Artificial Neural Network

ANN - Description

An Artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information.

It is composed of a large number of highly interconnected processing elements (neurons) working in union to solve specific problems. ANNs,

An artificial neuron is characterized by:

- Architecture (connection between neurons)
- Training or learning (determining weights on the connections)
- Activation function

Simple Neural Net

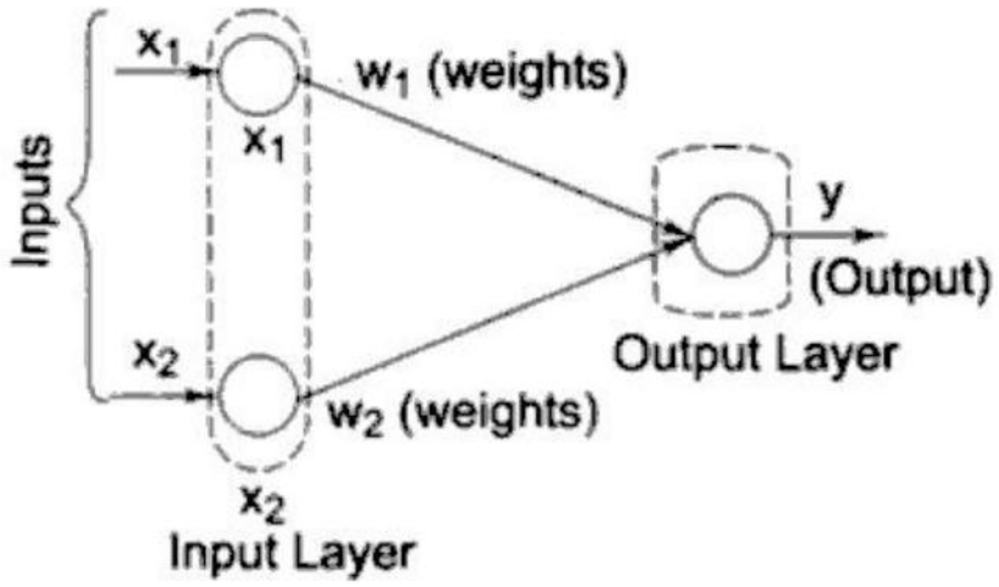


Figure shows a simple artificial neural network
with two input neurons (x1, x2) and one output neuron
(y).

The inter connected weights are given by w1 and w2

An artificial neuron is a p-input single-output signal processing element, which can be thought of as a simple model of a non-branching biological neuron.

Important Terminologies-ANN

1. **Weights:**

- Represent the strength of the connection between neurons.
- Equation:

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

Where w_i is the weight and x_i is the input.

Important Terminologies-ANN

2. Bias:

- Adjusts the output of the neuron independently of the input values.
- Equation:

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

Where b is the bias term.

Important Terminologies-ANN

- **3. Activation Function:**
- Introduces non-linearity into the model to help it learn complex patterns.
- Purpose: Determines the output of a neuron by applying a non-linear transformation.

Identity Function:

1. **Equation:** $f(z)=z$

   Summary: The simplest activation function, it returns the input as the output. It is linear and doesn't introduce any non-linearity.

Important Terminologies-Activation functions – contd.

- Binary Step Function: Outputs a binary value (0 or 1). It's used for binary classification but not suitable for multi-class problems or deep networks due to its non-differentiability.

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

Important Terminologies-ANN

- Linear Function: Similar to the identity function but with a slope (a) and intercept (b). It doesn't introduce non-linearity, limiting the network's ability to learn complex patterns.

$$f(z) = az + b$$

Important Terminologies-ANN

- Sigmoid Function: Maps input values to a range between 0 and 1. It's useful for binary classification but can suffer from vanishing gradients.

$$f(z) = \frac{1}{1+e^{-z}}$$

Important Terminologies-ANN

- Tanh (Hyperbolic Tangent) Function:Maps input values to a range between -1 and 1. It has zero-centered outputs, which can help with optimization, but also suffers from vanishing gradients.

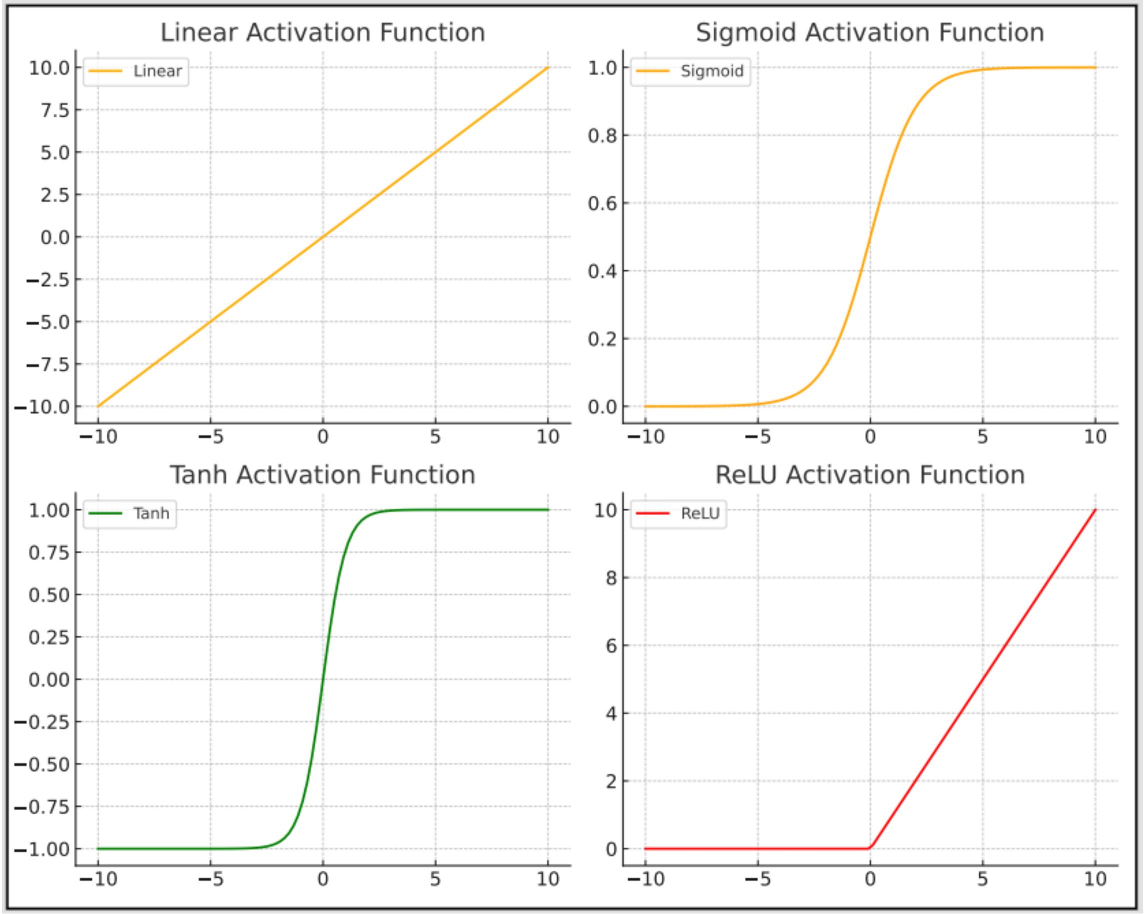
$$f(z) = \tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Important Terminologies-ANN

- ReLU (Rectified Linear Unit):Introduces non-linearity by outputting the input directly if it's positive; otherwise, it outputs zero. It's widely used due to its simplicity and effectiveness.

$$f(z)=\begin{cases} z & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Visualization of Activation Functions



Summary of Activation Functions

- **Identity:** Outputs a straight line.
- **Binary Step:** Outputs 0 or 1 based on the input threshold.
- **Linear:** Outputs a straight line.
- **Sigmoid:** S-shaped curve, outputs between 0 and 1.
- **Tanh:** Outputs between -1 and 1.
- **ReLU:** Outputs zero for negative inputs, linear for positive inputs.

Example 1

- For the network shown in Figure I, calculate the net input to the output neuron.

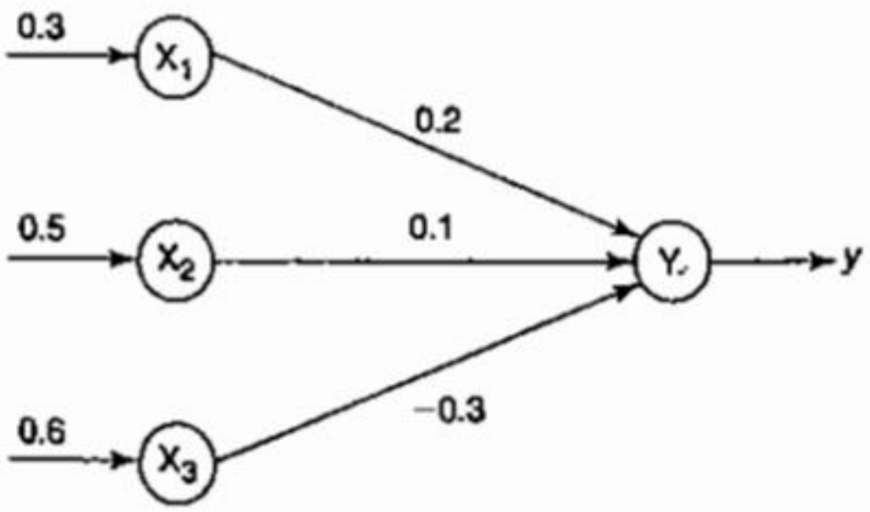
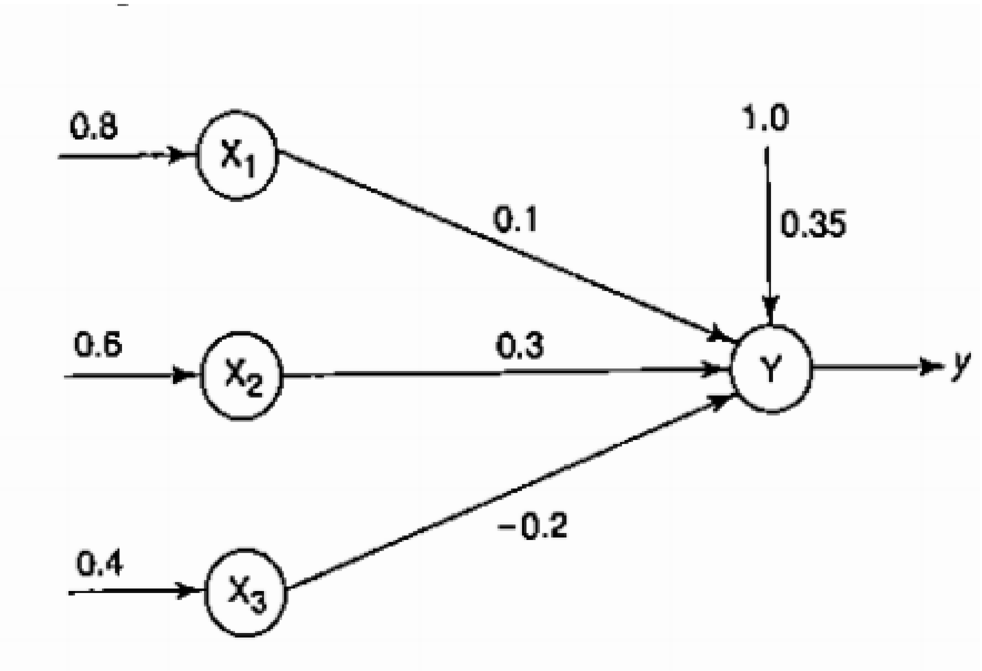


Figure 1 Neural net.

Example 2

Obtain the output of the neuron Y for the network shown in figure 2 using binary sigmoidal activation function.



First – ANN (McCulloch-Pitts Neuron) MP Neuron

- The McCulloch-Pitts (M-P) neuron is a simple model of an artificial neuron introduced by Warren McCulloch and Walter Pitts in 1943.

- It serves as the foundation of artificial neural networks and is used to simulate logical operations like AND, OR, and NOT gates.

First – ANN (McCulloch-Pitts Neuron)
MP Neuron

Key Features of M-P Neuron

1. **Input**: Receives binary inputs $x_1, x_2, \ldots, x_n$ (values 0 or 1).
2. **Weights**: Each input has a weight $w_1, w_2, \ldots, w_n$ (importance of inputs).
3. **Summation**: Computes a weighted sum $\mathrm{Sum} = w_1x_1 + w_2x_2 + \cdots + w_nx_n$.
4. **Threshold**: If the weighted sum exceeds a threshold θ, the neuron "fires" (output = 1); otherwise, it doesn't fire (output = 0).
5. **Output**: Binary output $y = 1$ (neuron fires) or $y = 0$ (does not fire).

The activation function can be defined as:

$$y = \begin{cases} 1 & \text{if } \mathrm{Sum} \geq \theta \\ 0 & \text{if } \mathrm{Sum} < \theta \end{cases}$$

First – ANN (McCulloch-Pitts Neuron) MP Neuron

1. Excitatory Weights

- **Nature**: Positive weights that encourage the neuron to "fire" (produce an output of 1).
- **Effect**: If an input has an excitatory weight, it contributes positively to the weighted sum, increasing the chances of the neuron reaching or exceeding the threshold.
- **Biological Analogy**: Excitatory synapses in biological neurons increase the likelihood of a neuron firing.

2. Inhibitory Weights

- **Nature**: Negative weights that prevent the neuron from firing (output = 0).
- **Effect**: If any input with an inhibitory weight is activated (value = 1), it dominates the output, forcing it to **0**, regardless of other excitatory inputs.
- **Biological Analogy**: Inhibitory synapses in biological neurons suppress or stop the firing of neurons.

MP- Neuron // Threshold Logic Units

Using M-P Neuron for Binary Logic Gates

1. AND Gate

- **Logic**: Output is 1 only if both inputs are 1.
- **Truth Table**:

x_1	x_2	**Output**
0	0	0
0	1	0
1	0	0
1	1	1

MP Neuron for And Gate

- M-P Neuron Parameters:
  - Weights $w_1 = 1$, $w_2 = 1$
  - Threshold $\theta = 2$

Explanation:

- Compute $\text{Sum} = w_1 x_1 + w_2 x_2 = x_1 + x_2$.
- If $\text{Sum} \geq 2$, output $y = 1$ (both inputs must be 1).

Example:

- $x_1 = 1, x_2 = 1$:

$$\text{Sum} = 1 \cdot 1 + 1 \cdot 1 = 2 \quad (\text{Sum} \geq 2,\ y = 1)$$

- $x_1 = 1, x_2 = 0$:

$$\text{Sum} = 1 \cdot 1 + 1 \cdot 0 = 1 \quad (\text{Sum} < 2,\ y = 0)$$

HW Question
Try to design an MP neuron for NOT gate using inhibitory weight and suitable threshold value

Limitations of MP Neuron

- 1. Lack of Learning Capability
- 2. Binary Nature of Inputs and Outputs
- 3. Linear Separability Limitation
- 4. Fixed Activation Function
- 5. Manual Configuration of Parameters

Hebb Network and Hebb Learning Rule

Hebb Learning Rule

The Hebb learning rule is summarized as:

> **"Cells that fire together, wire together"**

This means that if two neurons (a presynaptic neuron and a postsynaptic neuron) are activated simultaneously (fire together), the connection (synapse) between them is **strengthened**.

Hebb Network

A **Hebb network** is a simple neural network that uses Hebb's learning rule for weight updates. It typically includes:

- **Input neurons** and **output neurons**
- **Weights** connecting the input neurons to the output neurons
- Learning is based on Hebb's rule, where the weights are adjusted based on the product of inputs and outputs.

Hebb Learning - Algorithm

Steps in Hebb Learning

1. **Initialize Weights:**

   Start with small random or zero weights.

2. **Present Input Patterns:**

   Feed the input vector x to the network.

3. **Compute the Output:**

   Output y is calculated as:

   $$y = \sum_i w_i x_i$$

4. **Update Weights:**

   Update the weights using Hebb's rule:

   $$w_i^{\text{new}} = w_i^{\text{old}} + \eta \cdot x_i \cdot y$$

5. **Repeat** for all input patterns.

Ex.3

- Design a Hebb net to implement logical AND function (use bipolar inputs and targets).

Inputs			Target
x_1	x_2	b	y
1	1	1	1
1	−1	1	−1
−1	1	1	−1
−1	−1	1	−1

Ex.3

- Design a Hebb net to implement logical AND function (use bipolar inputs and targets).

Inputs			Target
x_1	x_2	b	y
1	1	1	1
1	−1	1	−1
−1	1	1	−1
−1	−1	1	−1

Inputs				Weight changes			Weights		
x_1	x_2	b	y	Δw_1	Δw_2	Δb	w_1	w_2	b
							(0	0	0)
1	1	1	1	1	1	1	1	1	1
1	−1	1	−1	−1	1	−1	0	2	0
−1	1	1	−1	1	−1	−1	1	1	−1
−1	−1	1	−1	1	1	−1	2	2	−2

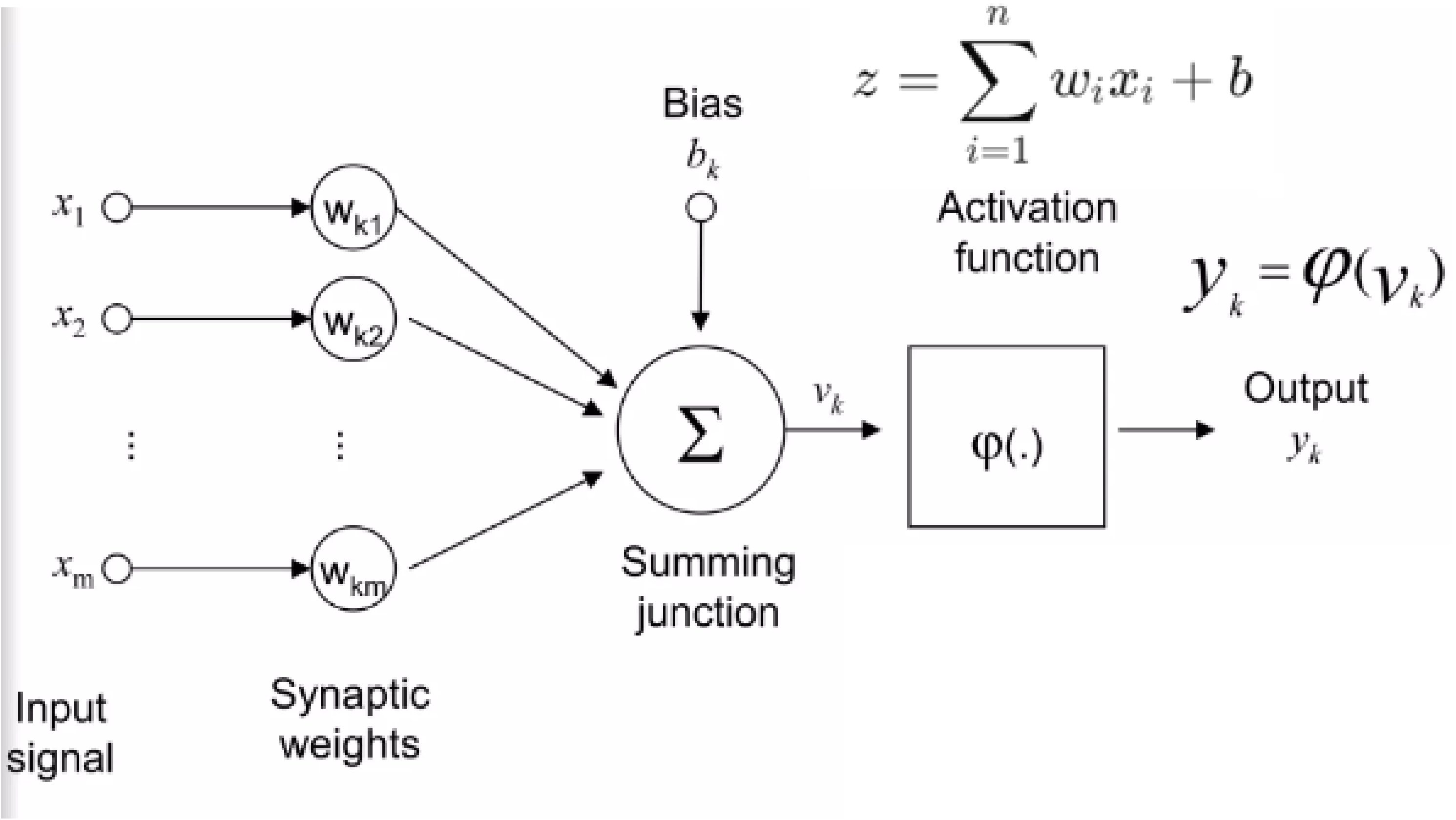
Perceptron Networks

- A Perceptron is one of the simplest types of artificial neural networks, introduced by Frank Rosenblatt in 1958.

- It is a foundational model for modern neural networks, widely used for binary classification tasks.

- The perceptron is a single-layer neural network and is often considered a step beyond the McCulloch-Pitts (MP) Neuron,

Single Layer Perceptron – One output Neuron



Components of Perceptron

A perceptron consists of the following components:

1. **Inputs** ($x_1, x_2, \ldots, x_n$): Feature values or signals from the environment.
2. **Weights** ($w_1, w_2, \ldots, w_n$): Real-valued parameters associated with each input. They determine the importance of each input.
3. **Bias** (b): A constant term that shifts the decision boundary.

Components of Perceptron

4. **Summation Function (z):** Calculates the weighted sum of inputs:

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

5. **Activation Function:** A step function that determines the output of the perceptron:

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

$$y = \begin{cases} 1 \;\; if \;\; Z > \theta \\ 0 \;\; if \;\; Z = \theta \\ -1 \;\; if \;\; Z < \theta \end{cases}$$

Perceptron – Weight Updation

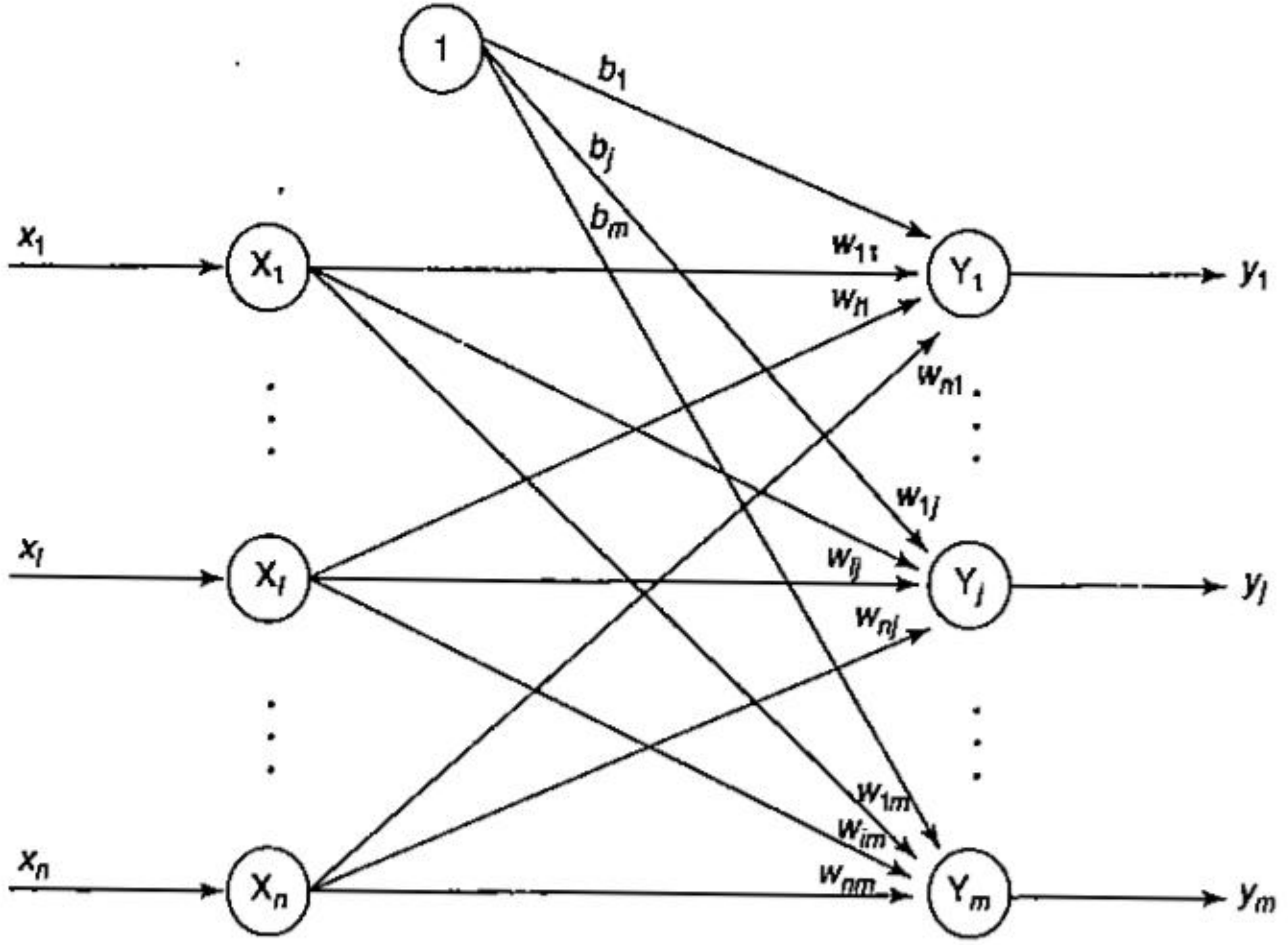
- 6. Compare the value of the calculated output and the desired (target, t) output. If y≠t, then update based on the following rule:

$$w^{new}{}_i \leftarrow w^{old}{}_i + \eta \cdot t \cdot x_i$$

$$b^{new} \leftarrow b^{old} + \eta \cdot (t)$$

- 7. Train the network until there is no weight change. This is the stopping condition of the network.

Single Layer Perceptron – Multiple Output Neuron



Ex.4

- Implement AND function using perceptron network for bipolar inputs and targets.

Inputs			Target
x_1	x_2	b	y
1	1	1	1
1	−1	1	−1
−1	1	1	−1
−1	−1	1	−1

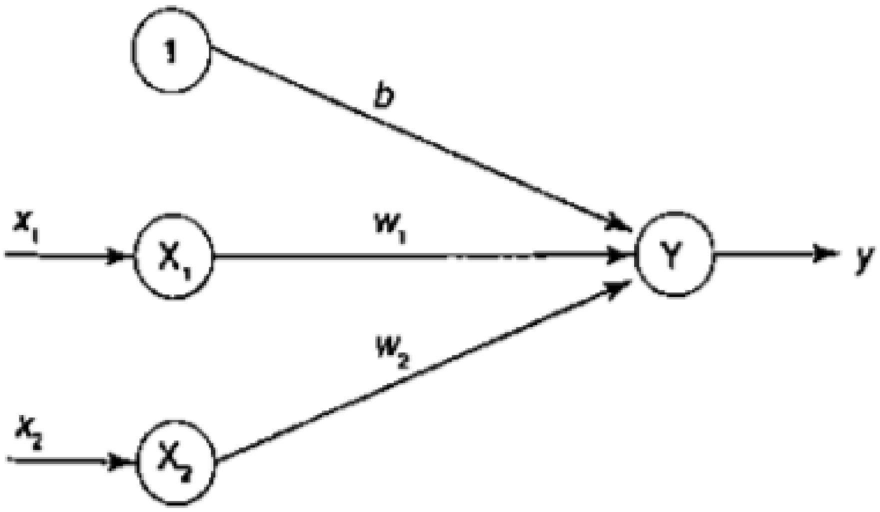


Figure 1 Perceptron network for AND function.

Ex.4

- Implement AND function using perceptron network for bipolar inputs and targets.

Inputs			Target
x_1	x_2	b	y
1	1	1	1
1	−1	1	−1
−1	1	1	−1
−1	−1	1	−1

Input			Target	Net input	Calculated output	Weight changes			Weights		
									w_1	w_2	b
x_1	x_2	1	(t)	(y_{in})	(y)	Δw_1	Δw_2	Δb	(0	0	0)
EPOCH-1											
1	1	1	1	0	0	1	1	1	1	1	1
1	−1	1	−1	1	1	−1	1	−1	0	2	0
−1	1	1	−1	2	1	+1	−1	−1	1	1	−1
−1	−1	1	−1	−3	−1	0	0	0	1	1	−1
EPOCH-2											
1	1	1	1	1	1	0	0	0	1	1	−1
1	−1	1	−1	−1	−1	0	0	0	1	1	−1
−1	1	1	−1	−1	−1	0	0	0	1	1	−1
−1	−1	1	−1	−3	−1	0	0	0	1	1	−1

Adaptive Linear Neuron - Adaline