Guru Ghasidas Vishwavidyalaya, Bilaspur (C.G.)
B.Sc Vth Semester, Examination-2013
Subject- Introduction to Artificial Neural Network

Time: Three Hours] **[Maximum Marks: 30**

Section A

Attempts all the questions. All carry equal marks (10×1)

1. **Answer the following questions:**

i. Synaptic weights provides the……method for the design of neural networks.
Ans. **Traditional method**

ii. Elements of NN is
A. Adopters B Adder C Collector D Bias.
Ans. **B Adder**

iii. The bias has the effect of decreasing the net input of the activation function, depending on whether it is:
A. Only positive B. Only negative C positive or negative. D All false
Ans. **B. Only negative.**

iv. A node signal equals the …….of all signals entering pertinent node via the incoming link
A. Algebraic multiplied B. algebraic sum C. multiplication of sums D. All false
Ans. **B. algebraic sum**

v. Delta rule was invented by…….
Ans**. Widrow and Hoff**

vi. In memory based learning the vector X'n ϵ {x1, x2,x3…Xn} is said to be the nearest neighbor of X_{test} if min d(X_i,X_{test})=……
Ans $d(\mathbf{x}'_N, \mathbf{x}_{\text{test}})$

vii. In hebb's learning, if two neurons on either side of a synapse are activated simultaneously, then the strength of that synapse is…..
A. Remain constant B. Decreased C. Increased. D. Either B or C depends on synapse.
Ans. **C. Increased**

viii. In single layer perceptron, when the neuron produce output +1 and -1, then the hard limiter input is…….
Ans **Positive and negative respectively**

ix. In Competitive learning, the output of the wining neuron and all other neurons are set equal to
A. +1, -1, B. -1,0 C. 1,0 D. 0,+1
Ans. **C. 1,0**

x. The neurons of Boltzmann machine partitions into two groups……..
Ans. **Visible and hidden**

Section B

Attempt any four questions. All carry equal marks (4 ×5)

2. Explain the benefits of ANN.

Ans:

Benefits of Neural Networks

It is apparent that a neural network derives its computing power through, first, its massively parallel distributed structure and, second, its ability to learn and therefore generalize. *Generalization* refers to the neural network producing reasonable outputs for inputs not encountered during training (learning). These two information-processing capabilities make it possible for neural networks to solve complex (large-scale) problems that are currently intractable. In practice, however, neural networks cannot provide the solution by working individually. Rather, they need to be integrated into a consistent system engineering approach. Specifically, a complex problem of interest is *decomposed* into a number of relatively simple tasks, and neural networks are assigned a subset of the tasks that *match* their inherent capabilities. It is important to recognize, however, that we have a long way to go (if ever) before we can build a computer architecture that mimics a human brain.

The use of neural networks offers the following useful properties and capabilities:

1. *Nonlinearity*. An artificial neuron can be linear or nonlinear. A neural network, made up of an interconnection of nonlinear neurons, is itself nonlinear. Moreover, the nonlinearity is of a special kind in the sense that it is *distributed* throughout the network. Nonlinearity is a highly important property, particularly if the underlying physical mechanism responsible for generation of the input signal (e.g., speech signal) is inherently nonlinear.

2. *Input–Output Mapping*. A popular paradigm of learning called *learning with a teacher or supervised learning* involves modification of the synaptic weights of a neural network by applying a set of labeled *training samples* or *task examples*. Each example consists of a unique *input signal* and a corresponding *desired response*. The network is presented with an example picked at random from the set, and the synaptic weights (free parameters) of the network are modified to minimize the difference between the desired response and the actual response of the network produced by the input signal in accordance with an appropriate statistical criterion. The training of the network is repeated for many examples in the set until the network reaches a steady state where there are no further significant changes in the synaptic weights. 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. Such an approach brings to mind the study of *nonparametric statistical inference*, which is a branch of statistics dealing with model-free estimation, or, from a biological viewpoint, *tabula rasa* learning (Geman et. al., 1992); the term "nonparametric" is used here to signify the fact that no prior assumptions are made on a statistical model for the input data. Consider, for example, a *pattern classification* task, where the requirement is to assign an input signal representing a physical object or event to one of several prespecified categories (classes). In a nonparametric approach to this problem, the requirement is to "estimate" arbitrary decision boundaries in the input signal space for the pattern-classification task using a set of examples, and to do so *without* invoking a probabilistic distribution model. A similar point of view is implicit in the supervised learning paradigm, which suggests a close analogy between the input–output mapping performed by a neural network and nonparametric statistical inference.

3. *Adaptivity.* Neural networks have a built-in capability to *adapt* their synaptic weights to changes in the surrounding environment. In particular, a neural network trained to operate in a specific environment can be easily *retrained* to deal with minor changes in the operating environmental conditions. Moreover,when it is operating in a *nonstationary* environment (i.e., one where statistics change with time), 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, and control applications, coupled with the adaptive capability of the network, make it a useful tool in adaptive pattern classification, adaptive signal processing, and adaptive control. As a general rule, it may be said that the more adaptive we make a system, all the time ensuring that the system remains stable, the more robust its performance will likely be when the system is required to operate in a nonstationary environment. It should be emphasized, however, that adaptivity does not always lead to robustness; indeed, it may do the very opposite. For example, an adaptive system with short time constants may change rapidly and therefore tend to respond to spurious disturbances, causing a drastic degradation in system performance. To realize the full benefits of adaptivity, the principal time constants of the system should be long enough for the system to ignore spurious disturbances and yet short enough to respond to meaningful changes in the environment; the problem described here is referred to as the *stability–plasticity dilemma* (Grossberg, 1988b).

4. *Evidential Response.* In the context of pattern classification, a neural network can be designed to provide information not only about which particular pattern to *select*, but also about the *confidence* in the decision made. This latter information may be used to reject ambiguous patterns, should they arise, and thereby improve the classification performance of the network.

5. *Contextual Information.* Knowledge is represented by the very structure and activation state of a neural network. Every neuron in the network is potentially affected by the global activity of all other neurons in the network. Consequently, contextual information is dealt with naturally by a neural network.

6. *Fault Tolerance.* A neural network, implemented in hardware form, has the potential to be inherently *fault tolerant*, or capable of robust computation, in the sense that its performance degrades gracefully under adverse operating conditions. For example, if a neuron or its connecting links are damaged, recall of a stored pattern is impaired in quality. However, due to the distributed nature of information stored in the network, the damage has to be extensive before the overall response of the network is degraded seriously. Thus, in principle, a neural network exhibits a graceful degradation in performance rather than catastrophic failure. There is some empirical evidence for robust computation, but usually it is uncontrolled. In order to be assured that the neural network is in fact fault tolerant, it may be necessary to take corrective measures in designing the algorithm used to train the network (Kerlirzin and Vallet, 1993).

7. *VLSI Implementability.* The massively parallel nature of a neural network makes it potentially fast for the computation of certain tasks. This same feature makes a neural network well suited for implementation using *very-large-scale-integrated* (VLSI) technology. One particular beneficial virtue of VLSI is that it provides a means of capturing truly complex behavior in a highly hierarchical fashion (Mead, 1989).

8. *Uniformity of Analysis and Design.* Basically, neural networks enjoy universality as information processors. We say this in the sense that the same notation is used in all domains involving the application of neural networks. This feature manifests itself in different ways:

- Neurons, in one form or another, represent an ingredient *common* to all neural networks.
- This commonality makes it possible to *share* theories and learning algorithms in different applications of neural networks.
- Modular networks can be built through a *seamless integration of modules.*

9. *Neurobiological Analogy.* The design of a neural network is motivated by analogy with the brain, which is a living proof that fault tolerant parallel processing is not only physically possible but also fast and powerful. Neurobiologists look to (artificial) neural networks as a research tool for the interpretation of neurobiological phenomena. On the other hand, engineers look to neurobiology for new ideas to solve problems more complex than those based on conventional hard-wired design technique

3. Explain the properties of neural network as directed graph.

Ans: The block diagram is given below. The block diagram provides the functional description of the various elements that constitute the model of an artificial neuron. We may simplify the appearance of the model by using the idea of signal-flow graphs without sacrificing any of the functional details of the model. Signal-flow graphs with a well-defined set of rules were originally developed by Mason (1953, 1956) for linear networks. The presence of nonlinearity in the model of a neuron limits the scope of their application to neural networks. Nevertheless, signal-flow graphs do provide a neat method for the portrayal of the flow of signals in a neural network, which we pursue in this section.

A *signal-flow graph* is a network of directed *links* (*branches*) that are interconnected at certain points called *nodes*. A typical node j has an associated *node signal* x_j. A typical directed link originates at node j and terminates on node k; it has an associated

transfer function or *transmittance* that specifies the manner in which the signal y_k at node k depends on the signal x_j at node j. The flow of signals in the various parts of the graph is dictated by three basic rules:

Rule 1. A signal flows along a link only in the direction defined by the arrow on the link.

Two different types of links may be distinguished:

- *Synaptic links*, whose behavior is governed by a *linear* input–output relation. Specifically, the node signal x_j is multiplied by the synaptic weight w_{kj} to produce the node signal y_k, as illustrated in Fig. (a)
- *Activation links*, whose behavior is governed in general by a *nonlinear* input–output relation. This form of relation is illustrated in fig. (b). where $\varphi(\cdot)$ is the nonlinear activation function.

Rule 2. A node signal equals the algebraic sum of all signals entering the pertinent node via the incoming links.

This second rule is illustrated in fig. (c) for the case of synaptic convergence or fan-in

Rule 3. The signal at a node is transmitted to each outgoing link originating from that node, with the transmission being entirely independent of the transfer functions of the outgoing links.

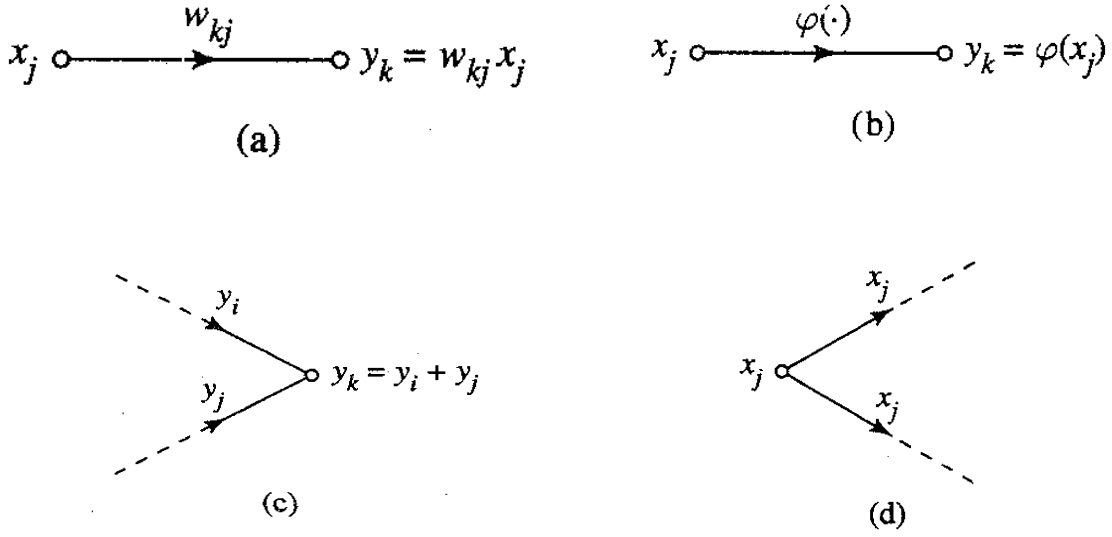

Fig. Illustrating the basic rulles for the constuction of thesignal flow graphs.

This third rul is illustrated n fig. (d) for the case of synaptic divergence or fan-out.

For example, using these rules we may construct the signal-flow graph of Fig 2 as the model of a neuron.

Based on the signal flow graph of fig. 2 as the model of neural network we may now offer the following mathematical definition of neural network:

A neural network is a directed graph consisting of nodes with interconnecting synaptic and activation links, and is characterized by four properties:

1. *Each neuron is represented by a set of linear synaptic links, an externally applied bias, and a possibly nonlinear activation link. The bias is represented by a synaptic link connected to an input fixed at +1.*
2. *The synaptic links of a neuron weight their respective input signals.*
3. *The weighted sum of the input signals defines the induced local field of the neuron in question.*
4. *The activation link squashes the induced local field of the neuron to produce an output.*

The state of the neuron may be defined in terms of its induced local field or its output signal.

A directed graph so defined is *complete* in the sense that it describes not only the signal flow from neuron to neuron, but also the signal flow inside each neuron. When, however, the focus of attention is restricted to signal flow from neuron to neuron, we may use a reduced form of this graph by omitting the details of signal flow inside the individual neurons. Such a directed graph is said to be *partially complete*. It is characterized as follows:

1. *Source nodes* supply input signals to the graph.
2. Each neuron is represented by a single node called a *computation node*.
3. The *communication links* interconnecting the source and computation nodes of the graph carry no weight; they merely provide directions of signal flow in the graph.

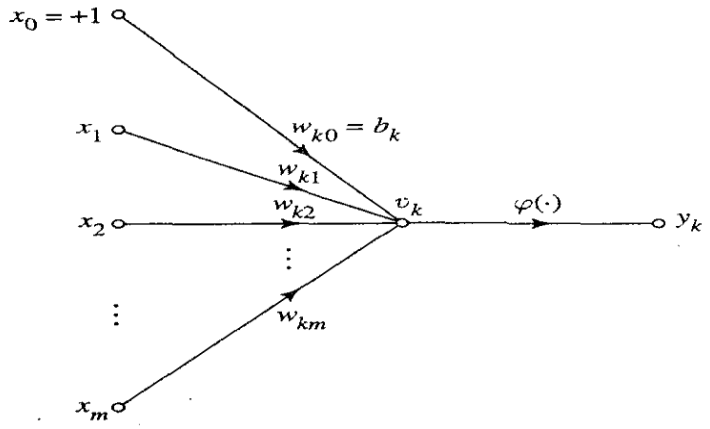

Fig. 2 Signal flow graph of a neuron.

4. Explain Hebbian learning

Ans:

HEBBIAN LEARNING

Hebb's postulate of learning is the oldest and most famous of all learning rules; it is named in honor of the neuropsychologist Hebb (1949). Quoting from Hebb's book, *The Organization of Behavior* (1949, p.62):

> 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 metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased.

Hebb proposed this change as a basis of associative learning (at the cellular level), which would result in an enduring modification in the activity pattern of a spatially distributed "assembly of nerve cells."

This statement is made in a neurobiological context. We may expand and rephrase it as a two-part rule (Stent, 1973; Changeux and Danchin, 1976):

1. *If two neurons on either side of a synapse (connection) are activated simultaneously (i.e., synchronously), then the strength of that synapse is selectively increased.*
2. *If two neurons on either side of a synapse are activated asynchronously, then that synapse is selectively weakened or eliminated.*

Such a synapse is called a *Hebbian synapse.*[3] (The original Hebb rule did not contain part 2.) More precisely, we define a Hebbian synapse as a synapse that uses a *time-dependent, highly local, and strongly interactive mechanism to increase synaptic efficiency as a function of the correlation between the presynaptic and postsynaptic activities.* From this definition we may deduce the following four key mechanisms (properties) that characterize a Hebbian synapse (Brown et al., 1990):

1. *Time-dependent mechanism.* This mechanism refers to the fact that the modifications in a Hebbian synapse depend on the exact time of occurrence of the presynaptic and postsynaptic signals.

2. *Local mechanism.* By its very nature, a synapse is the transmission site where information-bearing signals (representing ongoing activity in the presynaptic and postsynaptic units) are in *spatiotemporal* contiguity. This locally available information is used by a Hebbian synapse to produce a local synaptic modification that is input specific.

3. *Interactive mechanism.* The occurrence of a change in a Hebbian synapse depends on signals on both sides of the synapse. That is, a Hebbian form of learning depends on a "true interaction" between presynaptic and postsynaptic signals in the sense that we cannot make a prediction from either one of these two activities by itself. Note also that this dependence or interaction may be deterministic or statistical in nature.

4. *Conjunctional* or *correlational mechanism.* One interpretation of Hebb's postulate of learning is that the condition for a change in synaptic efficiency is the conjunction of presynaptic and postsynaptic signals. Thus, according to this interpretation, the co-occurrence of presynaptic and postsynaptic signals (within a short interval of time) is sufficient to produce the synaptic modification. It is for this reason that a Hebbian synapse is sometimes referred to as a *conjunctional synapse.* For another interpretation of Hebb's postulate of learning, we may think of the interactive mechanism characterizing a Hebbian synapse in statistical terms. In particular, the correlation over time between presynaptic and postsynaptic signals is viewed as being responsible for a synaptic change. Accordingly, a Hebbian synapse is also referred to as a *correlational synapse.* Correlation is indeed the basis of learning (Eggermont, 1990).

5. Briefly explain single layer perceptron.

Ans:

In the formative years of neural networks (1943–1958), several researchers stand out for their pioneering contributions:

- McCulloch and Pitts (1943) for introducing the idea of neural networks as computing machines.
- Hebb (1949) for postulating the first rule for self-organized learning.
- Rosenblatt (1958) for proposing the perceptron as the first model for learning with a teacher (i.e., supervised learning).

The perceptron is the simplest form of a neural network used for the classification of patterns said to be *linearly separable* (i.e., patterns that lie on opposite sides of a hyperplane). Basically, it consists of a single neuron with adjustable synaptic weights and bias. The algorithm used to adjust the free parameters of this neural network first appeared in a learning procedure developed by Rosenblatt (1958, 1962) for his perceptron brain model.[1] Indeed, Rosenblatt proved that if the patterns (vectors) used to train the perceptron are drawn from two linearly separable classes, then the perceptron algorithm converges and positions the decision surface in the form of a hyperplane between the two classes. The proof of convergence of the algorithm is known as the *perceptron convergence theorem*. The perceptron built around a *single neuron* is limited to performing pattern classification with only two classes (hypotheses). By expanding the output (computation) layer of the perceptron to include more than one neuron, we may correspondingly form classification with more than two classes. However, the classes have to be linearly separable for the perceptron to work properly. The important point is that insofar as the basic theory of the perceptron as a pattern classifier is concerned, we need consider only the case of a single neuron. The extension of the theory to the case of more than one neuron is trivial.

The single neuron also forms the basis of an *adaptive filter*, a functional block that is basic to the ever-expanding subject of *signal processing*. The development of adaptive filtering owes much to the classic paper of Widrow and Hoff (1960) for pioneering the so-called *least-mean-square (LMS) algorithm*, also known as the *delta rule*. The LMS algorithm is simple to implement yet highly effective in application. Indeed, it is the workhorse of *linear* adaptive filtering, linear in the sense that the neuron operates in its linear mode. Adaptive filters have been successfully applied in such diverse fields as antennas, communication systems, control systems, radar, sonar, seismology, and biomedical engineering (Widrow and Stearns, 1985; Haykin, 1996).

The perceptron is built around a nonlinear neuron, namely, the *McCulloch–Pitts model* of a neuron. i.e. McCulluch-Pitts model of a neuron. Such a neuron model consist of a linear combiner followed by a hard limiter (performing the signum function), as depicted in Fig. 1 . The summing node of the neuronal model computes a linear combination of the inputs applied to its synapses, and also incorporates an externally applied bias. The resulting sum, that is, the induced local field, is applied to a hard limiter. Accordingly, the neuron produces an output equal to $+1$ if the hard limiter input is positive, and -1 if it is negative.

In the signal-flow graph model of Fig. 1 , the synaptic weights of the perceptron are denoted by $w_1, w_2, \ldots, w_m$. Correspondingly, the inputs applied to the perceptron

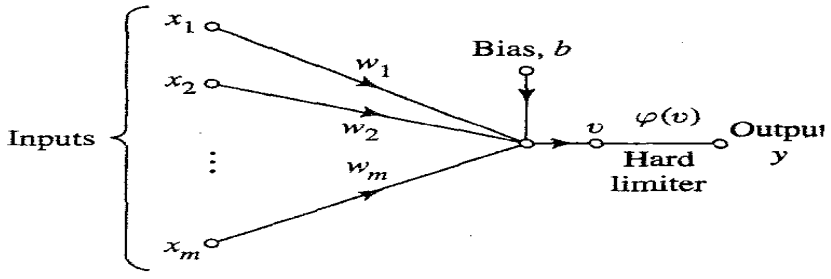

FIGURE 1 Signal-flow graph of the perceptron.

are denoted by $x_1, x_2, \ldots, x_m$. The externally applied bias is denoted by b. From the model we find that the hard limiter input or induced local field of the neuron is

$$v = \sum_{i=1}^{m} w_i x_i + b$$

The goal of the perceptron is to correctly classify the set of externally applied stimuli $x_1, x_2, \ldots, x_m$ into one of two classes, $\mathscr{C}_1$ or $\mathscr{C}_2$. The decision rule for the classification is to assign the point represented by the inputs $x_1, x_2, \ldots, x_m$ to class $\mathscr{C}_1$ if the perceptron output y is $+1$ and to class $\mathscr{C}_2$ if it is -1.

6. In perceptron convergence derive $W^TX<=0$ for all input vector belongs to class 6 .

Ans: Perceptron Convergence Theorem is equivalent to that of Fig. 1 , the bias $b(n)$ is treated as a synaptic weight driven by a fixed input equal to $+1$. We may thus define the $(m + 1)$-by-1 input vector

$$\mathbf{x}(n) = [+1, x_1(n), x_2(n), \ldots, x_m(n)]^T$$

where n denotes the iteration step in applying the algorithm. Correspondingly we define the $(m + 1)$-by-1 weight vector as

$$\mathbf{w}(n) = [b(n), w_1(n), w_2(n), \ldots, w_m(n)]^T$$

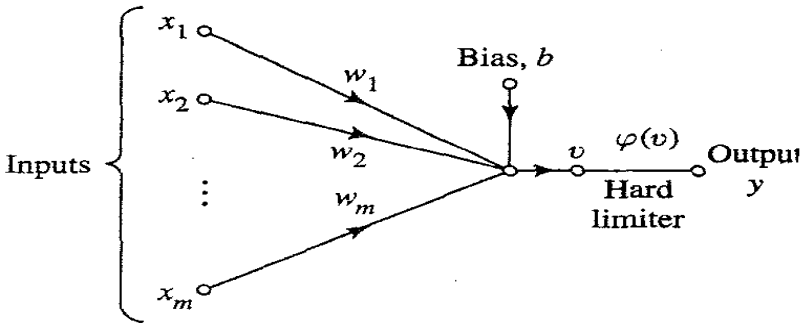

FIGURE 1 Signal-flow graph of the perceptron.

Accordingly, the linear combiner output is written in the compact form

$$\begin{aligned} v(n) &= \sum_{i=0}^{m} w_i(n)x_i(n) \\ &= \mathbf{w}^T(n)\mathbf{x}(n) \end{aligned} \tag{3.52}$$

where $w_0(n)$ represents the bias $b(n)$. For fixed n, the equation $\mathbf{w}^T\mathbf{x} = 0$, plotted in an m-dimensional space (plotted for some prescribed bias) with coordinates $x_1, x_2, \ldots, x_m$, defines a hyperplane as the decision surface between two different classes of inputs.

For the perceptron to function properly, the two classes $\mathscr{C}_1$ and $\mathscr{C}_2$ must be *linearly separable*. This, in turn, means that the patterns to be classified must be sufficiently separated from each other to ensure that the decision surface consists of a hyperplane. This requirement is illustrated in Fig. 2 for the case of a two-dimensional perceptron. In Fig. 2. a, the two classes $\mathscr{C}_1$ and $\mathscr{C}_2$ are sufficiently separated from each other for us to draw a hyperplane (in this case a straight line) as the decision boundary. If, however, the two classes $\mathscr{C}_1$ and $\mathscr{C}_2$ are allowed to move too close to each other, as in Fig. 2.b, they become nonlinearly separable, a situation that is beyond the computing capability of the perceptron.

Suppose then that the input variables of the perceptron originate from two linearly separable classes. Let $\mathscr{X}_1$ be the subset of training vectors $\mathbf{x}_1(1), \mathbf{x}_1(2), \ldots$ that belong to class $\mathscr{C}_1$, and let $\mathscr{X}_2$ be the subset of training vectors $\mathbf{x}_2(1), \mathbf{x}_2(2), \ldots$ that belong to class $\mathscr{C}_2$. The union of $\mathscr{X}_1$ and $\mathscr{X}_2$ is the complete training set $\mathscr{X}$. Given the sets

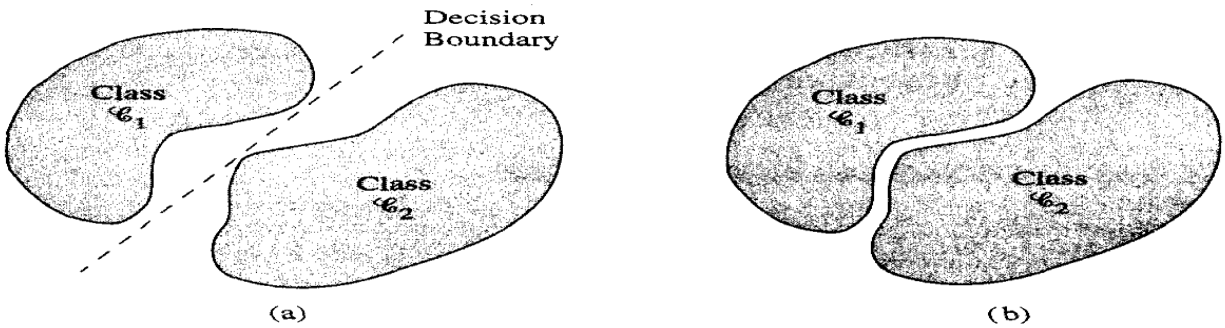

FIGURE 2 (a) A pair of linearly separable patterns. (b) A pair of nonlinearly separable patterns.

of vectors $\mathscr{X}_1$ and $\mathscr{X}_2$ to train the classifier, the training process involves the adjustment of the weight vector $\mathbf{w}$ in such a way that the two classes $\mathscr{C}_1$ and $\mathscr{C}_2$ are linearly separable. That is, there exists a weight vector $\mathbf{w}$ such that we may state

$$\begin{aligned} \mathbf{w}^T\mathbf{x} &> 0 \text{ for every input vector } \mathbf{x} \text{ belonging to class } \mathscr{C}_1 \\ \mathbf{w}^T\mathbf{x} &\leq 0 \text{ for every input vector } \mathbf{x} \text{ belonging to class } \mathscr{C}_2 \end{aligned} \tag{3.53}$$

In the second line of Eq. (3.53) we have arbitrarily chosen to say that the input vector $\mathbf{x}$ belongs to class $\mathscr{C}_2$ if $\mathbf{w}^T\mathbf{x} = 0$. Given the subsets of training vectors $\mathscr{X}_1$ and $\mathscr{X}_2$, the training problem for the elementary perceptron is then to find a weight vector $\mathbf{w}$ such that the two inequalities of Eq. (3.53) are satisfied.

7. Explain neural network Architecture

Ans:

There are mainly two types of ANNs: feed forward neural networks (FFNNs) and recurrent neural networks (RNNs). In FFNN there are no feedback loops. The flow of signals/information is only in the forward direction. The behavior of FFNN does not depend on past input. The network responds only to its present input. In RNN there are feedback loops (essentially FFNN with output fed back to input). Different types of neural network architectures are briefly described next.

1. ***Single-layer feed forward networks:*** It has only one layer of computational nodes (output layer). It is a feed forward network since it does not have any feedback. i.e. A neural network in which the input layer of source nodes projects into an output layer of neurons but not vice-versa is known as single feed-forward or acyclic network. In single layer network, 'single layer' refers to the output layer of computation nodes. The single layer feed-forward network consist of a single layer of weights, where the inputs are directly

connected to the outputs, via a series of weights. The synaptic links carrying weights connect every input to every output, but not other way. This way it is considered a network of feed-forward type. The sum of products of the weights and the inputs is calculated in each neuron node, and if the value is above some threshold (typically 0) the neuron fires and takes the activated value (typically 1); otherwise it takes the deactivated value (typically -1). The figure as given below.

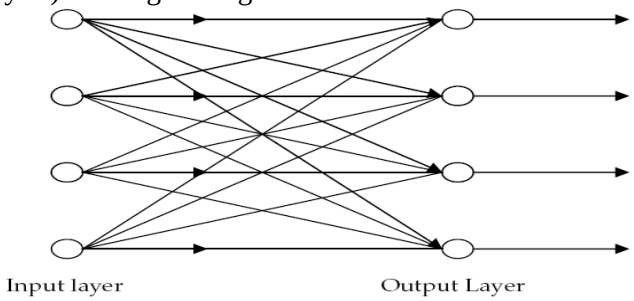

Fig. Single layer feed forward network

2. ***Multi-layer feed forward networks:*** It is a feed forward network with one or more hidden layers. The source nodes in the input layer supply inputs to the neurons of the first hidden layer. The outputs of the first hidden layer neurons are applied as inputs to the neurons of the second hidden layer and so on. If every node in each layer of the network is connected to every other node in the adjacent forward layer, then the network is called fully connected. If however some of the links are missing, the network is said to be partially connected. These networks can be used to realize complex input/output mappings. This type of network consists of one or more hidden layers, whose computation nodes are called hidden neurons or hidden units. The function of hidden neurons is to interact between the external input and network output in some useful manner and to extract higher order statistics. The source nodes in input layer of network supply the input signal to neurons in the second layer (1st hidden layer). The output signals of 2nd layer are used as inputs to the third layer and so on. The set of output signals of the neurons in the output layer of network constitutes the overall response of network to the activation pattern supplied by source nodes in the input first layer.

It consist of multiple layers. The architecture of this class of network have the input layer, output layer and one or more hidden layers. The computational units of the hidden layer are known as hidden neurons. The fig. is given below.

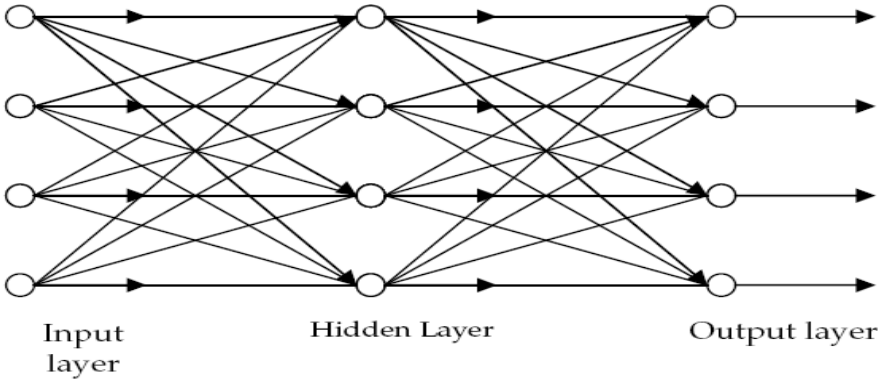

Fig. Multi-layer feed forward networks

Some Facts:

a. The hidden layer does intermediate computation before directing the input to output layer.

b. The input layer neurons are linked to the hidden layer neurons; the weights on these links are referred to as **input-hidden layer weights**.

c. The hidden layer neuron and the corresponding weights are reffered to as **output-hidden layer weights.**

d. A multi-layer feed forward network with e input neurons andn m1 neurons in the first hidden layers, m2 neurons in the second hidden layers, and n output neurons in the output layers is written as (e-m1-m2-n).

Short characterization of feed forward networks:

a. Typically, activation is fed forward from input to output through 'hidden layers', though many other architectures exist.
b. Mathematically, they implement static input-output mappings.
c. Most popular supervised training algorithm: back propagation algorithm.
d. Have proven useful in many practical applications as approximators of nonlinear functions and as pattern classificators.

3. ***Recurrent neural networks:*** A recurrent neural network is one in which there is at least one feedback loop. There are different kinds of recurrent networks depending on the way in which the feedback is used. In a typical case it has a single layer of neurons with each neuron feeding its output signal back to the inputs of all other neurons. Other kinds of recurrent networks may have self-feedback loops and also hidden neurons A feed forward neural network having one or more hidden layers with at least one feedback loop is known as recurrent network as shown in Fig. 4.9. The feedback may be a self-feedback, i.e., where output of neuron is fed back to its own input. Sometimes, feedback loops involve the use of unit delay elements, which results in nonlinear dynamic behavior, assuming that neural network contains nonlinear units.
It is differ from feed forward. A recurrent network has at least one feedback loop. There could be neurons with self-feedback links; that is the output of neuron id feedback into itself as input.
There are various other types of networks like; delta-bar-delta, Hopfield, vector quantization, counter propagation, probabilistic, Hamming, Boltzman, bidirectional associative memory, spacio-temporal pattern, adaptive resonance, self-organizing map, recirculation etc.
A recurrent neural network has (at least one) cyclic path of synaptic connections. The fig is given below.

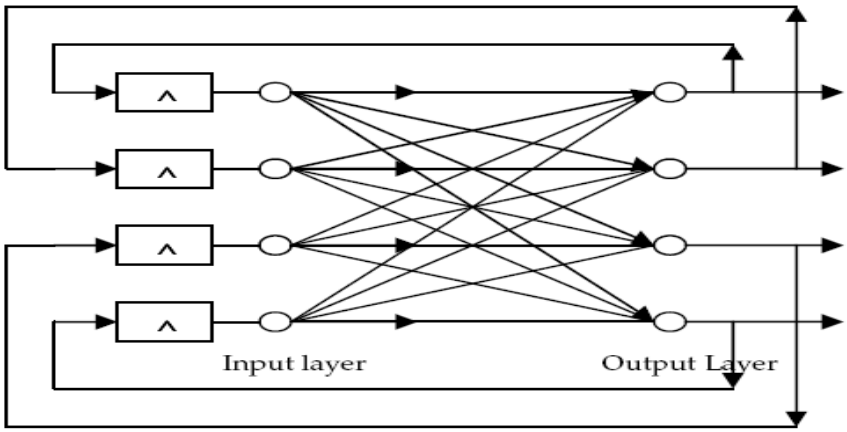

Fig. Recurrent neural networks

Basic characteristics:

a. All biological neural networks are recurrent.
b. Mathematically, they implement dynamical systems.
c. Several types of training algorithms are known, no clear winner.
d. Theoretical and practical difficulties by and large have prevented practical applications so far.

8. Explain reinforcement learning

Ans:

In *reinforcement learning,*[8] the learning of an input–output mapping is performed through continued interaction with the environment in order to minimize a scalar index of performance. Figure 2.7 shows the block diagram of one form of a reinforcement learning system built around a *critic* that converts a *primary reinforcement signal* received from the environment into a higher quality reinforcement signal called the *heuristic reinforcement signal,* both of which are scalar inputs (Barto et al., 1983). The

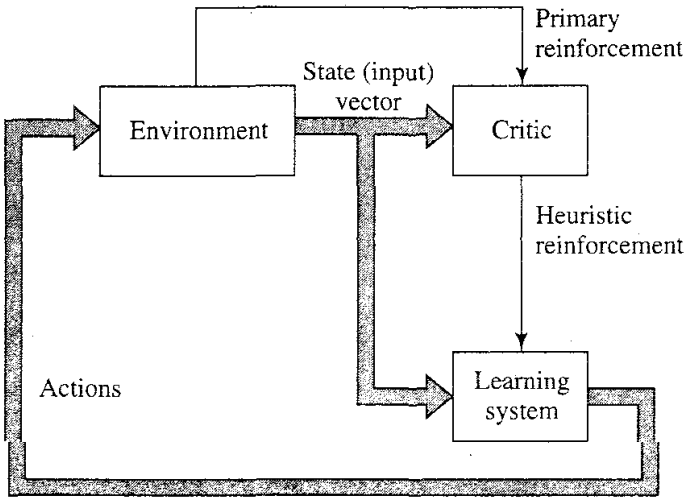

FIGURE 2.7 Block diagram of reinforcement learning.

system is designed to learn under *delayed reinforcement,* which means that the system observes a temporal sequence of stimuli (i.e., state vectors) also received from the environment, which eventually result in the generation of the heuristic reinforcement signal. The goal of learning is to minimize a *cost-to-go function,* defined as the expectation of the cumulative cost of *actions* taken over a sequence of steps instead of simply the immediate cost. It may turn out that certain actions taken earlier in that sequence of time steps are in fact the best determinants of overall system behavior. The function of the *learning machine,* which constitutes the second component of the system, is to *discover* these actions and to feed them back to the environment.

Delayed-reinforcement learning is difficult to perform for two basic reasons:

- There is no teacher to provide a desired response at each step of the learning process.
- The delay incurred in the generation of the primary reinforcement signal implies that the learning machine must solve a *temporal credit assignment problem.* By this we mean that the learning machine must be able to assign credit and blame individually to each action in the sequence of time steps that led to the final outcome, while the primary reinforcement may only evaluate the outcome.

Notwithstanding these difficulties, delayed-reinforcement learning is very appealing. It provides the basis for the system to interact with its environment, thereby developing the ability to learn to perform a prescribed task solely on the basis of the outcomes of its experience that result from the interaction.

Reinforcement learning is closely related to *dynamic programming,* which was developed by Bellman (1957) in the context of optimal control theory. Dynamic programming provides the mathematical formalism for sequential decision making. By casting reinforcement learning within the framework of dynamic programming, the subject matter becomes all the richer for it, as demonstrated in Bertsekas and Tsitsiklis