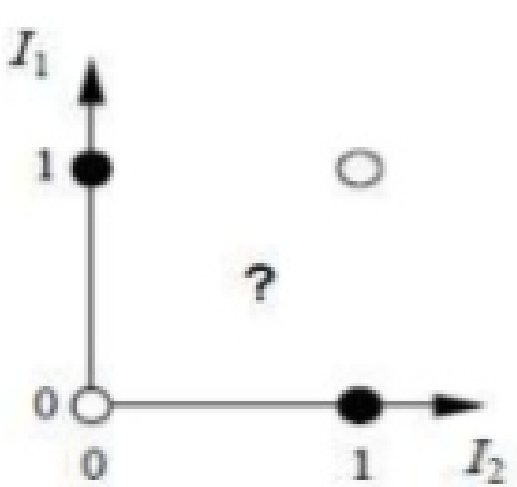
NOT



(a) I_1 and I_2



(b) I_1 or I_2



(c) I_1 xor I_2

- The net input to the output Neuron is:

$$Y_{in} = w_0 + \Sigma_i x_i w_i$$

Where Y_{in} = The net inputs to the ouput neurons.

i = any integer

w_0 = initial weight

- The following relation gives the boundary region of ne input.

$$b + \Sigma_i x_i w_i = 0$$

- The equation can be used to determine the decision boundary between the region where $Y_{in} > 0$ and $Y_{in} < 0$.
- Depending on the number of input neurons in the network. this equation represents a line, a plane or a hyper-plane.
- If it is possible to find the weights so that all of the training input vectors for which the correct response is 1. lie on the either side of the boundary, then the problem is called linearly separable.
- Otherwise. If the above criteria is not met, the problem is called linearly non-separable.

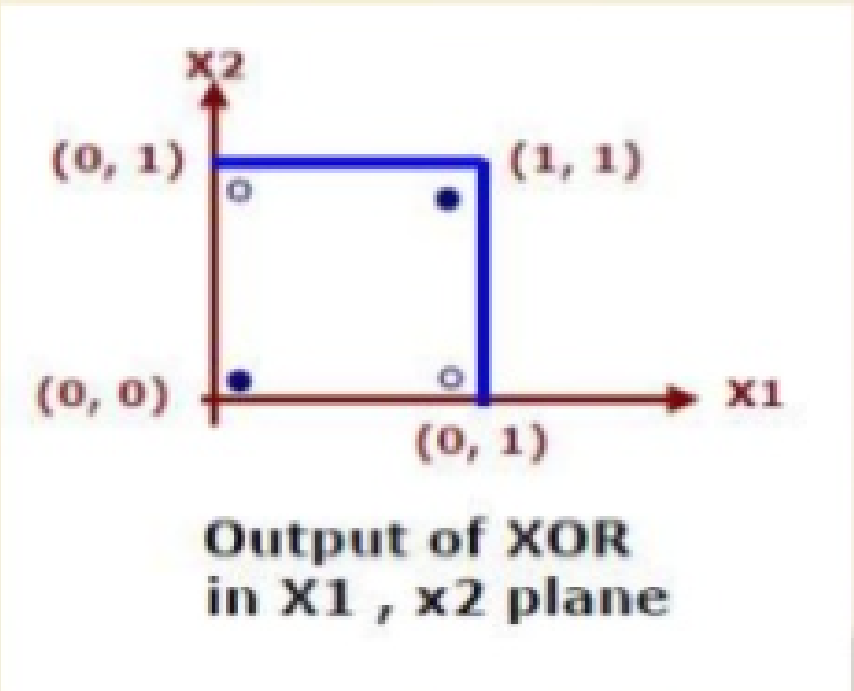
Exclusive OR (XOR) Problem :-

Input x1	Input x2	Output	
0	0	0	} Even parity •
1	1	0	
0	1	1	} Odd parity °
1	0	1	

XOR truth table

- Even parity means even number of 1 bits in the input
- Odd parity means odd number of 1 bits in the input

- There is no way to draw a single straight line so that the circles are on one side of the line and the dots on the other side.
- Perceptron is unable to find a line separating even parity input patterns from odd parity input patterns.



Limitation of Perceptron :-

- The perceptron can only model linearly separable functions,
  - those functions which can be drawn in 2-dim graph and single straight line separates values in two part.

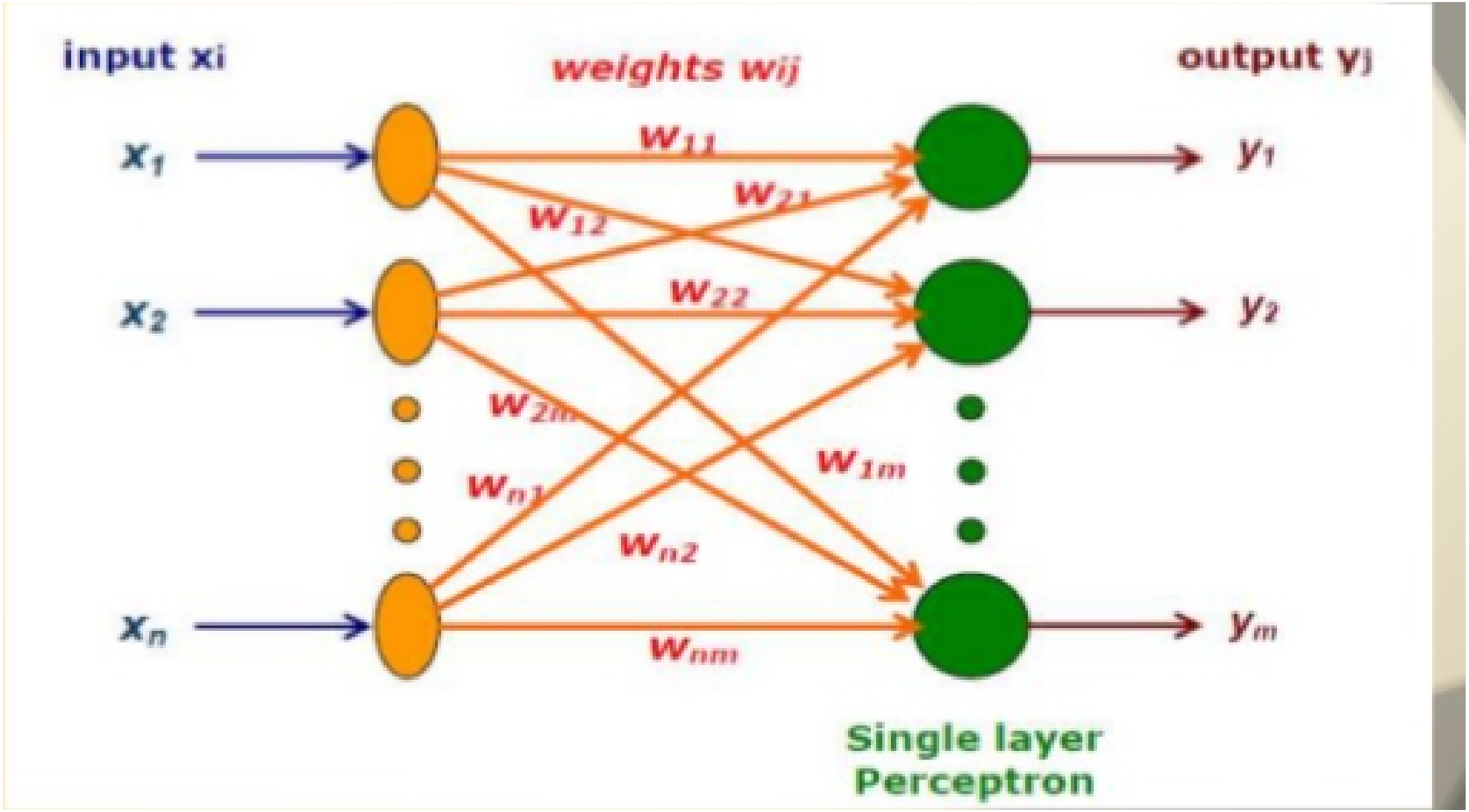
  Boolean functions given below are linearly separable:
  - AND
  - OR
  - COMPLEMENT

  It cannot model XOR function as it is non linearly separable.
  - When the two classes are not linearly separable, it may be desirable to obtain a linear separator that minimizes the mean squared error.

Single Layer Perceptron :-

- A Single Layer Perceptron consists of an input and an output layer. T activation function employed is a hard limiting function.
- Definition : An arrangement of one input layer of neurons feed forwar to one output layer of neurons is known as Single Layer Perceptron.

$$y_j = f(net_j) = \begin{cases} 1 & \text{if } net_j \geq 0 \\ 0 & \text{if } net_j < 0 \end{cases} \qquad \text{where } net_j = \sum_{i=1}^{n} x_i \, w_{ij}$$


input xi
weights wij
output yj
x1
x2
xn
w11
w21
w12
w22
w2m
wn1
w1m
wn2
wnm
y1
y2
ym
Single layer
Perceptron

Single Layer Perceptron Learning Algo :-

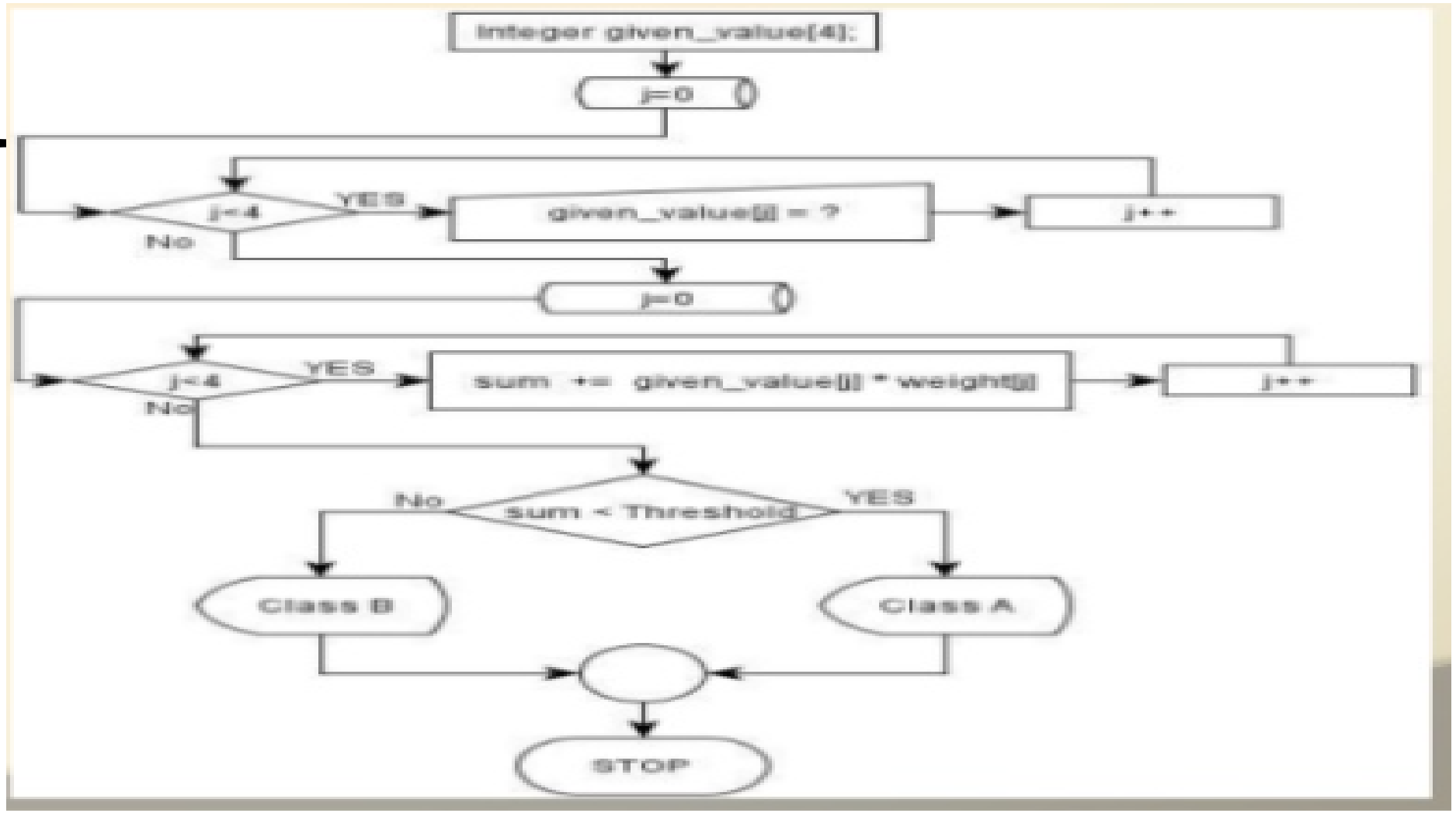
- Step 1 : Create a perceptron with (n+1) input neurons x_0 , x_1 , ... , . x_n , where x_0 = 1 is the bias input. Let O be the output neuron.
- Step 2 : Initialize weight W = (w_0, w_1, ... , . w_n) to random weights.
- Step 3 :Iterate through the input patterns x_i *of the training set using the* weight set; i.e compute the weighted sum of inputs

  $$\text{net } j = \Sigma X_i \, w_i \qquad \text{For } i=1 \text{ to } n$$

  for each input pattern j .
- Step 4 : Compute the output Y_j using the step function

$$y_j = f(\text{net}_j) = \begin{cases} 1 & \text{if } \text{net}_j \geq 0 \\ 0 & \text{if } \text{net}_j < 0 \end{cases} \qquad \text{where } \text{net}_j = \sum_{i=1}^{n} x_i \, w_{ij}$$

- Step 5 :Compare the computed output y_j with the target output y_j for each input pattern *j* .
- If all the input patterns have been classified correctly, then output (read) the weights and exit.
- Step 6 : Otherwise, update the weights as given below : If the computed outputs y_j is 1 but should have been 0,
- Then $w_i = w_i - \alpha x_i$, i= 0, 1, 2, ... , n
- If the computed outputs y_j is 0 but should have been 1,Then $w_i = w_i + \alpha x_i$, i= 0, 1, 2, ... , n
- where α is the learning parameter and is constant.
- Step 7 : goto step 3
- END


Integer given_value[4];
j=0
j<4
YES
No
given_value[j] = ?
j++
j=0
j<4
YES
No
sum += given_value[j] * weight[j]
j++
No
sum < Threshold
YES
Class B
Class A
STOP

Multi Layer Perceptron :-

- Multilayer perceptrons (MLP) are the most popular type of neural networks in use today. They belong to a general class of structures called feedforward neural networks, a basic type of neural network capable of approximating generic classes of functions, including continuous and integrable functions.
- A multilayer perceptron:

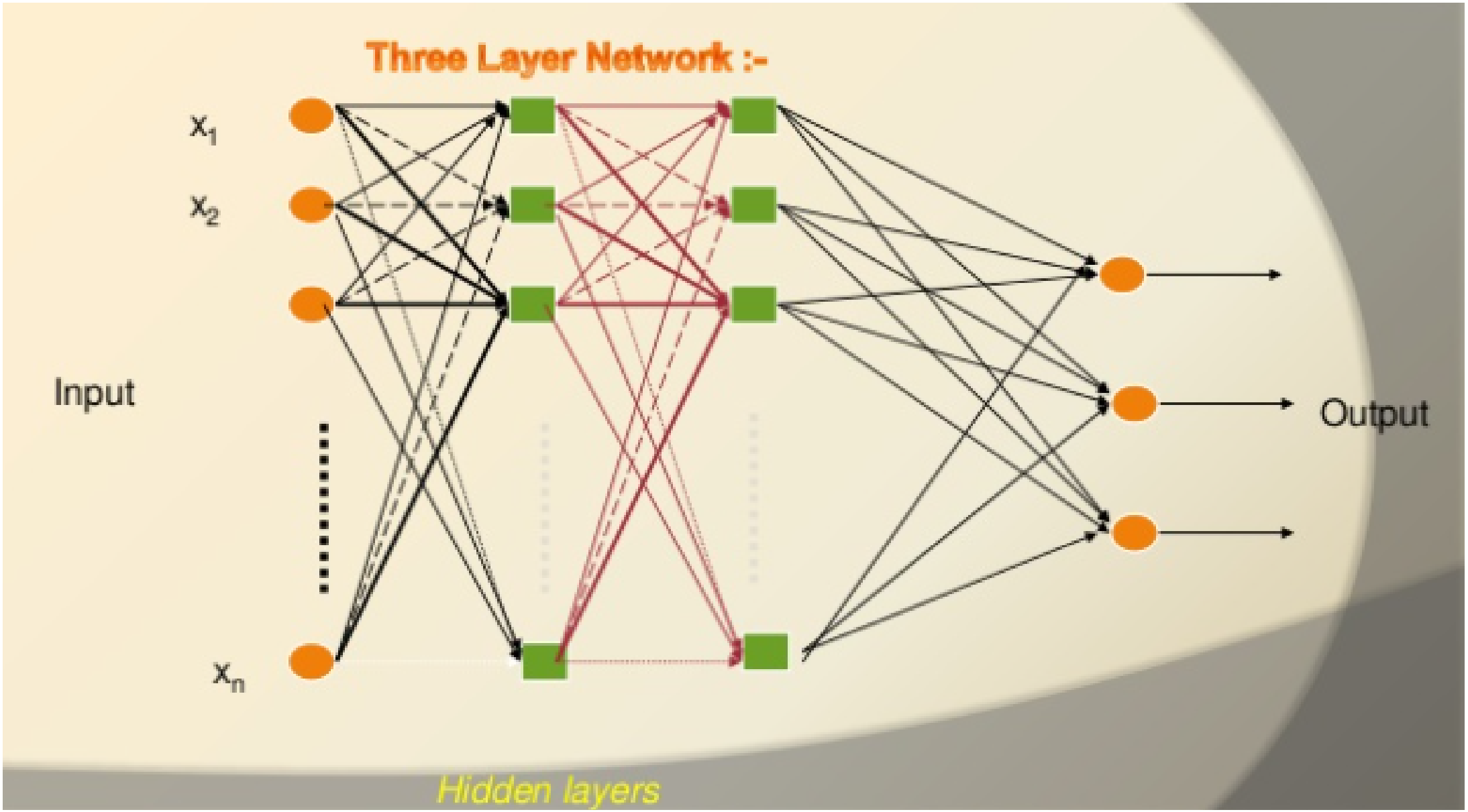
  has one or more hidden layers with any number of units.

  uses linear combination functions in the input layers.

  uses generally sigmoid activation functions in the hidden layers.

  has any number of outputs with any activation function.

  has connections between the input layer and the first hidden layer, between the hidden layers, and between the last hidden layer and the


Three Layer Network :-
x_1
x_2
x_n
Input
Output
Hidden layers

Layers in Neural Network :-

- **The input layer:**
  - Introduces input values into the network.
  - No activation function or other processing.

The hidden layer(s):

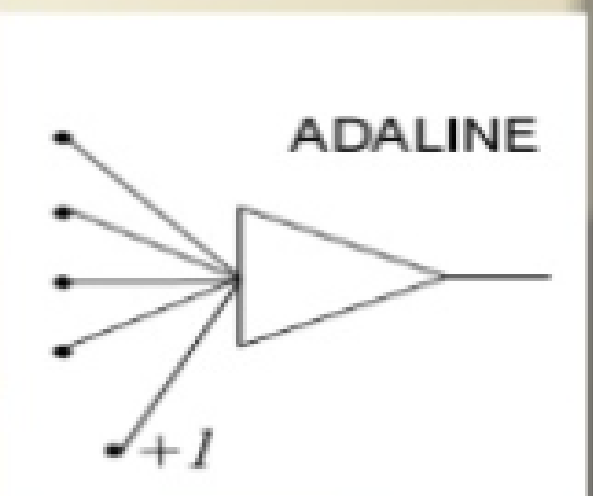
- Performs classification of features.
- Two hidden layers are sufficient to solve any problem.
- Features imply more layers may be better.

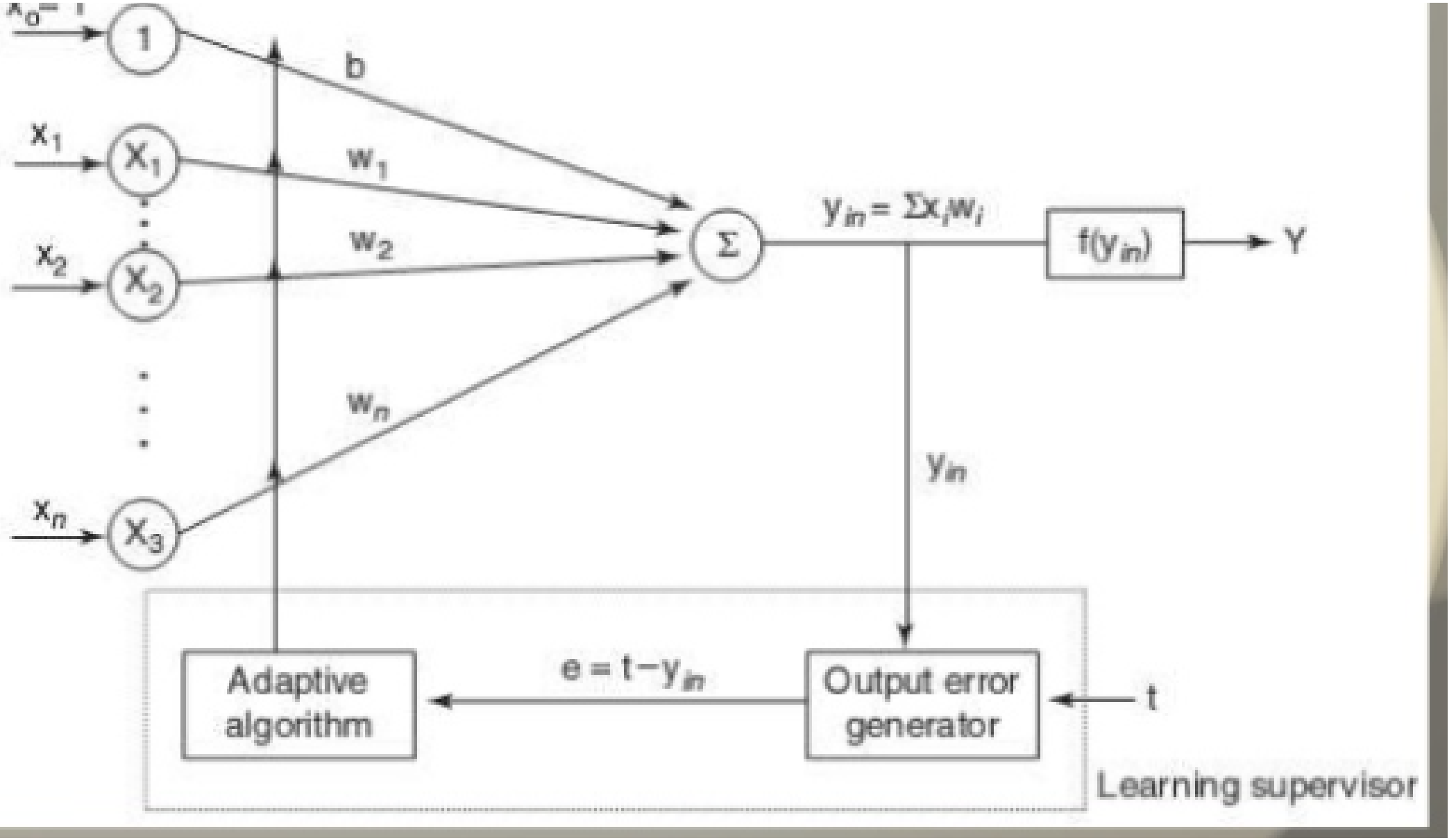
The output layer:

- Functionally is just like the hidden layers.
- Outputs are passed on to the world outside the neural network.

Adaptive Linear neuron (ADALINE):-

- In 1959, Bernard Widrow and Marcian Hoff of Stanford developed models they called ADALINE (Adaptive Linear Neuron) and MADALINE (Multilayer ADALINE). These models were named for their use of Multiple ADAptive LINear Elements. MADALINE was the first neural network to be applied to a real world problem. It is an adaptive filter which eliminates echoes on phone lines.




1
b
x_1
X_1
w_1
x_2
X_2
w_2
w_n
x_n
X_3
Σ
$y_{in} = \Sigma x_i w_i$
$f(y_{in})$
Y
y_{in}
Adaptive algorithm
$e = t - y_{in}$
Output error generator
t
Learning supervisor

Using ADALINE Network:-

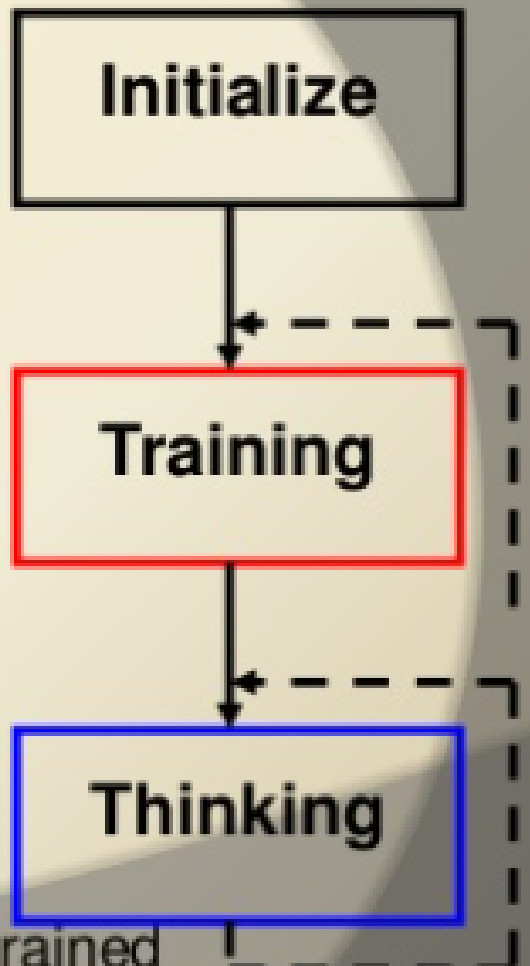
- Initialize
  - Assign random weights to all links

Training

- Feed-in known inputs in random sequence
- Simulate the network
- Compute error between the input and the output (Error Function)
- Adjust weights (Learning Function)
- Repeat until total error < ε

Thinking

- Simulate the network
- Network will respond to any input
- Does not guarantee a correct solution even for trained inputs



Perceptron Learning Algo :-

- Training patterns are presented to the network's inputs; the output is computed. Then the connection weights $\mathbf{w}_j$ are modified by an amount that is proportional to the product of the difference between the actual output, **y**, and the desired output, **d**, and the input pattern, **x**.
- The algorithm is as follows:
- Initialize the weights and threshold to small random numbers.
- Present a vector **x** to the neuron inputs and calculate the output.
- Update the weights according to:

$$w_j(t+1) = w_j(t) + \eta(d-y)x$$

- where
  - **d** is the desired output,
  - **t** is the iteration number, and
  - **eta** is the gain or step size, where $0.0 < n < 1.0$
- Repeat steps 2 and 3 until:
  - the iteration error is less than a user-specified error threshold or
  - a predetermined number of iterations have been completed.

Perceptron Training Algo :-

Let $X = X^+ \cup X^-$ be the set of training examples. and let $S_X = X_1, X_2, \ldots, X_k, \ldots$ be a training sequence on X.

Let w_k be the weight vector at step k.

Choose w_0 arbitrarily. For example. $w_0 = (0, 0, \ldots, 0)$.

Each each step k, k = 0, 1, 2, . . .

Classify X_k using w_k.

If X_k is correctly classified, take $w_{k+1} = w_k$.

If X_k is in X^- but misclassified, take $w_{k+1} = w_k - c_k X_k$.

If X_k is in X^+ but misclassified, take $w_{k+1} = w_k + c_k X_k$.

The sequence c_k should be chosen according to the data. Overly large constant values can lead to oscillation during training. Values that are too small will increase training time. However, $c_k = c_0/k$ will work for any positive c_0.

Classification of Patterns:-

- Training of Network : Given a set of inputs ‘x’, and output/targ values ‘y’, the network finds the best linear mapping from x to

- Given an unpredicted ‘x’ value, we train our network to predic what the most likely ‘y’ value will be.

- Classification of pattern is also a technique of training the network, in which we assign a physical object, event or phenomenon to one set of pre-specified classes (or categori

- Let us consider an example to illustrate the concept, with 2 inputs (x1 and x2) and 1 output node, classifying input into 2 Classes (class 0 and class 1).

Suppose we want the output of the net to be
0 for class 0 and
1 for class 1.
This means that we want
net > 0 for class 1
net < 0 for class 0
=> The dividing line between two classes is defined by net = 0