multi-task learning, end-to-end deep learning, Introduction to CNN models : LeNet - 5, AlexNet, VGG - 16, Residual Networks. **(Chapter - 5)**

Unit VI Applications of ANN

Pattern classification - Recognition of Olympic games symbols, Recognition of printed Characters. Neocognitron - Recognition of handwritten characters. NET Talk : to convert English text to speech. Recognition of consonant vowel (CV) segments, texture classification and segmentation. **(Chapter - 6)**

TABLE OF CONTENTS

Unit I

Chapter - 1 Introduction to ANN (1 - 1) to (1 - 34)

1.1 Introduction to ANN ..... 1 - 2

1.2 History of Neural Network ..... 1 - 3

1.3 Structure and Working of Biological Neural Network ..... 1 - 3

1.4 Introduction to Neural Network ..... 1 - 5

1.4.1 Advantages of Neural Network ..... 1 - 7

1.4.2 Application of Neural Network ..... 1 - 7

1.4.3 Difference between Digital Computer and Neural Networks ..... 1 - 8

1.5 Neural Net Architecture ..... 1 - 8

1.5.1 Single Layer Feed Forward Network ..... 1 - 8

1.5.2 Multi-Layer Feed Forward Network ..... 1 - 12

1.5.3 Recurrent Neural Network ..... 1 - 17

1.6 Activation Functions ..... 1 - 18

1.6.1 Identity or Linear Activation Function ..... 1 - 20

1.6.2 Sigmoid ..... 1 - 21

1.6.3 Tanh and ReLU ..... 1 - 22

1.7 Models of Neuron-McCulloch and Pitts Model ..... 1 - 23

1.8 Perceptron ..... 1 - 25

1.8.1 Single Layer Perceptron ..... 1 - 25

1.8.2 Multilayer Perceptron ..... 1 - 27

1.8.3 ADALINE Network ..... 1 - 29

1.9 Basic Learning Laws ..... 1 - 31

1.9.1 Weights and Bias ..... 1 - 31

Unit II

Chapter - 2 Learning Algorithms **(2 - 1) to (2 - 26)**

2.1 Learning and Memory ..... 2 - 2
2.1.1 Auto-associative Memory ..... 2 - 3
2.1.2 Hetero-associative Memory Network ..... 2 - 4
2.1.3 The Hopfield Network ..... 2 - 5
2.1.4 Bidirectional Associative Memory (BAM) ..... 2 - 6
2.1.5 Difference between Auto-associative Memory and Hetero-Associative Memory ..... 2 - 7
2.2 Hebbian Learning ..... 2 - 7
2.3 Competitive Learning ..... 2 - 10
2.4 Error Correction ..... 2 - 11
2.5 Gradient Decent Rules ..... 2 - 12
2.5.1 Finding the Optimal Hyper-Parameters through Grid Search ..... 2 - 13
2.5.2 Vanishing Gradient Problem ..... 2 - 15
2.6 Supervised Learning ..... 2 - 16
2.6.1 Advantages and Disadvantages of Supervised Learning ..... 2 - 19
2.6.2 Difference between Supervised and Unsupervised Learning ..... 2 - 19
2.7 Backpropagation ..... 2 - 20
2.7.1 Advantages and Disadvantages ..... 2 - 23
2.8 Feed Forward Neural Networks ..... 2 - 23
2.8.1 Feedback Neural Networks ..... 2 - 24
2.8.3 Example and Applications of Feedforward NN ..... 2 - 25
2.8.4 Difference between RNNs and feed-forward NN ..... 2 - 25

Unit III

Chapter - 3 Associative Learning **(3 - 1) to (3 - 58)**

3.1 Associative Learning, Hopfield Networks, Simulated Annealing, Boltzman Machine ..... 3 - 2
3.1.1 Associative Learning ..... 3 - 2
3.1.2 Hopfield Network ..... 3 - 4
3.1.2.1 Introduction ..... 3 - 4

3.1.2.2 Characteristics of Hopfield Network - Qualities and Attributes ..... 3 - 5
3.1.2.3 Activation Function and Thresholding Function ..... 3 - 6
3.1.2.4 Evolution Towards a Memorized Pattern in Hopfield Network ..... 3 - 8
3.1.2.5 Types of Hopfield Networks
(Discrete Hopfield Network and Continuous Hopfield Network) ..... 3 - 9
3.1.2.6 Energy Function ..... 3 - 11
3.1.2.7 Error Performance in Hopfield Networks ..... 3 - 11

3.2 Simulated Annealing ..... 3 - 12
3.2.1 Introduction ..... 3 - 12
3.2.2 Simulated Annealing terms ..... 3 - 14
3.2.3 Simulated Annealing Algorithm ..... 3 - 15
3.2.4 Advantages vs. Disadvantages of Simulated Annealing ..... 3 - 18

3.3 Boltzmann Machine and Boltzmann Learning ..... 3 - 18
3.3.1 Introduction ..... 3 - 18
3.3.2 Objective of Boltzmann Machine and Architecture ..... 3 - 19
3.3.3 Learning in Boltzmann Machine ..... 3 - 22
3.3.4 Uses of Boltzmann Machine ..... 3 - 23
3.3.5 Energy-Based Models ..... 3 - 23
3.3.6 Types of Boltzmann Machines
(Restricted BMs, Deep Belief Networks, Deep BMs) ..... 3 - 24
3.3.7 Limitations of Boltzmann Machines ..... 3 - 31

3.4 Basic Functional Units of ANN for Pattern Recognition Tasks ..... 3 - 32
3.4.1 Introduction to Pattern Recognition ..... 3 - 32
3.4.2 Pattern Recognition Tasks ..... 3 - 32
3.4.3 Input Processing for Pattern Recognition ..... 3 - 34
3.4.4 Functional Units of Pattern Recognition and Working of ANN ..... 3 - 36
3.4.4.1 Pattern Recognition Tasks by Feedforward ANN ..... 3 - 37
3.4.4.2 Pattern Recognition Tasks by Feedback ANN ..... 3 - 40
3.4.4.3 Pattern Recognition Tasks by Feedforward and Feedback ANN ..... 3 - 54

Review Questions with Answers ..... 3 - 57

Unit IV

Chapter - 4 Competitive Learning Neural Network (4 - 1) to (4 - 24)

4.1 Competitive Learning and Adaptive Resonance Theory (ART) ..... 4 - 2
4.1.1 Competitive Learning ..... 4 - 2
4.1.1.1 Introduction to Competitive Learning ..... 4 - 2

4.1.1.2 Components of CL network ..... 4 - 2
4.1.2 ART Networks - Working, Pattern Clustering, Feature Mapping, Advantages, Applications ..... 4 - 8
4.1.2.1 Introduction to ART Networks ..... 4 - 8
4.1.2.2 Operational Principal and Features of ART Models, ART 1 ..... 4 - 9
4.1.2.3 ART Applications Character Recognition using ART Network ..... 4 - 14
4.2 Self-Organization Maps (SOM) - Two Basic Feature Mapping Models, Self-Organization Map, SOM Algorithm, Properties of Feature Map, Computer Simulations, Learning Vector Quantization, Adaptive Pattern Classification ..... 4 - 16
4.2.1 Basics of SOM ..... 4 - 16
4.2.2 Working of SOM ..... 4 - 17
4.2.3 Feature Map ..... 4 - 19
4.2.4 Advantages and Disadvantages of Self - organizing Maps ..... 4 - 22
4.2.5 Learning Vector Quantization ..... 4 - 22
Review Questions with Answers ..... 4 - 24

Unit V

Chapter - 5 Convolution Neural Network (5 - 1) to (5 - 36)

5.1 Introduction to Convolution Neural Network ..... 5 - 2
5.1.1 Advantages and Disadvantages of CNN ..... 5 - 3
5.1.2 Application of CNN ..... 5 - 3
5.2 The Basic Structure of CNN ..... 5 - 4
5.3 Convolution Operation ..... 5 - 4
5.3.1 Parameter Sharing ..... 5 - 7
5.3.2 Equivariant Representation ..... 5 - 8
5.3.3 Padding ..... 5 - 9
5.3.4 Stride ..... 5 - 10
5.3.5 Typical Setting ..... 5 - 11
5.3.6 ReLU Layer ..... 5 - 11
5.4 Pooling ..... 5 - 11
5.5 Fully Connected Layers ..... 5 - 13
5.6 Convolutions Over Volume ..... 5 - 13
5.7 SoftMax Regression ..... 5 - 14
5.8 Deep Learning Frameworks ..... 5 - 15

5.8.1 TensorFlow ..... 5 - 17
5.8.2 Keras ..... 5 - 19
5.8.2.1 Difference Between Keras and Tensorflow ..... 5 - 20
5.8.3 PyTorch ..... 5 - 20
5.8.4 Caffe ..... 5 - 21
5.8.5 Shogun ..... 5 - 21
5.9 Bias and Variance with Mismatched Data Distributions ..... 5 - 22
5.10 Learning Representations from Data ..... 5 - 24
5.10.1 Greedy Layer - Wise Unsupervised Pretraining ..... 5 - 25
5.10.2 Transfer Learning and Domain Adaptation ..... 5 - 27
5.11 Multi-Task Learning ..... 5 - 27
5.11.1 Types of Multi-Task Learning ..... 5 - 28
5.11.2 Advantages of Multi-task Learning ..... 5 - 29
5.12 End-to-End Deep Learning ..... 5 - 29
5.13 Introduction to CNN Models ..... 5 - 30
5.13.1 LeNet - 5 ..... 5 - 30
5.13.2 AlexNet ..... 5 - 31
5.13.3 VGG - 16 ..... 5 - 33
5.13.4 Residual Networks ..... 5 - 34

Unit VI

Chapter - 6 Applications of ANN (6 - 1) to (6 - 24)

6.1 Applications Artificial Neural Network I - Pattern Classification ..... 6 - 2
6.2 Applications Artificial Neural Network II - Recognition of Printed Characters ..... 6 - 7
6.3 Applications Artificial Neural Network III - Neocognitron - Recognition of Handwritten Characters ..... 6 - 8
6.4 Applications Artificial Neural Network IV - NETtalk - Convert English Text to Speech Application ..... 6 - 18
6.5 Applications Artificial Neural Network V - Recognition of Consonant Vowel (CV) Segments, Texture Classification and Segmentation ..... 6 - 20
Review Questions with Answers ..... 6 - 23

Solved Model Question Papers (M - 1) to (M - 4)

Notes

Unit I

1 Introduction to ANN

Syllabus

Introduction to ANN, History of Neural Network, Structure and working of Biological Neural Network, Neural net architecture, Topology of neural network architecture, Features, Characteristics, Types, Activation functions, Models of neuron-Mc Culloch & Pitts model, Perceptron, Adaline model, Basic learning laws, Applications of neural networks, Comparison of BNN and ANN.

Contents

1.1 Introduction to ANN

1.2 History of Neural Network

1.3 Structure and Working of Biological Neural Network

1.4 Introduction to Neural Network

1.5 Neural Net Architecture

1.6 Activation Functions

1.7 Models of Neuron-McCulloch and Pitts Model

1.8 Perceptron

1.9 Basic Learning Laws

1.1 Introduction to ANN

- Artificial Neural Network (ANN) is a computational system inspired by the structure, processing method, learning ability of a biological brain. An artificial neural network is composed of many artificial neurons that are linked together according to specific network architecture. The objective of the neural network is to transform the inputs into meaningful outputs.
- ANNs do not execute programmed instructions; they respond in parallel to the pattern of inputs presented to it. There are also no separate memory addresses for storing data. Instead, information is contained in the overall activation 'state' of the network. 'Knowledge' is thus represented by the network itself, which is quite literally more than the sum of its individual components.
- Fig. 1.1.1 shows artificial neural network.

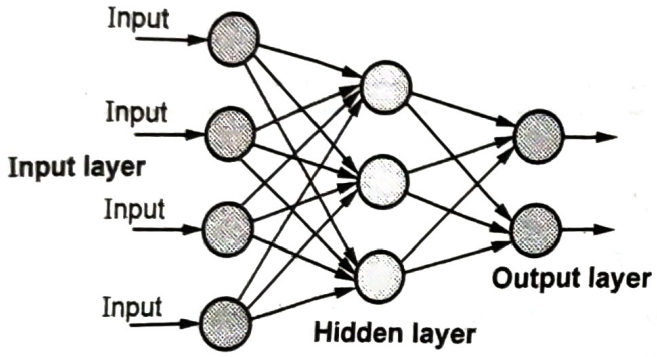


Fig. 1.1.1 Artificial neural network

- Elements of ANN are processing units, topology and learning algorithm.
- Tasks to be solved by artificial neural networks :
  1. Controlling the movements of a robot based on self-perception and other information;
  2. Deciding the category of potential food items in an artificial world;
  3. Recognizing a visual object;
  4. Predicting where a moving object goes, when a robot wants to catch it.
- Characteristics of artificial neural networks :
  1. Large number of very simple processing neuron-like processing elements.
  2. Large number of weighted connections between the elements.
  3. Distributed representation of knowledge over the connections.
  4. Knowledge is acquired by network through a learning process.

1.2 History of Neural Network

- Neural networks were first proposed in 1943 by Warren McCullough and Walter Pitts, two University of Chicago researchers who moved to MIT in 1952 as founding members of what's sometimes called the first cognitive science department.
- Pitts and McCulloch researched and wrote a paper describing how neurons might work and then developed a simple neural network using electrical circuits.
- By deploying logic gates, they were able to demonstrate how outputs could be activated when specified inputs are active, to model how neurons work in the brain. As a model of the brain, the functionality provided was considerably simpler compared to the billions of neurons in the human brain. In addition, their model was only able to process binary inputs and outputs, providing a very simplified model of a neuron.
- Frank Rosenblatt proposed the idea of a Perceptron in 1958, calling it Mark I Perceptron. It was a system with a simple input output relationship, modeled on a McCulloch-Pitts neuron. A McCulloch-Pitts neuron takes in inputs, takes a weighted sum and returns '0' if the result is below threshold and '1' otherwise.
- The beauty of Mark I Perceptron lay in the fact that its weights would be 'learnt' through successively passed inputs, while minimizing the difference between desired and actual output.
- In 1959 at Stanford, Bernard Widrow and Marcian Hoff developed the first neural network successfully applied to a real world problem. These systems were named ADALINE and MADALINE after their use of Multiple ADAptive LINear Elements.
- Backpropagation along with Gradient Descent forms the backbone and powerhouse of neural networks. While Gradient Descent constantly updates and moves the weights and bias towards the minimum of the cost function, backpropagation evaluates the gradient of the cost w.r.t. weights and biases, the magnitude and direction of which is used by gradient descent to evaluate the size and direction of the corrections to weights and bias parameters.
- By the 1980s, however, researchers had developed algorithms for modifying neural nets' weights and thresholds that were efficient enough for networks with more than one layer, removing many of the limitations identified by Minsky and Papert. The field enjoyed a renaissance.

1.3 Structure and Working of Biological Neural Network

- Artificial neural systems are inspired by biological neural systems. The elementary building block of biological neural systems is the neuron.

- The brain is a collection of about 10 billion interconnected neurons. Each neuron is a cell [right] that uses biochemical reactions to receive, process and transmit information. Fig. 1.3.1 shows biological neural systems.

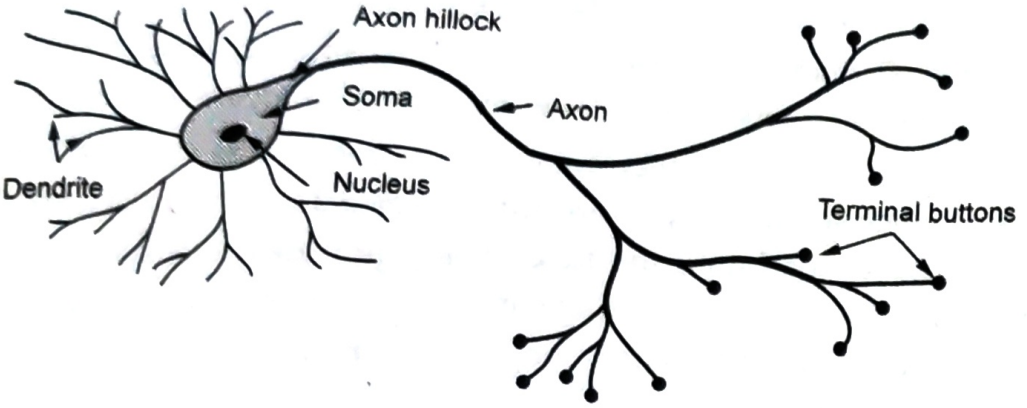


Fig. 1.3.1 Schematic of biological neuron

- The single cell neuron consists of the cell body or soma, the dendrites and the axon. The dendrites receive signals from the axons of other neurons. The small space between the axon of one neuron and the dendrite of another is the synapse. The afferent dendrites conduct impulses toward the soma. The efferent axon conducts impulses away from the soma.

Basic Components of Biological Neurons

1. The majority of ***neurons*** encode their activations or outputs as a series of brief electrical pulses (i.e. spikes or action potentials).
2. The neuron's ***cell body (soma)*** processes the incoming activations and converts them into output activations.
3. The neuron's ***nucleus*** contains the genetic material in the form of DNA. This exists in most types of cells, not just neurons.
4. ***Dendrites*** are fibres which emanate from the cell body and provide the receptive zones that receive activation from other neurons.
5. ***Axons*** are fibres acting as transmission lines that send activation to other neurons.
6. The junctions that allow signal transmission between the axons and dendrites are called ***synapses***. The process of transmission is by diffusion of chemicals called ***neurotransmitters*** across the synaptic cleft.

Comparison between Biological NN and Artificial NN

Biological NN	Artificial NN
soma	unit
Axon, dendrite	dendrite
synapse	weight
potential	weighted sum
threshold	bias weight
signal	activation

1.4 Introduction to Neural Network

- Neural networks consists of many numbers of simple elements (neurons) connected between them in system. Whole system is able to solve of complex tasks and to learn for it like a natural brain.
- Neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths and the processing performed at computing elements or nodes.
- For user, NN is black box with input vector (source data) and output vector (result).
  - A Neural network is usually structured into an input layer of neurons, one or more hidden layers one output layer.
  - Neurons belonging to adjacent layers are usually fully connected and the various types and architectures are indentified both by the different topologies adopted for the connections as well as by the choice of the activation function.
  - The values of the functions associated with the connections are called "weights".
  - The whole game of using NNs is in fact that, in order for the network to yield appropriate outputs for given inputs, the weight must be set to suitable values. The way this is obtained allows a further distinction among modes of operations.
  - A neural network is a processing device, either an algorithm or actual hardware, whose design was motivated by the design and functioning of human brains and components thereof.

- Most neural networks have some sort of "training" rule whereby the weights of connections are adjusted on the basis of presented patterns.
- In other words, neural networks "learn" from example, just like children learn to recognize dogs from examples of dogs and exhibit some structural capability for generalization.
- Neural networks normally have great potential for paralleism, since the computations of the components are independent of each other.
- Neural networks are a different paradigm for computing :
  1. Von Neumann machines are based on the processing/memory abstraction of human information processing.
  2. Neural networks are based on the parallel architecture of animal brains.

- Neural networks are a form of multiprocessor computer system, with
  a. Simple processing elements
  b. A high degree of interconnections
  c. Simple scalar messages
  d. Adaptive interaction between elements
- The advantages of neural networks are due to its adaptive and generalization ability.
  a) Neural networks are adaptive methods that can learn without any prior assumption of the underlying data.
  b) Neural network, namely the feed forward multilayer perception and radial basis function network have been proven to be universal functional approximations.
  c) Neural networks are non-linear model with good generalization ability.
- **Useful properties and capabilities of neural network**
  1. **Nonlinearity :** An artificial neuron can be linear or nonlinear. A neural network, made up of an interconnection of nonlinear neurons, is itself nonlinear.
  2. **Adaptivity :** Neural networks have a built-in capability to adapt their synaptic weights to changes in the surrounding environment.

3. **Contextual information :** Knowledge is represented by the very structured and activation state of a neural network.
4. **Evidential response :** In the context of pattern classification, 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. **Uniformity of analysis and design :** Neural networks enjoy universality as information processors.
6. **VLSI implement-ability :** The massively parallel nature of a neural network makes it potentially fast for the computation of certain tasks.

1.4.1 Advantages of Neural Network

- The advantages of neural networks are due to its adaptive and generalization ability.
  a) Neural networks are adaptive methods that can learn without any prior assumption of the underlying data.
  b) Neural network, namely the feed forward multilayer perception and radial basis function network have been proven to be universal functional approximations.
  c) Neural networks are non-linear model with good generalization ability.

1.4.2 Application of Neural Network

Neural network applications can be grouped in following categories :

1. **Clustering :** A clustering algorithm explores the similarity between patterns and places similar patterns in a cluster. Best known applications include data compression and data mining.
2. **Classification/Pattern recognition :** The task of pattern recognition is to assign an input pattern (like handwritten symbol) to one of many classes. This category includes algorithmic implementations such as associative memory.
3. **Function approximation :** The tasks of function approximation is to find an estimate of the unknown function f() subject to noise. Various engineering and scientific disciplines require function approximation.
4. **Prediction/Dynamical systems :** The task is to forecast some future values of a time-sequenced data. Prediction has a significant impact on decision support systems. Prediction differs from function approximation by considering time factor.

1.4.3 Difference between Digital Computer and Neural Networks

Sr. No.	Digital Computers	Neural Networks
1.	Deductive reasoning : We apply known rules input data to produce output.	Inductive reasoning : Given input and output data (training examples), we construct the rules.
2.	Computation is centralized, synchronous and serial.	Computation is collective asynchronous and parallel.
3.	Memory is packetted, literally stored and location addressable.	Memory is distributed, internalized and content addressable.
4.	Not fault toerant. One transistor goes and it no longer works.	Fault tolerant, redundancy and sharing of responsibilities.
5.	Fast. Measured in millionths of a second.	Slow. Measured in thousands of a second.
6.	Exact.	Inexact.
7.	Static connectivity.	Dynamic connectivity.
8.	Applicable if well defined rules with precise input data.	Applicable if rules are unknown or complicated or if data is noisy or partial.

1.5 Neural Net Architecture

1.5.1 Single Layer Feed Forward Network

- The architecture of the neural network refers to the arrangement of the connection between neurons, processing element, number of layers and the flow of signal in the neural network.
- There are mainly two category of neural network architecture :
  a. Feed-forward
  b. Feedback (recurrent) neural networks.

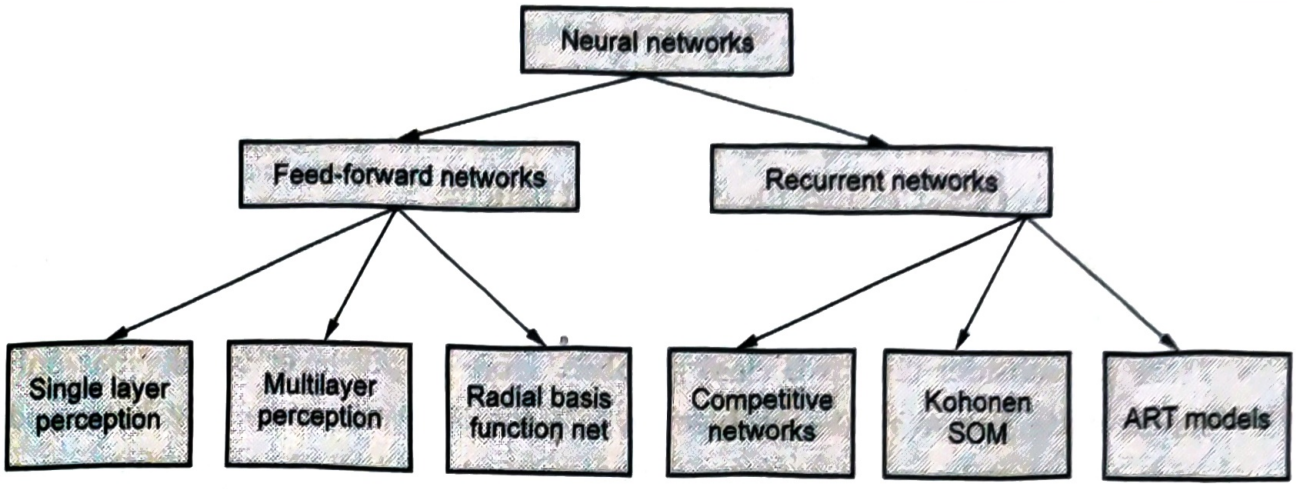


Fig. 1.5.1

1. Architecture and Learning Rule

- In late 1950s, Frank Rosenblatt introduced a network composed of the units that were enhanced version of McCulloch-Pitts Threshold Logic Unit (TLU) model.
- Rosenblatt's model of neuron, a perceptron, was the result of merger between two concepts from the 1940s, McCulloch-Pitts model of an artificial neuron and Hebbian learning rule of adjusting weights.
- In addition to the variable weight values, the perceptron model added an extra input that represents bias. Thus, the modified equation is now as follows :

$$\text{Sum} = \sum_{i=1}^{N} I_i W_i + b,$$

where **b** represents the bias value.

- Fig. 1.5.2 shows a typical perception setup for pattern recognition applications, in which visual patterns are represented as matrices of elements between 0 and 1.

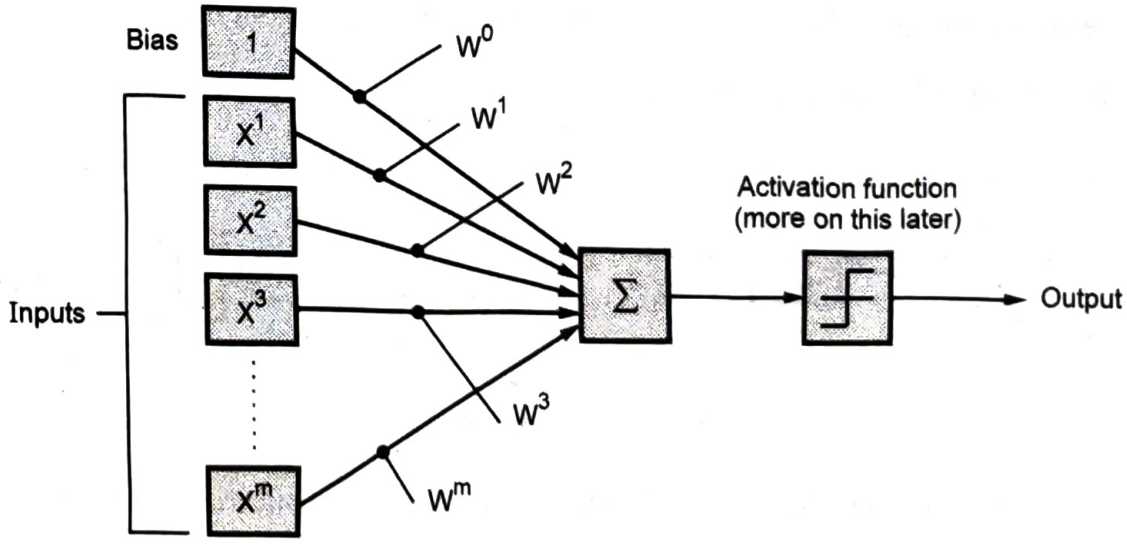


Fig. 1.5.2 Perception setup

1. First layer act as a set of feature detectors that are hardwired to the input signals to detect specific features.
2. Second layer i.e. output layer takes the outputs of the feature detectors in the first layer and classifies the given input pattern.

- Learning is initiated by making adjustments to the relevant connection strengths and a threshold value.
- Here we consider only two class problem. Here output layer usually has only a single node. For an n-class problem (n > 3), the output layer usually has n-nodes, each corresponding to a class and the output node with the largest value indicates which class the input vector belongs to.
- In the first stage, the linear combination of inputs is calculated. Each value of input array is associated with its weight value, which is normally between 0 and 1. Also, the summation function often takes an extra input value Theta with weight value of 1 to represent threshold or bias of a neuron.
  a. The term x_i is referred to as **active or excitatory** if its value is 1.
  b. If the value is 0 then it is **inactive**.
  c. If the value is – 1 then it is **inhibitory**.
- The output unit is a linear threshold element with a threshold value :

$$0 = f\left(\sum_{i=1}^{n} w_i x_i - \theta\right)$$

$$= f\left(\sum_{i=1}^{n} w_i x_i - w_0\right), \; w_0 \equiv -\theta$$

$$= f\left(\sum_{i=1}^{n} w_i x_i\right), \; x_0 \equiv 1$$

where w_i is a modifiable weight associated with an incoming signal x_i.

- Fig. 1.5.3 shows the bias term w_0.

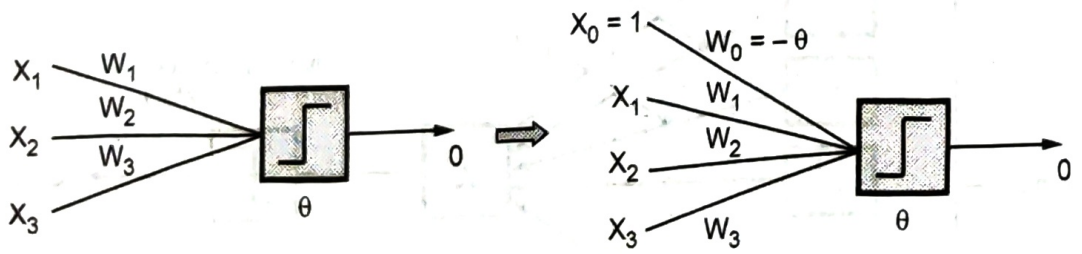


Fig. 1.5.3 Bias term w_0

- The function y = f(x) describes relationship, an input-output mapping from x to y.

- The equation (1.5.1), the f(.) is the activation function of the perceptron and it is typically either a signum function sgn(x) or step function step(x) :

$$\text{sgn}(x) = \begin{cases} 1 & \text{if } x > 0, \\ -1 & \text{otherwise} \end{cases}$$

$$\text{step}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{otherwise} \end{cases} \qquad \ldots (1.5.1)$$

- The sum-of-product value is then passed into the second stage to perform the activation function which generates the output from the neuron. The activation function "squashes" the amplitude of the output in the range of [0, 1] or [– 1, 1] alternately. The behavior of the activation function will describe the characteristics of an artificial neuron model.
- The basic learning algorithm for a single layer perceptron repeats the following steps until the weights converge :
  1. Select an input vector x from the training data set.
  2. If the perceptron gives an incorrect response, modify all connection weights w_i according to

$$\Delta w_i = \eta t_i X_i$$

Where t_i is a target output and η is a learning state.

Perceptron Convergence Theorem

Theorem : If there is a set of weights that correctly classify the (linearly separable) training patterns, then the learning algorithm will find one such weight set, w^* in a finite number of iterations.

Assumptions :

1. At least one such set of weights, w^*, exists and
2. There are a finite number of training patterns.
3. The threshold function is uni-polar (output is 0 or 1).

2. Exclusive OR problem

- XOR problem is a pattern recognition problem in neural network.
- Neural networks can be used to classify boolean functions depending on their desired outputs.
- For a two input binary XOR problem, the desired output is given in the form of truth table.

	X	Y	Class
Desired I/O pair 1	0	0	0
Desired I/O pair 2	0	1	1
Desired I/O pair 3	1	0	1
Desired I/O pair 4	1	1	0

- The XOR problem is not linearly separable. We cannot use a single layer perceptron to construct a straight line to partition the two dimensional input space into two regions, each containing only data points of the same class.
- Let us consider following four equations :

$$0 \times w_1 + 0 \times w_2 + w_0 \leq 0 \Leftrightarrow w_0 \leq 0,$$

$$0 \times w_1 + 1 \times w_2 + w_0\, 0 \Leftrightarrow w_0 > -w_2$$

$$1 \times w_1 + 0 \times w_2 + w_0 > 0 \Leftrightarrow w_0 > -w_1$$

$$1 \times w_1 + 1 \times w_2 + w_0 \leq 0 \Leftrightarrow w_0 \leq -w_1 - w_2$$

1.5.2 Multi-Layer Feed Forward Network

- A multilayer feed-forward neural network is a network consisting of multiple layers of units, all of which are adaptive. The network is not allowed to have cycles from later layers back to earlier layers, hence the name "feed-forward". Let us consider a network with a single complete hidden layer. i.e., the network consists of some input nodes, some output nodes and a set of hidden nodes. Every hidden node takes inputs from each of the input nodes and feeds into each of the output nodes.

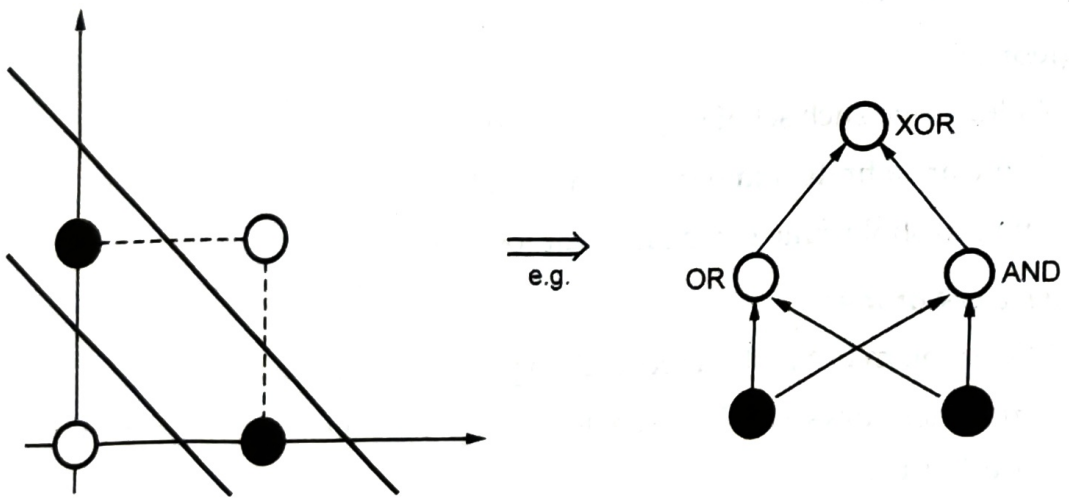


Fig. 1.5.4

- In multi-layer feed forward neural networks, the sigmoid activation function, defined by $g(x) = \frac{1}{1+e^{-x}}$ is normally used.
- A Multi-Layer Perceptron (MLP) has the same structure of a single layer perceptron with one or more hidden layers. An MLP is a network of simple neurons called perceptrons.
- A typical multilayer perceptron network consists of a set of source nodes forming the input layer, one or more hidden layers of computation nodes, and an output layer of nodes.
- It is not possible to find weights which enable single layer perceptrons to deal with non-linearly separable problems like XOR :
- Multi-layer perceptrons are able to cope with non-linearly separable problems.
- Each neuron in one layer has direct connections to all the neurons of the subsequent layer. MLP can implement nonlinear discriminants (for classification) and nonlinear regression functions (for regression).
- Historically, the problem was that there were no known learning algorithms for training MLPs. Fortunately; it is now known to be quite straightforward. The procedure for finding a gradient vector in the network structure is generally referred to as **backpropagation**. Because the gradient vector is calculated in the direction opposite to the flow of the output of each node.
- Procedure of backpropagation :
  1. The output values are compared with the target to compute the value of some predefined error function.
  2. The error is then feedback through the network.
  3. Using this information, the algorithm adjusts the weights of each connection in order to reduce the value of the error function.
- Continue this process until the connection weights in the network have been adjusted so that the network output has converged, to an acceptable level, with the desired output.
- If we use the gradient vector in a simple steepest descent method, the resulting learning paradigm is often referred to as the **backpropagation** learning rule. Backpropagation works by approximating the non-linear relationship between the input and the output by adjusting the weight values internally.

- Generally, the backpropagation network has two stages, training and testing. During the training phase, the network is "shown" sample inputs and the correct classifications. For example, the input might be an encoded picture of a face and the output could be represented by a code that corresponds to the name of the person.
- Fig. 1.5.5 shows three most commonly used activation functions in backpropagation MLPs.

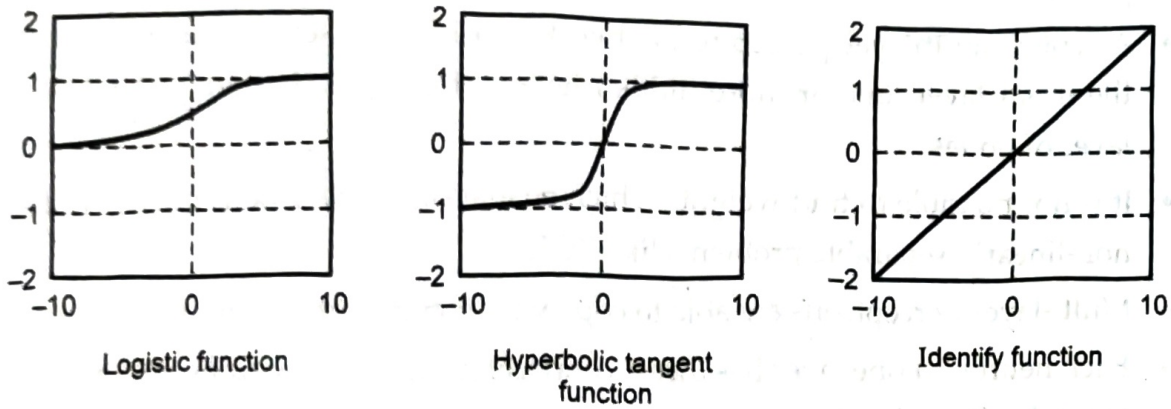


Fig. 1.5.5 Activation function

Logistic function :

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

Hyperbolic tangent function :

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

Identity function :

$$f(x) = x$$

- Both the hyperbolic tangent function and logistic function approximate the signum and step function respectively. Sometimes these two function are referred to as **squashing functions** since the inputs to these functions are squashed to the range [0, 1] or [– 1, 1].
- These functions are also called **sigmoidal functions** because their S-shaped curves exhibits smoothness and asymptotic properties.
- A learning process is organized through a learning algorithm, which is a process of updating the weights in such a way that a machine learning tool implements a given input/output mapping with no errors or with some minimal acceptable error.

- Any learning algorithm is based on a certain learning rule, which determines how the weights shall be updated if the error occurs.

Backpropagation Learning Rule

- The net input of a node is defined as the weighted sum of the incoming signals plus a bias term. Fig. 1.5.6 shows the backpropagation MLP for node j. The net input and output of node j is as follows :

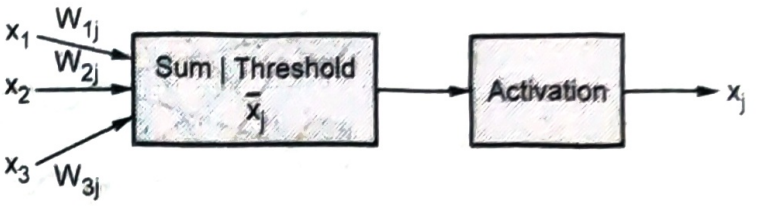


Fig. 1.5.6 Backpropagation MLP for node j

$$\bar{X}_j = \Sigma_i + W_{ij} + W_j$$

$$x_j = f(\bar{X}_j) = \frac{1}{1 + \exp(-\bar{X}_j)}$$

Where

x_i is the output of node i located in any one of the pervious layers,

W_{ij} is the weight associated with the link connecting nodes i and j,

W_j is the bias of node j.

- Internal parameters associated with each node j is the weight W_{ij}. So changing the weights of the node will change the behaviour of the whole backpropagation MLP.
- Fig. 1.5.7 shows two layer backpropagation MLP.
- The above backpropagation MLP will refer to as a 3-4-3 network, corresponding to the number of nodes in each layer.
- The backward error propagation also known as the backpropagation (BP) or the Generalized Delta Rule (GDR). A squared error measure for the pth input-output pair is defined as

$$E_p = \sum_k (d_k - x_k)^2$$

Where d_k is the desired output for node k and x_k is the actual output for node k when the input part of the pth data pair is presented.

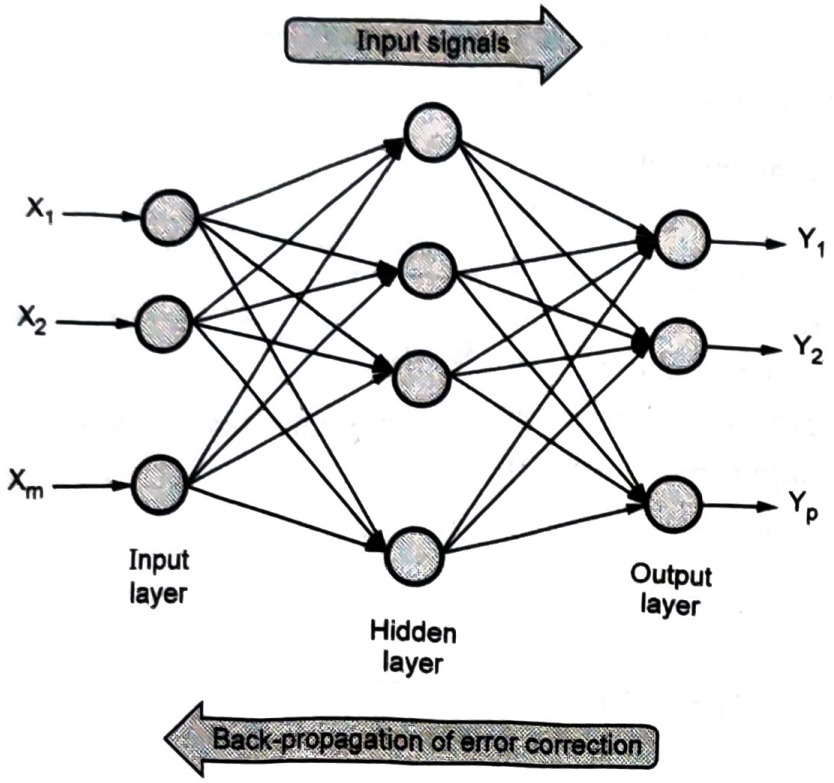


Fig. 1.5.7 Two layer backpropagation MLP

- To find the gradient vector, an error term $\bar{\epsilon}_i$ for node i is defined as :

$$\bar{\epsilon}_i = \frac{\partial + E_p}{\partial \bar{X}_i}$$

- The partial derivative can be rewritten as product of two terms using chain rule for partial differentiation :

$$\frac{\partial E(t)}{\partial W_{ij}(t)} = \frac{\partial E(t)}{\partial a_i(t)} \bullet \frac{\partial a_i(t)}{\partial w_{ij}(t)}$$

- Features of the delta rule are as follows :
  1. Simplicity
  2. Distributed learning : Learning is not reliant on central control of the network.
  3. Online learning : Weights are updated after presentation of each pattern.

Rules for Feedforward Multilayer Perceptron

- The training algorithm is called Error Back Propagation (EBP) training algorithm. If a submitted pattern provides an output far from desired value, the weights and thresholds are adjusted so that the current mean square classification error is reduced.

- The training is repeated for all patterns until the training set provide an acceptable overall error. Usually the mapping error is computed over the full training set.
- Error back propagation algorithm is working in two stages :
  1. The trained network operates feed-forward to obtain output of the network
  2. The weight adjustment propagate backward from output layer through hidden layer toward input layer.

1.5.3 Recurrent Neural Network

- A recurrent neural network is a type of neural network that contains loops, allowing information to be stored within the network.
- A RNN is particularly useful when a sequence of data is being processed to make a classification decision or regression estimate but it can also be used on non-sequential data. Recurrent neural networks are typically used to solve tasks related to time series data.
- Applications of recurrent neural networks include natural language processing, speech recognition, machine translation, character-level language modeling, image classification, image captioning, stock prediction, and financial engineering.
- Fig. 1.5.8 shows architecture of recurrent neural network.

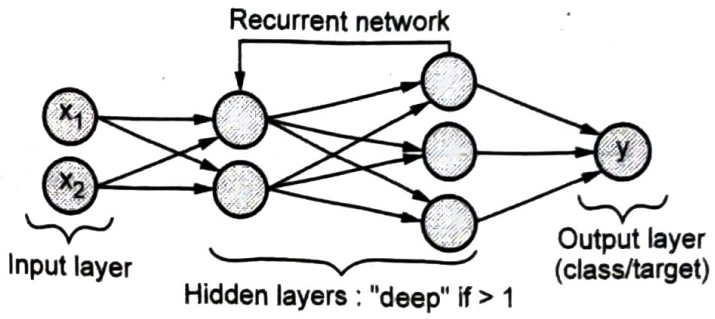


Fig. 1.5.8 Architecture of RNN

- Recurrent Neural Networks can be thought of as a series of networks linked together. They often have a chain-like architecture, making them applicable for tasks such as speech recognition, language translation, etc.
- An RNN can be designed to operate across sequences of vectors in the input, output, or both. For example, a sequenced input may take a sentence as an input and output a positive or negative sentiment value. Alternatively, a sequenced output may take an image as an input and produce a sentence as an output.

1.6 Activation Functions

- Activation functions also known as transfer function is used to map input nodes to output nodes in certain fashion.
- The activation function is the most important factor in a neural network which decided whether or not a neuron will be activated or not and transferred to the next layer.
- Activation functions help in normalizing the output between 0 to 1 or – 1 to 1. It helps in the process of backpropagation due to their differentiable property. During backpropagation, loss function gets updated and activation function helps the gradient descent curves to achieve their local minima.
- Activation function basically decides in any neural network that given input or receiving information is relevant or it is irrelevant.
- These activation function makes the multilayer network to have greater representational power than single layer network only when non-linearity is introduced.
- The input to the activation function is sum which is defined by the following equation.

$$\text{sum} = I_1W_1 + I_2W_2 + \ldots + I_nW_n = \sum_{j=1}^{n} I_j W_j + b$$

- **Activation Function : Logistic Function**

$$f(\text{sum}) = \frac{1}{(1 + e^{-s * \text{sum}})}$$

$$= (1 + e^{-s \times \text{sum}})^{-2}$$

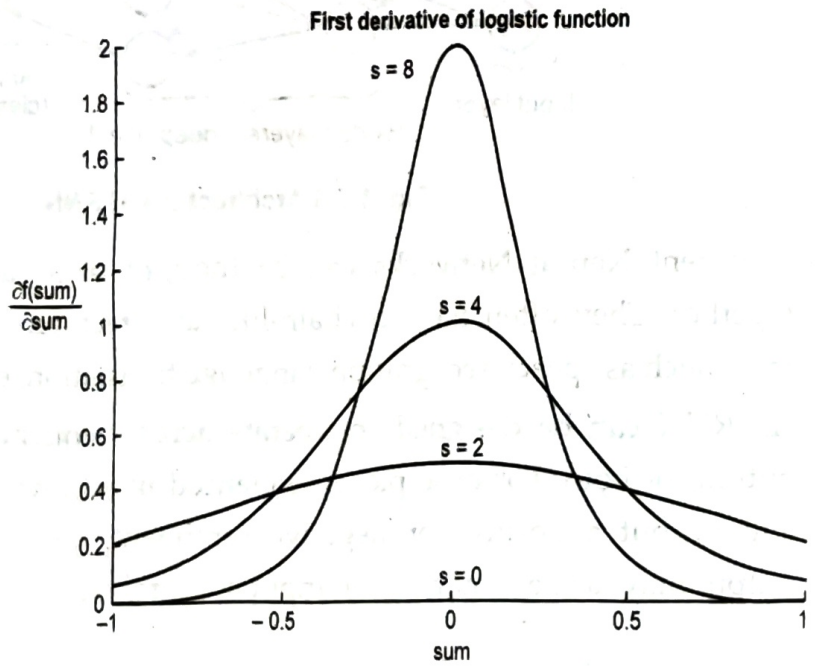


Fig. 1.6.1

- Logistic function monotonically increases from a lower limit (0 or – 1) to an upper limit (+ 1) as sum increases. In which values vary between 0 and 1, with a value of 0.5 when I is zero.

- **Activation Function : Arc Tangent**

$$f(sum) = \frac{2}{\pi} \tan^{-1} (s \times sum)$$

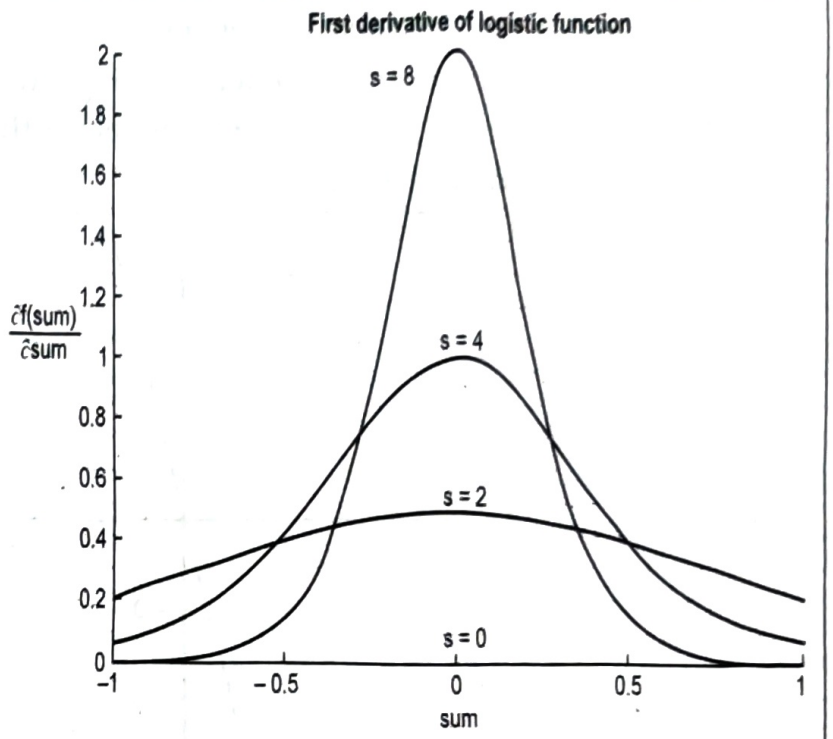


Fig. 1.6.2

- **Activation Function : Hyperbolic Tangent**

$$f(sum) = \tanh (s * I)$$

$$= \frac{e^{s \times sum} - e^{-s \times sum}}{e^{s \times sum} + e^{-s \times sum}}$$

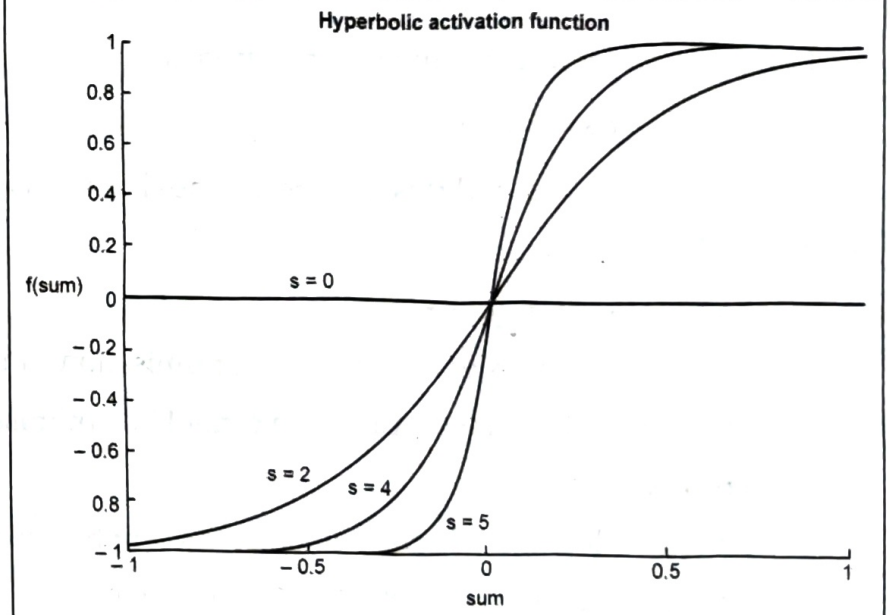


Fig. 1.6.3

1.6.1 Identity or Linear Activation Function

- A linear activation is a mathematical equation used for obtaining output vectors with specific properties.
- It is a simple straight line activation function where our function is directly proportional to the weighted sum of neurons or input.
- Linear activation functions are better in giving a wide range of activations and a line of a positive slope may increase the firing rate as the input rate increases.
- Fig. 1.6.4 shows identity function.

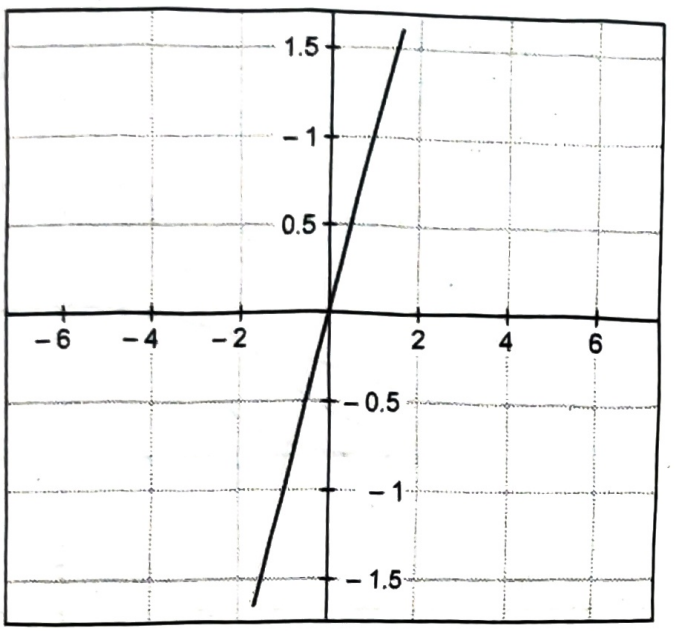


Fig. 1.6.4 Identity function

- The equation for linear activation function is :

$$f(x) = a \cdot x$$

When a = 1 then f(x) = x and this is a special case known as identity.

- **Properties :**
  1. Range is – infinity to + infinity
  2. Provides a convex error surface so optimisation can be achieved faster.
  3. df(x)/dx = a which is constant. So cannot be optimised with gradient descent.
- **Limitations :**
  1. Since the derivative is constant, the gradient has no relation with input.
  2. Back propagation is constant as the change is delta x.
  3. Activation function does not work in neural networks in practice.

1.6.2 Sigmoid

- A sigmoid function produces a curve with an "S" shape. The example sigmoid function shown on the left is a special case of the logistic function, which models the growth of some set.

$$\text{sig}(t) = \frac{1}{1+e^{-t}}$$

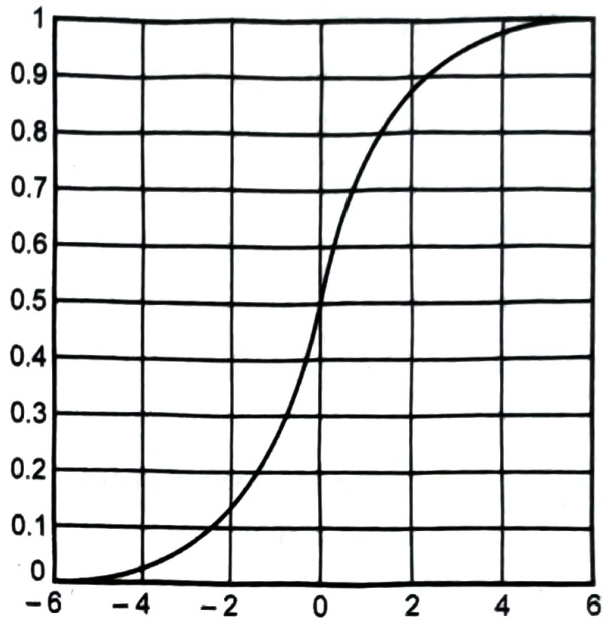


Fig. 1.6.5

- In general, a sigmoid function is real-valued and differentiable, having a non-negative or non-positive first derivative, one local minimum and one local maximum.
- The logistic sigmoid function is related to the hyperbolic tangent as follows :

$$1 - 2\text{sig}(x) = 1 - 2\frac{1}{1+e^{-x}} = -\tanh\frac{x}{2}$$

- Sigmoid functions are often used in artificial neural networks to introduce nonlinearity in the model.
- A neural network element computes a linear combination of its input signals, and applies a sigmoid function to the result.
- A reason for its popularity in neural networks is because the sigmoid function satisfies a property between the derivative and itself such that it is computationally easy to perform.

$$\frac{d}{dt}\text{sig}(t) = \text{sig}(t)(1 - \text{sig}(t))$$

- Derivatives of the sigmoid function are usually employed in learning algorithms.

1.6.3 Tanh and ReLU

- Tanh is also like logistic sigmoid but better. The range of the tanh function is from (– 1 to 1). Tanh is also sigmoidal (s - shaped).
- Fig. 1.6.6 shows tanh v/s Logistic sigmoid.

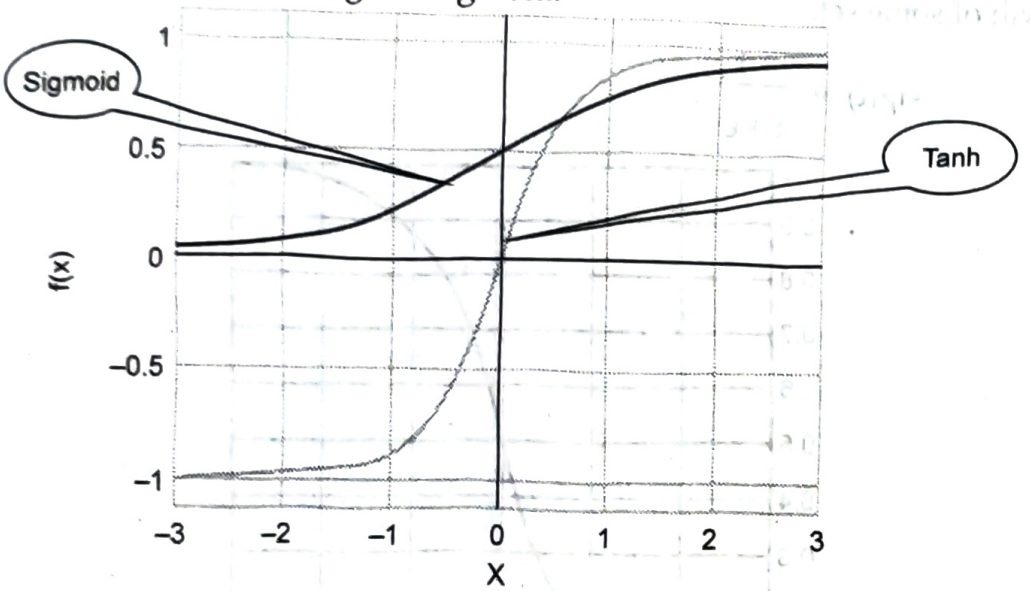


Fig. 1.6.6 tanh v/s Logistic sigmoid

- Tanh neuron is simply a scaled sigmoid neuron.
- Problems resolved by Tanh

  1. The output is not zero centered 2. Small gradient of sigmoid function

- ReLU (Rectified Linear Unit) is the most used activation function in the world right now. Since, it is used in almost all the convolution neural networks or deep learning.
- Fig. 1.6.7 shows ReLU v/s Logistic sigmoid.

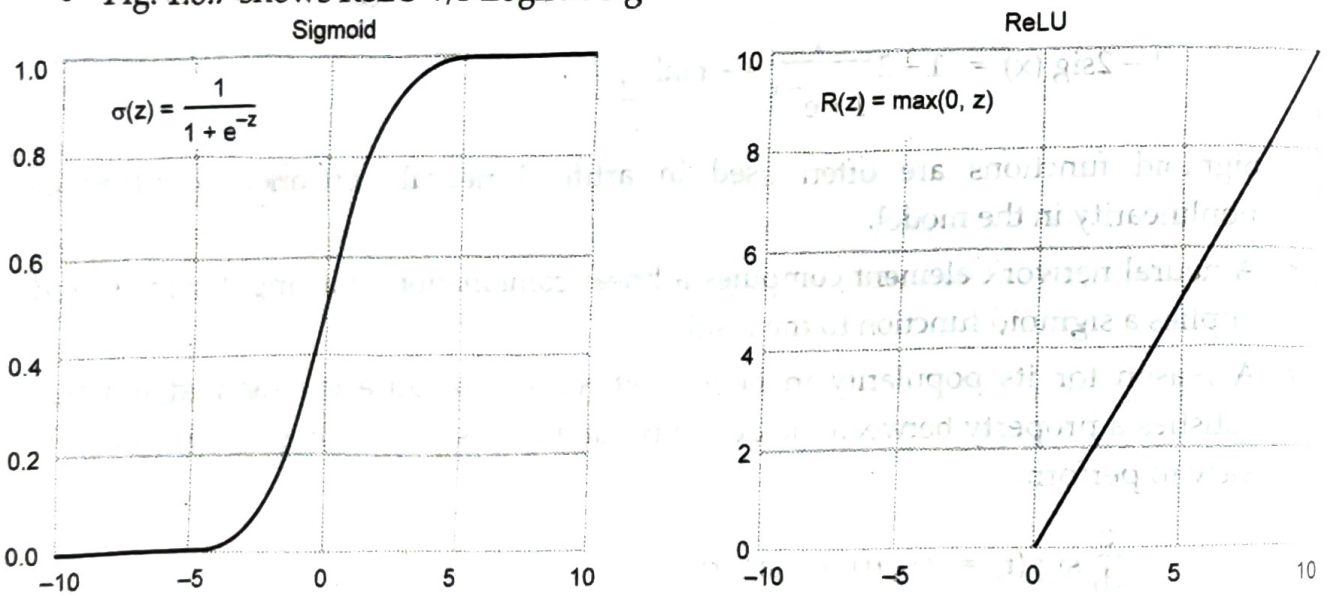


Fig. 1.6.7 ReLU v/s Logistic Sigmoid

- As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.
- Compared to tanh/sigmoid neurons that involve expensive operations (exponentials, etc.), the ReLUcan be implemented by simply thresholding a matrix of activations at zero.

Function	Advantages	Disadvantages
Sigmoid	1. Output in range (0,1)	1. Saturated neurons 2. Not zero entered 3. Small gradient 4. Vanishing gradient
Tanh	1. Zero centered, 2. Output in range (– 1,1)	1. Saturated neurons
ReLU	1. Computational efficiency, 2. Accelerated convergence	1. Dead neurons, 2. Not zero centered

1.7 Models of Neuron-McCulloch and Pitts Model

- The first mathematical model of a biological neuron was presented by McCulloch and Pitts. This model is known as McCulloch Pitt model. It is basic building block of neural network.
- Directed weight graph is used for connecting neurons.
- McCulloch and Pitts describe a neuron as a logical threshold element with two possible states. Such a threshold element has "N" input channels and one output channel. An input channel is either active (input 1) or silent (input 0).
- The activity states of all input channels thus encode the input information as a binary sequence of N bits. The state of the threshold element is then given by linear summation of all a different input signals x_i and comparison of the sum with a threshold value s.
- The system of neurons is static and acts synchronously. A processor (system) with multiple inputs and a single output.
- Effective input : Weighted sum of all inputs.
- Bias or threshold : If the effective input is larger than the bias, the neuron outputs a one, otherwise, it outputs a zero.

- Fig. 1.7.1 shows McCulloch Pitt model.

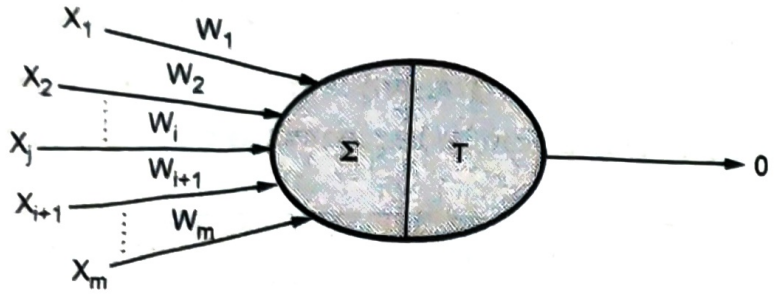


Fig. 1.7.1 McCulloch Pitt model

- This model can be described in a mathematical formalism as follows :

$$0 = \theta(a)$$

Where

$$a = \sum W_j X_j - T$$

And

$\theta(x)$ is a function such that $\theta(x) = 1$ if $x > 0$, otherwise $\theta(x) = 1$.

- The parameters used to scale the inputs are called the weights. The effective input is the weighted sum of the inputs. The parameter to measure the switching level is the threshold or bias. Neuron fires (output of one) when its net input excitation exceeds a certain value called 'threshold.' Threshold is the minimum value of the sum of the weighted active inputs needed for the postsynaptic neuron to fire.
- The function for producing the final output is called the activation function, which is the step function in the McCulloch-Pitts model.

$$0 = f\left(\sum_{j=1}^{N} W_j X_j - T\right)$$

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

Their "neurons" operated under the following assumptions :

1. They are binary devices (0, 1) .
2. Each neuron has a fixed threshold (theta).
3. The neuron receives inputs from excitatory synapses, all having identical weights.
4. Inhibitory inputs have an absolute veto power over any excitatory inputs.

5. At each time step the neurons are simultaneously (synchronously) updated by summing the weighted excitatory inputs and setting the output to 1 iff the sum is greater than or equal to the threshold AND if the neuron receives no inhibitory input.

- In general, there are many different kinds of activation functions. The step function used in the McCulloch-Pitts model is simply one of them. Because the activation function takes only two values, this model is called discrete neuron.
- To make the neuron learnable, some kind of continuous function is often used as the activation function. This kind of neurons is called continuous neurons. Typical functions used in an artificial neuron are sigmoid functions, radial basis function, sinusoidal functions, etc.

Problems with McCulloch-Pitts neurons

1. Weights and thresholds are analytically determined. We cannot learn.
2. It is very difficult to minimize size of a network.

1.8 Perceptron

- The perceptron is a feed-forward network with one output neuron that learns a separating hyper-plane in a pattern space.
- The "n" linear Fx neurons feed forward to one threshold output Fy neuron. The perceptron separates linearly separable set of patterns.

1.8.1 Single Layer Perceptron

- The perceptron is a feed-forward network with one output neuron that learns a separating hyper-plane in a pattern space. The "n" linear Fx neurons feed forward to one threshold output Fy neuron. The perceptron separates linearly separable set of patterns.
- SLP is the simplest type of artificial neural networks and can only classify linearly separable cases with a binary target (1, 0).
- We can connect any number of McCulloch-Pitts neurons together in any way we like. An arrangement of one input layer of McCulloch-Pitts neurons feeding forward to one output layer of McCulloch-Pitts neurons is known as a **Perceptron**.
- A single layer feed-forward network consists of one or more output neurons, each of which is connected with a weighting factor W_{ij} to all of the inputs X_i.
- The Perceptron is a kind of a single-layer artificial network with only one neuron. The Perception is a network in which the neuron unit calculates the linear

combination of its real-valued or boolean inputs and passes it through a threshold activation function. Fig. 1.8.1 shows Perceptron.

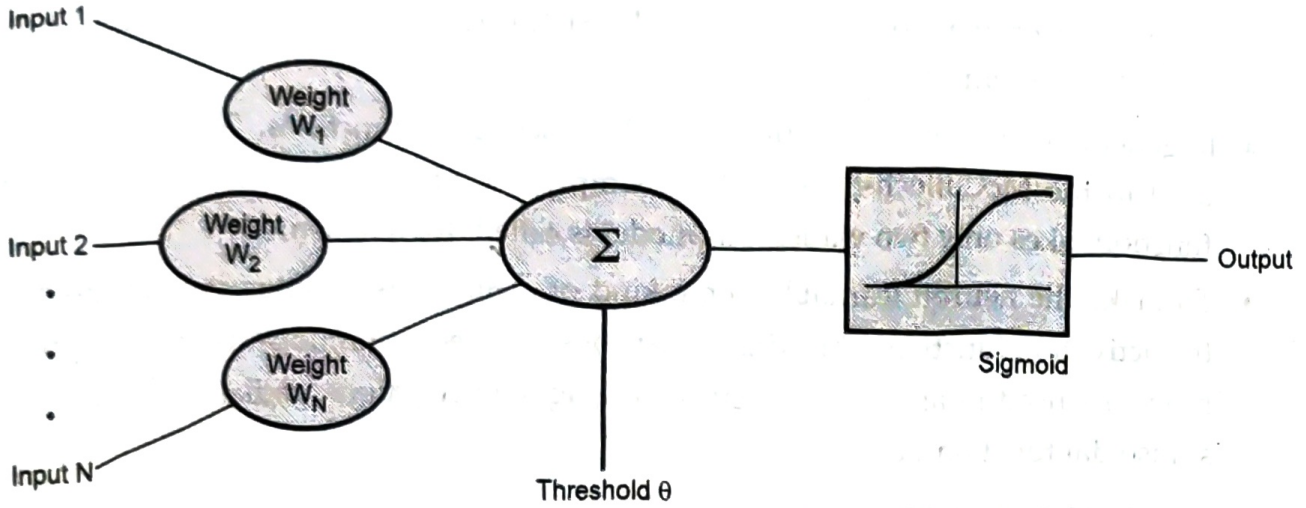


Fig. 1.8.1 Perceptron

- The Perceptron is sometimes referred to a Threshold Logic Unit (TLU) since it discriminates the data depending on whether the sum is greater than the threshold value.
- In the simplest case the network has only two inputs and a single output. The output of the neuron is :

$$y = f\left(\sum_{i=1}^{2} W_i X_i + b\right)$$

- Suppose that the activation function is a threshold then

$$f = \begin{cases} 1 & \text{if } s > 0 \\ -1 & \text{if } s \leq 0 \end{cases}$$

- The Perceptron can represent most of the primitive boolean functions : AND, OR, NAND and NOR but can not represent XOR.
- In single layer perceptron, initial weight values are assigned randomly because it does not have previous knowledge. It sum all the weighted inputs. If the sum is greater than the threshold value then it is activated i.e. output = 1.

Output

$$W_1X_1 + W_2X_2 + \ldots + W_nX_n > \theta \Rightarrow 1$$

$$W_1X_1 + W_2X_2 + \ldots + W_nX_n \leq \theta \Rightarrow 0$$

- The input values are presented to the perceptron, and if the predicted output is the same as the desired output, then the performance is considered satisfactory and no changes to the weights are made.
- If the output does not match the desired output, then the weights need to be changed to reduce the error.
- The weight adjustment is done as follows :

$$\Delta W = \eta \times d \times x$$

Where

x = Input data

d = Predicted output and desired output

η = Learning rate

- If the output of the perceptron is correct then we do not take any action. If the output is incorrect then the weight vector is $W \rightarrow W + \Delta W$.
- The process of weight adaptation is called **learning**.
- Perceptron Learning Algorithm :

1. Select random sample from training set as input.
2. If classification is correct, do nothing.
3. If classification is incorrect, modify the weight vector W using

$$W_i = W_i + \eta_d (n) X_i (n)$$

Repeat this procedure until the entire training set is classified correctly.

1.8.2 Multilayer Perceptron

- The perceptron is very useful for classifying data sets that are linearly separable.
- The Multilayer Perceptron (MLP) model features multiple layers that are interconnected in such a way that they form a feed-forward neural network. Each neuron in one layer has directed connections to the neurons of a separate layer.
- It consists of three types of layers: the input layer, output layer and hidden layer.
- Fig. 1.8.2 shows multilayer perceptron model.
- The **input layer** receives the input signal to be processed. The input layer distributes the values to each of the neurons in the hidden layer. In addition to the predictor variables, there is a constant input of 1.0, called the bias that is fed to each of the hidden layers; the bias is multiplied by a weight and added to the sum going into the neuron.

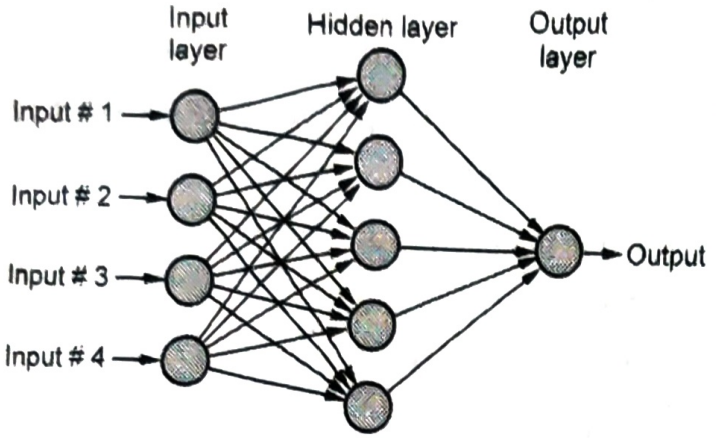


Fig. 1.8.2 Multilayer perceptron model

- **Hidden layer :** Arriving at a neuron in the hidden layer, the value from each input neuron is multiplied by a weight and the resulting weighted values are added together producing a combined value. The weighted sum is fed into a transfer function, which outputs a value. The outputs from the hidden layer are distributed to the output layer.
- **Output layer :** Arriving at a neuron in the output layer, the value from each hidden layer neuron is multiplied by a weight, and the resulting weighted values are added together producing a combined value. The weighted sum is fed into a transfer function, which outputs a value. The output values are the outputs of the network.
- The required task such as prediction and classification is performed by the output layer. An arbitrary number of hidden layers that are placed in between the input and output layer are the true computational engine of the MLP.
- The neurons in the MLP are trained with the back propagation learning algorithm. MLPs are designed to approximate any continuous function and can solve problems which are not linearly separable. The major use cases of MLP are pattern classification, recognition, prediction and approximation.
- The perceptron is very useful for classifying data sets that are linearly separable. They encounter serious limitations with data sets that do not conform to this pattern as discovered with the XOR problem. The XOR problem shows that for any classification of four points that there exists a set that are not linearly separable.
- The MultiLayer Perceptron breaks this restriction and classifies datasets which are not linearly separable. They do this by using a more robust and complex architecture to learn regression and classification models for difficult datasets.

- **Deciding how many neurons to use in the hidden layers :**

  a) One of the most important characteristics of a perceptron network is the number of neurons in the hidden layer(s). If an inadequate number of neurons are used, the network will be unable to model complex data, and the resulting fit will be poor.

  b) If too many neurons are used, the training time may become excessively long, and, worse, the network may over fit the data. When overfitting occurs, the network will begin to model random noise in the data. The result is that the model fits the training data extremely well, but it generalizes poorly to new, unseen data. Validation must be used to test for this.

1.8.3 ADALINE Network

- ADALINE (Adaptive Linear Neuron) is an early single-layer artificial neural network.
- An important generalized of the perceptrons training algorithm was presented by Widrow and Hoff as the least mean square learning procedure also known as the delta rule.
- The learning rule was applied to the "**adaptive linear element**" also named Adaline.
- The perceptron learning rule uses the output of the thersold function for learning. The delta rule uses the net output without further mapping into output values – 1 or + 1.
- Fig. 1.8.3 shows adaline.

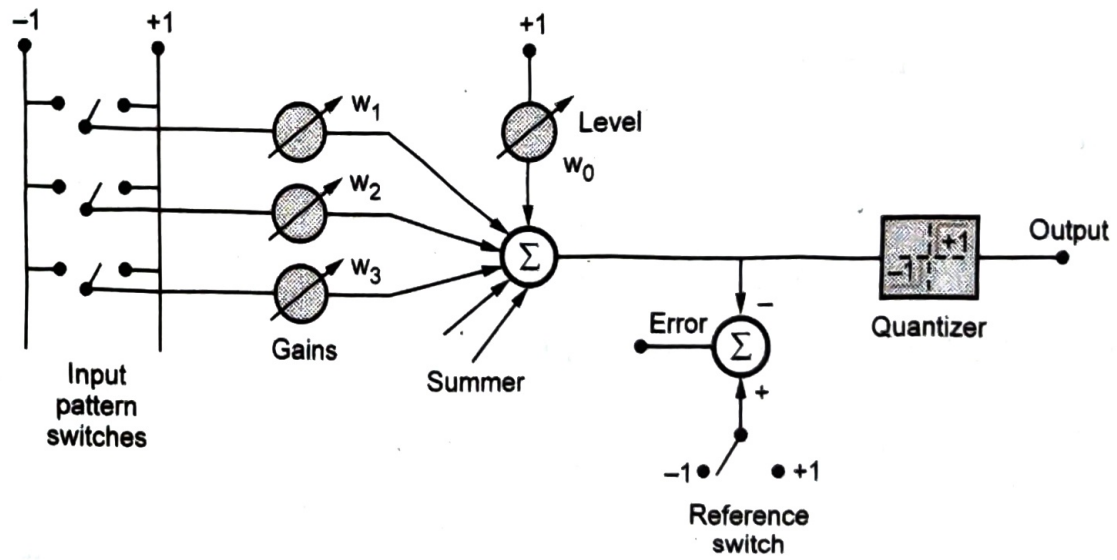


Fig. 1.8.3 Adaline

If the input conductances are denoted by w_i where $m_i = 0, 1, 2, \ldots, n$ and input and output signals by x_i and y respectively, then the output of the central block is defined to be :

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

Where, $\theta \equiv w_0$

- In a simple physical implementation, this device consists of a set of controllable resistors connected to a circuit which can sum up currents caused by the input voltage signals. Usually the central block, the summer is also followed by a quantizer which outputs + 1 of – 1, depending on the polarity of the sum.
- The problem is to determine the coefficients w_i, where $i = 0, 1 \ldots, n$, in such way that the input output response is correct for a large number of arbitrarily chosen signal sets.
- If an exact mapping is not possible the average error must be minimized, for instance, in the sense of least squares.
- An adaptive operation means that there exists a mechanism by which the w_i can be adjusted, usually iteratively to attain the correct values.
- For the Adaline, Widrow introduced the delta rule to adjust the weights.
- For the pth input-output pattern, the error measure of a single-output Adaline can be expressed as,

$$E_p = (t_p - o_p)^2$$

Where

t_p = Target output

o_p = Actual output of the Adaline

- The derivation of E_p with respect to each weight w_i is

$$\frac{\partial E_p}{\partial w_i} = -2(t_p - o_p)\, x_i$$

- To decrease E_p by gradient descent, the update formula for w_i on the pth input-output pattern is

$$\Delta_p w_i = \eta(t_p - o_p) x_i$$

- The delta rule tries to minimize squared errors, it is also referred to as the least mean square learning procedure or Widrow - Hoff learning rule.

1.9 Basic Learning Laws

By learning rule, we mean a procedure for modifying the weights and biases of a network. The purpose of the learning rule is to train the network to perform some tasks. There are many types of neural network learning rules. They fall into three broad categories : Supervised learning, Unsupervised learning and Reinforcement learning.

1.9.1 Weights and Bias

- Weights and biases are the learnable parameters that help a neural network correctly learn a function. A machine learning model uses lots of examples to learn the correct weights and bias to assign to each feature in a dataset to help it correctly predict outputs.
- Weight is the parameter within a neural network that transforms input data within the network's hidden layers. A neural network is a series of nodes, or neurons. Within each node is a set of inputs, weight and a bias value.
- As an input enters the node, it gets multiplied by a weight value and the resulting output is either observed or passed to the next layer in the neural network. Often the weights of a neural network are contained within the hidden layers of the network.

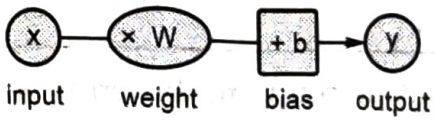


Fig. 1.9.1

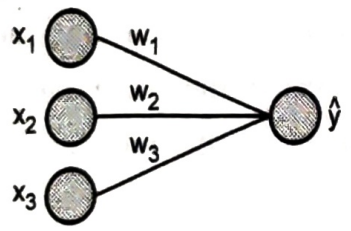
- Choose the weights w_i to best fit the set of training examples.
- Minimize the squared error E between the train values and the values predicted by the hypothesis

$$E = \sum_{(b,\, V_{train}(b)) \in \text{ training examples}} (V_{train}(b)) - \hat{V}((b))^2$$

- Require an algorithm that will incrementally refine weights as new training values.
- Least Mean Squares (LMS) is one such algorithm.

Weight vs. Bias

- A teachable neural network will randomize both the weight and bias values before learning initially begins. As training continues, both parameters are adjusted toward the desired values and the correct output.
- The two parameters differ in the extent of their influence upon the input data. Simply, bias represents how far off the predictions are from their intended value.
- Biases make up the difference between the function's output and its intended output. A low bias suggest that the network is making more assumptions about the form of the output, whereas a high bias value makes less assumptions about the form of the output.
- Weights, on the other hand, can be thought of as the strength of the connection. Weight affects the amount of influence a change in the input will have upon the output. A low weight value will have no change on the input and alternatively a larger weight value will more significantly change the output.
- In most learning networks, error is calculated as the difference between the actual output y and the predicted output y ? Loss functions are quantitative measures of how satisfactory the model predictions are.



Output :

$$\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + b = w^T x + b$$

Neural network parameters

$$W = \{ | w_1, w_2, w_3 | b \}$$

- The function that is used to compute this error is known as Loss Function also known as Cost function.

Mean Squared Error (MSE)

- The mean squared error calculates the average squared error between the outputs from the neural network and the targets in the data set.
- Mean Squared Error is one of the most common loss functions. MSE loss function is widely used in linear regression as the performance measure

$$\text{Mean Squared Error} = \frac{\Sigma\,(\text{Outputs} - \text{Targets})^2}{\text{Samples Number}}$$

Example 1.9.1 *For a given data having 100 examples, if saturated errors SE_1, SE_2 and SE_3 are 13.3, 3.33 and 4.00 respectively. Calculate Mean Squared Error (MSE). State the formula for MSE.*

Solution :

$$\text{Mean Squared Error} = \frac{\text{Squared Error}_1 + \text{Squared Error}_2 + \ldots + \text{Squared Error}_N}{\text{Number of data samples}}$$

$$= \frac{13.3 + 3.33 + 4.00}{100} = 0.2066$$

❑❑❑

Notes

Unit II

2 Learning Algorithms

Syllabus

Learning and Memory, Learning Algorithms, Numbers of hidden nodes, Error Correction and Gradient Decent Rules, Perceptron Learning Algorithms, Supervised Learning Backpropagation, MultilayeredNetwork Architectures, Back propagation Learning Algorithm, Feed forward and feedback neural networks,example and applications.

Contents

2.1 Learning and Memory

2.2 Hebbian Learning

2.3 Competitive Learning

2.4 Error Correction

2.5 Gradient Decent Rules

2.6 Supervised Learning

2.7 Backpropagation

2.8 Feed Forward Neural Networks

2.1 Learning and Memory

- One of the primary functions of the brain is associative memory. Learning can be considered as a process of forming associations between related patterns. The associative memory is composed of a cluster of units which represent a simple model of a real biological neuron.
- An associative memory, also known as Content-Addressable Memory (CAM) can be searched for a value in a single memory cycle rather than using a software loop.
- Associative memories can be implemented using networks with or without feedback. Such associative neural networks are used to associate one set of vectors with another set of vectors, say input and output patterns.
- The aim of an associative memory is, to produce the associated output pattern whenever one of the input patterns is applied to the neural network. The input pattern may be applied to the network either as input or as initial state and the output pattern is observed at the outputs of some neurons constituting the network.
- Associative memories belong to class of neural network that learn according to a certain recording algorithm. They require information a priori and their connectivity matrices most often need to be formed in advance. Writing into memory produces changes in the neural interconnections. Reading of the stored info from memory named recall, is a transformation of input signals by the network.
- All memory information is spatially distributed throughout the network. Associative memory enables a parallel search within a stored data. The purpose of search is to output one or all stored items that matches the search argument and retrieve it entirely or partially.
- The Fig. 2.1.1 shows a block diagram of an associative memory.

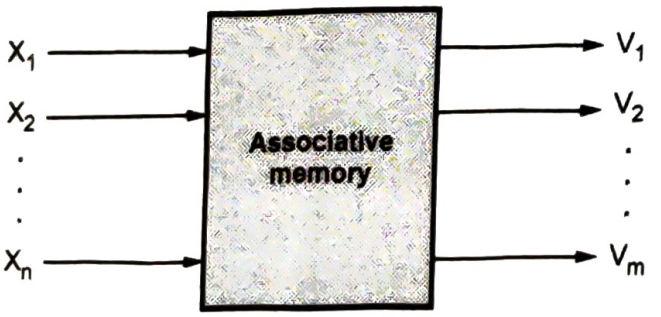


Fig. 2.1.1 Block diagram of an associative memory

- In the initialization phase of the associative memory no information is stored; because the information is represented in the w weights they are all set to zero.

- The advantage of neural associative memories over other pattern storage algorithms like lookup tables of hash codes is that the memory access can be fault tolerant with respect to variation of the input pattern.
- In associative memories many associations can be stored at the same time. There are different schemes of superposition of the memory traces formed by the different associations. The superposition can be simple linear addition of the synaptic changes required for each association (like in the Hopfield model) or nonlinear.
- The performance of neural associative memories is usually measured by a quantity called information capacity, that is, the information content that can be learned and retrieved, divided by the number of synapses required.
- An associative memory is a content-addressable structure that maps specific input representations to specific output representations. It is a system that "associates" two patterns (X, Y) such that when one is encountered, the other can be recalled.
- Associative network memory can be static or dynamic.
  - **Static :** Networks recall an output response after an input has been applied in one feed-forward pass and theoretically without delay. They were termed instantaneous.
  - **Dynamic :** Memory networks produce recall as a result of output/input feedback interaction, which requires time.
- There are two classes of associative memory : Auto-associative and hetero-associative.
- Whether auto-or hetero-associative, the net can associate not only the exact pattern pairs used in training, but is also able to obtain associations if the input is similar to one on which it has been trained.

2.1.1 Auto-associative Memory

- Auto-associative networks are a special subset of the hetero-associative networks, in which each vector is associated with itself, i.e. $y^i = x^i$ for i = 1, ..., m. The function of such networks is to correct noisy input vectors.
- Fig. 2.1.2 shows auto-associative memory.
- Auto-associative memories are content based memories which can recall a stored sequence when they are presented with a fragment or a noisy version of it. They are very effective in de-noising the input or removing interference from the input which makes them a promising first step in solving the cocktail party problem.
- The simplest version of auto-associative memory is linear associator which is a two-layer feed-forward fully connected neural network where the output is constructed in a single feed-forward computation.

Fig. 2.1.2 Auto-associative memory

- Artificial neural networks can be used as associative memories. One of the simplest artificial neural associative memory is the linear associator. The Hopfield model and Bidirectional Associative Memory (BAM) models are some of the other popular artificial neural network models used as associative memories

2.1.2 Hetero-associative Memory Network

- Hetero-associative networks map "m" input vectors $X^1, X^2, \ldots, X^m$ in n-dimensional space to m output vectors $y^1, y^2, \ldots, y^m$ in k-dimensional space, so that $X^i \rightarrow y^i$.
- If $\|\tilde{X} - X^i\| < \in$ then $\tilde{x} - y^i$. This should be achieved by the learning algorithm, but becomes very hard when the number m of vectors to be learned is too high.
- Fig. 2.1.3 shows block diagram of hetero-associative network.

Fig. 2.1.3 Auto-associative memory

- Fig. 2.1.4 shows the structure of a hetero-associative network without feedback.

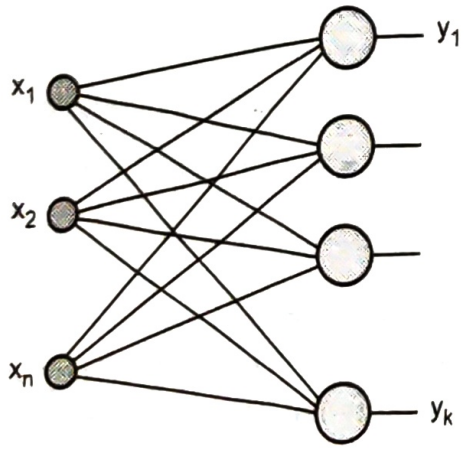


Fig. 2.1.4 Hetero-associative network without feedback

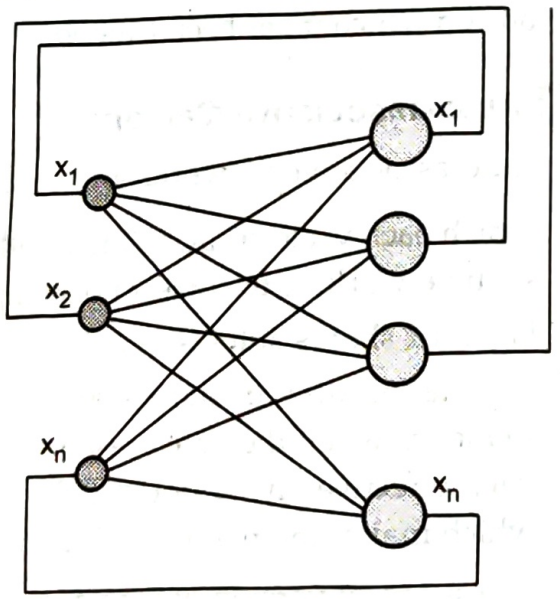


Fig. 2.1.5 Hetero-associative network with feedback

2.1.3 The Hopfield Network

- The **Hopfield** model is a single-layered recurrent network. Like the associative memory, it is usually initialized with appropriate weights instead of being trained.
- Hopfield Neural Network (HNN) is a model of auto-associative memory. It is a single layer neural network with feedbacks. Fig. 2.1.6 shows Hopfield network of three units. The Hopfield network is created by supplying input data vectors, or pattern vectors, corresponding to the different classes. These patterns are called class patterns.

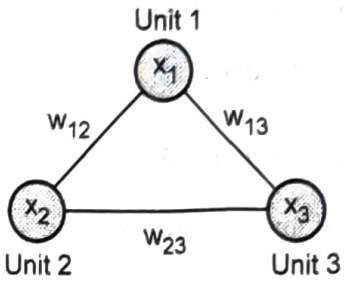


Fig. 2.1.6 Hopfield network of three units

- Hopfield model consists of a single layer of processing elements where each unit is connected to every other unit in the network other than itself.
- The output of each neuron is a binary number in {– 1, 1}. The output vector is the state vector. Starting from an initial state (given as the input vector), the state of the network changes from one to another like an automaton. If the state converges, the point to which it converges is called the attractor.
- In its simplest form, the output function is the sign function, which yields 1 for arguments ≥ 0 and – 1 otherwise.
- The connection weight matrix W of this type of network is square and symmetric. The units in the Hopfield model act as both input and output units.
- A Hopfield network consists of "n" totally coupled units. Each unit is connected to all other units except itself. The network is symmetric because the weight w_{ij} for the connection between unit i and unit j is equal to the weight w_{ij} of the connection from unit j to unit i. The absence of a connection from each unit to itself avoids a permanent feedback of its own state value.
- Hopfield networks are typically used for classification problems with binary pattern vectors.
- Hopfield model is classified into two categories :
  1. Discrete Hopfield Model
  2. Continuous Hopfield Model

- In both discrete and continuous Hopfield network weights trained in a one-shot fashion and not trained incrementally as was done in case of Perceptron and MLP.
- In the discrete Hopfield model, the units use a slightly modified bipolar output function where the states of the units, i.e., the output of the units remain the same if the current state is equal to some threshold value.
- The continuous Hopfield model is just a generalization of the discrete case. Here, the units use a continuous output function such as the sigmoid or hyperbolic tangent function. In the continuous Hopfield model, each unit has an associated capacitor C_i and resistance r_i that model the capacitance and resistance of real neuron's cell membrane, respectively.

2.1.4 Bidirectional Associative Memory (BAM)

- BAM consists of two layers, x and y. Signals are sent back and forth between both layers until an equilibrium is reached. Equilibrium is reached if the x and y vectors no longer change. Given an x vector the BAM is able to produce the y vector and vice versa.
- BAM consists of bi-directional edges so that information can flow in either direction. Since the BAM network has bidirectional edges, propagation moves in both directions, first from one layer to another and then back to the first layer. Propagation continues until the nodes are no longer changing values.
- Fig. 2.1.7 shows BAM network.

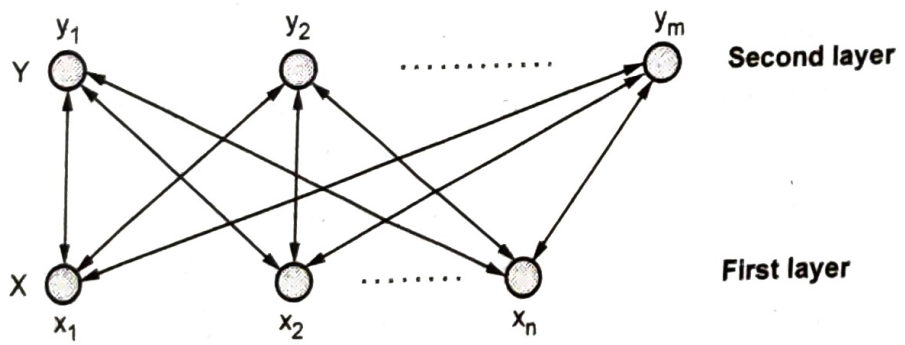


Fig. 2.1.7 BAM network

- Since the BAM also uses the traditional Hebb's learning rule to build the connection weight matrix to store the associated pattern pairs, it too has a severely low memory capacity.

- BAM can be classified into two categories :
  1. **Discrete BAM :** The network propagates an input pattern X to the Y layer where the units in the Y layer will compute their net input.
  2. **Continuous BAM :** The units use the sigmoid or hyperbolic tangent output function. The units in the X layer have an extra external input I_i, while the units in the Y layer have an extra external input J_j for i = 1, 2, ..., m and j = 1, 2, ..., n.
- These extra external inputs lead to a modification in the computation of the net input to the units.

2.1.5 Difference between Auto-associative Memory and Hetero-Associative Memory

Auto-associative memory	Hetero-associative memory
The inputs and output vectors s and t are the same.	The inputs and output vectors s and t are different.
Recalls a memory of the same modality as the one that evoked it.	Recalls a memory that is different in character from the input.
A picture of a favorite object might evoke a mental image of that object in vivid detail.	A particular smell or sound, for example, might evoke a visual memory of some past event.
An auto-associative memory retrieves the same pattern.	Hetero-associative memory retrieves the stored pattern.
Example : color correction, color constancy	Example : 1. Space transforms : Fourier, 2. Dimensionality reduction : PCA

2.2 Hebbian Learning

- In 1949, Donald Hebb proposed one of the key ideas in biological learning, commonly known as **Hebb's Law**. Hebb's Law states that if neuron i is near enough to excite neuron j and repeatedly participates in its activation, the synaptic connection between these two neurons is strengthened and neuron j becomes more sensitive to stimuli from neuron i.
- Hebb's Law can be represented in the form of two rules :
  1. If two neurons on either side of a connection are activated synchronously, then the weight of that connection is increased.
  2. If two neurons on either side of a connection are activated asynchronously, then the weight of that connection is decreased.

- Hebb's Law provides the basis for learning without a teacher. Learning here is a **local phenomenon** occurring without feedback from the environment.
- Using Hebb's Law we can express the adjustment applied to the weight w_{ij} at iteration p in the following form :

$$\Delta w_{ij}(p) = F[y_i(p), x_i(p)]$$

- As a special case, we can represent Hebb's Law as follows :

$$\Delta w_{ij}(p) = \alpha\, y_i(p)\, x_i(p)$$

where α is the learning rate parameter. This equation is referred to as the **activity product rule**.

- Hebbian learning implies that weights can only increase. To resolve this problem, we might impose a limit on the growth of synaptic weights. It can be done by introducing a non-linear **forgetting factor** into Hebb's Law :

$$\Delta w_{ij}(p) = \alpha\, y_i(p)\, x_i(p) - \varphi\, y_i(p)\, w_{ij}(p)$$

where φ is the forgetting factor.

- Forgetting factor usually falls in the interval between 0 and 1, typically between 0.01 and 0.1, to allow only a little "forgetting" while limiting the weight growth.

Hebbian learning algorithm

Step 1 : Initialisation

Set initial synaptic weights and thresholds to small random values, say in an interval [0, 1].

Step 2 : Activation

Compute the neuron output at iteration p

$$y_j(p) = \sum_{i=1}^{n} x_i(p)\, w_{ij}(p) - \theta_j$$

where n is the number of neuron inputs, and is the threshold value of neuron j.

Step 3 : Learning

Update the weights in the network :

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p)$$

where $w_{ij}(p)$ is the weight correction at iteration p. The weight correction is determined by the generalised activity product rule :

$$w_{ij}(p) = \varphi\, y_i(p)\, [\lambda\, x_i(p) - w_{ij}(p)]$$

Step 4 : Iteration

Increase iteration p by one, go back to Step 2.

Hebbian learning example

- To illustrate Hebbian learning, consider a fully connected feed forward network with a single layer of five computation neurons. Each neuron is represented by a McCulloch and Pitts model with the sign activation function. The network is trained on the following set of input vectors :

$$x_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad x_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \quad x_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \quad x_4 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \quad x_5 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Initial and final weight matrices

$$\begin{array}{cc} & \text{Output layer} \\ & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \text{Input layer} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

$$\begin{array}{cc} & \text{Output layer} \\ & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \text{Input layer} \begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} & \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2.0204 & 0 & 0 & 2.0204 \\ 0 & 0 & 1.0200 & 0 & 0 \\ 0 & 0 & 0 & 0.9996 & 0 \\ 0 & 2.0204 & 0 & 0 & 2.0204 \end{bmatrix} \end{array}$$

- A test input vector, or probe, is defined as

$$X = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

- When this probe is presented to the network, we obtain :

$$Y = \left\{ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 2.0204 & 0 & 0 & 2.0204 \\ 0 & 0 & 1.0200 & 0 & 0 \\ 0 & 0 & 0 & 0.9996 & 0 \\ 0 & 2.0204 & 0 & 0 & 2.0204 \end{bmatrix} \right\} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 0.4940 \\ 0.2661 \\ 0.0907 \\ 0.9478 \\ 0.0737 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

2.3 Competitive Learning

- Fig. 2.3.1 shows basic architecture of a competitive learning system.

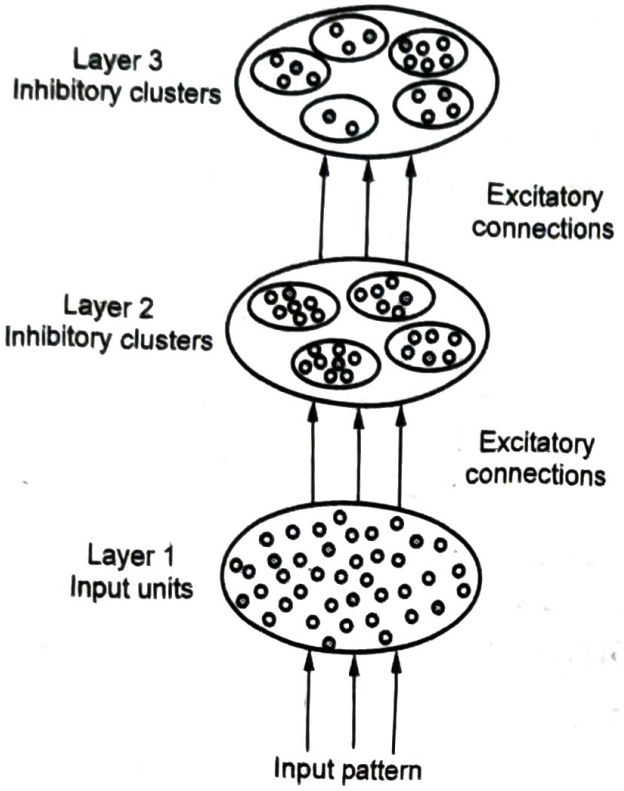


Fig. 2.3.1 Competitive learning system

- It consists of a set of hierarchically layered units in which each layer connects, via excitatory connections, with the layer immediately above it, and has inhibitory connections to units in its own layer.
- In the most general case, each unit in a layer receives an input from each unit in the layer immediately below it and projects to each unit in the layer immediately above it.
- Moreover, within a layer, the units are broken into a set of inhibitory clusters in which all elements within a cluster inhibit all other elements in the cluster.
- Thus the elements within a cluster at one level compete with one another to respond to the pattern appearing on the layer below. The more strongly any particular unit responds to an incoming stimulus, the more it shuts down the other members of its cluster.
- Units are represented in the diagram as dots. Units may be active or inactive. Active units are represented by filled dots, inactive ones by open dots.
- **Properties :**
  1. The units in a given layer are broken into several sets of non-overlapping clusters. Each unit within a cluster inhibits every other unit within a cluster.

Within each cluster, the unit receiving the largest input achieves its maximum value while all other units in the cluster are pushed to their minimum value 1. We have arbitrarily set the maximum value to 1 and the minimum value to 0.

2. Every unit in every cluster receives inputs from all members of the same set of input units.
3. A unit learns if and only if it wins the competition with other units in its cluster.
4. A stimulus pattern Sj consists of a binary pattern in which each element of the pattern is either active or inactive. An active element is assigned the value 1 and an inactive element is assigned the value 0.
5. Each unit has a fixed amount of weight (all weights are positive) that is distributed among its input lines.

- There are several characteristics of a competitive learning mechanism :
  1. Each cluster classifies the stimulus set into M groups, one for each unit in the cluster. Each of the units captures roughly an equal number of stimulus patterns.
  2. If there is structure in the stimulus patterns, the units will break up the patterns along structurally relevant lines.
  3. If the stimuli are highly structured, the classifications are highly stable. If the stimuli are less well structured, the classifications are more variable, and a given stimulus pattern will be responded to first by one and then by another member of the cluster.
  4. The particular grouping done by a particular cluster depends on the starting value of the weights and the sequence of stimulus patterns actually presented.
  5. To a first approximation, the system develops clusters that minimize within-cluster distance, maximize between-cluster distance, and balance the number of patterns captured by each cluster.

2.4 Error Correction

- Learning is a process where unknown ANN parameters are adapted through continuous process of stimulation from the environment. Learning is determined by the way how the change of parameters takes place. A set of rules that are solution to the learning problem is called a learning algorithm.
- Error correction belongs to the supervised learning paradigm. Fig. 2.4.1 shows error correction learning.

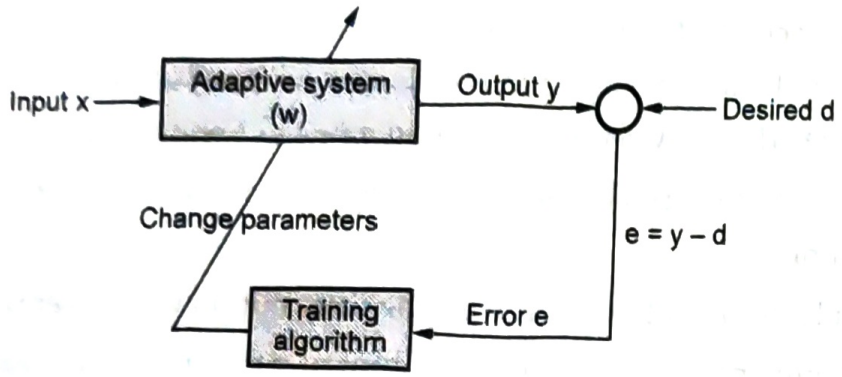


Fig. 2.4.1 Error correction learning

- Neuron (k) is driven by a signal vector x(n) produced by one or more layers of hidden neurons, which are themselves driven by an input vector applied to the source nodes of the neural network.
- Let $d_k(n)$ be desired output of neuron k at moment n
- Let $y_k(n)$ be obtained output of neuron k at moment n
- Output $y_k(n)$ is obtained using input vector x(n).
- Input vector x(n) and desired output $d_k(n)$ represent an example that is presented to ANN at moment n.
- Error is the difference between desired and obtained output of neuron k at moment n :

$$e_k(n) = d_k(n) - y_k(n)$$

- Error signal $e_k(n)$ actuates a control mechanism.
- The goal of error-correction learning is to minimize an error function derived from errors $e_k(n)$ so that the obtained output of all neurons approximates the desired output in some statistical sense. A frequently used error function is mean square error :

$$J = E\left[\frac{1}{2}\sum_k e_k^2(n)\right]$$

where E[.] is the statistical expectation operator and summation is for all neurons in the output layer.

2.5 Gradient Decent Rules

- Much of machine learning can be written as an optimization problem.
- Example loss functions : Logistic regression, linear regression, principle component analysis, neural network loss.

- A very efficient way to train logistic models is with **Stochastic Gradient Descent (SGD).**
- One challenge with training on power law data (i.e. most data) is that the terms in the gradient can have very different strengths.
- The idea behind stochastic gradient descent is iterating a weight update based on the gradient of loss function :

$$\bar{w}(k+1) = \bar{w}(k) - \gamma \nabla L(\bar{w})$$

- Logistic regression is designed as a **binary classifer** (output say {0, 1}) but actually **outputs the probability** that the input instance is in the "1" class.
- A logistic classifier has the form :

$$p(X) = \frac{1}{1 + \exp(-X\beta)}$$

where $X = (X_1, \ldots, X_n)$ is a vector of features.

- Stochastic gradient has some serious limitations however, especially if the gradients vary widely in magnitude. Some coefficients change very fast, others very slowly.
- This happens for text, user activity and social media data (and other power-law data), because gradient magnitudes scale with feature frequency, i.e. over several orders of magnitude.
- It is not possible to set a single learning rate that trains the frequent and infrequent features at the same time.
- An example of stochastic gradient descent with perceptron loss is shown as follows :

```
from sklearn.linear_model import SGDClassifier
```

2.5.1 Finding the Optimal Hyper-Parameters through Grid Search

- In statistics, hyperparameter is a parameter from a prior distribution; it captures the prior belief before data is observed.
- In any machine learning algorithm, these parameters need to be initialized before training a model.
- **Model hyperparameters** *are the properties that govern the entire training process.*
- Hyperparameters are important because they directly control the behaviour of the training algorithm and have a significant impact on the performance of the model is being trained.

- Choosing appropriate hyperparameters plays a crucial role in the success of our neural network architecture. Since it makes a huge impact on the learned model.
- For example, if the learning rate is too low, the model will miss the important patterns in the data. If it is high, it may have collisions.
- Choosing good hyperparameters gives two benefits :
  1. Efficiently search the space of possible hyperparameters.
  2. Easy to manage a large set of experiments for hyperparameter tuning.
- **The process of finding most optimal hyperparameters in machine learning is called hyperparameter optimisation.**
- Grid search is a very traditional technique for implementing hyperparameters. It brute force all combinations. Grid search requires to create two set of hyperparameters.

  1. Learning rate 2. Number of layers
- Grid search trains the algorithm for all combinations by using the two set of hyperparameters and measures the performance using "Cross Validation" technique.
- This validation technique gives assurance that our trained model got most of the patterns from the dataset.
- One of the best methods to do validation by using "K-Fold Cross Validation" which helps to provide ample data for training the model and ample data for validations.
- With this technique, we simply build a model for each possible combination of all of the hyperparameter values provided, evaluating each model and selecting the architecture which produces the best results.
- For example, say you have two continuous parameters α and β, where manually selected values for the parameters are the following :

$$\alpha \in \{0, 1, 2\}$$

$$\beta \in \{.25, .50, .75\}$$

- Then the pairing of the selected hyperparametric values, H, can take on any of the following :

  $H \in \{(0, .25), (0, .50), (0, .75), (1, .25), (1, .50), (1, .75), (2, .25), (2, .50), (2, .75)\}$
- Grid search will examine each pairing of and to determine the best performing combination. The resulting pairs, *H*, are simply each output that results from taking the Cartesian product of α and β.

- While straightforward, this "brute force" approach for hyperparameter optimization has some drawbacks. Higher-dimensional hyperparametric spaces are far more time consuming to test than the simple two-dimensional problem presented here.
- Also, because there will always be a fixed number of training samples for any given model, the model's predictive power will decrease as the number of dimensions increases. This is known as Hughes phenomenon.

2.5.2 Vanishing Gradient Problem

- When back-propagation is used, the earlier layers will receive very small updates compared to the later layers. This problem is referred to as the vanishing gradient problem.
- The vanishing gradient problem is essentially a situation in which a deep multilayer feed-forward network or a Recurrent Neural Network (RNN) does not have the ability to propagate useful gradient information from the output end of the model back to the layers near the input end of the model.
- Weight initialization is one technique that can be used to solve the vanishing gradient problem. It involves artificially creating an initial value for weights in a neural network to prevent the backpropagation algorithm from assigning weights that are unrealistically small.
- The most important solution to the vanishing gradient problem is a specific type of neural network called Long Short-Term Memory Networks (LSTMs).
- Indication of vanishing gradient problem :
  a) The parameters of the higher layers change to a great extent, while the parameters of lower layers barely change.
  b) The model weights could become 0 during training.
  c) The model learns at a particularly slow pace and the training could stagnate at a very early phase after only a few iterations.
- Some methods that are proposed to overcome the vanishing gradient problem :
  a) Residual neural networks (ResNets)
  b) Multi-level hierarchy
  c) Long short term memory (LSTM)
  d) Faster hardware
  e) ReLU
  f) Batch normalization

2.6 Supervised Learning

- Supervised learning is the machine learning task of inferring a function from supervised training data. The training data consist of a set of training examples. The task of the supervised learner is to predict the output behavior of a system for any set of input values, after an initial training phase.
- Supervised learning in which the network is trained by providing it with input and matching output patterns. These input-output pairs are usually provided by an external teacher.
- Human learning is based on the past experiences. A computer does not have experiences.
- A computer system learns from data, which represent some "past experiences" of an application domain.
- To learn a target function that can be used to predict the values of a discrete class attribute, e.g., approve or not-approved and high-risk or low risk. The task is commonly called : Supervised learning, Classification or inductive learning.
- Training data includes both the input and the desired results. For some examples the correct results (targets) are known and are given in input to the model during the learning process. The construction of a proper training, validation and test set is crucial. These methods are usually fast and accurate.
- Have to be able to generalize : give the correct results when new data are given in input without knowing a priori the target.
- Supervised learning is the machine learning task of inferring a function from supervised training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object and a desired output value.
- A supervised learning algorithm analyzes the training data and produces an inferred function, which is called a classifier or a regression function. Fig. 2.6.1 shows supervised learning process.

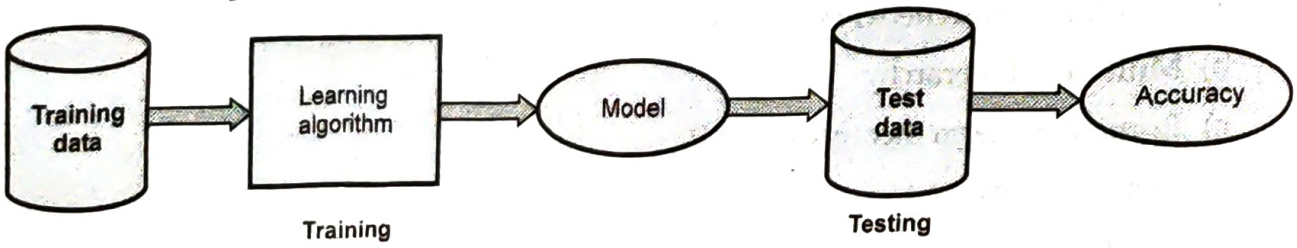


Fig. 2.6.1 Supervised learning process

- The learned model helps the system to perform task better as compared to no learning.
- Each input vector requires a corresponding target vector.

  Training Pair = (Input Vector, Target Vector)
- Fig. 2.6.2 shows input vector.
- Supervised learning denotes a method in which some input vectors are collected and presented to the network. The output computed by the net-work is observed and the deviation from the expected answer is measured. The weights are corrected according to the magnitude of the error in the way defined by the learning algorithm.

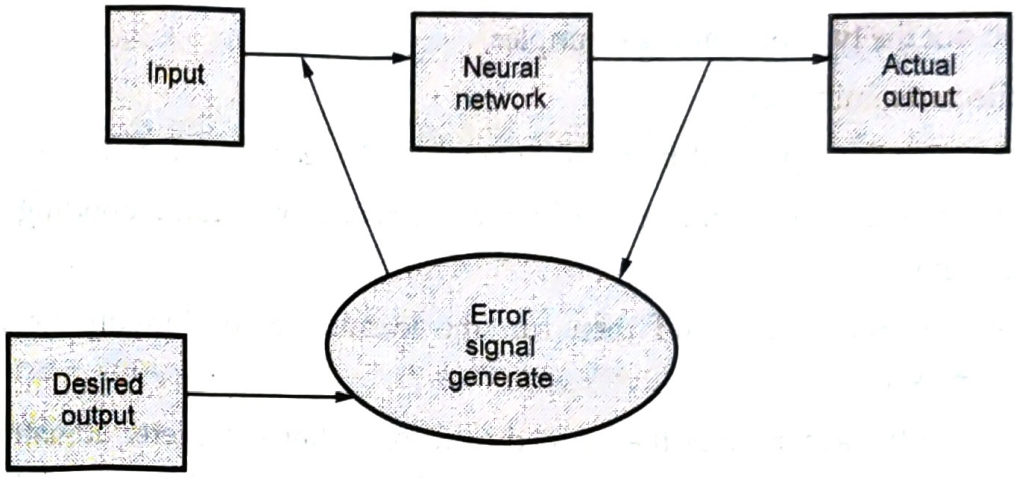


Fig. 2.6.2 Input vector

- Supervised learning is further divided into methods which use reinforcement or error correction. The perceptron learning algorithm is an example of supervised learning with reinforcement.

Data formats in supervised learning :

- Supervised learning always uses a dataset to define finite set of real vectors with m features each :

$$X = \{\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n\} \text{ where } \bar{X}_i \in \mathbb{R}^m$$

- Considering that user approach is always probabilistic, we need to consider each X as drawn from a statistical multivariate distribution D. It is also useful to add an important condition upon the whole dataset X. Here we consider that all samples to be independent and identically distributed. This means all variables belong to the same distribution D and considering an arbitrary subset of m values, it happens that :

$$P(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_m) = \prod_{i=1}^{m} P(\bar{x}_i)$$

- The corresponding output values can be both numerical - continuous and categorical. In the first case, the process is called regression, while in the second, it is called classification.
- Example : Dataset contains city populations by year for the past 100 years and user want to know what the population of a specific city will be four years from now. The outcome uses labels that already exist in the data set : **population, city and year.**
- In order to solve a given problem of supervised learning, following steps are performed :
  1. Find out the type of training examples.
  2. Collect a training set.
  3. Determine the input feature representation of the learned function.
  4. Determine the structure of the learned function and corresponding learning algorithm.
  5. Complete the design and then run the learning algorithm on the collected training set.
  6. Evaluate the accuracy of the learned function. After parameter adjustment and learning, the performance of the resulting function should be measured on a test set that is separate from the training set.
- Supervised learning is divided into two types : Classification and Regression.

1. Classification :

- Classification predicts categorical labels (classes), prediction models continuous - valued functions. Classification is considered to be supervised learning.
- Classifies data based on the training set and the values in a classifying attribute and uses it in classifying new data. **Prediction** means models continuous-valued functions, i.e., predicts unknown or missing values.
- Preprocessing of the data in preparation for classification and prediction can involve data cleaning to reduce noise or handle missing values, relevance analysis to remove irrelevant or redundant attributes and data transformation, such as generalizing the data to higher level concepts or normalizing data.
- Numeric prediction is the task of predicting continuos values for given input. For example, we may wish to predict the salary of college employee with 15 years of work experience or the potential sales of a new product given its price.

- Some of the classification methods like back - propagation, support vector machines and k-nearest-neighbor classifiers can be used for prediction.

2. Regression :

- For an input x, if the output is continuous, this is called a regression problem. For example, based on historical information of demand for tooth paste in supermarket, user are asked to predict the demand for the next month.
- Regression is concerned with the prediction of continuous quantities. Linear regression is the oldest and most widely used predictive model in the field of machine learning. The goal is to minimize the sum of the squared errors to fit a straight line to a set of data points.
- Regression algorithm used in supervised learning is linear regression, Bayesian linear regression, polynomial regression, regression tree etc.

2.6.1 Advantages and Disadvantages of Supervised Learning

1. Advantages of supervised learning

- It performs classification and regression tasks.
- It allows estimating or mapping the result to a new sample.
- We have complete control over choosing the number of classes we want in the training data.

2. Disadvantages of supervised learning

- Supervised learning cannot handle all complex tasks in Machine Learning.
- Computation time is vast for supervised learning.
- It requires a labelled data set.
- It requires a training process.

2.6.2 Difference between Supervised and Unsupervised Learning

Sr. No.	Supervised learning	Unsupervised learning
1.	Desired output is given.	Desired output is not given.
2.	It is not possible to learn larger and more complex models than with supervised learning.	It is possible to learn larger and more complex models with unsupervised learning.
3.	Use training data to infer model.	No training data is used.
4.	Every input pattern that is used to train the network is associated with an output pattern.	The target output is not presented to the network.

5.	Trying to predict a function from labeled data.	Try to detect interesting relations in data.
6.	Supervised learning requires that the target variable is well defined and that a sufficient number of its values are given.	For unsupervised learning typically either the target variable is unknown or has only been recorded for too small a number of cases.
7.	Example : Optical character recognition.	Example : Find a face in an image.
8.	We can test our model.	We can not test our model.
9.	Supervised learning is also called classification.	Unsupervised learning is also called clustering.

2.7 Backpropagation

- Backpropagation is a training method used for a multi-layer neural network. It is also called the generalized delta rule. It is a gradient descent method which minimizes the total squared error of the output computed by the net.
- The backpropagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem.
- Backpropagation is a systematic method for training multiple layer ANN. It is a generalization of Widrow-Hoff error correction rule. 80 % of ANN applications uses backpropagation.
- Fig. 2.7.1 shows backpropagation network.

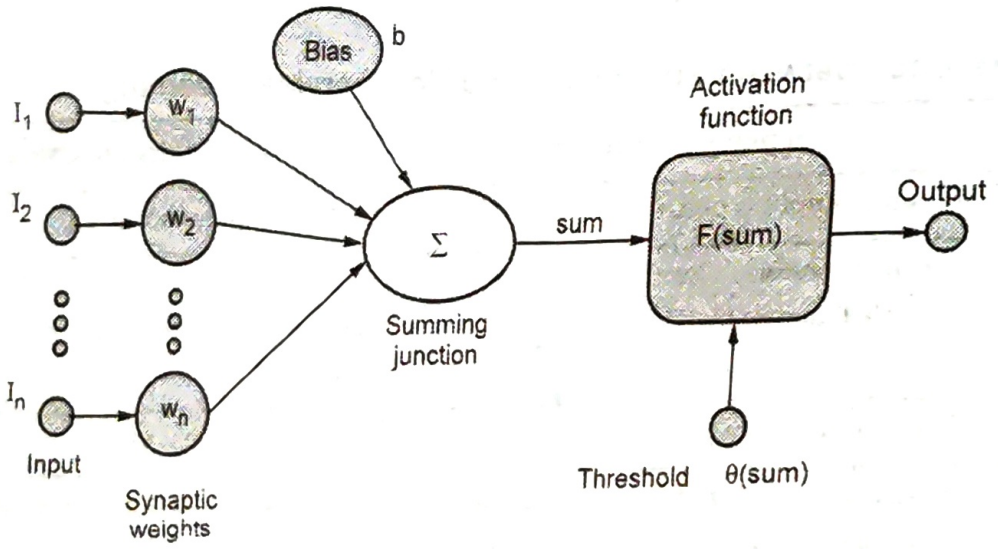


Fig. 2.7.1 Backpropagation network

- Consider a simple neuron :
  a. Neuron has a summing junction and activation function.
  b. Any non linear function which differentiable everywhere and increases everywhere with sum can be used as activation function.
  c. Examples : Logistic function, Arc tangent function, Hyperbolic tangent activation function.
- These activation function makes the multilayer network to have greater representational power than single layer network only when non-linearity is introduced.
- **Need of hidden layers :**
  1. A network with only two layers (input and output) can only represent the input with whatever representation already exists in the input data.
  2. If the data is discontinuous or non-linearly separable, the innate representation is inconsistent, and the mapping cannot be learned using two layers (Input and Output).
  3. Therefore, hidden layer(s) are used between input and output layers.
- **Weights** connects unit (neuron) in one layer only to those in the next higher layer. The output of the unit is scaled by the value of the connecting weight and it is fed forward to provide a portion of the activation for the units in the next higher layer.
- Backpropagation can be applied to an artificial neural network with any number of hidden layers. The training objective is to adjust the weights so that the application of a set of inputs produces the desired outputs.
- **Training procedure :** The network is usually trained with a large number of input-output pairs.
  1. Generate weights randomly to small random values (both positive and negative) to ensure that the network is not saturated by large values of weights.
  2. Choose a training pair from the training set.
  3. Apply the input vector to network input.
  4. Calculate the network output.
  5. Calculate the error, the difference between the network output and the desired output.
  6. Adjust the weights of the network in a way that minimizes this error.
  7. Repeat steps 2 - 6 for each pair of input-output in the training set until the error for the entire system is acceptably low.

Forward pass and backward pass :

- Backpropagation neural network training involves two passes.
  1. In the forward pass, the input signals moves forward from the network input to the output.
  2. In the backward pass, the calculated error signals propagate backward through the network, where they are used to adjust the weights.
  3. In the forward pass, the calculation of the output is carried out, layer by layer, in the forward direction. The output of one layer is the input to the next layer.
- In the reverse pass,
  a. The weights of the output neuron layer are adjusted first since the target value of each output neuron is available to guide the adjustment of the associated weights, using the delta rule.
  b. Next, we adjust the weights of the middle layers. As the middle layer neurons have no target values, it makes the problem complex.
- **Selection of number of hidden units :** The number of hidden units depends on the number of input units.
  1. Never choose h to be more than twice the number of input units.
  2. You can load p patterns of I elements into $\log_2 p$ hidden units.
  3. Ensure that we must have at least 1/e times as many training examples.
  4. Feature extraction requires fewer hidden units than inputs.
  5. Learning many examples of disjointed inputs requires more hidden units than inputs.
  6. The number of hidden units required for a classification task increases with the number of classes in the task. Large networks require longer training times.

Factors influencing backpropagation training

- The training time can be reduced by using :
  1. **Bias :** Networks with biases can represent relationships between inputs and outputs more easily than networks without biases. Adding a bias to each neuron is usually desirable to offset the origin of the activation function. The weight of the bias is trainable similar to weight except that the input is always +1.
  2. **Momentum :** The use of momentum enhances the stability of the training process. Momentum is used to keep the training process going in the same general direction analogous to the way that momentum of a moving object behaves. In backpropagation with momentum, the weight change is a combination of the current gradient and the previous gradient.

2.7.1 Advantages and Disadvantages

Advantages of backpropagation :

1. It is simple, fast and easy to program.
2. Only numbers of the input are tuned and not any other parameter.
3. No need to have prior knowledge about the network.
4. It is flexible.
5. A standard approach and works efficiently.
6. It does not require the user to learn special functions.

Disadvantages of backpropagation :

1. Backpropagation possibly be sensitive to noisy data and irregularity.
2. The performance of this is highly reliant on the input data.
3. Needs excessive time for training.
4. The need for a matrix-based method for backpropagation instead of mini-batch.

2.8 Feed Forward Neural Networks

- Feed Forward Neural Network is an artificial neural network in which the connections between nodes does not form a cycle. The feed forward model is the simplest form of neural network as information is only processed in one direction. While the data may pass through multiple hidden nodes, it always moves in one direction and never backwards.
- They are called feed forward because information only travels forward in the network (no loops), first through the input nodes, then through the hidden nodes (if present) and finally through the output nodes.
- Feed-forward networks tends to be simple networks that associates inputs with outputs. It can be used in pattern recognition. This type of organization is represented as bottom-up or top-down.
- Fig. 2.8.1 shows basic structure of a Feed Forward (FF) Neural Network.
- Input layer contains one or more input nodes. For example, suppose we want to predict whether it will rain tomorrow and base our decision on two variables, humidity and wind speed. In that case, our first input would be the value for humidity and the second input would be the value for wind speed.

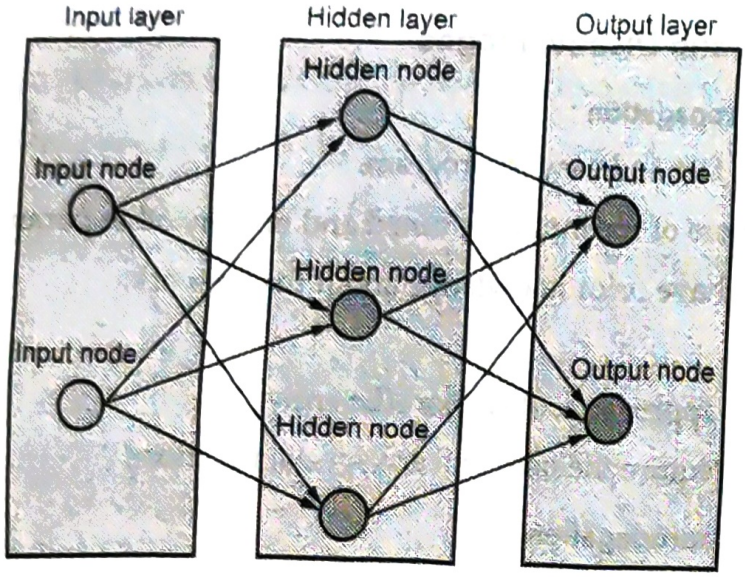


Fig. 2.8.1 Basic structure of a Feed Forward Neural Network

- Hidden layer : This layer contains an activation function.
- Output layer contains one or more output nodes.
- Feed foward neural networks are primarily used for supervised learning in cases where the data to be learned is neither sequential nor time-dependent.
- Feed-forward networks have the following characteristics :
  1. Perceptron's are arranged in layers, with the first layer taking in inputs and the last layer producing outputs. The middle layers have no connection with the external world and hence are called hidden layers.
  2. Each perceptron in one layer is connected to every perceptron on the next layer. Hence information is constantly "fed forward" from one layer to the next and this explains why these networks are called feed-forward networks.
  3. There is no connection among perceptron's in the same layer.

2.8.1 Feedback Neural Networks

- Feedback networks also known as recurrent neural network or interactive neural network are the deep learning models in which information flows in backward direction. It allows feedback loops in the network. Feedback networks are dynamic in nature, powerful and can get much complicated at some stage of execution.
- Fig. 2.8.2 shows feedback neural network.
- Signals can travel in both the directions in feedback neural networks. Feedback neural networks are very powerful and can get very complicated. Feedback neural networks are dynamic.

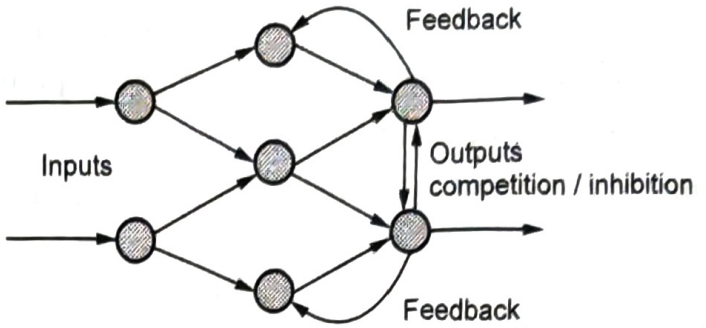


Fig. 2.8.2 Feedback neural network

- The 'state' in such network keep changing until they reach an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found.
- Feedback neural network architecture is also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations. Feedback loops are allowed in such networks. They are used in content addressable memories.

2.8.3 Example and Applications of Feedforward NN

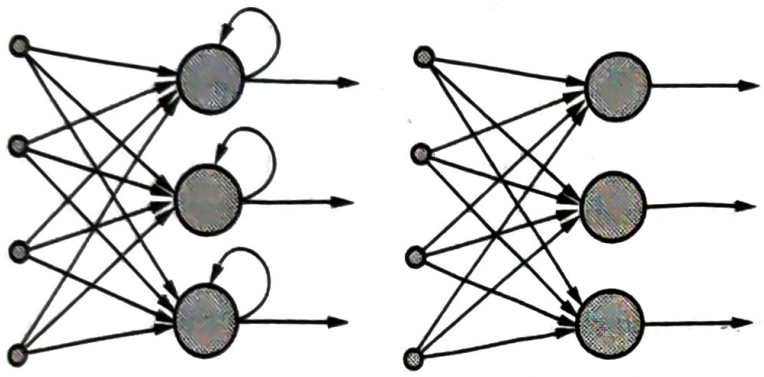
- **Physiological feedforward system :** during this, the feedforward management is epitomized by the conventional prevenient regulation of heartbeat prior to work out by the central involuntary.
- **Gene regulation and feedforward :** during this, a motif preponderantly seems altogether the illustrious networks and this motif has been shown to be a feedforward system for the detection of the non-temporary modification of atmosphere.
- **Automation and machine management :** Feedforward control may be discipline among the sphere of automation controls.
- **Parallel feedforward compensation with derivative :** This a rather new technique that changes the part of AN open-loop transfer operates of a non-minimum part system into the minimum part.

2.8.4 Difference between RNNs and feed-forward NN

- In a feed-forward neural network, the information only moves in one direction from the input layer, through the hidden layers, to the output layer. The information moves straight through the network.
- Feed-forward neural networks have no memory of the input they receive and are bad at predicting what's coming next. Because a feed-forward network only

considers the current input, it has no notion of order in time. It simply can't remember anything about what happened in the past except its training.

- In a RNN the information cycles through a loop. When it makes a decision, it considers the current input and also what it has learned from the inputs it received previously.



Recurrent neural network Feed-forward neural network

Fig. 2.8.3

❑❑❑

Unit III

3 Associative Learning

Syllabus

Introduction, Associative Learning, Hopfield network, Error Performance in Hopfield networks, simulated annealing, Boltzmann machine and Boltzmann learning, State transition diagram and false minima problem, stochastic update, simulated annealing. Basic functional units of ANN for pattern recognition tasks: Pattern association, pattern classification and pattern mapping tasks.

Contents

3.1 Associative Learning, Hopfield Networks, Simulated Annealing, Boltzman Machine

3.2 Simulated Annealing

3.3 Boltzmann Machine and Boltzmann Learning

3.4 Basic Functional Units of ANN for Pattern Recognition Tasks