Artificial Neural Network

Chapter 10

Module 5

INTRODUCTION

- The human nervous system has billions of neurons that are the processing units which make humans to perceive things, to hear, to see and to smell.
- The human nervous system work beautifully, making us understand who we are, what we do, where we are and everything is our surrounding.
- It makes us to remember, recognize and correlate things around us. It is learning system that consists of functional units called nerve cells, typically called as neurons.
- The human nervous system is divided into two sections called the Central Nervous System (CNS) and the Peripheral Nervous System (PNS).

INTRODUCTION

- The brain and the spinal cord constitute the CNS and the neurons inside and outside the CNS constitute the PNS. The neurons are basically classified into three types called sensory neurons, motor neurons and interneurons.
- Sensory neurons get information from different parts of the body and bring it into the CNS, where as motor neurons receive information from other neurons and transmit commands to the body parts.
- The CNS consists of only interneurons which connect one neuron to another neuron by receiving information from one neuron and transmitting it to another. The basic functionality of a neuron is to receive information, process it and then transmit it to another neuron or to a body part.

BIOLOGICAL NEURONS

- A typical biological neuron has four parts called dendrites, soma, axon and synapse.
- The body of the neuron is called as soma. Dendrites accept the input information and process it in the cell body called soma.
- A single neuron is connected by axons to around 10,000 neurons and through these axons the processed information is passed from one neuron to another neuron.
- A neuron gets fired if the input information crosses a threshold value and transmits signals to another neuron through a synapse.
- A synapse gets fired with an electrical impulse called spikes which are transmitted to another neuron. A single neuron can receive synaptic inputs from one neuron or multiple neurons.

BIOLOGICAL NEURONS

- These neurons form a network structure which processes input information and gives out a response. The simple structure of a biological neuron is shown in Figure 10.1.

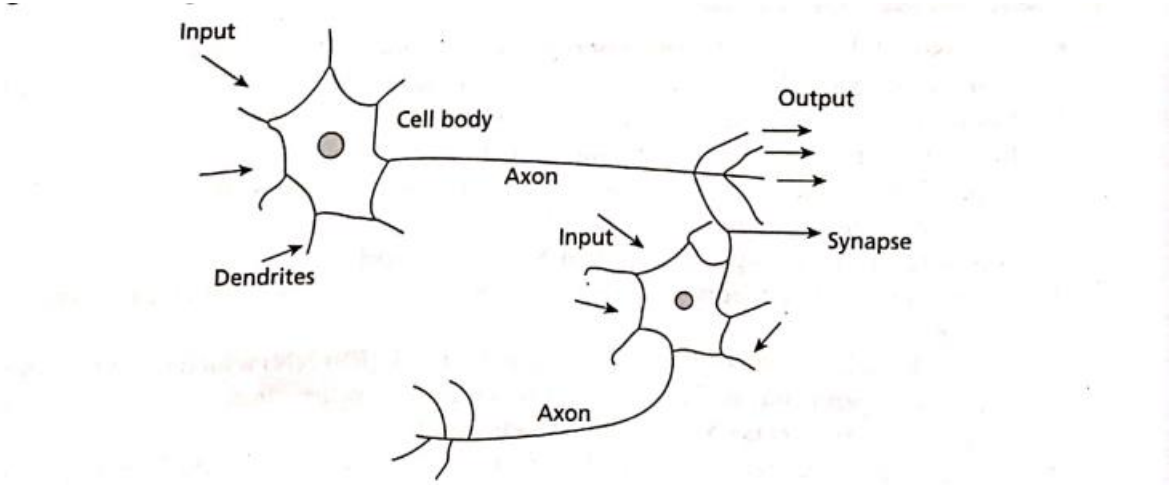


Figure 10.1: A Biological Neuron

ARTIFICIAL NEURONS

- Artificial neurons are like biological neurons which are called as nodes.
- A node or a neuron can or more input information and process it. Artificial neurons or nodes are connected by connection links to one another.
- Each connection link is associated with a synaptic weight. The structure of a single neuron is shown in Figure 10.2.

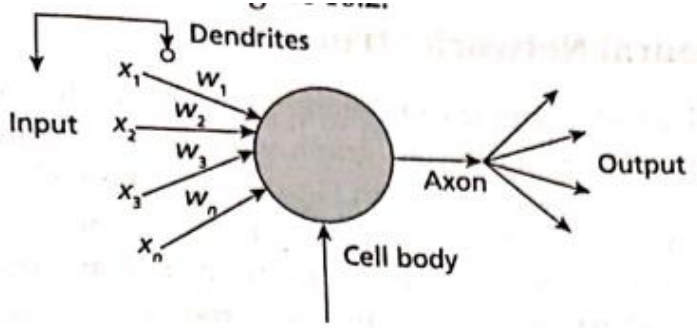


Figure 10.2: An Artificial Neuron

Simple Model of an Artificial Neuron

- The first mathematical model of a biological neuron was designed by McCulloch & Pitts in 1943.
- It includes two steps:
  - It receives weighted inputs from other neurons
  - It operates with a threshold function or activation function

  The received inputs are computed as a weighted sum which is given to the activation function and if the sum exceeds the threshold value the neuron gets fired. The mathematical model of a neuron is shown in Figure 10.3.

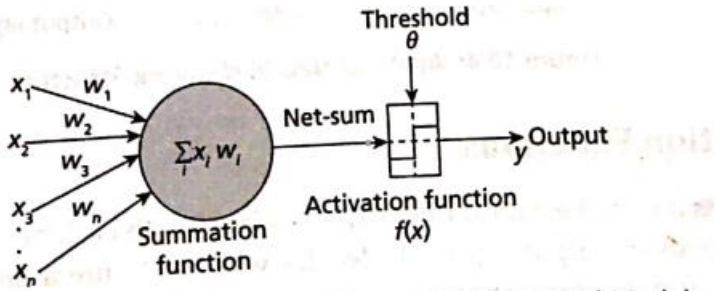


Figure 10.3: McCulloch & Pitts Neuron Mathematical Model

Simple Model of an Artificial Neuron

- The neuron is the basic processing unit that receives a set of inputs x1,x2, ... ,x, and their associated weights w1,w2, ... ,wn.
- The Summation function 'Net-sum' Eq. (10.1) computes the weighted sum of the inputs received by the neuron.

$$\text{Net-sum} = \sum_{i=1}^{n} x_i w_i$$

- The activation function is a binary step function which outputs a value 1 if the Net-sum is above the threshold value 0, and a 0 if the Net-sum is below the threshold value 0.
- Therefore, the activation function is applied to Net-sum as shown in Eq. (10.2).

$$f(x) = \text{Activation function } (Net - sum)$$

$$\text{Then, output of a neuron } Y = \begin{cases} 1 & \text{if } f(x) \geq \theta \\ 0 & \text{if } f(x) < \theta \end{cases}$$

Artificial Neural Network Structure

- Artificial Neural Network (ANN) imitates a human brain which inhibits some intelligence.
- It has a network structure represented as a directed graph with a set of neuron nodes and connection links or edges connecting the nodes as shown in Figure 10.4.
- The nodes in the graph are arrayed in a layered manner and can process information in parallel.
- The network given in the figure has three layers called input layer, hidden layer and output layer.
- The input layer receives the input information (x1, x2, ... , x,) and passes it to the nodes in the hidden layer.

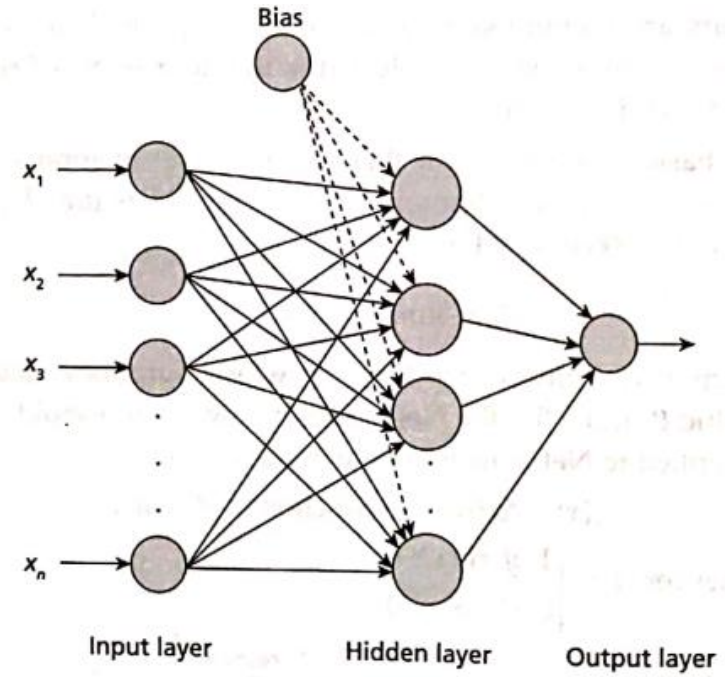


Figure 10.4: Artificial Neural Network Structure

Artificial Neural Network Structure

- The edges connecting the nodes from the input layer to the hidden layer are associated with synaptic weights called as connection weights.
- These computing nodes or neurons perform some computations based on the input information (x1, x2 .... , x,) received and if the weighted sum of the inputs to a neuron is above the threshold or the activation level of the neuron, then the neuron fires.
- Each neuron employs an activation function that determines the output of the neuron.
- The neuron transforms linearly the input signals by computing the sum of the product of input signals and weights and adds biases to it. Then, the activation function maps the weighted input sum to a non-linear output value. The node in the output layer gives the output as a single value.

Activation Functions

- Activation functions are mathematical functions associated with each neuron in the neural network that map input signals to output signals.
- It decides whether to fire a neuron or not based on the input signals the neuron receives.
- These functions normalize the output value of each neuron either between 0 and 1 or between -1 and +1.
- Typical activation functions can be linear or non-linear.

Some activation function used in ANN

- Identity Function or Linear Function $f(x) = x \ \forall x$
  - The value of f(x) increases linearly or proportionally with the value of x. This function is useful when we do not want to apply any threshold. The output would be just the weighted sum of input values. The output value ranges between -o and +00.
- Binary Step Function $f(x) = \begin{cases} 1 & if\ f(x) \geq \theta \\ 0 & if\ f(x) < \theta \end{cases}$
  - The output value is binary, i.e., 0 or 1 based on the threshold value 0. If value of f(x)is greater than or equal to 0, it outputs 1 or else it outputs 0.
- Bipolar Step Function $f(x) = \begin{cases} 1 & if\ f(x) \geq \theta \\ -1 & if\ f(x) < \theta \end{cases}$
  - The output value is bipolar, i.e., +1 or -1 based on the threshold value 0. If value off(x) is greater than or equal to 0, it outputs +1 or else it outputs-1.
- Sigmoidal Function or Logistic Function $\sigma(x) = \frac{1}{1 + e^{-x}}$
  - It is a widely used non-linear activation function which produces an S-shaped curve and the output values are in the range of 0 and 1. It has a vanishing gradient problem, i.e., no change in the prediction for very low input values and very high input values.

Some activation function used in ANN

- Bipolar Sigmoid Function $\sigma(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$
  - It outputs values between -1 and +1.
- Ramp Functions $f(x) = \begin{cases} 1 & if\ x > 1 \\ x & if\ 0 \le x \le 1 \\ 0 & if\ x < 0 \end{cases}$
  - It is a linear function whose upper and lower limits are fixed.
- Tanh - Hyperbolic Tangent Function $\tan h(x) = \frac{2}{1 + e^{-2x}} - 1$
  - The Tanh function is a scaled version of the sigmoid function which is also non-linear. It also suffers from the vanishing gradient problem. The output values range between-1 and 1.

Some activation function used in ANN

- ReLu - Rectified Linear Unit Function $r(x) = \max(0, x) = \begin{cases} x & if\ x \geq 0 \\ 0 & if\ x < 0 \end{cases}$
  - This activation function is a typical function generally used in deep learning neural network models in the hidden layers. It avoids or reduces the vanishing gradient problem. This function outputs a value of hidden layers. It avoids or reduces the vanishing gradient problem. This function outputs a value of 0 for negative input values and works like a linear function if the input is positive
- Softmax Function $s(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{k} e^{x_j}}$ where $i = 0 \ldots k$
  - This is a non-linear function used in the output layer that can handle multiple classes. It calculates the probability of each target class which ranges between 0 and 1. The probability of the input belonging to a particular class is computed by dividing the exponential of the given input value by the sum of the exponential values of all the inputs.

PERCEPTRON AND LEARNING THEORY

- The first neural network model 'Perceptron', designed by Frank Rosenblatt in 1958, is a linear binary classifier used for supervised learning.
- He modified the McCulloch & Pitts Neuron model by combining two concepts, McCulloch-Pitts model of an artificial neuron and Hebbian learning rule of adjusting weights.
- He introduced variable weight values and an extra input that represents bias to this model.
- He proposed that artificial neurons could actually learn weights and thresholds from data and came up with a supervised learning algorithm that enabled the artificial neurons to learn the correct weights from training data by itself.

PERCEPTRON AND LEARNING THEORY

- The perceptron model (shown in Figure 10.5)consists of 4 steps:
  - Inputs from other neurons
  - Weights and bias
  - Net sum
  - Activation function

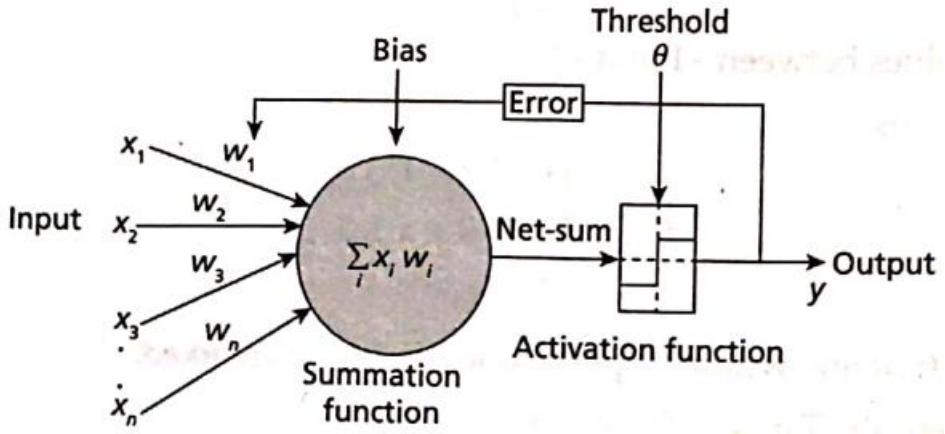


Figure 10.5: Perceptron Model

- Thus, the modified neuron model receives a set of inputs x,x2, ... ,xn, their associated weights w1,w2, ... ,w, and a bias. The summation function 'Net-sum' Eq. (10.13) computes the weighted sum of the inputs received by the neuron.

$$\text{Net-sum} = \sum_{i=1}^{n} x_i w_i$$

PERCEPTRON AND LEARNING THEORY

- After computing the 'Net-sum', bias value is added to it and inserted in the activation function

  f(x) = Activation function (Net-sum + bias)

- The activation function is a binary step function which outputs a value 1 if f(x) is above the threshold value 6, and a 0 if f(x) is below the threshold value 0. Then, output of a neuron:

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

- Before learning how a neural network works, let us learn about how a perceptron model works.

PERCEPTRON AND LEARNING THEORY

Algorithm 10.1: Perceptron Algorithm

Set initial weights $w_1, w_2, \ldots, w_n$ and bias θ to a random value in the range [–0.5, 0.5].

For each Epoch,

1. Compute the weighted sum by multiplying the inputs with the weights and add the products.
2. Apply the activation function on the weighted sum:

$$Y = \text{Step}((x_1 w_1 + x_2 w_2) - \theta)$$

3. If the sum is above the threshold value, output the value as positive else output the value as negative.
4. Calculate the error by subtracting the estimated output $Y_{estimated}$ from the desired output $Y_{desired}$:

$$\text{error } e(t) = Y_{desired} - Y_{estimated}$$

[If error $e(t)$ is positive, increase the perceptron output Y and if it is negative, decrease the perceptron output Y.]

5. Update the weights if there is an error:

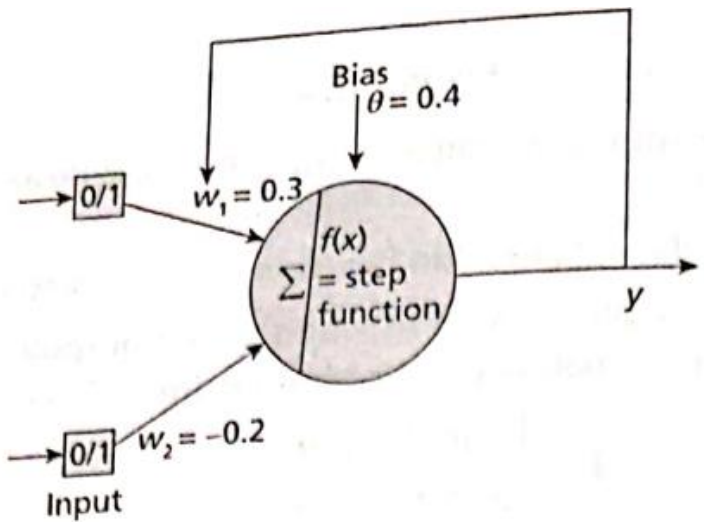
$$\Delta w_i = \propto \times e(t) \times x_i$$

$$w_i = w_i + \Delta w_i$$

where, x_i is the input value, $e(t)$ is the error at step t, $\propto$ is the learning rate and Δw_i is the difference in weight that has to be added to w_i.

Example

Example 10.1: Consider a perceptron to represent the Boolean function AND with the initial weights $w_1 = 0.3$, $w_2 = -0.2$, learning rate $\propto = 0.2$ and bias $\theta = 0.4$ as shown in Figure 10.6. The activation function used here is the Step function $f(x)$ which gives the output value as binary, i.e., 0 or 1. If value of $f(x)$ is greater than or equal to 0, it outputs 1 or else it outputs 0. Design a perceptron that performs the Boolean function AND and update the weights until the Boolean function gives the desired output.



Example - solution

- Desired output for Boolean function AND is shown in Table
- For each Epoch, weighted sum is calculated and the activation function is applied to compute the estimated output Y_{est}.
- Then, Y_{est} is compared with Y_{des} to find the error. If there is an error, the weights are updated.

Table 10.1: AND Truth Table

x_1	x_2	Y_{des}
0	0	0
0	1	0
1	0	0
1	1	1

Table 10.2: Epoch 1

Epoch	x_1	x_2	Y_{des}	Y_{est}	Error	w_1	w_2	Status
1	0	0	0	Step $((0 \times 0.3 + 0 \times -0.2) - 0.4) = 0$	0	0.3	−0.2	No change
	0	1	0	Step $((0 \times 0.3 + 1 \times -0.2) - 0.4) = 0$	0	0.3	−0.2	No change
	1	0	0	Step $((1 \times 0.3 + 0 \times -0.2) - 0.4) = 0$	0	0.3	−0.2	No change
	1	1	1	Step $((1 \times 0.3 + 1 \times -0.2) - 0.4) = 0$	1	0.5	0	Change

For input (1, 1) the weights are updated as follows:

$$\Delta w_1 = \propto \times e(t) \times x_1 = 0.2 \times 1 \times 1 = 0.2$$

$$w_1 = w_1 + \Delta w_1 = 0.3 + \Delta w_1 = 0.3 + 0.2 = 0.5$$

$$\Delta w_2 = \propto \times e(t) \times x_2 = 0.2 \times 1 \times 1 = 0.2$$

$$w_2 = w_2 + \Delta w_1 = -0.2 + \Delta w_2 = -0.2 + 0.2 = 0$$

Example - solution

Table 10.3: Epoch 2

Epoch	x_1	x_2	Y_{des}	Y_{est}	Error	w_1	w_2	Status
2	0	0	0	Step $((0 \times 0.5 + 0 \times 0) - 0.4) = 0$	0	0.5	0	No change
	0	1	0	Step $((0 \times 0.5 + 1 \times 0) - 0.4) = 0$	0	0.5	0	No change
	1	0	0	Step $((1 \times 0.5 + 0 \times 0) - 0.4) = 1$	–1	0.3	0	Change
	1	1	1	Step $((1 \times 0.3 + 1 \times 0) - 0.4) = 0$	1	0.5	0.2	Change

For input (1, 0) the weights are updated as follows:

$$\Delta w_1 = \propto \times e(t) \times x_1 = 0.2 \times -1 \times 1 = -0.2$$

$$w_1 = w_1 + \Delta w_1 = 0.5 + \Delta w_1 = 0.5 - 0.2 = 0.3$$

$$\Delta w_2 = \propto \times e(t) \times x_2 = 0.2 \times -1 \times 0 = 0$$

$$w_2 = w_2 + \Delta w_2 = 0 + \Delta w_2 = 0 + 0 = 0$$

For input (1, 1), the weights are updated as follows:

$$\Delta w_1 = \propto \times e(t) \times x_1 = 0.2 \times 1 \times 1 = 0.2$$

$$w_1 = w_1 + \Delta w_1 = 0.3 + \Delta w_1 = 0.3 + 0.2 = 0.5$$

$$\Delta w_2 = \propto \times e(t) \times x_2 = 0.2 \times 1 \times 1 = 0.2$$

$$w_2 = w_2 + \Delta w_2 = 0 + \Delta w_2 = 0 + 0.2 = 0.2$$

Example - solution

Table 10.4: Epoch 3

Epoch	x_1	x_2	Y_{des}	Y_{est}	Error	w_1	w_2	Status
3	0	0	0	Step $((0 \times 0.5 + 0 \times 0.2) - 0.4) = 0$	0	0.5	0.2	No change
	0	1	0	Step $((0 \times 0.5 + 1 \times 0.2) - 0.4) = 0$	0	0.5	0.2	No change
	1	0	0	Step $((1 \times 0.5 + 0 \times 0.2) - 0.4) = 1$	–1	0.3	0.2	Change
	1	1	1	Step $((1 \times 0.3 + 1 \times 0.2) - 0.4) = 1$	0	0.3	0.2	No change

For input (1, 0) the weights are updated as follows:

$\Delta w_1 = \propto \times e(t) \times x_1 = 0.2 \times -1 \times 1 = -0.2$

$w_1 = w_1 + \Delta w_1 = 0.5 + \Delta w_1 = 0.5 - 0.2 = 0.3$

$\Delta w_2 = \propto \times e(t) \times x_2 = 0.2 \times -1 \times 0 = 0$

$w_2 = w_2 + \Delta w_2 = 0 + \Delta w_2 = 0.2 + 0 = 0.2$

Table 10.5: Epoch 4

Epoch	x_1	x_2	Y_{des}	Y_{est}	Error	w_1	w_2	Status
4	0	0	0	Step $((0 \times 0.3 + 0 \times 0.2) - 0.4) = 0$	0	0.3	0.2	No change
	0	1	0	Step $((0 \times 0.3 + 1 \times 0.2) - 0.4) = 0$	0	0.3	0.2	No change
	1	0	0	Step $((1 \times 0.3 + 0 \times 0.2) - 0.4) = 0$	0	0.3	0.2	No change
	1	1	1	Step $((1 \times 0.3 + 1 \times 0.2) - 0.4) = 1$	0	0.3	0.2	No change

It is observed that with 4 Epochs, the perceptron learns and the weights are updated to 0.3 and 0.2 with which the perceptron gives the desired output of a Boolean AND function.

TYPES OF ARTIFICIAL NEURAL NETWORKS

- ANNs consist of multiple neurons arranged in layers.
- There are different types of ANNs that differ by the network structure, activation function involved and the learning rules used.
- In an ANN, there are three layers called input layer, hidden layer and output layer.
- Any general ANN would consist of one input layer, one output layer and zero or more hidden layers.

TYPES OF ARTIFICIAL NEURAL NETWORKS

- Feed Forward Neural Network
  - This is the simplest neural network that consists of neurons which are arranged in layers and the information is propagated only in the forward direction.
  - This model may or may not contain hidden layer and there is no back propagation.
  - Based on the number of hidden layers they are further classified into single-layered and multi-layered feed forward networks.
  - These ANNs are simple to design and easy to maintain. They are fast but cannot be used for complex learning.
  - They are used for simple classification and simple image processing, etc.
  - The model of a Feed Forward Neural Network is shown in Figure 10.7.

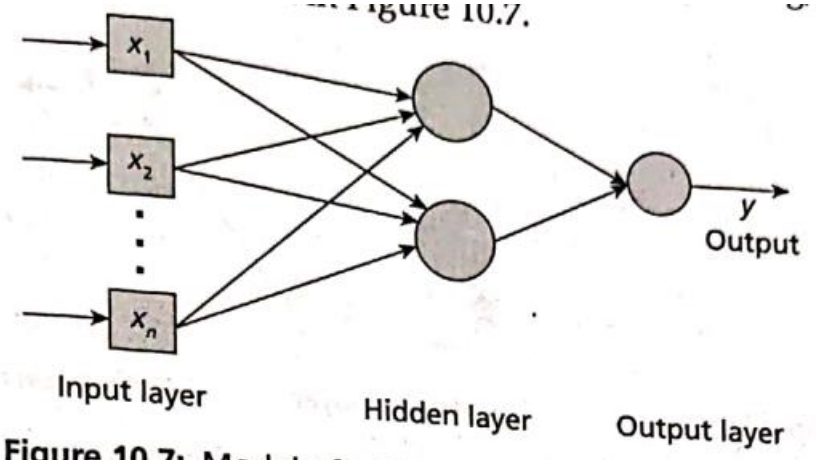


Figure 10.7: Model of a Feed Forward Neural Network

TYPES OF ARTIFICIAL NEURAL NETWORKS

- Fully Connected Neural Network
  - Fully connected neural networks are the ones in which all the neurons in a layer are connected to all other neurons in the next layer. The model of a fully connected neural network is shown in Figure 10.8.

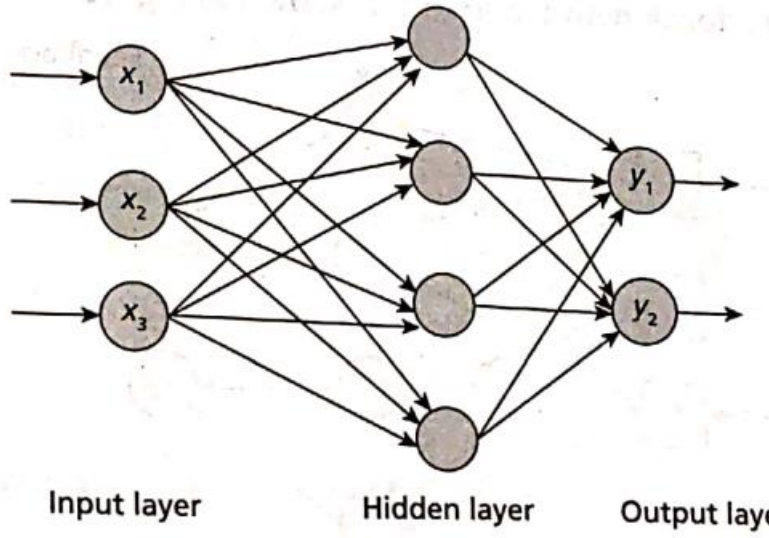


Figure 10.8: Model of a Fully Connected Neural Network

TYPES OF ARTIFICIAL NEURAL NETWORKS

- Multi-Layer Perceptron (MLP)
  - This ANN consists of multiple layers with one input layer, one output layer and one or more hidden layers. Every neuron in a layer is connected to all neurons in the next layer and thus they are fully connected. The information flows in both the directions.
  - In the forward direction, the inputs are multiplied by weights of neurons and forwarded to the activation function of the neuron and output is passed to the next layer.

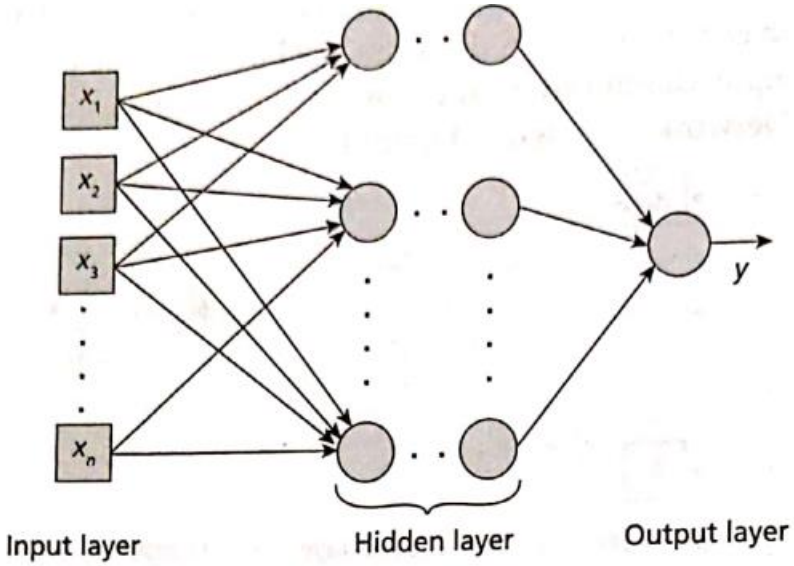


Figure 10.9: Model of a Multi-Layer Perceptron

If the output is incorrect, then in the backward direction, error is back propagated to adjust the weights and biases to get correct output. Thus, the network learns with the training data.

This type of ANN is used in deep learning for complex classification, speech recognition, medical diagnosis, forecasting.

TYPES OF ARTIFICIAL NEURAL NETWORKS

- Feedback Neural Network
  - Feedback neural networks have feedback connections between neurons that allow information flow in both directions in the network.
  - The output signals can be sent back to the neurons in the same layer or to the neurons in the preceding layers. Hence, this network is more dynamic during training. The model of a feedback neural network is shown in Figure 10.10.

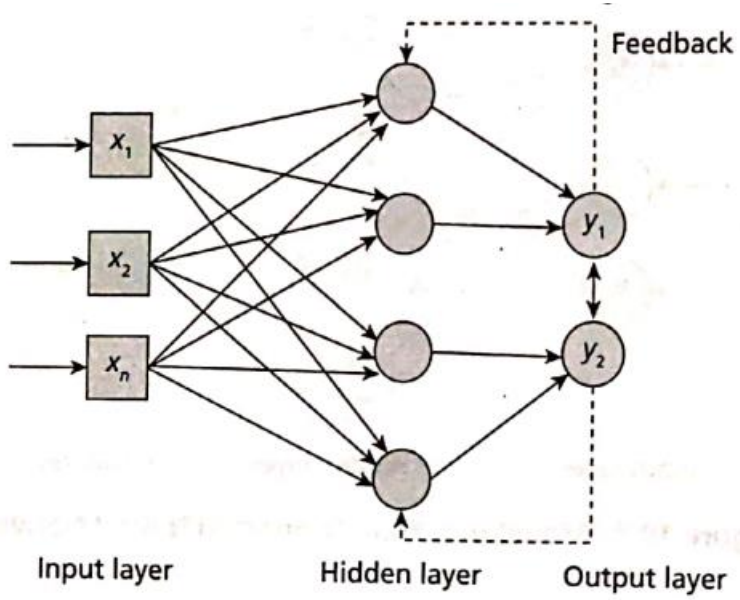


Figure 10.10: Model of a Feedback Neural Network

LEARNING IN A MULTI-LAYER PERCEPTRON

- A multi-layer perceptron is a type of Feed Forward Neural Network with multiple neurons arranged in layers.
- All the neurons in a layer are fully connected to the neurons in the next layer.
- The network has at least three layers with an input layer, one or more hidden layers and an output layer
- The input layer is the visible layer.
  - It just passes the input to the next layer.
- The layers following the input layers are hidden layers
  - The hidden layers neither directly receive input nor send outputs to the external environment.
- The final layer is the output layer which outputs a single value or a vector of values.
- An MLP has at least one hidden layer and as the number of hidden layers is increased, the learning becomes more complex and they form a deep neural network.

LEARNING IN A MULTI-LAYER PERCEPTRON

- It uses back propagation for supervised learning of the network.
- The activation functions used in the layers can be linear or non-linear depending on the type of the problem modelled.
- Typically, a sigmoid activation function is used if the problem is a binary classification problem and a softmax activation function is used in a multi-class classification problem.
- The MLP network learns with two phases called the forward phase and the backward phase.
  - In the forward phase, an input vector from the training dataset is taken and given to the network which outputs a value called estimated value $O_{Estimated}$
  - This value is compared with the desired output value $O_{Desired}$ and the error is calculated.
  - The calculated error is back propagated in the backward phase to update the weights and biases of the network.
  - This process is repeated for the entire training dataset.

MLP Algorithm

Algorithm 10.2: Learning in an MLP

Input: Input vector $(x_1, x_2, ..., x_n)$

Output: Y_n

Learning rate: α

Assign random weights and biases for every connection in the network in the range [−0.5, +0.5].

Step 1: Forward Propagation

1. **Calculate Input and Output in the *Input Layer*:**

   (Input layer is a direct transfer function, where the output of the node equals the input).

   Input at Node j 'I_j' in the Input Layer is

$$\boxed{I_j = x_j}$$

MLP Algorithm

where,

x_j is the input received at Node j

Output at Node j 'O_j' in the Input Layer is

$$O_j = I_j$$

2. Calculate Net Input and Output in the *Hidden Layer* and *Output Layer*:

*Net Input at Node j in the **Hidden Layer** is*

$$I_i = \sum_{i=1}^{n} x_i w_{ij} + x_0 \times \theta_j$$

where,

x_i is the input from Node i

w_{ij} is the weight in the link from Node i to Node j

x_0 is the input to bias node '0' which is always assumed as 1

θ_j is the weight in the link from the bias node '0' to Node j

*Net Input at Node j in the **Output Layer** is*

$$I_j = \sum_{i=1}^{n} O_i w_{ij} + x_0 \times \theta_j$$

where,

O_i is the output from Node i

w_{ij} is the weight in the link from Node i to Node j

x_0 is the input to bias node '0' which is always assumed as 1

θ_j is the weight in the link from the bias node '0' to Node j

MLP Algorithm

Output at Node j

$$O_j = \frac{1}{1 + e^{-Ij}}$$

where,

I_j is the input received at Node j

3. **Estimate error at the node in the *Output Layer*:**

$$\text{Error} = O_{Desired} - O_{Estimated}$$

where,

$O_{Desired}$ is the desired output value of the Node in the Output Layer

$O_{Estimated}$ is the estimated output value of the Node in the Output Layer

Step 2: Backward Propagation

1. **Calculate Error at each node:**

*For each Unit k in the **Output Layer***

$$\text{Error}_k = O_k(1 - O_k)(O_{Desired} - O_k)$$

MLP Algorithm

where,

O_k is the output value at Node k in the Output Layer.

$O_{Desired}$ is the desired output value of the Node in the Output Layer.

For each unit j in the ***Hidden Layer***

$$\text{Error}_j = O_j (1 - O_j) \sum_k \text{Error}_k \, w_{jk}$$

where,

O_j is the output value at Node j in the Hidden Layer.

Error_k is the error at Node k in the Output Layer.

w_{jk} is the weight in the link from Node j to Node k.

2. Update all weights and biases:

Update weights

$$\Delta w_{ij} = \propto \times \text{Error}_j \times O_i$$
$$w_{ij} = w_{ij} + \Delta w_{ij}$$

where,

O_i is the output value at Node i.

Error_j is the error at Node j.

$\propto$ is the learning rate.

w_{ij} is the weight in the link from Node i to Node j.

Δw_{ij} is the difference in weight that has to be added to w_{ij}.

Update Biases

$$\Delta \theta_j = \propto \times \text{Error}_j$$
$$\theta_j = \theta_j + \Delta \theta_j$$

where,

Error_j is the error at Node j.

$\propto$ is the learning rate.

θ_j is the bias value from Bias Node 0 to Node j.

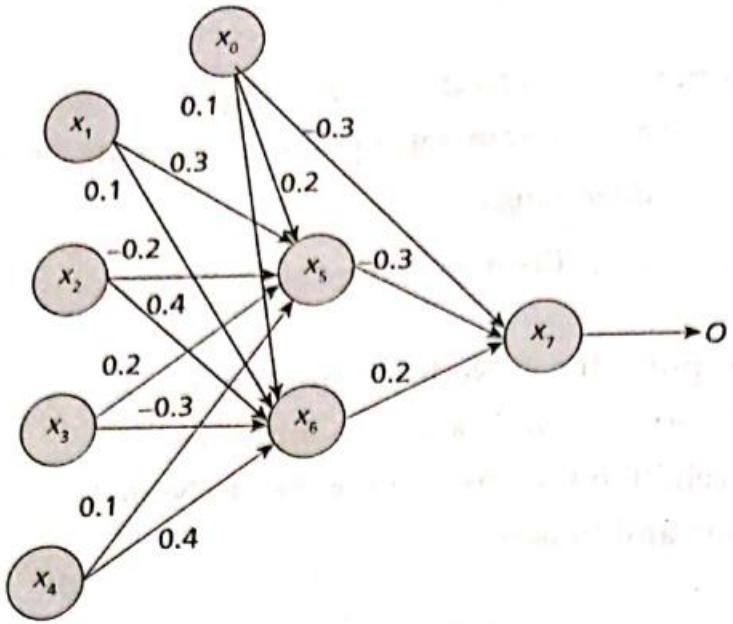
$\Delta \theta_j$ is the difference in bias that has to be added to θ_j.

MLP Example

Example 10.2: Consider learning in a Multi-Layer Perceptron. The given MLP consists of an Input layer, one Hidden layer and an Output layer. The input layer has 4 neurons, the hidden layer has 2 neurons and the output layer has a single neuron. Train the MLP by updating the weights and biases in the network.

x_1	x_2	x_3	x_4	$O_{Desired}$
1	1	0	1	1

Learning rate: = 0.8.



Solution

Solution: From the Figure 10.11, the weights and biases are tabulated in Table 10.7.

Table 10.7: Weights and Biases

x_1	x_2	x_3	x_4	w_{15}	w_{16}	w_{25}	w_{26}	w_{35}	w_{36}	w_{45}	w_{46}	w_{57}	w_{67}	θ_5	θ_6	θ_7
1	1	0	1	0.3	0.1	–0.2	0.4	0.2	–0.3	0.1	0.4	–0.3	0.2	0.2	0.1	–0.3

Step 1: Forward Propagation

1. *Calculate* Input and Output in the Input Layer shown in Table 10.8.

Table 10.8: Net Input and Output Calculation

Input layer	I_i	O_i
x_1	1	1
x_2	1	1
x_3	0	0
x_4	1	1

Solution

2. Calculate Net Input and Output in the Hidden Layer and Output Layer as shown in Table 10.9.

Table 10.9: Unit j at Hidden Layer and Output Layer – Net Input and Output Calculation

Unit j	Net Input I_j	Output O_j
x_5	$I_5 = x_1 \times w_{15} + x_2 \times w_{25} + x_3 \times w_{35} + x_4 \times w_{45} + x_0 \times \theta_5$ $I_5 = 1 \times 0.3 + 1 \times -0.2 + 0 \times 0.2 + 1 \times 0.1 + 1 \times 0.2 = 0.4$	$O_5 = \frac{1}{1+e^{-I_5}} \quad \frac{1}{1+e^{-0.4}} = 0.599$
x_6	$I_6 = x_1 \times w_{15} + x_2 \times w_{26} + x_3 \times w_{36} + x_4 \times w_{46} + x_0 \times \theta_6$ $I_5 = 1 \times 0.3 + 1 \times 0.4 + 0 \times -0.3 + 1 \times 0.4 + 1 \times 0.1 = 1.2$	$O_6 = \frac{1}{1+e^{-I_6}} \quad \frac{1}{1+e^{-1.2}} = 0.769$
x_7	$I_7 = O_5 \times w_{57} + O_6 \times w_{67} + x_0 \times \theta_7$ $I_7 = 0.599 \times -0.3 + 0.769 \times 0.2 + 1 \times -0.3 = -0.326$	$O_7 = \frac{1}{1+e^{-I_7}} \quad \frac{1}{1+e^{0.326}} = 0.419$

3. Calculate Error $= O_{desired} - O_{Estimated}$

So, error for this network is:

Error $= O_{desired} - O_7 = 1 - 0.419 = 0.581$

So, we need to back propagate to reduce the error.

Step 2: Backward Propagation

1. Calculate Error at each node as shown in Table 10.10.

For each unit k in the output layer, calculate:

$\text{Error}_k = O_k(1 - O_k)(O_{desired} - O_k)$

For each unit j in the hidden layer, calculate:

$\text{Error}_j = O_j(1 - O_j) \sum_k \text{Error}_k \, w_{jk}$

Solution

reduce the error.

Step 2: Backward Propagation

1. Calculate Error at each node as shown in Table 10.10.

   For each unit k in the output layer, calculate:

   $\text{Error}_k = O_k(1 - O_k)\,(O_{desired} - O_k)$

   For each unit j in the hidden layer, calculate:

   $\text{Error}_j = O_j\,(1 - O_j)\,\Sigma_k\,\text{Error}_k\,w_{jk}$

Table 10.10: Error Calculation for Each Unit in the Output Layer and Hidden Layer

For Output Layer Unit_k	Error_k
x_7	$\text{Error}_7 = O_7(1 - O_7)\,(Y_n - O_7)$ $= 0.419 \times (1 - 0.419) \times (1 - 0.419) = 0.141$
For Hidden Layer Unit_j	**Error_j**
x_6	$\text{Error}_6 = O_6(1 - O_6)\,\Sigma_k\,\text{Error}_k\,w_{jk} = O_6(1 - O_6)\,\text{Error}_7\,w_{67}$ $= 0.769\,(1 - 0.769) \times 0.2 \times 0.141 = 0.005$
x_5	$\text{Error}_5 = O_5(1 - O_5)\,\Sigma_k\,\text{Error}_k\,w_{jk} = O_5(1 - O_5)\,\text{Error}_7\,w_{57}$ $= 0.599(1 - 0.599) \times 0.141 \times -0.3 = -0.0101$

Solution

2. Update weight using the below formula:

Learning rate $\alpha = 0.8$.

$\Delta w_{ij} = \propto \times \text{Error}_j \times O_i$

$w_{ij} = w_{ij} + \Delta w_{ij}$

The updated weights and bias are shown in Tables 10.11 and 10.12, respectively.

Table 10.11: Weight Updation

w_{ij}	$w_{ij} = w_{ij} + \propto \times \text{Error}_j \times O_i$	New Weight
w_{15}	$w_{15} = w_{15} + 0.8 \times \text{Error}_5 \times O_1$ $= 0.3 + 0.8 \times -0.0101 \times 1$	0.292
w_{16}	$w_{16} = w_{16} + 0.8 \times \text{Error}_6 \times O_1$ $= 0.1 + 0.8 \times 0.005 \times 1$	0.104
w_{25}	$w_{25} = w_{25} + 0.8 \times \text{Error}_5 \times O_2$ $= -0.2 + 0.8 \times -0.0101 \times 1$	−0.208
w_{26}	$w_{26} = w_{26} + 0.8 \times \text{Error}_6 \times O_2$ $= 0.4 + 0.8 \times 0.005 \times 1$	0.404
w_{35}	$w_{35} = w_{35} + 0.8 \times \text{Error}_5 \times O_3$ $= 0.2 + 0.8 \times -0.0101 \times 0$	0.2
w_{36}	$w_{36} = w_{36} + 0.8 \times \text{Error}_6 \times O_3$ $= -0.3 + 0.8 \times 0.005 \times 0$	−0.3
w_{45}	$w_{45} = w_{45} + 0.8 \times \text{Error}_5 \times O_4$ $= 0.1 + 0.8 \times -0.0101 \times 1$	0.092
w_{46}	$w_{46} = w_{46} + 0.8 \times \text{Error}_6 \times O_4$ $= 0.4 + 0.8 \times 0.005 \times 1$	0.404
w_{57}	$w_{57} = w_{57} + 0.8 \times \text{Error}_7 \times O_5$ $= -0.3 + 0.8 \times 0.141 \times 0.599$	−0.232
w_{67}	$w_{67} = w_{67} + 0.8 \times \text{Error}_7 \times O_6$ $= 0.2 + 0.8 \times 0.141 \times 0.769$	0.287

(Continued)

Solution

Update bias using the below formula:

$$\Delta\theta_j = \propto \times \text{Error}_j$$

$$\theta_j = \theta_j + \Delta\theta_j$$

Table 10.12: Bias Updation

θ_j	$\theta_j = \theta_j + \propto \times \text{Error}_j$	New Bias
θ_5	$\theta_5 = \theta_5 + \propto \times \text{Error}_5$ $= 0.2 + 0.8 \times -0.0101$	0.192
θ_6	$\theta_6 = \theta_6 + \propto \times \text{Error}_6$ $= 0.1 + 0.8 \times 0.005$	0.104
θ_7	$\theta_7 = \theta_7 + \propto \times \text{Error}_7$ $= -0.3 + 0.8 \times 0.141$	−0.187

Iteration 2

Now, with the updated weights and biases:

1. Calculate Input and Output in the Input Layer as shown in Table 10.13.

Table 10.13: Net Input and Output Calculation

Input Layer	I_j	O_j
x_1	1	1
x_2	1	1
x_3	0	0
x_4	1	1

2. Calculate Net Input and Output in the Hidden Layer and Output Layer as shown in Table 10.14.

Solution

Table 10.14: Net Input and Output Calculation in the Hidden Layer and Output Layer

Unit j	Net Input I_j	Output O_j
x_5	$I_5 = x_1 \times w_{15} + x_2 \times w_{25} + x_3 \times w_{35} + x_4 \times w_{45} + x_0 \times \theta_5$ $I_5 = 1 \times 0.292 + 1 \times -0.208 + 0 \times 0.2 + 1 \times 0.092 + 1 \times 0.192 = 0.368$	$O_5 = \frac{1}{1+e^{-I_5}} = \frac{1}{1+e^{-0.368}} = 0.591$
x_6	$I_6 = x_1 \times w_{15} + x_2 \times w_{26} + x_3 \times w_{36} + x_4 \times w_{46} + x_0 \times \theta_6$ $I_6 = 1 \times 0.292 + 1 \times 0.404 + 0 \times -0.3 + 1 \times 0.404 + 1 \times 0.104 = 1.204$	$O_6 = \frac{1}{1+e^{-I_6}} = \frac{1}{1+e^{-1.204}} = 0.7692$
x_7	$I_7 = O_5 \times w_{57} + O_6 \times w_{67} + x_0 \times \theta_7$ $I_7 = 0.591 \times -0.232 + 0.7692 \times 0.287 + 1 \times -0.187 = -0.326$	$O_7 = \frac{1}{1+e^{-I_7}} = \frac{1}{1+e^{0.1034}} = 0.474$

The output we receive in the network at node 7 is 0.474.

Error = 1 – 0.474 = 0.526

Now, when we compare the error we get in the previous iteration and in the current iteration, it is visible that the network has learnt and reduced the error by 0.055.

Error is reduced by 0.055: 0.581 – 0.526.

Thus, the training is continued for a predefined number of epochs or until the training error is reduced below a threshold value.

APPLICATIONS OF ARTIFICIAL NEURAL NETWORKS

- Real-time applications: Face recognition, emotion detection, self-driving cars, navigation systems, routing systems, target tracking, vehicle scheduling, etc.
- Business applications: Stock trading, sales forecasting, customer behaviour modelling, Market research and analysis, etc.
- Banking and Finance: Credit and loan forecasting, fraud and risk evaluation, currency price prediction, real-estate appraisal, etc.
- Education: Adaptive learning software, student performance modelling, etc.
- Healthcare: Medical diagnosis or mapping symptoms to a medical case, image interpretation and pattern recognition, drug discovery, etc.
- Other Engineering Applications: Robotics, aerospace, electronics, manufacturing, communications, chemical analysis, food research, etc.

ADVANTAGES OF ANN

- ANN can solve complex problems involving non-linear processes.
- ANNs can learn and recognize complex patterns and solve problems as humans solve a problem.
- ANNs have a parallel processing capability and can predict in less time.
- They have an ability to work with inadequate knowledge. It can even handle incomplete and noisy data.
- They can scale well to larger data sets and outperforms other learning mechanisms.

DISADVANTAGES OF ANN

- An ANN requires processors with parallel processing capability to train the network running for many epochs.
  - The function of each node requires a CPU capability which is difficult for very large networks with a large amount of data.
- They work like a 'black box' and it is exceedingly difficult to understand their working in inner layers.
  - Moreover, it is hard to understand the relationship between the representations learned at each layer.

Clustering Algorithms

Chapter 13

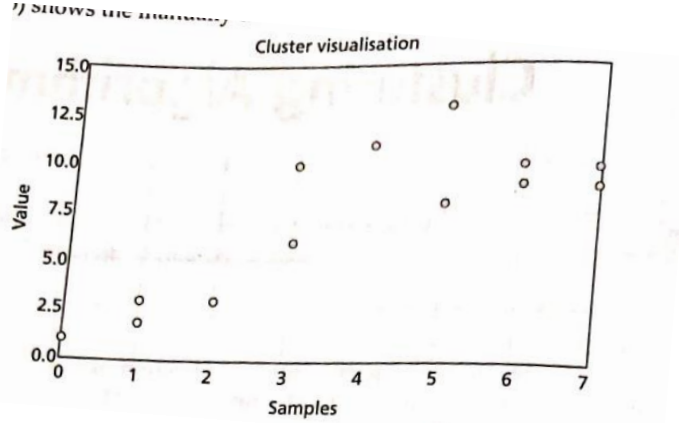
Module 5

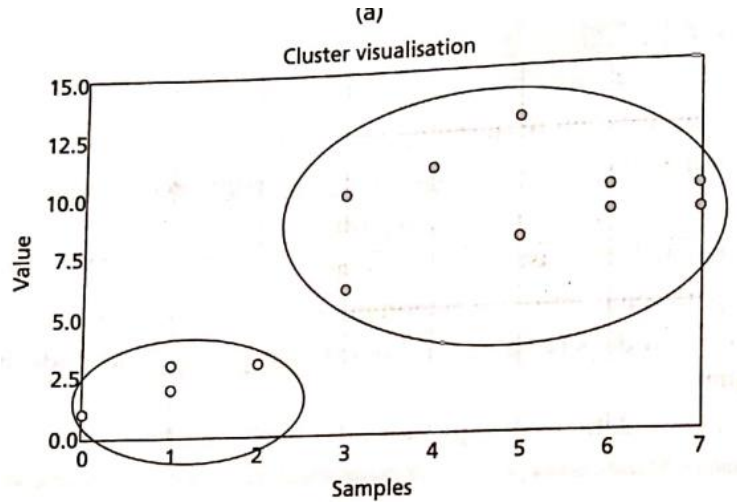
INTRODUCTION TO CLUSTERING APPROACHES

- Cluster analysis is the fundamental task of unsupervised learning.
- Unsupervised learning involves exploring the given dataset. Cluster analysis is a technique of partitioning a collection of unlabelled objects that have many attributes into meaningful disjoint groups or clusters.
- This is done using a trial and error approach as there are no supervisors available as in classification.
- The characteristic of clustering is that the objects in the clusters or groups are similar to each other within the clusters while differ from the objects in other clusters significantly.

INTRODUCTION TO CLUSTERING APPROACHES

- The input for cluster analysis is examples or samples.
- These are known as objects, datapoints or data instances.
- All these terms are same and used interchangeably in this chapter. All the samples or objects with no labels associated with them are called unlabelled.
- The output is the set of clusters (or groups) of similar data if it exists in the input.
  - For example, the following Figure 13.1(a) shows data points or samples with two features shown in different shaded samples and Figure 13.1(b) shows the manually drawn ellipse to indicate the clusters formed.





INTRODUCTION TO CLUSTERING APPROACHES

- Visual identification of clusters in this case is easy as the examples have only two features.
- But, when examples have more features, say 100, then clustering cannot be done manually and automatic clustering algorithms are required. Also, automating the clustering process is desirable as these tasks are considered difficult by humans and almost impossible. All clusters are represented by centroids.
- Input examples or data is (3, 3), (2, 6) and (7, 9), then the centroid is given by $\left(\frac{3+2+7, 3+6+9}{3}\right) = (4, 6).$

Introduction to clustering approaches

- The clusters should not overlap and every cluster should represent only one class.
- Therefore, clustering algorithms use trial and error method to form clusters that can be converted to labels.
- Thus, the important differences between classification and clustering are given in Table 13.1.

Clustering	Classification
Unsupervised learning and cluster formation are done by trial and error as there is no supervisor	Supervised learning with the presence of a supervisor to provide training and testing data
Unlabelled data	Labelled data
No prior knowledge in clustering	Knowledge of the domain is must to label the samples of the dataset
Cluster results are dynamic	Once a label is assigned, it does not change

Applications of Clustering

- Grouping based on customer buying patterns
- Profiling of customers based on lifestyle
- In information retrieval applications (like retrieval of a document from a collection of documents)
- Identifying the groups of genes that influence a disease
- Identification of organs that are similar in physiology functions
- Taxonomy of animals, plants in Biology
- Clustering based on purchasing behavior and demography
- Document indexing
- Data compression by grouping similar objects and finding duplicate objects

Challenges of Clustering Algorithms

- A huge collection of data with higher dimensions (i.e., features or attributes) can pose a problem for clustering algorithms. With the arrival of the internet, billions of data are available for clustering algorithms.
- This is a difficult task, as scaling is always an issue with clustering algorithms. Scaling is an issue where some algorithms work with lower dimension data but do not perform well for higher dimension data.
- Also, units of data can post a problem, like some weights in kg and some in pounds can pose a problem in clustering. Designing a proximity measure is also a big challenge.

Advantages and Disadvantages

Advantages	Disadvantages
Cluster analysis algorithms can handle missing data and outliers.	Cluster analysis algorithms are sensitive to initialization and order of the input data.
Can help classifiers in labelling the unlabelled data. Semi-supervised algorithms use cluster analysis algorithms to label the unlabelled data and then use classifiers to classify them.	Often, the number of clusters present in the data have to be specified by the user.
It is easy to explain the cluster analysis algorithms and to implement them.	Scaling is a problem.
Clustering is the oldest technique in statistics and it is easy to explain. It is also relatively easy to implement.	Designing a proximity measure for the given data is an issue.

PROXIMITY MEASURES

- Clustering algorithms need a measure to find the similarity or dissimilarity among the objects to group them.
- Similarity and dissimilarity are collectively known as proximity measures.
- Often, the distance measures are used to find similarity between two objects, say i and j.
- Distance measures are known as dissimilarity measures, as these indicate how one object is different from another.
- Measures like cosine similarity indicate the similarity among objects.

PROXIMITY MEASURES

- Distance between two objects, say i and j, is denoted by the symbol D_{ij}.
- The properties of the distance measures are:
- 1. D_{ij} is always positive or zero.
- 2. D_{ii} =0, i.e., the distance between the object to itself is 0.
- 3. $D_{ij} = D_{ji}$. This property is called symmetry.
- 4. $D_{ij} \leq D_{ik} + D_{kj}$. This property is called triangular inequality.
- If all these conditions are satisfied, then the distance measure is called a metric.

Quantitative variables

- **Euclidean Distance:** It is one of the most important and common distance measures. It is also called as L_2 norm.
- It can be defined as the square root of squared differences between the coordinates of a pair of objects.
- The Euclidean distance between objects x_i, and x_j, with k features is given as follows:

$$\text{Distance}(x_i, x_j) = \sqrt{\sum_{k=1}^{n}(x_{ik} - x_{jk})^2}$$

- The advantage of Euclidean distance is that the distance does not change with the addition of new objects. But the disadvantage is that if the units change, the resulting Euclidean or squared Euclidean changes drastically.
- Another disadvantage is that as the Euclidean distance involves a square root and a square, the computational complexity is high for implementing the distance for millions or billions of operations involved.

Quantitative variables

- **City Block Distance:** City block distance is known as Manhattan distance. This is also known as boxcar, absolute value distance, Manhattan distance, Taxicab or L_1 norm. The formula for finding the distance is given as follows:

$$\text{Distance}(x_i, x_j) = \sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right|$$

- **Chebyshev Distance:** Chebyshev distance is known as maximum value distance. This is the absolute magnitude of the differences between the coordinates of a pair of objects. This distance is called supremum distance or L_{max} or L_{norm}. The formula for computing Chebyshev distance is given as follows:

$$\text{Distance}(x_i, x_j) = \max_{k} \left| x_{ik} - x_{jk} \right|$$

Example

Example 13.1: Suppose, if the coordinates of the objects are (0, 3) and (5, 8), then what is the Chebyshev distance?

Solution: The Euclidean distance using Eq. (13.1) is given as follows:

$$\text{Distance}(x_i, x_j) = \sqrt{(0-5)^2 + (3-8)^2}$$

$$= \sqrt{50} = 7.07$$

The Manhattan distance using Eq. (13.2) is given as follows:

$$\text{Distance}(x_i, x_j) = \left|(0-5) + (3-8)\right| = 10$$

The Chebyshev distance using Eq. (13.3) is given as follows:

$$\text{Max}\left\{\left|0-5\right|, \left|3-8\right|\right\} = \text{Max}\{5, 5\} = 5$$

Quantitative variables

- **Minkowski Distance** In general, all the above distance measures can be generalized as:

$$\text{Distance}(x_i, x_j) = \left(\sum \left| x_{ik} - x_{jk} \right|^r \right)^{\frac{1}{r}}$$

- This is called Minkowski distance. Here, r is a parameter.
- When the value of r is 1, the distance measure is called city block distance.
- When the value of r is 2, the distance measure is called Euclidean distance.
- When r is ∞, then this is Chebyshev distance.

Binary Attributes

- Binary attributes have only two values. Distance measures discussed above cannot be applied to find distance between objects that have binary attributes.
- For finding the distance among objects with binary objects, the contingency Table 13.3 can be used. Let x and y be the objects consisting of N-binary objects.
- Then, the contingency table can be constructed by counting the number of matching of transitions, 0-0, 0-1, 1-0 and 1-1.

Attributes Matching	0	1
0	a	b
1	c	d

- In other words, 'a' is the number of attributes where x attribute is 0 and y attribute is 0, 'b' is the number of attributes where x attribute is 0 and y attribute is 1, 'c' is the number of attributes where x attribute is 1 and y attribute is 0 and 'd' is the number of attributes where x attribute is 1 and y attribute is 1.

Binary Attributes

- **Simple Matching Coefficient (SMC)** SMC is a simple distance measure and is defined as the ratio of number of matching attributes and the number of attributes. The formula is given as:

$$\frac{a + d}{a + b + c + d}$$

- **Jaccard Coefficient** Jaccard coefficient is another useful measure for and is given as follows:

$$\frac{d}{b + c + d}$$

Example

Example 13.2: If the given vectors are $x = (1, 0, 0)$ and $y = (1, 1, 1)$ then find the SMC and Jaccard coefficient?

Solution: It can be seen from Table 13.2 that, $a = 0$, $b = 2$, $c = 0$ and $d = 1$.

The SMC using Eq. (13.5) is given as $\frac{a+d}{a+b+c+d} = 0+1/3 = 0.33$

Jaccard coefficient using Eq. (13.6) is given as $J = \frac{d}{b+c+d} = 1/3 = 0.33$

Binary Attributes

- **Hamming Distance** Hamming distance is another useful measure that can be used for knowing the sequence of characters or binary values. It indicates the number of positions at which the characters or binary bits are different.
- For example, the hamming distance between x = (1 0 1) and y = (1 1 0) is 2 as x and y differ in two positions. The distance between two words, say wood and hood is 1, as they differ in only one character.
- Sometimes, more complex distance measures like edit distance can also be used.
- **Categorical Variables** In many cases, categorical values are used. It is just a code or symbol to represent the values. For example, for the attribute Gender, a code 1 can be given to female and 0 can be given to male. To calculate the distance between two objects represented by variables, we need to find only whether they are equal or not. This is given as:

$$\text{Distance}(x, y) = \begin{cases} 1 & \text{if } x \neq y \\ 0 & \text{if } x = y \end{cases}$$

Binary Attributes

- **Ordinal Variables** are like categorical values but with an inherent order. For example, designation is an ordinal variable. If job designation is 1 or 2 or 3, it means code 1 is higher than 2 and code 2 is higher than 3. It is ranked as 1 > 2 > 3.
- Let us assume the designations of office employees are clerk, supervisor, manager and general manager. These can be designated as numbers as clerk = 1, supervisor = 2, manager = 3 and general manager = 4. Then, the distance between employee X who is a clerk and Y who is a manager can be obtained as:

$$\text{Distance}(X, Y) = \frac{|position(X) - position(Y)|}{n-1}$$

- Here, position (X) and position(Y) indicate the designated numerical value. Thus, the distance between X (Clerk =1) and Y (Manager =3) using Eq. (13.8) is given as:

$$\text{Distance}(X, Y) = \frac{|position(X) - position(Y)|}{n-1} = \frac{|1-3|}{4-1} = \frac{2}{3} \approx 0.66$$

Binary Attributes

- **Vector Type Distance Measures** For text classification, vectors are normally used. Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Cosine similarity measures the cosine of the angle between two vectors projected in a multi-dimensional space. The similarity function for vector objects can be defined as:

$$sim\,(X,Y) = \frac{X \bullet Y}{\|X\| \times \|Y\|} = \frac{\sum_{i=1}^{n} X_i \times Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2} \times \sqrt{\sum_{i=1}^{n} Y_i^2}}$$

- The numeration is the dot product of the vectors A and B. The denominator is the product of the norm of vectors A and B.

Example

Example 13.3: If the given vectors are $A = \{1, 1, 0\}$ and $B = \{0, 1, 1\}$, then what is the cosine similarity?

Solution: The dot product of the vector is $1 \times 0 + 1 \times 1 + 0 \times 1 = 1$. The norm of the vectors A and B is $\sqrt{2}$.

So, the cosine similarity using Eq. (13.9) is given as $\frac{1}{\sqrt{2}\sqrt{2}} = \frac{1}{2} = 0.5$

HIERARCHICAL CLUSTERING ALGORITHMS

- Hierarchical methods produce a nested partition of objects with hierarchical relationships among objects. Often, the hierarchy relationship is shown in the form of a dendrogram.
- Hierarchical methods include categories, agglomerative methods and divisive methods.
- In agglomerative methods, initially all individual samples are considered as a cluster, that is a cluster with a single element. Then, they are merged and the process is continued to get a single cluster.
- Divisive methods use another kind of philosophy, where a single cluster of all samples of the dataset taken initially is chosen and then partitioned. This partition process is continued until the cluster is split into smaller clusters.

HIERARCHICAL CLUSTERING ALGORITHMS

- Agglomerative methods merge clusters to reduce the number of clusters. This is repeated each time while merging two closest clusters to get a single cluster. The procedure of agglomerative clustering is given as follows:

Algorithm 13.1: Agglomerative Clustering

1. Place each N sample or data instance into a separate cluster. So, initially N clusters are available.
2. Repeat the following steps until a single cluster is formed:
   (a) Determine two most similar clusters.
   (b) Merge the two clusters into a single cluster reducing the number of clusters as N–1.
3. Choose resultant clusters of step 2 as result.

- All the clusters that are produced by hierarchical algorithms have equal diameters. The main disadvantage of this approach is that once the cluster is formed, it is an irreversible decision.

Single Linkage or MIN Algorithm

- In single linkage algorithm, the smallest distance (x, y), where x is from one cluster and y is from another cluster, is the distance between all possible pairs of the two groups or clusters (or simply the smallest distance of two points where points are in different clusters) and is used for merging the clusters.
- This corresponds to finding of minimum spanning tree (MST) of a graph. The distance measures between individual samples or data points is already demonstrated in the previous sections.

Example

- Consider the array of points as shown in the following Table.

Objects	X	Y
0	1	4
1	2	8
2	5	10
3	12	18
4	14	28

- Apply Single linkage algorithm

Example

- Euclidean distance is computed as shown in the following Table

Proximity Matrix

Objects	0	1	2	3	4
0	-	4.123	7.211	17.804	27.295
1		-	3.606	14.142	23.324
2			-	10.630	20.124
3				-	10.198
4					-

- The minimum distance is 3.606. Therefore items 1 and 2 are clustered together.

Example

- After clustering the resultant distances are as shown: After iteration 1

Clusters	{1,2}	0	3	4
{1,2}	-	4.123	10.630	20.124
0		-	17.804	27.295
3			-	10.198
4				-

- $D_{SL}(C_i, C_j)$ = minimum a $\in C_i$,b $\in C_j$ d(a, b)
- Thus, the distance between {1, 2} and {0} is: Minimum {{1, 0}, {2, 0}} = Minimum {4.123, 7.211} = 4.123
- The distance between {1, 2} and {3} is given as: Minimum {{1, 3), {2, 3}} = Minimum {14.412, 10.630} = 10.630
- The distance between {1, 2} and {4} is given as: Minimum {{1, 4), {2, 4]} = Minimum {23.324, 20.124) = 20.124

Example

- After clustering the resultant distances are as shown: After iteration 2

Clusters	{0,1,2}	3	4
{0,1,2}	-	10.630	20.124
3		-	10.198
4			-

- The minimum distance is 4.123. Therefore items 0, 1 and 2 are clustered together.
- After clustering the resultant distances are as shown: After iteration 3

Clusters	{0,1,2}	{3,4}
{0,1,2}	-	?
{3,4}	-	-

- The computation of ? in above Table is not needed as no other items are left. Therefore, the clusters {0, 1, 2} and {3, 4} are merged.

Example

Dendrograms are used to plot the hierarchy of clusters.
Dendrogram for the above clustering is shown in the following Figure.