R.M.K
GROUP OF ENGINEERING INSTITUTIONS

R.M.K
GROUP OF
INSTITUTIONS

R.M.K
GROUP OF INSTITUTIONS

R.M.K GROUP OF INSTITUTIONS

Please read this disclaimerbefore proceeding:

This document is confidential and intended solely for the educational purpose of RMK Group of Educational Institutions. If you have received this document through email in error, please notify the system manager. This document contains proprietary information and is intended only to the respective group / learning community as intended. If you are not the addressee you should not disseminate, distribute or copy through e-mail. Please notify the sender immediately by e-mail if you have received this document by mistake and delete this document from your system. If you are not the intended recipient you are notified that disclosing, copying, distributing or taking any action in reliance on the contents of this information is strictly prohibited.

22AI401 MACHINE LEARNING

Department : AI&DS

Batch/Year : 2022-2026 / II

Created by : Dr. G. Sangeetha & Ms. G. Mageshwari

Date : 10.1.2024

R.M.K GROUP OF INSTITUTIONS

1.TABLE OF CONTENTS

R.M.K GROUP OF INSTITUTIONS

2. COURSE OBJECTIVES

- To discuss the basics of Machine Learning and Supervised Algorithms.
- To understand the various classification algorithms.
- To study dimensionality reduction techniques.
- To elaborate on unsupervised learning techniques.
- To discuss various Graphical models and understand the basics of reinforcement learning.

SYLLABUS

L T P C

22AI401 - MACHINE LEARNING

3 0 2 4

OBJECTIVES:

- To discuss the basics of Machine Learning and model evaluation.
- To study dimensionality reduction techniques.
- To understand the various classification algorithms.
- To elaborate on unsupervised learning techniques.
- To discuss the basics of neural networks and various types of learning.

UNIT I INTRODUCTION **9+6**

Machine Learning – Types – Applications – Preparing to Model – Activities – Data – Exploring-structure of Data – Data Quality and Remediation – Data Pre-processing – Modelling and Evaluation: Selecting a Model -Training a Model – Model representation and Interpretability – Evaluating Performance of a Model – Improving Performance.

Lab Programs:

1. Implementation of Candidate Elimination algorithm
2. Implementation of ML model evaluation techniques (R-Squared/Adjusted R-Squared/Mean Absolute Error/Mean Squared Error)

Implementation of ML model evaluation techniques (Confusion Matrix/F1 Score/AUC-ROC Curve)

UNIT II FEATURE ENGINEERING AND DIMENSIONALITY REDUCTION **9+6**

Feature Engineering – Feature Transformation – Feature Subset Selection - Principle Component Analysis – Feature Embedding – Factor Analysis – Singular value decomposition and Matrix Factorization – Multidimensional scaling – Linear Discriminant Analysis – Canonical Correlation Analysis – Isomap – Locally linear Embedding – Laplacian Eigenmaps.

Lab Programs:

1. Write python code to identify feature co-relations (PCA)
2. Interpret Canonical Covariates with Heatmap
3. Feature Engineering is the way of extracting features from data and transforming them into formats that are suitable for Machine Learning algorithms. Implement python code for Feature Selection/ Feature Transformation/ Feature Extraction.
4. Mini Project – Feature Subset Selection

UNIT III SUPERVISED LEARNING **9+6**

Linear Regression -Relation between two variables – Steps – Evaluation – Logistic Regression – Decision Tree – Algorithms – Construction – Classification using Decision Tree – Issues – Rule-based Classification – Pruning the Rule Set – Support Vector Machines – Linear SVM – Optimal Hyperplane – Radial Basis Functions – Naïve Bayes Classifier – Bayesian Belief Networks.

Lab Programs:

1. Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points. Select the appropriate data set for your experiment and draw graphs.
2. Implement and demonstrate the working of the decision tree-based ID3 algorithm

Build a Simple Support Vector Machines using a data set

UNIT IV UNSUPERVISED LEARNING **9+6**

Clustering – Types – Applications - Partitioning Methods – K-means Algorithm – K-Medoids –Hierarchical methods – Density based methods DBSCAN – Finding patterns using Association Rules – Hidden Markov Model.

Lab Programs:

1. Implement a k-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions
2. Implement market basket analysis using association rules
3. Mini Project using Clustering analysis

UNIT V NEURAL NETWORKS AND TYPES OF LEARNING **9+6**

Biological Neuron – Artificial Neuron – Types of Activation function – Implementations of ANN – Architectures of Neural Networks – Learning Process in ANN – Back propagation – Deep Learning – Representation Learning – Active Learning – Instance based Learning – Association Rule Learning – Ensemble Learning Algorithm – Regularization Algorithm- Reinforcement Learning – Elements- Model-based- Temporal Difference Learning.

Lab Programs:

1. Build an ANN by implementing the Single-layer Perceptron. Test it using appropriate data sets.
2. Implement Multi-layer Perceptron and test the same using appropriate data sets.
3. Build a RBF Network to calculate the fitness function with five neurons.
4. Mini Project – Face recognition,

TOTAL: 45+30 = 75 PERIODS

OUTCOMES:

At the end of this course, the students will be able to:

CO1: Explain the basics of Machine Learning and model evaluation.
CO2: Study dimensionality reduction techniques.
CO3: Understand and implement various classification algorithms.
CO4: Understand and implement various unsupervised learning techniques.
CO5: Build Neural Networks and understand the different types of learning.

TEXT BOOKS:

1. Saikat Dutt, Subramanian Chandramouli, Amit Kumar Das, "Machine Learning", Pearson,2019. (Unit 1 – chap 1,2,3/ Unit 2 – Chap 4 / Unit 4 – 9 / Unit 5 – Chap 10, 11)

Ethem Alpaydin, "Introduction to Machine Learning, Adaptive Computation and Machine Learning Series", Third Edition, MIT Press, 2014. (Unit 2 – Chap 6 / Unit 4 – chap 8.2.3/ Unit 5 – Chap 18)

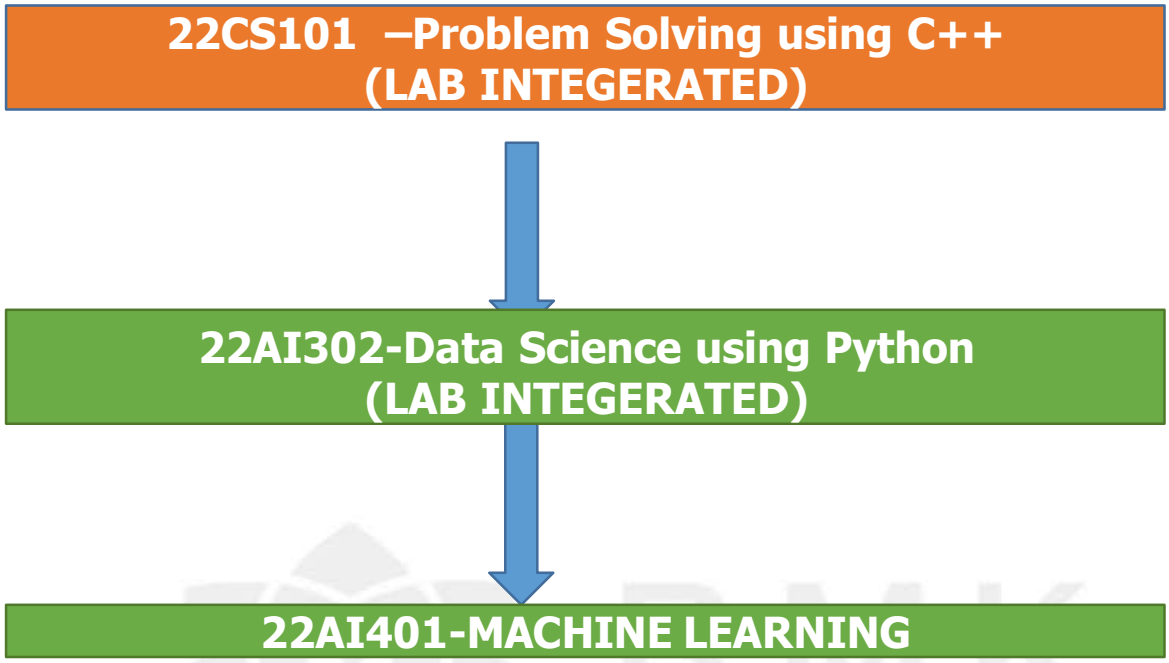
REFERENCES:

1. Anuradha Srinivasaraghavan,Vincy Joseph, "Machine Learning", First Edition, Wiley, 2019.(Unit 3 – Chap 7,8,9,10,11 / Unit 4 – 13, 11.4, 11.5,12)
2. Peter Harrington, "Machine Learning in Action", Manning Publications, 2012.
3. Stephen Marsland, "Machine Learning – An Algorithmic Perspective", Second Edition,
4. Chapman and Hall/CRC Machine Learning and Pattern Recognition Series, 2014.
5. Tom M Mitchell, "Machine Learning", First Edition, McGraw Hill Education, 2013.
6. Christoph Molnar, "Interpretable Machine Learning - A Guide for Making Black Box Models Explainable", Creative Commons License, 2020.
7. NPTEL Courses:

Introduction to Machine Learning - https://onlinecourses.nptel.ac.in/noc23_cs18/preview

3. PRE REQUISITES

PRE-REQUISITE CHART



COURSE OUTCOME

Course Code	Course Outcome Statement	Cognitive / Affective Level of the Course Outcome	Course Outcome
Course Outcome Statements in Cognitive Domain			
22AI401.1	Explain the basics of Machine Learning and model evaluation.	Apply K3	CO1
22AI401.2	Study dimensionality reduction techniques	Apply K3	CO2
22AI401.3	Understand and implement various classification algorithms.	Apply K3	CO3
22AI401.4	Understand and implement various unsupervised learning techniques.	Apply K4	CO4
22AI401.5	Build Neural Networks and understand the different types of learning	Apply K4	CO5

UNIT V
NEURAL NETWORKS AND TYPES OF LEARNING

LECTURE PLAN – UNIT V

	UNIT IV						
Sl. No	TOPIC	NO OF PERIO DS	PROPOSED LECTURE PEROID	ACTUAL LECTURE PERIOD	PERTAINING CO(s)	TAXONOMY LEVEL	MODE OF DELIVERY
1	Clustering – Types	1	21.3.2024		CO5	K4	MD1, MD 5
2,3	Applications - Partitioning Methods	1	21.3.2024		CO5	K4	MD1, MD 5
4,5	K-means Algorithm – K-Medoids	1	26.3.2024		CO5	K4	MD1. MD5
6	Hierarchical methods	1	26.3.2024		CO5	K4	MD1, MD5
7	Density based methods DBSCAN	1	27.3.2024		CO5	K4	MD1,MD5
8	Density based methods DBSCAN	1	28.3.2024		CO5	K4	MD1 , MD5
9,10	Finding patterns using Association Rules	1	28.3.2024		CO5	K4	MD1,MD5
11	Hidden Markov Model.	1	11.4.2024		CO5	K4	MD1, MD5
12	Hidden Markov Model.	1	11.4.2024		CO5	K4	MD1, MD5
13,14	Build an ANN by implementing the Single-layer Perceptron. Test it using appropriate data sets	1	15.4.2024		CO5	K4	MD1,MD5
15	Implement Multi-layer Perceptron and test the same using appropriate data sets.	1	15.4.2024		CO5	K4	MD1, MD5

LECTURE PLAN – UNIT V

ASSESSMENT COMPONENTS

- AC 1. Unit Test
- AC 2. Assignment
- AC 3. Course Seminar
- AC 4. Course Quiz
- AC 5. Case Study
- AC 6. Record Work
- AC 7. Lab / Mini Project
- AC 8. Lab Model Exam
- AC 9. Project Review

MODE OF DELEIVERY

MD 1. Oral presentation
MD 2. Tutorial
MD 3. Seminar
MD 4 Hands On
MD 5. Videos
MD 6. Field Visit

8. ACTIVITY BASED LEARNING : UNIT – V

ACTIVITY 1:

Using the Netlogo platform to run simulations of a basic neural network called the perceptron, students explore a basic, yet powerful, model of machine learning as they are challenged to understand the logic. Students engage in the perceptron model and discover a weakness of the model. The students then move on to run simulations on Netlogo with the multi-layer perceptron which overcomes the weakness in the original perceptron model.This engineering curriculum aligns to Next Generation Science Standards (NGSS).

Engineering Connection

Machine learning is the process by which a computer is able to improve its own performance by continuously incorporating new data into an existing statistical model.

Computer scientists use machine learning to automate the process of data analysis in a variety of industries. In the financial sector, engineers apply principles of machine learning to engineer systems that can gain insights into complex data sets to prevent fraud. Healthcare professionals can use machine learning to help improve diagnoses and treatments. Computer scientists must understand the logic of the particular machine learning model and mathematics involved in machine learning in order to choose the appropriate model and parameters.

In this activity, students play the role of the computer scientist by comparing the logic of two machine learning models including their strengths and weaknesses.

Learning Objectives

After this activity, students should be able to:

- Identify the parameters used in the perceptron model.
- Describe the process the perceptron uses to learn a rule.
- Identify the major weakness of the single perceptron model.

NEURAL NETWORKS AND TYPES OF LEARNING

Syllabus:

Biological Neuron – Artificial Neuron – Types of Activation function – Implementations of ANN – Architectures of Neural Networks – Learning Process in ANN – Back propagation – Deep Learning – Representation Learning – Active Learning – Instance based Learning – Association Rule Learning – Ensemble Learning Algorithm – Regularization Algorithm- Reinforcement Learning – Elements- Model-based- Temporal Difference Learning.

UNIT V-LECTURE NOTES
Neural Network

5.1 BIOLOGICAL NEURON

The human nervous system has two main parts –

- the central nervous system (CNS) consisting of the brain and spinal cord
- the peripheral nervous system consisting of nerves and gangliaoutside the brain and spinal cord.

The CNS integrates all information, in the form of signals, from the different parts of the body. The peripheral nervous system, on the other hand, connects the CNS with the limbs and organs. Neurons are basic structural units of the CNS. A neuron is able to receive, process, and transmit information in the form of chemical and electrical signals. Figure 10.1 presents the structure of a neuron. It has three main parts to carry out its primary functionality of receiving and transmitting information:

Dendrites – to receive signals from neighbouring neurons.

Soma – main body of the neuron which accumulates the signals coming from the different dendrites. It 'fires' when a sufficient amount of signal is accumulated.

Axon – last part of the neuron which receives signal from soma, once the neuron 'fires', and passes it on to the neighbouring neurons through the axon terminals (to the adjacent dendrite of the neighbouring neurons).

There is a very small gap between the axon terminal of one neuron and the adjacent dendrite of the neighbouring neuron. This small gap is known as synapse. The signals transmitted through synapse may be excitatory or inhibitory.

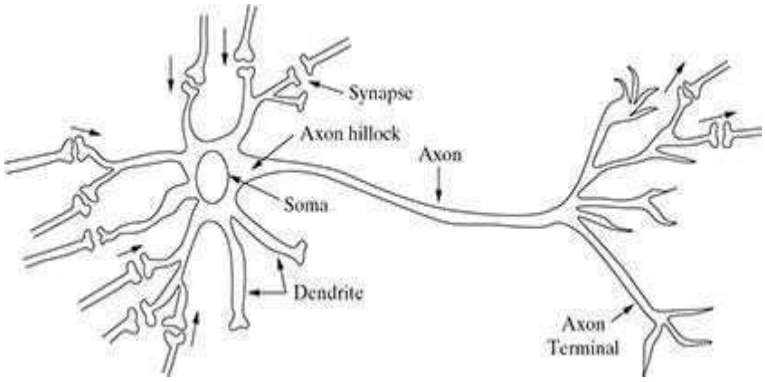


FIG. 5.1 Structure of biological neuron

5.2 ARTIFICIAL NEURON

The biological neural network has been modelled in the form of ANN with artificial neurons simulating the function of biological neurons. As depicted in Figure 10.2, input signal xi (x1, x2, …, xn) comes to an artificial neuron. Each neuron has three major components:

A set of 'i' synapses having weight wi. A signal xi forms the input to the i-th synapse having weight wi. The value of weight wi may be positive or negative. A positive weight has an excitatory effect, while a negative weight has an inhibitory effect on the output of the summation junction, ysum.

A summation junction for the input signals is weighted by the respective synaptic weight. Because it is a linear combiner or adder of the weighted input signals, the output of the summation junction, ysum, can be expressed as follows:

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

[Note: Typically, a neural network also includes a bias which adjusts the input of the activation function. However, for the sake of simplicity, we are ignoring bias for the time being. In the case of a bias 'b', the value of ysum would have been as follows:

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

A threshold activation function (or simply activation function, also called squashing function) results in an output signal only when an input signal exceeding a specific

threshold value comes as an input. It is similar in behaviour to the biological neuron which transmits the signal only when the total input signal meets the firing threshold.

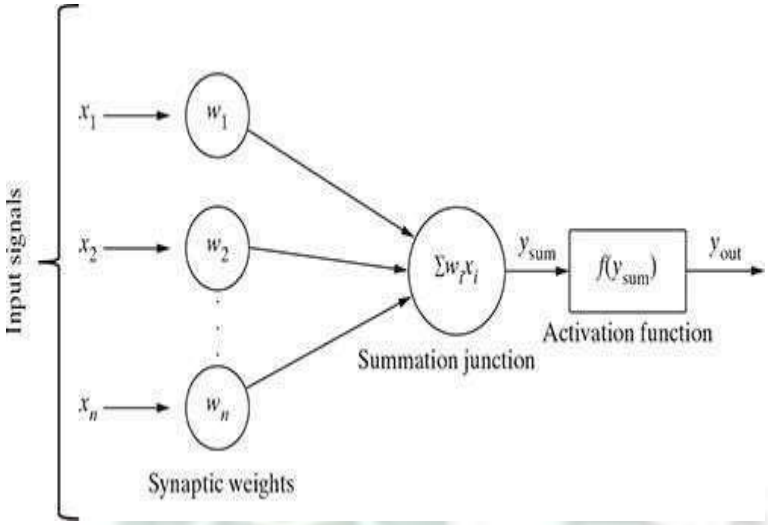


FIG. 5.2 Structure of an artificial neuron

Output of the activation function, yout, can be expressed as follows:

$$y_{out} = f(y_{sum})$$

3. TYPES OF ACTIVATION FUNCTIONS

There are different types of activation functions. The most commonly used activation functions are highlighted below.

1. Identity function

Identity function is used as an activation function for the input layer. It is a linear function having the form

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

As obvious, the output remains the same as the input.

5.3.2 Threshold/step function

Step/threshold function is a commonly used activation function. As depicted in Figure 10.3a, step function gives 1 as output if the input is either 0 or positive. If the input is negative, the step function gives 0 as output.

Expressing mathematically,

$$y_{out} = f(y_{sum}) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

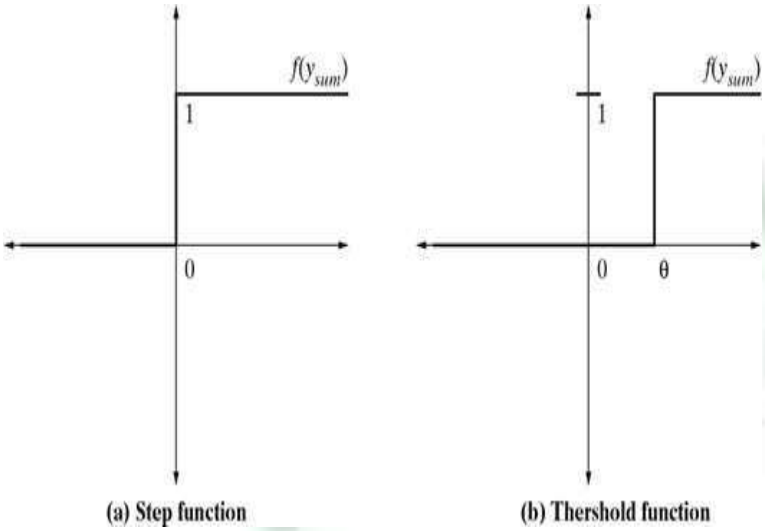


FIG. 5.3 Step and threshold functions

The threshold function (depicted in Fig. 10.3b) is almost like the step function, with the only difference being the fact that θ is used as a threshold value instead of 0. Expressing mathematically,

$$y_{out} = f(y_{sum}) = \begin{cases} 1, & x \geq \theta \\ 0, & x < \theta \end{cases}$$

5.3.3 ReLU (Rectified Linear Unit) function

ReLU is the most popularly used activation function in the areas of convolutional neural networks and deep learning. It is of the form

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

This means that f(x) is zero when x is less than zero and f(x) is equal to x when x is above or equal to zero. Figure 10.4 depicts the curve for a ReLU activation function.

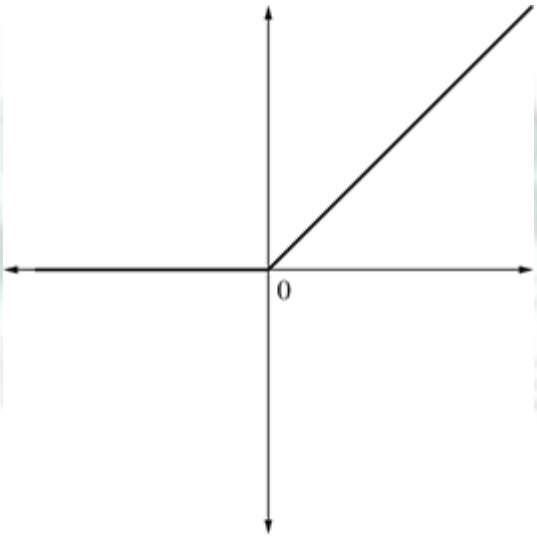


FIG. 5.4 ReLU function

This function is differentiable, except at a single point x = 0. In that sense, the derivative of a ReLU is actually a sub-derivative.

5.3.4 Sigmoid function

Sigmoid function, depicted in Figure 5.5, is by far the most commonly used activation function in neural networks. The need for sigmoid function stems from the

fact that many learning algorithms require the activation function to be differentiable and hence continuous. Step function is not suitable in those situations as it is not continuous. There are two types of sigmoid function:

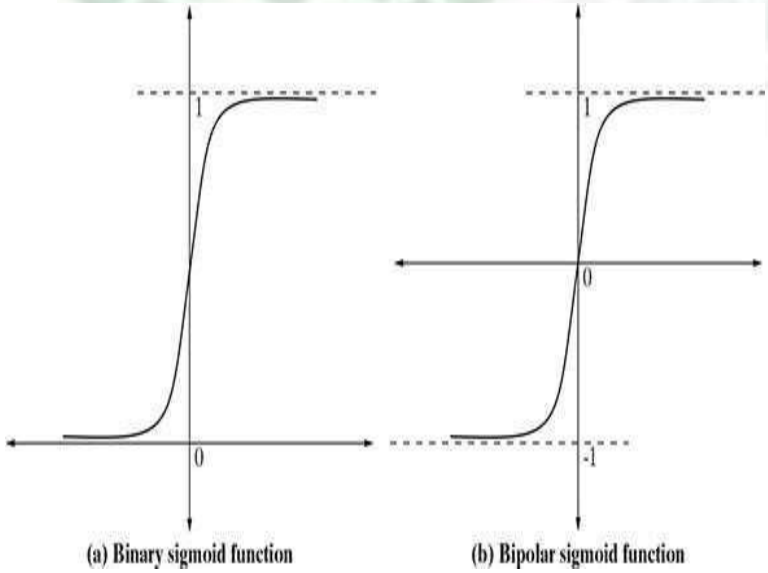
- Binary sigmoid function
- Bipolar sigmoid function

Binary sigmoid function

A binary sigmoid function, depicted in Figure 10.5a, is of the form

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

where k = steepness or slope parameter of the sigmoid function. By varying the value of k, sigmoid functions with different slopes can be obtained. It has range of (0, 1).



(a) Binary sigmoid function (b) Bipolar sigmoid function

FIG. 5.5 Sigmoid function

The slope at origin is k/4. As the value of k becomes very large, the sigmoid function becomes a threshold function.

Bipolar sigmoid function

A bipolar sigmoid function, depicted in Figure 10.5b, is of the form

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

The range of values of sigmoid functions can be varied depending on the application. However, the range of (−1,

+1) is most commonly adopted.

5.3.5 Hyperbolic tangent function

Hyperbolic tangent function is another continuous activation function, which is bipolar in nature. It is a widely adopted activation function for a special type of neural network known as backpropagation network (discussed elaborately in Section 10.8). The hyperbolic tangent function is of the form

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

This function is similar to the bipolar sigmoid function.

$$y_{out} = f(y_{sum}) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases}$$

Note that all the activation functions defined in Sections 10.4.2 and 10.4.4 have values ranging between 0 and 1. However, in some cases, it is desirable to have values ranging from −1 to +1. In that case, there will be a need to reframe the activation function. For example, in the case of step function, the revised definition would be as follows:

Works related to neural network dates long back to the 1940s. Warren McCulloch and Walter Pitts in 1943 created a computational network inspired by the biological processes in the brain. This model paved the way for neural networks.

The next notable work in the area of neural network was in the late 1940s by the Canadian psychologist Donald Olding Hebb. He proposed a learning hypothesis based on the neurons and synaptic connection between neurons. This became popular as Hebbian learning.

In 1958, the American psychologist Frank Rosenblatt refined the Hebbian concept and evolved the concept of perceptron. It was almost at the same time, in 1960, that Professor Bernard Widrow of Stanford University came up with an early single- layer ANN named ADALINE (Adaptive Linear Neuron or later Adaptive Linear Element). Marvin Lee Minsky and Seymour Aubrey Papert stated two key issues of perceptron in their research paper in 1969. The first issue stated was the

inability of perceptron of processing the XOR (exclusive-OR) circuit. The other issue was lack of processing power of computers to effectively handle the large neural networks.

From here till the middle of the 1970s, there was a slowdown in the research work related to neural networks. There was a renewed interest generated in neural networks and learning with Werbos' backpropagation algorithm in 1975. Slowly, as the computing ability of the computers increased drastically, a lot of research on neural network has been conducted in the later decades. Neural network evolved further as deep neural networks and started to be implemented in solving large-scale learning problems such as image recognition, thereby getting a more popular name as deep learning.

4. IMPLEMENTATIONS OF ANN

1. McCulloch–Pitts model of neuron

The McCulloch–Pitts neural model (depicted in Fig. 5.6), which was the earliest ANN model, has only two types of inputs – excitatory and inhibitory. The excitatory inputs have weights of positive magnitude and the inhibitory weights have weights of negative magnitude. The inputs of the McCulloch–Pitts neuron could be either 0 or 1. It has a threshold function as activation function. So, the output signal yout is 1 if the input ysum is greater than or equal to a given threshold value, else 0.

Simple McCulloch–Pitts neurons can be used to design logical operations. For that

purpose, the connection weights need to be correctly decided along with the threshold function (rather the threshold value of the activation function). Let us take a small example.

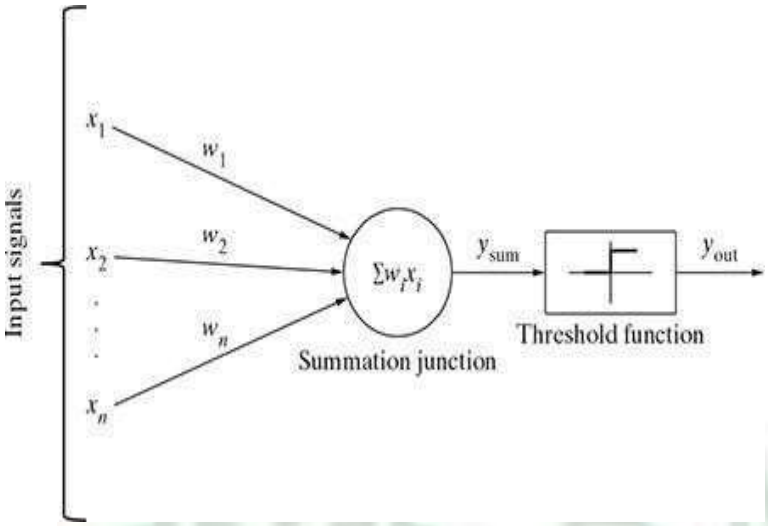


FIG. 5.6 McCulloch–Pitts neuron

John carries an umbrella if it is sunny or if it is raining. There are four given situations. We need to decide when John will carry the umbrella. The situations are as follows:

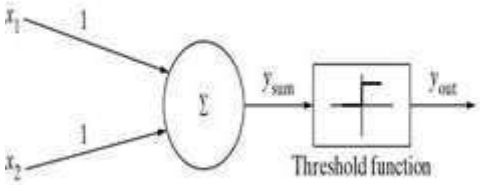
Situation 1 – It is not raining nor is it sunny. Situation 2 – It is not raining, but it is sunny. Situation 3 – It is raining, and it is not sunny.

Situation 4 – Wow, it is so strange! It is raining as well as it is sunny.

To analyse the situations using the McCulloch–Pitts neural model, we can consider the input signals as follows:

x1 → Is it raining? x2 → Is it sunny?

So, the value of both x1 and x2 can be either 0 or 1. We can use the value of both weights x1 and x2 as 1 and a threshold value of the activation function as 1. So, the neural model will look as Figure 5.7a.



(a) McCulloch–Pitts neural model

Situation	x_1	x_2	y_{sum}	y_{out}
1	0	0	0	0
2	0	1	1	1
3	1	0	1	1
4	1	1	2	1

(b) Truth table

FIG. 5.7 McCulloch–Pitts neural model (illustration)

Formally, we can say,

$$y_{sum} = \sum_{i=1}^{2} w_i x_i$$

$$y_{out} = f(y_{sum}) = \begin{cases} 1, & x \geq 1 \\ 0, & x < 1 \end{cases}$$

The truth table built with respect to the problem is depicted in Figure 5.7b. From the truth table, we can conclude that in the situations where the value of yout is 1, John needs to carry an umbrella. Hence, he will need to carry an umbrella in situations 2, 3, and 4. Surprised, as it looks like a typical logic problem related to 'OR' function? Do not worry, it is really an implementation of logical OR using the McCulloch–Pitts neural model.

5.4.2- Rosenblatt's perceptron

Rosenblatt's perceptron is built around the McCulloch– Pitts neural model. The perceptron, as depicted in Figure 5.7, receives a set of input x1, x2,…, xn. The linear combiner or the adder node computes the linear combination of the inputs applied to the synapses with synaptic weights being w1, w2, …, wn. Then, the hard limiter checks whether the resulting sum is positive or negative. If the input of the hard limiter node is positive, the output is +1, and if the input is negative, the output is −1.

Mathematically, the hard limiter input is

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

However, perceptron includes an adjustable value or bias as an additional weight w0. This additional weight w0 is attached to a dummy input x0, which is always assigned a value of 1. This consideration modifies the above equation to

$$v = \sum_{i=0}^{n} w_i x_i$$

The output is decided by the expression

$$y_{out} = f(v) = \begin{cases} +1, v > 0 \\ -1, v < 0 \end{cases}$$

The objective of perceptron is to classify a set of inputs into two classes, c1 and c2. This can be done using a very simple decision rule – assign the inputs x0, x1, x2, …, xn to c1 if the output of the perceptron, i.e. yout, is +1 and c2 if yout is −1. So, for an n-dimensional signal space, i.e. a space for 'n' input signals x0, x1, x2, …, xn, the simplest form of perceptron will have two decision regions, resembling two classes, separated by a hyperplane defined by

$$\sum_{i=0}^{n} w_i x_i = 0$$

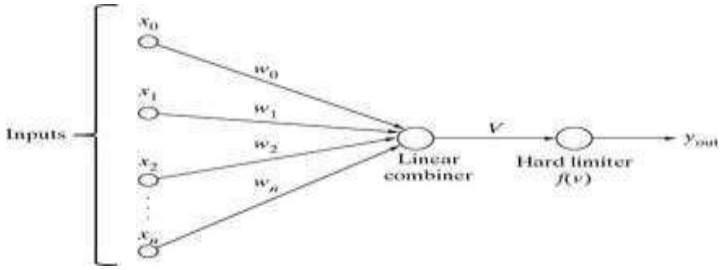


FIG. 5.8 Rosenblatt's perceptron

Therefore, for two input signals denoted by variables x1 and x2, the decision boundary is a straight line of the form

$$w_0x_0 + w_1x_1 + w_2x_2 = 0$$
$$\text{or, } w_0 + w_1x_1 + w_2x_2 = 0 [\because x_0 = 1]$$

So, for a perceptron having the values of synaptic weights w0, w1, and w2 as −2, ½, and ¼, respectively, the linear decision boundary will be of the form

$$-2 + \tfrac{1}{2}x_1 + \tfrac{1}{4}x_2 = 0$$
$$\text{or, } 2x_1 + x_2 = 8$$

So, any point (x1, x2) which lies above the decision boundary, as depicted by Figure 10.9, will be assigned to class c1 and the points which lie below the boundary are assigned to class c2.

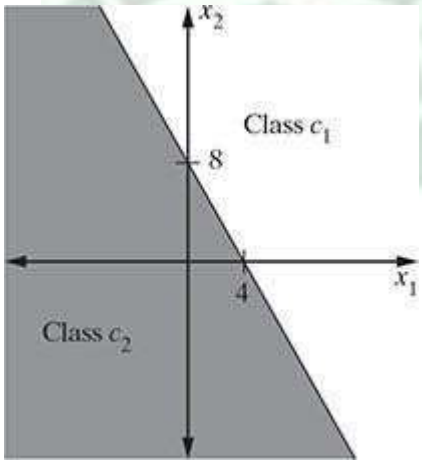


FIG. 5.9 Perceptron decision boundary

Let us examine if this perceptron is able to classify a set of points given below:

p1 = (5, 2) and p2 = (−1, 12) belonging to c1

p3 = (3, −5) and p4 = (−2, −1) belonging to c2

As depicted in Figure 5.10, we can see that on the basis of activation function output, only points p1 and p2 generate an output of 1. Hence, they are assigned to class c1 as expected. On the other hand, p3 and p4 points having activation function output as negative generate an output of 0. Hence, they are assigned to class c2, again as expected.

point	$v = \sum w_i x_i$	$y_{out} = f(v)$	Class
p_1	$-2 + (½)*5 + (¼)*2 = 1$	1	c_1
p_2	$-2 + (½)*(-1) + (¼)*12 = 0.5$	1	c_1
p_3	$-2 + (½)*3 + (¼)*(-5) = -1.75$	0	c_2
p_4	$-2 + (½)*(-2) + (¼)*(-1) = -3.25$	0	c_2

FIG. 5.10 Class assignment through perceptron

The same classification is obtained by mapping the points in the input space, as shown in Figure 5.11.

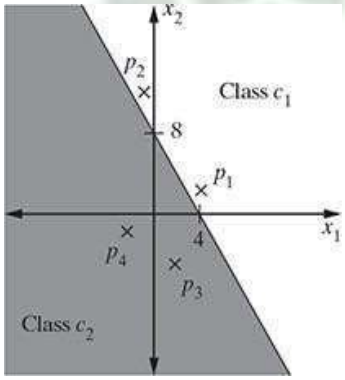


FIG. 5.11 Classification by decision boundary

Thus, we can see that for a data set with linearly separable classes, perceptrons can always be employed to solve classification problems using decision lines (for two-dimensional space), decision planes (for three- dimensional space), or decision hyperplanes (for n- dimensional space).

Appropriate values of the synaptic weights w0, w1, w2,

…, wn can be obtained by training a perceptron. However, one assumption for perceptron to work properly is that the two classes should be linearly

separable (as depicted in Figure 5.12a), i.e. the classes should be sufficiently separated from each other.

Otherwise, if the classes are non-linearly separable (as depicted in Figure 5.12b), then the classification problem cannot be solved by perceptron.

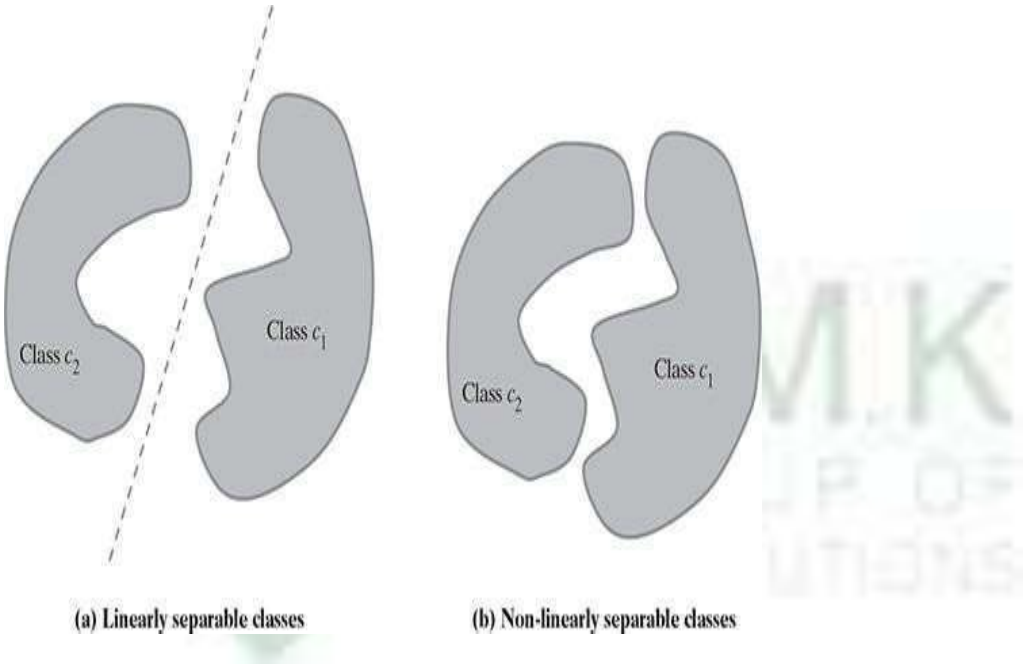


FIG. 5.12 Class separability

Multi-layer perceptron

A basic perceptron works very successfully for data sets which possess linearly separable patterns. However, in practical situation, that is an ideal situation to have. This was exactly the point driven by Minsky and Papert in their work (1969). They showed that a basic perceptron is not able to learn to compute even a simple 2-bit XOR. Why is that so? Let us try to understand.

Figure 5.13 is the truth table highlighting output of a 2-bit XOR function.

x_1	x_2	x_1 XOR x_2	Class
1	1	0	c_2
1	0	1	c_1
0	1	1	c_1
0	0	0	c_2

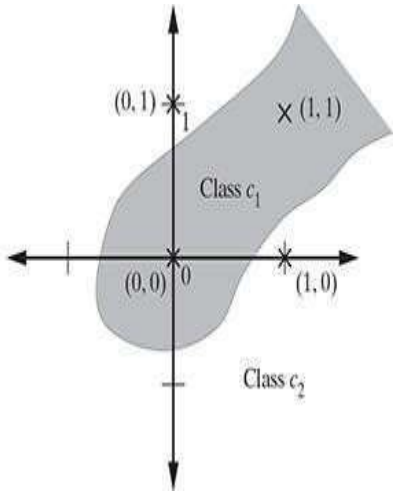


FIG. 5.13 Class separability of XOR function output

As we can see in Figure 5.13, the data is not linearly separable. Only a curved decision boundary can separate the classes properly.

To address this issue, the other option is to use two decision lines in place of one. Figure 5.14 shows how a linear decision boundary with two decision lines can clearly partition the data.

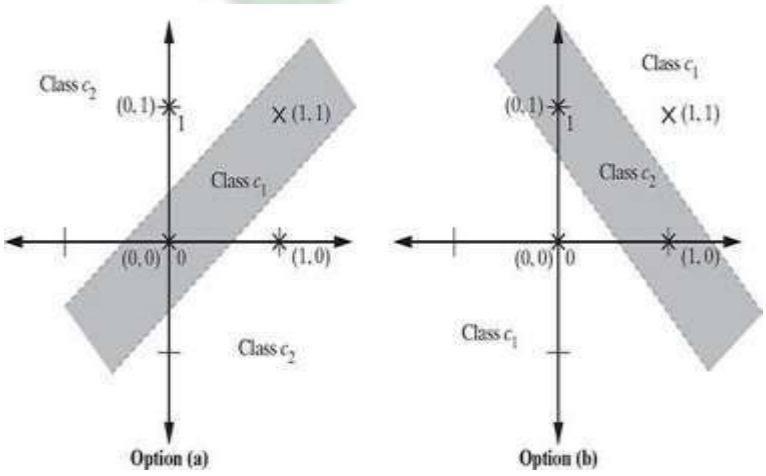


FIG. 5.14 Classification with two decision lines in XOR function output

This is the philosophy used to design the multi-layer perceptron model. The major highlights of this model are as follows:

- The neural network contains one or more intermediate layers between the input and the output nodes, which are hidden from both input and output nodes.
- Each neuron in the network includes a non-linear activation function that is differentiable.
- The neurons in each layer are connected with some or all the neurons in the previous layer.

The diagram in Figure 5.15 resembles a fully connected multi-layer perceptron with multiple hidden layers between the input and output layers. It is called fully connected because any neuron in any layer of the perceptron is connected with all neurons (or input nodes in the case of the first hidden layer) in the previous layer. The signals flow from one layer to another layer from left to right.

5.4.3 ADALINE network model

Adaptive Linear Neural Element (ADALINE) is an early single-layer ANN developed by Professor Bernard Widrow of Stanford University. As depicted in

Figure 10.16, it has only output neuron. The output value can be +1 or −1. A bias input x0 (where x0 = 1) having a weight w0 is added. The activation function is such that if the weighted sum is positive or 0, then the output is 1, else it is −1. Formally, we can say,

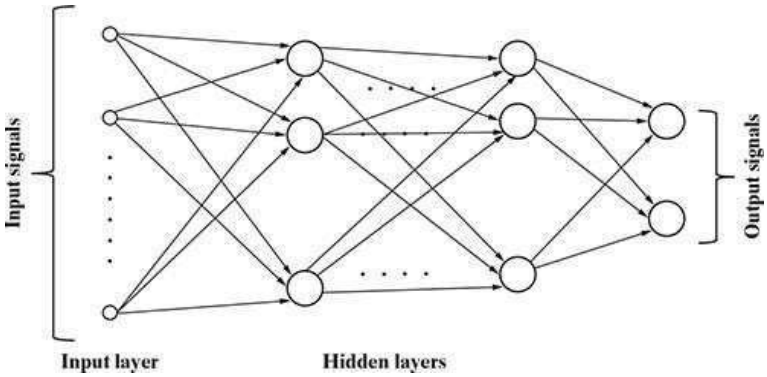
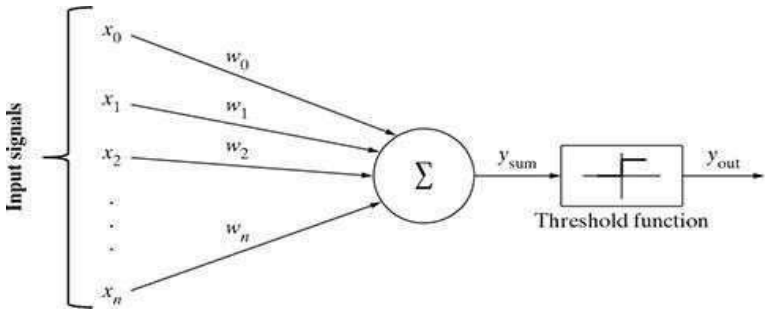


FIG. 5.15 Multi-layer perceptron



$$y_{sum} = \sum_{i=1}^{n} w_i x_i + b, \text{ where } b = w_0$$

$$y_{out} = f(y_{sum}) = \begin{cases} 1, & x \geq 1 \\ -1, & x < 1 \end{cases}$$

FIG. 5.16 ADALINE network

The supervised learning algorithm adopted by the ADALINE network is known as Least Mean Square (LMS) or Delta rule.

A network combining a number of ADALINEs is termed as MADALINE (many ADALINE). MADALINE networks can be used to solve problems related to nonlinear separability.

5.5 ARCHITECTURES OF NEURAL NETWORK

As we have seen till now, ANN is a computational system consisting of a large number of interconnected units called artificial neurons. The connection between artificial neurons can transmit signal from one neuron to another. There are multiple possibilities for connecting the neurons based on which architecture we are going to adopt for a specific solution. Some of the choices are listed below:

- There may be just two layers of neuron in the network – the input and output layer.

- Other than the input and output layers, there may be one or more intermediate 'hidden' layers of neuron.
- The neurons may be connected with one or more of the neurons in the next layer.
- The neurons may be connected with all neurons in the next layer. There may be single or multiple output signals. If there are multiple output signals, they might be connected with each other.
- The output from one layer may become input to neurons in the same or preceding layer.

5.5.1 Single-layer feed forward network

Single-layer feed forward is the simplest and most basic architecture of ANNs. It consists of only two layers as depicted in Figure 5.17 – the input layer and the output layer. The input layer consists of a set of 'm' input neurons X1, X2, …, Xm connected to each of the 'n' output neurons Y1, Y2, …, Yn. The connections carry weights w11, w12, …, wmn. The input layer of neurons does not conduct any processing – they pass the input signals to the output neurons. The computations are performed only by the neurons in the output layer. So, though it has two layers of neurons, only one layer is performing the computation. This is the reason why the network is known as single layer in spite of having two layers of neurons. Also, the signals always flow from the input layer to the output layer. Hence, this network is known as feed forward.

The net signal input to the output neurons is given by

$$y_{\text{in\_}k} + x_1 w_{1k} + x_2 w_{2k} + \ldots + x_m w_{mk} = \sum_{i=1}^{m} x_i w_{ik},$$

for the k-th output neuron. The signal output from each output neuron will depend on the activation function used.

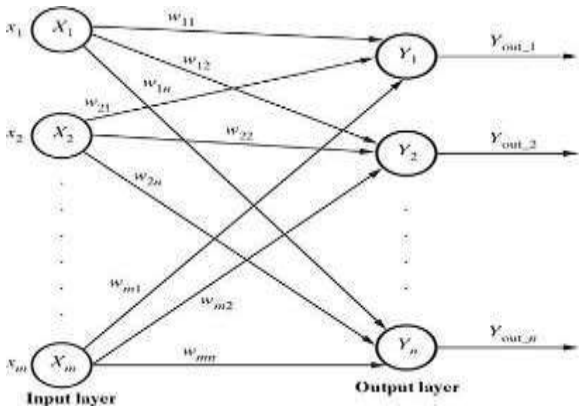


FIG. 5.17 Single-layer feed forward

5.5.2 Multi-layer feed forward ANNs

The multi-layer feed forward network is quite similar to the single-layer feed forward network, except for the fact that there are one or more intermediate layers of neurons between the input and the output layers. Hence, the network is termed as multi-layer. The structure of this network is depicted in Figure 5.18.

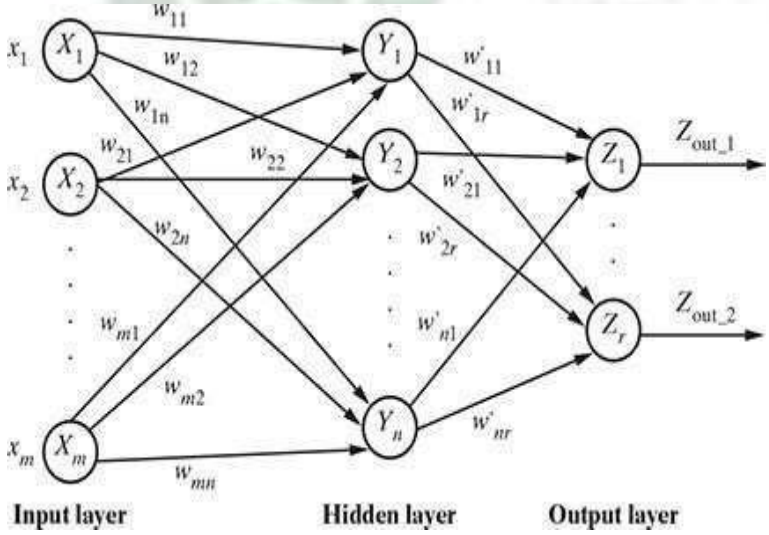


FIG. 5.18 Multi-layer feed forward

Each of the layers may have varying number of neurons. For example, the one shown in Figure 5.18 has 'm' neurons in the input layer and 'r' neurons in the output layer, and there is only one hidden layer with 'n' neurons.

The net signal input to the neuron in the hidden layer is given by

$$y_{in\_k} = x_1 w_{1k} + x_2 w_{2k} + \ldots + x_m w_{mk} = \sum_{i=1}^{m} x_i w_{ik},$$

for the k-th hidden layer neuron. The net signal input to the neuron in the output layer is given by

$$z_{in\_k} = y_{out\_1} w'_{1k} + y_{out\_2} w'_{2k} + \ldots + y_{out\_n} w'_{nk} = \sum_{i=1}^{n} y_{out\_i} w'_{ik},$$

for the k-th output layer neuron.

5.5.3 Competitive network

The competitive network is almost the same in structure as the single-layer feed forward network. The only difference is that the output neurons are connected with each other (either partially or fully). Figure 10.19 depicts a fully connected competitive network.

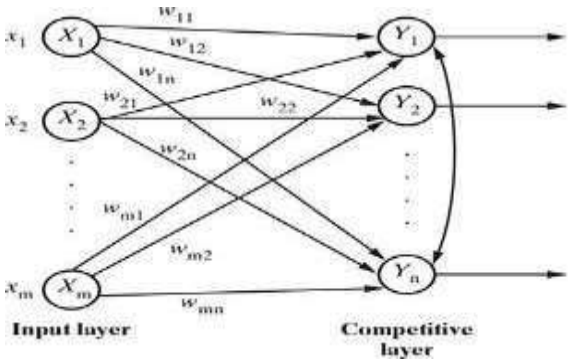


FIG. 5.19 Competitive network

In competitive networks, for a given input, the output neurons compete amongst themselves to represent the input. It represents a form of unsupervised learning algorithm in ANN that is suitable to find clusters in a data set.

5.5.4 Recurrent network

We have seen that in feed forward networks, signals always flow from the input layer towards the output layer (through the hidden layers in the case of multi-layer feed forward networks), i.e. in one direction. In the case of recurrent neural networks, there is a small deviation.

There is a feedback loop, as depicted in Figure 5.20, from the neurons in the output layer to the input layer neurons. There may also be self-loops.

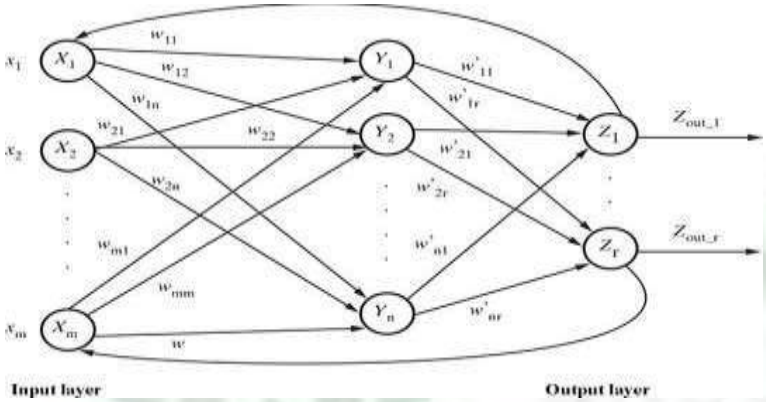


FIG. 5.20 Recurrent neural network

5.6 LEARNING PROCESS IN ANN

Now that we have a clear idea about neurons, how they form networks using different architectures, what is activation function in a neuron, and what are the different choices of activation functions, it is time to relate all these to our main focus, i.e. learning. First, we need to understand what is learning in the context of ANNs? There are four major aspects which need to be decided:

The number of layers in the network

The direction of signal flow

The number of nodes in each layer

The value of weights attached with each interconnection between neurons

Number of layers

As we have seen earlier, a neural network may have a single layer or multi-layer. In the case of a single layer, a set of neurons in the input layer receives signal, i.e. a single feature per neuron, from the data set. The value of the feature is transformed by the activation function of the input neuron. The signals processed by the neurons in the input layer are then forwarded to the neurons in the output layer. The neurons in the output layer use their own activation function to generate the final prediction.

More complex networks may be designed with multiple hidden layers between the input layer and the output layer. Most of the multi-layer networks are fully connected.

Direction of signal flow

In certain networks, termed as feed forward networks, signal is always fed in one direction, i.e. from the input layer towards the output layer through the hidden layers, if there is any. However, certain networks, such as the recurrent network, also allow signals to travel from the output layer to the input layer.

This is also an important consideration for choosing the correct learning model.

Number of nodes in layers

In the case of a multi-layer network, the number of nodes in each layer can be varied. However, the number of nodes or neurons in the input layer is equal to the number of features of the input data set. Similarly, the number of output nodes will depend on possible outcomes, e.g. number of classes in the case of supervised learning. So, the number of nodes in each of the hidden layers is to be chosen by the user. A larger number of nodes in the hidden layer help in improving the performance. However, too many nodes may result in overfitting as well as an increased computational expense.

Weight of interconnection between neurons

Deciding the value of weights attached with each interconnection between neurons so that a specific learning problem can be solved correctly is quite a difficult problem by itself. Let us try to understand it in the context of a problem. Let us take a step back and look at the problem in Section 10.6.2.

We have a set of points with known labels as given below. We have to train an ANN model using this data, so that it can classify a new test data, say p5 (3, −2).

p1 = (5, 2) and p2 = (−1, 12) belonging to c1

p3 = (3, −5) and p4 = (−2, −1) belonging to c2

When we were discussing the problem in Section 10.6.2, we assumed the values of the synaptic weights w0, w1, and w2 as −2, ½, and ¼, respectively. But where on earth did we get those values from? Will we get these weight values for every learning problem that we will attempt to solve using ANN? The answer is a big NO.

For solving a learning problem using ANN, we can start with a set of values for the synaptic weights and keep doing changes to those values in multiple iterations. In the case of supervised learning, the objective to be pursued is to reduce the number of misclassifications.

Ideally, the iterations for making changes in weight values should be continued till there is no misclassification. However, in practice, such a stopping criterion may not be possible to achieve. Practical stopping criteria may be the rate of misclassification less than a specific threshold value, say 1%, or the maximum number of iterations reaches a threshold, say 25, etc.

There may be other practical challenges to deal with, such as the rate of misclassification is not reducing progressively. This may become a bigger problem when the number of interconnections and hence the number of weights keeps increasing. There are ways to deal with those challenges, which we will see in more details in the next section.

So, to summarize, learning process using ANN is a combination of multiple aspects – which include deciding the number of hidden layers, number of nodes in each of the hidden layers, direction of signal flow, and last but not the least, deciding the connection weights.

Multi-layer feed forward network is a commonly adopted architecture. It has been observed that a neural network with even one hidden layer can be used to reasonably approximate any continuous function. The learning method adopted to train a multi-layer feed forward network is termed as backpropagation, which we will study in the next section.

5.6.1 BACK PROPAGATION

We have already seen that one of the most critical activities of training an ANN is to assign the inter- neuron connection weights. It can be a very intense work, more so for the neural networks having a high number of hidden layers or a high number of nodes in a layer. In 1986, an efficient method of training an ANN was discovered. In this method, errors, i.e. difference in output values of the output layer and the expected values, are propagated back from the output layer to the preceding layers.

Hence, the algorithm implementing this method is known as backpropagation, i.e. propagating the errors backward to the preceding layers.

The backpropagation algorithm is applicable for multi- layer feed forward networks. It is a supervised learning algorithm which continues adjusting the weights of the connected neurons with an objective to reduce the deviation of the output signal from the target output.

This algorithm consists of multiple iterations, also known as epochs. Each epoch consists of two phases –

A forward phase in which the signals flow from the neurons in the input layer to the neurons in the output layer through the hidden layers. The weights of the interconnections and activation functions are used during the flow. In the output layer, the output signals are generated.

Å backward phase in which the output signal is compared with the expected value. The computed errors are propagated backwards from the output to the preceding layers. The errors propagated back are used to adjust the interconnection weights between the layers.

The iterations continue till a stopping criterion is reached. Figure 5.21 depicts a reasonably simplified version of the backpropagation algorithm.

One main part of the algorithm is adjusting the interconnection weights. This is done using a technique termed as gradient descent. In simple terms, the algorithm calculates the partial derivative of the activation function by each interconnection weight to identify the 'gradient' or extent of change of the weight required to minimize the cost function. Quite understandably, therefore, the activation function needs

to be differentiable. Let us try to understand this in a bit modetails.

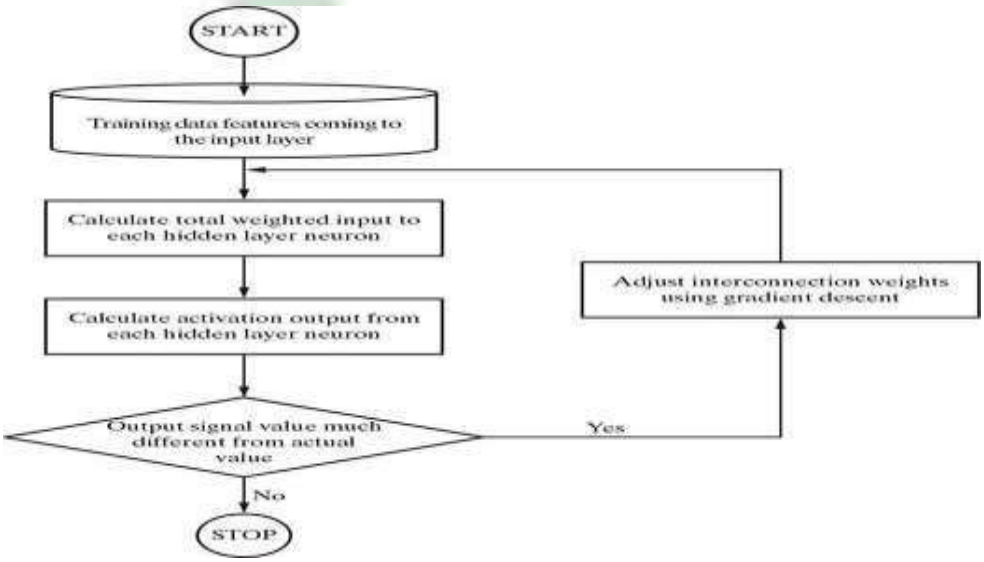


FIG. 5.21 Backpropagation algorithm

We have already seen that multi-layer neural networks have multiple hidden layers. During the learning phase, the interconnection weights are adjusted on the basis of the errors generated by the network, i.e. difference in the output signal of the network vis-à-vis the expected value. These errors generated at the output layer are propagated back to the preceding layers. Because of the backward propagation of errors which happens during the learning phase, these networks are also called back-propagation networks or simply backpropagation nets.

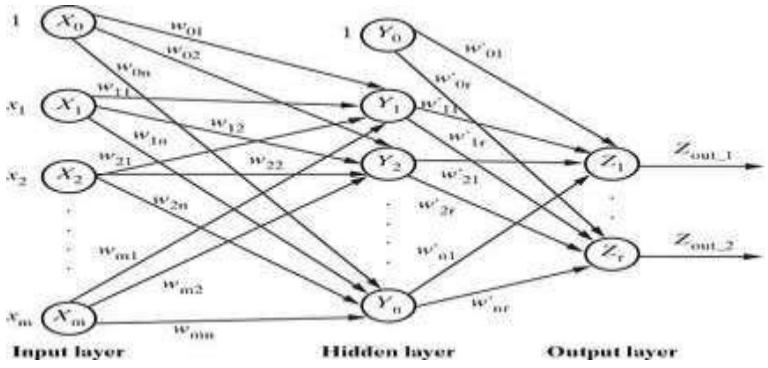
One such backpropagation net with one hidden layer is depicted in Figure 10.22. In this network, X0 is the bias input to the hidden layer and Y0 is the bias input to the output layer.

The net signal input to the hidden layer neurons is given by

$$y_{in\_k} = x_0 w_{0k} + x_1 w_{1k} + x_2 w_{2k} + \ \ldots \ + x_m w_{mk} = w_{0k} + \sum_{i=1}^{m} x_i w_{ik},$$

for the k-th neuron in the hidden layer. If fy is the activation function of the hidden layer, then

$$y_{out\_k} = f_y(y_{in\_k})$$



5.22 Backpropagation net

The net signal input to the output layer neurons is given by

$$z_{in\_k} = y_0 w'_{0k} + y_{out\_1} w'_{1k} + y_{out\_2} w'_{2k} + \ldots + y_{out\_n} w'_{nk} = w'_{0k} + \sum_{i=1}^{n} y_{out\_i} w'_{ik}$$

for the k-th neuron in the output layer. Note that the input signals to X0 and Y0 are assumed as 1. If fz is the activation function of the hidden layer, then

$$z_{out\_k} = f_z(z_{in\_k})$$

If tk is the target output of the k-th output neuron, then the cost function defined as the squared error of the output layer is given by

$$E = \frac{1}{2}\sum_{k=1}^{n}(t_k - z_{out\_k})^2$$

$$= \frac{1}{2}\sum_{k=1}^{n}(t_k - f_z(z_{in\_k}))^2$$

So, as a part of the gradient descent algorithm, partial derivative of the cost function E has to be done with respect to each of the interconnection weights w'01,w'02,

… ,w'nr. Mathematically, it can be represented as follows:

$$\frac{\partial E}{\partial w'_{jk}} = \frac{\partial}{\partial w'_{jk}}\left\{\frac{1}{2}\sum_{k=1}^{n}(t_k - f_z(z_{in\_k}))^2\right\},$$

for the interconnection weight between the j-th neuron in the hidden layer and the k-th neuron in the output layer. This expression can be deduced to

$$\frac{\partial E}{\partial w'_{jk}} = -(t_k - z_{out\_k}) \cdot f'_z(z_{in\_k}) \cdot \frac{\partial}{\partial w'_{jk}} \left(\sum_{i=0}^{n} y_{out\_i} \cdot w_{ik} \right),$$

where $f'_z(z_{in\_k}) = \frac{\partial}{\partial w'_{jk}} (f_z(z_{in\_k}))$

$$\text{or, } \frac{\partial E}{\partial w'_{jk}} = -(t_k - z_{out\_k}) \cdot f'_z(z_{in\_k}) \cdot y_{out\_i}$$

If we assume $\delta w_k = -(t_k - z_{out\_k}) \cdot f'_z(z_{in\_k})$ as a component of the weight adjustment needed for weight wjk corresponding to the k-th output neuron, then

$$\frac{\partial E}{\partial w'_{jk}} = \delta w'_k \cdot y_{out\_i}$$

On the basis of this, the weights and bias need to be updated as follows:

$$\text{For weights: } \Delta w_{jk} = -\alpha \cdot \frac{\partial E}{\partial w'_{jk}} = -\alpha \cdot \delta w'_k \cdot y_{out\_i}$$

$$\text{Hence, } w'_{jk}(\text{new}) = w'_{jk}(\text{old}) + \Delta w'_{jk}$$

$$\text{For bias: } \Delta w_{0k} = -\alpha \cdot \delta w'_k$$

$$\text{Hence, } w'_{0k}(\text{new}) = w'_{0k}(\text{old}) + \Delta w'_{0k}$$

Note that 'α' is the learning rate of the neural network.

In the same way, we can perform the calculations for the interconnection weights between the input and hidden layers. The weights and bias for the interconnection between the input and hidden layers need to be updated as follows:

$$\text{For weights: } \Delta w_{ij} = -\alpha \cdot \frac{\partial E}{\partial w_{ij}} = -\alpha \cdot \delta w_j \cdot x_{out\_i}$$

$$\text{Hence, } w_{ij}\,(\text{new}) = w_{ij}\,(\text{old}) + \Delta w_{ij}$$

$$\text{For bias: } \Delta w_{0j} = -\alpha \cdot \delta w_j$$

$$\text{Hence, } w_{0j}\,(\text{new}) = w_{0j}\,(\text{old}) + \Delta w_{0j}$$

5.6.2 DEEP LEARNING

Neural networks, as we have already seen in this chapter, are a class of machine learning algorithms. As we have also seen, there are multiple choices of architectures for neural networks, multi-layer neural network being one of the most adopted ones. However, in a multi-layer neural network, as we keep increasing the number of hidden layers, the computation becomes very expensive. Going beyond two to three layers becomes quite difficult computationally. The only way to handle such intense computation is by using graphics processing unit (GPU) computing.

When we have less number of hidden layers – at the maximum two to three layers, it is a normal neural network, which is sometimes given the fancy name 'shallow neural network'. However, when the number of layers increases, it is termed as deep neural network.

One of the earliest deep neural networks had three hidden layers. Deep learning is a more contemporary branding of deep neural networks, i.e. multilayer neural networks having more than three layers.

5.7 REPRESENTATION LEARNING

Another name for representation learning is feature learning. While many enterprises nowadays retain substantial amounts of business-related data, most of those are often unstructured and unlabelled. Creating new labels is generally a time-consuming and expensive endeavour. As a result, machine learning algorithms that can straight away mine structures from unlabelled data to improve the business performance are reasonably valued. Representation learning is one such type where the extraction from the overall unlabelled data happens using a 'neural network'. The main objective of representation learning (feature learning) is to find an appropriate representation of data based on features to perform a machine learning task. Representation learning has become a field in itself because of its popularity. Refer the chapter 4 titled 'Basics of Feature Engineering' to understand more on feature and feature selection process. An example of some of the characteristics of a triangle are its corners (or vertices), sides, and degree of angle. Every triangle has three corners and three angles. Sum of the three angles of a triangle is equal to 180°.

Consider you are tasked to classify/identify the types of triangles (Refer Table 11.1). The way you do that is to find unique characteristics of the type of triangle given as input.

The system may work something like this

Input – Triangular image

Representation – Three angles, length of sides.
Model – It gets an input representation (angles, length of sides) and applies rules as per Table 11.1 to detect the type of triangle.

Great! Now, we have a primary representational working system to detect the type of triangle.

What happens when we begin to input shapes like square, rectangle, circle, or all sort of shapes instead of a triangle? What if the image contains both a triangle and a circle? Designing to adapt to this kind of features requires deep domain expertise as we start working with real-world applications. Deep Learning goes one step further and learns/tries to learn features on its own. All we would do is to feed in an image as input and let the system learn features like human beings do.

Representation learnings are most widely used in words, speech recognition, signal processing, music, and image identification. For example, a word can have meaning based on the context.

Table 11.1 Types of Triangle

S. No	Types of triangle	Characteristics
1	Acute triangle	A triangle in which all three angles are less than 90°
2	Obtuse triangle	An obtuse triangle is a triangle in which one of the angles is greater than 90°.
3	Right triangle	A right triangle is triangle with an angle of 90°.
4	Scalene triangle	A triangle with three unequal sides.
5	Isosceles triangle	An isosceles triangle is a triangle with (at least) two equal sides.
6	Equilateral triangle	An equilateral triangle is a triangle with all three sides of equal length.
7	Equiangular triangle	An equiangular triangle is a triangle, which has three equal angles.

Generation and Recognition in Representation Learning

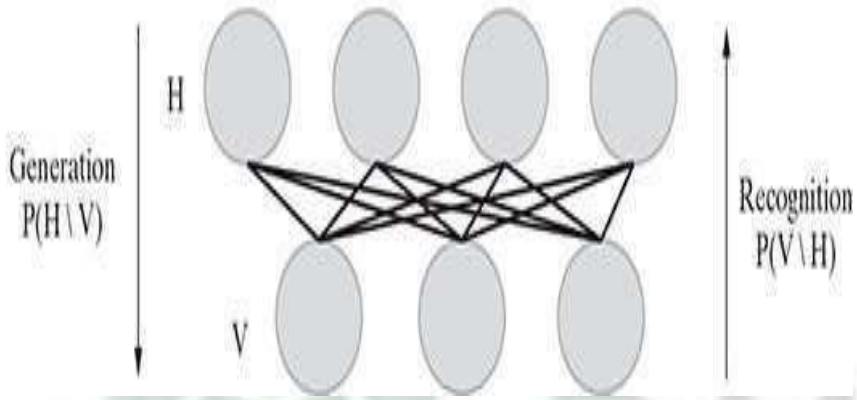


FIG. 5.22 Generation versus recognition

In Figure 5.22, V represents the input data, and H represents the causes. When the causes (H) explains the data (V) we observe, it is called as Recognition. When unknown causes (H) are combined, it can also generate the data (V), it is called as generative (or) Generation.

Representation learning is inspired by the fact that machine learning tasks such as classification regularly requires numerical inputs that are mathematically and computationally easy to process. However, it is difficult to define the features algorithmically for real-world objects such as images and videos. An alternative is to discover such features or representations just through examination, without trusting on explicit algorithms. So, representation learning can be either supervised or unsupervised.

1. Supervised neural networks and multilayer perceptron
2. Independent component analysis (unsupervised)
3. Autoencoders (unsupervised) and
4. Various forms of clustering.

8. ACTIVE LEARNING

Active learning is a type of semi-supervised learning in which a learning algorithm can interactively query (question) the user to obtain the desired outputs at new data points.

Let P be the population set of all data under consideration. For example, all male students studying in a college are known to have a particularly exciting study pattern.

During each iteration, I, P (total population) is broken up into three subsets.

1. P(K, i): Data points where the label is known (K).
2. P(U, i): Data points where the label is unknown (U)
3. P(C, i): A subset of P(U, i)that is chosen (C) to be labelled.

Current research in this type of active learning concentrates on identifying the best method P(C, i) to chose (C) the data points.

1. Heuristics for active learning

1. Start with a pool of unlabelled data P(U, i)
2. Pick a few points at random and get their labels P(C, i)
3. Repeat

Some active learning algorithms are built upon aupport vector machines (SVMs) to determine the data points to label. Such methods usually calculate the margin (W) of each unlabelled datum in P(U, i) and assume W as an n- dimensional space distance commencing covering datum and the splitting hyperplane.

Minimum marginal hyperplane (MMH) techniques consent that the data records with the minimum margin
(W) are those that the SVM is most uncertain about and therefore should be retained in P(C, i) to be labelled. Other comparable approaches, such as Maximum Marginal Hyperplane, select data with the significant W (n-dimensional space). Other approaches to select a combination of the smallest as well as largest Ws.

2. Active learning query strategies

Few logics for determining which data points should be labelled are

1. Uncertainty sampling
2. Query by committee
3. Expected model change

4. Expected error reduction
5. Variance reduction

Uncertainty sampling: In this method, the active learning algorithm first tries to label the points for which the current model is least specific (as this means a lot of confusion) on the correct output.

Query by committee: A range of models are trained on the current labelled data, and vote on the output for unlabelled data; label those points for which the 'committee' disagrees the most.

Expected model change: In this method, the active learning algorithm first tries to label the points that would most change the existing model itself.

Expected error reduction: In this method, the active learning algorithm first tries to label the points that would mostly reduce the model's generalization error.

Variance reduction: In this method, the active learning algorithm first tries to label the points that would reduce output variance (which is also one of the components of error).

5.9 INSTANCE-BASED LEARNING (MEMORY-BASED LEARNING)

In instance-based learning (memory-based learning), the algorithm compares new problem instances with instances already seen in training, which have been stored in memory for reference rather than performing explicit generalization. It is called instance-based as it constructs hypotheses from the training instances (memories) themselves. Computational complexity O(n) of the hypothesis can grow when we have some data (n). It can quickly adapt its model to previously unseen data. New instances are either stored or thrown away based on the previously set criteria.

Examples of instance-based learning include K-nearest neighbour algorithm (K-NN), kernel machines (support vector machine, PCA), and radial basis function (RBF) networks.

All these algorithms store already known instances and compute distances or similarities between this instance and the training instances to make the final decision. Instance reduction algorithm can be used to overcome memory complexity problem and overfitting problems. Let us look into radial basis function (RBF) in detail.

5.9.1 Radial basis function

Radial basis function (RBF) networks typically have three layers: one input layer, one hidden layer with a non-linear RBF activation function, and one linear output layer (Refer Fig. 11.4).

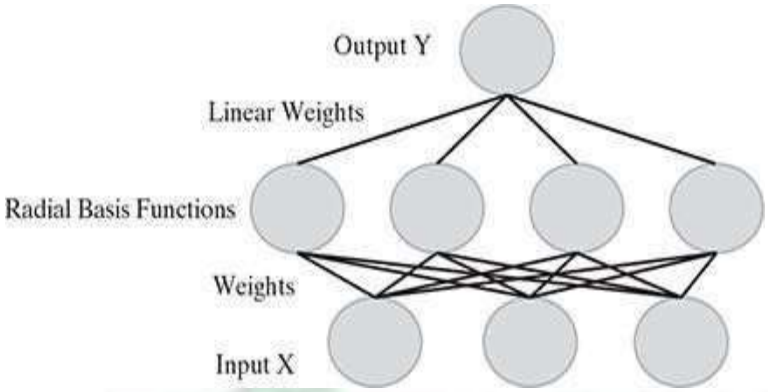


FIG. 5.23 Radial Basis Function

In the above diagram, an input vector(x) is used as input to all radial basis functions, each with different parameters. The input layer (X) is merely a fan-out layer, and no processing happens here. The second layer (hidden) performs radial basis functions, which are the non-linear mapping from the input space (Vector X) into a higher order dimensional space. The output of the network (Y) is a linear combination of the outputs from radial basis functions. If pattern classification is required (in Y), then a hard-limiter or sigmoid function could be placed on the output neurons to give 0/1 output values.

The distinctive part of the radial basis function network (RFFN) is the procedure implemented in the hidden layer. The idea is that the patterns in the input space form clusters. Beyond this area (clustering area, RFB function area), the value drops dramatically. The Gaussian function is the most commonly used radial- basis function. In an RBF network, r is the distance from the cluster centre.

Space (distance) computed from the cluster centre is usually the Euclidean distance. For each neuron that is part of the hidden layer, the weights represent the coordinates of the centre of the cluster.

Therefore, when that neuron receives an input pattern, X, the distance is found using

$$r_j = \sqrt{\sum_{i=1}^{n} (x_i - w_{ij})^2}$$

the following equation:

The variable sigma, σ, denotes the width or radius of the bell-shape and is to be determined by calculation. When the distance from the centre of the Gaussian reaches σ, the output drops from 1 to 0.6.

$$(hidden_{unit})\phi_j = \exp\left(-\frac{\sum_{i=1}^{n}(x_i - w_{ij})^2}{2\sigma^2}\right)$$

The output y(t) is a weighted sum of the outputs of the hidden layer, given by

$$y(t) = \sum_{i=1}^{n} w_i \phi(\| u(t) - c_i \|),$$

where

u(t) is the input

ϕ(.) is an arbitrary non-linear radial basis function

‖.‖ denotes the norm that is usually assumed to be Euclidean

ci are the known centres of the radial basis functions

wi are the weights of the function

In radial functions, the response decreases (or increases) monotonically with distance from a central point and they are radially symmetric. The centre, the distance scale, and the exact shape of the radial function are the necessary considerations of the defined model. The most commonly used radial function is the Gaussian radial filter, which in case of a scalar input is

$$h(x) = \exp\left(-\frac{(x - c)^2}{\beta^2}\right)$$

Its parameters are its centre c and its radius β (width), illustrates a Gaussian RBF with centre c = 0 and radius β = 1. A Gaussian RBF monotonically decreases with distance from the centre.

5.9.2 Pros and cons of instance-based learning method

Advantage: Instance-based learning (IBL) methods are particularly well suited to problems where the target function is highly complex, but can also be defined by a group of not as much of complex local approximations.

The disadvantage I: The cost of classification of new instances can be high (a majority of the calculation takes place at this stage).

Disadvantage II: Many IBL approaches usually study all characteristics of the instances leading to dimensionality-related problems.

10. ASSOCIATION RULE LEARNING ALGORITHM

Association rule learning is a method that extracts rules that best explain observed relationships between variables in data. These rules can discover important and commercially useful associations in large multidimensional data sets that can be exploited by an organization. The most popular association rule learning algorithms are Apriori algorithm and Eclat process.

1. Apriori algorithm

Apriori is designed to function on a large database containing various types of transactions (for example, collections of products bought by customers, or details of websites visited by customers frequently). Apriori algorithm uses a 'bottom-up' method, where repeated subsets are extended one item at a time (a step is known as candidate generation, and groups of candidates are tested against the data). The Apriori Algorithm is a powerful algorithm for frequent mining of itemsets for boolean association rules. Refer chapter 9 titled 'unsupervised learning' to understand more on this algorithm.

2. Eclat algorithm

ECLAT stands for 'Equivalence Class Clustering and bottom-up Lattice Traversal'. Eclat algorithm is another set of frequent itemset generation similar to Apriori algorithm. Three traversal approaches such as 'Top- down', 'Bottom-up', and Hybrid approaches aresupported. Transaction IDs (TID list) are stored here. It represents the data in vertical format.

Step 1: Get Transaciton IDs : tidlist for each item (DB scan).

Step 2. Transaciton IDs (tidlist) of {a} is exactly the list of transactions covering {a}.

Step 3. Intersect tidlist of {a} with the tidlists of all other items, resulting in tidlists of {a,b}, {a,c}, {a,d}, … = {a}–conditional database (if {a} removed).

Step 4. Repeat from 1 on {a}–conditional database 5. Repeat for all other items.

11. ENSEMBLE LEARNING ALGORITHM

Ensemble means collaboration/joint/group. Ensemble methods are models that contain many weaker models that are autonomously trained and whose forecasts are combined approximately to create the overall prediction model. More efforts are required to study the types of weak learners and various ways to combine them.

The most popular ensemble learning algorithms are

Bootstrap Aggregation (bagging) Boosting

AdaBoost

Stacked Generalization (blending) Gradient Boosting Machines (GBM) Gradient Boosted Regression Trees (GBRT) Random Forest

1. Bootstrap aggregation (Bagging)

Bootstrap Aggregation (bagging) works using the principle of the Bootstrap sampling method. Given a training data set D containing m examples, bootstrap drawing method draws a sample of training examples Di, by selecting m examples in uniform random with replacement. It comprises two phases namely Training phase and Classification Phase.

Training Phase:

1. Initialize the parameters 2. D = {Φ }
3. H = the number of classification
4. For k = 1 to h
5. Take a bootstrap sample Sk from training set S
6. Build the classifier Dk using Sk as a training set
7. D = DUDi
8. Return D

Classification Phase:

1. Run D1, D2,… Dk on the input k

2. The class with a maximum number of the vote is chosen as the label for X.

Original Sample	1	2	3	4	5	6	7	8

Bootstrap Sample 1	1	1	2	8	3	4	5	3
Bootstrap Sample 2	1	4	5	7	4	5	1	2
Bootstrap Sample 3	5	2	1	2	1	8	4	2
Bootstrap Sample 4	7	4	2	8	5	6	6	6

In the above example, the original sample has eight data points. All the Bootstrap samples also have eight data points. Within the same bootstrap sample, data points are repeated due to selection with the replacement.

5.11.2 Boosting

Just like bagging, boosting is another key ensemble- based technique. Boosting is an iterative technique. It decreases the biasing error. If an observation was classified incorrectly, then it boosts the weight of that observation. In this type of ensemble, weaker learning models are trained on resampled data, and the outcomes are combined using a weighted voting approach based on the performance of different models. Adaptive boosting or AdaBoost is a special variant of a boosting algorithm. It is based on the idea of generating weak learners and learning slowly.

5.11.3 Gradient boosting machines (GBM)

As we know already, boosting is the machine learning algorithm, which boosts week learner to strong learner. We may already know that a learner estimates a target function from training data. Gradient boosting is the boosting with Gradient.

Gradient: Consider L as a function of an ith variable, other N – 1 variables are fixed at the current point, by calculating derivative of L at that point (L is differentiable), we have – if the derivative is positive/negative – a decrease/increase value of the ith variable in order to make the loss smaller (*). Applying the above calculation for all i = 1… N, we will get N derivative values forming a vector, so-called gradient of an N-variable function at the current point.

5.12 REGULARIZATION ALGORITHM

This is an extension made to another method (typically regression method) that penalizes models based on their complexity, favouring simpler models that are also better at generalizing. Regularization algorithms have been listed separately here because they are famous, influential, and generally simple modifications made to other methods.

The most popular regularization algorithms are

- Ridge Regression
- Least Absolute Shrinkage and Selection Operator (LASSO) Elastic Net
- Least-Angle Regression (LARS)
- Elastic Net

When we are working with high-dimensional data sets with a large number of independent variables, correlations (relationships) amid the variables can be often result in multicollinearity. These correlated variables which are strictly related can sometimes form groups or clusters called as an elastic net of correlated variables. We would want to include the complete group in the model selection even if just one variable has been selected.

10.ASSIGNMENT 1- UNIT 5

1. Explain, in details, Rosenblatt's perceptron model. How can a set of data be classified using a simple perceptron? Use a simple perceptron with weights w , w , and w as −1, 2, and 1, respectively, to classify data points (3, 4); (5, 2); (1, −3); (−8, −3); (−3, 0). **(Category 1)**
2. Bird classification.

   (a) Download the dataset birds from http://www-cvr.ai.uiuc.edu/ponce grp/data/.

   (b) Use Caffe and the AlexNet pre-trained model, to classify the images in the birds dataset. Construct a confusion matrix that relates the bird classes with the 10 most frequent classes from ImageNet predicted by the model.

   (c) Use Caffe and the AlexNet pre-trained model to extract features for all the images in the bird dataset. Use the output of the 'fc6' layer. Train a linear classifier (logistic regression or linear svm) and evaluate it, using the train, validation and test partitions suggested for the dataset.

   (d) Repeat the previous step, but this time using as features the output of the 'fc7' layer. Compare and discuss. **(Category 2)**

3. Implement a simple feedforward neural network using Python and a library like TensorFlow or PyTorch to classify the Iris dataset. **(Category 3)**

4. Implement backpropagation from scratch in Python to train a simple neural network for a classification task. **(Category 4)**

5. Experiment with different activation functions in a neural network model and analyze their impact on training and performance. **(Category 5)**

11.PART A Q & A (WITH K LEVEL AND CO) UNIT 5

1. What is Neural Network? (CO5,k3)

A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

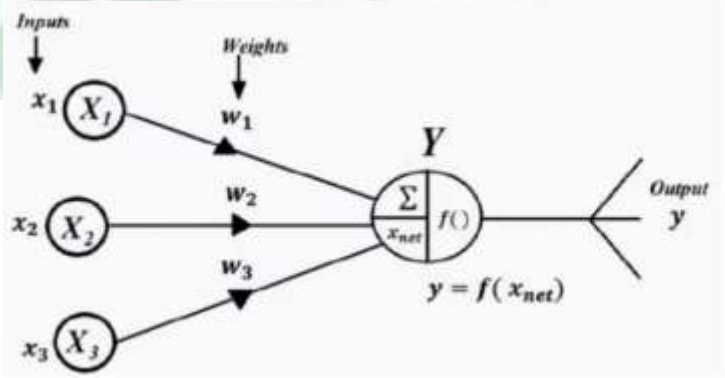
1. Knowledge is acquired by the network from its environment through a learning process.

2. Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge. (CO5,k3)

List the properties of neural network.

- Nonlinearity
- nput-Output Mapping
- Adaptivity
- Evidential Response
- Contextual Information
- Fault tolerance
- VLSI Implementability
- Uniformity of Analysis and Design
- Neurobiological Analogy

3. **Draw the structure of Artificial Neuron(CO5,k3)**



4. What are the three basic elements of the neuronal model? (CO5,k3)

The three basic elements of the neuronal model are

1. A set of synapses or connecting links, each of which is characterized by a weight or strength of its own.
2. An adder for summing the input signals,weighted by the respective synapses of the neuron.
3. An activation function for limiting the amplitude of the output of a neuron.

5. Write the Mathematical notation of Threshold activation function(CO5,k3)

1. *Threshold Function.* For this type of activation function, described in Fig. 1.8a, we have

$$\varphi(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ 0 & \text{if } v < 0 \end{cases} \qquad (1.8)$$

6. What is a signal flow graph? (CO5,k3)

A signal flow graph is a network of directed links that are interconnected at certain points called nodes. Each node has an associated node signal and each directed link originates at one node and terminates at another node.

7. What is a single layer Feed Forward network? (CO5,k4)

The simplest kind of neural network is a single-layer feed forward network, which consists of a single layer of output nodes; the inputs are fed directly to the outputs via a series of weights.

8. What is a multilayer Feed Forward network? (CO5,k4)

A multilayer feedforward neural network is an interconnection of perceptrons in which data and calculations flow in a single direction, from the input data to the outputs. The number of layers in a neural network is the number of layers of perceptrons.

9. What is a recurrent network?

A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes.

10. Define the term 'knowledge' in Neural Network. (CO5,k4)

Knowledge refers to stored information or models used by a person or machine to interpret ,predict and appropriately respond to the outside world

11. List the four rules for Knowledge Representation in an Artificial Neural Network?

Rule 1: Similar inputs from similar classes should usually produce similar representations inside the network, and should therefore be classified as belonging to the same category.

Rule 2: Items to be categorized as separate classes should be given widely different representations in the network.

Rule 3: If a particular feature is important, then there should be a large number of neurons involved in the representation of that item in the network.

Rule 4: Prior information and invariances should be built into the design of a neural network, thereby simplifying the network design by not having to learn them

12. Define the learning process in a neural network. (CO5,k4)

Learning is a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. The type
of learning is determined by the manner in which the parameter changes takes place

13.Define Nearest Neighbour Rule in memory-based learning. (CO5,k5)

The K- nearest classifier proceeds as follows,

Identify the K classified patterns that lie nearest to the test vector Xtest for some interger k.

Assign Xtest to the class that is most frequently represented in the K-nearest neighbours to Xtest (i.e.,use the majority vote to make the classification)

14. List the applications of Linear Adaptive Filters. (CO5,k5)

- Radar
- Sonar
- Communications
- Seisomology
- Biomedical signal processing

15. State the delta rule. (CO5,k5)

The delta rule is stated as:
The adjustment made to a synaptic weight of a neuronis propotional to the product of the error signal and the input signal of the synapse.

12.PART B Q s (WITH K LEVEL AND CO) UNIT 5

1.What is a Describe the structure of an artificial neuron. How is it similar to a biological neuron? What are its main components? **(CO5,k4)**

2. What are the different types of activation functions popularly used? Explain each of them. **(CO5,k4)**

3. Explain the basic structure of a multi-layer perceptron. Explain how it can solve the XOR problem. **(CO5,k4)**

4. What is artificial neural network (ANN)? Explain some of the salient highlights in the different architectural options for ANN. **(CO5,k4)**

5. Explain the learning process of an ANN. Explain, with example, the challenge in assigning synaptic weights for the interconnection between neurons? How can this challenge be addressed? **(CO5,k4)**

6. Explain, in details, the backpropagation algorithm. What are the limitations of this algorithm? **(CO5,k4)**

7. Describe, in details, the process of adjusting the interconnection weights in a multi-layer neural network. **(CO5,k4)**

8. What are the steps in the backpropagation algorithm? Why a multi-layer neural network is required? **(CO5,k4)**

9. Write short notes on any two of the following:**(CO5,k4)**
    1. Artificial neuron
    2. Multi-layer perceptron
    3. Deep learning
    4. Learning rate

10. Write the difference between (any two) **(CO5,k4)**
    1. Activation function vs threshold function
    2. Step function vs sigmoid function
    3. Single layer vs multi-layer perceptron

11. Derive primary representational working system to detect the type of triangle. **(CO5,k4)**

12. Discuss Generation and Recognition in Representation Learning**(CO5,k4)**

13. Discuss Independent component analysis (unsupervised) **(CO5,k4)**

14. Discuss Autoencoders in detail with diagram**(CO5,k4)**

15. What is active learning? Explain its heuristics**(CO5,k4)**

16. Discuss various Active learning Query Strategies**(CO5,k4)**

17. Discuss Instance-based Learning (Memory-based learning) **(CO5,k4)**

18. Discuss Radial Basis Function in detail**(CO5,k4)**

19. Discuss various Association Rule Learning Algorithm in detail**(CO5,k4)**

20. What is Ensemble Learning Algorithm? Discuss various types. **(CO5,k4)**

13. Supportive online Certification courses

1. NPTEL COURSE LINK-
   https://onlinecourses.nptel.ac.in/noc22_cs97/preview
2. COURSERA- https://www.coursera.org/learn/machine-learning

14. REAL TIME APPLICATIONS IN DAY-TO-DAY LIFE AND TO INDUSTRY

1. Optical Character Recognition (OCR) for Handwritten Text
2. Disease Diagnosis from Medical Reports Content-Based Recommender Systems
3. Semantic Segmentation of Road Object
4. Next Word Prediction
5. Time Series Prediction using LSTMs
6. Conversational Chatbot
7. Text Summarization — Summarization of Large Documents

16. PRESCRIBED TEXT BOOKS & REFERENCE BOOKS

TEXT BOOKS:

1. Saikat Dutt, Subramanian Chandramouli, Amit Kumar Das, Machine Learning, Pearson, 2019. (Unit 1 – chap 1,2,3/ Unit 2 – Chap 4 / Unit 4 – 9 / Unit 5 – Chap 10, 11)
2. Ethem Alpaydin, Introduction to Machine Learning, Adaptive Computation and Machine Learning Series, Third Edition, MIT Press, 2014. (Unit 2 – Chap 6 / Unit 4 – chap 8.2.3 / Unit 5 – Chap 18)

REFERENCES:

1. Anuradha Srinivasaraghavan, Vincy Joseph, Machine Learning, First Edition, Wiley, 2019. (Unit 3 – Chap 7,8,9,10,11 / Unit 4 – 13, 11.4, 11.5,12)
2. Peter Harrington, "Machine Learning in Action", Manning Publications, 2012.
3. Stephen Marsland, "Machine Learning – An Algorithmic Perspective", Second Edition, Chapman and Hall/CRC Machine Learning and Pattern Recognition Series, 2014.
4. Tom M Mitchell, Machine Learning, First Edition, McGraw Hill Education, 2013.
5. Christoph Molnar, "Interpretable Machine Learning - A Guide for Making Black Box Models Explainable", Creative Commons License, 2020.

15. ASSESSMENT SCHEDULE

Tentative schedule for the Assessment During 2022-2023 EVEN semester

S.NO	Name of the Assessment	Start Date	Portions
1	IAT 1	10-02-2024	UNIT 1 & 2
2	IAT 2	01-04-2024	UNIT 3 & 4
3	Model	20-04-2024	ALL 5 UNITS

17. MINI PROJECT SUGGESTION

1. Neural Network Project on Automatic Music Generation System (Category 1)

The scope for neural networks is massive. It can be used to develop basic to advanced applications in any domain. One such project we have covered is the automatic music generation system. You can make real music without any background knowledge on playing the instruments. You may develop these systems for leisure or professional music generation. You can use MIDI file data to develop these applications and can also build an LSTM model to come up with new and interesting compositions. With deep neural networks, you can program varied methods to learn and discover a wide range of patterns. These may include different music styles and harmonies. Based on the patterns, neural networks can predict the next tokens to develop rhythmic compositions. You can combine different instruments and music forms to discover interesting music pieces.

2. Neural Network Project on Human Activity Recognition for Senior Citizens (Category 2)

We have proposed this simple project idea based on neural network technology for assistive care. The application can detect human activities, such as sitting on the sofa, opening the door, closing the door, falling, and likewise. Many senior citizens now live alone and are at major health risks. For example, people infected with Dementia often forget their medications, way back to their homes, etc. Similarly, senior citizens alone at home may trip and fall. They may hurt themselves significantly. You can develop these human activity recognition systems combining a series of images and classifying the actions.

3. Reinforcement Learning: Build a neural network-based reinforcement learning agent to learn to play simple games like Tic-Tac-Toe or CartPole. Experiment with different reinforcement learning algorithms such as Q-learning or Deep Q-Networks (DQN). **(Category 3)**

4. **Handwritten Digit Recognition**: Implement a neural network to recognize handwritten digits from the MNIST dataset. You can experiment with different network architectures, activation functions, and optimization algorithms. **(Category 4)**

5. **Handwritten Digit Recognition**: Implement a neural network to recognize handwritten digits from the MNIST dataset. You can experiment with different network architectures, activation functions, and optimization algorithms. **(Category 5)**

Thank you

R.M.K
GROUP OF
INSTITUTIONS