UNIT 1

INTRODUCTION TO NEURAL NETWORKS

1.1 INTRODUCTION

Neural computers are based on the biological processess of the brain. Terms like can learn brain like, massively, parallel, learning machine and revolutionary have been used to describe neural computing. Conventional computers concentrated on emulating human thought processes, rather than actually how they are achieved by the human brain. Neural computers, however, take an alternative approach in that they directly model the biological structure of the human brain and the way it processes information. This necessitates a new kind of architecture, which like the human brain, consists of a large number of heavily interconnected processing elements operating in parallel manner. Such architectue is now both technically and commercially fearible to be developed on a standard computer and is certain to increase in general usage.

Artifical nuron networks are the result of academic investigations that use mathematical formulations to model nervous system operations. The resulting techniques are being successfully applied in a variety of everyday business applications. Neural networks are mathematical models, originally inspired by biological processes in human barin. They are constructed from a number of simple processing elements inconnected by weighted pathways to form networks. Each element computes its output as a non-linear function of its weighted inputs. When combined into networks, these processing elements can implement arbitrarily complex non-linear functions, which can be used to solve classification, prediction or optimization problems.

Neural networks (NN_S) represent a meaningfully different approach to using computers in the work place. A Neural network is used to learn patterns and relationship in data. The data may be result of a market research effort, a production process given varying operational conditions, on the decisions of a loan officer given a set of loan applications. Regardless of the specifies involved, applying a neural network is substantially different from traditional approaches. Neural networks do not require explicit coding of the problems. These advancements are due to the creation of neural network learning rules, which are the algorithms used to learn the relationships in the data. The learning rules enable the network to 'gain knowledge' from available data and apply that knowledge to assist a manager in making key decisions.

1.2 CONCEPT OF NEURAL NETWORKS

Neural networks are simplified models of the biological nervous system and therefore have drawn their motivation from the kind of computing performed by a human brain. The key element of this paradigm is the noval structure of the information processing system. It is composed of a large number of highly inconnected elements called neurons working in Union to solve specific problem. An neural network can be massively parallel and therefore is said to exhibit parallel distributed processing. Neural Networks exhibit characterstics such as mapping capabilities or pattern association, generalization, robustness, fault tolerance and parallel and high speed information processing.

Neural networks like people, learn by example. They can therefore be trained with some example of problems to acquire knowledge about it. An artifical neural networks is configured for a specific application, such as pattern recoginition or data classification, through a learning process. Learning in biological system involves adjustments to the synaptic connections that exist between the neurons.

Artifical neural networks are a type of artifical intelligence that attempts to initiate the way a human brain works. Rather than using a digital model, in which all computations manipulate zeros and ones, a neural network works by creating connections between processing elements, the computer equivalent of neurons. The organization and weights of the connection determine the output.

A neural network is a messively parallel-distributes processor that has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects:

1. Knowledge is acquired by the network through a learning process, and
2. Inter-neuron connection strengths known as synaptic weights are used to store the knowledge.

Neural networks can also be defined as parameterized computational nonlinear algorithms for data/signal/image processing. These algorithms are either implemeted on a general purpose computer or are built into a dedicated hardware.

Artificial Neural Networks thus is an information processing system. In this information processing system, the elements called as neurons, process for information. These signals are transmitted by means of communication links. The links possess an associated weight, which is multiplied along with the incoming signal for any typical neural net. The output signal is obtained by applying activations to the net input.

An artificial neuron is characterized by :

1. Architecture (connection between neurons)
2. Training or learning (determining weights on the connections)
3. Activation function.

The structure of the simple artifical neural networks is shown in fig. 1.1

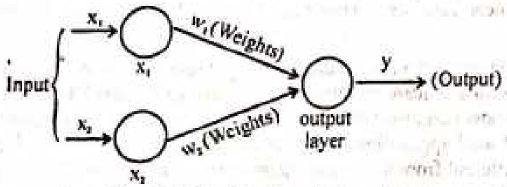


Fig. 1.1 A Simple Neural Network

Figure 1.1 shows a simple artificial neural networks with two input neurons (x_1, x_2) and one output neuron (y). The interconnected weights are given by W_1 and W_2. An artificial neuron is a P- input. Single output signal-processing element, which can be though of as a simple model of a non-branching biological neuron. In fig. 1.1, various inputs to the network are represented by the mathematical symbol, X (n). Each of these inputs are multiplied by a connection weights. These weights are represented by W (n). In the simplest case, these products are simply summed, fed through a transfer function to generate a result, and then delivered as output.

1.3 CHARACTERISTICS OF NEURAL NETWORKS

Q.1 Describe various characteristics of Neural Networks. **(KUK, May 2010)**

Ans. There are various characteristics of Neural Networks.

1. **Non-linearity–** *An artificial neuron can be linear or non-linear. A neural network made up of an inter connection of non-linear neuron is itself non linear. Non-linearity is a highly important property if the underlying physical mechanism responsible for generation of input signal is inherently non-linear.*

2. **Input-out mapping –** Supervised learning involves modification of synaptic weights of a neural networks by applying a set of labeled training samples. Each sample consists of a distinct input signal & corresponding desired response. The network is represented with an example picked at random from the set, and the synaptic weights of network are modified to minimize the difference between desired response & actual response of network produced by the input signal. The previously applied training examples may be reapplied during the training session but in a different order. Thus, the network learns from the examples by constructing an input-output mapping for the problem at hand.
3. **Adaptavity –** Neural Networks have a built in capability to adapt their synaptic weights to changes in surrounding enviornment. In particular, a neural network trained to operate in a specific environment can be easily retained to deal with minor changes in the operating environment conditions. Moreover, when it is operating in a non-stationary environment, a neural network can be designed to change its synaptic weights in real time. The natural architecture of a neural network for pattern classification, signal processing & control applications coupled with the adaptive capability of the network, make it as a useful tool in adaptive pattern classification and adaptive control. As a general rule it may be said that the more adaptive we make a system, all the time ensuring that system remains stairs, the more robust its performance is likely be when the system. is required to operate in non-stationary environment.
4. **Evidental Response –** In the context of patterm 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. The information may be used to reject ambiguous patterns & improve classification performance of the networks.
5. **Contextual information –** Knowledge is respresented by the structure & activation state of neural network. Every neurons is the network is potentially affected by the global activity of other neurons in the network. Consequently, contextual information is dealt with naturally by a neural network.
6. **Fault tolerance –** A neural network is hardware form has the potential to be fault tolerant or capable of robust computation in the sense that its performance degrades gracefully under adverse operating conditions. For example, if connection links of a neuron are damaged, recall of a stored pattern is unpaired in quality. Moreover, due to distributed nature of information stored in the network, the damage has to be extensive before the overall response of network is degraded seriously. Thus, neural network exhibits a graceful degrated in performance them atmosphere further.
7. **VLSI implementability –** The massively parallel nature of a neural network makes it potentially fast for computation of certain tasks. This same feature makes a neural network well suited for implementation using VLSI technology.

   — Beneficial virtue of VLSI is that it provide a means for capturing truly complex behaviour in a highly hierarchical fashion.
8. **Uniformity of Analysis & Design–** Basically, neural network enjoy universality as information processers. This is said in these that the same notation is used in all domains involving application of neural networks. This feature is itself in different ways :
   - Neurons in one or different from represent a common ingridient to all neural nets.
   - This commonality makes it possible to show and learning algorithm in different applications of neural nets.
   - Modular networks can be built through a frameless integrations of modules.

9. **Neurobiological Analogy –** The design of neural net is motivated by analogy with brain which is a living proof that fault tolerant parallel processing is not only physically possible but also fast & powerful. Neurobiologists look to neuron networks as a research tool for interpretation of neurobiological phenomena. While engineers looks to neurobiology for new ideas to solve problem more complex than based on conventional wired design techniques.

1.4 HISTORICAL PERSPECTIVE OF NEURAL NETWORKS

The historical perspective of the neural networks can be traced as follows:

(I) 1943- Mc Culloch and Pitts : start of the modern era of neural networks – This forms a logical calculus of neural networks. A networks consists of sufficient number of neurons (using a simple model) and properly set synapic connections can compute any computable function. A simple logic function is performed by a neuron in this case based upon the weights set in the Mc Culloch-Pitts. neuron. The arrangement of neuron in this case may be represented as a combination of logic functions. The most important feature of this type of neuron is this concept of threshold when the net input to a particular neuron is greater than the specified threshold by the user, then the neuron fires. Logic circuits are found to use this type of neurons extensively.

(II) 1949 - Hebb's book "The organization of behavior"– An explicit statement of a physiological learn rule for synptic modification was presented for the first time. Hebb proposed that the connective of the brain is continually changing as an organism learns differing functional tasks, and that neural assemblies are created by such changes.

Hebb's work was immensely influential among psychologists. The concept behind the Hebb theory is that if two neurons are found to be active simultaneously the strength of connection between the two neurons should be increased. This concept is simmilar to that of the correlation matrix learning.

(III) 1958 - Rasenblatt introduces Perceptron [Block (1990) Minsky and papert (1988)]– In perceptron network the weights on the connection paths can be adjusted. A method of iterative weight adjustment can be used in the Perceptron net. The Perceptron net is found to converge if the weights obtained allow the net to reproduce exactly all the training input and target out vector pairs.

(iv) 1960- Widrow and Hoff introduce adaline– ADALINE, abbreviated from Adaptive Linear Neuron uses a learning rule called as least Mean Square rule or Delta rule. This rule is found to adjust the weights so as to reduce the difference between the net input to the output unit and the desired output. The convergence criteria in this case are the reduction of mean square error to minimum value. This delta rule for a single layer net can be called a precursor of the backpropagation net used for multi-layer nets. The multilayer extensions of Adaline formed the Madaline {Widrow and Lehr, 1990.}

(v) 1982- John Hopfield's networks– Hopfield showed how to use "Ising spin glass" type of model to store information in dynamically stable networks. His work proved the way for physicists to enter neural modeling, thereby transforming the field of neural networks. These nets are widely used as associative memory nets. The Hopfield nets are found to be both continuous valued and discrete valued. this net provides an efficient solution for the "Travelling Sales-man Problem."

(vi) 1972- Kohonen's Self Organizing Maps (SOM) kohonen's self organizing Maps are capable of reproducing important aspects of the structure of biological neural nets. They makes use of data representation using topographic maps, which are common in the nervous systems. SOM also has a wide range of applications. It shows how the output layer can pick up the correlational structure (from the inputs) in the form of the spatical arrangement of units. These nets are applied to many recognition problems.

(vii) 1985- parker, 1986- Lecum– During this period backpropagation netpaved its way into the Neural Networks. This method propagates the error information at the output units back to the hidden units using a generalized delta rule. This net is basically a multilayer, feed forward net trained by means

of backpropagation. Originally, even though the work was performed by Parker (1986) backpropogation net emerged as the most popular learning algorithm for the training of multilayer perceptrons and has been the workhorse. For many neural network applications.

(viii) 1988- Grossberg– Grossberg developed a learning rule similar to that of Kohonen, which is widely used in the counter Propagation net. This Grossberg type to learning is also used as outstar learning. this learning occurs for all the units in a particular layer; no competition among these units is assumed.

(ix) 1987, 1990- Carpenter and Grossberg– Carpenter and Grossberg invented Adaptive Resonance Theory (ART). ART was designed for both binary inputs and the continuous valued inputs. The design for the binary inputs formed ART 1, and ART 2 came into being when the design became applicable to the continuous valued inputs. The most important feature of these nets that is that the input patterns can be presented in any order.

(x) 1988- broomhead and Lowe developed– Radial Basis Functions (RBF). This is also a multilayer net that is quiet similar to the back propagation net.

(xi) 1990- Vapnik developed the support vector machine.

1.5 APPLICATIONS AND ADVANTAGES OF NEURAL NETWORKS –

Q.2 Describe briefly some important applications of Artificial Neural Networks highlighting the type of neural network that is used in each case. (KUK, June 2008, 2012)

Q.3 Describe various advantages and applications of Neural Networks. (KUK, May 2011)

Ans. Applications of Neural Networks – Artificial neural networks have become an accepted information analysis technology in a variety of disciplines. This all resulted in a veriety of commerical applications of neural network technology. Given below are of commercial applications of neural networks technology.

Business –
1. Marketing 2. Real Estate

Document and Form Processing –
1. Machine Printed Character recognition
2. Graphics recognition
3. Hard printed character recognition
4. Cursive handwritten character recognition.

Finance Industry –
1. Market Trading 2. Fraud detection 3. Credit rating

Food Industry ;-
1. Odour/aroma analysis 2. Product development 3. Quality assurance.

Energy Industry –
1. Electrical load forecasting 2. Hydro electric dam operation 3. Natural gas

Manufacturing –
1. Process Control 2. Quality Control

Medical & Health Care Industry –
1. Image analysis 2. Drug development 3. Resource allocation.

Science & Engineeing –
1. Chemical engineering 2. Electrical engineering 3. weather forecasting

Transportation & Communication –

Some application of neural networks are –

1. **Forcasting the Behaviour of complex systems–** It is a broad application domain for neural networks. Specific examples include: electric load, forecasting, economic forecasting and

forecasting natural and physical phenomena. One of the recent applications being studied is the given flow forecasting. It is an important applications that can have significant economic impact. It can help in prediciting agricultural water supply and potential flood damage, estimating loads on bridge etc.

3. **Pattern Recognition (PR)/Image processing** – Neural networks have shows remarkable progress in the recognition of visual images, handwritten character, printed characters, speech and other PR based tasks.

3. **Control systems**– Neural networks have grained commercial ground by finding application in control systems. Dozens of computer product, especially by the Japanese companies incorporating NN technology, is a standing examples. Besides they have also been used for the control of chemical plants, robots and so on.

4. **Forecasting and Risk Assessment**– Neural networks have exhibited the capability to predict situation from past trends. They have therefore, found sample applications in areas such as meterology, stock market, banking and econometrics with high success rates.

5. **Optimization/Constraint satisfaction**– This comprises problems which need to satisfy constraints and obtain optimal solution. Examples of such problems include manufacturing scheduling, finding the shortest possible tour given a set of cities, etc. several problems of this nature arising out of industrial and manufacturing fields have found acceptable solutions using NN_s.

6. **Signal processing**– Over the past decade or so, neural network approaches have been successfully combined with other signal processing techniques to produce a wide variety of applications. It can very well be argued that the commercial success of neural networks has been form its ready incorporation into other information processing approaches, such as pattern recognition and statistical inference, as well as symbolic processing.

7. **Data Compression**– A class of neural networks called the back propagation network (BPN) is useful in addressing diverse problems requiring recognition of complex patterns and preforming non-trivial mapping function. Data compression is a common problem in today is world. Specifically, one would like to find a way to reduce the data needed to encode and reproduce accurately a moderately high resolution video image, so that these images may be transmitted over low- to-medium bandwidth communication equipment. Although there are many algorithmic approach to perform data compression, most of these are designed to deal with static data, such as ASCII text, or with display images that are fairly consistent, such as computer graphics. Because video data rarely contains regular, well-defined forms (and even less frequently contain empty space), video data compression is a difficult problem form an algorithmic viewpoint. On the hand, a neural network approach is ideal for a video data-reduction application, because BPN can be trained easily to map a set of patterns from an n-dimensional space to an m-dimensional space. Since any video image can be thought cf as a matrix of picture elements (pixels) the image can be conceptualized as vector in n-space. If we limit the video to be encoded to monochromatic, images can be represented as vectors of elements, each representing the gray scale value of a single pixel.

8. **Point Quality Inspection** – Visual inspection of painted surfaces, such automobile body panels is currently a very time consuming and labour intensive process. To reduce the amount of time required to perform this inspection, one of the major US automobile manufactures reflects a laser beam off the painted panel and on to a projection secreen. Since the light source is a coherent beam the amount of scatter observed in the reflected images of the laser provides an indication of the quality of the paint finish on the car. In the past inspection of scatter pattern would have been performed primarily by humans, because conventional computer programming techniques that could be used to automate the observation and scoring process suffered from a lack of flexibility and were not particularly robust.

By using a back propagation type of neural network to perform the quality-scoring operation, a system can be construted that captures the expertise of human inspectors and is elatively easy to maintain and update. To improve the performance of the system,algorthmic techniques can be coupled with the above approach to simplify the problem.

ADVANTAGES OF NEURAL NETWORKS

1. As seen already, neural computers have the ability to learn from experience to improve their performance and to adopt their behaviour to new and changing environment. Unlike conventional rule-based systems, neural networks are not programmed to perform a particular task using rules. Instead, they are trained on histroical data, using a learning algorithm. The learning algorithm changes the functionally to the network to suit the problem by modifying the values of the connection weights between processing elements. Once trained, the network interprets new data in a way that consistent with the experience gathered during training.
2. Neural networks can provide highly accurate and robust solutions for complex non-linear tasks such as fraud detection, business lapse/churn analysis, risk analysis and data mining. One of their main benefits is that the method for performing a task need not be known in advance; instead it is automatically inferred from the data. Once learned the method can be quickly and easily adjusted to track changes in the business environment.
3. A further advantages of neural networks over conventional rule- based systems and fuzzy systems is that, once trained, they are far more efficient in their storage requirement and operation; a single mathematical function can replace a large number or rules. An added, benefit of this more compact mathematical representation is that introduces a natural form of regularization or generalization. This makes neural systems extremely robust to noisy, imprecise or incomplete data.
4. The time needed to develop a neural application is often less than that in a conventional approach since the interaction between the analyst and the expert is minimized there are non-algorithms or rules to define. The scope and accuracy of the finished application is improved since the neural computer can be exposed to many more examples than can be assimilated by a single human.
5. Early criticisms relating to the lack of explanatory information of how a neural network performs in task have now been largely overcame. Techniques such as sensitivity analysis can be used to identify which input variables have the largest effect on a particular decision or prediction. Further more, neural networks can now be structures to incorporate prior expert knowledge and present result in a form that is meaningful to human users.

1.6 THE BIOLOGICAL PROTOTYPE AND NEURON CONCEPTS

Q.4 Describe the basic structure of biological neuron. **(KUK, May 2009)**

Q.5 Draw the structure of Biological neuron. **(KUK, May 2010)**

Ans. The biological neuron or a nerve cell consists of synapses, dendrites, the cell body, and the axon. The building blocks are discussed as

Synapses– The synapses are elementary signal processing devices.

--- A synapse is biochemical device, which converts a pre-synaptic electrical signal into a chemical signal and then back into a post-synaptic electrical signal.

--- The input pulse train has its amplitude modified by parameters stored in the synapse. The nature of this modification depends on the type of the synapse, which can be either inhibitory or excitatory.

---The Postsynaptic signals are aggregated and transferred along the dendrites to the nerve cell body.

The cell body– The cell body generates the output neuronal signal, a spine, which is transferred along the axon to the synaptic terminals of other neurons. The frequency of firing of a neuron is proportional to the total synaptic activities and is controlled by the synaptic parameters (weights). The phyramidal cell can receive 104 synaptic inputs and it can fanout the output signal to thoushands of target cells- a connectivity difficult to achieve in the artificial neural networks.

Dendrite– It receives signals from other neurons.

Soma– Sums all the incoming signals.

Axon– When a particular amount of input is received then the cell fires. It transmits signals though axon to other cell.

The fundamental processing elements of a neural network is a neuron. This building block of human awarness encompasses a few general capabilities. basically a biological neuron receives inputs from other sources combines them in some way, performs a generally nonlinear operation on the result, and the output the final result. Figure 1.2 shows the relationship of these four parts.

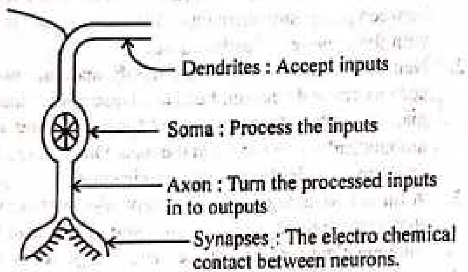


Fig. 1.2 A Biological Neuron

The properties of the biological neuron pose some features on the artificial neuron. They are:

1. Signals are received by the processing elements. This element shows the weighted inputs.
2. The weight at the receiving end has the capability to modify the incoming signal.
3. the neuron fires (transmits output), when sufficient input is obtained.
4. The output produced from one neuron may be transmitted to other neurons.
5. The processing of information is found to be local.
6. The weights can be modified by experience.
7. Neurotransmitters for the synapse may be excitatory or inhibitory.
8. Both artificial and biological neurons have inbuilt fault tolerance.

Figure 1.3 and table 1.1 indicates how the biological neural net is associated with the artificial neural

Associated Terminologies of Biological & Artificial Neural net

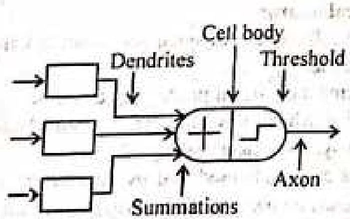


Fig. 1.3 Association of Biological Net with Artificial Net

Table 1.1

Biological Neural Network	Artificial Neural Network
Cell body	Neurons
Dendrite	Weights or interconnections
Soma	Net input
Axon	Output

1.6.1 Comparison Between the Brain and the Computer : The main differences between the brain and the computer are :

1. Biological neurons, the basic building blocks of the brain, are slower than silicon logic gates. The neurons operate in millisec, which is about six orders of magnitude slower than the silicon gates operating in the nanosecond range.
2. The brain makes up for the slow rate of operation with two factors :
   — A huge number of nerve cells (neurons) and interconnection between them. The human brain contains approximately 10^{14} to 10^{15} interconnections.
   — A function of a biological neuron seems to be much more complex than that of a logic gate.
3. The brain is very energy efficient. It consumes only about 10-16 joules per operation per second, comparing with 10-6 joules per operations per sec, for a digital computer.
4. The brain is a highly complex, non-linear, parallel information processing system. It performs tasks like pattern recognition, perception, motor control, many times faster than the fastest digital computers.
5. Consider an efficiency of the visual system which provide a representation of the environment which enables us to interact with the environment. For example, a complex task of perceptual recognition, eg. recognition of a familiar face embedded in an unfamiliar scence can be acomplished in 100-200 ms, whereas tasks of much lesser complexity can take hours if not days on conventional computers.
6. As another example consider an efficiency of the SONAR System of a bat. SONAR is an active echolocation system. A bat SONAR provides information about the distance from a target, its relative velocity and size, the size of various features of the target, and its azimuth and elevation. The complex neural computations needed to extract all this information from the target echo occur within the brain, which has the size of a plum.

1.6.2 Comparison between Artificial and Biological Neural Networks—The table below shows the major difference between the biological and the artificial neural network.

1.7 ARCHITECTURE OF NEURAL NETWORKS

Characterstic	Artificial Neural Network	Biological Neural network
speed	Neural networks are faster in processing information. The cycle time corresponding to execution of one step of a program in the central processing unit is in the range of few nano seconds.	Biological neurons are slow in processing information. The cycle time corresponding to a neural event prompted by an external stimulus occurs in a milli seconds range.
Processing	Many programs have large number of instructions, and they operate in a sequential mode one instruction after another on, a conventional computer.	Biological neural network can perform massively parallel operations. The brain possesses the capability to operations, each of them having only few steps.

Size and complexitg	These do not involve as much computational neurons. Hence it is different to perform complex pattern recognition.	Neural networks have large number of computing elements and the computing is not restricted to within neurons. The number of neurons in the brain is estimated to about 10^{11} and the total number of interconnections to be around 10^{15}. The size and complexity of connections gives the brain the power of performing complex pattern recognition tasks, which cannot be realized on a computer.
Storage	In a computer, the information is stored in the memory which is addressed by its location. Any new information in the same location destroys the old information. Hence here it is strictly replaceable.	Neural networks store information in the strengths of the interconnections. Information in the brain is adaptable, because new information is added by adjusting the interconnection strengths without destroying the old information.
Fault tolerance	Artificial nets are inherently not fault tolerant, since the information corrupted in the memory cannot be retrieved.	They exhibit fault tolerance since the information is distribute in the connections throughout the network. Even though if few connections are not working the information is still preserved due to the distributed nature of the encoded information.
Control Mechanism	There is a control unit, which monitors all the activities of computing.	There is no control for processing information in the brain. The neuron acts based on the information locally available and transmits its output of the neurons connected to it. There is no specific control mechanism external to computing task.

Q.6 What are the different types of architecture of Neural Networks? **(KUK, May 2009)**

Ans. The arrangement of neurons into layers and the pattern of connection within and in-between layer are generally called the architecture of the net. The neurons within a layer are found to be fully interconnected or not interconnected. The number of layers in the net can be defined to be the number of layers of weighted interconnected links between the particular slaps of neurons. If two layers of interconnected weights are present, then it is found to have hidden layers. There are various type of network architecture: Feed Forward, feedback, fully interconnected net, competitive net.

Artificial neural networks come in many different shapes and sizes. In feed forward architectures, the activations of the input units are set and then propagated through the network untill the values of the output units are determined. The networks acts as a vector-valued function taking one vector on the

input and returning another vector on the output. For instance, the input vector might represent the characteristics of a bank customer and the output might be a prediction of whether that customer is likely to default on a loan. or the input might represent the characteristics of a gang member and the output might be a prediction of the gang to which that person belongs.

Generally, an ANN structure can be represented using a directed graph. A graph G in an ordered 2-tuple (V,E) consisting of a set E of edges. When each edge is assigned an orientation. The graph is directed and is called directed grap or a diagraph. Figure 1.4 illustrates a digraph.

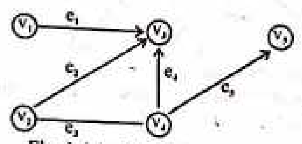


Fig. 1.4 An Example Digraph.

There are several classes of NN, Classified according to their learning mechanisms. All the classes employ the digraph structure of their representation.

1. **Single layer feed forward networks –** It is a feed forward net. This type of network comprises of two layers, namely the input and the output layer. The input layer neurons receive the input signals and the output layer neurons receive the output signals. The synaptic links carrying the weights connect every input neurons to the output neuron but not vice-versa. Such a network is said to be feed forward in type or acyclic in nature. Despite the two layer, the network is termed single layer since it is the output layer, along which performs computation. The input layer merely transmits the signals to the output. Hence, the name single layer feed forward network. Figure 1.5 illustrate an example networks.

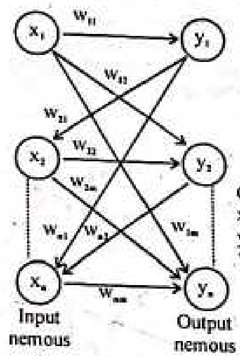


Fig. 1.5. Single layer feed forward Network

2. **Multilayer feedforward network–** This network, as its name indicates is made up of multiple layers. Thus, architecture of this class besides possessing an input and an output layer also have one or more intermediate layers called hidden layers. The computational units of the hidden layer are known as hidden neurons or hidden units. The hidden layer aids in performing useful intermediate computations before directing the input to the output layer. The input layer neurons are linked to the hidden layer neurons and the weights on these links are referred to as input hidden layer weights. Again, the hidden layer neurons are linked to the output layer neurons and corresponding weights are referred to as hidden output layer weights. A multilayer feedforward network with l input neurons m_1 neurons in the first hidden layer, m_2 neurons in the second hidden layer and n output neurons in the output layer is written as l-m_1-m_2-n

Figure 1.6 Illustrates a multilayer feedforward networks with a configuration l-m-n.

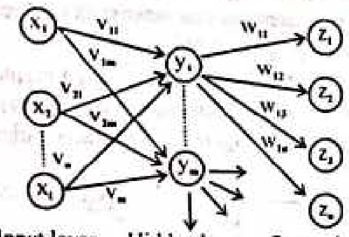


Fig. 1.6 A Multilayer Feed forward Network (l – m – n configuration)

3. **Competitive Net** – The competitive net is similar to a single layered feed forward network except that there are connections, usually negative, between the output nodes. because of these connection the output nodes tend to compete to represent the current input pattern. Sometimes the output layer is completely connected and sometimes the connection are restricted to units that are close to each other. with an appropriate learning alogwith the latter type of network can be made to organize itself topologically. In a topological map, neurons near each other represent similar input patterns. Networks of this kind have been used to explain the formation of topological maps that occur in many animal sensory system including vision, audition, touch and smell.

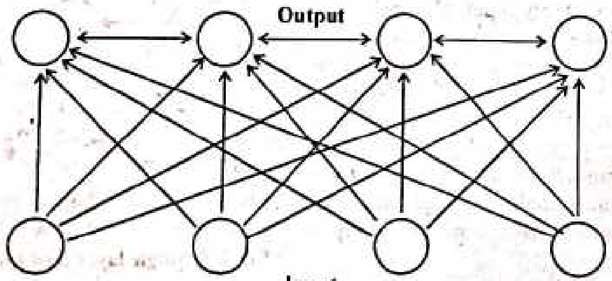


Fig. 1.7 Competitive Network

4. **Recurrent net** – The fully recurrent network is perhaps the simplest of neural network architecture. All units are connected to all other units and every unit is both an input and an output. Typically, a set of pattern is instantiated on all of the unit, one at a time. As each pattern is instantiated the weights are modified when a degraded version of one of the patterns is presented, the network attempts to reconstruct the pattern.

Recurrent networks are also useful in that they allow networks to process sequential information. Processing in recurrent networks depends on the state of the network at the last time step. Consequently the response to the current input depends on previous inputs.

These networks differ from feed forward network architectures in the sense that there is at least one feedback loop. Thus, in these networks, for example, there could exist one layer with feed back connections as shown in figure 1.8. there could also be neurons with self-feedback links, i.e. the output of a neuron is feedback into itself as input.

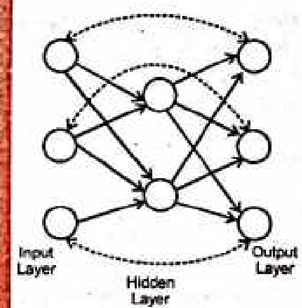


Fig. 1.8 Recurrent Net

Q.7 Explain Multilayered neural networks in brief.

Ans. See questions 5, Subheading (2)

1.8 TRAINING OF ARTIFICIAL NEURAL NETWORKS –

Q.8 What are the applications of Neural Networks? Explain the training procedure of artificial neural networks.

Ans. Applications of Neural Networks – See section 1.5

Training of Artificial Neural Networks– The method of setting the value for the weights enables the process of learning training. The process of modifying the weights in the connections between networks layers with the object of achieving the expected output is called training network. The internal process that takes place where a network is trained is called learning. Generally there types of training as follows:

1. **Supervised Training**– Supervised training is the process of providing the network with a serise of sample inputs and comparing the output with the expected responses. The training continues untill the network is able to provide the expected response. In a neural net, for a sequence of training input vector there may exist target output vectors. The weights may then be adjusted according to a learning algorithms. This process is called supervised training.

In a logic circuit, we might have the target out as + 1' if the necessary logic condition is satisified or '-1', if the logic condition is not satisfied. These type of logic nets are trained using supervised algorithm. The same criterion is applicable for patterns classification net also.

Supervised training is adopted in pattern association as well. If a neural net is trained to associate a set of input vectors with a corresponding set of output vectors, then it is called associative memory net. if the output is same as the input, then it forms auto-associative memory, if the output is different from the input then it is hetero-associative.

Some of the supervised learning algorithms include Hebb net, pattern association memory net, Back propagation net, counter propagation net, etc.

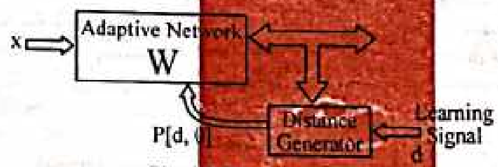


Fig. 1.9 Block diagram of supervised Learing rule.

2. **Unsupervised training–** In a neural net if for the training input vectors, the target out is not known, the training method adapted is called as unsupervised training. The net may modify the weight so that the most similar input vector is assigned to the same output unit. The net is found to form a examplar or code book vector for each cluster formed.

Unsupervised networks are for more complex and difficult to implement. It involves looping connections back into feedback layers and iterating through the process untill same sort of stable recall can be achieved. Unsupervised networks are also called self learning network of self-organizing networks because of the ability to carry out self learning. This is method adopted in the case of self-organizing feature maps, adaptive resonance theory, etc. The training process extracts the satistical properties of training set and groups similar vectors into classes.

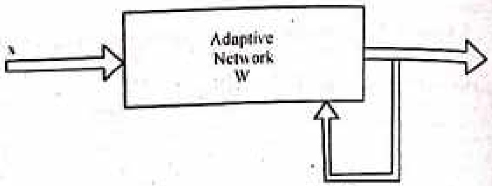


Fig. 1.10 Block diagram of unsupervised Learning rule

3. **Reinforcement training** – In this method teacher is also assumed to be present, but the right answer is not presented to the network. Instead, the network is only presented with an indication of wheather the output answer is right or wrong. The network must then use this information to improve its performance. Reinforcement learning is a very general approach to learning that can be applied when the knowledge required to apply supervised learning is not available. If sufficient information is available, the reinforcement learning can readily handle a specific problem. However, it is usually better to use other methods such as supervised and unsupervised learning because they are more direct and their underlying analytical basis is usually well understood.

Reinforcement training is related to supervised training. The output in this case may not be indicated as the desired output, but the condition whether it is 'success' (+1) or 'Failure' (0) may be indicated based on this, error may be calculated and the training process may be continued. The error signal produced from reinforcement training is found to be binary. Reinforcement learning attempts to learn the input output mapping through trial and error with a view to maximize a performance index called the reinforcement signal. The system knows whether the output is correct or not, but does not know the correct output.

Many of these learning methods are closely connected with a certain (class of) network topology. Now here is the list, just giving some names.

1. **Unsupervised learning (i.e. without a 'teacher')**
   (i) Feedback Nets
   (a) Binary Adaptive Resonance Theory (ARTI)
   (b) Analog Adaptive Resonance Theory (ART 2, ART 2-A)
   (c) Discrete Hopfield (DH)
   (d) Continuous Hopfield (CH)
   (e) Discrete Bi-directional Associative Memory (BAM)
   (f) Temporal Associative Memory (TAM)
   (g) Adaptive Bi-directional Associative Memory (ABAM)
   (h) kohonen Self-Organizing Map/Topology Preserving map (SOM/TPM)
   (i) Competitive learning
   (ii) Feedforward only Nets :

   (a) Learning matrix (LM)
   (b) Driver Reinforcement Learning (DR)
   (c) Counter propagation (CPN)

2. **Supervised learning (i.e. with a "teacher")**
   (1) Feedback Nets
   (a) Boltzmann machine (BM)
   (b) Mean field Annealing)MFT)
   (c) Recurrent Cascade Correlation (RCC)
   (d) Learning vector Quantization (LVQ)
   (e) Backpropagation through time (BPTT)
   (f) Real-time recurrent learning (RTRL)
   (ii) Feedforward only nets:
   (a) perceptron
   (b) Adaline, Madaline
   (c) Backpropagation (BP)
   (d) Cauchy machine (CM)
   (e) Artmap
   (f) cascade correlation (CasCor)

Q.9 How does the neural network learn through supervised learning. **(KUK, May 2010)**

Ans. See 1.8 Section, subheading (1)

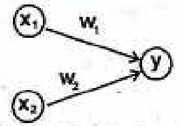
1.9 ARTIFICIAL NEURAL NETWORK TERMINOLOGY

The various terms used in the discussion of artificial neural networks are discussed below.

1. **Weights** – A neural network consists of a large number of simple processing elements called neurons. These neurons are connected to the each other by directed communication links, which are associated with weights.

'Weight is an information used by the neural net to solve a problem."

Figure 1.11 indicates a simple neural networks. The weights that carry information are denoted by w_1 and w_2. They may be fixed, or can take random values. Weights can be set to zero, or can be calculated by some methods. Initialization of weights is an important criteria in a neural net. The weight changes indicate the overall performance of the neural net. From Fig. 1.11



X_1 = Activation of neuron 1 (input signal)
X_2 = Activation of neuron 2 (input signal)
Y = Output neuron
W_1 = Weight connecting neuron 1 to output
W_2 = Weight connecting neuron 2 to output

Fig. 1.11 A Simple Newal Net

Based on all these parameters, the net input 'Net' is calculated. The Net is the summation of the products of the weights and the input signals

$$Net = x_1 w_1 + x_2 w_2$$

generally, it can be written as,

$$X_1 = A$$

$$\text{Net input} = \text{Net} = \sum_i x_i w_i$$

from the calculated net input, applying the activation functions, the output may be calculated.

2. ACTIVATION FUNCTIONS –

Q.10 What is meant by an activation function is an artificial neuron mode? Describe the various activation functions that they are employed and compare their merits and demerits. (KUK, June 2008)

Or

Q.11 What is an activation function? Explain various types of activation functions. (KUK, May 2009)

Ans. The activation function is used to calculated the output response of a neuron. The sum of the weighted input signal is applied with an activation to obtain the response. For neurons in same layer, same activation functions are used. There may be linear as well as nonlinear activation functions. The non linear activation functions are used in a multilayer Net. A few linear or non linear activation function are discussed as:

1. **Identity function** – The function is given by: $f(x) = x$; for all x

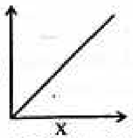


Fig. 1.12 Identity Function

2. **Binary Step Function** – The function is given by:

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

The threshold 'θ' is discussed in the following sections. Mostly single layer nets use binary step function for calculating the output from the net input. The binary step function is also called as a threshold function or Heaviside function. Figure 1.13 shows a binary step function.

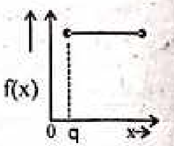


Fig. 1.13 Binary Step Function

3. **Sigmoidal Functions**– These functions are usually S-shaped curves. The hyperbolic and logistic functions are commonly used. These are used in multilayer nets like back propagation network, radial basis functions network etc. there are two main type of sigmoidal functions:

   (a) Binary Sigmoidal function– This is also called logistic function.

It ranges between 0 to 1.

$$f(x) = \frac{1}{1+\exp(-\sigma x)}$$

Where, σ is called the steepness parameter. If f(x) is differentiated we get,

$$f'(x) = \sigma f(x)[1-f(x)]$$

Figure below shows the binary sigmoidal function.

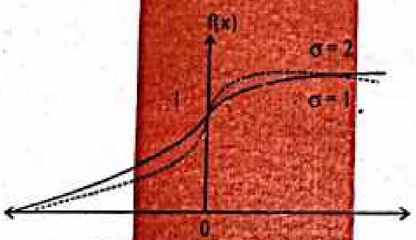


Fig. 1.14 Binary Sigmoidal funtain

(b) Bipolar Sigmoidal function– The desired range here is between +1 and -1. This function is related to the hyperbolic tangent function. The bipolar sigmoidal function is given as,

$$b(x) = 2f(x) - 1$$

$$b(x) = 2 \times \frac{1}{1+\exp(-\sigma x)} - 1$$

$$= \frac{2-1-\exp(-\sigma x)}{1+\exp(-\sigma x)}$$

$$b(x) = \frac{1-\exp(-\sigma x)}{1+\exp(-\sigma x)}$$

On differentiating the function b(x), we get

$$b'(x) = \frac{6}{2}[(1+b(x))(1-b(x))]$$

Mostly it is found that bipdar data is used. Hence this activation function is widely used. Figure 1.15 shows the bipolar sigmoidal function.

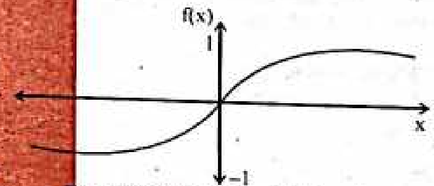


Fig. 1.15 Binary Sigmoidal funtain

4. **Signum Function**– It is also called as hard limiter activation function.

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

if f (net)= Sgn (net) (or) Sign (net)
Where Sgn (net) is the signum function.

5. **Binary Step function** – There are two types of binary step function.
(a) Bipolar binary (b) Unipolar binary
(a) Bipolar Binary function – It is called as hardlimiter function

$$\text{Net} = w_1x_1 + w_2x_2 + \ldots + w_{n-1}x_{n-1} + w_nx_n - T$$
$$\text{Net} = w_1x_1 + w_2x_2 + \ldots - T$$
$$\text{Net value} \quad = u - T$$

Take threshold as T or O notation

$$f(\text{net}) = \begin{cases} +1 & u \geq T \\ -1 & u < T \end{cases}$$

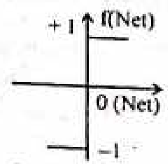


Fig. 1.16 Bipolar Binary function

Fig. 1.16 shows the bipolar binary function

(b) unipolar binary function–

$$f(\text{net}) = \begin{cases} +1 & \text{net} \geq 0 \\ 0 & \text{net} < 0 \end{cases} \text{ or}$$

$$f(\text{net}) = \begin{cases} +1 & \text{net} \geq T \\ 0 & \text{net} < T \end{cases}$$

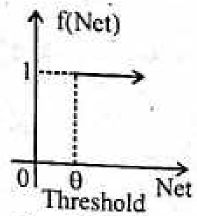


Fig. 1.17 Binary unipolar function

6. **Calculation of Net Input using matrix multiplication Method–** If the weights are given as, W= (wij) in a matrix form, The net input to output unit y_j is given as the dot product of the input vectors $x = [x_1 \text{-} x_i .. x_n]$ and Wj [ith column of the weight vector matrix].

$Yinj = x_i w_j$

$$Yinj = \sum_{i=1}^{n} x_i w_{ij}$$

Hence net input can be calculated using matrix multiplication

1. **Bias –** A bias acts exactly as a weight on a connection from a unit whose activation is always 1. increasing the bias increases the net input to the (b=Wo). A simple neural net with the bias included.

The bias improves the performance of neural networks. Similar to initialization of weight bias should also be initialization either to 0, or to any specified value, based on the neural net. If bias is present, then net input is calculated as,

$$Net = b + \sum_i xiwi$$

Where, Net - net input

The Activation of the unit is always

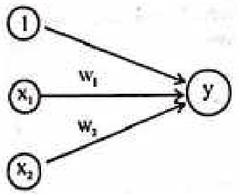


Fig. 1.16 A simple Net with Bias included

b - bias

X_i - Input from neuron i

W_i - Weight of the neuron i to the output neuron.

Hence the activation function is obtained as,

$$f(net) = \sum \begin{matrix} +1; \text{if net} \geq 0; \\ -1; \text{if net} < 0; \end{matrix}$$

if bias is in included.

8. **Threshold–** The threshold 'θ' is a factor which is used in calculating the activations of the given net. Based on the value of threshold the output may be calculated, i.e. the activation function is based on the value of θ.

For example, the activation functions may be,

$$\text{(i)}\ y = f(Net) = \begin{vmatrix} +1; \text{if net} \geq \theta; \\ -1; \text{if net} \leq \theta; \end{vmatrix}$$

$$\text{(ii)}\ yj = f(Net) = \begin{vmatrix} 1 \text{ if } yinj > \theta j \\ yj \text{ if } yinj = \theta j \text{ used for a bidirectional associative memory net} \\ \div 1 \text{ if } yinj < \theta j \end{vmatrix}$$

Hence, θ and θj indicate the thresholds, due to which the systems response is calculated. The threshold value is defined by the user.

1.10 NOTATIONS USED FOR NEURAL NETWORK

The following notations ae used to explain about the concept of artificial neural network–

x_i - Activation of Unit xi. For input Unit x_i, x_i the input signal.

y_j - Activation of Unit y_j

$$y_j = f(y_{inj})$$

w_{ij} - weight connecting uint x_i to y_j

b_j - bias on Unit y_j

y_{inj} - net input calculated on Unit y w = (εw_{ij})

w - Entire weight matrix, w = (εw_{ij})

w_{ij} - vector of weights $w_{ij} = (w_{ij}, w_{2j} ... w_{nj} T)$ for j th column of weight matrix.

$\|X\|$ - Norm or magnitude of vector x.

θ_j - Threshold for activation of neuron yj.

S - training input vector $S = (S_1 ...... S_i ..... S_n)$

t - training (or target) output vector $t = (t_1 .... t_j .... t_m)$

x - Input vector that has to respond to

$$X = (X_1 ...... X_i .... X_n)$$

Δw_{ij} - change in weight wij

$$\Delta w_{ij} = [w_{ij}(new) - w_{ij}(old)]$$

α - Learning rate. This is used for the adjustment of weights at each step of training.

1.11 MCCULLOCH - PITTS NEURON MODEL

The first formal definition of a synthetic neuron model based on the highly simplification considerations of the biological model was formulated by Warren Mc Culloch and Walter Pitts in 1943. The Mc Culloch- Pitts model of a neuron is characterized by its formalism, elegant and precise mathematical definition.

Mc Culloch- Pitts neuron allows binary 0 or 1 states only, i.e. it is being binary activated. These neurons are connected by direct weighted path. The connected path can be excitatory or inhibitory. Excitatory connections have positive weights and inhibity connections have negative weight. There will be same weight for the excitatory connection entering into a particular neuron. The neuron is associated with the threshold value. The neuron fires if the net input to the neuron is greater than the threshold. The threshold is set so that the inhibition is absolute, because, non zero inhibitory input will prevent the neuron from firing. It takes only one time step for a signal to pass over one connection Link.

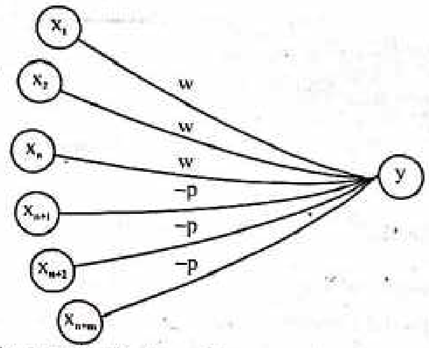


Fig. 1.17 Architecture of Mc Culloch-Pitts Neuron

Architecture – The architectue of the Mc Culloch Pitts neuron is shown in fig. 1.17

'y' is the Mc Culloch-Pitts neuron, it can receive signal from any number of other neurons. The connection weights from $x_1, ... x_n$ are excitatory, denoted by 'w' and the connection weight from $x_{n+1} .... x_{n+m}$ are inhibitory denoted by '-p'. The Mc Culloch- Pitts neuron y has the activation function,

$$f(y_{in}) = \begin{Bmatrix} 1 \text{ if } y_{-in} \geq \theta \\ 0 \text{ if } y_{-in} < \theta \end{Bmatrix}$$

where θ is the threshold y_{-in} is the total net input signal received by neuron y.
The threshold θ should satisfy the relation

$$\theta > nw - P$$

This is the condition for absolute inhibition. The Mc Culloch-Pitts neuron will fire if it reveives k or more excitatory inputs and no inhibitory inputs, where

$$kw \geq \theta > (k-1)\,w$$

Example 1.1 Generate the output of logic AND functions by Mc Culloch-Pitts neuron model.
Solution : The AND function returns a true value only if both the inputs are true, else it returns a false value '1' represents true value and '0' represent false value.

The truth table for AND function is

x_1	x_2	y
1	1	1
1	0	0
0	1	0
0	0	0

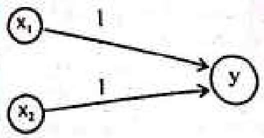


Fig. 1.18 Mc Culloch-Pitts Neuron to perform logical AND function

A Mc culloch Pitts neuron to implement AND function is shown in fig. 1.19. The threshold on unit y is 2.

The output y is

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

The net input is given by

$$y_{in} = \Sigma \text{ weights} * \text{input}$$
$$y_{in} = 1 * x_1 + 1 * x_2$$
$$y_{in} = x_1 + x_2$$

From this the activations of output neuron can be found

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

Now Present the inputs

(i) $x_1 = x_2 = 1$, $\quad f_{in} = x_1 + x_2 = 1 + 1 = 2$
$y = f(y_{in}) = 1$ since $y_{in} = 2$

(ii) $x_1 = 1, x_2 = 0$, $\quad x_{in} = x_1 + x_2 = 0 + 1 = 1$
$y = f(y_{in}) = 0$ since $y_{in} = 1 < 2$

This is same when $x_1 = 0$ and $x_2 = 1$

(iii) $x_1 = 0, x_2 = 0$, $\quad y_{in} = x_1 + x_2 = 0 + 0 = 0$
Hence, $y = f(y_{in}) = 0$ since $f_{in} = 0 < 2$

Example 1.2 Generate the function using Mc Culloch-Pitts neuron model.
Solution : The OR function returns a high ('1') if any one of the input is high, return a low ('0') if none of the inputs is high. The truth table for OR function is

x_1	x_2	y
1	1	1
1	0	1
0	1	1
0	0	0

Fig. 1.19 Mc Culloch-Pitts Neuron for OR Function

A Mc Culloch Pitts neuron for OR function is shown in figure 1.19. The threshold for the unit is 3.
The net input is calculated as

$$y_{in} = 3x_1 + 3x_2$$

The output is given by

$$y = f(y_{in}) = \begin{bmatrix} 1 \text{ if } y_{-in} \geq 3 \\ 0 \text{ if } y_{-in} < 3 \end{bmatrix}$$

Presenting the input.

(i) $x_1 = x_2 = 1$, $\quad y_{in} = 3x_1 + 3x_2 = 1 + 1 = 2$
$= 3 \times 1 + 3 \times 1 = 6 > \text{threshold } 3.$

Hence $y = 1$

(ii) $x_1 = 1, x_2 = 0$
$y_{in} = 3x_1 + 3x_2$
$= 3 \times 1 + 3 \times 0 = 3 = \text{threshold}$

Applying activation formula
$y = f(y_{in}) = 1$

This is also the case when $x_1 = 0, x_2 = 1$

(iii) $x_1 = x_2 = 0$,
$y_{in} = 3x_1 + 3x_2 = 3 \times 0 + 3 \times 0 = 0 < \text{Threshold}$

Hence, Output $y = 0$

Example 1.3 Realize NOT function using MC Culloch-Pitts neuron model.
Solution : The NOT function returns a true value ('1') if the input is false ('0') and returns a false value ('0') if the input is true ('1')

The truth table for NOT function is

x_1	y
1	0
0	1

Fig. 1.20 McCulloch-Pitts Neuron of NOT function

The Mc Culloch - Pitts neuron for this function is given in fig. 1.20. The threshold for unit y is 1.
The net input is

$$y_{in} = x.w$$

since $w = 1, y_{in} = x$.

The output activation is given by.

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

Presenting the input

(i) $x_1 = 1$, $\quad y_{in} = 1$
Applying activation $y = f(y_{in}) = 0$

(ii) $x_1 = 0$, $\quad y_{in} = 0$
Applying activation $y = f(y_{in}) = 1$

Example 1.4 Realize the Exclusive - OR function using Mc Culloch -Pitts neuron.
Solution : XOR function returns a true value if exactly one of the input value is true; otherwise it returns the response as false.

The truth table for XOR function is,

x_1	x_2	y
1	1	0
1	0	1
0	1	1
0	0	0

The Mc Culloch - Pitts neuron model for this is given in figure 1.21. The threshold of unit y is 1

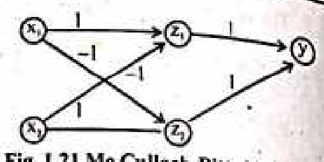


Fig. 1.21 Mc Culloch-Pitts Neuron for XOR Function

With one layer alone, it was not able to predict the value of the threshold for the neuron to fire, hence another layer is introduced

x_1XOR x_2 = (x_1 AND NOT x_2) or (x_2 AND NOT x_1)

x_1XOR x_2 = Z_1 OR z_2

Where z_1 = x_1 AND NOT x_2

and z_2 = x_2 AND NOT x_1

The activation of z_1 and z_2 are given as,

$$z_1 = (z_{in} - 1) = \begin{cases} 1 \text{ if } z_{in-1} \geq 1 \\ 0 \text{ if } z_{in-1} < 1 \end{cases}$$

$$z_2 = (z_{in} - 2) = \begin{cases} 1 \text{ if } z_{in-2} \geq 1 \\ 0 \text{ if } z_{in-2} < 1 \end{cases}$$

The calculation of net input and activation of z_1 and z_2 are shown below :

z_1 = (x_1 AND NOT x_2) $Z_{in-1} = x_1w_1 + x_2w_2$

x_1	x_2	Z_{in}-1 ($w_1 = 1, w_2 = -1$)	z_1
1	1	0	0
1	0	1	1
0	1	-1	0
0	0	0	0

z_2 = (x_2 AND NOT x_1) $Z_{in-2} = x_1w_1 + x_2w_2$

x_1	x_2	Z_{in-2} ($w_1 = -1, w_2 = 1$)	z_2
1	1	0	0
1	1	-1	0
0	1	1	1
0	0	0	0

The activation for the output unit y is 1.

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

Presenting the input patterns (z_1 and z_2) and calculating net input and activation gives output of XOR.

Here, $y_{in} = z_1w_1 + z_2w_2$

z_1	z_2	y_{in} ($w_1 = 1, w_2 = 1$)	$y = z_1$ or z_2
0	0	0	0
1	0	1	1
0	1	1	1
0	0	0	0

Thus, the Exclusive -OR function is realized.

1.12 LEARNING RULES

Learning is the process by which the free parameters of a neural network get 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 take place. The set of well defined rules for the solution of a learning differs from the other in the way in which the adjustment to a synaptic weight of a neuron is formulated. Also, the manner in which a neural network is made up of a set of interconnected neurons relating to its environment, is also to be considered. There are various learning rules, some of them are dealth in the following section.

1. **Hebbian Learning Rule** – Hebb's learning rule is the oldest and most famous of all learning rules. It states that, "when an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or network changes take place in one or both cells such that A's efficiently as one of the cells firing B, is increased". This learning can also be called correlational Learning.

   This statement may be split into a two part rule:

   (i) If two neurons on either side of a synapse are activated simultaneously, then the Strength of that synapse is selectively increased.

   (ii) If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.

   This type of synapse is called Hebbian Synapse. The four key mechanisms that characterize a Hebbian Synapse are time dependent mechanism, local mechanism interactive mechanism and correlational mechanism. The simplest form of Hebbian learning is described by $\Delta w = xy$

   This Hebbian learning rule represent a purely feed forward, Unsupervised learning. It states that if the cross product of output the input is positive, this results in increase of weight, otherwise the weight decrease.

   In some cases, the Hebbian rule needs to be modified to counter act Unconstrained growth of weight values, which take place when excitations and response consitently agree in sign. This corresponds to the Hebbian learning rule with saturation of weights at a certain present level.

2. **Perceptron learning Rule** – For the perceptron learning rule, the learning signal is the difference between the desired and actual neuron's response. This type of learning is supervised.

   The fact that the weight vector is perpendicular to the plane separating the input patterns during the learning processes, can be used to interpret the degree of difficulty of training a perceptron for different types of inputs.

   The perceptron learning rule states that for a finite 'n' number of input training vectors,

   x (n) where n = 1 to N

   each with an associated target value,

   t (n) where n = 1 to N

which is +1 or -1 and an activation function $y=f(y_{in})$ where

$$y=\begin{cases}1 \text{ if } y > -\upsilon \\ 0 \text{ if } \vartheta \le y_{in} < -\upsilon \\ -1 \text{ if } y_{in} < -\upsilon\end{cases}$$

the weight updation is given by
if $y \ne t$, then

$$W_{new} = W_{old} + tx$$

if $y = t$, then there is no change in weights.

The perceptron learning rule is of central importance for supervised learning of neural networks. The weights can be initialized at any values in this method.

There is a perceptron learning rule convergence theorem which states, "If there is a weight vector w" such that $f(x(p)w^*) = t(p)$ for all P, then for any starting vector w_1 the perceptron learning rule will converge to a weight vector gives the correct response for all training patterns, and this will be done in a finite no . of steps."

3. Delta Learning Rule (windrow - Hoff rule or Least mean square rule (LMS) – The delta learning rule is also referred to as windrow Hoff rule, named due to the originators. The delta Learning rule is valid only for continous activation functions and in the supervised training mode. The learning signal for this rule is called delta. The delta rule may be stated as,

"The adjustment made to a synaptic weight of a neuron is proportional to the product of the error signal and the input signal of the synapse."

The delta rule assumes that the error signal is directly measurable. The aim of the delta rule is to minimize the error over all training patterns. Delta rule can be applied for signal output unit and several output units. The derivations of these are given below.

Delta Rule for simple output unit– The delta rule changes the weight of the connections to minimize the difference between the net input to the output Unit, y_{in} and the target value t.

The delta rule is given by

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

where, X is the vector of activation of input units.
y_{in} is the net input to output unit, $-\Sigma x_i w_i$
t is the target vector, α - learning rate
The derivation is as follows.
The mean square error for a particular training pattern is

$$E = \sum_j (t_j - y_{-inj})^2$$

The gradient of E is a vector consisting of the partial derivatives of E with respect to each of the weights. The error can be reduced rapidly by adjusting weight w_{ij}
Taking partial differentiation of E w.r.t w_{ij}

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \sum_i (t_j - y_{inj})^2$$

$$= \frac{\partial}{\partial W_{ij}} (t_j - y_{inj})^2$$

Since the weight w_{ij} influences the error only at o/p unit y_j

Also, $y_{inJ} = \sum_{i=1}^{n} (t_j - y_{inj})^2$

we get, $\frac{\partial E}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} (t_j - y_{inj})^2$

$$= 2 (t_j - y_{inj})(-1) \frac{\partial y_{inj}}{\partial W_{ij}}$$

$$\frac{\partial E}{\partial W_{ij}} = -2 (t_{inj} - J_{inj}) \frac{\partial y_{inj}}{\partial W_{ij}}$$

$$\frac{\partial E}{\partial W_{ij}} = -2 (t_j - J_{inj}) x_i$$

Thus the error will be reduced depending upon the given learning by adjusting the weights according to the delta rule given by,

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

Delta Rule for several output Units– The derivation of delta rule for several output units is similar to that in previous section. The weights are charges to reduce the difference between net input and target.

The weight correction involving delta rule for adjusting the weight from the Ith input unit to the Jth output unit is,

$$\Delta w_{IJ} = \alpha (t_j - y_{inj}) x_i$$

Extended delta rule – This can also be called as generalized delta rule. The update for the weight from the Ith input unit to the Jth output unit is,

$$\Delta w_{IJ} = \alpha (t_j - y_{inj}) x_i x_i f'(y_{in}-j)$$

The derivation is as follows:
The squared error for a particular training pattern is,

$$E = \sum_j (t_j - y_j)^2$$

where E is a function of all the weights.
The gradient of E is a vector consisting of the partial derivative of E w.r.t. each of the weights. The error can be educed rapidly by adjusting the weight w_{ij} in the direction of $\frac{-\partial E}{\partial W_{ij}}$

Differentiating E partially w.r.t. w_{ij},

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \sum_j (t_j - y_j)^2$$

$$\frac{\partial E}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}} \sum_j (t_j - y_j)^2$$

since the weights w_{ij} only influences the error at output unit Y_j

since $\quad y_{inJ} = \sum_{i=1} x_i w_{ij}$

$$y_j = f(y_{in}-J)$$

$$\frac{\partial E}{\partial W_{ij}} = 2(t_j - y_j)(-1)\frac{\partial y_j}{\partial W_{ij}}$$

$$= 2(t_j - y_{nj})\frac{\partial f(y_{inj})}{\partial w_{ij}}$$

$$\frac{\partial E}{\partial W_{ij}} = 2(t_j - y_j)x f'(y_{inj})$$

Hence, the error is reduced rapidly for a given learning rate α by adjusting the weight according to the delta rule.

$\Delta w_{ij} = \alpha (t_j - y_j)\, x_i f'(y_{inj})$ gives the extended delta rule.

4. **Competitive Learning Rule–** In this learning, the output neurons of a neural network compete among themselves to become active. The basic idea behind this rule is that there are a set of neurons that are similar in all aspects except for some randomly distributed synaptic weights, and therefore respond differently to a given set of input patterns. However, a limit is imposed on the strength of the neurons. This rule has a mechanism that permits the neurons to compete for the right to respond to a given subset of inputs, such that only one output neurons, or only one neuron per group, is active at a time. The winner neuron during competition is called winner-takes-all neuron.

For a neuron P to be winning neuron, its induced local field Vp, for a given particular input pattern must be largest among all the neurons in the network. The output signal of winning neuron is set to one and the signals that lose the competitions are set to zero. Hence,

$$N = \begin{cases} 1 \text{ if } Vp > Vq \text{ for all } q\ p \neq q \\ 0 \text{ otherwise} \end{cases}$$

This rule is suited for unsupervised network training. The winner takes all or the competitive learning is used for learning satistical properties of inputs. This uses the standard kohanen learning rule.

Let W_{ij} denote the weight of input node j to neuron i. Suppose the neuron has a fixed weight, which are distributed among its input modes;

$\sum_j W_{ij} = 1$ for all i

A neuron than learns by shifting weights from its inactive to active input modes. If a neuron does not respond to a particular input pattern, no learning takes place in that neuron. If a particular neuron wins the competition, its corresponding weights are adjusted.

Using standard competitive rule, the charge Δw_{ij} is given as,

$$\Delta W_{ij} = \begin{cases} \alpha(x_j - w_{ij}) \text{ if neuron i wins the competition} \\ 0 \text{ if neuron i losses the competition} \end{cases}$$

Where α is the learning rate. This rule has the effect of moving the weight vector W_i of winning neuron i toward the input pattern x. Through competitive learning, the neural network can perform clustering.

In this network, the winning neighborhood is sometimes entered beyond the simple neuron winner so that it includes the neighboring neurons. The winner takes all neuron is selected either by the dot product or Euclidean norm. Enclidean norm is most widely used because dot product may require nomalization.

5. **Out star Learning Rule–** Out star learning rule can be well explained when the neurons are arranged in a layer. This rule is designed to produce the designed response t from the layer of n neurons. This type of learning is also called as grossberg learning.

Out star learning occurs for all units in a particular layer and no competition among these units are assumed. However the forms learning updates for kohonen learning and grossberg learning are closely releated.

In the case of out star learning.

$$\Delta w_{jk} = \begin{cases} \alpha(y_k - w_{jk}) \text{ if neuron j wins the competition} \\ 0 \text{ if neuron j losses the competition} \end{cases}$$

The rule is used to provide learning of repetitive and characteristic properties of input-output relationships. Though it is concerned with supervised learning, it allows the network to extract statistical properties of the input and output signals. It ensures that the output pattern becomes similar to the undistorted desired output after repetitively applying on distorted output versions. The weight change here will be a times the error calculated.

6. **Boltzmann Learning–** The learning is a stochastic learning. A neural net designed based on this learning is called Boltzmann learning. In this learning, the neurons constitute a recurrent structure and they work in binary form. This learning is characterized by an energy fnction, E, the value of which is determined by the particular states occupied by the individual neurons of the machine, given by,

$$E = \frac{-1}{2}\sum_i\sum_j w_{ij} x_j x_i,\ i \neq j$$

Where x_i is the state of neurons i and W_{ij} is the weight from neuron i to neuron j. The value i≠j means that none of the neurons in the machine has self feedback. The operation of machine is performed by choosing a neuron at random.

The neuron of this learning process are divided into two groups, visible and hidden. In visible neurons there is an interface between the network and the environment in which it operates but it hidden neurons, they operates independent of the environment. The visible neurons might be clamped on to specific states determined by the environment called as clamped condition.

7. **Memory Based learning–** In memory based learning, all the previous experiences are stored in a large memory of correctly classified input output examples : $[x_i t_i]^N i = 1$ where x_i is the input vector and t_j is the desired response. The desired response is a scalar.

The memory based algorithm involves two parts. They are :

(a) Criterion used for defining the local neighborhood of the test vector, and,

(b) Learning rule applied to the training in the local neighborhood. There are various algorithms in which these parts with neighborhoods are defined.

One of the most widely used memory based learning is the nearest neighbor rule, where the local neighborhood is defined as the training example that lies in immediate neighborhood of the test vector x. The vector,

$x'n \in (x_1, \ldots x_n)$

is said to be nearest neighbor of x_i if, min $d(x_i x_i) = d[x'n, x_i]$ where $d[x_i, x_i]$ is the Euclidean distance between the vector x_i and x_i.

A variant of nearest neighbor classifier is the k-nearest neighbor classifier, which is stated as,

(a) Identify the k-classified patterns that is nearest to test vector x_i for some integer k.

(b) Assign x_i to the class that is most frequently represented in the k-nearest neighbors to x_i, hence K- nearest neighbors classifier acts like on averaging device. The memory based learning clasifier can also be applied to radial basis function network.

Q.12 What is perceptron learning rule. Discuss the terminology underline perceptron weight updation equation.

Ans. See 1.12, Subheading (2)

Q.13 Differentiate between Delta learning rule and perceptron learning rule for a feedback network.

(KUK, May 2011)

Ans. See Section 1.12, Sub heading (2) & (3).

1.13 REPRESENTATION OF PERCEPTRON AND TYPES OF PERCEPTRON

Frank Rosenblatt and Minsky and papert developed large class of artifical neural network called Perceptron. The perceptron learning rule uses an iterative weight adjestment that is more powerful than the Hebb rule. The perceptron use threshold output function and the Mc Culloch pitts model of a neuron. Their iterative learning converges to correct weights, i.e. the weights that produce the exact output value for the training input pattern. The original perceptron is found to have three layers, sensory, associator and response units as shown below.

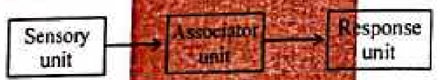


Fig. 1.22 Original Perceptrons

The sensory and association units have binary activations and an activation of +1, 0 or -1 is used for the response unit. All the Units have their corresponding weighted interconnecting. Training in perceptron will continue untill no error occurs. This net solves the problem and is also used to learn the classification.

TYPES OF PERCEPTRON

Q.14 Compare the similarities and differences between single layers and multilayer perceptron and also discuss in what aspects multilayer perceptron are advantageous over single layer function.

(KUK, June 2008)

Ans. There are two types of perceptron : single layer and multilayer perceptron.

1. **Single layer Perceptron–** A single layer perceptron is the single form of a neural network used for the classification of patterns. That are linearly separable. Fundamentally, it consists of a single neuron with adjustable weights and bias. Rosenblatt found that if the pattern used to train the perceptron one drawn from two linerly separable classes, the perceptron algorithm converges and position the decision surface in the form of a hyperplane between the two classes. The perceptron built around a single neuron is limited to performing pattern classification with only two classes. Also classes have to be linearly separable for the perception to work properly.

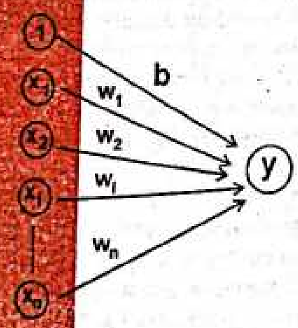


Fig. 1.23 Architecture of Single layer Perceptron

The basic concept of a single layer perceptron as used in pattern classification is that, it is concerned with only a single neuron. The linearity and the integrity learning makes the perceptron network very simple. Training in the perceptron continues till number of error occurs.

Architecture– The architecture of the single perceptron is shown in fig. 1.23

The perceptron has sensory, associated and response units. The input to the response unit will be output from the associator unit, which is a binary vector. Since only the weight between the associator and the response unit is adjusted.

In the architecture, only the associator unit and the response uint is shown. The sensor unit is hidden, because only the weights between the associator and the response unit are adjusted. The input layer consists of input neurons from x_1-x_i-x_n. These always exists a common bias of '1'. The input neurons are connected to the output neurons through weighted interconnecting. This is a single layer network because it has only one layer of interconnections between the input & the output neurons. This network perceives the input signal received and performs the classifications.

Algorithm– To start the training process, initially the weights and the bias are set to zero. The initial weights of the network can be formulated from other techniques like fuzzy system, Genetic Algorithm etc. It is also essential to set the learning rate parameter, which ranges between 0 to 1. Then the input is presented the net input is calculated by multiplying the weights with the inputs and adding the result with the bias entity. Once the net input is calculated, by applying the activation function the output of network is also obtained. Thus output is compared with the target, where if any difference occurs, we go in for weight updation based on perceptron learning rule, else the network training is stopped. The algorithm can be used for both binary and bipolar input vectors. It was a bipolar target with fixed threshold and adjustable bias.

The training algorithm is as follows;

Step 1 : Initialize weights and bias (Initially it can be zero).

Set learning rate α (0 to 1).

Steps 2: While stopping condition is false do steps 3-7.

Steps 3: For each training pair s:t do steps 4-6.

Steps 4: Set activations of input units.

$$x_i = s_j \text{ for } i = 1 \text{ to } x.$$

Steps 5: Compute the output unit response.

$$y_{in} = b + \sum_i x_i w_i$$

The activation function used is

$$y = f(yin) = \begin{bmatrix} 1, \text{if } y_{in} > \theta \\ 0, \text{if } -\theta \le y_{-in} \le \theta \\ -1, \text{if } y_{-in} < -\theta \end{bmatrix}$$

Step 6: The weights and bias are updated if the target in not equal to the output response.

If $t \neq y$ and the value of x_i is not zero

$$w_i(new) = w_{i\,(old)} + \alpha t x_i$$
$$b\,(new) = b_{(old)} + \alpha t$$

else

$$w_{i(new)} = w_{i(old)}$$
$$b_{(new)} = b_{(old)}$$

Step 7: Test for stopping condition

The stopping conditions may be the weight changes

(1) only weights connecting active input units (xi#0) are updates.

(2) Weights are updated only for patterns that donot produce the correct value of y.

Application Procedure– This procedure enables the user to test the network performance. The network should be trained with sufficient number of training data and using the testing data is its performance can be tested. The application procedure used testing perception network is as follow.

Step 1: The weights to be used here are taken from the training algorithm.

Step 2: For each input vector x to be classified do steps 3-4.

Step 3: Input units activations are set.

step 4: Calculate the response of output unit,

$$y_{-in} = \sum_i x_i w_i$$

$$y = f(y_{-in}) = \begin{vmatrix} 1, & \text{if} \quad y_{in} > \theta \\ 0, & \text{if} -\theta \le y_{-in} \le \theta \\ -1, & \text{if} \quad yin < -\theta \end{vmatrix}$$

Perceptron Algorithm for Several Output Classes– The perceptron network for single output class is extended for several output classes. Here there exist more number of output neurons, but the weight updation in this case also is based on the perceptron learning rule. The algorithm is as follows:

Step 1: Initialize the weights and biases. Set the learning rate.

Step 2: When stopping condition is false, perform step 3–7

Step 3: For each input training pair, do step 4-6.

Step 4: Set activation for the input units.

$$x_i = s_i \text{ for } = 1 \text{ to } n$$

Step 5: Compute the activation output of each output unit y_{-inj}.

$$= b_j + e_i \; x_i w_i \text{ for } j = 1 \text{ to } m.$$

$$y_j = f(y_{-inj}) = \begin{bmatrix} 1, & \text{if} \quad y_{-inj} > \theta \\ 0, & \text{if} -\theta \le y_{-inj} \le \theta \\ -1, & \text{if} \quad y_{-inj} < -\theta \end{bmatrix}$$

Step 6: The weights and bias are to be updated for j=1 to m and i= 1 to n.

If $\quad y_j \ne t_j$ and $x_i \ne 0$, then

$$w_{ij(new)} = w_{ij(old)} + \alpha t_j x_i$$
$$b_{j(new)} = b_{j(old)} + \alpha t_j$$

else if $\quad y_j = t_j$

$$w_{ij(new)} = w_{ij(old)}$$
$$b_{j(new)} = b_{j(old)}$$

That is, the biases and weights remain unchanged.

Step 7: Test for stopping condition.

The stopping condition may be the weight changes.

2. **Multilayer Perceptron Networks –** Multilayer perceptron networks is an important class of neural networks. The network consists of a set of sensory units that constitute the input layer and one or more hidden layer of computation modes. The input singal passed through the network in the forward direction. The network of this type is called multilayer perceptron [MLP].

The multilayer perceptron are used with supervised learning and have led to the successful back propagation algorithm. The disadvantage of the single layer perceptron is that it cannot be extended to multi layered version. In MLP networks there exist a non-linear activation function. The widely used non-linear activation function is logistic sigmoid function. The MLP network active for highly complex tasks. The layers of the network are connected by synaptic weights. The MLP thus has a high computational efficiency.

A disadvantage of MLP may also be the presence of non-linearity and complex connections of the network which leads to highly complex theoretical analysis. Also the existence of hidden neurons makes the learning process tedious.

The MLP networks are usually fully connected networks. There are various multilayer preceptron networks which includes Back propagation networks, Radial basis Function network etc.

1.14 LINEAR SEPARABILITY –

Q.15 Define Linear seperability. How is boundary region determine using Linear seperability equation.

Q.16 What do you understand by Linear Separability? Why it is more difficult to learn the weights of a perceptron with a hidden layer than one with out? Explain with examples. (KUK, June 2007)

Q. 17 Briefly discuss about linear separability and the solution for EX-OR problem. Also suggest a network that can solve EX-OR problem. (KUK June 2008)

Q. 18 Explain concept of Linear Separability - IS Ex-OR linear separable. (KUK, May 2009)

Ans. **Linear Separability** – In general, for any output unit, the desired response is '1' if its corresponding input is a member of class or '0' if it is not the purpose of training is to make the input pattern to get similar with the training pattern by adjusting the weights.

The activation function is taken as step function. This function retains a high if net input is positive and a low 1 If the net is input is negative. The net input to the output network is,

$$y_{-in} = b + \sum_i x_i w_i$$

The relation,

$$b + \sum_i x_i w_i = 0$$

gives the boundry region of the net input. The boundary between the region where $y_{in}>0$ and $y_{in}<0$ is called 'decision boundary'. The equation denoting this decision boundary can represent a line, plane or hyper plane.

On training, if the weights of training input vectors of correct response +1 lie on one side of the boundary and of the training input vector of response -1 lie on the otherside of the boundary, then the problem is linear seperable else it is linearly non-separable.

Say with two input vector, the equation of the line separating the positive region and negative region is given by

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

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

These two regions are called the decision regions of the net.

Example : Response regions for AND function.

The AND function for bipolar inputs & target is designed as:

Input $\{x_1, x_2\}$	Output $\{t\}$
(1,1)	+1
(1,-1)	-1
(-1,1)	-1
(-1, -1)	-1

An example of weights that would give the decision boundary namely, the separating line is

$$x_2 = -x_1 + 1$$
$$b = -1$$
$$w_1 = 1$$
$$w_2 = 1$$

$b + x_1 w_1 + x_2 w_2$

$x_2 = -x_1 w_1 - b / w_2$

The choice for sign of b is determined by requirement that

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

Where

$$x_1 = 0 \;\&\; x_2 = 0$$

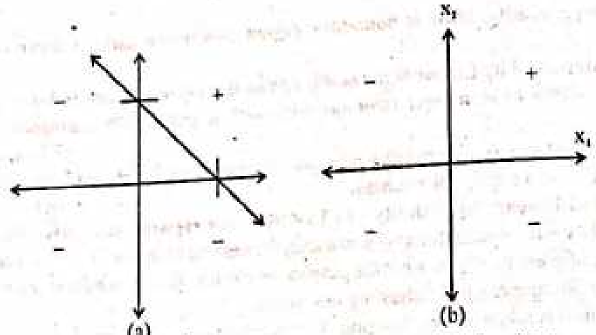


Fig. 1.24 Possible Decision boundary for AND Function

Example: Response region for OR function.

The OR function for bipolar inputs & target is defined as:

Input $\{x_1, x_2\}$	Output $\{t\}$
(1,1)	+1
(1,-1)	+1
(-1, 1)	+1
(-1, -1)	-1

The weights can be chosen to provide a separating line. One example of suitable weight is:

$$b = 1$$
$$w_1 = 1$$
$$w_2 = 1$$

giving the separating line,

$$x_2 = -x_1 -$$

1

The choice of sign for b is determined by requirement the

$$b + x_1 w_1 = x_2$$

$w_2 > 0$

Where

$$x_1 = 0 \;\&$$

$x_2 = 0$

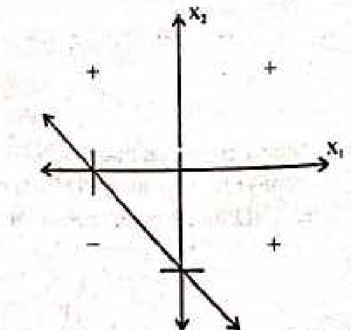


Fig. 1.25 Possible decision boundary for OR function

Ex-OR Problem – A single layer perceptron cannot simulate simple exclusive -OR Operation. This function accepts two inputs that can be zero or one. It produce a output of 1 only if either input is 1. The problem shown below:

- All the combination of x & y are shown on x-y plane

Fig. 1.26 Single neuron system

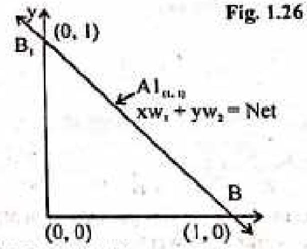


Fig. 1.27 Ex-or Problem as points o x – y plane

Function f is a simple threshold producing a 0 for OUT when NET is below 0.5 & a 1 when it is equal to or above it. The neuron then performs the following calculation.

$$\text{NET} = x w_1 + y\, w_2 \qquad \text{..........(1)}$$

No combination of values for two weights w_1 & w_2 will produce the input output relationship as shown.

x value	y value	desired value
0	0	0
1	0	1
0	1	1
1	1	0

For example : Consider NET to be held constant at the threshold of 0.5 equation (2) describes network in this case

$$x w_1 + y w_2 = 0.5 \qquad \text{.....(2)}$$

This equation is linear in x and y that is, all values for x and y that satisfy this equation will fall or some straight line on x-y plane. Any input values for x and y on this line will produce the threshold value of 0.5 for NET. Input values on one side of line will produce NET less than threshold value making out = 0; values on the other side will produce NET greater than threshold hence OUT= 1. changing the values of W_1, W_2 and the threshold will change the slope & position of the line. For network to produce XOR function, it is necessary to plane the line so that all A's are one side & all B's are another which can not be done. This means that no matter what values are assigned to weights and threshold, this network is unable to represent the XOR function.

1.15 PERCEPTRON LEARNING

Learning is a process by which the free parameters of a neural network get 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 take place. The set of well defined rules for the solution of a learning problem is called as learning algorithm.

For perceptron learning rule, the learning signal is difference between the desired & actual neuron is reponse. The type of learning is supervised & learning signal is equal to,

$$r = d_i - o_i$$

where d_i - o_i are the desired & actual neuron's response.

Infact that the weight vector is perpendicular total plane separating the input patterns during the learning process, can be used to interpret the degree of difficulty of training a perceptron for different types of input. The perceptron learning rule states that for a finite 'n' number of input training vectors,

x (n) where $\quad n = 1$ to n.

each with an associated target value,

t(n) where $\quad n = 1$ to N.

which is +1 or -1 & an activation function

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

$$y = \begin{cases} 1 \text{ if } y > \mu \\ 0 \text{ if } - V \le y\text{ in} \le \mu \\ -1 \text{ if } y\text{in} \le -\mu \end{cases}$$

& weight updation is given by if $y \neq t$, then

$$W_{new} = W_{old} + tx$$

if y = t, there is no change in weights. The perceptron learning rule is of central importance for supervised learning of neural networks. The weights can be initialized at any value in this method.

1.16 PERCEPTRON TRAINING

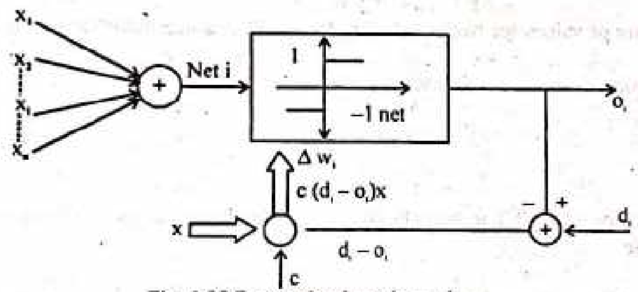


Fig. 1.28 Perception learning rule

To start the training process initially the weights and the bias are set to zero.

1. The initial weights of the networks can be formulated from other techniques like Fuzzy system, Genetic algorithm etc. It is also essential to set the learning rate parameter which ranges between 0 & 1.
2. Then the input is presented, the net input is calculated by multiplying the weights with inputs & adding the result with bias entity.
3. Once the net input is calculated, by applying the activation function the output of the network is also obtained. This output is compared with the target, where if any difference occurs, we go in for weight updation based on perceptron learning rule, else the network training is stopped.
4. This algorithm can be used for both binary & bipolar input vectors. It uses a bipolar target with fixed threshold & adjustable bias:

The training algorithm is as follows:

Step 1: Initialize weights and bias (initially it can be 0) set learning rate α $(0 \le \alpha \le 1)$

Step 2: While stopping conditions is false do steps 3 to 7.

Step 3: For each training pair s.t. do steps 4-6.

Step 4: Set activations of input units

$$x_i = S_i \text{ for } i = 1 \text{ to n.}$$

Step 5: Compute the output unit response

$$y_{in} = b + \sum_i x_i w_i$$

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

Step 6: The weights & bias are updated if the target is not equal to the output response.

If $t \neq y$ & the value of x_i is not zero.

$$w_{i(new)} = W_{i(old)} + \alpha t x_i$$

$$b_{(new)} = b_{(old)} + \alpha t$$

else

$$w_{i(new)} = w_{i(old)}$$

$$b_{(new)} = b_{(old)}$$

Step 7: Test for stopping condition.

The stopping condition may be the weight changes.

Character Recognition :-

Application Procedure : This procedure enables the user to test network performance. The network should be trained with sufficient number of training data & using the testing data, its performance can be tested. The application procedure is as follows:

Step 1: Apply training algorithm, to set the weights.

Step 2: For each input vector x to be classified do steps 3-4.

Step 3: Input unit activations are set.

Step 4: Calculate response of each output unit,

$$y - in = \sum_i x_i w_i$$

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

Q.19 Explain various types of models used in Artifical Neural networks. (KUK, May 2010)

Ans. The various types of models used in Artificial Neural networks are.

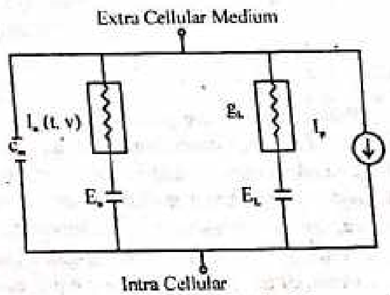


Fig. 1.29 Components of Hodgkin Neuron model

1. **Hodgkin - Huxley Neuron Model** – Hodgkin in Huxley type models represent the biophysical characteristic of cell membranes. The lipid bilayer is represented as a capacitance [Cm]. Voltage gated and leak ion channels are represents by non linear [gn] and linear [gi] conductances, respectively. The electrochemical gradients during the flow of ions are represented by batteries [E], and ion pumps and exchangers represented by current sources [Ip].

The Hodgkin-Huxley model is a scientific model that describes how action potentials in neurons are initiated and propagated. It is a set of non-linear ordinary differential equations that approximated the electrical characteristics of excitable cell such as neurons and cardiac myocytes.

Alan loyd Hodgkin and Andrew Huxley describe the model in 1952 to explain the Ionic mechanisms underlying the initiation and propagation of action potentials in the squid giant axon. They received the 1963 novel Prize in physiology or Medicine of this work.

Basic Components—The components of a typical Hodgkin-Huxely model are shown in the figure. Each component of an excitable cell has a biophysical analog. The lipid bilayer is represented as a capacitance [cm]. Voltage gated cation channel are represented by non-linear electrical conductances [gri where n is specific ion channel], meaning that the conductance is voltage and time-depends. This was later shown to be mediated by voltage gated Cation Channel proteins, each of which has an open probability that is voltage dependent. Leak channel are represented by linear conductances [gl]. The electrochemical gradients driving the flow of ions are represented by batteries [En and El], the values of which are determined from the Nernst potential of the ionic species of interest. Finally, ion pumps are presented by current sources [Ip].

The time derivative of the potential across the membrane [Vm] is propartional to the sum of the currents in the circuit. This is represented as follows :

$$V_m = -\frac{1}{cm}\left[\Sigma I_i\right]$$

Where I_i denotes the individual ionic currents of the model.

Ionic Current Characterization– The current flowing through the ion channels is mathematically represented by the following equation:

$$I_i\,[Vmt] \quad = [V_m\text{-}E]\;gi$$

Where F_i is the reversal potential of the ith ion channel. In Voltage-gated ion channels, the channel conductance gi is a consant [gl in the figure]. The current generated by ion pumps is dependent on the ionic specific to that pump. The following section will describe these formulations in more detail.

Voltage-gated Ion Channels– Under the Hodgkin Huxley formulation conductances for voltage-gated channels, {gn(t,v)} are expressed as:

$$I_n\,[Vm't] \;= g_n \Psi \alpha x^{\beta}$$

$$\vartheta[Vm't] = \frac{1}{t_Q}[\vartheta_{\infty} - \vartheta]$$

$$x[Vm't] = \frac{1}{tx}[x_{\infty} - x]$$

Where φ and x are gating variables for activation and inactivation, respectively, representing the fraction of the maximum conductance available at any given time and voltage. gn is the maximal value of the conductance. α and β are constants are τφ and T_x are the time constants for activation and inactivation, respectively φ∞ and x∞ are the steady state values for activation and inactivation, respectively and are, usually represented by Boltzmann equations as functions of Um.

In order to characterize voltage-gated channels, the equations will be fit to voltage-clamp data. For a derivation of the Hodgkin-huxley equations under Voltage-Clamp see. Briefly, when the membrance potential is held at a constant value [i.e. Voltage-clamp], for each value of the membrane potential the non-linear gating equations reduce to linear differential equations of the form:

$$\varphi(t) = \varphi 0 - [(\varphi 0 - \varphi\infty)(1 - e^{-t/\tau\varphi})]$$

$$x(t) = x0 - [(x_0 - x_{\infty})(1 - e^{-t/\tau x})]$$

Thus, for every value of membrane potential, Vm the following equation can be fit to the current curve.

$$In(t) = \bar{g}n\varphi^{\alpha}.x^{\beta}(V_m - E_n)$$

The levenberg-Marquardt algorithm, a modified Gauss-Neuton algorithm, is often used fit the these equations to voltage- clamp data.

Leak Channels– Leak channels account for the natural permeability of membrane to Ions and take the form of the equation for voltage-gated channels, where the conductance g_i is a constant.

Pumps and Exchanger– The membrane potential depends upon the maintenance of ionic concentration gradients across it. The maintenance of these cancentration gradients required active transport of ionic species. The sodium-potassium and sodim-calcium exchangers are the best know of these. Some of the basic properties of the Na/Ca exchange have already been well established: the stoichiometry of exchange is $3Na^+ : 1\ Ca^{2+}$ and the exchanger is electrogenic and voltage sensitive. The Nalk exchanger has also been described in detail.

Improvements and Alternative models– The Hodgkin-Huxley model is widely regarded as one of the great achievements of gradients across it. The maintenance of these concentration gradients active transport of ionic species. The sodium-potassium and sodium-calcium exchangers are the best known of these. Some of the basic properties of the Na/Ca exchanger have already been well established the stoichiometry of exchaner is $3Na^+ 1Ca^{2+}$ and the exchanger is electrogenic and voltage sensitive. The Na/k exchanger has also been described in detail.

Improvements and Alternative Models – The Hodgkin-Huxley model is wodely regard as one of the great achievement of 20th century biophysics. Nevertheless, modern Hodgkin-Huxley type models have been extended in several important ways–

– Additional ion channel population have been incorporated based on experimental data.

— Models often incorporate highly complex geometries of dendrites and axons, often based on microscopy data.

Several simplified neuronal models have been developed, facilitating efficient large scale simulation of groups of neurons, as well as mathematical insight into dynamics of action poential generation.

2. Integrate And Fire neuron model – In this section, we given an overview of integrate and fire models. The leaky integrate-and-fire neuron introduced is probably the best known example of a formal spiking neuron model. Generalizations of the leaky integrate and fire model include the non-linear integrate and fire model that is discussed in section. All integrate and fire neurons can either be stimulated by external current or by synaptic input from presyanptic neurons.

The basic circuit is the module inside the dashed circle on the right hand side. A current I(t) charges the RC circuit. The voltage u(t) across the capacitance (points) is compared to a threshold U. If u(t) =μ at time $t_i^{(f)}$ an output pulse $\delta(t\text{-}t_i^{(f)})$ is generated. left part: A presynaptic spike $\delta(t\text{-}t_i^{(f)})$ is low pass filtered at the synapase and generates an input currentpluse $\alpha(t\text{-}t_i^{(f)})$

The basic circuit of an integrate-and-fire model consists of a capacitor (in parallel with a resistor R driven by a current I(t), The driving current can be split into two components, $I(t) = I_R + I_c$. The first component is the resistive current I_R which passes through the linear resistor R. It can be calculated from Ohm's law as I_R =u/R where u is the voltage across the resistor. The second component I_c charges the capaciter C. From the definition of the capacity as C= q/u [Where q is the charge and u the Voltage] we find a capacitive current I_c = C du/dt. Thus

$$I(t) = \frac{ut}{R} + C\frac{du}{dt}$$

We multiply by R and introduce the time contstant τ_m=RC of the leaky integrator. This yields the standard form

$\tau_m \frac{du}{dt} = -u(t) + RI(t)$

We refer to u as the membrance potential and to τ_m as membrane time constant of the neuron.

In integrate and fire models the form of an action potential is not described explicitly spikes are formal events characterized by firing time $t^{(f)}$.

The firing time $t^{(f)}$ is defined by threshold direction

$t(f) : u\,(t^{(f)}) = \upsilon$

Immediately after $t_i^{(f)}$ the potential is reset to a new value $u_r < \upsilon$

$L_{im}\ u(t) = u_r$

$t \to t^{(f)},\ t > t^{(f)}$

Leahy Integrate and fire model

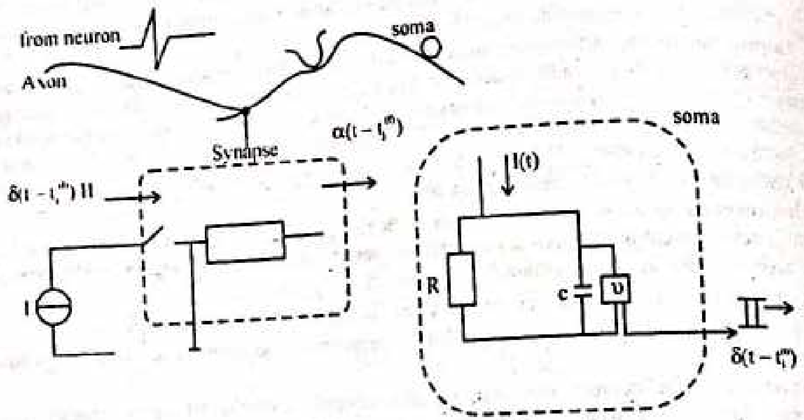


Fig. 1.30 Schematic diagram of the integrate and fire model

For $t > t^{(f)}$ the dynamics is again by untill the next threshold crossing occurs. The combination of leaky integration and reset defines the basic integrate-and-fire model (Setin, 1967b). We note that, since the membrane potential is never above threshold, the threshold condition reduces to the criterion, i.e. the condition on the solpe du/dt can be dropped.

In its general version, the leaky integrate and fire neuron may also incorporate an abosolute refractory period, in which case we proceed as follows. If u reaches the threshold at time $t = t^{(f)}$, we interrput the dynamics during an absolute refractory time $\Delta^{abs}$and with the new intitial condition u_r.

3. **Spiking Neural Model –**

The spiking Neuron Model: Temporal Noisy-Leaky Integrator– The TNLI neuron model [christodoulauet al. 1992] is a simple, biologically inspired and hardware realisable computational model. Figure 1.31 shows an analogons hardware outline of the TNLI using a PRAM at each input and a Hodgkin and Huxely equivalent circuit for a leaky cell membrane.

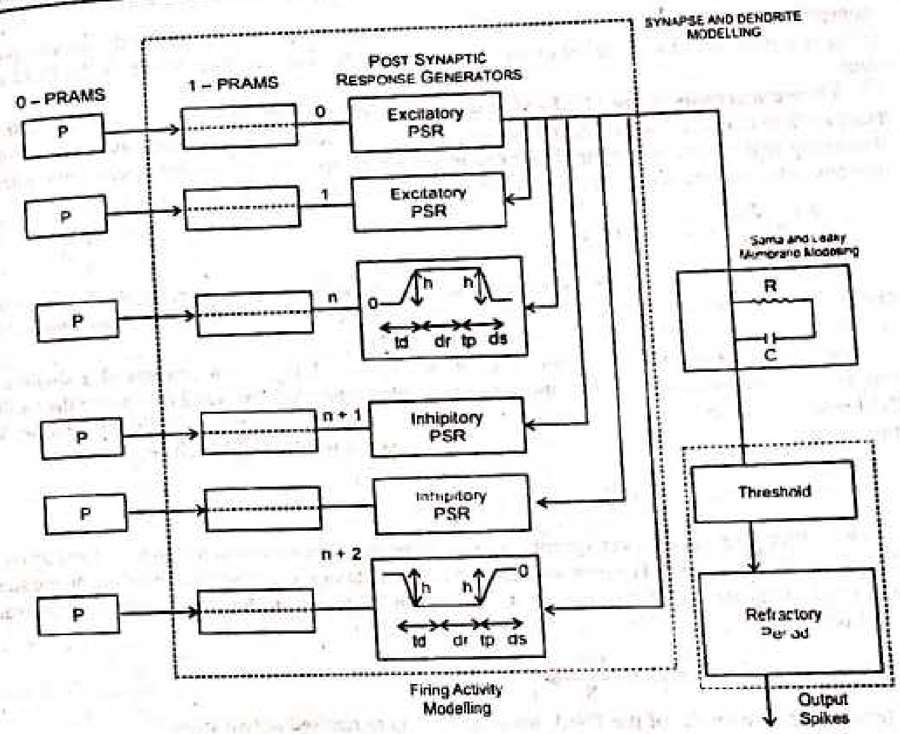


Fig. 1.31 Analogue Hardware Outline of TNLI Neuron Model

The dotted line boxes indicate the corresponding parts of the real neuron which the TNLI modules are inspired by. At inputs n and n+m, the post synaptic current Response spon taneously utilized are shown.

Synaptic and Dentritic modelling– The 1-PRAM in the TNLI model the stochastic and spontaneously neurotransmitter release by the synapses of real neurons, i.e. stochastic synapses can either generate EPOSPs spontaneously or cause presynaptic spikes to fail to produce EPSPs. The O-PRAMs shown in the model are used in the simulations to produce random spike input trains from other neurons of controlled mean input frequency, according to their probability P.

The presynaptic transmitter release creates an ion-specific conductance change in the postsynaptic neuron which in the TNLI we approximate with an inward or outward current flow model. For every spike generated by the PRAMs, the PSR generators produce postsynaptic current responses PSE_{ij} (t), of controlled shapes.

Modeling of the soma, the somatic membrane and the firing time

The EPSCs ad IPSCS are then summed temporally and the total postsynaptic current response is fed into the RC circuit. The synaptic Saturation that occurs in the real neuron during the temporal summation of the postsyantic potentials is not currently modelled in the TNLI, but it could be easily incorporated by applying the method used in. The capacitance C and the resistance R respresent the somatic leaky membrane of real neurons and therefore this circuit model the decay that occurs in the somatic potential of the real neuron. If the potential of the capacitor exceeds a constant threshold, then the TNLI neuron fires.

Theoretical basis of the TNLI– Most of the leaky integrator models are based on the equation that Hodgkin and Huxley used to describe the generation of an action potential in the giant squid axon. By linearising that equation, a simplified version for a network of single-compartment leaky integrator neurons with synaptic noise, can be described by the shunting differential equation:

$$C_i \frac{dvi}{dt} = \frac{Vi(t)}{R_i} + \sum_{j+i} Dgij(t)\left[S_{ij} - V_i(t)\right] \qquad (1.1)$$

Where the term on the LHS is the variation of accmulated change in neuron i, the first term on the RHS the membrane leakge current in neuron i and threshold term on the RHS is the synaptic input current which is excitatory for Sij>0 and inhibitoy for Sij<0

The TNLI corresponds to the model described by Eq. (1.1) since it consists of a single active compartment representing the Soma and performing intergration of EPSCs and IPSCs. For the hardware TNLI model however, eq (1.1) is further simplified synapse and S_{ij} << Vi(t) for inhibitory synapse. Thus, after the above approximation, the leaky integrator equation for the TNLI becomes:

$$C_i \frac{dui}{dt} = \frac{-V_i(t)}{R_i} + \sum_j \sum_{0 \le k \le T} PSR_{ij}(t - t_k) \qquad (1.2)$$

Where PSR_{ij} $(t-t_k)$ is the post synaptic current response caused by an input spike having arrived at time k_i from input neuron j. The response can either be excitatory or inhibitory depending on the sign of PSR_{ij}. T is the total number of time steps that the system is left to operate. So with the above simplification Eq. 1.2 because

$$CV(t+\Delta t) = (Vt) + I(t) \times \Delta t - \frac{V(t)}{R}\Delta t \Big|$$

In the hardware model of the TNLI this equation can to realised in two steps:

First Step:

$$CV^*(t) = CV(t)\text{before} + I(t) \times \Delta t$$

and the second step :

$$CV(t+\Delta t) = \alpha \times (U^*(t), \alpha < 1$$

where Δ is the delay rate with which the initial, counter output contents $C(U^*(t))$ are multiplied before they are routed back to the counter via the load input. The relationship between the decay rate Δ and the time constant Δ=RC canbe deduced from equation and is given by:

$$\alpha = 1 - \frac{1}{RC}\Delta t \qquad (1.3)$$

From EGs (1.2) and (1.3) we deduce that the capacitor is charged according to the equation:

$$V(t+\Delta t) = \alpha \times \left[V(t) + \frac{I(t) \times \Delta t}{C}\right] \qquad (1.4)$$

The firing times in the TNLI neuron i (TnT) one determined by the iterative threshold condition:

$$Tn^i = \inf(t)\left\{V_i(t) \ge V_{th}; t \ge T^i_{n-1} + t_R\right\} n > 1$$

Where Vth, is the threshold voltage of neuron i and n is the time of firing and if means a union satisfying both terms in the brackets.

4. Mc Culloch- Pitts Neuron Model–

See Section 1.11

Example 1.15 : Realise a Hebbnet for the AND function with bipolar inputs and targets.

Solution: The AND function gives a high '1' if both the inputs are high else returns a low '-1'.

The training patterns are,

Input			Target
x_1	x_2	B	y
1	1	1	1
1	-1	1	-1
-1	1	1	-1
-1	-1	1	-1

Forming the table, initialize all the weights and the bias to be zero i.e.

$w_1 = w_2 = 0$ and $b = 0$

the weight change is calculated using,

$\Delta w_i = x_i y$ and $\Delta b = y$

Input			Target	Weight Charges			Weights		
(x_1	x_2	b)	y	Δw_1	Δw_2	Δb	w_1	w_2	B
1	1	1	1	1	1	1	1	1	1
1	-1	1	-1	-1	1	-1	0	2	0
-1	1	1	-1	1	-1	-1	1	1	-1
-1	-1	1	-1	1	1	-1	2	2	-2

This completes one epoch of training. The straight live separating the regions can be obtained after presently each input pair. Thus,

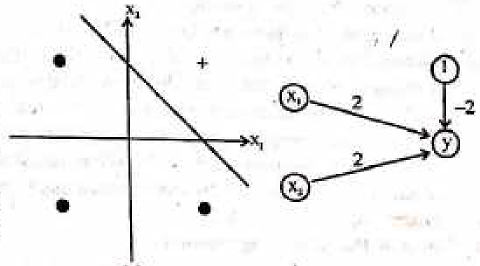
$$x_2 = -x_1 \frac{w_1}{w_2} - \frac{b}{w_2}$$

After Ist input, $x_2 = -x_1 \frac{1}{1} - \frac{1}{1} = -x_1 - 1$

$x_2 = -x_1 - 1$

Similarly after 2nd, 3rd and 4th epochs 1 the sperating lines are $x_2 = 0$, $x_2 = -x_1 + 1$, $x = -x_1 + 1$.

For the 3rd and 4th epoch the seperating line remaing the same, hence this line separates the boundary regions are shown in fig. 1.32



(a)

Fig. 1.32 Hebb net for AND function

The same procedure can be repeated for generating the logic function OR, NOT, AND NOT etc.

Example 1.6: Apply the Hebb net to the training patterns that define XOR function with bipolar input and targets.

Solution :

Input x_1	x_2	b	Target y
1	1	1	-1
1	-1	1	1
-1	1	1	1
-1	-1	1	-1

By Hebb training algorithm. assigning initial values of the weights w_1 & w_2 to be zero. and bias to be zero.

$w_1 = w_2 = 0$ and b= 0

Input (x_1	x_2	b)	Target y	Weight Charges Δw_1	Δw_2	Δb	Weights w_1	w_2	B
1	1	1	-1	-1	-1	-1	-1	-1	-1
1	-1	1	1	+1	-1	1	0	-2	0
-1	-1	1	1	-1	1	1	-1	-1	1
-1	-1	1	-1	1	1	-1	0	0	0

The weight changes are called using.

$\Delta w_1 = xy$ and $\Delta b = y$

The new weights by,

$W_{(n)} = W_{(old)} + ow$ and $b_{(n)} = b_{(o)} + \Delta b$.

The final weights obtained for the XOR function are $w_1 = w_2 = 0$ and b= 0. Hence it is clear that the separating line can not drawn.

Thus, Hebb rule cannot be used to form a training pattern to define XOR function.

Q.20 Explain the following terms in context of Neural Network. **(KUK, May 2011)**

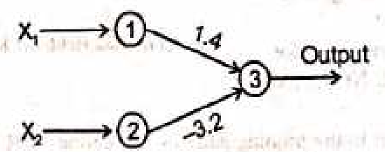
Ans.

(i) Synapse – See Question No. 4

(ii) **Activation function–** See Question No. (8, 9)

(iii) **Generalization–** Once trained, a network's is response can be, to a degree insensitive to minor variations in its input. This ability to see through wise and distortion to the pattern that lies within is vital to pattern recognition in a real-world environment. Overcoming the of the conventional computer, it produces a system that can deal with the imperfact would in which we live. It is important to note that the artificial neural network generalizes automatically as a result of its structure, not by using human intelligence embedded in the form of adhoc computer programs.

Q.21 Consider the following network. **(KUK, May 2010)**



Here, $x_1 = x_2 = 1$, Target T =1, Bias = 1, eta = 0.4

Train the networks for (1, 1) using activation function.

F (x) = 1 is x>0. F(x) = 0 if x<0.

Ans. $x_1 = 1$, $x_2 = 1$, T = 1, B = 1

Q = 0.4, $w_1 = 1.4$, $w_2 = -3.2$

$y_{in} = b + \Sigma_{x_1w_i}$
$= 1 + 1\times1.4 + 1\times(-3.2)$
$= 1 + 1.4 - 3.2 = 2.4 - 3.2 = -0.8$
$y = F(x) = 0$
$W_1 = 1.4 + 0.4 \times 1 \times 1$
$= 1.4 + 0.4 = 1.8$
$b_{(new)} = 1 + 0.4 \times 1 = 1.4$
$w_2 = -3.2 + 0.4 \times 1 \times 1$
$= -3.2 + 0.4 = -2.8$
$y = 1.4 + \Sigma_{x_1w_i}$
$= 1.4 + 1\times1.8 + 1\times(-2.8)$
$= 1.4 + 1.8 - 2.8 = 3.2 - 2.8 = 0.4$
$0.4 > 0$
$\therefore f(x) = 1$

• • •