CS 321 SOFT COMPUTING

Dr. Shamama Anwar
Assistant Professor
Department of Computer Science and Engineering,
BIT, Mesra

Module - IV

Introduction to Artificial Neural Networks:

- What is a Neural Network?
- Human Brain
- Models of Neuron
- Neural Network viewed as Directed Graphs
- Feedback, Network Architecture, Knowledge Representation
- Learning processes:
    - Error correction
    - Memory-Based
    - Hebbian
    - Competitive
    - Boltzman
    - Supervised, Unsupervised
- Memory
- Adaptation

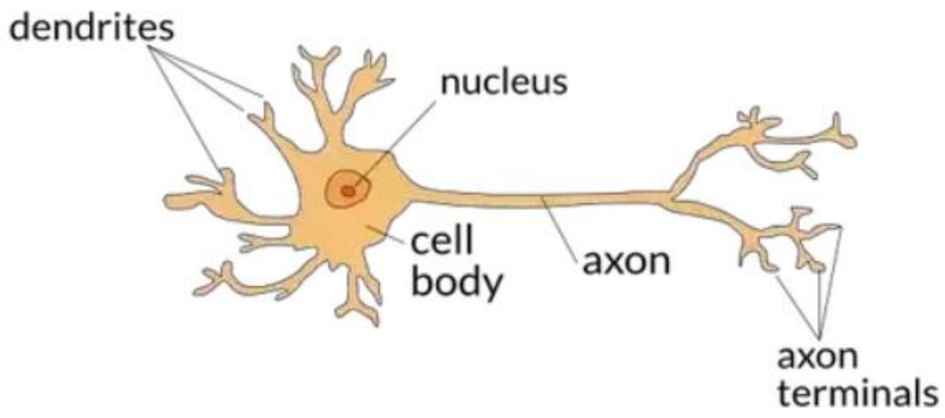
What is Neural Network ?

- A neural net is an artificial representation of the human brain that tries to simulate its learning process.
- Traditionally, neuron is used to refer to a network of biological neurons in the nervous system.
- ANN is an interconnected group of artificial neurons that uses a mathematical model or computational model for information processing.
- It is a network of simple processing elements (neurons) which can exhibit complex global behaviour, determined by the connection between the element parameters.

Human Brain

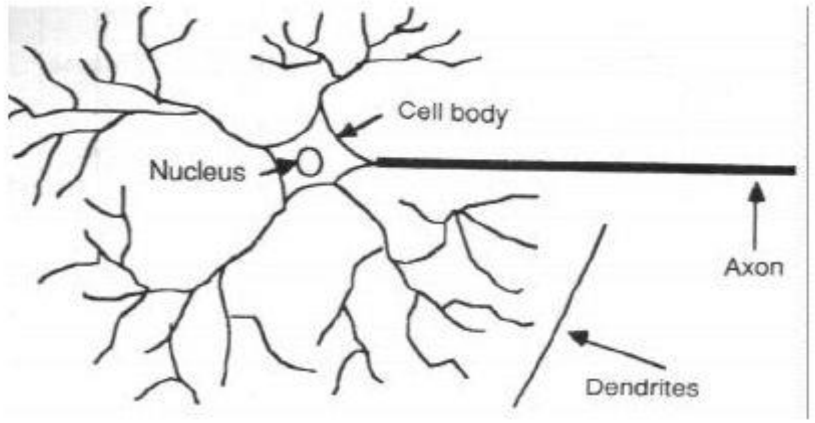
- Neurons are the fundamental constituent of the brain.
- Brain contains about 1010 basic units called neurons.
- Each neuron in turn is connected to 104 other neurons.
- A neuron is a small cell that receives electro-chemical signals and responds by transmitting electrical impulses to other neurons.
- Some neurons perform input and output operations (afferent and efferent cells) while others are a part of an interconnected network which transfer signal and are storage of information.

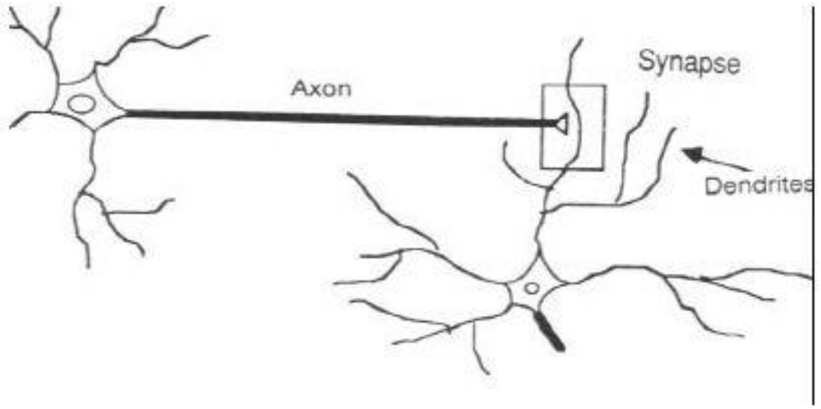
Neuron Structure



- A neuron is composed of a nucleus (Soma).
- Attached to the Soma are filaments called dendrites.
- Axon is electrically active and serves as an output channel.
- Synapse or synaptic junction connects the axon with the dendritic link of other neurons.

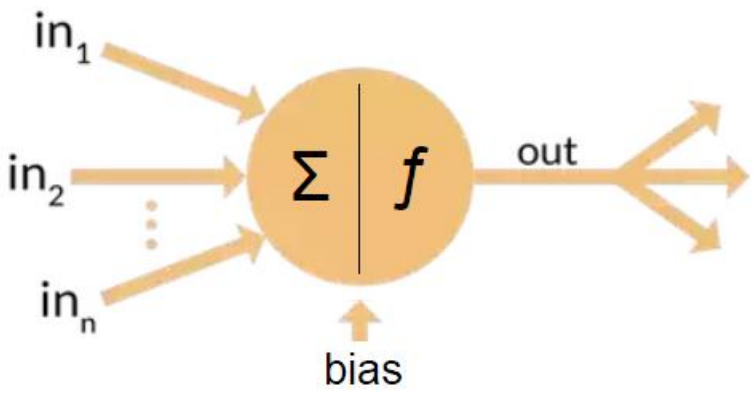
Neuron Structure

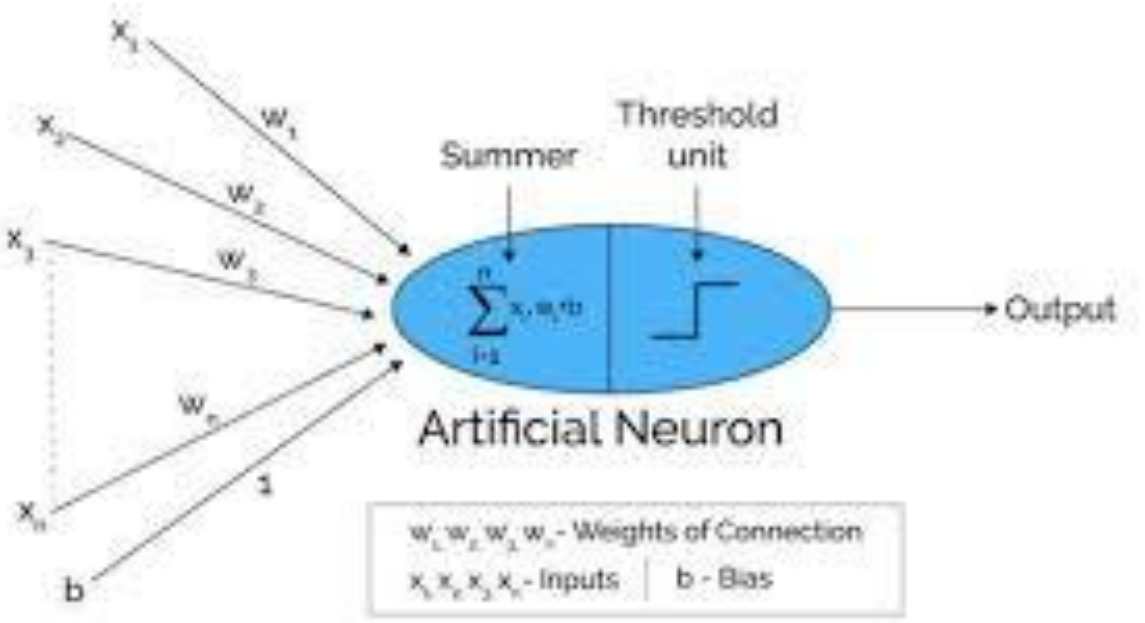




The information transmission happens at the synapses.

Artificial Neuron

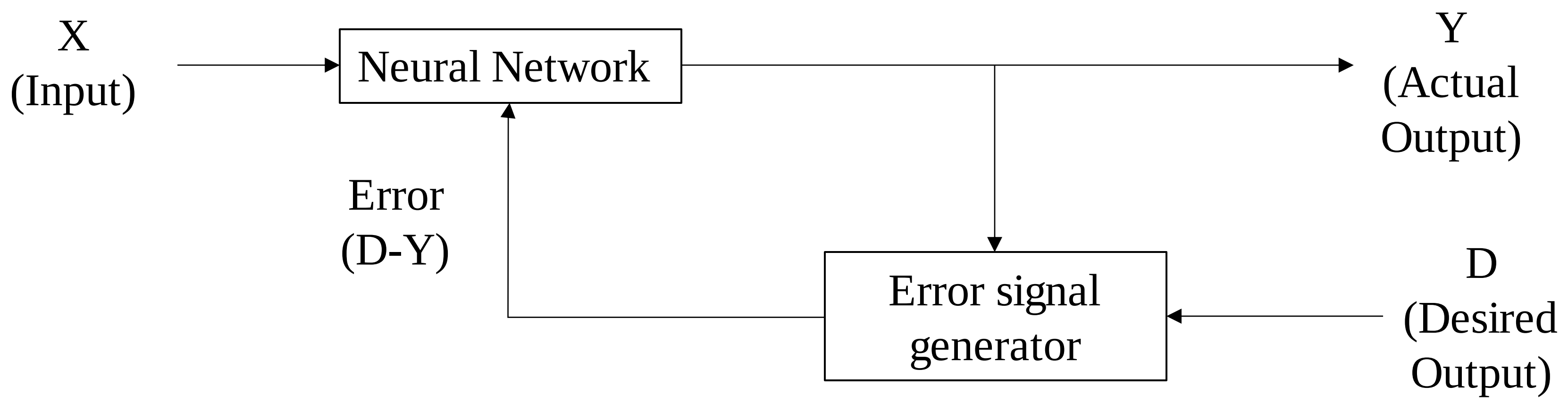




x_1
x_2
x_3
x_n
b
w_1
w_2
w_3
w_n
1
Summer
Threshold unit
$\sum_{i=1}^{n} x_i w_i + b$
Output
Artificial Neuron
w_1, w_2, w_3, w_n - Weights of Connection
x_1, x_2, x_3, x_n - Inputs
b - Bias

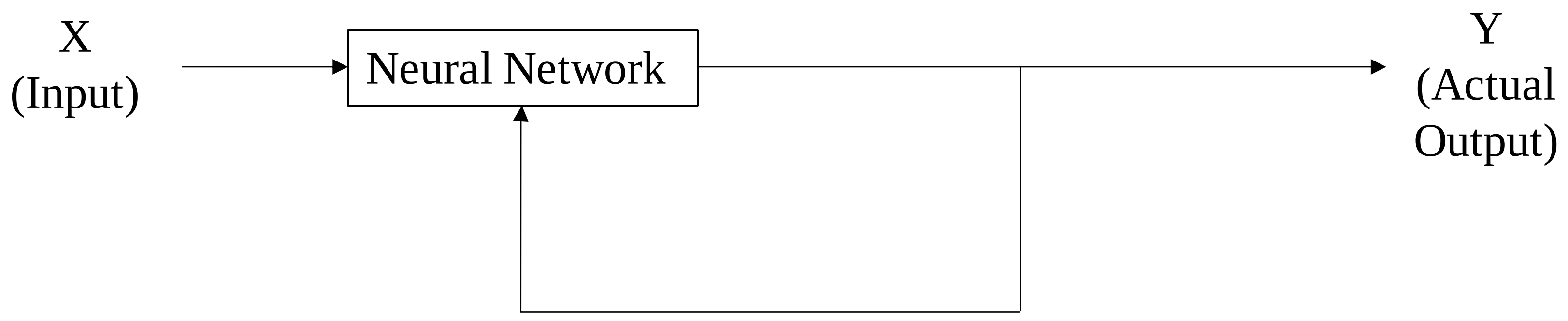
Learning

- **Supervised Learning**
  - Training is done by a training pair (input and target vector).
  - The network is informed precisely about what should be the output.



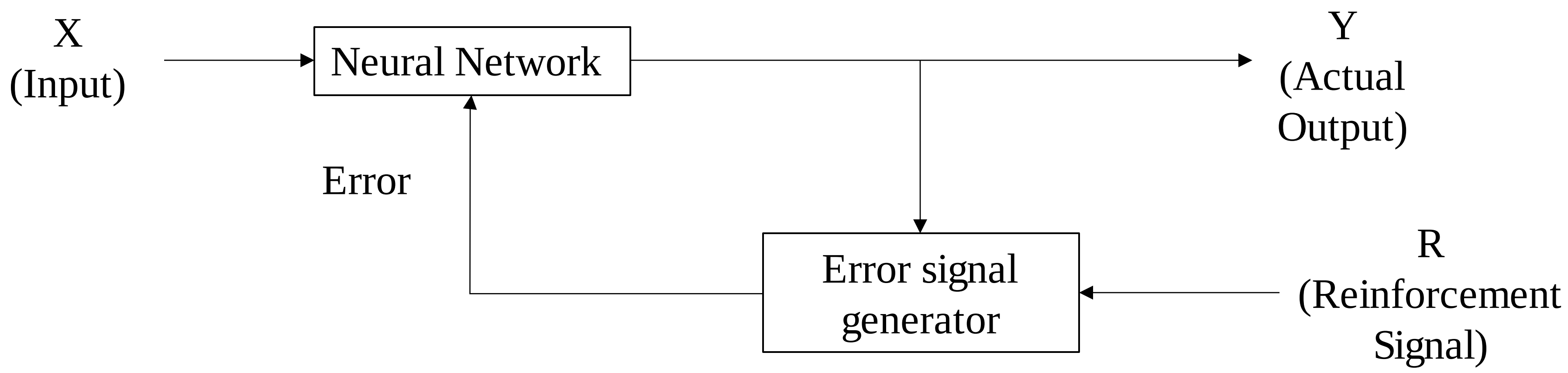
Learning

- **Unsupervised Learning**
  - Training is done by identifying similar inputs and forming groups of them.
  - The network itself must discovers patterns of similarity.



Learning

- **Reinforcement Learning**
    - Similar to supervised learning, only partial inputs maybe available i.e. only critic information is available.
    - Network learns by feedback from environment (reinforcement signal).



Activation Function

- Identity function: It is a linear function and can be defined as $f(x) = x, \forall x$. The output remains same as the input.

- Binary step function: This function can be defined as:

$$f(x) = \begin{cases} 1 \;\; if\, x \geq \theta \\ 0 \;\; if\, x < \theta \end{cases}$$ where θ represents the threshold value.

- Bipolar step function: This function can be defined as:

$$f(x) = \begin{cases} 1 \;\; if\, x \geq \theta \\ -1 \;\; if\, x < \theta \end{cases}$$ where θ represents the threshold value.

Activation Function

- Binary Sigmoidal function: Ranges between 0 to 1.

$$f(x) = \frac{1}{e^{-\lambda x}}$$ where λ is the steepness parameter.

- Bipolar Sigmoidal function: Ranges between -1 to 1.

$$f(x) = \frac{2}{e^{-\lambda x}} - 1 = \frac{1-e^{-\lambda x}}{1+e^{-\lambda x}}$$ where λ is the steepness parameter.

- Ramp function: Defined as:

$$f(x) = \begin{cases} 1 \quad if\ x > 1 \\ x \quad if\ 0 \leq x \leq 1 \\ 0 \quad if\ x < 0 \end{cases}$$

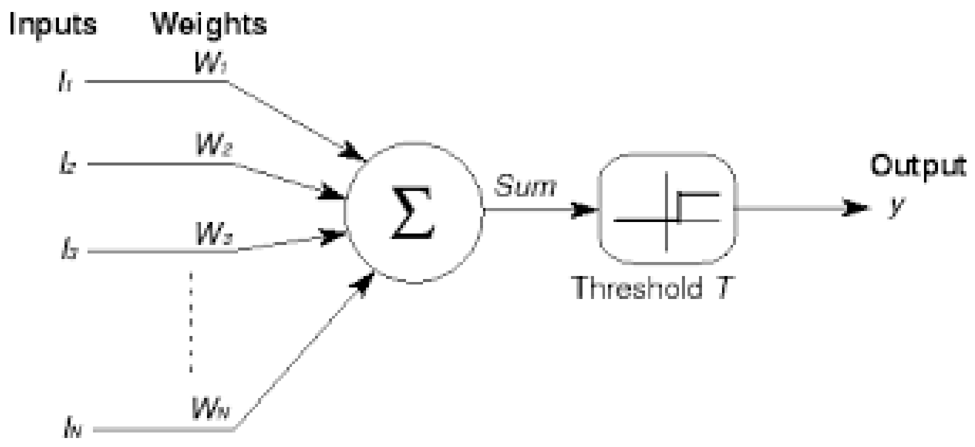
ANN Terminologies

- Weights
  - Weights contain information about the input.
  - The weight can be represented as a matrix called connection matrix.
- Bias
  - Bias is considered as an input with weight 1.
  - Bias can be either positive or negative.
- Threshold
  - It is used in the activation function
- Learning Rate
  - Denoted by α and is used to control the amount of weight adjustment at each step of training.
  - Ranges from 0 to 1.

Practice Questions

McCulloch Pitts Neuron

- M-P neuron
- The neurons are connected by directed weighted paths.
- The activation is binary.
- The weights may be excitatory (positive) or inhibitory (negative).



McCulloch Pitts Neuron

- Summation: $y_{in} = \sum_{i=1}^{n} w_i . x_i + b$
- Activation function: $f(y_{in}) = \begin{cases} 1 \;\; if\; y_{in} \geq \theta \\ 0 \;\; if\; y_{in} < \theta \end{cases}$
- The threshold should satisfy $\theta > nw \; - p$

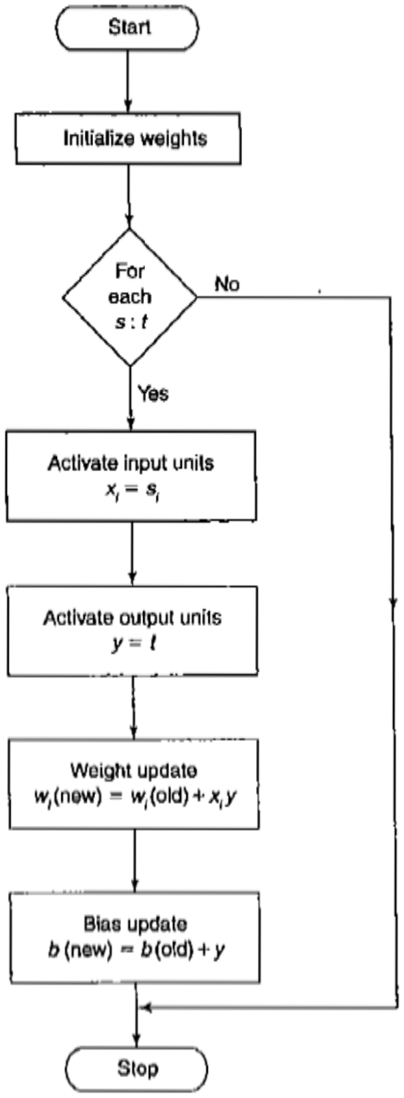
Where w is the excitatory weight and p is the inhibitory weights.

Hebb Network

- In the brain learning is performed by the change in the synaptic gap.
- In Hebb learning, if two interconnected neurons are 'on' simultaneously then the weights associated with these neurons can be increased by the modification made in their synaptic gap (strength).
- The weight update in Hebb rule is given by: $w_i\ (new) = w_i(old) + x_i y$
- The Hebb rule is more suited for bipolar data than binary data.

Hebb Network

- Steps:
  - Initialize the weights
  - Input units activation are set
  - Output units activations are set.
  - Weights adjustment and bias adjustment.



Start
Initialize weights
For each $s:t$
No
Yes
Activate input units $x_i = s_i$
Activate output units $y = t$
Weight update $w_i(\text{new}) = w_i(\text{old}) + x_i y$
Bias update $b(\text{new}) = b(\text{old}) + y$
Stop

Perceptron Network

- Single layer feed forward networks
- Consists of three units: sensory (input) unit, associator (hidden) unit, response (output) unit.
- Sensory units are connected to associator units with fixed weights having values 1, 0 or -1, which are assigned at random.
- The binary activation function is used in sensory and associator unit.
- The response unit has an activation of 1, 0 or -1.
- The output of the perceptron is given by $y = f(y_{in})$ defined as:

$$f(y_{in}) = \begin{cases} 1 \ \ if\ y_{in} > \theta \\ 0 \ \ if\ -\theta \le y_{in} \le \theta \\ -1 \ \ if\ y_{in} < -\theta \end{cases}$$

Perceptron Network

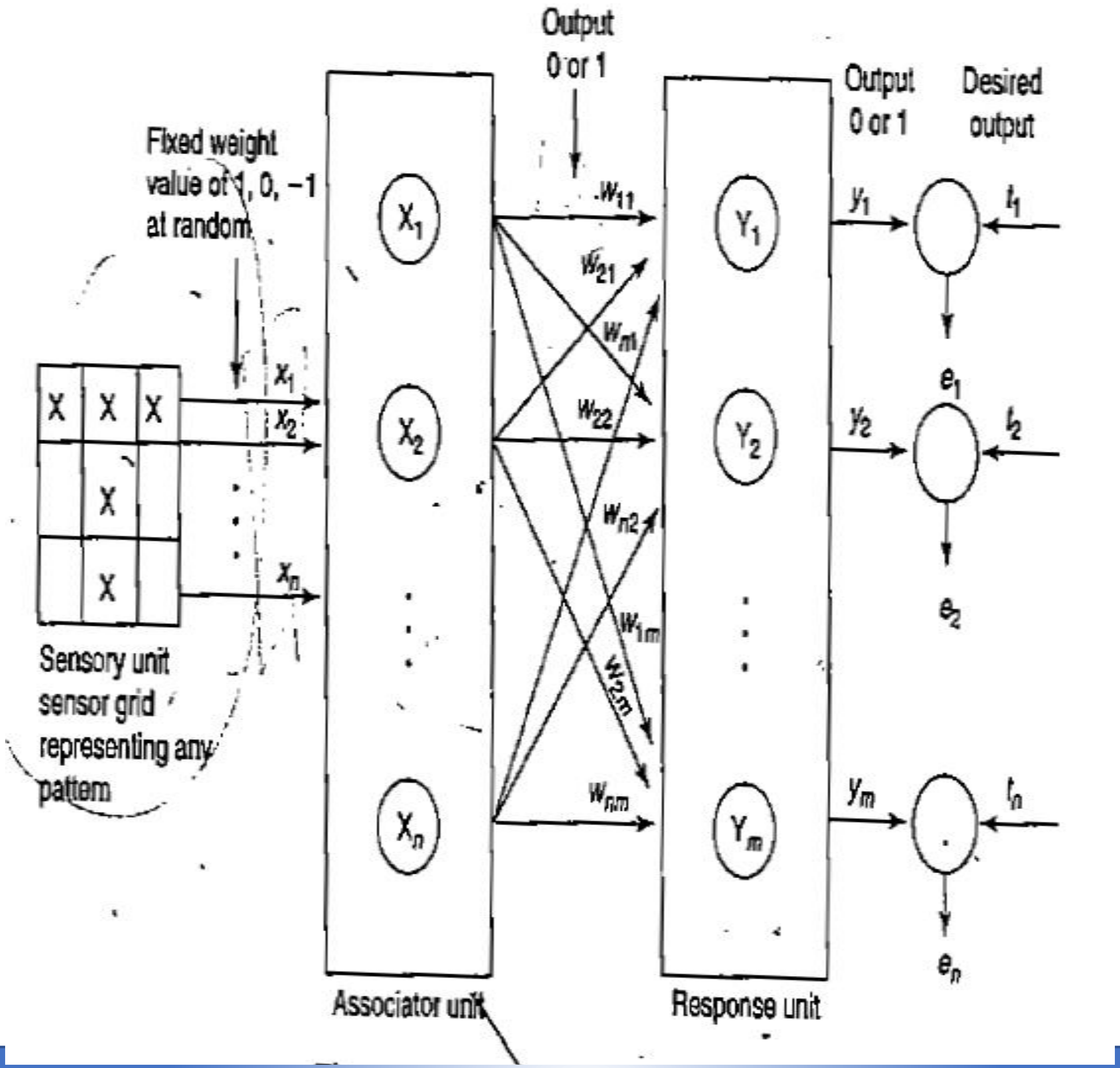
- The perceptron learning rule is used in the weight updation between the associator and response unit. For each training input, the net will calculate the response and it will determine whether or not an error has occurred.
- The error calculation is based on the comparison of the values of targets with those of the calculated outputs.
- The weights on the connections from the units that send the non zero signal will get adjusted suitably.
- The weights will be adjusted on the basis of the learning rule if an error has occurred for a particular training pattern, i.e.

$$w_i(new) = w_i(old) + \alpha t x_i$$

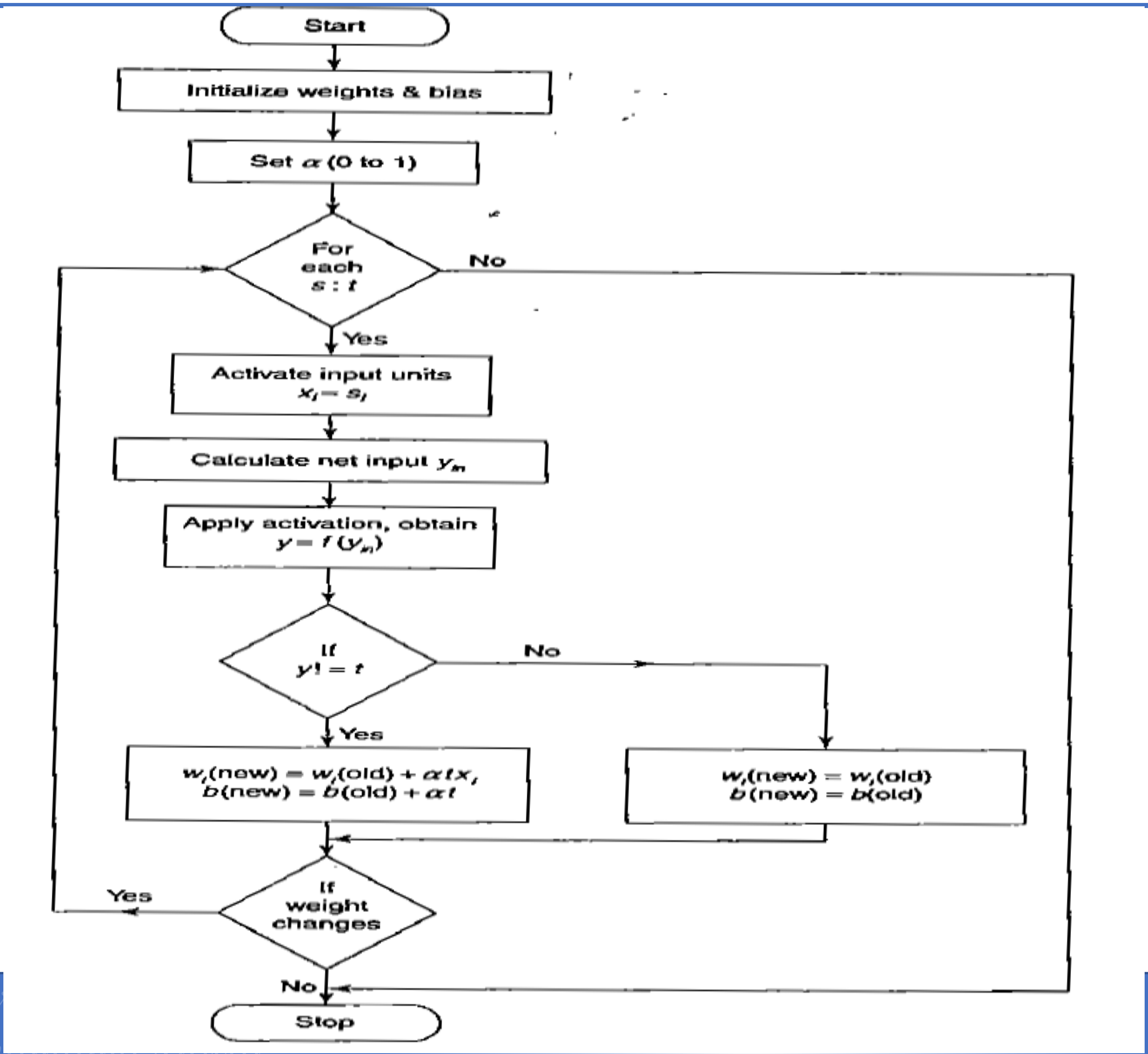
$$b(new) = b(old) + \alpha t$$

- If no error occurs, then there is no weight updation.

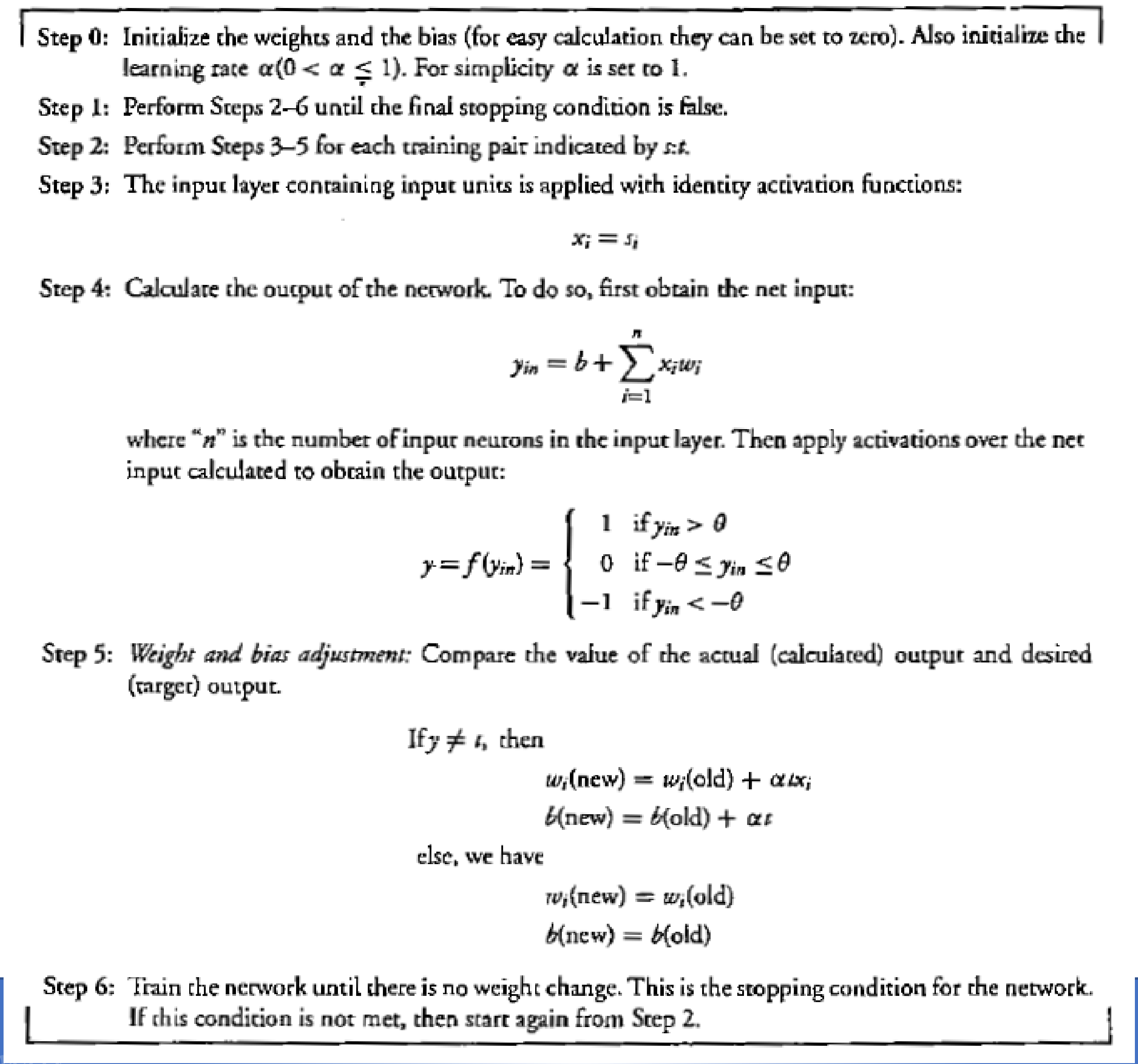
Perceptron Network



Perceptron Network


Start
Initialize weights & bias
Set α (0 to 1)
For each $s : t$
No
Yes
Activate input units $x_i = s_i$
Calculate net input y_{in}
Apply activation, obtain $y = f(y_{in})$
If $y != t$
No
Yes
$w_i(\text{new}) = w_i(\text{old}) + \alpha t x_i$
$b(\text{new}) = b(\text{old}) + \alpha t$
$w_i(\text{new}) = w_i(\text{old})$
$b(\text{new}) = b(\text{old})$
If weight changes
Yes
No
Stop

Perceptron training for single output class



Step 0: Initialize the weights and the bias (for easy calculation they can be set to zero). Also initialize the learning rate $\alpha (0 < \alpha \leq 1)$. For simplicity α is set to 1.

Step 1: Perform Steps 2–6 until the final stopping condition is false.

Step 2: Perform Steps 3–5 for each training pair indicated by *s:t*.

Step 3: The input layer containing input units is applied with identity activation functions:

$$x_i = s_i$$

Step 4: Calculate the output of the network. To do so, first obtain the net input:

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

where "*n*" is the number of input neurons in the input layer. Then apply activations over the net input calculated to obtain the output:

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \leq y_{in} \leq \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

Step 5: *Weight and bias adjustment:* Compare the value of the actual (calculated) output and desired (target) output.

If $y \neq t$, then

$$w_i(\text{new}) = w_i(\text{old}) + \alpha t x_i$$
$$b(\text{new}) = b(\text{old}) + \alpha t$$

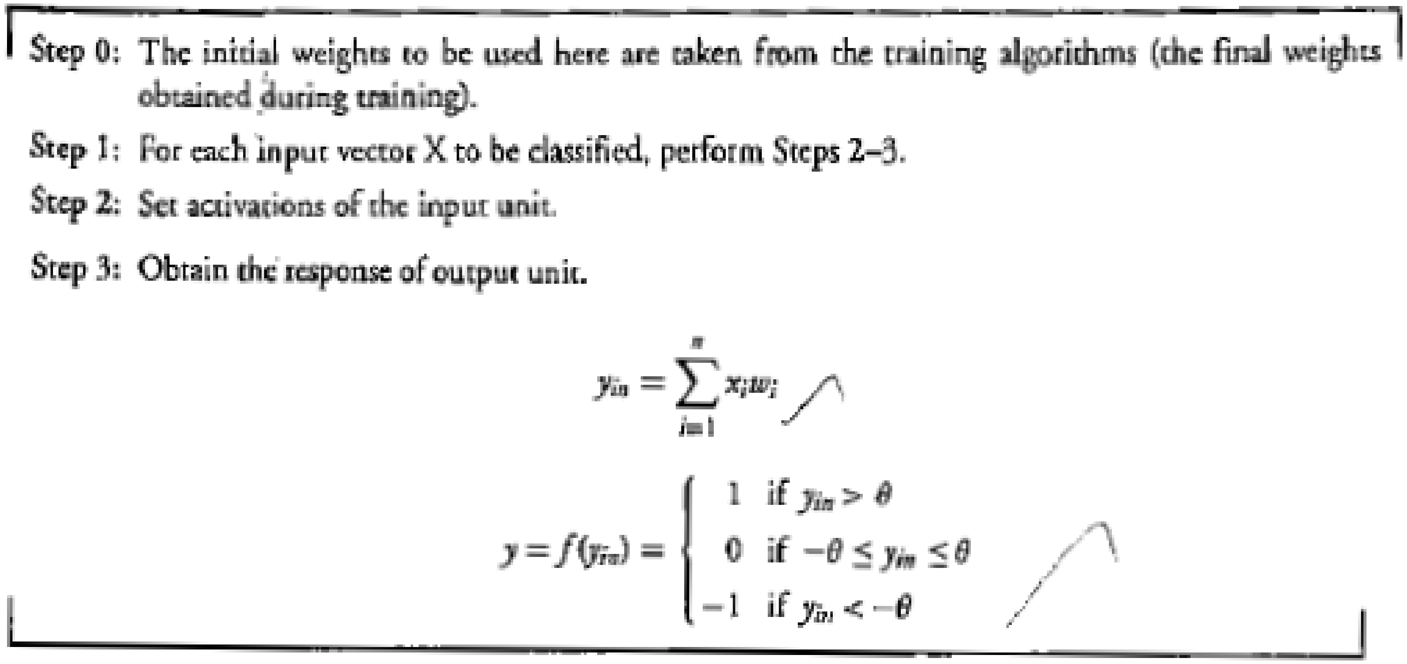
else, we have

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

Step 6: Train the network until there is no weight change. This is the stopping condition for the network. If this condition is not met, then start again from Step 2.

Perceptron training for multiple output class

Perceptron testing algorithm



Step 0: The initial weights to be used here are taken from the training algorithms (the final weights obtained during training).

Step 1: For each input vector X to be classified, perform Steps 2–3.

Step 2: Set activations of the input unit.

Step 3: Obtain the response of output unit.

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

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \le y_{in} \le \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$