MODULE 5

CHAPTER 5

# Classification Models

**Syllabus**

Introduction, Fundamental concept, Evolution of Neural Networks, Biological Neuron, Artificial Neural Networks, NN architecture, McCulloch-Pitts Model. Designing a simple network, Non-separable patterns, Perceptron model with Bias. Activation functions, Binary, Bipolar, continuous, Ramp. Limitations of Perceptron. Perceptron Learning Rule. Delta Learning Rule (LMS-Widrow Hoff), Multi-layer perceptron network. Adjusting weights of hidden layers. Error back propagation algorithm. Logistic regression.


5.1 Introduction ..... 5-3

5.2 Artificial Neural Network (ANN) ..... 5-3

5.2.1 Concept ..... 5-3

5.2.2 Types of Neural Networks Activation Functions (Common) ..... 5-3

5.2.3 Advantages of Neural Networks ..... 5-3

5.3 Biological Neural Network (BNN) ..... 5-4

5.3.1 Operations of Dendrite, Soma and Axon ..... 5-5

UQ. Explain the operations of Dendrite, soma and axon in the biological neuron. (MU - Q. 1(e), May 2012, 5 Marks) ..... 5-5

5.3.2 Difference between Computer and Human Brain. 5-6

5.3.3 Comparison between Biological Neuron (BNN) and Artificial Neuron (Brain v/s Computer) ..... 5-6

UQ. Differentiate biological neural network and artificial neural network. (MU - Q. 1(f), Dec. 2016, 4 Marks) ..... 5-6

5.4 Mc culloch-pitts neuron Architecture ..... 5-7

UQ. Draw and explain McCulloch Pitts neuron architecture. (MU - Q. 1(b), May 2017, 5 Marks, Q. 3(B), Dec. 2016, 10 Marks) ..... 5-7

5.4.1 Linear Separability ..... 5-7

5.4.2 Single-Layer Network ..... 5-8

5.5 Solved examples ..... 5-8

5.6 Characteristics of Mc-Culloch-Pitts ANN ..... 5-15

5.7 Classes of neural networks ..... 5-16

5.7.1 Single Layer Feed Forward Network ..... 5-16

5.7.2 Multilayer Feedforward Network ..... 5-16

5.7.3 Single Node with its Own Feedback ..... 5-17

5.7.4 Single Layer Recurrent Network ..... 5-17

5.7.5 Multilayer Recurrent Network ..... 5-17

5.7.6 Difference between Multi Layer Feed Forward Neural Network and Signal Layer Feed Forward Neural Network ..... 5-18

5.8 Applications of Neural Networks ..... 5-18

UQ. What are the performance measures to see whether training of neural network is successful ? Explain. (MU - Q. 4(a), May 2016, 10 Marks) ..... 5-18

5.9 Important Terminologies of ANN ..... 5-19

5.9.1 Weights ..... 5-19

5.9.2 Bias ..... 5-19

5.9.3 Threshold ..... 5-19

5.10 Limitations of neural networks ..... 5-20

5.11 Introduction to Perceptron ..... 5-20

5.11.1 Perceptron Networks ..... 5-20

MODULE 5

CHAPTER 5

# Classification Models

**Syllabus**

Introduction, Fundamental concept, Evolution of Neural Networks, Biological Neuron, Artificial Neural Networks, NN architecture, McCulloch-Pitts Model. Designing a simple network, Non-separable patterns, Perceptron model with Bias. Activation functions, Binary, Bipolar, continuous, Ramp. Limitations of Perceptron. Perceptron Learning Rule. Delta Learning Rule (LMS-Widrow Hoff), Multi-layer perceptron network. Adjusting weights of hidden layers. Error back propagation algorithm. Logistic regression.


5.1 Introduction ..... 5-3

5.2 Artificial Neural Network (ANN) ..... 5-3

5.2.1 Concept ..... 5-3

5.2.2 Types of Neural Networks Activation Functions (Common) ..... 5-3

5.2.3 Advantages of Neural Networks ..... 5-3

5.3 Biological Neural Network (BNN) ..... 5-4

5.3.1 Operations of Dendrite, Soma and Axon ..... 5-5

UQ. Explain the operations of Dendrite, soma and axon in the biological neuron. (MU - Q. 1(e), May 2012, 5 Marks) ..... 5-5

5.3.2 Difference between Computer and Human Brain. 5-6

5.3.3 Comparison between Biological Neuron (BNN) and Artificial Neuron (Brain v/s Computer) ..... 5-6

UQ. Differentiate biological neural network and artificial neural network. (MU - Q. 1(f), Dec. 2016, 4 Marks) ..... 5-6

5.4 Mc culloch-pitts neuron Architecture ..... 5-7

UQ. Draw and explain McCulloch Pitts neuron architecture. (MU - Q. 1(b), May 2017, 5 Marks, Q. 3(B), Dec. 2016, 10 Marks) ..... 5-7

5.4.1 Linear Separability ..... 5-7

5.4.2 Single-Layer Network ..... 5-8

5.5 Solved examples ..... 5-8

5.6 Characteristics of Mc-Culloch-Pitts ANN ..... 5-15

5.7 Classes of neural networks ..... 5-16

5.7.1 Single Layer Feed Forward Network ..... 5-16

5.7.2 Multilayer Feedforward Network ..... 5-16

5.7.3 Single Node with its Own Feedback ..... 5-17

5.7.4 Single Layer Recurrent Network ..... 5-17

5.7.5 Multilayer Recurrent Network ..... 5-17

5.7.6 Difference between Multi Layer Feed Forward Neural Network and Signal Layer Feed Forward Neural Network ..... 5-18

5.8 Applications of Neural Networks ..... 5-18

UQ. What are the performance measures to see whether training of neural network is successful ? Explain. (MU - Q. 4(a), May 2016, 10 Marks) ..... 5-18

5.9 Important Terminologies of ANN ..... 5-19

5.9.1 Weights ..... 5-19

5.9.2 Bias ..... 5-19

5.9.3 Threshold ..... 5-19

5.10 Limitations of neural networks ..... 5-20

5.11 Introduction to Perceptron ..... 5-20

5.11.1 Perceptron Networks ..... 5-20

5.12 Key-Points ....5-20
5.12.1 The Perceptron Rule Convergence Theorem ....5-20
5.13 Architecture ....5-21
5.13.1 Flowchart for Training Process ....5-21
5.14 Perceptron Training Algorithm for single output Classes ....5-21
5.15 Multilayer feedforward network ....5-22
5.15.1 Perceptron Training Algorithm for Multiple output Classes ....5-22
5.16 Activation Function ....5-23
UQ. Explain common activation functions used in neural network. (MU - Q. 1(b), May 2011, 5 Marks) ....5-23
5.16.1 Activation Functions ....5-23
UQ. What are the important properties of activation function used in neural networks ? (MU - Q. 1(c), May 2016, 5 Marks) ....5-23
UQ. List the different activation functions used in neural network. (MU - Q. 1(d), May 2017, 5 Marks) ....5-23
UQ. With mathematical list four different activation functions used in neurons. (MU - Q. 1(e), May 2019, 5 Marks) ....5-24
5.16.2 Illustrative Examples for Logistic Function ....5-25
5.17 Advantages, Disadvantages and Limitations of Perceptron ....5-27
5.17.1 Advantages of Perceptron (Multi-layer) ....5-27
5.17.2 Disadvantages of Perceptron (Multi-layer) ....5-27
5.17.3 Limitations of Perceptron ....5-27
5.18 Program for Perceptron Training Algorithm ....5-27
5.18.1 Perceptron Network Testing Algorithm ....5-28
5.18.2 Recurrent Networks ....5-29
5.18.3 Characteristics of Neural Network ....5-29
5.18.4 Learning Method ....5-29
5.18.5 Rosenblatt's Perceptron ....5-30
5.18.6 Learning Algorithm for Training The Perceptron 5-30
5.19 ADALine Network ....5-31
5.19.1 Delta Rule ....5-31
5.19.2 Adaline Model ....5-32
5.19.3 Architecture ....5-32
5.19.4 Algorithm for Training Process ....5-32
5.19.5 Testing of Algorithm ....5-32
5.20 Multiple Adaptive Linear Neurons : (Madaline) ....5-32
5.20.1 Architecture ....5-33
5.20.2 Flowchart of Training Process ....5-33
5.20.3 Training Algorithm ....5-33
5.20.4 Algorithm Preparation ....5-33
5.20.5 For Convenience, We Mention Here the Flowchart for Training of Madaline ....5-34
5.21 Back-Propagation Network (BPN) ....5-35
5.21.1 Architecture ....5-36
5.21.2 Algorithm (Training) ....5-37
5.21.3 Flow-chart for Back-Propagation Network Training ....5-38
5.22 Learning factors of back propagation network ....5-39
5.23 Calculation of Error ....5-39
5.23.1 Method of Steepest Descent ....5-39
5.23.2 Some Applications ....5-40
5.24 Solved examples ....5-40
5.25 Medaline-rule algorithm (MRA) ....5-49
5.26 Logistic Regression ....5-52
UQ. Write short note on : Logistic Regression. (Ref. MU (Comp.) - May 17, 10 Marks) ....5-52
UQ. Explain how regression problem can be solved using Steepest descent method. Write down the steps. (Ref. MU (Comp.) - May 19, 5 Marks) ....5-52
• **Chapter End** ....5-54

age no. (5-2)
.....5-31
.....5-31
.....5-32
.....5-32
.....5-32
.....5-32
line)..5-32
.....5-33
.....5-33
.....5-33
.....5-33
Flowchart
.....5-34
.....5-35
.....5-36
.....5-37
....5-38
...5-39
...5-39
..5-39
.5-40
.5-40
5-49
5-52
5-52
ved
the
-52
-54



## 5.1 INTRODUCTION

The human brain is a highly complex and amazing processor. The most basic element of the human brain is a specific type of cell, known as Neuron, which does not regenerate. The power of human brain (mind) comes from the number of Neurons and their multiple interconnections. A brain has the ability to **Learn** to build up its own rules through what we refer to as Experience.

A **Neural-Network** is a machine that is designed to model the way in which the brain performs a particular task. Clearly, our interest is confined to an important class of **neural networks** that perform useful computation through process of **learning.** To achieve good performance, Neural networks employ a massive interconnection of simple computing cells, referred to as "Neurons" or "Processing units".

## 5.2 ARTIFICIAL NEURAL NETWORK (ANN)

- **Definition :** Artificial Neural Network thus, we can define an 'artificial neural network' (ANN) is a massively distributed processor made up of simple processing units which store experiential knowledge and make it available for use.

**The important features of 'Artificial Neural-Network (ANN)' are :**

(1) Knowledge is acquired from its environment through a learning process.

(2) Interneuron-connections store the acquired knowledge.

(3) 'Artificial Neural Networks' learn by examples. In biological processes, learning involves adjustments to the connections that exist between neurons, ANNS undergo a similar changes and do the job on computers accurately all the time.

### 5.2.1 Concept

(1) Neural networks are those information processing systems, which are constructed and implemented to **model the human brain.**

(2) The concept that is of primary importance for a Neural Network is the ability of the network to **learn** from its environment, and to **improve** its performance through learning.

### 5.2.2 Types of Neural Networks Activation Functions (Common)

1. Binary step function (Threshold Function)
2. Linear Activation Function
3. Sigmoid/Logistic Activation Function
4. The derivative of the sigmoid Activation Function
5. Tanh Function (Hyperbolic Tangent)
6. Gradient of the Tanh Activation function
7. ReLu Activation Function

### 5.2.3 Advantages of Neural Networks

The use of neural networks offers the following useful capabilities :

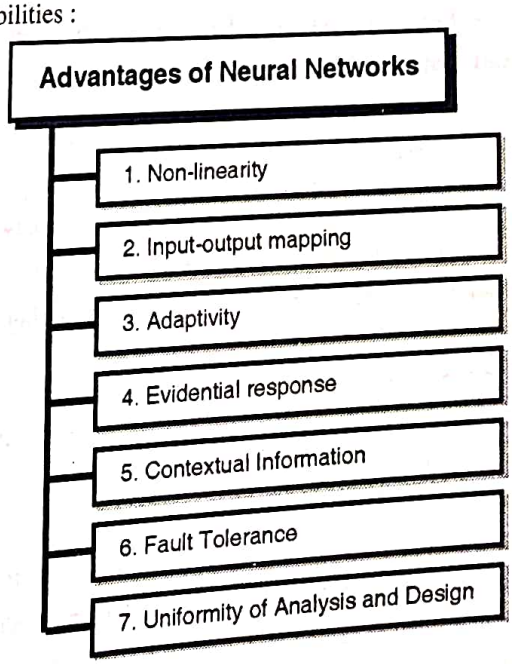



**Fig. 5.2.1**

(1) **Non-linearity :** An artificial neuron can be **linear or non-linear.** A neural network consists of an interconnection of non-linear neurons, and hence it is itself non-linear. The non-linearity is distributed throughout the network.

(2) **Input-output mapping :** Each example consists of a unique **input signal** and corresponding desired response. Thus the network learns from the examples by constructing an **input-output mapping** for the problem at hand.

(3) **Adaptivity :** A neural network is trained to operate in a specific environment can be easily retrained to deal with minor changes in the operating environmental conditions.

(4) **Evidential response :** 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 :** Knowledge is represented by the very structure of a neural network. Every neuron in the network is potentially affected by the global activity of all other neurons in the network.

(6) **Fault Tolerance :** A neural network has the potential to be **fault tolerant.** In order to have neural network **fault tolerant,** it is necessary to take corrective measures in designing the algorithm used to develop the network.

(7) **Uniformity of Analysis and Design :** Neural networks are basically information processors. Hence the same notation is used in all domains involving the application of neural networks.

## 5.3 BIOLOGICAL NEURAL NETWORK (BNN)

- An artificial neural network (ANN) is an efficient information processing system which resembles in characteristics with a biological neutral network (BNN).
- ANN's behaviour is similar to that of a human brain, so far as ability to learn, recall and generalise training patterns. Thus, the ANN processing elements are called **neurons** or **artificial neurons.**

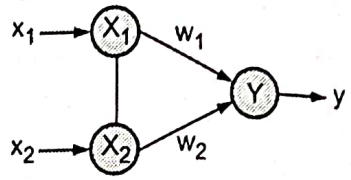



**Fig. 5.3.1 : Architecture of a simple artificial neutron network**

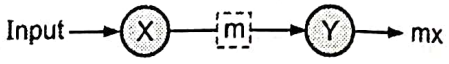



**Fig. 5.3.2 : Neural net of pure linear equation**

> **Note :**
> (i) Each neurons has its own internal state, and is called the **activation** or **activity level** of neuron.
> (ii) The state of neuron is a function of inputs it receives, and this activation signal is transmitted to other neurons.
> (iii) A neuron can send only one signal at a time and is transmitted to other several neurons.

For the above linear network architecture, the net input is given by, $y_{in} = x_1 w_1 + x_2 w_2$

where $x_1$ and $x_2$ are the activations of the input neurons $X_1$ and $X_2$;

- The output y of the output neuron Y is the function of the **net input.**

$$y = f(y_{in})$$

- The function f is called **activation function.** When the input is linearly varied, output is also linearly varied, as shown in **Fig. 5.3.3.**

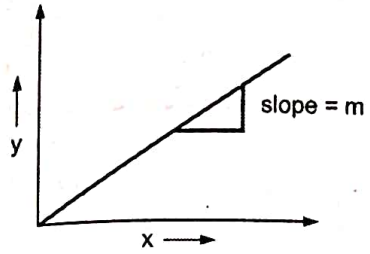



**Fig. 5.3.3 : Graph for y = mx**

### 5.3.1 Operations of Dendrite, Soma and Axon

**UQ.** Explain the operations of Dendrite, soma and axon in the biological neuron.

**(MU - Q. 1(e), May 2012, 5 Marks)**

Human brain consists of a huge number of neurons. Approximately 10". (Interestingly, there are approximately 10" stars in our Galaxy. But don't think that this is an astronomical figure, because our national deficit is still larger than that).

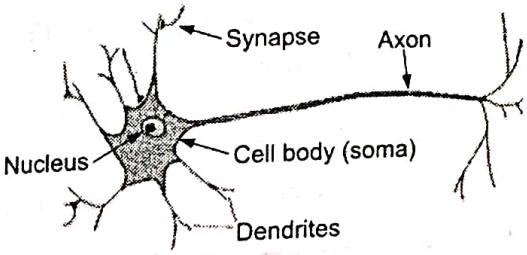



**Fig. 5.3.4 : Diagram of a Biological Neuron**

The biological neuron shown in the Fig. 5.3.4 consists of three main parts :

1. **Soma or cell body :** Here the cell nucleus is located.
2. **Dendrites :** Where the nerve is connected to the cell body.
3. **Axon :** It carries the impulses of the neuron.
   (i) **Dendrites** are networks made of nerve-fibre and are connected to the cell body.
   (ii) An **Axon** carries signals from the neuron. It is a single, long connection extending from the cell body.
   (iii) The **end** of the **axon** splits into **fine strands.** Each **strand** terminates into a small-bulb-like organ called **synapse.** It is through **synapse** that the neuron introduces its signals to other nearby neurons. The receiving ends of these synapses on the nearby neurons can be found on the dendrites and also on the cell-body. Approximately there are $10^4$ **synapses** per neuron in the human brain.

- Electric impulses are passed between synapse and the dendrites. This signal transmission involves a chemical process in which specific transmitter substances are releases from the sending side of the junction. This causes and increase or decrease in the electric potential inside the body of the receiving cell.
- If the electric potential reaches or crosses a threshold then the receiving cell fires and a pulse or action potential of fixed strength and duration is sent through the **axon** to the **synaptic** junctions of the other cells.
- The synapses are inhibitory it they hinder the firing of the receiving cell, or they are excitatory if they pass impulses which cause the firing of the receiving cell.

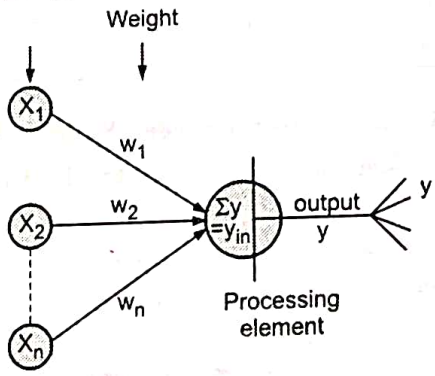



**Fig. 5.3.5 : Mathematical model of artificial neuron**

**Terminology**

| Sr. No. | Biological neuron | Artificial neuron |
|---|---|---|
| 1. | Cell | Neuron |
| 2. | Dendrites | Weights or interconnections |
| 3. | Soma | Net input |
| 4. | Axon | Output |

In this model, the net input is given by,

$$y_{in} = x_1 w_1 + x_2 w_2 + \ldots + x_n w_n = \sum_{i=1}^{n} x_i w_i$$

Where i is $i^{th}$ processing element.

### 5.3.2 Difference between Computer and Human Brain

| Sr. No. | Human Brain | Computer |
|---|---|---|
| 1. | Biological neurons or nerve cells | Silicon Transistors |
| 2. | **200 billion** neurons, 32 trillion inter-connections. | **1 billion bytes** RAM, trillion of bytes on disk. |
| 3. | Neuron **size 10-6m** | Single transistor **size 10-9 m** |
| 4. | Energy consumption **6-10 joules** per operation per sec. | Energy consumption **10-16 joules** per operation per sec. |
| 5. | **Learning** capability. | **Programming** capability. |

### 5.3.3 Comparison between Biological Neuron (BNN) and Artificial Neuron (Brain v/s Computer)

**UQ.** Differentiate biological neural network and artificial neural network.

**(MU - Q. 1(f), Dec. 2016, 4 Marks)**

| Sr. No. | Characteristics | Artificial Neural Network | Biological (Real) Neural Network |
|---|---|---|---|
| 1. | Speed | **Faster** in processing information. Response time is in nanoseconds. | **Slower** in processing information. The response time is in milliseconds. |
| 2. | Processing | **Serial** processing | Massively **parallel** processing. |
| 3. | Size and complexity | **Less** size and complexity. It does not perform complex pattern recognition tasks. | A **highly complex** and dense network of interconnected neurons containing neurons of the order of 1011 with 1015 of interconnections. |
| 4. | Storage | New data with old data can be replaced, i.e. information storage is **replaceable.** | New information or data is added by adjusting the interconnection strength **without destroying old information.** |
| 5. | Anism | There is a **control unit** for controlling computing activities. | **No specific control** mechanism external to the computing task. |
| 6. | Resemblance | A neural network acquires knowledge through learning | - |
| 7. | Resemblance | A neural network's knowledge is a store within inter neurons connection known as synaptic weights. | - |



Tech-Neo Publications...A SACHIN SHAH Venture

## 5.4 MC CULLOCH-PITTS NEURON ARCHITECTURE

**UQ.** Draw and explain McCulloch Pitts neuron architecture.

**(MU - Q. 1(b), May 2017, 5 Marks, Q. 3(B), Dec. 2016, 10 Marks)**

The Mc-Culloch-Pitts Neuron is the neural network. It is usually called a **m-p neuron.**

(1) The M – P neurons are **connected** by **directed weighted paths.**

(2) Activation of a M-P neuron is binary, that is, at any time step, the neuron may fire or may not fire.

(3) The weight associated with the links are of **two types** :

(a) They may be **excitatory** (weight is positive) or

(b) They may be **inhibitory** (weight is negative).

(4) The threshold plays a major role.

(5) For each neuron, threshold is fixed,

(6) If the net input to the neuron is greater than the threshold the neuron fires.

(7) Any non-zero inhibitory input will prevent the neuron from firing.

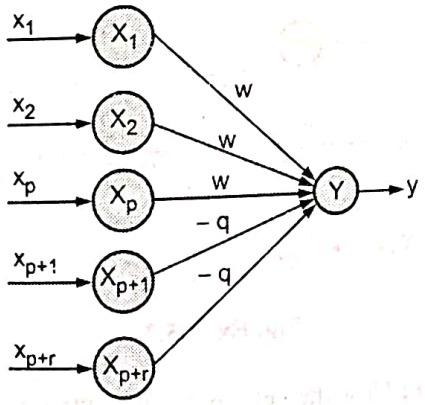



**Fig. 5.4.1 : McCulloch-pitts neuron Architecture**

Here the inputs refer Fig. 5.4.1 $x_1$ to $x_p$ (in the figure) possess excitatory weighted connections and input from $x_{p+1}$ to $x_{p+r}$ possess inhibitory weighted connections.

We define activation function as, $f(y_{in}) = \begin{cases} 1 \text{ if } y_a \geq \theta \\ 0 \text{ if } y_p < \theta \end{cases}$

; Where $\theta > pw - q$

The output will fire if it receives "k" or more excitatory inputs but no inhibitory inputs,

i.e., $kw \geq \theta \geq (k-1)\,w$.

### 5.4.1 Linear Separability

To separate positive and negative responses, a **decision line** is drawn. The decision line is also called as the decision making line or **'decision-support line'** or **'linear-separable line'.**

Generally the net input calculated to the output unit is given as :

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i \quad \text{...(i)}$$

The value of the function is 1 for a +ve net input and –1 for a negative net input. Also there exists a boundary between the region. Where $y_{in} > 0$ and $y_{in} < 0$.

This region is called is called as **decision boundary region,** and can be determined by the relation.

$$b + \sum_{i=1}^{n} x_i w_i = 0 \quad \text{....(ii)}$$

**Remarks**

(1) Depending upon the number of input units in the network, the above equation (ii) may represent a line, a plane or a hyper plane.

(2) If there exist weights for which the input vectors having positive response, +1, lie on one side of the decision boundary line and all other vectors having negative response, –1, lie on the other side of the decision boundary, then the problem is **"linearly separable".**

### 5.4.2 Single-Layer Network

Let us consider a Single-Layer Network

Now, the separating line is given by,

$$b + x_1 w_1 + x_2 w_2 = 0$$

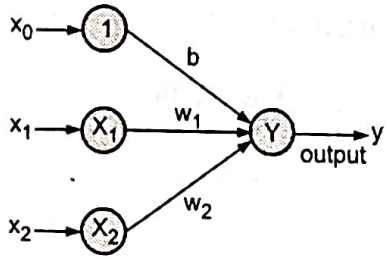



**Fig. 5.4.2 : A single-layer neural-net**

If $w_2 \neq 0$, then

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}$$

Thus the requirement for the positive response is : $b + x_1 w_1 + x_2 w_2 > 0$

**Remarks**

If the threshold value is being used, then the condition for the positive response from the output unit is

Net input received > θ (threshold value)

$$\therefore\ y_{in} > \theta\ ; \qquad \therefore\ x_1 w_1 + x_2 w_2 > \theta$$

Then the equation of the separating line will be

$$x_1 w_1 + x_2 w_2 = \theta\ ;$$

$$\therefore\ x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2};\ (\text{with } w_2 \neq 0)$$

## 5.5 SOLVED EXAMPLES

**Ex. 5.5.1 :** For the network shown in the Fig. Ex. 5.5.1 calculate the net input to the output neuron.

**Soln. :**

The given neural net consists of three input neurons and one output neuron.

The inputs are : $x_1 = 0.3$, $x_2 = 0.5$, $x_3 = 0.6$

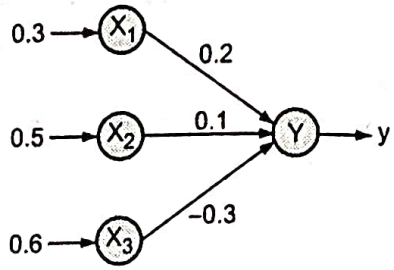



**Fig. Ex. 5.5.1**

The weights are : $w_1 = 0.2$, $w_2 = 0.1$, $w_3 = -0.3$

The net-input is given by,

$$y_{in} = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$= (0.3)(0.2) + (0.5)(0.1) + (0.6)(-0.3)$$

$$\therefore\ y_{in} = 0.06 + 0.05 - 0.18 = -0.07$$

**Ex. 5.5.2 :** Obtain the output of the neuron Y for the network shown in the Fig. Ex. 5.5.2, using activation functions as : (i) binary sigmoidal and (ii) bipolar sigmoidal.

**Soln. :**

**Step (I) :** (i) The given network has three input neurons with bias and one output neuron.

These form a single – layer network.

(ii) The inputs are : $x_1 = 0.8$ ; $x_2 = 0.6$ ; $x_3 = 0.4$

and weights are $w_1 = 0.1$ ; $w_2 = 0.3$ ; $w_3 = -0.2$

and bias is b = 0.35 (its input is always 1)

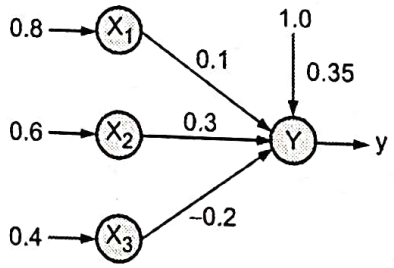



**Fig. Ex. 5.5.2**

**Step (II) :** Now the net input to the output neuron is

$$y_{in} = b + \sum_{i=1}^{3} x_i w_i;$$

(∵ only 3 input neurons are given)

$$= b + x_1 w_1 + x_2 w_2 + x_3 w_3$$

$$= 0.35 + (0.8)(0.1) + (0.6)(0.3) + (0.4)(-0.2)$$

$$= 0.35 + 0.08 + 0.18 - 0.08 = 0.53$$

**Step (III) :** (i) Binary sigmoidal activation function given by,

$$y = f(y_{in}) = \left(\frac{1}{1+e^{-y_{in}}}\right) = \frac{1}{1+e^{-0.53}} = 0.625$$

(ii) bipolar sigmoidal activation functions is given by,

$$y = f(y_{in}) = \left(\frac{2}{1+e^{-y_{in}}}\right) - 1$$

$$= \left(\frac{2}{1+e^{-0.53}}\right) - 1 = 0.259$$

**UEx. 5.5.3 (MU - Q. 1(d), June 14, 5 Marks)**

Calculate the output of the neuron Y for the net given below. Use binary and bipolar sigmoidal activation functions :

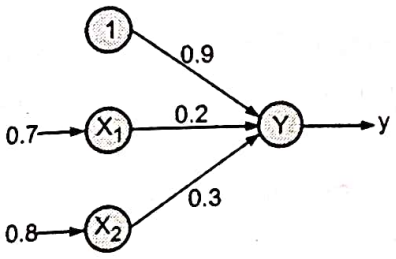



**Fig. Ex. 5.5.3**

**Soln. :**

**Step (I) :**

(i) The given network has two input neurons with bias and one output neuron.

These form a single-layer network.

(ii) The inputs are : $x_1 = 0.7$ ; $x_2 = 0.8$ and weights are : $w_1 = 0.2$ ; $w_2 = 0.3$ and bias = b = 0.9 (its input is always 1)

**Step (II) :** Now the net input to the output neuron is

$$y_{in} = b + \sum_{i=1}^{2} x_i w_i ;$$

($\because$ only 2 input neurons are mentioned)

$$= b + (x_1 w_1 + x_2 w_2)$$

$$= 0.9 + [(0.7)(0.2) + (0.8)(0.3)]$$

$$= 0.9 + 0.14 + 0.24 = 1.28$$

**Step (III) :** (i) Binary sigmoidal function is given by y

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

$$= \frac{1}{1+e^{-1.28}} = \frac{1}{1+0.278} = 0.7824$$

(ii) Bipolar sigmoidal activation function is,

$$y = f(y_{in}) = \left[\frac{2}{1+e^{-y_{in}}}\right] - 1$$

$$= \left[\frac{2}{1+e^{-1.28}}\right] - 1 = 0.5649$$

**Ex. 5.5.4 :** Implement AND function using Mc-Culloch-Pitts neuron (take binary data).

**Soln. :**

**Step (I) :** First we construct truth-table for AND function :

Here we assume the weights to be $w_1 = 1$ and $w_2 = 1$.

With these assumed weights we calculate the net input for four inputs as follows :

**Truth Table**

| $x_1$ | $x_2$ | y |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

For inputs :

(1, 1), $y_{in} = x_1 w_1 + x_2 w_2 = 1 \times 1 + 1 \times 1 = 2$

(1, 0), $y_{in} = x_1 w_1 + x_2 w_2 = 1 \times 1 + 1 \times 0 = 1$

(0, 1), $y_{in} = x_1 w_1 + x_2 w_2 = 0 \times 1 + 1 \times 1 = 1$

(0, 0), $y_{in} = x_1 w_1 + x_2 w_2 = 0 \times 1 + 0 \times 1 = 0$

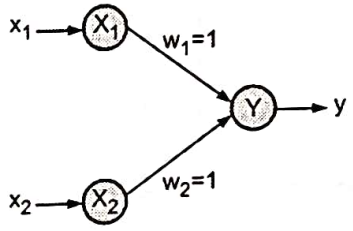



**Fig. Ex. 5.5.4 : Network architecture**

- **Step (II) :** For an AND function, the output is high if both the inputs are high, Hence we calculate the net input as 2.

Based on this net input, the thresh is set. If the threshold is greater than or equal to 2, then the neuron fires, otherwise it does not fire.

$\therefore$ Threshold value is set equal to 2, ($\theta = 2$). Also, we can use the formula : $\theta \geq nw - p$

Here, $n = 2$, $w = 1$ and $p = 0$

($\because$ no inhibitory weights).

$\therefore \theta \geq 2(1) - 0 = 2$

Thus, the output of neuron Y can be written as :

$$y = f(y_{in}) = \begin{cases} 1 \text{ if } y_{in} \geq 2 \\ 0 \text{ if } y_{in} < 2 \end{cases}$$

where '2' represents the threshold value.

**Ex. 5.5.5 :** Implement ANDNOT function using MC-Culloch-Pitts-neuron (use binary data representation).

☑ **Soln. :**

- **Step (I) :** We prepare the truth-table for ANDNOT function.

In this case, the response is true if the first input is true and the second input is false. For all other input variations, the response is false.

**Truth Table**

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

The given function gives an output when $x_1 = 1$ and $x_2 = 0$.

We represent the neural net :

- **Step (II) : Case(i) :** Let $w_1$ and $w_2$ be excitatory, i.e.,

$$w_1 = w_2 = 1$$

Now, we calculate the net input using four inputs.

$$y_{in} = x_1 w_1 + x_2 w_2$$

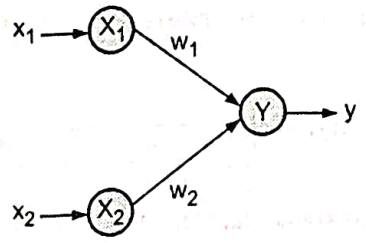



**Fig. Ex. 5.5.5**

For inputs

(1, 1) : $y_{in} = 1 \times 1 + 1 \times 1 = 2$ ;

(1, 0) : $y_{in} = 1 \times 1 + 0 \times 1 = 1$

(0, 1) : $y_{in} = 0 \times 1 + 1 \times 1 = 1$ ;

(0, 0) : $y_{in} = 0 \times 1 + 0 \times 1 = 0$

From the calculated net inputs, it is not possible to fire the neuron for input (1, 0) only. Hence these weights are not suitable.

So, we assume : one weight as excitatory and the other as inhibitory, i.e.,

$w_1 = 1$, $w_2 = -1$.

Now, we calculate the net input : For the input :

(1, 1) : $y_{in} = 1 \times 1 + 1 \times (-1) = 0$ ;

(1, 0) : $y_{in} = 1 \times 1 + 0 \times (-1) = 1$

(0, 1) : $y_{in} = 0 \times 1 + 1 \times (-1) = -1$ ;

(0, 0) : $y_{in} = 0 \times 1 + 0 \times (-1) = 0$

Now, it is possible to fire the neuron for input (1, 0) only by firing a threshold of 1 ;

i.e. $\theta \geq 1$. $\therefore w_1 = 1$, $w_2 = -1$ ; $\theta \geq 1$

Thus the output of neuron Y can be written as,

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

**Ex. 5.5.6 :** Using the linear separability concept, obtain the response for OR function (taking bipolar inputs and bipolar targets).

**Soln. :**

**Step (I) :** We prepare truth-table for OR function with bipolar inputs and targets.

Noting the points, we plot the figure :

Using co-ordinates (1, – 1) and (– 1, 1);

$$\text{Slope of line} = \frac{1-(-1)}{(-1)-(1)} = \frac{2}{-2} = -1$$

**Truth-table**

| $x_1$ | $x_2$ | y |
|---|---|---|
| 1 | 1 | 1 |
| 1 | – 1 | 1 |
| – 1 | 1 | 1 |
| – 1 | – 1 | – 1 |

If the output is 1.

It is denoted as '+ve', otherwise '–ve'.

Using this value the equations for the line is,

$$y = mx + c = (-1)x - 1 = -x - 1$$

**Note :** Here the quadrants are $x_1$ and $x_2$ and not x and y.

$\therefore$ The above equation becomes $x_2 = -x_1 - 1$ ...(i)

This can be written as $x_2 = -\frac{w_1}{w_2}x_1 - \frac{b}{w_2}$ ...(ii)

Comparing (i) and (ii); $\frac{w_1}{w_2} = \frac{1}{1}$; $\frac{b}{w_2} = \frac{1}{1}$

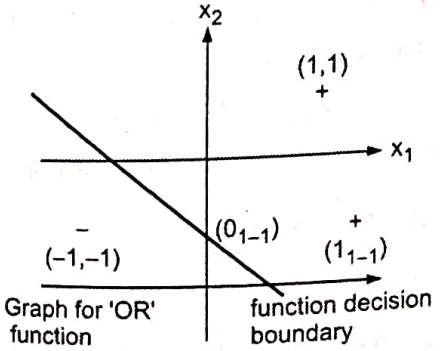



**Fig. Ex. 5.5.6 : Graph for 'OR'**

$\therefore$ $w_1 = 1$ and $w_2 = 1$, and $b = 1$

**Step (II) :** Calculating the not input and output of OR function on the basis of the above weights, we have the **entries**

Thus the output of the neuron Y can be written as

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

**Truth-table**

| $x_1$ | $x_2$ | b | $y_{in} = b + x_1 w_1 + x_2 w_2$ | y |
|---|---|---|---|---|
| 1 | 1 | 1 | 3 | 1 |
| 1 | – 1 | 1 | 1 | 1 |
| – 1 | 1 | 1 | 1 | 1 |
| – 1 | – 1 | 1 | – 1 | – 1 |

Where the threshold is taken as "1" ($\theta = 1$) based on calculated net input.

Hence using the linear separability concept, He response is obtained for "OR" function.

**UEx. 5.5.7 (MU - Q. 3(B), Dec. 2016, 10 Marks)**

Implement XOR function using Mc-Culloch-Pitts neuron (consider binary data).

**Soln. :**

**Step (I) :** Prepare truth-table for XOR function

In this case, only for add numbers of 1's the output is 'ON'. And for the rest it is 'OFF'.

Such a function cannot be represented by simple and single logic functions; so we represent it as follows :

$$y = x_1 \bar{x}_2 + \bar{x}_1 x_2 \quad \text{or} \quad y = z_1 + z_2$$

where $z_1 = x_1 \bar{x}_2$

**(function 1) Truth-Table**

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$z_2 = \bar{x}_1 x_2$ (function 2)

and $y = z_1 \text{ (OR) } z_2$ (function 3)

Here a single-layer is not sufficient to represent the function. An intermediate layer is necessary :

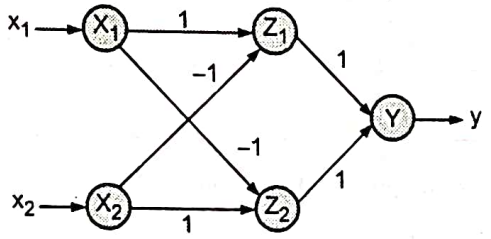



**Fig. Ex. 5.5.7(a) : Neural net for XOR function the weights shown are obtained after analysis**

- **Step (II) :** Prepare truth-table for first function

$$z = x_1 \bar{x}_2 :$$

The net representation is given as :

- **Case (1) :** Assume both weights as excitatory, i.e.

$$w_{11} = w_{21} = 1$$

**Truth-Table**

| $x_1$ | $x_2$ | $z_1$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

We calculate the net inputs :

For (0, 0) ; $z_1(in) = 0 \times 1 + 0 \times 1 = 0 + 0 = 0$

For (0, 1) ; $z_1(in) = 0 \times 1 + 1 \times 1 = 0 + 1 = 1$

For (1, 0) ; $z_1(in) = 1 \times 1 + 0 \times 1 = 1 + 0 = 1$

For (1, 1) ; $z_1(in) = 1 \times 1 + 1 \times 1 = 1 + 1 = 2$

Hence, using these weights, it is not possible to obtain function $z_1$ ;

[$\because$ for $\theta > 2$, there is not excitatory].

- **Case (2) :** We assume one weight as excitatory and the other as inhibitory, i.e.,

$w_{11} = 1; w_{21} = -1.$

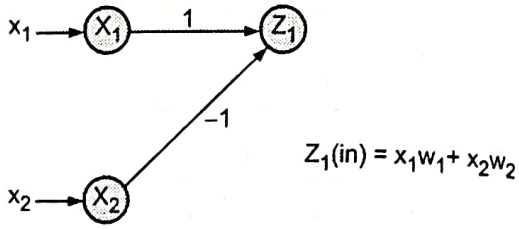



**Fig. Ex. 5.5.7(b) : Neural for $Z_1$**

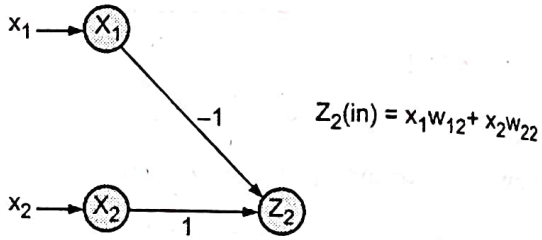



**Fig. Ex. 5.5.7(c) : Neural for $Z_2$**

We calculate the net inputs for inputs :

(0, 0), $z_1$ (in) $= 0 \times 1 + 0 \times (-1) = 0 + 0 = 0$ ,

(0, 1), $z_1$ (in) $= 0 \times 1 + 1 \times (-1) = -1$

(1, 0), $z_1$ (in) $= 1 \times 1 + 0 \times (-1) = 1$ ;

(1, 1), $z_1$ (in) $= 1 \times 1 + 1 \times (-1) = 0$

On the basis of this calculated net input, it is possible to get the required output. Hence.

$$w_{11} = 1, \quad w_{21} = -1$$

and $\theta \geq 1$ for $z_1$ neuron.

- **Step (III) :** Second function : $z_2 = \bar{x}_1 x_2$

We prepare the truth-table for $z_2$ :

- **Case (1) :** Assume both weights as excitatory,

i.e. $w_{12} = w_{22} = 1.$

**Truth-Table for $z_2$**

| $x_1$ | $x_2$ | $z_2$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

With this, we calculate the inputs.

For the inputs :

(0, 0), $z_2$ (in) = $0 \times 1 + 0 \times 1 = 0$ ;

(0, 1), $z_2$ (in) = $0 \times 1 + 1 \times 1 = 1$

(1, 0), $z_2$ (in) = $1 \times 1 + 0 \times 1 = 1$ ;

(1, 1), $z_2$ (in) = $1 \times 1 + 1 \times 1 = 2$

Again, it is not possible to obtain function $Z_2$

Using these weights (as above case).

- **Case (2) :** (As in the previous step) we assume one weight as excitatory and the other as inhibitory, i.e., $w_{12} = -1$; $w_{22} = 1$.

Now calculate the net inputs. For the inputs :

(0, 0), $z_2$ (in) = $0 \times (-1) + 0 \times 1 = 0$ ; (0, 1), $z_2$ (in) = $0 \times (-1) + 1 \times 1 = 1$

(1, 0), $z_2$ (in) = $1 \times (-1) + 0 \times 1 = -1$ ;(1, 1), $z_2$ (in) = $1 \times (-1) + 1 \times (1) = 0$

Thus, based on this calculated net input, it is possible to get the required output, i.e.,

$w_{12} = -1$, $w_{22} = 2$. $\therefore$ $\theta \geq 1$ for $z_2$ neuron.

- **Case (3) :** Third function : ($y = z_1$ Or $z_2$)

The truth table for this function is an follows :

The net input is to be calculated as :

$$y_{in} = z_1 v_1 + z_2 v_2;$$

Where $V_1$ and $V_2$ are weights to be assigned.

| $x_1$ | $x_2$ | y | $Z_1$ | $z_2$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 |

- **Case (i) :** Assume both weights as excitatory, i.e.,

$$v_1 = v_2 = 1$$

We calculate the net input. For inputs :

(0, 0), $y_{in} = 0 \times 1 + 0 \times 1 = 0$

(0, 1), $y_{in} = 0 \times 1 + 1 \times 1 = 1$ ;

(1, 0), $y_{in} = 1 \times 1 + 0 \times 1 = 1$

(1, 1), $y_{in} = 0 \times 1 + 0 \times 1 = 0$

[it is $\because$ for $x_1 = 1, x_2 = 1$ ; $z_1 = 0$ and $z_2 = 0$]

[from the table].

Setting a threshold of $\theta \geq 1$,

$v_1 = v_2 = 1$; it implies that the net is recognised.

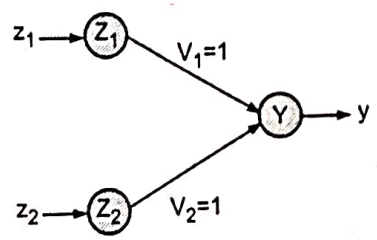



**Fig. Ex. 5.5.7(d) : Neural net for Y($Z_1$ or $Z_2$)**

The analysis made for XOR function using M-P neurons.

Thus for XOR – function, the weights to be assigned are

$w_{11} = w_{22} = 1$ (excitatory) ;

$w_{12} = w_{21} = -1$ (inhibitory)

$v_1 = v_2 = 1$ (excitatory)

➢ **Program 5.5.1 :** Implement XOR function using McCulloch Pitts neural net.

☞ **Source code**

```
% Implementation of XOR function using McCulloch-
Pitts neuron
%input and output patterns are considered as binary
data
clear;
clc;
% Getting weights and threshold value from the user
disp('Enter the weights');
w11=input('Weight w11=');
w12=input('Weight w12=');
w21=input('Weight w21=');
w22=input('Weight w22=');
v1=input('Weight v1=');
v2=input('Weight v2=');
disp('Enter threshold value');
theta=input('theta=');
x1=[0 0 1 1];
x2=[0 1 0 1];
z=[0 1 1 0];
    con=1;
        while con
            zin1 = x1*w11+x2*w21;
            zin2 = x1*w21+x2*w22;
            for i=1:4
                if zin1(i)>=theta
                    y1(i)=1;
                else y1(i)=0;
                end
                    if zin2(i)>=theta
                        y2(i)=1;
                    else y2(i)=0;
                    end
                end
            yin=y1*v1+y2*v2;
            for i=1:4
                if yin(i)>=theta;
                    y(i)=1;
                else
                    y(i)=0;
                end
            end
            disp('Output of net=');
            disp(y);
                if y==z
                    con=0;
                else
            disp('Net is not learning Enter another set of
weights and threshold value');
            w11=input('Weight w11=');
            w12=input('Weight w12=');
            w21=input('Weight w21=');
            w22=input('Weight w22=');
            v1=input('Weight v1=');
            v2=input('Weight v2=');
            theta=input('theta=');
            end
        end
        disp('McCulloch Pitts Net for XOR function');
        disp('Weights of neuron Z1');
        disp(w11);
        disp(w21);
        disp('Weights of neuron Z2');
        disp(w12);
        disp(w22);
        disp('Weights of neuron Y');
```

```
disp(v1);
disp(v2);
disp('Threshold value=');
disp(theta);
```

**Output**

```
McCulloch Pitts Net for XOR function
Weights of neuron Z1
  1
  -1
Weights of neuron Z2
  -1
  1
Weights of neuron Y
  1
  1
Threshold value=
  1
```

**Ex. 5.5.8 :** For the network shown in the Fig. Ex. 5.5.8, calculate the output Y for the network using activation function as :

(i) binary sigmoid and (ii) bipolar sigmoid

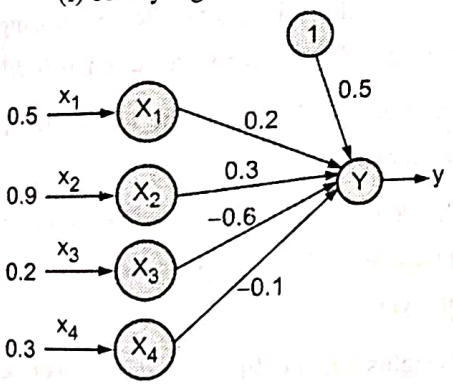



**Fig. Ex. 5.5.8**

**Soln. :**

**Step (I) :** First we find $y_{in}$.

Now, $y_{in} = b + x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4$ ...(i)

Where $x_1 = 0.5$, $x_2 = 0.9$,

$x_3 = 0.2$, $x_4 = 0.3$

$w_1 = 0.2$, $w_2 = 0.3$,

$w_3 = -0.6$, $w_4 = -0.1$

and $b = 0.5\ (x_0 = 1)$

From Equation (i),

$$y_{in} = 0.5 + (0.5)(0.2) + (0.9)(0.3) + (0.2)(-0.6) + (0.3)(-0.1)$$

$$= 0.5 + 0.1 + 0.27 - 0.12 - 0.03$$

$$= 0.87 - 0.15 = 0.72$$

$$\therefore y_{in} = 0.72$$

**Step (II) :** (i) to Find y for binary sigmoidal activation function

$$y = f(y_{in}) = \frac{1}{1 + e^{-y_{in}}}$$

$$= \frac{1}{1 + e^{-0.72}} = 0.6726$$

(ii) For bipolar sigmoidal activation function

$$y = f(y_{in}) = \left(\frac{2}{1 + e^{-y_{in}}}\right)$$

$$= \left(\frac{2}{1 + e^{-0.72}}\right) - 1 = 0.3452$$

## 5.6 CHARACTERISTICS OF MC-CULLOCH-PITTS ANN

1. The **activation is binary**. A neuron fires when its activation is 1, otherwise, its activation is zero.
2. Neurons are connected by **directed paths**.
3. If the weights are positive, the path is **excitatory**, otherwise the path is **inhibitory**.
4. All excitatory connections into a particular neuron have the **same weights**.
5. Each neuron has a fixed threshold. If the input to the neuron is **greater** than the threshold, **neuron fires**.
6. The neuron **will not fire**, if there is even one inhibitory input.

7. The weights are set so that **inhibition happens.**
8. The weights for Mc-Culloch-Pitts neurons along with the threshold are set by analysis of the problem at hand.

## 5.7 CLASSES OF NEURAL NETWORKS

According to their learning mechanism, Neural Network are classified into several classes. We identify three fundamentally different classes of networks. All these three classes employ the digraph structure for their representation. The arrangement of neurons to form layers to from layers and the connection pattern formed within and between layers is called **network architecture.** There exist five basic types of neuron connection architectures.

They are :

1. Single-layer feed-forward network;
2. Multilayer feed-forward network;
3. Single node with its own feedback;
4. Single-layer recurrent network;
5. Multilayer recurrent network

### 5.7.1 Single Layer Feed Forward Network

(1) This type of network comprises two layers, namely the **input layer** and the **output layer.**

(2) The **input layer neurons** receive' the input signals,

(3) The **output layer neurons** receive the output signals.

(4) The synaptic links carrying the weights connect every input neuron to the output neuron but not conversely.

(5) This network is called as feed forward in type or acyclic in nature.

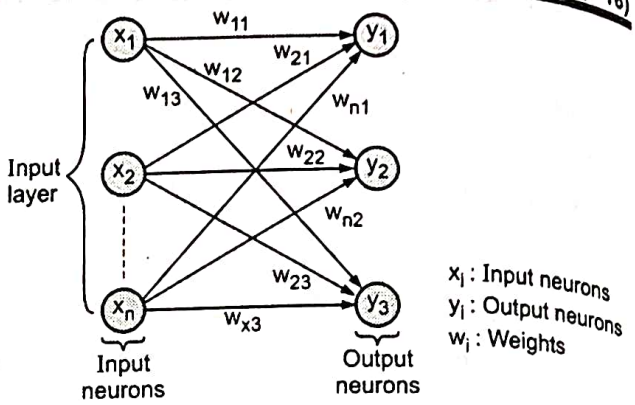



**Fig. 5.7.1 : Single layer feedforward network**

(6) The network is termed as single layer since the alone output layer, alone which performs computation.

(7) Even though there are two layers, it is called as single layer because of output layer.

(8) The input layer merely transmits the signals to the output layer.

We illustrate an example network as shown in Fig. 5.7.1.

### 5.7.2 Multilayer Feedforward Network

(1) As the name indicates, this network is made up of multiple layers.

(2) It has intermediary layers besides possessing input and output layers. These layers are called as **hidden layers.**

(3) The computational units of the hidden layer are called as hidden neurons or hidden units.

(4) The hidden layers adds in performing intermediary computations and then input layers are directed to the output layers.

(5) The weights on the links of input layer neurons and hidden layers are called as **Input-hidden layer weights.**

We illustrate this by the Fig. 5.7.2.

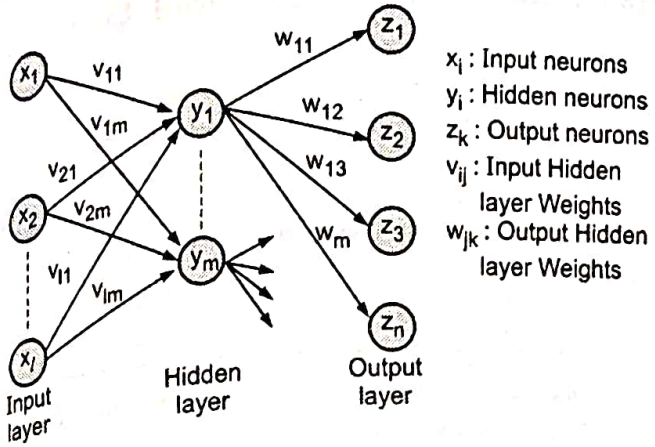



**Fig. 5.7.2 : A multilayer feed forward network**

## 5.7.3 Single Node with its Own Feedback

- When outputs can be directed back as inputs to the same layer or preceding layer nodes, then it results in feedback networks.
- If the feedback of the output of the processing element is directed back as input to the processing element in the same layer then it is called lateral feedback.

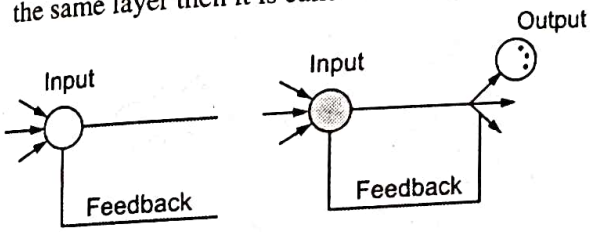



**Fig. 5.7.3**

## 5.7.4 Single Layer Recurrent Network

Recurrent networks are feedback networks with closed loop. Fig. 5.7.4 shows a single-layer network with a feedback connection in which a processing elements output can be directed back to the processing element itself or to the other processing element or to both.

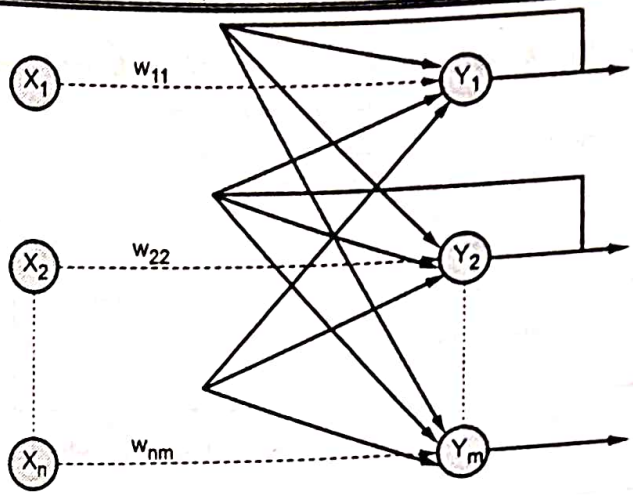



**Fig. 5.7.4 : Single-layer recurrent network**

## 5.7.5 Multilayer Recurrent Network

- The main feature of a multiplayer recurrent network is its hidden state, which captures information about a sequence.
- To form a multilayer recurrent network, a processing element output can be directed back to the nodes in a preceding layer. Also, in these networks, a processing element output can be directed back to the processing element itself and to other processing elements in the same layer.

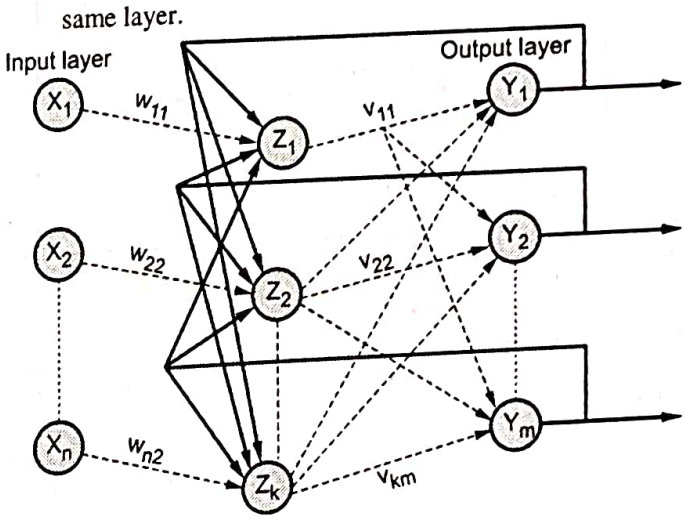
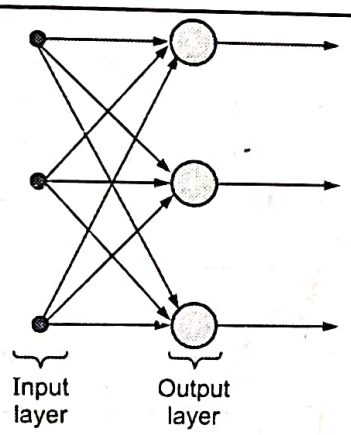
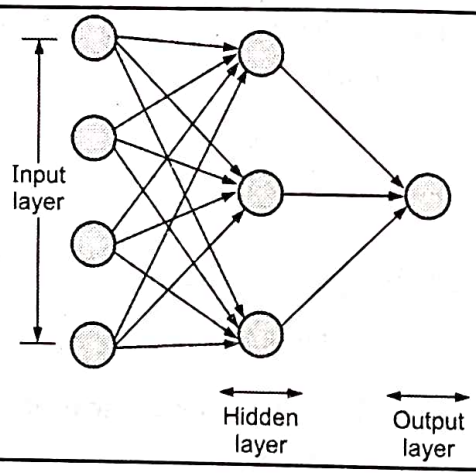



**Fig. 5.7.5 : Multiplayer recurrent network**

### 5.7.6 Difference between Multi Layer Feed Forward Neural Network and Signal Layer Feed Forward Neural Network

| Sr. No. | Single layer feed forward Neural Network | Multi layer feed forward Neural Network |
|---|---|---|
| 1. | Formation of layer is done by **processing** and **combining** elements with other processing elements. | Multi layer is formed by **interconnection** of **several** layers. |
| 2. | There is **linking** between **input** and **output** layer. | Between input and output layers, there are **multiple layers,** which are known as **hidden layers.** |
| 3. | Inputs are connected to the **processing** nodes with various **weights** resulting series of output **one for node.** | Input layers receive input and stores input signal and output layer generates output. |
| 4. | There is **no hidden layer.** | There are **several hidden layers** in a network. |
| 5. | In certain applications, it is **not efficient.** | More hidden layers create **complexity** of network, but the output is **efficient.** |
| |   |   |

## 5.8 APPLICATIONS OF NEURAL NETWORKS

**UQ.** What are the performance measures to see whether training of neural network is successful ? Explain.

**(MU - Q. 4(a), May 2016, 10 Marks)**

Variety of problems can be solved by applications of neural networks.

Some of the common applications are :

(1) Pattern Recognition (PR) (2) Image Processing (IP)

(3) Neural Networks are very useful in visual images handwritten characters, printed characters, speech and other PR based tasks.

(4) **Optimisation/Constraint Satisfactions**

To obtain optimal solution satisfying given constraints, such problems can be solved by NN successfully.

**Examples**

(i) Manufacturing scheduling,

(ii) Finding the shortest possible tour given a set of cities,

(iii) Problems of this nature arising out of industrial and manufacturing fields have been found acceptable solutions using NNS.

(5) **Forecasting and Risk Assessment :** Neural networks have shown capability to predict situations form past trends. There are ample applications in area such as meteorology, banking, stock market, econometrics with high success rates.

(6) **Control System :** NNS have gained commercial ground by applying applications in control systems. Companies incorporating NN technology have produced dozers of computer successfully.

(7) There are many applications for the control of chemical plants, robots and so on.

## 5.9 IMPORTANT TERMINOLOGIES OF ANN

### 5.9.1 Weights

(1) Each neuron, in the architecture of ANN, is connected to other neurons by means of direct communication links, and each links is associated with weights.

(2) Weights contain information about the input signal.

(3) This information is used by the net to solve a problem.

(4) The weight can be represented in terms of matrix. The weight matrix is also called as connection matrix.

(5) To make mathematical formulation, let there be 'n' processing elements in an ANN and each processing element has exactly m adaptive weights.

Thus the weight matrix w is defined as :

$$W = \begin{bmatrix} w_1^T \\ w_2^T \\ \vdots \\ w_n^T \end{bmatrix} \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & & & \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{bmatrix}$$

Where, $w_i = [w_{i1}, w_{i2}, \cdots, w_{im}]^T$

i = 1 to n. is the weight vector of processing element

And wij is the weight from processing element 'i' (source node) to processing element 'j' (destination node).

(6) The set of all W matrices will determine the set of all information for the ANN.

(7) The ANN can be realised by finding an appropriate matrix w.

(8) The weights encode long*term memory (LTM) and the activation states of neurons encode short-term memory (STM) in a neural network.

### 5.9.2 Bias

(1) The bias is included in the network.

(2) It has impact in calculating the net input.

(3) The input vector becomes, $X = \{1, X_1\ X_2, ..., X_n\}$

(4) The bias is another weight, say $w = b_1$.

(5) Bias plays a major role in determining the output of the network.

(6) The bias is of two types :

(a) Positive bias increases the net input of the network and

(b) The negative bias decreases the net input of the network.

(7) Due to bias-effect, the output of the network can be varied.

### 5.9.3 Threshold

(1) Threshold is a set value on which the final output of the network may be calculated.

(2) In activation function, threshold value is used.

(3) To obtain the network output, a comparison is made between the calculated net input and the threshold.

(4) There is a threshold limit for each and every applications.

(5) The activations function using threshold is defined as

$$f(net) = \begin{cases} 1 & \text{if net} \geq \theta \\ -1 & \text{if net} < \theta; \end{cases};$$

Where θ is fixed threshold value.

## 5.10 LIMITATIONS OF NEURAL NETWORKS

(1) The NN needs training to operate

(2) The architecture of a neural network is different from the architecture of microprocessors. Hence emulation is necessary.

(3) Requires high processing time for large neural networks.

## 5.11 INTRODUCTION TO PERCEPTRON

### 5.11.1 Perceptron Networks

Single-layer feed-forward network is also called as simple perception.

**Single layer feed forward network**

This type of network comprises two layers, the **input layer** and the **output layer.** The **input layer neurons** receive the input signals and the **output layer** neurons receive the output signals. The synaptic links carrying the weights connect every **input** neuron to the **output neuron** but not in the reverse way. Such a network is said to be feed forward in type or acyclic in nature. Since the output single layer alone performs computation, hence it is called as single layer feed forward network.

## 5.12 KEY-POINTS

In a Perceptron network, the key points are

(i) Perceptron-network consists of three units, namely, sensory unit (input unit), associate unit (hidden unit), response unit (output unit).

(ii) The sensory units are connected to associator units with fixed weights having values 1, 0 or – 1, which are given at random.

(iii) The binary step is used with fixed threshold θ as activation for associator.

(iv) The response unit has an activation of 1, 0 or – 1.

(v) The output of the perception network is given by, y = $f(y_{in})$, where

$f(y_{in})$ is the activation function defined by

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

$y_{in}$ is the input unit

(vi) The perceptron learning rule is used in the weight updation between the associator unit and the response unit.

(vii) The weights on the connections from the units that send the non-zero signal will get adjusted suitably.

(viii) The weights will be adjusted if an error has occurred for a particular training pattern, i.e.,

$$w_i \text{ (new)} = w_i \text{ (old)} + \alpha + xi \quad \text{and} \quad b \text{ (new)} = b \text{ (old)} + \alpha \cdot t$$

Here 't' is target value; and it is + 1 or – 1 and 'α' is the learning rate.

### 5.12.1 The Perceptron Rule Convergence Theorem

**The theorem states that :** "If there is a weight vector W, such that f(x(n) w) = t(n), for all n, then for any starting vector $w_1$, the perceptron learning rule will converge to a weight vector that gives the correct response for all training patterns, and this learning takes place within a finite number of steps provided that the solutions exists.

Refer Fig. 5.12.1.

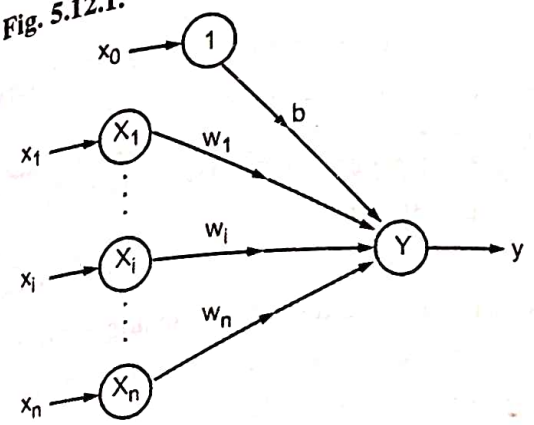



**Fig. 5.12.1 : Single classification perceptron network : [A simple perceptron network architecture]**

## 5.13 ARCHITECTURE

In the original perceptron network, the output obtained from the associator units is a binary vector, and hence that output can be taken as input signal to the response unit, and then classification can be made.

In Fig. 5.12.1, a simple perceptron network architecture is shown.

In the above diagram, there are 'n' input neurons, 1 output neuron and a bias. The input-layer and output-layer neurons are connected through a directed communication link, which is given weights. The aim of the perceptron net is to classify the input pattern as a **member.**

### 5.13.1 Flowchart for Training Process

The training is done on the basis of the comparison between the calculated and desired output. The loop stands terminated it there is no change in weights.

**Remark**

First one has to perform the basic. Initialisation required for the training process.

**Flowchart for Perceptron network with single output**

Start

↓

Initialise weights and bias and set learning rate $\alpha$ (0 to 1)

↓

Activate input units $x_i$

↓

Calculate net input and call it as yin

↓

Apply activation function to $y_{in}$ to find output y as : y = f(y, n)

↓

If y = t, then find new weights,

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

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

↓

and proceed as above.

If $y \neq t$; then process is over.

## 5.14 PERCEPTRON TRAINING ALGORITHM FOR SINGLE OUTPUT CLASSES

In the following algorithm the inputs are initially assigned and then the net input is calculated. Applying activation formula over the calculated net input, the output of the network is obtained.

We compare the calculated and desired output, and then carry out the weight updation. The perceptron algorithm is used for either binary or bipolar input vectors, having bipolar targets, with varying bias and fixed threshold. The entire network is based on stopping criterion.

**The algorithm function as follows :**

- **Step (I) :** First we initialise weights and bias. For easy calculation, we put them equal to zero. And learning rate $\alpha$ is between : $0 < \alpha \leq 1$. Conveniently, we take $\alpha = 1$.
- **Step (II) :** We perform the steps (III) to (vii) until the final stopping condition is false.
- **Step (III) :** We apply identity activation functions containing the input layer containing input, unit.
- **Step (IV) :** We 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.

Now to obtain the output, we apply activation formula over the net input :

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

- **Step (V) :** Now we compare the actual output (calculated output) and desired (target) output :

If $y \neq t$, then we use the formulae :

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

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

else, we have

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

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

- **Step (VI) :** We train the network until there is no weight change.

This is called stopping condition for the network. If this condition is not satisfied, we start again from Step (IV).

**Remark**

This algorithm is not sensitive to the initial values of the weights or the valve of the learning rate.

## 5.15 MULTILAYER FEEDFORWARD NETWORK

This network is made up of multiple layers. The architecture of this class consists of one or more intermediary layers called as hidden layers, besides having an input and an output layer. The hidden layers perform intermediary computations before directing the input to the output layer. The input layer neurons are linked to the hidden layer neurons and the weights on these links are called as **input-hidden layer weights.**

Again, the hidden layer neurons are linked to the output layer neurons and the corresponding weights are referred to as **hidden-output layer weights.**

### 5.15.1 Perceptron Training Algorithm for Multiple output Classes

We mention below the Algorithm of Perceptron training for multiple output classes :

- **Step (I) :** Initialise the weights, biases and learning rate conveniently.
- **Step (II) :** Check for stopping condition; if it is not true, perform steps 3-7.
- **Step (III) :** For each bipolar or binary training vector pair S : t, perform steps 4–6.
- **Step (IV) :** Set activation function (identity function) of each input unit i = 1 to n :

$$xi = si$$

- **Step (V) :** Calculate the net input as,

$$yinj = bj + \sum_{i=1}^{n} xi\ wij$$

Then calculate output response of each output unit j = 1 to m : To calculate the output response, apply activation formula over the net input :

$$y_j = f(y_{inj}) = \begin{cases} 1, & \text{if } y_{inj} > \theta \text{ ; } \theta \text{ is threshold valve} \\ 0, & \text{if } -\theta \le y_{inj} \le \theta \\ -1, & \text{if } y_{inj} < -\theta \end{cases}$$

- **Step (VI) :** If $t_j \neq y_j$, then make adjustment in weights and bias for j = 1 to m and i = 1 to n.

i.e. $w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha\, t_j\, x_i$

$b_j(\text{new}) = b_j(\text{old}) + \alpha\, t_j$

Otherwise we have

$w_{ij}(\text{new}) = w_{ij}(\text{old})$

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

- **Step (VII) :** If there is no change in weights then stop the training process, otherwise begin from step (III). We exhibit the architecture for the above algorithm :

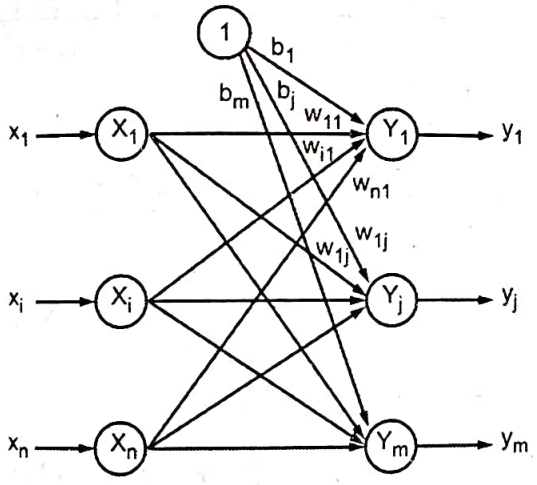



**Fig. 5.15.1 : Network architecture for perceptron network for several output classes**

## 5.16 ACTIVATION FUNCTION

**UQ.** Explain common activation functions used in neural network.

**(MU - Q. 1(b), May 2011, 5 Marks)**

(1) A model of the behaviour of a neuron can be presented as shown in Fig. 5.16.1.

Here $x_1, x_2, ..., x_n$ are the n-inputs to the artificial neuron. $w_1, w_2, ..., w_n$ are the weights attached to the input links.

(2) **Biological** neuron receives all inputs through the dendrites, sums them and produces an output. If the sum is greater than a threshold value. The input signals are passed on to the cell body through the synapse which may accelerate or retard an arriving signal.

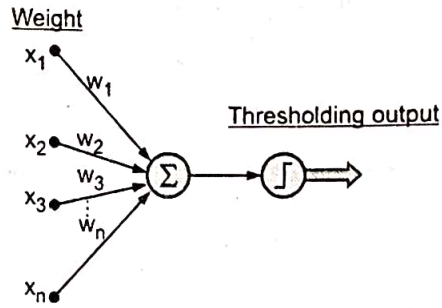



**Fig. 5.16.1 : Simple model of an artificial neuron**

(3) Here weights model the acceleration or retardation of the input signals. In short, weights are multiplicative factors of the inputs.

(4) The total input (say I) received by the soma (body) of the artificial neuron is

$$I = w_1 x_1 + w_2 x_2 + ... + w_n x_n ...(i)$$

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

(5) This sum is passed on to a non-linear filter $\phi$ called Activation function, or Transfer function, or Squash Function which releases the outputs.

i.e. $y = f(I)$ ...(ii)

### 5.16.1 Activation Functions

**UQ.** What are the important properties of activation function used in neural networks ?

**(MU - Q. 1(c), May 2016, 5 Marks)**

**UQ.** List the different activation functions used in neural network.

**(MU - Q. 1(d), May 2017, 5 Marks)**

**UQ.** With mathematical list four different activation functions used in neurons.

**(MU - Q. 1(e), May 2019, 5 Marks)**

The obtain exact output, the activation function is applied over the net input of an ANN.

There are several activation functions. Here we consider a few :

1. **Linear function :** It is defined as

$$f(x) = x \text{ for all } x$$

$$\text{Here input} = \text{output}$$

2. **Bipolar Step function :** The function is defined as :

$$f(x) = \begin{cases} 1, & \text{if } x \geq \theta \\ -1, & \text{if } x < \theta \end{cases}$$

Where θ is threshold value. The function is also used in single-layer nets to convert the net input to an bipolar output (+ 1, – 1).

3. **Binary step function :** The function is defined as :

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

This function is used in single-layer nets to convert the net input to an output that is binary (0 or 1).

4. **Sigmoidal function :** This function is used in back-propagation nets. They are of two types :

(i) **Binary sigmoidal function :** It is also called as unipolar sigmoid function or a logistic sigmoid function, and defined as

$$f(x) = \frac{1}{1+e^{-\lambda x}}, \text{ where } \lambda \text{ is steepness parameter}$$

The derivative of f(x) is :

$$f'(x) = \frac{-1}{[1+e^{-\lambda x}]^2} \cdot (-\lambda e^{-\lambda x}) = \frac{\lambda (e^{-\lambda x})}{[1+e^{-\lambda x}]^2} = \frac{\lambda [1+e^{-\lambda x} - 1]}{[1+e^{-\lambda x}]^2}$$

$$= \lambda \left\{ \frac{1}{(1+e^{-\lambda x})} - \frac{1}{(1+e^{-\lambda x})^2} \right\}$$

$$= \lambda \left[ \frac{1}{(1+e^{-\lambda x})} \left\{ 1 - \frac{1}{(1+e^{-\lambda x})} \right\} \right]$$

$$= \lambda [f(x)(1 - f(x))] = \lambda f(x) [1 - f(x)]$$

Range of this sigmoid function is 0 to 1 [∵ denominator is always greater than or equal to 1]

(ii) **Bipolar sigmoid function :**

The function is given by

$$f(x) = \frac{2}{1+e^{-\lambda x}} - 1 = \frac{2 - 1 - e^{-\lambda x}}{1+e^{-\lambda x}} = \frac{1 - e^{-\lambda x}}{1+e^{-\lambda x}}$$

where again, λ is steepress parameter and the range is between – 1 and + 1.

Derivative of f(x) is :

$$f'(x) = \frac{[1+e^{-\lambda x}][\lambda e^{-\lambda x}] - (1 - e^{-\lambda x})(-\lambda e^{-\lambda x})}{(1+e^{-\lambda x})^2}$$

$$= \frac{\lambda e^{-\lambda x}[1 + e^{-\lambda x} + 1 - e^{-\lambda x}]}{(1+e^{-\lambda x})^2} = \frac{2\lambda e^{-\lambda x}}{(1+e^{-\lambda x})^2}$$

$$= \frac{\lambda}{2}\left[\frac{4e^{-\lambda x}}{(1+e^{-\lambda x})^2}\right]$$

$$= \frac{\lambda}{2}\left[\frac{(1+e^{-\lambda x})^2 - (1-e^{-\lambda x})^2}{(1+e^{-\lambda x})^2}\right]$$

$$= \frac{\lambda}{2}\left[1 - \left(\frac{1-e^{-\lambda x}}{1+e^{-\lambda x}}\right)^2\right] = \frac{\lambda}{2}[1 - f(x)^2]$$

$$= \frac{\lambda}{2}[(1 + f(x))(1 - f(x))]$$

$$[\because a^2 - b^2 = (a+b)(a-b)]$$

☞ **Remark**

(1) Here, $f(x) = \frac{1 - e^{-\lambda x}}{1+e^{-\lambda x}} = \tanh\left(\frac{\lambda x}{2}\right)$

$$\therefore f'(x) = \frac{\lambda}{2}\operatorname{sech}^2\left(\frac{x}{2}\right) = \frac{\lambda}{2}\left[1 - \tanh^2\left(\frac{x}{2}\right)\right]$$

$$= \frac{\lambda}{2}\left[\left(1 + \tanh\frac{x}{2}\right)\left(1 - \tanh\frac{x}{2}\right)\right]$$

$$f'(x) = \frac{\lambda}{2}[(1 + f(x))(1 - f(x))]$$

(2) Sigmoid functions are so-called because their graphs are "S-shaped".

(i) Simple sigmoids defined to be odd, are monotone functions of one variable,

(ii) Hyperbolic sigmoids are subset of simple sigmoids and natural generalisation of the hyperbolic tangent function.

(3) **Ramp function**

It is defined as

$$f(x) = \begin{cases} 1, & \text{if } x > 1 \\ x, & 0 \le x \le 1 \\ 0, & x < 0 \end{cases}$$

**Graphs of activation functions**

**(1) Linear function**

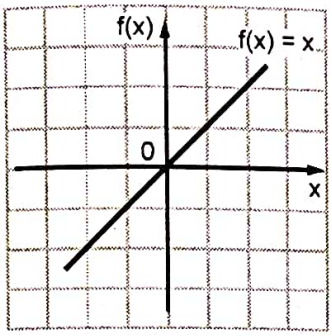



Fig. 5.16.2

**(2)**

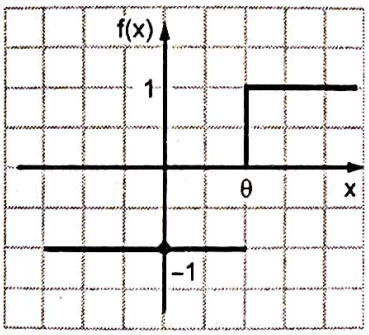



Fig. 5.16.3

**(3) Binary-step function**

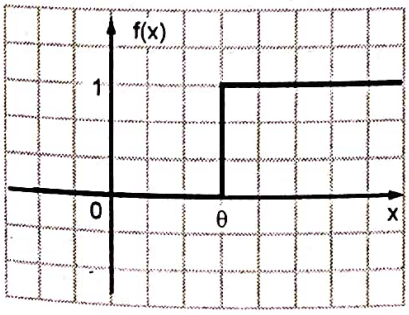



Fig. 5.16.4

**(4)**

(i) **Binary sigmoidal function**

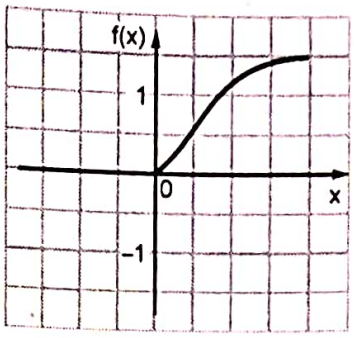



Fig. 5.16.5(i)

(ii) **Bipolar sigmoidal function**

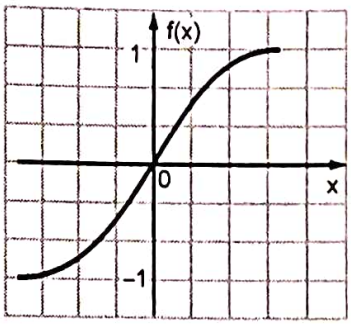



Fig. 5.16.5(ii)

**(5) Ramp function**

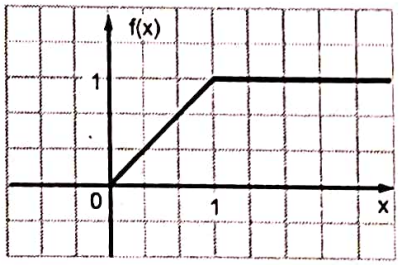



Fig. 5.16.6

## 5.16.2 Illustrative Examples for Logistic Function

**Ex. 5.16.1 :** Logistic function is given by,

$$F(v) = \frac{1}{1 + \exp(-av)}$$

Show that the derivative of f(v) w.r.t. v is given by,

$$\frac{df}{dv} = af(v)\,[1 - f(v)]$$

What is the value of this derivative at the origin ?

☑ **Soln. :**

▶ **Step (I) :** We have, $f(v) = \dfrac{1}{1 + \exp(-av)}$

Different w.r.t. v ; we get

$$f(v) = \frac{-1}{[1 + \exp(-av)]^2} [\exp(-av)(-a)]$$

$$= \frac{a \exp(-av)}{[1 + \exp(-av)]^2}$$

$$= \frac{a}{[1 + \exp(-av)]} \frac{\exp(-av)}{[1 + \exp(-av)]}$$

$$= a f(v) \left[\frac{\exp(-av)}{[1 + \exp(-av)]}\right]$$

$$= a f(v) \left[1 - 1 - \frac{\exp(-av)}{1 + \exp(-av)}\right]$$

$$= a f(v) \left[1 - \left\{1 + \frac{\exp(-av)}{1 + \exp(-av)}\right\}\right]$$

$$= a f(v) \left[1 - \left\{\frac{1 + \exp(-av) - \exp(-av)}{1 + \exp(-av)}\right\}\right]$$

$$= a f(v) \left[1 - \left\{\frac{1}{1 + \exp(-av)}\right\}\right]$$

$$\therefore f'(v) = a f(v) [1 - f(v)]$$

▶ **Step (II) :** As $V \to 0$; $\exp(-av) = \exp(0) = 1$

$$\therefore f(v) = \frac{1}{1+1} = \frac{1}{2}$$

$\therefore$ $f'(v)$ at $v = 0$ is,

$$= a \cdot \frac{1}{2}\left[1 - \frac{1}{2}\right] = \frac{9}{4}$$

---

**Ex. 5.16.2 :** An odd sigmoid function is given by $f(v) = \dfrac{1 - \exp(-av)}{1 + \exp(-av)} = \tan h\left(\dfrac{av}{2}\right)$.

Show that the derivative of f(v) w.r.t. V is given by, $\dfrac{df}{dv} = \dfrac{a}{2}[1 - f^2(v)]$.

What is the valve of this derivative at the origin ? Suppose that the slope parameter a is made infinitely large. What is the resulting form of f(v).

☑ **Soln. :**

▶ **Step (I) :** For convenience we take, $f(v) = \tan h\left(\dfrac{av}{2}\right)$

Differentiating w.r.t. V;

$$\frac{df}{dv} = \operatorname{sech}^2\left(\frac{av}{2}\right) \cdot \frac{a}{2}$$

$$= \frac{a}{2}\left[1 - \tan h^2\left(\frac{av}{2}\right)\right]$$

$$[\because \operatorname{sech}^2 x = 1 - \tanh^2 x]$$

$$= \frac{a}{2}[1 - f^2(v)]$$

$$\therefore \frac{df}{dv} = \frac{a}{2}[1 - f^2(v)] \quad \text{...(i)}$$

▶ **Step (II) : We have $f(v) = \tanh\left(\dfrac{av}{2}\right)$**

At $v = 0$, $f(0) = \tan h(0) = 0$. $\therefore$ At origin, from Equation (i), $\dfrac{df}{dv} = \dfrac{a}{2}[1 - 0] = \dfrac{a}{2}$

As $a \to \infty$, $\tan h(\infty) = 1$. $\therefore$ As $a \to \infty$, $f(v) \to 1$.

---

**Ex. 5.16.3 :** The algebraic sigmoid function is given by, $f(v) = \dfrac{v}{\sqrt{1 + v^2}}$

Show that the derivative of f(v) w.r.t. v is given by, $\dfrac{df}{dv} = \dfrac{\phi^3(v)}{v^3}$

What is the value of this derivative at origin ?

☑ **Soln. :**

▶ **Step (I) :**

Differentiating $f(v) = \dfrac{v}{\sqrt{1 + v^2}}$ ; we get

$$f'(v) = \left[\frac{\sqrt{1 + v^2} \cdot 1 - v \cdot \dfrac{1 \cdot 2v}{2\sqrt{1 + v^2}}}{(1 + v^2)}\right]$$

$$= \left[\frac{\sqrt{1 + v^2} - \dfrac{v^2}{\sqrt{1 + v^2}}}{(1 + v^2)}\right] = \left[\frac{(1 + v^2) - v^2}{(1 + v^2)^{3/2}}\right]$$

$$\therefore f'(v) = \frac{1}{(1 + v^2)^{3/2}} \quad \text{...(i)}$$

$$\therefore f'(v) = \frac{1}{v^3}\left[\frac{v^3}{(1 + v^2)^{3/2}}\right] = \frac{1}{v^3}\left[\frac{v}{(1 + v^2)}\right]^3$$

$$= \frac{1}{v^3}[f(v)]^3 = \frac{f^3(v)}{v^3} \quad \text{...(ii)}$$

**Step (II) :** As $v \to 0$ $f'(v) = \frac{1}{(1+v^2)^{3/2}} \to 1$.

$\therefore$ At origin, $f'(v) = 1$.

## 5.17 ADVANTAGES, DISADVANTAGES AND LIMITATIONS OF PERCEPTRON

### 5.17.1 Advantages of Perceptron (Multi-layer)

(i) A multi-layered perceptron model can be used to solve complex non-linear problems.

(ii) It works well with both small and large input data

(iii) It helps us to obtain quick predications after the training.

(iv) It helps to obtain the same accuracy ratio with large as well as small data.

### 5.17.2 Disadvantages of Perceptron (Multi-layer)

(i) In multi-layer perceptron, computations are difficult and time-consuming,

(ii) In multi-layer perceptron, it is difficult to predict how much the dependent variable affects each independent variable

(iii) The model functioning depends on the quality of the training.

### 5.17.3 Limitations of Perceptron

(i) The output of a perceptron can only be a binary number (0 or 1) due to the hard limit transfer function.

(ii) Perceptron can only be used to classify the linearly separable sets of input vectors.

If input vectors are non-linear, it is not easy to classify them properly.

**Remark**

The perceptron model is continuously becoming more advanced and working efficiently on complex problems with the help of artificial neurons.

## 5.18 PROGRAM FOR PERCEPTRON TRAINING ALGORITHM

**Program 5.18.1 :** Write a program for solving linearly separable problem using Perceptron Model.

**Source code**

```
%Perceptron for AND function
clear;
clc;
x=[1 1 -1 -1;1 -1 1 -1];
t=[1 -1 -1 -1];
w=[0 0];
b=0;
alpha=input('Enter Learning rate=');
theta=input('Enter Threshold value=');
con=1;
epoch=0;
while con
  con=0;
    for i=1:4
      yin=b+x(1,i)*w(1)+x(2,i)*w(2);
        if yin>theta
        y=1;
        end
          if yin <=theta & yin>=-theta
          y=0;
          end
          if yin<-theta
          y=-1;
          end
          if y-t(i)
          con=1;
          for j=1:2
              w(j)=w(j)+alpha*t(i)*x(j,i);
          end
          b=b+alpha*t(i);
```

```
            end
        end
 epoch=epoch+1;
end
disp('Perceptron for AND function');
disp(' Final Weight matrix');
disp(w);
disp('Final Bias');
disp(b);
save 'result.mat','w','b';
```

☞ **Output**

```
Enter Learning rate=
0.5
Enter Threshold value=
1
Perceptron for AND function
 Final Weight matrix
   1.5000    1.5000

Final Bias
  -1.5000
```

### 5.18.1 Perceptron Network Testing Algorithm

(I) For efficient performance of the network, it is to be trained with more data.

We mention the testing algorithm :

- **Step (I) :** The initial weights to be used are to be taken from the final weights obtained during training.
- **Step (II) :** Perform steps 3-4 for each input vector X.
- **Step (III) :** Set activations formula of the input unit.
- **Step (IV) :** Obtain the response of output unit :

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

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

With these steps the performance of the algorithm can be tested.

➢ **Program 5.18.2 :** Test the above trained perceptron network (Using MATLAB)

```
%This is the recall program for perceptron_AND
%Trained weight vector and bias value is saved in file
'result.mat' which is loaded here using load command
load result.mat;
x=input('Enter the test pattern');
yin=b+x(1)*w(1)+x(2)*w(2);
 if yin>theta
   y=1;
 end
 if yin <=theta & yin>=-theta
   y=0;
 end
 if yin<-theta
  y=-1;
 end
 disp('output is');
 y
```

☞ **Output**

```
Enter the test pattern
[-1 1]
output is
y =
   -1
```

### 5.18.2 Recurrent Networks

These networks differ from feedforward network architecture, as it has at least one feedback loop; i.e. there may be at least one layer with feedback connections. There may be also self-feedback links, i.e., the output of a neuron is feedback into itself as input. Refer Fig. 5.18.1.

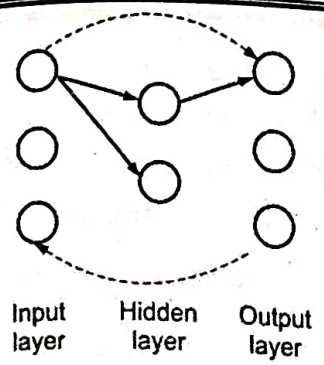



**Fig. 5.18.1 : A recurrent neural network**

### 5.18.3 Characteristics of Neural Network

(i) They can map input patterns to their associated output patterns.

(ii) The neural networks learn by examples.
They are 'trained' with known examples of a problem. Then they can infer the solution to the unknown problem.

(iii) From the past 'training', they can predict new outcomes of the problem. They possess the capability of generalisation.

(iv) Even if patterns are incomplete, they produce full patterns.

(v) They can process information at high speed.

### 5.18.4 Learning Method

Learning method, can be of three basic types :

1. **Supervised learning :** Here learning process is under observation. Every input pattern is associated with output pattern, and that is the desired pattern. Here error can be determined between network's computed output and correct expected output. From the error, network parameters can be changed. And that results in improved performance.

2. **Unsupervised Learning :** In this type, the system learns of its own accord by adapting itself to the features of the input patterns. Here target output is not presented to the network. It itself discovers output patterns.

3. **Reinforced learning :** In this type the information given to the input helps the network in its learning process. Here the system indicates whether the computed output is correct or not. But this method of learning is not so popular, form of learning. Here we mention some widely used learning rules :

4. **Hebbian learning :** This rule is based on correlative weight adjustment. In this method, the input-output patterns pairs $(X_i, Y_i)$ are associated by weight matrix w, and is known as correlation matrix. It is computed by,

$$W = \sum_{i=1}^{n} X_i Y_i^T$$

Where, $Y_i^T$ is the transpose matrix of the associated output vector $Y_i$.

5. **Gradient descent learning :** This method is based on minimising the error E which is defined in terms of weights and activation function of the network.

Since weight update in this method depends on the gradient of the error E, hence activation function in the network must be differentiable. If $\delta_{wij}$ is the weight update of the link connecting the $i^{th}$ and $j^{th}$ neuron of the two adjacent layers, then

$$\delta w_{ij} = \alpha \frac{\partial E}{\partial w_{ij}};$$

Where $\alpha$ is learning rate parameter.

and $\frac{\partial E}{\partial w_{ij}}$ Is error gradient with respect to the weight $W_0$.

6. Hoff's delta rule and Back propagation rule belong to this type of learning mechanism.

7. **Competitive Learning :** As the name indicates, it is a 'winner–take–all' strategy. When an input pattern is presented to the system, all neurons in the layer compete and winning neuron undergoes weight adjustment.

8. **Stochastic Learning :** Here the weights are adjusted in a probabilistic fashion.

### 5.18.5 Rosenblatt's Perceptron

The network of perceptron contains three units the sensory unit S, Association unit A, and Response unit R.

The output of R could be determined by a step function with binary values (0, 1) and output is defined as

$$y_j = f(y_{inj}) = 1 \quad \text{if} \quad y_{inj} > 0$$
$$= 0, \quad \text{otherwise.}$$

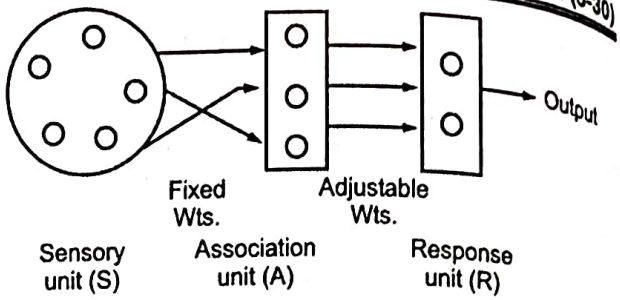



Fig. 5.18.2

Here $x_i$ is the input, $w_{ij}$ is the weight on the connection leading to the output units and $y_j$ is output. The training algorithm of the perceptron is a supervised learning algorithm where weights are adjusted to minimise error.

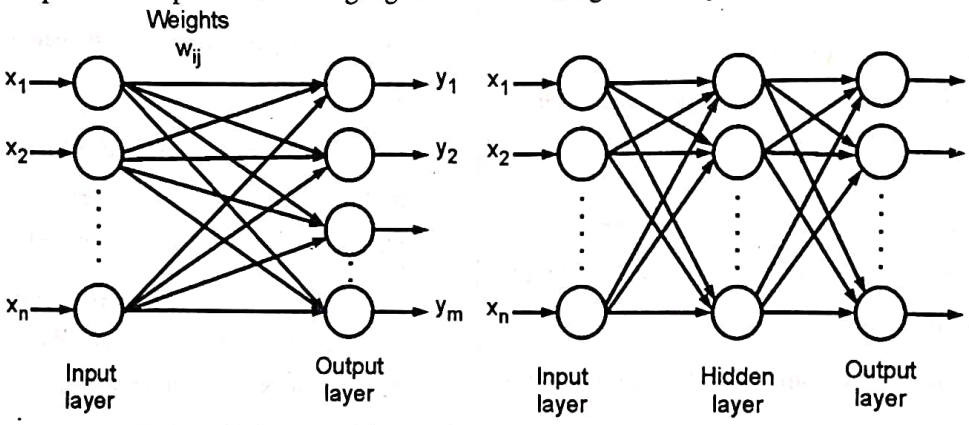



**Fig. 5.18.3 : A multilayer feed forward perceptron model**

### 5.18.6 Learning Algorithm for Training The Perceptron

A basic learning algorithm for training the perceptron is as follows :

(i) If the output is correct, then no adjustment of weights is done. i.e. $W_{ij}^{(k+1)} = W_{ij}^{(k)}$

(ii) If the output exceeds then the weights are decreased on the active input links.

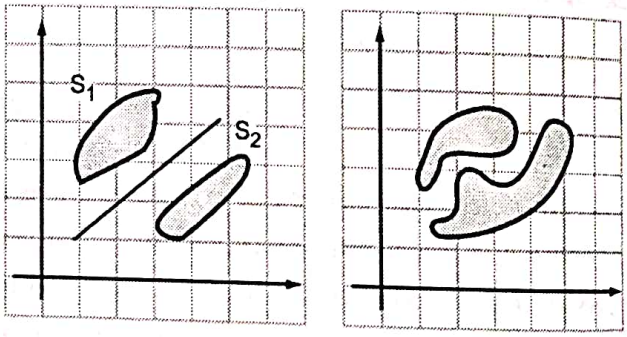
$$W_{ij}^{(k+1)} = W_{ij}^{(k)} - \alpha \cdot x_i$$

(iii) If the output is 0 but it should have been 1, then $W_{ij}^{(k+1)} = W_{ij}^{(k)} + \alpha \cdot x_i$

Where $W_{ij}^{(k+1)}$ is the new adjusted weight, $W_{ij}^{(k)}$ is the old weight, $x_i$ the input and $\alpha$ is the learning rate parameter. Perceptron cannot handle tasks which are not linearly separable.

In 2–dimension, the set of points are linearly separable if they can be separated by a st. line. Generalising, a set of points in n-dimensional space are linearly separable if there is a hyper plane of (n – 1) dimensions that separates the sets.

We exhibit linearly separable and non-linearly separable patterns by diagram.




(a) (b)

Fig. 5.18.4

**Remark**

For classification type of problems that are not linearly separable, the perceptron cannot find weights : An example is 'XOR' problem.

**We prepare truth-table**

The non-linear separable pattern of XOR problem. Here, odd parity means odd number of 1 bits in the inputs and even parity means even number of 1 bits in the inputs.

To classify the inputs as **odd parity** or **even parity**, the perceptron cannot separate even parity input patterns from odd parity input patterns, as is clear from Fig. 5.18.5. (The perception cannot find a line to separate the patterns).

XOR - Truth table

| Inputs | Inputs | Outputs | |
|---|---|---|---|
| 0 | 0 | 0 | Even parity |
| 1 | 1 | 0 | |
| 0 | 1 | 1 | Odd parity |
| 1 | 0 | 1 | |

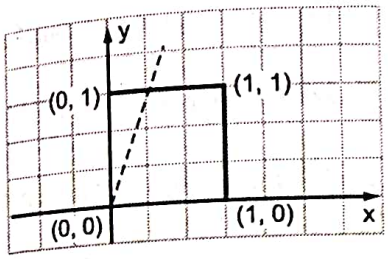



**Fig. 5.18.5 : The non-linear separable pattern of XOR problem**

## 5.19 ADALINE NETWORK

This network uses supervised learning. Here there is only one output neuron and output values are bipolar (- 1 or + 1). But the inputs $x_i$ can be binary, bipolar or real-values. The bias weight is $W_0$ with an input link of $x_0 = +1$.

If the weighted sum is greater than or equal to 0, then the output is 1, otherwise it is – 1.

The supervised learning algorithm of this network is similar to the perceptron learning algorithm. The learning algorithm is also known as Least Mean Square (LMS) or Delta-rule.

The rule is given by, $W_i^{new} = W_i^{old} + \alpha (t - y) x_i$

Where, $\alpha$ is the learning coefficient, t is the target output, y is the computed output, and $x_i$ is the input.

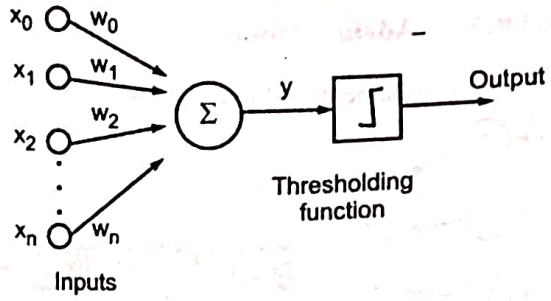



**Fig. 5.19.1 : A simple Adaline network**

Adaline network is used in all high-speed modems and telephone switching systems to cancel echo in long distance communication circuits.

### 5.19.1 Delta Rule

(i) The Delta-rule is similar to perceptron learning rule.

(ii) Delta rule is derived from gradient-descent method; and it can be generalised to more than one layer.

(iii) Perceptron learning rule stops after a finite number of learning steps, but the gradient-descent approach continues; and it converges only asymptotically to the solution.

(iv) The delta rule updates the weights between the connections so as to minimise the difference between the net input to the output unit and the target valve.

(v) The main aim of 'delta-rule' is to minimise the error over all training patterns.

This is achieved by reducing the error for each pattern, one at a time.

(vi) The delta-rule for adjusting the weight of $i^{th}$ pattern (i = 1 to n) is

$$\Delta w_i = \alpha (t - y_{in}) x_i$$

Where $\Delta w_i$ is the weight change ; $\alpha$ the learning rate; x the vector of activation function of input unit; $y_{in}$ the net input to output unit, and 't' the target output.

(vii) In case of several output units for adjusting the weight from $i^{th}$ input unit to the $j^{th}$ output unit, for each pattern, is given by,

$$\Delta w_{ij} = \alpha (t_j - y_{in}) x_i$$

### 5.19.2 Adaline Model

We exhibit Adaline model in Fig. 5.19.2.

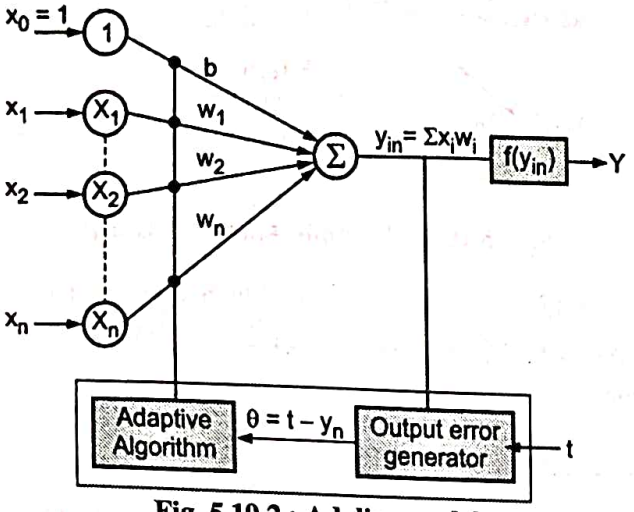



**Fig. 5.19.2 : Adaline model**

### 5.19.3 Architecture

Adaline is a single-unit neuron. It receives input from several units and also from one unit, bias. Inputs are either of the two values (+ 1 or – 1) and the weights have positive or negative signs.

The net input is calculated from activation functions, that restores the output to +1 or –1. The adaline model compares the actual output with the target output. And on the basis of the training algorithm the weights are adjusted.

### 5.19.4 Algorithm for Training Process

Now, we give a pictorial representation of the network training through flowchart.

**Training Algorithm**

- **Step (I) :** Set the non-zero values to weights and bias. Let $\alpha$ be the learning rate parameter.
- **Step (II) :** Steps 3 to 7 to be followed when stopping condition fails.
- **Step (III) :** Execute steps 4 to 6 for each bipolar training pair s : t.
- **Step (IV) :** Set activations for input units i = 1 to n.

$$x_i = s_i$$

- **Step (V) :** Calculate the net input

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

- **Step (VI) :** Update the weights and bias for i = 1 to n.

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

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

- **Step (VII) :** If the highest weight change that occurred during training is smaller than a specified tolerance then stop, the training process, else continue.

### 5.19.5 Testing of Algorithm

A step function is used to test the performance of the network. The method is as follows :

- **Step (I) :** Initialise the weights obtained from the training algorithm.
- **Step (II)** : Perform steps 3-5 for each bipolar input vector x.
- **Step (III)** : Set the activation function of the input units x.
- **Step (IV)** : Calculate the net input to the output unit.

$$Y_{In} = b + \Sigma x_i W_i$$

- **Step (V) :** Apply activation function over the net input calculated.

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

## 5.20 MULTIPLE ADAPTIVE LINEAR NEURONS : (MADALINE)

**Madaline Network**

A madaline network is a combination of a number of adalines. Using multiple Adalines, problem of non-linearity can be solved.

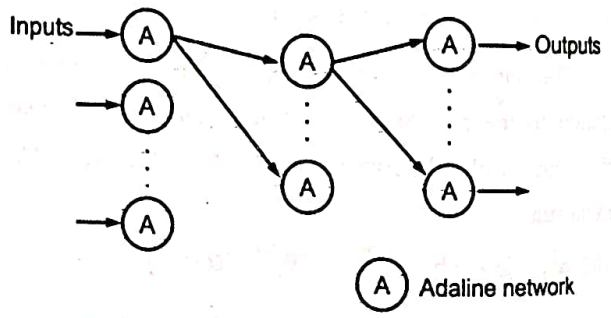



**Fig. 5.20.1**

The learning rule adopted by 'Madaline network' is called as 'Madaline adaptation Rule' (MR). It is a form of supervised learning. The aim is to adjust the weights so that error is minimum for the current training pattern.

### 5.20.1 Architecture

The simple Madaline architecture shown in the figure, consists of 'n' units of input layers, "m" units of Adaline layer and "1" unit of Madaline layer.

Each neuron in the Adaline and Madaline layers has a bias of excitation 1.

The Adaline layer is present between the input layer and the Madaline (output) layer. Hence the Adaline layer is regarded as a **hidden layer.**

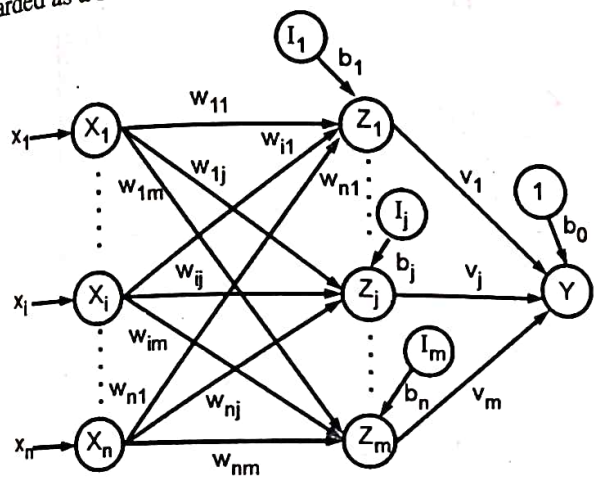



**Fig. 5.20.2 : Architecture of Madaline layer**

The Adaline and Madaline models can be applied effectively in communication systems and other cancellation circuits.

### 5.20.2 Flowchart of Training Process

In case of training, the weights between the input layer and the hidden layer are adjusted, and the weights between the hidden layer and the output layer are fixed. In the Madaline network, time taken for the training process is very high as compared to the Adaline network.

### 5.20.3 Training Algorithm

In the training algorithm, we have to adjust the weights between the hidden layer and the input layer. The weights for outer units are fixed.

Let the weights $v_1, v_2, ..., v_m$ and the bias $b_0$ that enter into output unit Y are so determined that the response of unit Y is 1.

So, we take the weights entering Y as,

$$v_1 = v_2 = ... = v_m = \frac{1}{2} \quad \text{and the bias } b_0 = \frac{1}{2}$$

The activation for the Adaline (hidden) and Madaline (output) units is given by,

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

### 5.20.4 Algorithm Preparation

- **Step (I)** : The weights entering the output unit are set as above. For Adaline, we set small random values.
- **Step (II) :** Stopping condition fails, we perform steps 3–4.
- **Step (III)** : For each bipolar training pair s : t, we perform steps 4–8.
- **Step (IV)** : Activate input layer units. For i = 1 to n,

$$x_i = s_i$$

- **Step (V)** : Calculate the net input to each hidden Adaline unit by using the formula :

$$Z_{inj} = b_j + \sum_{i=1}^{n} x_i w_i, \quad j = 1 \text{ to m.}$$

- **Step (VI)** : Calculate output of each hidden unit :

$$Z_j = f(Z_{inj})$$

- **Step (VII)** : Find out the output of the net

$$y_{in} = b_0 + \sum_{j=1}^{m} Z_j V_j$$

$$\therefore \; py = f(y_{in})$$

- **Step (VIII) :** Calculate the error and update the weights

(i) If t = y ; no updation of weight is required.

(ii) If $t \neq y$ ; and t = + 1, update weights on $Z_j$, where net input is closer to 0 (zero).

$$b_j \text{ (new)} = b_j \text{ (old)} + \alpha (1 - Z_{inj})$$

$$w_{ij} \text{ (new)} = w_{ij} \text{ (old)} + \alpha (1 - Z_{inj}) x_i$$

(iii) If $t \neq y$ and t = – 1, update weights on units $Z_k$ whose net input is positive :

$$w_{ik} \text{ (new)} = w_{ik} \text{ (old)} + \alpha (1 - Z_{ink}) x_i$$

$$b_k \text{ (new)} = b_k \text{ (old)} + \alpha (-1 - Z_{ink})$$

- **Step (IX) :** **Stopping condition** : If there is no weight change or if maximum number of iterations of weight updation are performed, then the process ends, or else continue the same process.

### 5.20.5 For Convenience, We Mention Here the Flowchart for Training of Madaline

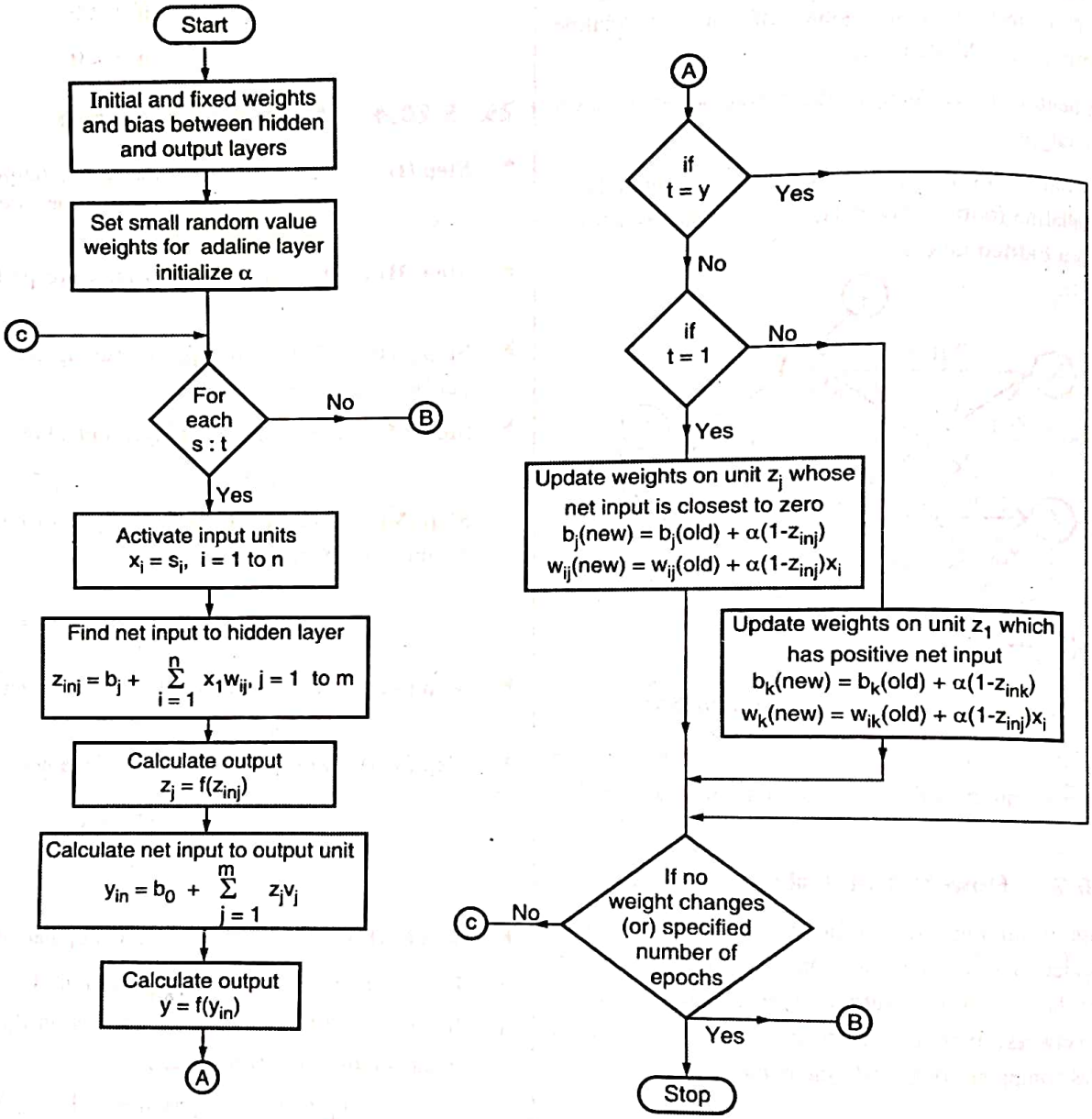



**Fig. 5.20.3 : Flowchart for Training of Madaline**

SPACE FOR NOTES

**Program 5.20.1 : Implement Madaline net using MATLAB.**

**Source code**

```
clc;
clear all;
x1=[-1 1 -1 1];
x2=[-1 -1 1 1];
x3=[1 1 1 1];
t=[-1 1 1 1];
w1=0.1;
w2=0.1;
b=0.1;
e=2;
alpha=0.1; delw1=0; delw2=0; delb=0; epoch=0;
while(e>1.0)
e=0;
for i=1:4
  ynet=w1*x1(i)+w2*x2(i)+b;
  error=t(i)-ynet;
  delw1=alpha*(error)*x1(i);
  delw2=alpha*(error)*x2(i);
  delb=alpha*(error)*x3(i);
  wchange=[delw1 delw2 delb];
  w1=w1+delw1; w2=w2+delw2;
  b=b+delb;
  w=[w1 w2 b];
  x=[x1(i) x2(i) x3(i)];
  disp(x)
  epoch=epoch+1
  error
end
  for i=1:4
    ynet=w1*x1(i)+w2*x2(i)+b;
    e=e+error^2;
  end
end
```

**Output**

```
  -1   -1    1
epoch =
   1
error =
  -0.9000
   1   -1    1
epoch =
   2
error =
   0.9900
  -1    1    1
epoch =
   3
error =
   1.0890
   1    1    1
epoch =
   4
error =
   0.4021
```

## 5.21 BACK-PROPAGATION NETWORK (BPN)

(i) Back-propagation network algorithm is applied to multilayer feed-forward networks consisting of processing elements.

(ii) The networks connected to back-propagation learning algorithm are also called as **back-propagation network (BPN).**

(iii) This algorithm provides a procedure for changing the weight in back-propagation network (BPN) to the given input pattern correctly.

(iv) The basic concept for this weight update is that where the error is propagated back to the hidden unit.

(v) In BPN, weights are calculated during the learning period of the network.

(vi) To update weights, the error must be calculated. There is no direct information of the error at the hidden layer. So we have to develop other techniques to calculate an error at the hidden layer, and this will cause minimisation of the output error.

(vii) Back propagation is a systematic method of training multilayer artificial neural networks.

It is developed on high mathematical foundation and it has very good application potential.

(viii) Back propagation learning rule is applicable on any feed forward network architecture.

(ix) Slow rate of convergence and local minima problems are its weaknesses.

(x) The multilayer feedforward (MLFF) network with back propagation (BP) learning is also called as 'multilayer perception' because of its similarity to perceptron networks with more than one layer.

(xi) **We carry out BPN as follows :**

(a) The feed forward of the input training pattern

(b) The back-propagation of the error and

(c) Updation of weights.

### 5.21.1 Architecture

A back-propagation neural network (BPN) consist of an input layer, hidden layer and an output layer. The neurons at the hidden and output layers have biases. The bias terms also act as weights. The outputs obtained could be either binary (0, 1) or bipolar (– 1, 1). Refer Fig. 5.21.1.

**We Exhibit the Architecture of BPN**

The terminologies used in the algorithm are as follows :

$x$ = input training vector$(x_1, x_2, \ldots, x_n)$

$t$ = target output vector

$(t_1, t_2, \ldots t_k, \ldots, t_m)$

$\alpha$ = learning rate parameter ;

$v_{oj}$ = bias on $j^{th}$ hidden unit,

$w_{ok}$ = bias on $k^{th}$ output unit ;

$Z_j$ = hidden unit j.

$x_j$ = input unit j,

(since identity activation function is used for input layer, the input and output signals are same).

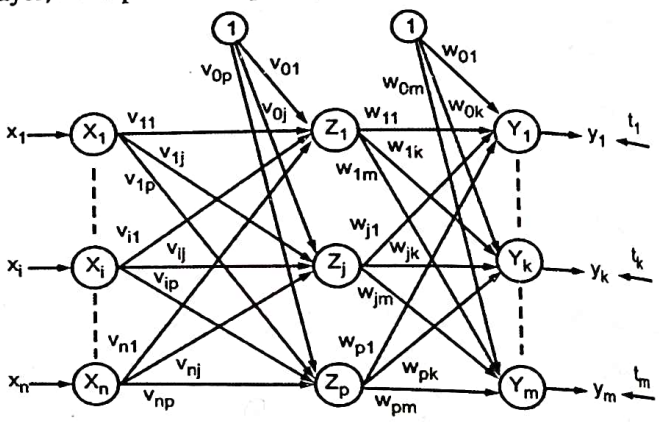



**Fig. 5.21.1 : Architecture of a back-propagation network**

The net input to $Z_j$ is,

$$Z_{inj} = V_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

and the output is, $Z_j = f(Z_{inj})$

$y_k$ = output unit k.

The net input to $y_k$ is,

$$y_{ink} = w_{ok} + \sum_{j=1}^{p} Z_j w_{jk}$$

and the output is $y_k = f(y_{ink})$

$\delta_k$ = error correction weight adjustment for $w_{jk}$, which is back-propagated to the hidden units that feed into unit $y_k$. and

$\delta_j$ = error connection weight adjustment for $v_{ij}$ and to the hidden unit $Z_j$.

### 5.21.2 Algorithm (Training)

We mention the error-back-propagation learning algorithm :

- **Step (I) :** Small random values for weights and learning rate,
- **Step (II) :** Carry on the steps 3-10 when stopping condition fails.
- **Step (III) :** Carry on steps 3-9 for each training pair.

**Phase (I) : Feed-forward phase**

- **Step (IV) :** Each input receives input signal xi and sends to hidden unit for i = 1 to n.
- **Step (V) :** Calculate net input

$$Z_{inj} = V_{oj} + \sum_{i=1}^{n} x_i v_{ij}$$

(To calculate output, we apply activation function $Z_{inj}$)(binary or bipolar sigmoidal activation function) : $Z_j = f(Z_{inj})$ and send the output signal to the input of output layer units.

- **Step (VI) :** For each output $y_k$ (k = 1 to m), calculate the net input :

$$y_{ink} = w_{ok} + \sum_{j=1}^{p} Z_j w_{jk}$$

and we apply activation function to evaluate output signal : $y_k = f(y_{ink})$

**Phase II + (Back-Propagation of Error)**

- **Step (VII) :** We compute error correction term for each output unit $y_k$ (K = 1 to m)

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

Where $f'(y_{ink})$ is derivative of $f(y_{ink})$.

Now, on the basis of the calculated error correction term, update the change in weights and bias :

$$\Delta w_{jk} = \alpha \delta_k z_j ;$$

$$\Delta w_{ok} = \alpha \delta_k$$

The error $\delta_k$ is to be sent to the **hidden layer backwards.**

- **Step (VIII) :** For each hidden unit ($Z_j$, J = 1 to p) ;

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk}$$

and to calculate the error term :

$$\delta_j = \delta_{inj} \cdot f'(Z_{inj})$$

Now, we update the change in weights and bias ;

$$\Delta V_{ij} = \alpha \delta_j x_i ; \quad \Delta V_{oj} = \alpha \delta_j$$

**Phase (III) : Weight and Bias Updation**

- **Step (IX) :** Each output unit ($y_k$, k = 1 to m) updates the bias and weights :

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk} ;$$

$$w_{ok}(new) = w_{ok}(old) + \Delta w_{ok}$$

Each hidden unit ($Z_j$, j = 1 to p) update its bias and weights :

$$V_{ij}(new) = V_{ij}(old) + \Delta V_{ij}$$

$$\text{and} \quad v_{oj}(new) = V_{oj}(old) + \Delta V_{oj}$$

- **Step (X) :** Check for stopping condition.

### 5.21.3 Flow-chart for Back-Propagation Network Training

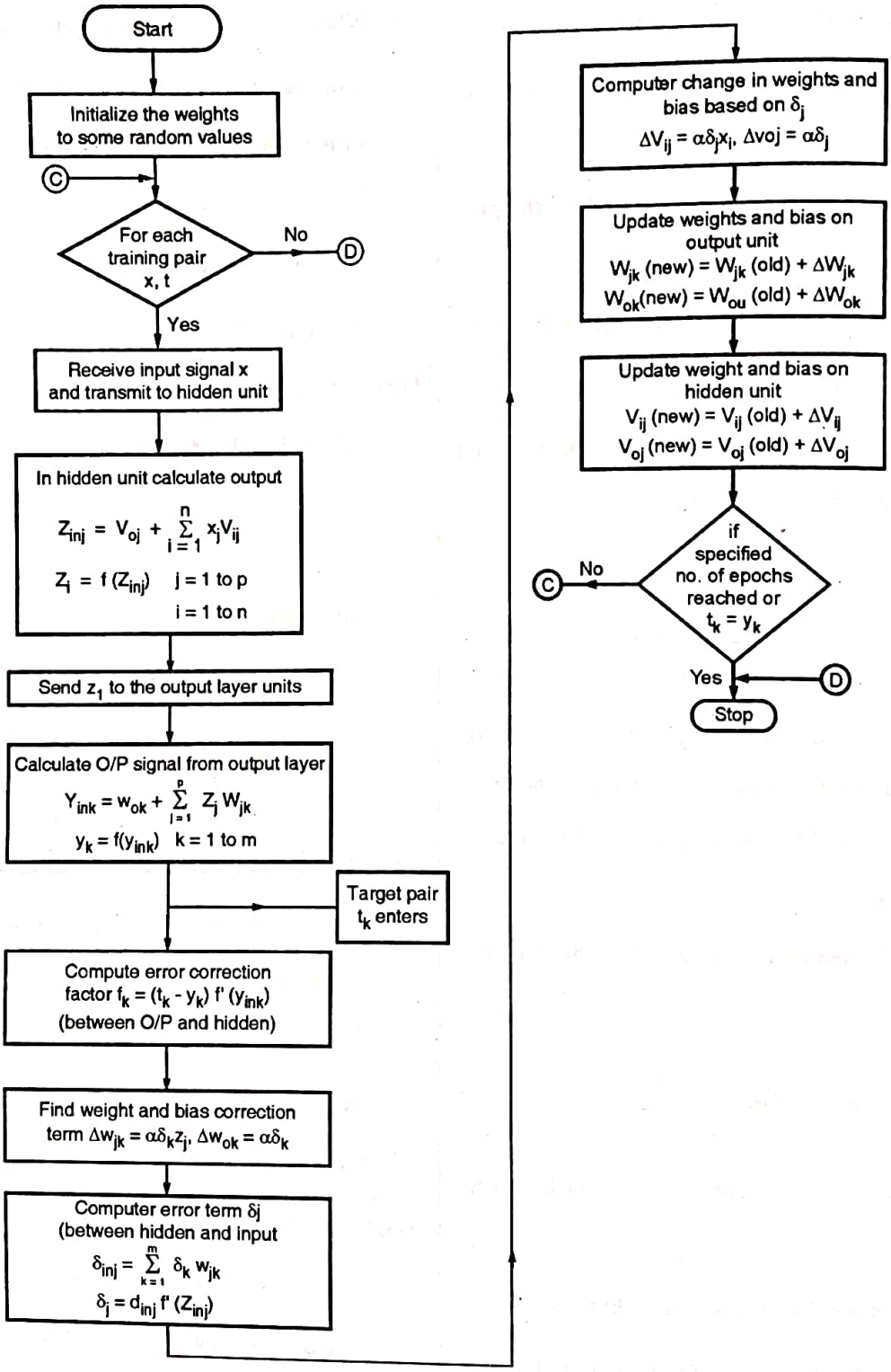



**Fig. 5.21.2 : Flow-chart for Back-Propagation Network Training**

## 5.22 LEARNING FACTORS OF BACK PROPAGATION NETWORK

(i) Learning rate $\alpha$ determines the size of the weight adjustments made at each iteration and hence influences the rate of convergence.

(ii) Poor choice of the rate can result in a failure in convergence.

It is convenient to keep the rate constant through all the iterations for best results. If the learning rate $\alpha$ is too large, the search path will oscillate and converges more slowly than a direct descent. The range of $\alpha$ from $10^{-3}$ to 0.9 is optimistic and has been used successfully for several back-propagation algorithmic experiments. If the rate is too small, the descent will progress in small steps, increasing the time to converge. Jacobs has suggested the use of adaptive coefficient where the value of the learning rate is the function of error derivative on successive updates.

## 5.23 CALCULATION OF ERROR

Consider any $r^{th}$ output neuron. Let 'O' be the output for which the target output is 'T'. The **error norm** in output for the $r^{th}$ output neuron is given by

$$E_r^1 = \frac{1}{2} e_r^2 = \frac{1}{2}(T-0)^2 \quad ...(i)$$

Where '$e_r$' is the error in the $r^{th}$ neuron. Error may be positive or negative. To consider only positive values, we consider the square of the error, i.e., we consider only absolute valve.

For the **first training pattern,** the Euclidean norm of error $E^1$ is given by

$$E^1 = \frac{1}{2} \sum_{r=1}^{n} (T_{or} - D_{or})^2 \quad ...(ii)$$

If we consider error function for all the training patterns, we get

$$E(V, W) = \sum_{j=1}^{n\,set} E^j (V, W, I), \text{ where}$$

E is the error function depending on the m (1 + n) weights of W and V. Based on error signal, neural network modifies its synaptic connections to improve the system performance.

### 5.23.1 Method of Steepest Descent

The error surface is given by, $E = \sum_{j=1}^{n\,set} E^p (V, W, I)$ and is shown in the Fig. 5.23.1

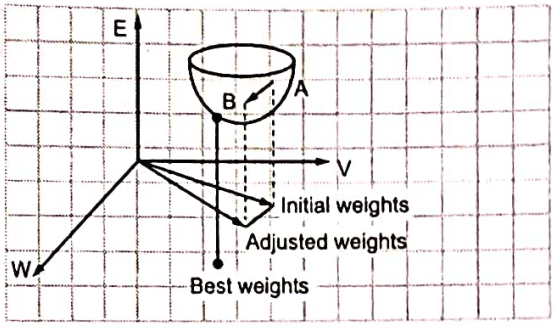



**Fig. 5.23.1 : Euclidean norm of errors**

(i) Multilayer feed forward networks with non-linear activation functions have mean squared error (MSE) surface, which is not in general, a smooth parabolic surface. In general, the error surface is complex and consists of many local and global minima,

(ii) In a single-layer linear activation case, the surface is smooth parabolic, as shown in Fig. 5.23.2.

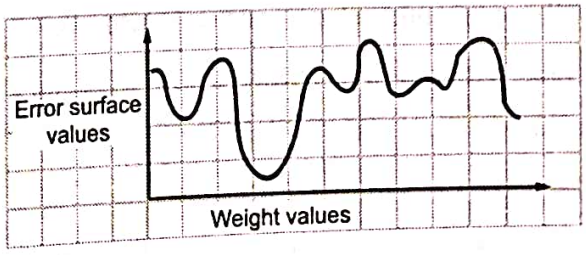



**Fig. 5.23.2 : Typical error surface with non-linear activation function**

### 5.23.2 Some Applications

Variety of problems have been successfully solved by Neural Networks. Pattern recognition and image processing Neural Networks are used in the recognition of visual images, handwritten characters, printed characters etc.

**1. Optimisation-constraint satisfaction**

Such type of problems have to satisfy constraints and to obtain optimal solutions. For examples, manufacturing scheduling, finding shortest possible tour given a set of countries, etc.

Forecasting Neural Networks can predict situations from past trends and can make risk assessment. Therefore in areas such as meteorology, stock market, banking they have ample applications.

**2. Control Systems**

Neural Networks have ample applications in control systems. Computer products, chemical plants, robots are standing examples of neural networks.

## 5.24 SOLVED EXAMPLES

**Ex. 5.24.1 :** Implement AND function using perceptron network for bipolar inputs and targets.

**Soln. :**

**Step (I) :**

1. We prepare the truth table for 'AND' function with bipolar inputs and targets :
2. We set the initial weights and threshold to zero.

   i.e, $w_1 = w_2 = 0$ ; $b = 0$ ; $\theta = 0$

   and the learning rate '$\alpha$' equal to 1

**Step (II) :**

1. We prepare the truth table for 'AND' function with bipolar inputs and targets :
2. We set the initial weights and threshold to zero.

   i.e, $w_1 = w_2 = 0$ ; $b = 0$ ; $\theta = 0$

   and the learning rate '$\alpha$' equal to 1.

**Table Ex. 5.24.1**

| $x_1$ | $x_2$ | t |
|---|---|---|
| 1 | 1 | 1 |
| 1 | – 1 | – 1 |
| – 1 | 1 | – 1 |
| – 1 | – 1 | – 1 |

3. We draw network architecture :

   Refer Fig. Ex. 5.24.1

**Step (III) :** We present the input patterns (from truth-table to the network one by one :

**Case (1) :** We have $x_1 = 1$, $x_2 = 1$, $t = 1$ (from table) and $w_1 = w_2 = \theta = 0$ ; $\alpha = 1$.

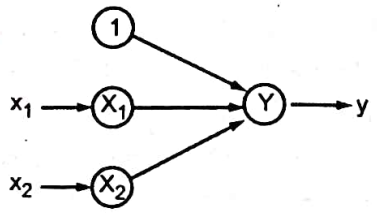



**Fig. Ex. 5.24.1**

(i) We calculate the net input :

$$y_{in} = b + x_1 w_1 + x_2 w_2$$

$$= 0 + 1 \times 0 + 1 \times 0 = 0 \quad \text{...(a)}$$

(ii) We apply bipolar activation formula to calculate output y :

The formula : $$y = f(y_{in}) = \begin{cases} 1, & \text{if } y_{in} > 0 \\ 0, & \text{if } y_{in} = 0 \\ -1, & \text{if } y_{in} < 0 \end{cases}$$

$\because$ From (a), $y_{in} = 0$ ; $\therefore$ $y = f(y_{in}) = 0$

(iii) Now, we check whether $t = y$

$\because$ $t = 1$ and $y = 0$, $\therefore$ $t \neq y$

$\therefore$ We carry out weight updation :

**Formula**

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

$\therefore$ for $i = 1, 2$

$$w_1 \text{ (new)} = w_1 \text{ (old)} + \alpha \cdot t \cdot x_1$$
$$= 0 + 1 \times 1 \times 1 = 1$$

and $w_2 \text{ (new)} = w_2 \text{ (old)} + \alpha \cdot t \cdot x_2 = 0 + (1) \cdot (1) \cdot (1) = 1$

and $b \text{ (new)} = b \text{ (old)} + \alpha \cdot t = 0 + (1)(1) = 1$

$\therefore$ The new weights are, $w_1 = 1, w_2 = 1, b = 1$, after first input pattern.

- **Step (IV) :** We repeat the same process for second input pattern.

Here, $x_1 = 1, x_2 = -1, t = -1$; and

$w_1 = 1, w_2 = 1, b = 1$.

(i) The net input :

$y_{in} = b + x_1 w_1 + x_2 w_2 = 1 + (1)(1) + (-1)(1) = 1$

(ii) Using bipolar activation formula

$$y_{in} = 1 > 0 \; ; \qquad \therefore \; y = f(y_{in}) = 1$$

(iii) We check whether $t = y$;

$\because t = -1$ and $y = 1$ ; $\therefore t \neq y$

$\therefore$ We again carry out weight updation :

Now, formula : $w_i \text{ (new)} = w_i \text{ (old)} + \alpha \cdot t \cdot x_i$ for $i = 1, 2$,

$w_1\text{(new)} = w_1 \text{ (old)} + \alpha \cdot t \cdot x_1 = 1 + (1)(-1)(-1) = \mathbf{0}$

$w_2\text{(new)} = w_2 \text{ (old)} + \alpha \cdot t \cdot x_2 = 1 + (1)(-1)(-1) = \mathbf{2}$

and $b\text{(new)} = b\text{(old)} + \alpha \cdot t = 1 + (1)(-1) = \mathbf{0}$

$\therefore$ The new weights are : $w_1 = 0, \; w_2 = 2, \; b = 0$

After second input pattern.

- **Step (V) :** We repeat the same for third input pattern :

Here $x_1 = -1, x_2 = 1, t = -1$ (from table)

and $w_1 = 0, \; w_2 = 2, \; b = 0$

(i) The net input :

$y_{in} = b + x_1 w_1 + x_2 w_2 = 0 + (-1)(0) + 1(2) = \mathbf{2}$

$$\therefore \; y_{in} = 2 > 0$$

(ii) We apply bipolar activation formula : $y = f(y_{in}) = 1$ $(\because y_{in} > 0)$

(iii) Since, $t = -1$, $\therefore y \neq t$

$\therefore$ We carry out weight updation : Now, $w_i$ (new)

$= w_i \text{ (old)} + \alpha \cdot t \cdot x_1$

for $i = 1, 2$

$$w_1\text{(new)} = w_1 \text{ (old)} + \alpha \cdot t \cdot x_1$$
$$= (0) + (1)(-1)(-1) = 1$$
$$w_2 \text{ (new)} = w_2 \text{ (old)} + \alpha t \cdot x_2$$
$$= (2) + (1)(-1)(1) = 1$$

and $b\text{(new)} = b \text{ (old)} + \alpha \cdot t$

$= 0 + (1)(-1) = -1$

$\therefore$ The new weights are, $w_1 = 1, w_2 = 1, b = -1$, after third input pattern.

- **Step (VI) :** We repeat the same process for fourth pattern.

$x_1 = -1, x_2 = -1, \; t = -1$ and $w_1 = 1, \; w_2 = 1, \; b = -1$,

(i) The net input :

$$y_{in} = b + x_1 w_1 + x_2 w_2$$
$$= -1 + (-1)(1) + (-1)(1) = -3$$
$$\therefore \; y_{in} = -3 < 0$$

(ii) By bipolar activation formula,

$$y = f(y_{in}) = -1 \quad (\because y_{in} < 0)$$

(iii) Since $t = -1$ and $y = -1$ ; $\therefore$ **t = y**

Now the process stops, since all the targets become equal to the calculated output.

$\therefore$ The final weights and bias after the last pattern

$w_1 = 1, w_2 = 1, b = -1$.

- **Step (VII) :** Separating the positive responses from negative responses, we can draw a separating line using the final weights.

Since the threshold for this problem is zero, the equation the separating line is

$$b + x_1 w_1 + x_2 w_2 = 0$$

$$\therefore \; x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}$$

Using final weights,

$$\therefore \; x_2 = -\frac{1}{1} x_1 - \frac{(-1)}{1} \; ;$$

$\therefore \quad x_2 = -x_1 + 1 \quad$ Or $\quad x_1 + x_2 = 1$

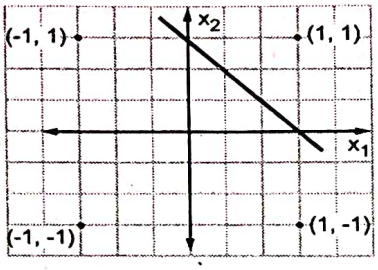



**Fig. Ex. 5.24.1(a)**

This line separates the positive response and negative response as shown in the Fig. Ex. 5.24.1(a). Decision boundary for AND function in perceptron training ($\theta = 0$).

**Ex. 5.24.2 :** Implement OR function with binary inputs and bipolar targets using Perceptron training algorithm up to 3 epochs.

> **Note :** When all the four input patterns are presented, then one Epoch is said to be completed.

**Soln. :**

**Step (I) :**

1. We prepare the truth-table for 'OR' function with binary inputs and bipolar targets.
2. We set the initial weights and bias to zero, i.e., $w_1 = w_2 = b = 0$.

**Learning rate $\alpha$ to 1 and threshold $\theta = 0.2$.**

(3) We exhibit the network architecture :

Refer Fig. Ex. 5.24.2.

**Truth-table**

| $x_1$ | $x_2$ | t |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | – 1 |

**Step (II) :** We present the input pattern (from truth-table) to the network one by one (to complete first Epoch).

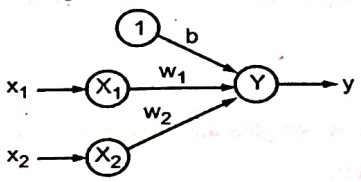



**Fig. Ex. 5.24.2**

**Case (1) :** We have $x_1 = 1$, $x_2 = 1$, $t = 1$; and $w_1 = w_2 = b = 0$ ; $\alpha = 1$ and $\theta = 0.2$

(i) We calculate the net input

$$y_{in} = b + x_1 w_1 + x_2 w_2 = 0 + (1)(0) + (1)(0) = 0 \quad \text{...(a)}$$

(ii) We apply binary activation formula to the net input to calculate output y :

The formula : $y = f(y_{in}) = \begin{cases} 1, & \text{if } y_{in} > 0.2 \\ 0, & \text{if } -0.2 \le y_{in} \le 0.2 \end{cases}$

($\because \theta = 0.2$).

$\because y_{in} = 0$ from Equation (a)

$\therefore y = 0$ and $t = 1$ ; $\quad \therefore y \neq t$

(iii) $\therefore y \neq t$; we carry out weight updation

**Formula**

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

for $i = 1, 2 \therefore \quad w_1 \text{ (new)} = w_1 \text{ (old)} + \alpha \cdot t \cdot x_1$

$$= 0 + (1)(1)(1) = \mathbf{1}$$

and $\quad w_2 \text{ (new)} = w_2 \text{ (old)} + \alpha \cdot t \cdot x_2 = 0 + (1)(1)(1) = \mathbf{1}$

and $\quad b \text{ (new)} = b \text{ (old)} + \alpha \cdot t = 0 + (1)(1) = \mathbf{1}$

$\therefore w_1 = 1, w_2 = 1, b = 1$ after first input pattern.

**Step (III) :** We repeat the same process for second input pattern.

Here $x_1 = 1$, $x_2 = 0$, $t = 1$ ; $\quad w_1 = 1$, $w_2 = 1$, $b = 1$.

(i) The net input

$$y_{in} = b + x_1 w_1 + x_2 w_2$$

$$= 1 + 1(1) + 0(1) = 2$$

$$\therefore y_{in} = 2 > 0.2 ;$$

$$\therefore y = f(y_{in}) = 1$$

[Since we are asked to carry 3 epoch we continue]

We again carry out weight updation :

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

$$\therefore w_1 \text{ (new)} = w_1 \text{ (old)} + \alpha \cdot t \cdot x_1$$

$$= 1 + (1)(1)(1) = 2$$

$$w_2 \text{ (new)} = w_2 \text{ (old)} + \alpha \cdot t \cdot x_2$$

$$\therefore w_2 \text{ (new)} = 1 + (1)(1)(0) = 1$$

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

$$= 1 + (1)(1)(1) = 2$$

The new weights are $w_1 = 2, w_2 = 1, b = 2$.

▶ **Step (IV) :** We have

$x_1 = 0, \quad x_2 = 1, \quad t = 1 ; \; w_1 = 2, \quad w_2 = 1, \quad b = 2 ;$

(i) The net input,

$$y_{in} = b + x_1 w_1 + x_2 w_2$$

$$= 2 + (0)(2) + 1(1) = 3$$

$$\because y_{in} = 3 > 0.2 ; \qquad \therefore y\, f(y_{in}) = 1$$

We carry out weight updation,

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

for $i = 1, 2$,

$$w_1 \text{ (new)} = w_1 \text{ (old)} + \alpha \cdot t \cdot x_1$$

$$= 2 + (1)(1)0 = 2$$

$$w_2 \text{ (new)} = w_2 \text{ (old)} + \alpha \cdot t \cdot x_2$$

$$= (1) + (1)(1)(1) = 2$$

$$b \text{ (new)} = b(\text{old}) + \alpha \cdot t = 2 + (1)(1) = 3.$$

$\therefore w_1 = 2, w_2 = 2, b = 3$ after third input pattern.

▶ **Step (V) :** We repeat the same process for fourth input pattern :

Here $x_1 = 0, x_2 = 0, t = -1; \; w_1 = 2, w_2 = 2, b = 3$

(i) The net input, $y_{in} = b + x_1 w_1 + x_2 w_2$

$$= 3 + 0 + 0 = 3$$

$$\therefore y_{in} = 3 > 0.2 ; \qquad \therefore f(y_{in}) = y = 1$$

(Since $y \neq t$); we carry out weight updation.

☞ **Formula**

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

for $i = 1, 2$ $\quad w_1 \text{ (new)} = w_1 \text{ (old)} + \alpha \cdot t \cdot x_1$

$$= 2 + (1)(-1)(0) = 2$$

$$w_2(\text{new}) = w_2 \text{ (old)} + \alpha \cdot t \cdot x_2$$

$$= 2 + (1)(1)(0) = 2$$

$$b(\text{new}) = b(\text{old}) + \alpha \cdot t = 3 + (1)(-1)$$

$$= 2$$

$\therefore$ The new weights are $w_1 = 2, w_2 = 2, b = 2$. With this process, first epoch is completed.

▶ **Step (VI) :** Epoch second is to be completed in the same fashion. (Exercise!)

and we get the final weights at the end of second epoch are :

$w_1 = 1, w_2 = 1, b = -1.$

▶ **Step (VII) :** At the end of third epoch, the final weights are, $w_1 = 2, w_2 = 1, b = -1$

---

**Ex. 5.24.3 :** Find the weights using perceptron network for ANDNOT function when all the inputs are presented only one time. Use bipolar input and targets.

☑ **Soln. :**

▶ **Step (I) :**

(1) We prepare the truth-table for 'ANDNOT' function with bipolar input and targets :

(2) We set the initial weights and threshold to zero ; i.e., $w_1 = w_2 = b = 0$, and let threshold $\theta = 0$ and learning rate $\alpha = 1$.

**Table Ex. 5.24.3**

| $x_1$ | $x_2$ | T |
|---|---|---|
| 1 | 1 | – 1 |
| 1 | – 1 | 1 |
| – 1 | 1 | – 1 |
| – 1 | – 1 | – 1 |

(3) We draw network architecture : Refer Fig. Ex. 5.24.3

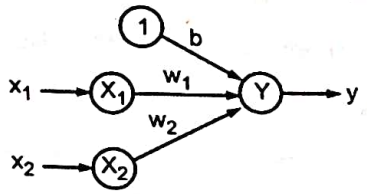



**Fig. Ex. 5.24.3 : Network for ANDNOT function**

▶ **Step (II) :** We present the input patterns to the network one by one.

▶ **Case (1) :** We have $x_1 = 1, x_2 = 1, t = -1$

$w_1 = 0, w_2 = 0, b = 0, \alpha = 1$

(i) To calculate the net-input : $y_{in} = b + x_1 w_1 + x_2 w_2 = 0 + 0 + 0 = 0$

(ii) We use bipolar activation formula to calculate output y :

The formula : $y = f(y_{in}) = \begin{cases} 1, & \text{if } y_{in} > 0 \\ 0, & \text{if } y_{in} = 0 \\ -1, & \text{if } y_{in} < 0 \end{cases}$

(iii) $\therefore\ y_{in} = 0,\ \therefore\ f(y_{in}) = y = 0$ but $t = -1$ ; $\therefore\ y \neq t$

$\therefore$ We carry out weight updation :

☞ **Formula**

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

for i = 1, 2 ;

$w_1(\text{new}) = w_1\,(\text{old}) + \alpha \cdot t \cdot x_1 = 0 + (1)(-1)(1) = -1$

$$w_2\,(\text{new}) = w_2\,(\text{old}) + \alpha \cdot t \cdot x_2$$
$$= 0 + (1)(-1)(1) = -1$$
$$\text{and } b(\text{new}) = b\,(\text{old}) + \alpha \cdot t$$
$$= 0 + (1)(-1) = -1$$

$\therefore$ The weights after first input pattern are,

$w_1 = -1, w_2 = -1, b = -1$

▶ **Step (III) :** For second input pattern

Here, $x_1 = 1, x_2 = -1, t = 1$ (from table)

$w_1 = -1, w_2 = -1, b = -1$

(i) The net input :

$$y_{in} = b + x_1 w_1 + x_2 w_2$$
$$= -1 + (1)(-1) + (-1)(-1) = -1$$
$$\therefore\ y_{in} = -1 < 0\ ;$$

$\therefore\ y = f(y_{in}) = -1$ ; $\therefore\ t = 1,\ \therefore\ t \neq y$

$\therefore$ We again carry out weight updation

☞ **Formula**

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

for i = 1, 2, $w_1(\text{new}) = w_1(\text{old}) + \alpha \cdot t \cdot x_1$

$$\therefore\ w_1\,(\text{new}) = -1 + (1)(1)(1) = 0$$
$$w_2\,(\text{new}) = w_2\,(\text{old}) + \alpha \cdot t \cdot x_2$$
$$= (-1) + (1)(1)(-1) = -2$$
$$b(\text{new}) = b(\text{old}) + \alpha \cdot t$$
$$= (-1) + (1)(1) = 0$$

$\therefore$ The new weights are $w_1 = 0, w_2 = -2, b = 0$ after second input pattern

▶ **Step (IV) :** For third input pattern :

Here : $x_1 = -1, x_2 = 1, t = -1$ (from table).

$w_1 = 0, w_2 = -2, b = 0$

For the fourth input pattern,

$x_1 = -1,\ x_2 = -1,\ t = -1$, (from table)

$w_1 = 0, w_2 = -2, b = 0$

$\therefore$ (i) The net input

$$y_{in} = b + x_1 w_1 + x_2 w_2$$
$$= 0 + (-1)(0) + (-1)(-2) = 2$$

Again $\therefore\ y_{in} = 2$ ; $\therefore\ y = f(y_{in}) = 1$ ;

$\therefore\ y = 1 \neq t = -1$

▶ **Step (V) :** We update the weights

$$w_1\,(\text{new}) = w_1\,(\text{old}) + \alpha \cdot t \cdot x_1$$
$$= 0 + 1(-1)(-1) = 1$$
$$w_2\,(\text{new}) = w_2\,(\text{old}) + \alpha \cdot t \cdot x_2$$
$$= (-2) + 1 \cdot (-1)(-1) = -1$$

$$b\,(new) = b\,(old) + \alpha \cdot t$$
$$= 0 + 1 \times (-1) = -1$$

$\therefore$ The weights after fourth input process :

$w_1 = 1, \quad w_2 = -1, \quad b = -1.$

**Ex. 5.24.4 :** Find the weights required to perform the following classification using perceptron network. The vectors (1, 1, 1, 1) and (– 1, 1, – 1, – 1) are belonging to the class (so have target value 1), vectors (1, 1, 1, – 1) and (1, – 1, – 1, 1) are not belonging to the class (so have target valve – 1). Assume learning rate as 1 and initial weights as 0.

**Soln. :**

**Step (I) :**

1. We prepare Input–table for the 4 vectors with 4 components each. Note that first two vectors belong to target value 1 and the last two vectors belong to target valve – 1.
2. We set the initial weights as 0.

   i.e. $w_1 = w_2 = w_3 = w_4 = 0, b = 0;$

**Let the learning rate $\alpha = 1$ and threshold $\theta = 0.2$**

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | b | t |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 |
| – 1 | 1 | – 1 | – 1 | 1 | 1 |
| 1 | 1 | +1 | – 1 | 1 | – 1 |
| 1 | – 1 | – 1 | 1 | 1 | – 1 |

**Step (II) :** We present the input patterns (from table) to the network one by one.

**Case (1) :** We have $x_1 = 1, x_2 = 1, x_3 = 1, x_4 = 1, t = 1$ $\alpha = 1$.

$w_1 = w_2 = w_3 = w_4 = 0, b = 0.$

(i) We calculate the net input :

$$y_{in} = b + x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4$$
$$= 0 + (1)(0) + (1)(0) + 1(0) + 1(0) = 0$$

(ii) We mention the activation formula :

Since $\theta = 0.2$

$$y = \begin{cases} 1 & \text{if } y_{in} > 0.2 \\ 0 & \text{if } -0.2 \le y_{in} \le 0.2 \\ -1 & \text{if } y_{in} < -0.2 \end{cases}$$

$\therefore y_{in} = 0, \quad \therefore y = f(y_n) = 0$

(iii) Here $t \ne y$ ; $\therefore$ we carry out weight updation :

**Formula**

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

For i = 1, 2, 3, 4

$w_1\,(new) = w_1\,(old) + \alpha\, t\, x_1 = 0 + (1)(1)(1) = 1$

$w_2\,(new) = w_2\,(old) + \alpha\, t\, x_2 = 0 + (1)(1)(1) = 1$

$w_3\,(new) = w_3\,(old) + \alpha\, t\, x_3 = 0 + (1)(1)(1) = 1$

$w_4\,(new) = w_4\,(old) + \alpha\, t\, x_4 = 0 + (1)(1)(1) = 1$

and $b\,(new) = b\,(old) + \alpha \cdot t = 0 + 1 \cdot 1 = 1.$

$\therefore$ New weights are $w_1 = 1, w_2 = 1, w_3 = 1, w_4 = 1, b = 1.$

**Step (III) :** For second input pattern :

Here $x_1 = -1, x_2 = 1, x_3 = -1, x_4 = -1, b = 1, \; w_1 = w_2 = w_3 = w_4 = 1.$

(i) The net input :

$$y_{in} = b + x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4$$
$$= 1 + (-1)(1) + (1)(1) + (-1)(1) + (-1)(1) = -1$$

$\therefore y_{in} = -1 < -0.2$

$\therefore y = f(y_{in}) = -1$

$\because t = 1$ and $y = -1$

$\therefore t \ne y$ ; $\therefore$ We again carry out weight updation :

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

i = 1, 2, 3, 4,

$\therefore w_1\,(new) = w_1\,(old) + \alpha \cdot t \cdot x_1 = 1 + (1)(1)(-1) = 0$

$w_2\,(new) = w_2\,(old) + \alpha \cdot t \cdot x_2 = 1 + (1)(1)(1) = 2$

$w_3\,(new) = w_3\,(old) + \alpha \cdot t \cdot x_3 = 1 + (1)\,(1)\,(-1) = 0$

$w_4\,(new) = w_4\,(old) + \alpha \cdot t \cdot x_4 = 1 + (1)\,(1)\,(-1) = 0$

$b\,(new) = b\,(old) + \alpha \cdot t = 1 + (1)(1) = 2$

$\therefore$ New weights are $w_1 = 0,\ w_2 = 2,\ w_3 = 0,\ w_4 = 0,\ b = 2.$

- **Step (IV) :** For third input pattern :

Here $x_1 = 1, x_2 = 1, x_3 = 1, x_4 = -1$

(1) The net input

$$y_{in} = b + x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4$$

$$= 2 + (1)(0) + 1(2) + (1)(0) + (-1)(0) = 4$$

$\therefore\ y_{in} = 4 > 0.2, \quad \therefore\ y = f(y_{in}) = 1$

Since, $t = -1,\ y = 1\ \ \therefore\ y \neq t$

$\therefore$ We further carry weight updation :

Here, $x_1 = 1,\ x_2 = -1,\ x_3 = -1,\ x_4 = 1,$

$w_1 = -1,\ w_2 = 1,\ w_3 = -1,\ w_4 = 1,\ b = 1$ (exercise!)

(i) The net fourth input

$$y_{in} = b + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$

$$\therefore\ y_{in} = 1 + (-1)(1) + 1(-1) + (-1)(-1) + (1)(1)$$

$$= 1 - 1 - 1 + 1 + 1 = 1$$

$\therefore\ y_{in} = 1 > 0.2$

$\therefore\ y = f(y_{in}) = 1$

but target value t is $-1$. $\therefore\ y \neq t$

- **Step (V) :** With this epoch I is completed

We carry out epoch II (exercise) and get the results :

$w_1 = -2,\ w_2 = 2,\ w_3 = 0,\ w_4 = 2,\ b = 0$

and $y_{in} = -2 < -0.2$

$\therefore\ y = f(y_{in}) = -1$ and $t = -1$

$\therefore\ y = t$

$\therefore$ Network converges :

- **Step (VI) :** We draw network architecture

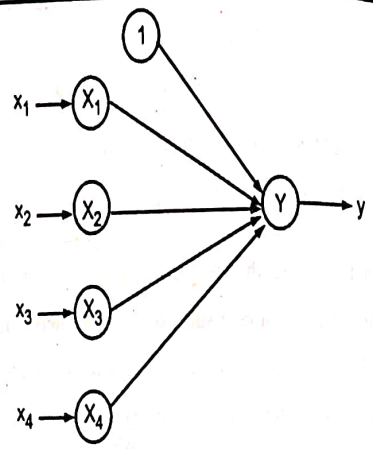



**Fig. Ex. 5.24.4 : Network architecture**

**Ex. 5.24.5 :** Implement OR function with bipolar inputs and targets using Adaline network.

☑ **Soln. :**

☞ **Method**

- **Step (I) :**

(i) First we prepare truth-table for OR function with bipolar inputs and targets, and draw network

(ii) Choose initial weights and learning rate.

(iii) Calculate $y_{in}$, by standard formula.

- **Step (II) :** Compute $(t - y_{in})$, and update the weights using the formula :

$$w_i\,(new) = w_i\,(old) + \alpha\,(t - y_{in})x_i \qquad \text{for } i = 1, 2, \ldots$$

- **Step (III) :**

Now, calculate error

$$E = (t - y_{in})^2$$

and write down the final weights after first input sample.

- **Step (IV) :** Repeat the procedure for all the input samples and calculate the errors.

Sum up the errors of each input sample during one epoch. That is the total mean square error of that epoch.

- **Step (V) :** We prepare truth-table for OR function with bipolar inputs and targets :

Let the weight be,

$w_1 = w_2 = b = 0.1$

and learning rate $\alpha = 0.1$

**Truth-table**

| $x_1$ | $x_2$ | t |
|---|---|---|
| 1 | 1 | 1 |
| 1 | – 1 | 1 |
| – 1 | 1 | 1 |
| – 1 | – 1 | – 1 |

- **Case (1) :** Let $x_1 = 1, x_2 = 1, t = 1, w_1 = w_2 = b = 0.1$, and $\alpha = 0.1$

(i) The net input is :

$$y_{in} = b + x_1 w_1 + x_2 w_2$$
$$= 0.1 + 1 \times (0.1) + 1(0.1) = 0.3$$

Now, we compute $(t - y_{in}) = (1 - 0.3) = 0.7$

(ii) We update the weights :

☞ **Formula**

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

for i = 1, 2, $w_1$(new) $= w_1$ (old) $+ \alpha (t - y_{in}) x_1$

$$= 0.1 + 0.1 (0.7) (1)$$
$$= 0.1 + 0.07 = 0.17$$

and $w_2$ (new) $= w_2$ (old) $+ \alpha (t - y_{in}) x_2$

$$= 0.1 + 0.1 (0.7)(0.1) = 0.17$$

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

$$= 0.1 + (0.7) = 0.17$$

- **Step (VI) :** To calculate error :

$$E = (t - y_{in})^2 = (0.7)^2 = 0.49$$

The final weights : $w_1 = 0.17$, $w_2 = 0.17$, b= 0.17, and E = 0.49, is error,

Now, formula : we update weights :

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

for i = 1, 2, $w_1$ (new) $= w_1$ (old) $+ 0.1 (1 - 0.3)(1)$

$$= 0.17 + 0.1 (0.7) = 0.253$$

$$w_2 \text{ (new)} = w_2 \text{ (old)} + \alpha (t - y_{in}) x_2$$
$$= (0.17) + 0.1 (0.7)(-1) = 0.087$$

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

$$= 0.17 + 0.1 (0.7) = 0.253$$

Now, we calculate $y_{in}$ :

$x_1 = 1, x_2 = -1, w_1 = 0.253,$

$w_2 = 0.087, b = 0.253$

$$y_{in} = b + x_1 w_1 + x_2 w_2$$
$$= 0.253 + (0.253) - (1)(0.087)$$
$$= 0.17$$

$\therefore$ Error $= (t - y_{in})^2 = (1 - 0.17)^2 = \mathbf{0.69}$

- **Step (VII) :** Let $x_1 = -1, x_2 = 1, t = 1,$

$w_1 = 0.253, w_2 = 0.087$ b = 0.253

Now, $y_{in} = b + x_1 w_1 + x_2 w_2$

$$= 0.253 + (-1) (0.253) + (1)(0.087)$$
$$= 0.253 - 0.253 + 0.087 = 0.087$$

$\therefore t - y_{in} = (1 - 0.087) = 0.913$

$\therefore$ Error $= (t - y_{in})^2 = \mathbf{0.83}$

- **Step (VIII) :** Let $x_1 = -1, x_2 = -1, t = -1$ and

$w_1 = 0.253, w_2 = 0.087$, b= 0.253

and $y_{in} = 0.087, \alpha = 0.1$

☞ **Weight updation**

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

for i = 1, 2, $w_1$ (new) $= w_1$ (old) $+ \alpha (t - y_{in}) x_1$

$\therefore$ $w_1$ (new) $= 0.253 + 0.1(0.913)(-1)$

$$= 0.1617$$

$w_2$ (new) $= w_2$ (old) $+ \alpha (t - y_{in}) x_2$

$\therefore$ $w_2$ (new) $= 0.087 + 0.1(0.913)(-1)$

$$= 0.1783$$

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

$$= 0.253 + 0.1 (0.913) = 0.3443$$

▶ **Step (IX) :** Now, we calculate $y_{in}$

$$y_{in} = b + x_1 w_1 + x_2 w_2 = 0.3443 + (-1)(0.1617) + (-1)(0.1783)$$

$$= +0.0043$$

$$\therefore t - y_{in} = -1 - (+0.043) = -1.0043$$

$$\therefore \text{Error} = (t - y_{in})^2 = (-1.0043)^2 = 1.01$$

With this total error

$$= (0.49) + (0.69) + (0.83) + (1.01) = 3.02$$

With this first epoch is completed.

> **Note :** Our aim is to minimize the error to large extend. Hence we have to continue the process for at least two more epochs. (Exercise)!

**Ex. 5.24.6 :** Use Adaline network to train AND NOT function with bipolar inputs and targets.

☑ **Soln. :**

▶ **Step (I) :** We prepare truth-table for AND NOT function with bipolar inputs and targets :

For convenience, we choose random values say 0.2 for weights and bias.

Let $w_1 = w_2 = b = 0.2$ and let learning rate $\alpha = 0.2$

▶ **Step (II) :**

**Case (1) :** Let $x_1 = 1, x_2 = 1, t = -1$ ;
$w_1 = w_2 = b = 0.2, \alpha = 0.2$

**Truth-table**

| $x_1$ | $x_2$ | t |
|---|---|---|
| 1 | 1 | – 1 |
| 1 | – 1 | 1 |
| – 1 | 1 | – 1 |
| – 1 | – 1 | – 1 |

(i) The net input is,

$$y_{in} = b + x_1 w_1 + x_2 w_2$$

$$= 0.2 + 1(0.2) + 1(0.2) = 0.6$$

$$\therefore t - y_{in} = -1 - 0.6 = -1.6$$

(ii) We update the weights,

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

for i = 1, 2 $\quad w_1 \text{ (new)} = w_1 \text{ (old)} + \alpha (t - y_{in}) x_1$

$$= 0.2 + 0.2(-1.6)(1) = -0.12 ;$$

Also, $w_2 \text{ (new)} = w_2 \text{ (old)} + \alpha (t - y_{in})$

$$x_2 = 0.2 + 0.2(-1.6)(1) = -0.12$$

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

$$= 0.2 + 0.2(-1.6) = -0.12$$

$$\therefore (t - y_{in}) = -1.6$$

$$\therefore \textbf{Error} = (t - y_{in})^2 = (-1.6)^2 = \mathbf{2.56}$$

▶ **Step (III) :** Now, we have

$w_1 = -0.12, \; w_2 = -0.12, \; b = -0.12, \; x_1 = 1,$
$x_2 = -1, t = 1$

$$\therefore \text{input } y_{in} = b + x_1 w_1 + x_2 w_2$$

$$= (-0.12) + (1)(-0.12) + (-1)(-0.12)$$

$$= -0.12$$

$$\therefore t - y_{in} = (1) - (-0.12) = 1.12$$

$$\therefore \text{Error} = (t - y_{in})^2 = (1.12)^2 = \mathbf{1.25}$$

▶ **Step (IV) :** We update weights

We have, $x_1 = 1, x_2 = -1, \; t = 1, \; y_{in} = -0.12, b = -0.12$

Now, $w_i(\text{new}) = w_i \text{ (old)} + \alpha (t - y_{in}) x_i$

$$w_1 \text{ (new)} = w_1(\text{old}) + \alpha (t - y_{in}) x_1$$

$$= -0.12 + 0.2(1 + 0.12)(1)$$

$$= 0.10$$

$$w_2 \text{ (new)} = w_2 \text{ (old)} + \alpha (t - y_{in}) x_2$$

$$= (-0.12) + 0.2(-1.12)(-1)$$

$$= -0.34$$

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

$$= -0.12 + 0.2(-1.12) = 0.10$$

**Step (V) :** Now, $x_1 = -1$, $x_2 = 1$, $t = -1$, $w_1 = 0.10$, $w_2 = -0.34$, $b = 0.10$

We calculate total input $y_{in}$

$$y_{in} = b + x_1 w_1 + x_2 w_2$$

$$= 0.10 + (-1)(0.1) + (1)(-0.34)$$

$$\therefore \quad y_{in} = -0.34 \;;$$

$$\therefore \quad t - y_{in} = -1 - (-0.34) = -0.66$$

$$\therefore \quad (t - y_{in})^2 = (-0.66)^2 = 0.43 \;;$$

$$\therefore \quad \textbf{Error} = (t - y_{in})^2 = \mathbf{0.43}$$

**Step (VI) :** Now, $x_1 = -1$, $x_2 = +1$, $t = -1$, $b = 0.10$

$$y_{in} = -0.34 \;; \quad w_1 = 0.10,$$

$$w_2 = -0.34$$

We carry weight up-dation

$$w_1 \text{ (new)} = w_1 \text{ (old)} + \alpha (t - y_{in}) x_1$$

$$= 0.10 + 0.2 (-0.66)(-1)$$

$$= 0.10 + 0.132 = 0.24$$

$$w_2 \text{ (new)} = w_2 \text{ (old)} + \alpha (t - y_{in}) x_2 = -0.34 + 0.2(-0.66)(1) = -0.48$$

$$b \text{ (new)} = b \text{ (old)} + \alpha (t - y_{in}) = 0.10 + 0.2(-0.66) = -0.03$$

$$\therefore \quad (t - y_{in}) = -0.66$$

$$\therefore \quad \textbf{Error} = (t - y_{in})^2 = (-0.66)^2 = \mathbf{0.43}$$

**Step (VII) :** Let $x_1 = -1$, $x_2 = -1$, $t = -1$, $w_1 = 0.24$, $w_2 = -0.48$, $b = -0.03$

We calculate net input $y_{in}$

$$y_{in} = b + x_1 w_1 + x_2 w_2$$

$$\therefore \quad y_{in} = -0.03 + (-1)(0.24) + (-1)(-0.48)$$

$$= -0.03 - 0.24 + 0.48$$

$$= -0.03 + 0.24 = 0.21$$

$$\therefore \quad y_{in} = 0.21$$

$$\therefore \quad t - y_{in} = -1 - (0.21) = -1.21$$

$$\therefore \quad \textbf{Error} = (t - y_{in})^2 = (-1.21)^2 = \mathbf{1.47}$$

**Step (VIII) :** Weight updation

$x_1 = -1$, $x_2 = -1$, $t = -1$, $y_{in} = 0.21$

$w_1 = 0.24$, $w_2 = -0.48$, $b = -0.03$

Now, $w_i \text{ (new)} = w_i \text{ (old)} + \alpha (t - y_{in}) x_i$

for $i = 1, 2$,

$$w_1 \text{ (new)} = w_1(\text{old}) + \alpha (t - y_{in}) x_1$$

$$= -0.24 + 0.2(-1.21)(-1) = 0.48$$

$$w_2 \text{ (new)} = w_2(\text{old}) + \alpha (t - y_{in}) x_2$$

$$= -0.48 + 0.2(-1.21)(-1) = -0.23$$

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

$$= -0.03 + 0.2(-1.21) = -0.27$$

$\therefore$ New weights are : $w_1 = 0.48$, $w_2 = -0.23$, $b = -0.27$

Total error $= 2.56 + 1.25 + 0.43 + 1.47 = 5.71$

With this epoch I is over.

☞ **Exercise :** To minimise the error, complete the epoch II.

## 5.25 MEDALINE-RULE ALGORITHM (MRA)

When there are hidden layers in the neural network, we use Madaline-rule Algorithm to implement the given function.

☞ **Method**

1. Prepare truth-table for the given function.
2. If the parameters are not mentioned, then consider initial random values too small.
3. Find the **net input** to **hidden units** using the standard formula,
4. Calculate the output of the hidden units using activation function (as mentioned), over the net input computed.
5. Then find the net input entering into the output unit.

6. Then evaluate the net input $y_{in}$ to calculate the output y :

7. If $t \neq y$, then we carry weight updation.

**Note :** The weights and bias between the input layer and hidden layer are to be adjusted.

**Ex. 5.25.1 :** Using medaline network, implement XOR function with bipolar inputs and targets. Assume the required parameters for training of the network.

**Soln. :**

**Step (I) :** The table for 'XOR' function is :

**Table Ex. 5.25.1**

| $x_1$ | $x_2$ | T |
|---|---|---|
| 1 | 1 | – 1 |
| 1 | – 1 | 1 |
| – 1 | 1 | 1 |
| – 1 | – 1 | – 1 |

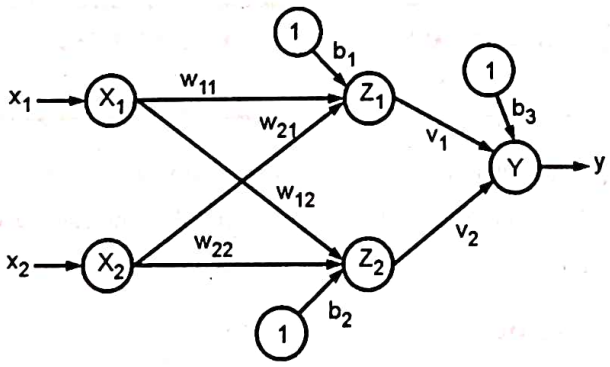



**Fig. Ex. 5.25.1**

Network architecture for Madeline

For 'XOR' functions. There are two hidden layers :

**Initial weights and bias :** (We choose small random values)

(a) $w_{11} = 0.05$, $w_{12} = 0.1$, $w_{21} = 0.2$, $w_{22} = 0.2$

$b_1 = 0.3$, $b_2 = 0.15$, and learning rate $\alpha = 0.5$

(b) $v_1 = 0.5$, $v_2 = 0.5$, $b_3 = 0.5$

**Step (II) :**

(i) To calculate net input to hidden unit :

$x_1 = 1,\ x_2 = 1,\ t = -1$

$$Z_{in1} = b_1 + x_1 w_{11} + x_2 w_{21}$$
$$= 0.3 + (1)(0.05) + 1(0.2) = 0.55$$
$$Z_{in2} = b_2 + x_2 w_{12} + x_2 w_{22}$$
$$= 0.15 + (1)(0.1) + (1)(0.2) = 0.45$$

(ii) We calculate output of $Z_1$, $Z_2$ by applying activation over the net input :

The bipolar activation function is given by,

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

Since $z_{in1} = 0.55 > 0$

$\therefore f(Z_{in1}) = 1$ and also, $Z_{in2} = 0.45 > 0$

$\therefore f(Z_{in2}) = 1$ (Using above formula)

(iii) We find net input entering into the output unit : (using the standard formula for $y_{in}$)

$$y_{in} = b_3 + Z_1 V_1 + Z_2 V_2$$
$$= 0.5 + 1 \times (0.5) + 1 \times (0.5) = 1.5$$

The output $y = f(y_{in}) = f(1.5) = 1$

($\because$ 1.5 > 0 above formula)

Since $t \neq y$ ($\because t = -1, y = 1$) ;

We update weights, (we update weights an $Z_1$ and $Z_2$)

(iv) $w_{ij}$ (new) = $w_{ij}$ (old) + $\alpha (t - Z_{inj}) x_i$

and $b_j$ (new) = $b_j$ (old) + $\alpha (t - Z_{inj}) x_1$

$\therefore w_{11}$ (new) = $w_{11}$ (old) + $\alpha (t - Z_{in1}) x_1$ = (0.05) + 0.5 (– 1 – 0.55) $x_1$ = – 0.725

$$w_{12} \text{ (new)} = w_{12} \text{ (old)} + \alpha (t - Z_{in2}) x_1$$
$$= 0.1 + 0.5 (-1 - 0.45)(1)$$
$$= -0.625$$
$$b_1 \text{ (new)} = b_1 \text{ (old)} + \alpha (t - Z_{in1})$$
$$= 0.3 + 0.5 (-1 - 0.55) = -0.475$$
$$w_{21} \text{ (new)} = w_{21} \text{ (old)} + \alpha (t - Z_{in1}) x_2$$

$= 0.2 + 0.5(-1 - 0.55)(1)$

$= -.575$

$w_{22}(\text{new}) = w_{22}(\text{old}) + \alpha(t - Z_{in2})x_2$

$= 0.2 + 0.5(-1 - 0.45)(1)$

$= -0.525$

$b_2(\text{new}) = b_2(\text{old}) + \alpha(t - Z_{in2})$

$= 0.15 + 0.5(-1 - 0.45)$

$= -0.575$

**Step (III) :** Now, we calculate net input to hidden units : $x_1 = 1, x_2 = -1, t = 1$.

(i) $Z_{in1} = b_1 + x_1 w_{11} + x_2 w_{21}$

$= (-0.475) + (1)(-0.725) + (-1)(-0.575)$

$= -0.625$

(ii) $Z_{in2} = b_2 + x_1 w_{12} + x_2 w_{22}$

$= (-0.575) + (1)(-0.625) + (-1)(-0.575)$

$= -0.675$

(iii) To apply activation formula for the net input to calculate output of $Z_1, Z_2$.

$\therefore Z_{in1} = -0.625 < 0$;

$\therefore Z_1 = f(Z_{in1}) = -1$

and also, $Z_{in2} = -0.675 < 0$;

$\therefore Z_2 = f(Z_{in2}) = -1$

(using the formulae)

(iv) Now, we find net input entering into the output using :

$y_{in} = b_3 + Z_1 V_1 + Z_2 V_2$

$= 0.5 + (-1)(0.5) + (-1)(0.5) = -0.5$

Now, the output $y = f(y_{in}) = f(-0.5) = -1$ $(\because -0.5 < 0)$

$\therefore t \neq y$ $(\because t = 1, y = -1)$

$\therefore$ We update weights on $Z_1$ and $Z_2$.

**Step (IV) :** We have

$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha(t - Z_{inj})x_i$

and $b_j(\text{new}) = b_j(\text{old}) + \alpha(t - Z_{inj})x_1$

$\therefore w_{11}(\text{new}) = w_{11}(\text{old}) + \alpha(t - Z_{in1})x_1$

$= -0.725 + 0.5(1 + 0.625)(1)$

$= 0.0875$

$\therefore w_{12}(\text{new}) = w_{12}(\text{old}) + \alpha(t - Z_{in2})x_2$

$b_1(\text{new}) = b_1(\text{old}) + \alpha(t - Z_{in1})$

$= -0.475 + 0.5(1 + 0.625) = 0.34$

$w_{21}(\text{new}) = w_{21}(\text{old}) + \alpha(t - Z_{in1})x_2$

$= (0.575) + 0.5(1 + 0.625)(-1)$

$= -1.39$

$w_{22}(\text{new}) = w_{22}(\text{old}) + \alpha(t - Z_{in2})x_2$

$= (-0.575) + 0.5(1 + 0.625)(-1) = -0.525$

$b_2(\text{new}) = b_2(\text{old}) + \alpha(t - Z_{in2})$

$= (-0.575) + 0.5(1 + 0.675)$

$= -0.575$

**Step (V) :** Now, to calculate net input to hidden units : $x_1 = -1, x_2 = 1, t = 1$;

(i) $Z_{in1} = b_1 + x_1 w_{11} + x_2 w_{21}$

$= 0.34 + (-1)(0.0875) + 1(-1.39) = -1.1375$

$Z_{in2} = b_2 + x_1 w_{12} + x_2 w_{22}$

$= -0.575 + (-1)(-0.625) + (1)(-0.525)$

$= -0.475$

(ii) To calculate output of $Z_1, Z_2$, we apply activation formula for the net input.

$\because Z_{in1} = -0.625 < 0$

$\therefore Z_1 = f(Z_{in1}) = -1$ (from the formula) and $Z_{in2} = -0.675 < 0$

$\therefore Z_2 = f(Z_{in2}) = -1$

(iii) The net input entering the output, we have

$y_{in} = b_3 + Z_1 V_1 + Z_2 V_2$

$= 0.5 + (-1)(0.5) + (-1)(0.5) = -0.5$

$\therefore$ The output $y = f(y_{in}) = f(-0.5) = -1$ $(\because -0.5 < 0)$

$\therefore t \neq y$ $(\because t = 1, y = -1)$

$\therefore$ We update weights on $Z_1, Z_2$.

▶ **Step (VI) :** Again,

$$W_{ij}\,(new) = w_{ij}\,(old) + \alpha\,(t - Z_{inj})\,x_i$$

and $$b_j\,(new) = b_j\,(old) + \alpha\,(t - Z_{inj})\,x_1$$

Here, $\therefore\ w_{11}\,(new) = w_{11}\,(old) + \alpha\,(t - Z_{in1})\,x_1 = 0.0875$

$$w_{12}\,(new) = w_{12}\,(old) + \alpha\,(t - Z_{in2})\,x_2 = -1.3625$$

$$b_1\,(new) = b_1\,(old) + \alpha\,(t - Z_{in1}) = 0.34$$

$$w_{21}\,(new) = w_{21}\,(old) + \alpha\,(t - Z_{in1})\,x_2 = -1.39$$

$$w_{22}\,(new) = w_{22}\,(old) + \alpha\,(t - Z_{in2})\,x_2$$

$$= (-0.525) + 0.5\,(1 + 0.475)(1) = 0.2125$$

$$b_2\,(new) = b_2\,(old) + \alpha\,(t - Z_{in2})$$

$$= (-0.575) + 0.5\,(1.475) = 0.1625$$

▶ **Step (VII) :** To calculate net input to hidden units : $x_1 = -1, x_2 = -1, t = -1$.

(i) $$Z_{in1} = b_1 + x_1\,w_{11} + x_2\,w_{21}$$

$$= 0.34 + (-1)\,(0.0875) + (-1)\,(-1.39)$$

$$= 1.6375$$

(ii) $$Z_{in2} = b_2 + x_1\,w_{12} + x_2\,w_{22}$$

$$= 0.1625 + (-1)\,(-1.3625) + (-1)(-0.2125)$$

$$= 1.3125$$

(iii) To calculate output of $Z_1$, $Z_2$; we apply activation formula

$$\because\ Z_{in1} = 1.6375 > 0$$

$$\therefore\ Z_1 = f(Z_{in1}) = 1 \quad \text{and}$$

$$\because\ Z_{in2} = 1.3125 > 0$$

$$\therefore\ Z_2 = f\,(Z_{in2}) = 1$$

(iv) The net input entering the output :

$$y_{in} = b_3 + Z_1\,V_1 + Z_2\,V_2$$

$$= 0.5 + (1)\,(0.5) + (1)\,(0.5) = 1.5$$

$$\therefore\ y = f(y_m) = y$$

$\because\ t = -1, y = 1 \quad \therefore\ t \neq 1$

Now, update the weights on $Z_1$, $Z_2$ (Exercise !)

## 5.26 LOGISTIC REGRESSION

**UQ.** Write short note on : Logistic Regression. **(Ref. MU (Comp.) - May 17, 10 Marks)**

**UQ.** Explain how regression problem can be solved using Steepest descent method. Write down the steps. **(Ref. MU (Comp.) - May 19, 5 Marks)**

- Suppose we have different training data which belongs to two different classes, and we have to design a system that will identify which data is from which class. The output of this function will be a real value and it is not suitable for classification method. We can use another function on this linear function so that we can use the result for classification. In logistic regression logistic function or the sigmoid can be used.
- Let's first see what is the meaning of Logistic or Sigmoid function.
- In Logistic regression classifier we will take features as the input. These features are multiplied with the logistic coefficients and we add the product. This is called as the net input that will be given to the sigmoid function. The output will be between 0 and 1. Anything above 0.5 is classified as 1and anything below 0.5 is classified as 0.
- The net input to the sigmoid function is,

$$Z = b_0 + b_1\,x_1 + b_2\,x_2 + \text{----------} + b_n\,x_n$$

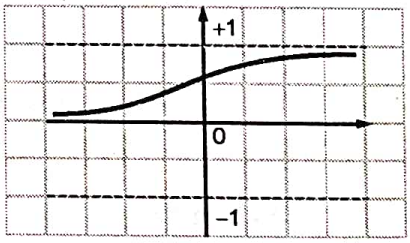



**Fig. 5.26.1 : Logistic or Sigmoid Function**

- In the net input equation x represents the input data or the features. When we use optimized coefficient b, Classifier will be successful. Optimized b can be calculated using the optimization concept.

### Gradient Ascent Method

- In gradient ascent method we move in the direction of the gradient to find the maximum point on a function.

$$\nabla f(x, y) = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix}$$

- In the above equation $\partial f(x, y)/\partial x$ represents the amount by which updation is applied in x direction and $\partial f(x, y)/\partial y$ represents the amount by which updation applied in y direction
- The gradient operator will always point towards the direction of gradient increase.

$$b = b + \alpha \times \nabla f(b)$$

- This step is repeated until we reach stopping criterion.

**Pseudo code**

Start using the logistic coefficient, all set to 0

Repeat no. of times

Find the gradient of complete dataset

Update, logistic coefficient = logistic coefficient + alpha × gradient

Return the logistic coefficient

**Gradient Descent Method**

- In gradient descent method we move in the opposite direction of the gradient to find the minimum point on a function
- The gradient operator will always point opposite to the direction of gradient increase.

$$b = b - \alpha \times \nabla f(b)$$

- This step is repeated until we reach stopping criterion.
- Using above mentioned methods optimized b is calculated, net input is calculated and then the prediction is calculated by giving the net input to the function,

$$\text{Prediction} = \frac{1}{1 + e^{-x}}$$

- Finally the data points are classified as,

$$\text{Class} = 1 \text{ if Prediction} \geq 0.5$$
$$= 0 \text{ else}$$

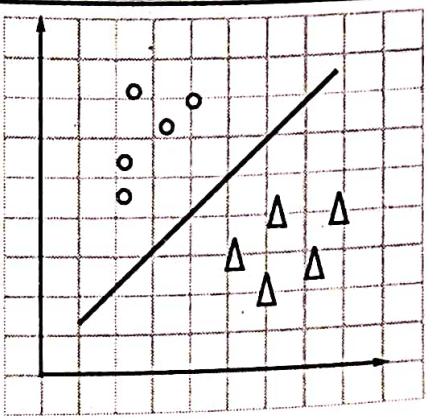


**Fig. 5.26.2 : Sample of output of Logistic regression**

- The classified data using the decision boundary is as shown in Fig. 5.26.2.
- Logistic regression is used to model a relationship between input variables and a target variable. Let's take an example of a house management system , we can use logistic regression to model the relationship between the parameters such as the total monthly income of the family, various liabilities, monthly expenditure of a family to predict if monthly savings (investment) can be done or not.
- There are three types of logistic regression as below,

**Binary Logistic Regression**

- Binary logistic regression is used if the target variable is binary.
- The application of this method can be above mentioned house management example.

**Nominal Logistic Regression**

- Nominal Logistic Regression is used if there are three or more classes without any specific orders.
- The examples of this type of regression could be different departments of the engineering college (Computer, IT, Mechanical, Civil, Electronics etc.).

**Ordinal Logistic Regression**

- Ordinal Logistic Regression is used if there are three or more classes with specific orders.
- The examples of this type of regression could be how customers rate the taste of food using a scale of 1-3 (Bad, Good, and Excellent).

**Ex. 5.26.1 :** A Bank has to decide whether to sanction loan or not based on two attributes as person's income and his savings. The data is given in the following table where 1 represents loan is sanctioned and 0 represents loan is not sanctioned. Predict whether a person 3 will get a loan or not having annual income as 12.5 lakhs and savings as 10 lakhs.

| Person | Annual income in lakhs ($x_1$) | Savings in lakhs ($x_2$) | Loan sanctioned? (y) |
|---|---|---|---|
| 1 | 14.5 | 12.5 | 1 |
| 2 | 8.5 | 4.5 | 0 |

**Soln. :**

Initially assume logistic regression coefficients

$b_0 = b_1 = b_2 = 0$

For 1st row, $x_1 = 14.5$, $x_2 = 12.5$ and $y = 1$

Now we will calculate prediction for the first row,

$$\text{Prediction} = 1 / (1 + \hat{e}^{(-(b_0 + b_1 \times x_1 + b_2 \times x_2))})$$

$$\text{Prediction} = 1 / (1 + \hat{e}^{(-(0 + 0 \times 14.5 + 0 \times 12.5))})$$

$$\text{Prediction} = 0.5$$

Now we will calculate the new coefficient values using a simple update equation. Ideal values for alpha are from 0.1 to 0.3. Let's take alpha as 0.3. For $b_0$ by default input is 1.

$$b_{new} = b_{old} + \text{alpha} \times (y - \text{prediction}) \times \text{prediction} \times (1 - \text{prediction}) \times \text{input}$$

$$b_{0new} = 0 + 0.3 \times (1 - 0.5) \times 0.5 \times (1 - 0.5) \times 1.0$$

$$= 0.0375$$

$$b_{1new} = 0 + 0.3 \times (1 - 0.5) \times 0.5 \times (1 - 0.5) \times 14.5$$

$$= 0.54375$$

$$b_{2new} = 0 + 0.3 \times (1 - 0.5) \times 0.5 \times (1 - 0.5) \times 12.5$$

$$= 0.46875$$

Now we will calculate prediction for the second row,

$$\text{Prediction} = 1 / (1 + \hat{e}^{(-(b_0 + b_1 \times x_1 + b_2 \times x_2))})$$

$$\text{Prediction} = 1 / (1 + \hat{e}^{(-(0.0375 + 0.54735 \times 8.5 + 0.46875 \times 4.5))})$$

$$\text{Prediction} = 0.99$$

Now we will calculate the new coefficient values

$$b_{new} = b_{old} + \text{alpha} \times (y - \text{prediction}) \times \text{prediction} \times (1 - \text{prediction}) \times \text{input}$$

$$b_{0new} = 0.0375 + 0.3 \times (0 - 0.99) \times 0.99 \times (1 - 0.99) \times 1.0$$

$$= 0.034$$

$$b_{1new} = 0.54735 + 0.3 \times (0 - 0.99) \times 0.99 \times (1 - 0.99) \times 8.5$$

$$= 0.523$$

$$b_{2new} = 0.46875 + 0.3 \times (0 - 0.99) \times 0.99 \times (1 - 0.99) \times 4.5$$

$$= 0.456$$

Now we will use these values for prediction

$$\text{Prediction} = 1 / (1 + \hat{e}^{(-(b_0 + b_1 \times x_1 + b_2 \times x_2))})$$

$$\text{Prediction} = 1 / (1 + \hat{e}^{(-(0.034 + 0.523 \times 12.5 + 0.456 \times 10))})$$

$$\text{Prediction} = 0.99$$

Since prediction $\geq 0.5$

Prediction for person 3 is, his loan will be sanctioned.

**Note :** Generally huge amount of data is used for training and number of iterations are also applied to get accuracy. Here for example purpose only two records for training are taken and a single iteration is shown.

*Chapter Ends...*

□□□

MODULE 6

CHAPTER 6

# Dimensionality Reduction

**Syllabus**

Curse of Dimensionality, Feature Selection and Feature Extraction, Dimensionality Reduction Techniques, Principal Component Analysis.


6.1 Curse of Dimensionality ..... 6-3

6.1.1 Combinatorics ..... 6-3

6.1.2 Sampling ..... 6-3

6.1.3 Optimisaiton ..... 6-3

6.1.4 Anomaly Detection ..... 6-3

6.1.5 Machine Learning ..... 6-4

6.1.6 How to Combat the COD ? ..... 6-4

6.1.7 Mitigating Curse of Dimensionality ..... 6-4

6.2 Statistical FEatures ..... 6-4

6.2.1 Introduction ..... 6-4

6.2.2 Selected Features ..... 6-4

6.2.3 Analysis of Features ..... 6-5

6.2.4 Features of Big Data ..... 6-5

6.2.5 Features in a Model ..... 6-6

6.2.6 Four basic Elements of Statistics ..... 6-6

6.2.7 Types of variables in Statistics ..... 6-6

6.2.8 Statistical Methods ..... 6-6

6.2.9 Statistical Function ..... 6-6

6.3 Features and classification ..... 6-7

6.3.1 Feature Extraction (In Machine Learning) ..... 6-7

6.3.2 Why Feature Extraction is Useful ? ..... 6-7

6.3.3 Time Domain Features ..... 6-7

6.3.4 Applications of Feature Extraction ..... 6-8

6.4 Principal COmponent Analysis (P.C.A) ..... 6-8

6.4.1 Introduction ..... 6-8

6.4.2 Definition of P.C.A. ..... 6-8

6.4.3 Principal in Principal Component Analysis (P.C.A.) ..... 6-8

6.4.4 PC1, PC2, axis in P.C.A. ..... 6-9

6.4.5 Calculation of Principal Components ..... 6-9

6.4.6 Principal Component Analysis (P.C.A.) ..... 6-9

6.4.7 Working Rule of PCA ..... 6-9

6.4.8 Use of P.C.A. In Machine Learning ..... 6-10

6.4.9 Output of PCA ..... 6-10

6.4.10 Limitations of PCA ..... 6-10

6.4.11 Comparison between PCA and Cluster Analysis ..... 6-10

6.5 Feature extraction and Selection ..... 6-10

6.5.1 Feature Selection Ranking ..... 6-11

6.5.2 Advantages of PCA ..... 6-12

6.5.3 Disadvantages of PCA ..... 6-12

6.6 Dimensionality Reduction Techniques ..... 6-12

UQ. Why Dimensionality Reduction is very Important step in Machine Learning ? (Ref. MU (Comp.) - May 19, Dec. 19, 5 Marks) ..... 6-12

6.6.1 Dimension Reduction Techniques in ML ..... 6-13

6.6.1(A) Feature Selection ..... 6-13

6.6.1(B) Feature Extraction ..... 6-14

6.7 PrincipAL Component Analysis ..... 6-14

UQ. Explain in detail Principal Component Analysis for Dimension Reduction. (Ref. MU (Comp.) - May 15, May 16, 10 Marks) ..... 6-14

UQ. Use Principal Component analysis (PCA) to arrive at the transformed matrix for the given data.

$$A = \begin{bmatrix} 2 & 1 & 0 & -1 \\ 4 & 3 & 1 & 0.5 \end{bmatrix}$$

(Ref. MU (Comp.) - May 17, 10 Marks) ..... 6-14

UQ. Describe the steps to reduce dimensionality using Principal Component Analysis method by clearly stating mathematical formulas used. (Ref. MU (Comp.) - May 19, Dec. 19, 5 Marks) ..... 6-14

UQ. Write Short note on ISA and compare it with PCA. (Ref. MU (Comp.) - May 19, 5 Marks) ..... 6-14

- **Chpater End** ..... 6-16

## 6.1 CURSE OF DIMENSIONALITY

- The curse of dimensionality refers to various phenomena that arise when we analyse and organise data in **high-dimensional spaces**. These phenomena do not occur in low-dimensional settings such as the thr ee dimensional **physical space** of everyday experience.
- Dimensionally cursed phenomena occur in domains such as **numerical analysis, sampling, combinatorics, machine learning data mining, and databases.**
- The actual and common problem is that when the dimensionality increases, the **volume** of the **space** increases. And it increases so fast that the available data becomes sparse.
- In order to obtain a reliable result, the amount of data needs often grows exponentially with the dimensionality.
- Also, organising and searching data often relies on detecting areas where objects form groups with similar properties. But, in high dimensional data, all objects appear to be sparse and dissimilar in many ways and that prevents common data organisation strategies from being efficient.

### 6.1.1 Combinatorics

- In some problems, each variable can take one of several discrete values, or the range of possible values is divided to give a finite number of possibilities.
- Taking the variables together, a huge number of combinations of values must be considered.
- This effect is known as **combinatorial explosion.**
- Even in the simplest case of **d binary** variables, the number of possible combinations is $2^d$.
- Thus, each additional dimension doubles the effect needed to try all combinations.

### 6.1.2 Sampling

- There is an exponential increase in volume associated with adding extra dimensions to a **mathematical space.**
- For example, $10^2 = 100$ evenly spaced sample points suffice to sample a **unit interval** with $10^{-2} = 0.01$ distance between points.
- An equivalent sampling of a 10-dimensional unit hypercube with a lattice that has a spacing of $10^{-2}$ $= 0.01$ between adjacent points would require $10^{20}$ $= (10^2)^{10}$ sample points.
- This effect is a combination of the combinatorics problems.

### 6.1.3 Optimisaiton

- When solving dynamic **optimisation**, problems by numerical **backword induction** the objective function must be computed for each combination of values.
- This is a significant obstacle when the dimension of the "state variable" is large.

### 6.1.4 Anomaly Detection

We come across the following problems when searching for anomalies in high dimensional data.

1. **Concentration of scores and distances :** derived values such as distances become numerically similar.
2. **Irrelevant attributes :** in high dimensional data, a significant number of attributes may be irrelevant.
3. **Definition of reference sets :** For local methods, reference sets are often nearest-neighbor based.
4. **Incomparable scores for different dimensionalities :** different subspaces produce incomplete scores.
5. **Interpretability of scores :** the scores often no longer convey a semantic meaning.
6. **Exponential search space :** the search space can no longer be systematically scanned.

7. **Data-snooping bias :** given the large search-space, for every desired significance a hypothesis can be found.

8. **Hubness :** certain objects occur more frequently in neighbour lists than others.

### 6.1.5 Machine Learning

- In machine learning, a marginal increase in dimensionality also requires a large increase in the volume in the data in order to maintain the same level of performance.
- The curse of dimensionality is the by-product of a phenomenon which appears with high dimensional data.

### 6.1.6 How to Combat the COD ?

- Combating COD is not such a big deal until we have dimensionality reduction. Dimensionality Reduction is the process of reducing the number of input variables in a dataset.
- This is known as converting high-dimensional variables into lower-dimensional variables without changing their attributes of the same.
- It does not contain any extra variables. That makes it very simple for analysts to analyse the data leading to faster results for algorithms.

### 6.1.7 Mitigating Curse of Dimensionality

- To mitigate the problems associated with high dimensional data, techniques generally, referred to as 'Dimensionality reduction' techniques are used.
- Dimensionality reduction techniques fall into one of the two categories - 'Feature selection' or 'feature extraction'.
- We shall deal with these techniques in the nest section.

## 6.2 STATISTICAL FEATURES

### 6.2.1 Introduction

- Statistical features is the most important features used in statistical data science.
- It is the first statistical technique which is applied while exploring a dataset and the expressions like bias, mean, variance, standard deviation, median, moments, percentiles etc.
- The field of utility of statistics has been increasing steadily and thus different people expressed features of statistics differently according to the developments of the subject.
- In old days statistics was regarded as the "Science of statecraft" but today it covers almost every sphere of natural and human activity. It is more exhaustive and elaborate in approach.

### 6.2.2 Selected Features

(1) Statistics are the classified facts representing the conditions of the people in a state specially those facts which can be stated in number or in tables of numbers or in any tabular or classified arrangement.

– Webster

(2) "Statistics are numerical statements of facts in any department of enquiry placed in relation to each other"

– Bowley

(3) "By statistics we mean quantitative data affected to a marked extent by multiplicity of causes"

– Yule and Kendall

(4) "Statistics is the aggregate of facts affected to a marked extent by multiplicity of causes, numerically expressed, enumerated or estimated according to a reasonable standard of accuracy, collected in a systematic manner, for a predetermined purpose and placed in relation to each other"

– Prof. Horace Secrist

### 6.2.3 Analysis of Features

- Statistical feature analysis includes frequency table, bar chart, pie chart, histogram, normal probability plot, cumulative frequency table etc.
- Term Analysis feature is used with attribute and variable. It is an independent variable.

**Remarks**

(1) According to Webster, features of statistics are only numerical facts. It only restricts the domain of statistics to the affairs of a state, i.e. to the social sciences, (economics etc). But this is very inadequate for modern times. Statistics covers all sciences – social, physical and natural.

(2) Bowley features out statistics in more general way. It relates the numerical data in any department of enquiry. It also refers to the comparative study of the figures as against mere classification and tabulation.

(3) In the case of social, economic and business phenomena, Yule and Kendall's feature refers to data affected by a multiplicity of causes. For example, supply of a particular commodity is affected by supply, demand, imports, exports, money in circulation, competitive products in the market and so on. Similarly, the yield of a particular crop depends upon multiplicity of factors like quality of seed, fertility of soil, method of cultivation, irrigation facilities, weather conditions, fertilizers used and so on.

(4) **Secrist's feature of statistics** is more exhaustive. We study them :

**(a) Aggregate of facts**

Simple or isolated items cannot be termed as statistics unless they are a part of aggregate of facts relating to any particular field of enquiry. The aggregate of figures of births, deaths, sales, purchase, production, profits etc. over different times, places etc. constitute statistics.

**(b) Affected by multiplicity of causes**

In physical sciences, it is possible to isolate the effect of various factors on a single item but it is very difficult to do so in social sciences, particularly when the effect of the some of the factors cannot be measured quantitatively. However, statistical techniques have been devised to study the joint effect of a number of factors on a single item or the isolated effect of a single factor on the given item provided the effect of each of the factors can be measured quantitatively.

**Definition of a Feature**

Feature is defined as to give or bring special attention to someone or something. For example, a part of the face, a quality, a special attraction, etc. An example : Nose.

**Feature in Data**

- Each feature represents a measurable piece of data that can be used for analysis. For example, name, age, sex, fare and so on.
- Features are also sometimes referred to as 'Variables' or 'Attributes'. Depending upon type of analysis, the features can vary widely.

### 6.2.4 Features of Big Data

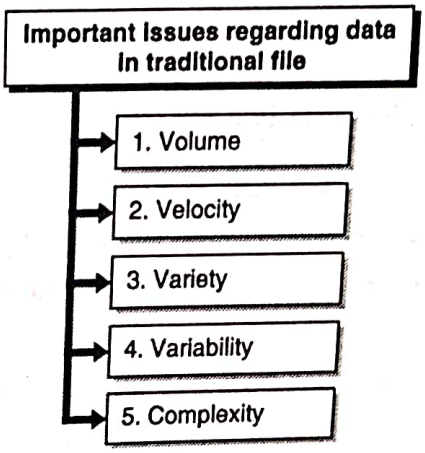



**Fig. 6.2.1 : Important issues regarding data in regarding data in traditional file**

**1. Volume**

Now a day the volume of data regarding different fields is high and potentially increasing day by day. Organizations collect data from a variety of sources, including business transactions, social media and information etc.

▶ **2. Velocity**

The configuration of system with single processor, limited RAM and limited storage capacity cannot store and manage high volume of data.

▶ **3. Variety**

The form of data from different sources is different.

▶ **4. Variability**

The flow of data coming from sources like social media is inconsistent because of daily emerging new trends. It can show sudden increase in size of data which is difficult to manage.

▶ **5. Complexity**

As the data is coming from various sources, it is difficult to link, match and transform such data across systems. It is necessary to connect and correlate relationships, hierarchies and multiple data linkages of the data.

All these issues are solved by the new advanced **Big Data Technology.**

### 6.2.5 Features in a Model

- It is a model that defines features and their dependencies, typically in the form of a feature – diagram (with constraints). Also it could be as a table of possible combinations.
- These are parts or patterns of an object in an image that help to identify it, e.g., a triangle, has three corners and 3 edges, these are features of the triangle, and they help us to identify it as a 'triangle'. Features include properties like corners, edges, regions of main points, ridges etc.

### 6.2.6 Four basic Elements of Statistics

They are : Population, sample; parameter, statistic (singular); and variable form the basic vocabulary of statistics.

### 6.2.7 Types of variables in Statistics

(1) Independent and dependent variables,

(2) Active and Attribute variables

(3) Continuous variable

(4) Discrete variable

(5) Categorical variable

(6) Extraneous variable, and

(7) Demographic variable

### 6.2.8 Statistical Methods

- In statistical analysis of raw research data, the methods used are : mathematical formulae, models and techniques (mathematical)
- The application of statistical methods extracts information from research data and provides different ways to assess the strength of research outputs. We use numerical (discrete, continuous), categorical and ordinal methods for the research output.

### 6.2.9 Statistical Function

- Statistics help in providing better understanding and accurate description of nature's phenomenon. Statistics helps in the proper and efficient planning of a statistical inquiry in any field of study.
- Statistics helps in collecting appropriate quantitative data. Basic three elements in statistics are : measurement, comparison and variation.
- A statistical function, such as mean, median, or variance summarizes of sample of values by a single value.
- They expect their parameters to be a probabilistic value represented by a random sample of values over the Run index.

The following functions are statistical functions and they are very similar to functions such as sum, max, min, average :

1. **Count Function**

We use count function when we need to count the number of cells containing a number.

2. **Counta function**

While the counta function only counts the numeric values, the COUNTA function counts all the cells in a range that are not empty. The function is useful for counting cells containing any type of information, including error values and empty text.

3. **Count blank function**

The count blank function counts the number of empty cells in a range of cells. Cells with formulae that return empty text are also counted but cells with zero values are not counted. This is an important function for summarizing empty cells while analyzing any data.

## 6.3 FEATURES AND CLASSIFICATION

Classification is the act of forming a class or classes; according to some common relations or attributes while feature is a distinguishable characteristic of a person or thing.

### 6.3.1 Feature Extraction (In Machine Learning)

- In real life, all the data we collect are in large amounts. To understand this data, we need a process. Manually, it is not possible to process them. Here's when the concept of feature extraction comes in.
- It is a general term for methods of constructing combination of the variables to get around these problems which still describe the data with sufficient accuracy.
- Property optimized feature extraction is the key to effective model construction.

### 6.3.2 Why Feature Extraction is Useful ?

- The technique of extracting the features is useful when you have a large data set and need to reduce the number of resources without losing any important or relevant information. Feature extraction helps to reduce the amount of redundant data from the data set.
- In the end, the reduction of the data helps to build the model with less machine's efforts and also increase the speed of learning and generalization steps in the machine learning process.

(1) The **goal** of statistical pattern feature extraction (SPFE) is 'low dimensional reduction'.

(2) Five measures of statistical features are :

(i) The most extreme values in the data set (the maximum and minimum values),

(ii) The lower and upper quartiles, and the

(iii) median

(3) These values are presented together and ordered from lowest to highest : minimum value, lower quartile ($Q_1$), median value ($Q_2$), upper quartile ($Q_3$), Maximum value.

(4) Statistical features of image are :

- They are the binary object features. They include area, centre of area, axis of least second moment, perimeter, Euler member, projection, thinness ratio, and aspect ratio.
- Area, center of area, axis of least second moment, perimeter tell us something about where the object is.
- The other four tell us about the shape of the objects.

### 6.3.3 Time Domain Features

- Time domain features refer to the analysis of mathematical functions, physical signals or time series of economic or environmental data, with respect to time.
- In time-domain, the signal of function's value is known for all real numbers. It is also valid for continuous time, or at various separate instants in the case of **discrete time**.

- For time series data, feature extraction can be performed using various time series analysis and decomposition techniques.
- Also, features can be obtained by sequence comparison techniques such as dynamic time wrapping and by subsequence discovery techniques such as motif analysis.
- Time-domain feature shows how a signal changes with time, while a frequency domain graph shows how much of the signal lies within each given frequency band over the range of frequencies.

**Feature Values**

It is a determination made by the customer about which feature is most important to the customers.

**Shape Features**

- Shape feature provides an alternative to describing an object, using its most important characteristic and reduce the amount of information stored.
- The algorithm is comprised of a curvature approximation technique.

**Global Features**

It describes the visual content of the whole image which represents an image by one vector, whereas local features describe them as a set of vectors.

### 6.3.4 Applications of Feature Extraction

(1) **Bag of Words- Bag-of-**Words is the_most used technique for natural language processing. In this process they extract the words or the features from a sentence, document, website, etc. and then they classify them into the frequency of use. So in this whole process feature extraction is one of the most important parts.

(2) **Image Processing** : Image processing is one of the best and most interesting domain. In this domain basically you will start playing with your images in order to understand them. So here we use many techniques which includes feature extraction as well and algorithms to detect features such as shaped, edges, or motion in a digital image or video to process them.

(3) **Auto-encoders** : The main purpose of the **auto-encoders** is efficient data coding which is unsupervised in nature. This process comes under unsupervised learning. So Feature extraction procedure is applicable here to identify the key features from the data to code by learning from the coding of the original data set to derive new ones.

## 6.4 PRINCIPAL COMPONENT ANALYSIS (P.C.A)

### 6.4.1 Introduction

- Principal component represents a multivariate data table as smaller set of variables in order to observe trends, jumps, clusters and outliers.
- This overview may give us the relationships between observations and variables and also among variables.

### 6.4.2 Definition of P.C.A.

- It is a technique used for identification of a smaller number of uncorrelated variables known as principal components from a large set of data. This technique is widely used to emphasise variation and capture strong patterns in a data.
- Mathematically, P.C.A. can be described as follows :

The principal component is a collection of points in a real coordinate space as a sequence of P-unit vectors, where the ($i^{th}$) vector is the direction of a line that best fits the data and is orthogonal to the first (i – 1) vectors.

### 6.4.3 Principal in Principal Component Analysis (P.C.A.)

- It is a mathematical algorithm that reduces the dimensionality of the data while retaining most of the variations in the data-set.

- It accomplishes this by identifying directions, called principle components (PC) along which the variation in the data is maximal.

### 6.4.4 PC1, PC2, axis in P.C.A.

- In PCA directions with the largest variances are the most important (i.e. most principal). PC1 axis is the first principal direction along which the samples show the largest variation.
- The PC2 is the second most direction and it is orthogonal to PC1 axis (Refer to above article 6.4.2 )

### 6.4.5 Calculation of Principal Components

1. Consider the whole dataset consisting of (d + 1) dimensions and ignore labels so that our new dataset becomes of 'd' dimension.
2. Compute mean for every dimension of the whole dataset.
3. Compute the covariance matrix of the whole dataset.
4. Compute eigenvalues and eigenvectors for each dataset (i.e. of each dimensional).
5. Variance is calculated as; the ratio of eigenvalue of a particular principal component (eigenvector) with the total eigen values.

### 6.4.6 Principal Component Analysis (P.C.A.)

- PCA is a method used for maximizing the information content in high-dimensional data. In PCA, a large number of correlated variables are transformed into a smaller number of uncorrelated variables and these are called as principal components.
- In mathematical terms, PCA finds a set of m orthogonal vectors in data space having largest variance and then we project the data of n-dimension to m-dimensional subspace where m << n, and we are able to reduce the dimensionality of the given data.

### 6.4.7 Working Rule of PCA

(i) We calculate covariance matrix

First consider two-dimensional data set (bivariate data set ) along X as Y-axis. Let ($x_i$; $y_i$) be the coordinates of the points.

Let $\bar{x}$ and $\bar{y}$ be the means of two data, (a) Covariance measures the relationship between the two variables.

(b) Variance operates on a single variable

$$\text{Now, COV (x, y)} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n} \quad \ldots(i)$$

$$\text{While Var (x)} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n} \quad \ldots(ii)$$

$$= \frac{\sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})}{n} \quad \ldots(iii)$$

The covariance matrix is

$$C = \begin{bmatrix} \text{cov (x, x)} & \text{cos (x, y)} \\ \text{cov (y,x)} & \text{cos (y,y)} \end{bmatrix}$$

$$= \begin{bmatrix} \text{var (x)} & \text{cov (x, y)} \\ \text{cov (y,x)} & \text{var (y)} \end{bmatrix} \quad \begin{bmatrix} \because \text{cov (x,x) = var (x) and} \\ \text{cov (y, y) = var (y)} \end{bmatrix}$$

(ii) To calculate eigenvalues and eigenvectors of the covariance matrix

Let $\lambda_1, \lambda_2, \ldots.\lambda_n$ be the eigenvalues.

We arrange them in decreasing order, i.e., $\lambda_1, \lambda_2, \ldots.\lambda_n$.

Let $e_1$, $e_2$, ....$e_n$ be the eigenvectors; then the eigenvector $e_1$ will represent the axis with maximum variance.

(iii) Once eigenvalues are determined and arranged in decreasing order, we ignore the eigenvalues of lesser significance, and the dimensionality of the original data is reduced. If there are n eigenvoues and we decide to keep only m largest eigenvalues.

Then the corresponding m eighvectors form the columns of a matrix called as a 'feature vector'. Then we project the original data set on the feature space formed by the feature vector

### 6.4.8 Use of P.C.A. In Machine Learning

- PCA is unsupervised learning algorithm that is used for the dimensionality reduction in machine learning.
- PCA works by considering the variance of each attribute because higher attribute shows the good split between the classes and hence it reduces dimensionality.
- P.C.A. is a statistical process that converts the observations of correlated features into a set of linearly uncorrelated features with the help of orthogonal transformations.
- These new transformed features are called the principal components.
- It is one of the popular tools that is used for exploratory data analysis and predictive modeling.
- It is a technique to draw strong patterns from the given dataset by reducing the variances.
- PCA tries to find the lower-dimensional surface to project the high-dimensional data.

### 6.4.9 Output of PCA

Just now we mentioned that PCA is dimensionality reduction algorithm. It helps in reducing the dimensions of our data. PCA gives an output of eigenvectors in decreasing order such as PC1, PC2, ...And these will become new axes, for our data.

### 6.4.10 Limitations of PCA

(i) Low interpretability of principal components. Principal components are linear combinations of the features from the original data but they are not so easy to interpret.

(ii) In dimensionality reduction there is information loss.

(iii) On applying P.C.A., the independent features become less interpretable because these principal components are also not readable or interpretable.

(iv) Data must be standardized before implementing PCA, otherwise PCA will not be able to find the optimal principal components.

(v) If care is not taken in selecting the principal components, it may miss some information.

### 6.4.11 Comparison between PCA and Cluster Analysis

(i) Cluster analysis groups observations while PCA groups variables rather than observations.

(ii) PCA can be used as a final method or to reduce number of variables to conduct another analysis such as regression analysis or other data mining techniques.

(iii) Clustering reduces the number of "data points" by summarising several points by their expectation/k-mean.

## 6.5 FEATURE EXTRACTION AND SELECTION

- Principal Component Analysis (PCA) is the process of computing the principal components and using them to perform a **change of basis** in the data, sometimes using only the first few principal components and ignoring the rest.
- PCA is used in **exploratory data analysis** and for making **predictive models**. It is commonly used for **dimensionality reduction** by projecting each data points onto only the first few principal components. Thereby lower dimensional data is obtained and it preserves as much of the data's variation as possible.
- The first principal component is defined as a direction that maximises the variance of the projected data.
- The $fi^{th}$ principal component can be taken as a direction orthogonal to the first (i – 1) principal components and

that maximises the variance of the projected data. The principal components are **eigenvectors** of the data's covariance matrix.

- Principal components are after computed by eigen decomposition of the data covariance matrix or **singular value decomposition** of the data matrix. PCA is closely related to **factor analysis.**
- Factor analysis incorporates more domain specific assumptions about the underlying structure and solves eigenvectors of a slightly different matrix. PCA is also related to canonical correlation Analysis (CCA).
- CCA defines coordinate systems that optimally describe the cross covariance between two datasets while PCA defines a new **orthogonal coordinate system** and it optimally describes variance in a single dataset.
- Thus PCA is defined as an **orthogonal linear transformation** that transforms the data to a **new coordinate system** such that the greatest variance by some scalar projection of the data comes to lie on the firs coordinate (called the first principal component), the second greatest variance on the second coordinate and so on.
- Consider an $n \times p$ data matrix, X with columnwise **zero empirical mean,** where each of the n rows represents a different repetition of the experiment, and each of the p columns gives a particular kind of feature.

Mathematically, the transformation is defined by a set of size *l* of p-dimensional vectors of weights or coefficients.

$w_{(k)} = (w_1,\ldots,w_p)_{(k)}$ that map peach row vector $X_{(i)}$ of X to a new vector of principal component scores $t_{(i)} = (t_1,..,t_l)_{(i)}$, given by $t_{k(i)} = X_{(i)} W_{(k)}$, for i = 1, 2,…,n. (l is usually selected as strictly less than p to reduce dimensionality)

☞ **First component**

In order to maximize variance, the first weight vector $W_{(1)}$ satisfies

$$W_{(1)} = \arg\max_{\|w\|=1} \left\{ \sum (t_1)^2_{(i)} \right\}$$

$$= \arg\max_{\|w\|=1} \left\{ \sum_i (x(i) \cdot w)^2 \right\}$$

☞ **Further components**

The $k^{th}$ component can be found by subtracting the first k – 1 principal components from X.

**Table 6.5.1 : Symbols and abbreviations**

| Symbol | Meaning | Dimensions | Indices |
|---|---|---|---|
| $X = \{X_{(i)}\}$ | Data matrix consisting of the set of all data vectors, one vector per raw. | $n \times p$ | i = 1,…,n<br>j = 1,…,n |
| n | The number of row vectors in the data set | $1 \times 1$ | Scalar |
| p | The number of element in each row vector (dimension) | $1 \times 1$ | Scalar |
| L | The number of dimensions in the dimensionally reduced subspace $1 \le L \le p$ | $1 \times 1$ | Scalar |
| $u = \{u_j\}$ | Vector of empirical means, one mean for each column j of the data matrix | $p \times 1$ | j = 1,…,p |
| $s = \{s_j\}$ | One standard deviation for each column j of the data matrix | $p \times 1$ | j = 1,…,p |
| $h = \{h_i\}$ | Vector of all 1's | $1 \times n$ | i = 1,..,n |

## 6.5.1 Feature Selection Ranking

- Many feature selection methods used in classification are directly applied to ranking. Specifically, for each

feature we use its value to rank the training instances and define the ranking accuracy in terms of a performance measure or a loss function as the importance of the feature.

- Feature-ranking can be used for feature selection by evaluating the information gain of each variable in the context of the target variable. The effective feature selection can improve both accuracy and efficiency of learning for ranking.

### 6.5.2 Advantages of PCA

1. Lack of redundancy of data given the orthogonal components.
2. Principal components are independent of each other, so removes correlated features.
3. PCA improves the performance of the ML algorithm as it eliminates correlated variables that don't contribute in any decision making.
4. PCA helps in overcoming data overfitting issues by decreasing the number of features.
5. PCA results in high variance and thus improves visualization.
6. Reduction of noise since the maximum variation basis is chosen and so the small variations in the background are ignored automatically.

### 6.5.3 Disadvantages of PCA

1. It is difficult to evaluate the covariance in a proper way.
2. Even the simplest invariance could not be captured by the PCA unless the training data explicitly provide this information.
3. Data needs to be standardized before implementing PCA else it becomes difficult to identify optimal principal components.
4. Though PCA covers maximum variance amid data features, sometimes it may skip a bit of information in comparison to the actual list of features.
5. Implementing PCA over datasets leads to transforming actual features in principal components that are linear combinations of actual features, therefore principle components are difficult to read or interpret as compared to actual features.

## 6.6 DIMENSIONALITY REDUCTION TECHNIQUES

**UQ.** Why Dimensionality Reduction is very Important step in Machine Learning ?
**(Ref. (MU (Comp.) - May 19, Dec. 19, 5 Marks))**

- Dimension Reduction alludes to the way toward changing over an arrangement of information having tremendous dimensions into information with lesser dimensions guaranteeing that it passes on comparable data briefly. These methods are normally utilized while tackling machine learning issues to acquire better properties for classification or regression problem.
- We should take a gander at the picture demonstrated as follows. It indicates 2 dimensions $P_1$ and $P_2$, which are given us a chance to state estimations of a few objects in cm ($P_1$) and inches ($P_2$). Presently, if we somehow managed to utilize both the measurements in machine learning, they will pass on comparable data and present a lot of noise in the system, so it is better to simply utilizing one dimension. Dimension of information present here is changed over from 2D (from $P_1$ and $P_2$) to 1D ($Q_1$), to make the information moderately less demanding to clarify.

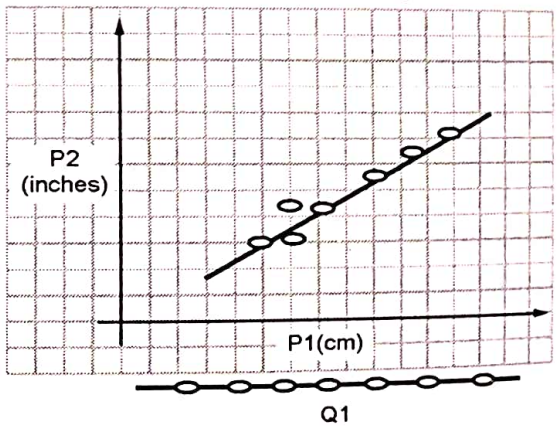



**Fig. 6.6.1 : Example of Dimension Reduction**

There are number of methods using which, we can get k dimensions by reducing n dimensions (k < n) of informational index. These k dimensions can be specifically distinguished (sifted) or can be a blend of dimensions (weighted midpoints of dimensions) or new dimension(s) that speak to existing numerous dimensions well. A standout amongst the most well-known use of this procedure is Image preparing.

Now we will see the significance of applying Dimension Reduction method :

- It assists in information packing and diminishing the storage room required
- It decreases the time which is required for doing same calculations. If the dimensions are less then processing will be less, added advantage of having less dimensions is permission to use calculations unfit for countless.
- It handles with multi-collinearity that is used to enhance the execution of the model. It evacuates excess highlights. For example: it makes no sense to put away an incentive in two distinct units (meters and inches).
- Lessening the dimensions of information to 3D or 2D may enable us to plot and visualize exactly. You would then be able to watch designs all the more unmistakably. Beneath we can see that, how a 3D information is changed over into 2D. It has distinguished the 2D plane at that point spoke to the focuses on these two new axis $z_1$ and $z_2$.

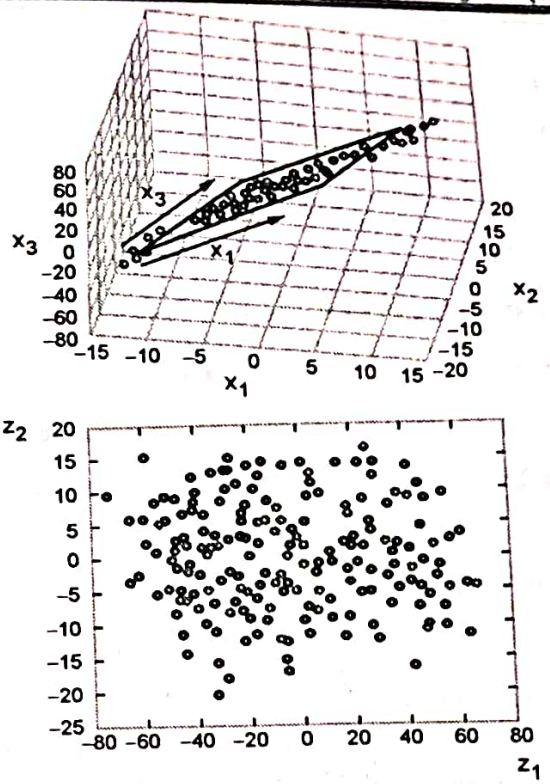



**Fig. 6.6.2 : Example of Dimension Reduction**

- It is helpful in noise evacuation additionally and because of this we can enhance the execution of models.
- The classifier's performance usually will degrade for a large number of features.

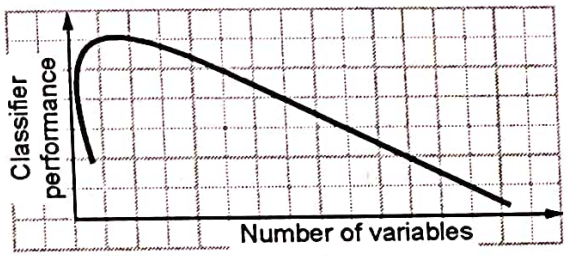



**Fig. 6.6.3 : Classifier performance and amount of data**

## 6.6.1 Dimension Reduction Techniques in ML

### 6.6.1(A) Feature Selection

Given a set of features $F = \{X_1, \ldots\ldots, X_n\}$. The feature selection problem is to find a subset $F' \subseteq F$ that maximizes the learner's ability to classify the patterns. Finally F' should maximize some scoring function.

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \xrightarrow[-\quad -]{\text{Feature selection}} \begin{bmatrix} x_{i_1} \\ x_{i_2} \\ \cdot \\ \cdot \\ x_{i_M} \end{bmatrix}$$

### ☞ Feature Selection Steps

Feature selection is an **optimization** problem.

- **Step 1 :** Search the space of possible feature subsets.
- **Step 2 :** Pick the subset that is optimal or near-optimal with respect to some objective function.

### ☞ Subset selection

- There are d initial features and $2^d$ possible subsets
- Criteria to decide which subset is the best:
  - Classifier based on these m features has the lowest probability of error of all such classifiers
- It is not possible to go over all $2^d$ possibilities, so we need some heuristics
- Here we select uncorrelated features
- Forward search
  - Start from empty set of features
  - Try each of remaining features
  - Estimate classification/regression error for adding specific feature
  - Select feature that gives maximum improvement in validation error
  - Stop when no significant improvement
- Backward search
  - Start with original set of size *d*
  - Drop features with smallest impact on error.

### 6.6.1(B) Feature Extraction

Suppose a set of features F = {$X_1$, ....., $X_N$} is given. The Feature Extraction task is to map F to some feature set F″ that will maximize the learner's ability to classify patterns.

$$\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_N \end{bmatrix} \xrightarrow[-\quad -]{\text{Feature extraction}} \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ y_m \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ x_N \end{bmatrix}\right)$$

A projection matrix w is computed from N-dimensional to M-dimensional vectors to achieve low error.

$Z = w^T x$. Principle component analysis and Independent component analysis are the feature extraction methods.

## 6.7 PRINCIPAL COMPONENT ANALYSIS

**UQ.** Explain in detail Principal Component Analysis for Dimension Reduction.

**(Ref. MU (Comp.) - May 15, May 16, 10 Marks)**

**UQ.** Use Principal Component analysis (PCA) to arrive at the transformed matrix for the given data.

$$A = \begin{bmatrix} 2 & 1 & 0 & -1 \\ 4 & 3 & 1 & 0.5 \end{bmatrix}$$

**(Ref. MU (Comp.) - May 17, 10 Marks)**

**UQ.** Describe the steps to reduce dimensionality using Principal Component Analysis method by clearly stating mathematical formulas used.

**(Ref. MU (Comp.) - May 19, Dec. 19, 5 Marks)**

**UQ.** Write Short note on ISA and compare it with PCA.

**(Ref. MU (Comp.) - May 19, 5 Marks)**

- Dimensional reduction is regularly the best beginning stage when dealing with high dimensional information.
- It is utilized for an assortment of reasons, from perception to denoising, and in a wide range of uses, from signal processing to bioinformatics.
- A standout amongst the most broadly utilized dimensional reduction tools is Principal Component Analysis (PCA).

- PCA verifiably accept that the dataset under thought is typically dispersed, furthermore, chooses the subspace which expands the anticipated difference.
- We consider a centered data set, and develop the sample covariance matrix, at that point q-dimensional PCA is identical to anticipating onto the q-dimensional subspace spread over by the q eigenvectors of S with biggest eigen values.
- In this system, variables are changed into another arrangement of variables, which are straight blend of unique variables. These new arrangement of variables are known as **principal components**. They are calculated so that first **principal component** s represents a large portion of the conceivable variety of unique information after which each succeeding component has the most noteworthy conceivable variance.
- The second principal component should be symmetrical to the primary principal component. As it were, it does its best to catch the difference in the information that isn't caught by the primary principal component. For two-dimensional dataset, there can be just two principal components. The following is a depiction of the information and its first and second principal component. You can see that second principal component is symmetrical to first principal component.

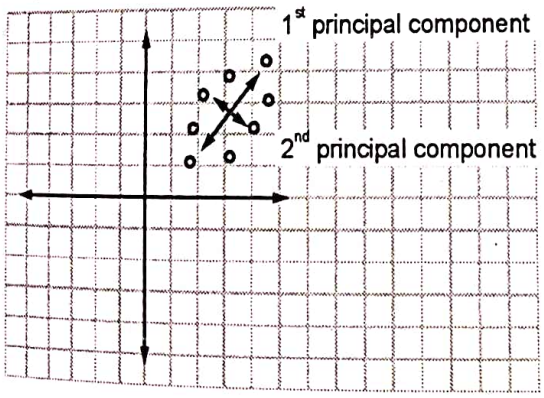



**Fig. 6.7.1**

- The principal components are sensitive to the size of estimation, now to settle this issue we ought to dependably institutionalize factors before applying PCA. Applying PCA to your informational collection loses its importance. In the event that interpretability of the outcomes is critical for your investigation, PCA isn't the correct system for your task.

**Ex. 6.7.1 :** Apply PCA on the following data and find the principal component

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| X | 2.5 | 0.5 | 2.2 | 1.9 | 3.1 | 2.3 | 2 | 1 | 1.5 | 1.1 |
| Y | 2.4 | 0.7 | 2.9 | 2.2 | 3 | 2.7 | 1.6 | 1.1 | 1.6 | 0.9 |

☑ **Soln. :**

- First we will find the mean values

$$X_m = \sum \frac{X}{N} \ldots\ldots\ldots\ldots\ldots\ldots$$

N = number of data points = 10

$$X_m = 1.81$$

$$Y_m = \sum \frac{Y}{N}$$

$$Y_m = 1.91$$

Now we will find the covariance matrix, $C = \begin{bmatrix} C_{XX} & C_{XY} \\ C_{YX} & C_{YY} \end{bmatrix}$

$$C_{XX} = \sum \frac{(X - X_m)^2}{N-1}$$

$$= 0.6165$$

$$C_{XY} = C_{YX} = \sum \frac{(X - X_m)(Y - Y_m)}{N-1}$$

$$= 0.61544$$

$$C_{YY} = \sum \frac{(Y - Y_m)^2}{N-1} = 0.7165$$

$$C = \begin{bmatrix} 0.6165 & 0.61544 \\ 0.61544 & 0.7165 \end{bmatrix}$$

- Now to find the eigen values following equation is used

$$|C - \lambda I| = 0$$

$$\left| \begin{bmatrix} 0.6165 & 0.61544 \\ 0.61544 & 0.7165 \end{bmatrix} - \lambda \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \right| = 0$$

- By solving the above determinant we will get quadratic equation and solving that equation we will get two eigen values of $\lambda$ as $\lambda_1 = 0.0489$ and $\lambda_2 = 1.284$
- Now we will find the eigen vectors corresponding to eigen values

- First we will find the first eigen vector corresponding to first eigen value, $\lambda_1 = 0.0489$ .

$$C \times V_1 = \lambda_1 \times V_1$$

$$= \begin{bmatrix} 0.6165 & 0.61544 \\ 0.61544 & 0.7165 \end{bmatrix} \begin{bmatrix} V_{11} \\ V_{12} \end{bmatrix}$$

$$= 0.0489 \begin{bmatrix} V_{11} \\ V_{12} \end{bmatrix}$$

From the above equation we will get two equations as

$$0.6165\ V_{11} + 0.61544\ V_{12} = 0.0489\ V_{11} \quad ...(1)$$

$$0.61544\ V_{11} + 0.7165\ V_{12} = 0.0489\ V \quad ...(2)$$

- To find the eigen vector we can take either Equation (1) or (2), for both the equations answer will be the same, let's take the first equation

$$0.6165\ V_{11} + 0.61544\ V_{12} = 0.0489\ V_{11}$$

$$0.5676\ V_{11} + 0.61544\ V_{12} = 0$$

$$V_{11} = -0.61544\ V_{12}$$

$$V_{11} = -1.0842\ V_{12}$$

- Now we will assume $V_{12} = 1$ then $V_{11} = -1.0842$

$$V_1 = \begin{bmatrix} -1.0842 \\ 1 \end{bmatrix}$$

- As we are assuming the value of $V_{12}$ we have to normalized $V_1$ as follow

$$V_{1N} = \frac{1}{\sqrt{1.0842^3 + 1^3}} \begin{bmatrix} -1.0842 \\ 1 \end{bmatrix}$$

$$V_{1N} = \begin{bmatrix} -0.735 \\ 0.677 \end{bmatrix}$$

- Similarly we will find the second eigen vector corresponding to second eigen value, $\lambda_2 = 1.284$

$$C \times V_2 = \lambda_2 \times V_2$$

$$= \begin{bmatrix} 0.6165 & 0.61544 \\ 0.61544 & 0.7165 \end{bmatrix} \begin{bmatrix} V_{11} \\ V_{12} \end{bmatrix}$$

$$= 1.284 \begin{bmatrix} V_{11} \\ V_{12} \end{bmatrix}$$

- From the above equation we will get two equations as

$$0.6165\ V_{21} + 0.61544 V_{22} = 0.0489\ V_{21}$$

$$0.61544\ V_{21} + 0.7165 V_{22} = 0.0489\ V_{22}$$

- To find the eigen vector we can take either Equation (1) or (2), for both the equations answer will be the same, let's take the first equation

$$= 0.6165\ V_{21} + 0.61544\ V_{22}$$

$$= 1.284\ V_{21} - 0.6675\ V_{21} = -0.61544 V_{22}$$

$$V_{21} = 0.922\ V_{22}$$

- Now we will assume $V_{22} = 1$ then $V_{21} = 0.922$

$$V_2 = \begin{bmatrix} 0.922 \\ 1 \end{bmatrix}$$

- As we are assuming the value of $V_{22}$ we have to normalized $V_2$ as follows

$$V_{2N} = \frac{1}{\sqrt{0.922^3 + 1^3}} \begin{bmatrix} 0.922 \\ 1 \end{bmatrix}$$

$$V_{2N} = \begin{bmatrix} 0.677 \\ 0.735 \end{bmatrix}$$

- Now we have to find the principal component, it is equal to the eigen vector corresponding to maximum Eigen value, in this $\lambda_2$ is maximum, hence principal component is Eigen vector $V_{2N}$

$$\text{Principal component} = \begin{bmatrix} 0.677 \\ 0.735 \end{bmatrix}$$

*Chapter Ends...*

□□□