SYLLABUS **CSE-407-F** **NEURAL NETWORKS**

Sessional : 50 Marks
Theory : 100 Marks
Total : 150 Marks
Duration of Exam. : 3 Hrs.

Section-A

Overview of biological neurons: Structure of biological neurons relevant to ANNs.

Fundamental concepts of Artificial Neural Networks: Models of ANNs; Feedforward & feedback networks; learning rules; Hebbian learning rule, perception learning rule, delta learning rule, Widrow-Hoff learning rule, correction learning rule, Winner –lake all elarning rule, etc.

Section-B

Single layer Perception Classifier: Classification model, Features & Decision regions; training & classification using discrete perceptron, algorithm, single layer continuous perceptron networks for linearlyseperable classifications.

Multi-layer Feed forward Networks: linearly non-seperable pattern classification, Delta learning rule for multi-perceptron layer, Generalized delta learning rule, Error back-propagation training, learning factors, Examples.

Section-C

Single layer feed back Networks: Basic Concepts, Hopfield networks, Training & Examples.

Associative memories: Linear Association, Basic Concepts of recurrent Auto associative memory: rentrieval algorithm, storage algorithm; By directional associative memory, Architecture, Association encoding & decoding, Stability.

Section-D

Self organizing networks: UN supervised learning of clusters, winner-take-all learning, recall mode, Initialisation of weights, seperability limitations.

NOTE : Examiner will set 9 questions in total, with two questions from each section and one question covering all sections which will be Q.1. This Q.1 is compulsary and of short answer type. Each question carries equal mark (20marks). Students have to attempt 5 questions in total at least one question from each section.

NEURAL NETWORKS

Dec 2013

Paper Code: CSE-407-F

Figures in the bracket indicate full marks.

Note: Attempt a total of five questions. Q. No. 1 is compulsary. One question each to be attempted from each section. All questions carry equal marks.

Q.1.(a) Differentiate Supervised and Unsupervised Learning., **(5)**

Ans. Supervised Learning : In supervised learning, we assume that at each instant of time when the input is applied, the desired response *d* of the system is provided by the teacher. The network then processes the inputs and compares its resulting outputs against the desired output *d*. Errors are then propagated back through the system, causing the system to adjust the weights, which control the network. This process occurs over and over as the weights are continually decreases. Since we assume the adjustable weights, the teacher may implement a reward – and -punishment scheme to adapt the network's weight matrix *W*. The set of data, which enables the training, is called the "training set". During the training of a network the same set of data is processed many times as the connection weights are ever refined.

Example : Like in the class a teacher is always present to ask the questions from the students. If the students give the correct ans., then the teacher may give the reward to the students, otherwise if the student is not able to give the right answer, then teacher will again explain that topic to the students until they will well understand the problem.

Unsupervised Learning : The other type of learning is called unsupervised learning. In unsupervised learning, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred toas self-organization or adaptation.

Example : Learning without supervision corresponds to learning the subject from a videotape lecture covering the material but not including any teacher's involvement. Therefore in this the students can't get the explanations of the unclear question.

Q.1.(b) Role Learning factors. **(5)**

Ans. Role learning factors are as follows :

– *Initialization of Weights* : The weights of the network to be trained are initialized at small random values. The initialization affects the complete solution. If all the weights start out with equal weight values and if the soultion requries that unequal weights be developed, the network may not be trained properly. artificial neural networks typically start out with randomized weights for all their neurons.

If training continues beyond a certain low error level, it will result in undersirable drift of weights. This casues error to increase and the quality of mapping by the network decreases. Therefore the choice of initial weights is however only one of several factors affecting training of error towards an acceptable error minimization.

– *Learning Constant* : Careful selection of step size (learning constant η) is often necessary to ensure the smooth convergence. However optimum value of η depends on problem being solved and there is no single learning constant value suitable for different training cases.

$$\eta = \frac{1.5}{N_1^2 + N_2^2 + N_n^2}$$

Where N_1 is the number of patterns of type 1 and there are total n different patterns.

The learning coefficient cant be –ve because this would cause the change of weight vector to move from ideal weight vector position. If the value of η is zero then, no learning takes place and hence the value of η must be +ve.

– *Sigmoidal gain* : If sigmoidal function is selected, the input-output relationship of the neuron can be set as $O = 1/[1 + \exp(-\lambda(I + O))]$ where λ is a scaling factor known as sigmoidal gain.

In some problems, when the weights become large and force the neuron to operate in a region where the sigmoidal function is very flat, a better method of coping with the network paralysis is to adjust the sigmoidal gain.

By decreasing the scvaling factor, this activation function can be spread out on wide range so that training proceeds faster.

– *Threshold value* : O in above equation is called as threshold value of a neuron, or bias or the noise factor. A neuron generates the output if the weighted sum of the input exceeds the threshold value.

Q.1.(c) Bi-directional Associative Memory. **(5)**

Ans. Bi-directional Associative Memory : BAM (Bi-directional Associative Memory) is a hetero asosciative, content-addressable memory consisting of 2 layers.

The network structure of BAM is similar to that of linear associator but connections are bidirectional that is

$$w_{ij} = w_{ji} \quad \text{for} \begin{cases} i = 1, 2, \ldots\ldots, m \\ j = 1, 2, \ldots\ldots, n \end{cases}$$

i.e., BAM allows forward and backward flow of information between the layers. In BAM the units in both layers serve as both input and output units depending on the direction of propagation.

Propagating the signals from X layer to Y layer makes the units in X layer to act as input units while units in Y layer act as output units. Same is true for other direction because memory is bidirectional.

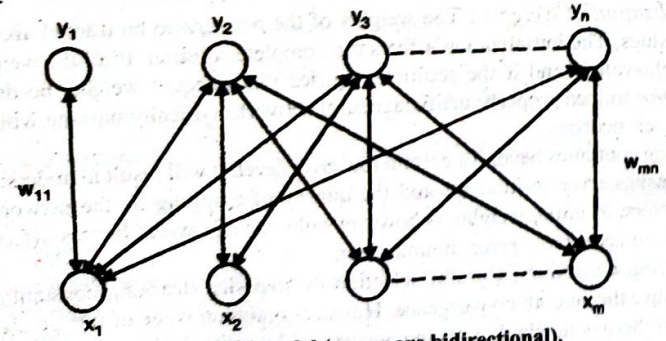


Fig. : BAM model (arrow are bidirectional).

BAM model can perform both auto and hetero associative recall of stored information.

Like is linear associator and hopfield network encoding in BAM can be carried out by using

$$w_k = x_k^T y_k \quad \text{(to store a single associated pattern pair)}$$

and $$w = \alpha \sum_{k=1}^{p} w_k \quad \text{(to store simultaneously several associated pattern pairs)}$$

This process is generally called storage or encoding.

Q.1.(d) What are separability limitations. **(5)**

Ans. A single layer perceptron consists of an input layer and an output layer. The activation function applied is hard-limiting function. An output unit will assume the value 1 if the sum of weighted input is greater than its threshold. *i.e.*,

$W_{ji} X_i > W_j$ where W_{ji} is weight from unit i to unit j.

X_i is the input from unit i and Δ_j is the threshold on unit j

Let there are 2 classes A and B. If $W_{ji} X_i > \Delta_j$ where $i = 1, 2, \ldots n$ forms a hyperplane, dividing the space in 2 halves.

Linear separability : When a linear hyperplane exists to place the instances of one class on one side and those of other class on the other side of plane. Then this is called Linear Separability. For example in case of OR gate we can draw a linear hyperplane to separate the input whose corresponding outputs are 1 on one side and the inputs whose corresponding outputs are 0 an other side of the plane.

Table : OR Gate

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	1

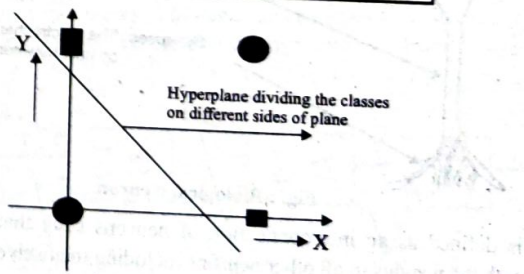


Fig. : The OR gate

But all the classification problems are not linear separable. Fox example the EX-OR problem is not linear separable because EX-OR is non equality gate.

Section-A

Q.2. What are biological Neurons ? How they help in creating artificial neuron model ? Also discuss the significance of such models. **(20)**

Ans. Biological Neuron : The elementary nerve cell called a neuron is the fundamental building block of biological neural network. The three main components of a biological neuron are :

1. A neuron cell body called *soma.*
2. Branching extensions called *dendrites* for receiving input, and
3. An *axon* that carries the neuron's output to the dendrites of other neurons.

In the human brain, a typical neuron collects signals from others through a host of fine structures called *dendrites*. The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. The axon-dendrite contact organ is called a synapse. At the end of each branch, a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes. The neuron is able to respond to the total of its inputs aggregated within a short time interval called period of latent summation.

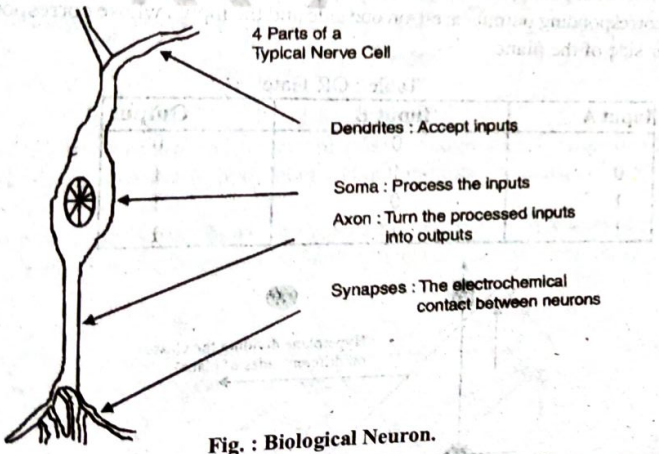


Fig. : Biological Neuron.

ANN is defined as an interconnection of neurons such that the neuron outputs are connected, through the weights to all other neurons including themselves; both lag-free and delay connections are allowed.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

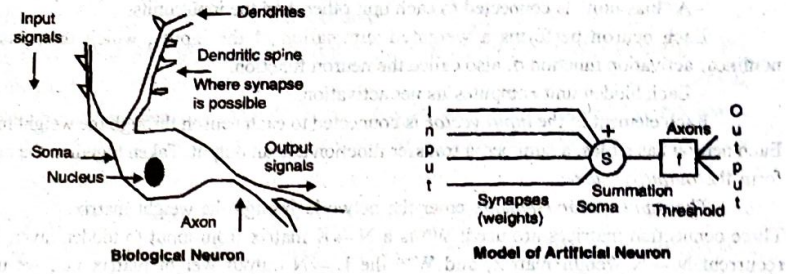


Fig. : Similarities between Biological Neuron and ANN

A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. Other advantages include :

1. Adaptive learning : An ability to learn how to do tasks based on the data given for training or initial experience.

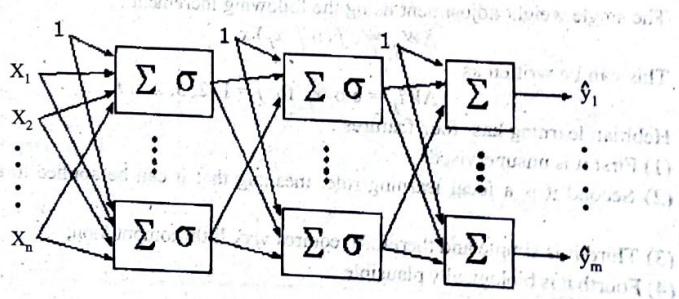
2. Self-Organization : An ANN can create its own organization or representation of the information it receives during learning time.

3. Real Time Operation : ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.

4. Fault Tolerance via Redundant Information Coding : Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

Q.3.(a) Make an arbitrary feed forward network with neurons in input layer, 2 in hidden and 2 in output layer. Explain the concept of Input vector, output vector connection matrix, signal (or activation function). **(10)**

Ans. Feed-forward neural networks are the most popular and most widely used models in many practical applications. Each arrow in the figure symbolizes a parameter in the network. The network is divided into layers. The input layer consists of just the inputs to the network. Then, there is a hidden layer which consists of any number of neurons or hidden units placed in parallel.



– A "bias unit" is connected to each unit other than the input units.

Each neuron performs a weighted summation of the inputs, which then passes a nonlinear activation function σ, also called the neuron function.

– Each hidden unit computes its net activation.

Each element of the *input vector* is connected to each neuron through the weight matrix. Each neuron has a bias, a summer, a transfer function and an output. Taken together, the outputs form the *output vector*.

The *input vector elements* enter the network through the weight matrix.

Three connection matrices are used: W^{in} is a $N \rightarrow K$ matrix from input to hidden layer, W the recurrent $N \rightarrow N$ weight matrix, and W^{out} the $L \rightarrow N$ output weight matrix i.e., we use no connections from input units to output, and no feedback from the output back into the network. Each neuron performs a weighted summation of the inputs, which then passes a nonlinear *activation function* σ, also called the *neuron* function. The nonlinear activation function in the neuron is usually chosen to be a smooth step function. The default is the standard sigmoid.

Sigmoid $[x] = 1/1 + e^{-x}$

Q.3.(b) Describe Hebbian Learning Rule. **(10)**

Ans. Hebbian Learning Rule : In Hebbian learning, weights between learning nodes are adjusted so that each weight bette represents the relationship between the nodes. Nodes which tend to be positive or negative at the same time will have strong positive weights while those which tend to be opposite will have strong negative weights. Nodes that are uncorrelated will have weights near zero. For example, if two nodes *A* and *B* are often simultaneously active, Hebbian learning will increase the connection strength between the two so that excitation of either one tends to cause excitation of the other. On the other hand, if nodes *A* and *C* were of opposite activations at all times, then Hebbian learning would gradually decrease the connection in between below zero so that an excited *A* or *C* would inhibit the other.

For this rule the learning signal is equal simple to the neuron's output

We have $$r \sim = f(W_i^t \, x) \quad \text{...(i)}$$

The increment ΔW_i of the weight vector becomes

$$\Delta W_i = c f(W_i^t \, x) \, x \quad \text{...(ii)}$$

The single weight adjustment using the following increment :

$$\Delta W_{ij} = c f(W_i^t \, x_i) \, x_j \quad \text{...(iii)}$$

This can be written as

$$\Delta Wi_j = c \, o_i \, x_j \quad \text{for } j = 1, 2, 3, \ldots\ldots n \quad \text{...(iv)}$$

Hebbian learning has four features :

(1) First it is unsupervised;

(2) Second it is a local learning rule, meaning that it can be applied to a network in parallel;

(3) Third it is simple and therefore requires very little computation;

(4) Fourth it is biologically plausible.

Section-B

Q.4.(a) What do you mean by simple perceptron ? Explain the multi layer perceptron model. **(10)**

Ans. Simple perceptron : Perceptron is a feedforward neural network. In priniple, a perceptron can contain an arbitrary number of layers of neurons, but only rather simple cases have been studied in depth. Here we begin with the so-called simple perceptron, where the input feeds directly into the output layer without the intervention of hidden layers of neurons. Denote the input by $x_k, k = 1, \ldots, n \left(\text{or } x \in R^n\right)$, and the output by $y = \pm 1$. The activation function $\phi(x)$ may be a continuous function, tanh $(w^T x)$:

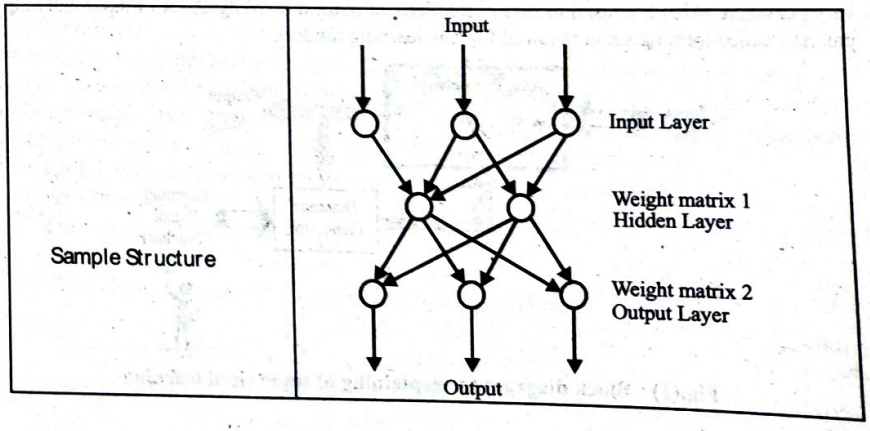
$$\phi(x) = \tanh(\lambda x) = \frac{1 - e^{-2\lambda x}}{1 + e^{-2\lambda x}} \quad \text{...(i)}$$

or a discrete function, *i.e.*, as $\lambda \longrightarrow \infty$ in (i), $\phi(x) \longrightarrow \text{sgn}(x)$ the hard limiter.

Multi layer perceptron model : An MLP is a network of simple neurons called perceptrons. The basic concept of a single perceptron was introduced by Rosenblatt in 1958. A Multilayer Perceptron is a feedforward neural network with at least one hidden layer. It can deal with nonlinear classification problems.

Example : Suppose an output node receives 2 inputs, its threshold is set to 1.5 and its input weights are both set to 1. Therefore when both the input units are active (i.e. 1) the output node will be active. In this case the output node performs a logical AND operation. On the other hand if threshold is set to 0.5, then any active input can activate the node. In this case the output node performs a logical OR operation. Thus by choosing a different set of weights and threshold, a node can implement a different logical operation.

Multilayer Perceptron Characteristics



Type	Feed-forward
Neuron Layers	1 input layer 1 or more hidden layers 1 output layer
Input value types	Binary
Activation Functions	Hard limiter/sigmoid
Learning Method	Supervised
Learning algorithm	Delta learning rule Back-propagation (mostly used)

Q.4.(b) Explain error back propagation training algorithm. **(10)**

Ans. Refer Q.5.(b) of paper Dec 2014.

Q.5.(a) What are various learning rules for multiperceptron neural networks ? Explain these learning rules with their equations. **(10)**

Ans. Various learning rules for multiperceptron neural networks are as follows :

(i) Supervised learning : The learning rule is provided with a set of examples (the trainng set) of proper network behaviour.

$$\{x_1, d_1\}, \{x_2, d_2\}, \ldots, \{x_n, d_n\} \quad \ldots(i)$$

where (x_n) is an input to the network, (d_n) is the corresponding correct target (desired) output. As the inputs are applied to the network, the network outputs are compared with the targets. The learning rule is then used to adjust the weights and the biases of the network in order to move the network outputs closer to the targets (desired).

In supervised learning we assume that at each instant of time when the input is applied, the desired response (d) of the system is provided by the teacher. This is illustrated in fig.(1), the distance between the actual and the desired response serve as an error measure and is used to correct network parameter externally.

For instance, in learning classifications of input patterns or situations with known responses, the error can be used to modify the weights so that the error decreases. This mode of learning is very pervasive. Also it is used in many situations of natural learning. A set of input and output patterns called training set is required for this learning mode.

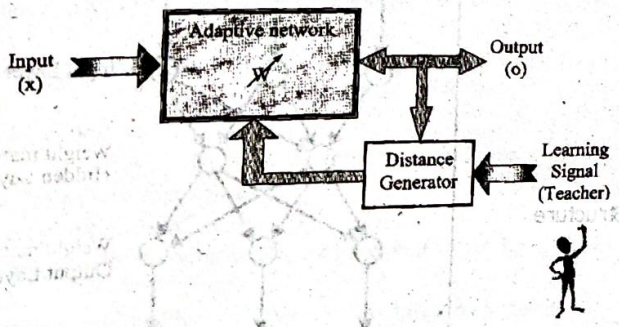


Fig.(1) : Block diagram for explaining of supervised learning

(ii) Unsupervised learning : In unsupervised learning, the weights and biases are modified in response to netork input only. There are no target outputs available. Fig.(2) shows the block diagram of unsupervised learning rule. In learning without supervision the desired response is not known ; thus, explicit error information cannot be used to improve network behavior. Since no information is available as to correctness or incorrectness of responses, learning must somehow be accomplished based on observations of responses to inputs that we have marginal or no knowledge about.

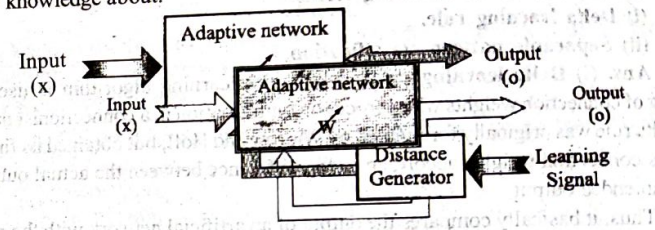


Fig.(2) : Block diagram for explaining of unsupervised learning

Unsupervised learning algorithms use patterns that are typically redudant raw data having no label regarding their class membership, or associations. In this mode of learning, a network must discover for itself any possibly existing patterns; regularities, separating properties, etc., while discovering these the network undergoes change in its parameters, unsupervised learning is sometimes called learning without teacher. This terminology is not the most appropriate because learning without a teacher is not possible at all. Although, the teacher does not have to be involved in every training step, he has to set goals even in an unsupervised learning mode.

(iii) Hebbian Learning Rule : In Hebbian learning, weights between learning nodes are adjusted so that each weight bette represents the relationship between the nodes. Nodes which tend to be positive or negative at the same time will have strong positive weights while those which tend to be opposite will have strong negative weights. Nodes that are uncorrelated will have weights near zero. For example, if two nodes A and B are often simultaneously active, Hebbian learning will increase the connection strength between the two so that excitation of either one tends to cause excitation of the other. On the other hand, if nodes A and C were of opposite activations at all times, then Hebbian learning would gradually decrease the connection in between below zero so that an excited A or C would inhibit the other.

For this rule the learning signal is equal simple to the neuron's output

We have $$r \sim = f(W_i^t \, x) \quad \ldots(i)$$

The increment ΔW_i of the weight vector becomes

$$\Delta W_i = c\, f(W_i^t \, x)\, x \quad \ldots(ii)$$

The single weight adjustment using the following increment :

$$\Delta W_{ij} = c\, f(W_i^t \, x_i)\, x_j \quad \ldots(iii)$$

This can be written as

$$\Delta W_j = c\, o_i\, x_j \quad \text{for } j = 1, 2, 3, \ldots\ldots n \quad \ldots(iv)$$

Hebbian learning has four features :

(1) First it is unsupervised;

(2) Second it is a local learning rule, meaning that it can be applied to a network in parallel;

(3) Third it is simple and therefore requires very little computation;

(4) Fourth it is biologically plausible.

Q.5.(b) Define the following terms :

(i) Delta learning rule.

(ii) Separable pattern classification. **(10)**

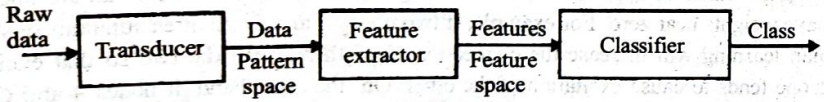
Ans. (i) Delta learning rule : The Delta learning algorithm is used for automatic updating of connection weights when processing information in a connectionist or neural network. This Delta rule was originally formulated by Widrow and Hoff, but obtained its final name because it adjusts connection weights according to the difference between the actual output of a network and the intended output.

Thus, it basically compares the output of an artificial network with the output that it was intended to produce, and then adjusts the connections within the network so as to better produce the intended output.

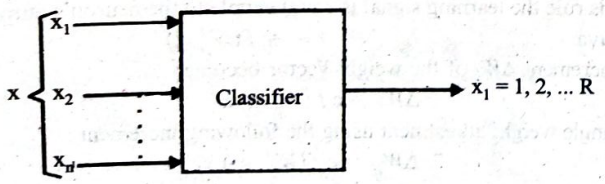
Ans. (ii) Separable pattern classification : The main goal of pattern classification is to assign a physical object, event or phenomenon one of prespecified classes (categories).

For example : We (human beings) classify various objects into different categories like living non-living things, plants, weather, voice etc. The interpretation of data has been learned as a result of repetitive inspection and classification of examples.

When a person preceives a pattern, an inductive inference is made and the perception is associated with some general concepts derived froms persons past experience. The problem of patterns classification may be regarded as one of discriminating the input data within object population via search for the different attributes among members of the population.



(a) Block diagram of classification system



(b) General diagram of pattern classifier

Fig.

In some of the applications, the various classifying aids frame helpful because of the growing complexity of the human environment. Fig. shows the block diagram of recognition and classification system.

Applications of Pattern Classifier are

(i) Fingerprint Identification,

(ii) Radar and Signal Detection,

(iii) Speech Recognition etc.

Section-C

Q.6. Explain the various architectures of Hopfield network in detail. How learning process occurs in Hopfield network? **(20)**

Ans. Refer Q.6 of paper Dec 2014.

Q.7. Design a bi-directional Associative memory to Encode the following pattern:

A_1 = 100001 **B_1 = 11000**
A_2 = 011000 **B_2 = 10100**
A_3 = 001011 **B_3 = 01110**
Check it for A_3 **(20)**

Ans. Consider $N = 3$ pattern pairs (A_1, B_1), (A_2, B_2), (A_3, B_3) given by

A_1 = (1 0 0 0 0 1) B_1 = (1 1 0 0 0)
A_2 = (0 1 1 0 0 0) B_2 = (1 0 1 0 0)
A_3 = (0 0 1 0 1 1) B_3 = (0 1 1 1 0)

Convert these three binary pattern to bipolar form replacing 0s by –1s.

X_1 = (1 –1 –1 –1 –1 1) Y_1 = (1 1 –1 –1 –1)
X_2 = (–1 1 1 –1 –1 1) Y_2 = (1 –1 1 –1 –1)
X_3 = (–1 –1 1 –1 1 1) Y_3 = (–1 1 1 1 –1)

The correlation matrix M is calculated as 6 × 5 matrix

$$M = X_1^T Y_1 + X_2^T Y_2 + X_3^T Y_3 = \begin{bmatrix} 1 & 1 & -3 & -1 & 1 \\ 1 & -3 & 1 & -1 & 1 \\ -1 & -1 & 3 & 1 & -1 \\ -1 & -1 & -1 & 1 & 3 \\ -3 & 1 & 1 & 3 & 1 \\ -1 & 3 & -1 & 1 & -1 \end{bmatrix}$$

Suppose we start with $\alpha = X_3$, and we hope to retrieve the associated pair Y_3. The calculations for the retrieval of Y_3 yield :

αM = [–1 –1 1 –1 1 1] (M) = (–6 6 6 6 –6)
$\Phi(\alpha M) = \beta'$ = (–1 1 1 1 –1)
$\beta' M^T$ = (–5 –5 5 –3 7 5)
$\Phi(\beta' M^T)$ = (–1 –1 1 –1 1 1) = α'
$\alpha' M$ = (–1 –1 1 –1 1 1) M = (–6 6 6 6 –6)
$\Phi(\alpha' M) = \beta''$ = (–1 1 1 1 1 –1)

This retrieved pattern β' is same as Y_3.

Hence, $(\alpha_f, \beta_f) = (X_3, Y_3)$ is correctly recalled, a desired result.

Section-D

Q.8. Explain in detail unsupervised learning of clusters. **(20)**

Ans. Unsupervised learning of clusters : Learning is based on clustering of input data. No prior knowledge is assumed to be available regarding an input membership in a particular class. Rather, gradually detected characteristics and a history of training will be used to assist the network in defining classes and possible boundaries between them.

Clustering : Clustering is understood to be the grouping of similar objects and separating of dissimilar ones. The objective of clustering neural networks is to categorize or cluster data. The classes must first be found from the correlations of an input data stream. Since the network actually deals with unlabeled data, the clustering should be followed by labeling clusters with appropriate category names or numbers. This process of providing the category of objects with a label is usually termed as calibration.

Although the algorithm won't have names to assign to these clusters, it can produce them and then use those clusters to assign new examples into one or the other of the clusters. This is a data-driven approach that can work well when there is sufficient data; for instance, social information filtering algorithms, such as those that Amazon.com use to recommend books, are based on the principle of finding similar groups of people and then assigning new users to groups. In some cases, such as with social information filtering, the information about other members of a cluster (such as what books they read) can be sufficient for the algorithm to produce meaningful results. In other cases, it may be the case that the clusters are merely a useful tool for a human analyst. Unfortunately, even unsupervised learning suffers from the problem of overfitting the training data. There's no silver bullet to avoiding the problem because any algorithm that can learn from its inputs needs to be quite powerful.

Suppose we are given a set of patterns without any information as to the number of classes that may be present in the set. The clustering problem in such a case is that of identifying the number of classes according to a certain criterion, and of assigning the membership of the patterns in these classes. The clustering technique presented below :

Assumptions : The number of classes is known a priori. The pattern set $\{x_1, x_2, \ldots\ldots, x_n\}$ is submitted to the input to determine decision functions required to identify possible clusters. Since no information is available from the teacher as far as the desired classifier's responses, we will use the similarity of incoming patterns as the criterion for clustering. To define a cluster, we need to establish a basis for assigning patterns to the domain of particular cluster.

How we will assign a pattern to domain of particular cluster? There are 2 rules for this.

(1) The most common similarity rule is to find the Euclidean distance between 2 patterns x for x_i defined as

$$|X - X_i| = [(X - X_i)^t (X - X_i)]^{1/2} \quad \ldots(i)$$

Smaller the distance, closer the patterns. Using (i) the distance between all the pairs are computed. Now how to distinguish the clusters? A distance T can then be chosen to discriminate clusters. The value T is understood as the maximum distance between patterns within a single cluster. Fig.(a) shows an example of two clusters with a T value chosen to be greater than the typical within-cluster distance but smaller than the between-cluster distance.

Let there are 2 patterns X_1 and X_2 and we have to find which pattern is similar to pattern X. Then using (1) distance is calculated. The value for which the distance is small, closer the pattern is with x.

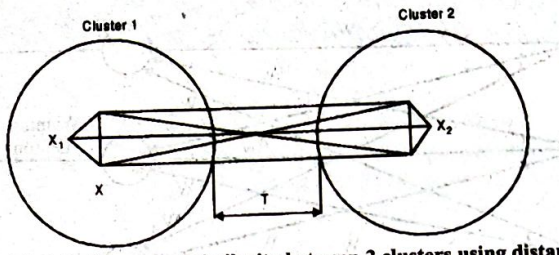


Fig.(a) : Measure of similarity between 2 clusters using distance

(2) Another similarity rule is the cosine of the angle between x and x_i:

$$\cos \alpha = \frac{(X^t X_i)}{\| X \| . \| X_i \|} \quad \ldots(ii)$$

This rule is particularly useful when clusters develop along certain principal and different axes as shown in fig.(b). For $\cos \alpha_2 < \cos \alpha_1$ pattern x is more similar to x_2 than to x_1. It would thus be natural to group it with the second of the two apparent clusters. To facilitate this decision, the threshold angle α_t can be chosen to define the minimum angular cluster distance. It should be noted, however, that the measure defined in equation (ii) should be used according to certain additional qualifications. If the angular similarity criterion equation (ii) is to be efficient, vectors x_1, x_2 and x should be of comparable, or better, identical lengths.

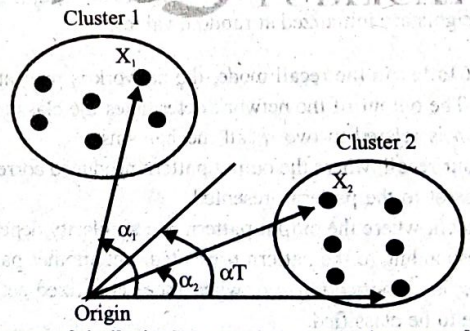


Fig.(b) : Measure of similarity between 2 clusters using normalized scalar product

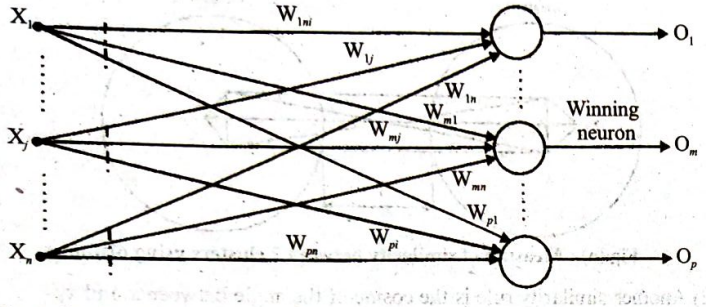
Q.9. Write short note on :

(a) Winner-take-all

(b) Recall Mode **(20)**

Ans. (a) Winner take all learning : (i) This rule is for unsupervised mode and it differ from all learning rules.

(ii) In this neurons are arranged in a layer of P units. This rule is an example of competitive learning and is used for unsuperbised training mode.



(iii) Learning is based on the fact that one of the neurons in the layer (suppose m) has maximum response due to input X, then that neuron is declared as Winner. As a result of this Winning event weight vector $w_m = [w_{m1}, w_{m2}, \ldots\ldots\ldots w_{mn}]^t$

(iv) The increment is the weight vector is computed as

$$\Delta w_m = \alpha (X - w_m) \quad \ldots(i)$$

for single weight adjustment for $j = 1, 2, \ldots\ldots, n$.

$\alpha > 0 \Rightarrow$ small learning constant

α decrease as learning constant increases.

(v) The winner selection is based on the criteria of maximum activation among all participating neurons in the competition :

$$\Delta w_{mt} \; X = \max \; (w_i^t x) \quad \text{where } i = 1, 2, \ldots\ldots, P \quad \ldots(ii)$$

(vi) In this rule weights are initialized at random values.

Ans. (b) Recall Mode : In the recall mode, the network is presented with the feature vector of a single sample. The output of the network determines the class to which the sample belongs. H*etero-association* is related to two recall mechanisms:

(i) Nearest-neighbour recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented.

(ii) Interpolative *r*ecall, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented. Yet another paradigm, which is a variant associative mapping is classification, *i.e.* when there is a fixed set of categories into which the input patterns are to be classified.

NEURAL NETWORKS

Dec - 2014
Paper Code:CSE-407-F

Note : ***Students have to attempt five questions in total, selecting at least one question from each section. Question No. 1 is compulsory. Each question carries equal marks (20 marks).***

Q.1.Write short note on : **(4 × 5 = 20)**

(a) Suppervised & unsupervised learning,
(b) Linear associator
(c) Weight adjustment
(d) Auto associative memory

Ans.(a) Supervised Learning : In supervised learning, we assume that at each instant of time when the input is applied, the desired response d of the system is provided by the teacher. The network then processes the inputs and compares its resulting outputs against the desired output d. Errors are then propagated back through the system, causing the system to adjust the weights, which control the network. This process occurs over and over as the weights are continually decreases. Since we assume the adjustable weights, the teacher may implement a reward – and -punishment scheme to adapt the network's weight matrix W. The set of data, which enables the training, is called the "training set". During the training of a network the same set of data is processed many times as the connection weights are ever refined.

Example : Like in the class a teacher is always present to ask the questions from the students. If the students give the correct ans., then the teacher may give the reward to the students, otherwise if the student is not able to give the right answer, then teacher will again explain that topic to the students until they will well understand the problem.

Unsupervised Learning : The other type of learning is called unsupervised learning. In unsupervised learning, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred toas self-organization or adaptation.

Example : Learning without supervision corresponds to learning the subject from a videotape lecture covering the material but not including any teacher's involvement. Therefore in this the students can't get the explanations of the unclear question.

Ans. (b) Linear associator : The linear associator is one of the simplest and first studied associative memory model. Linear associator is a feedforward type network where the output is produced in a single feedforward computation.

In the fig., all the m input units are connected to all the n output units via the connection weight matrix $W = [w_{ij}]_{m \times n}$ where w_{ij} denotes the synaptic strength of the unidirectional connection from the ith input unit to the jth output unit.

It is the connection weight matrix that stores the p different associated pattern pairs $\{(X_k, Y_k) | k = 1, 2, \ldots, p\}$ where $X_k \hat{I} \{-1,+1\}_m$ and $Y_k \hat{I} \{-1,+1\}_n$ in a distributed representation.

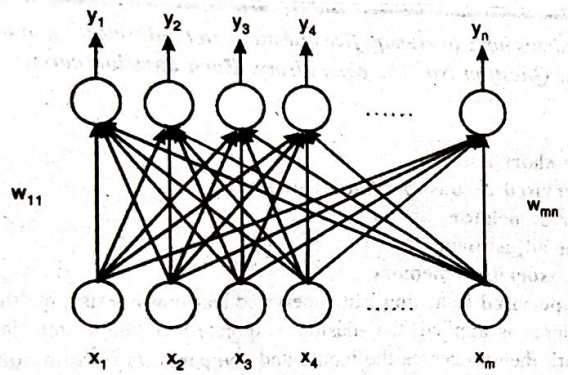


Fig. : Linear Associator

Ans. (c) Weight adjustment : Learning in a neural network is called *training*. Like training in athletics, training in a neural network requires a coach, someone that describes to the neural network what it should have produced as a response. From the difference between the desired response and the actual response, the *error* is determined and a portion of it is propagated backward through the network. At each neuron in the network the error is used to adjust the weights and threshold values of the neuron, so that the next time, the error in the network response will be less for the same inputs.

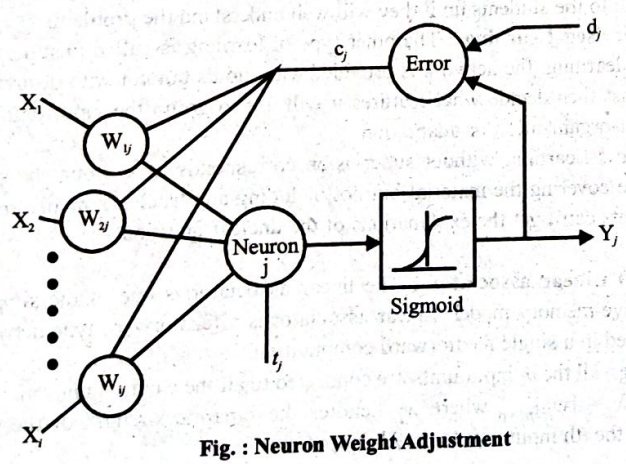
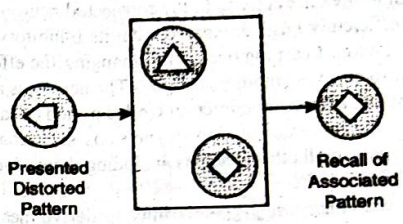


Fig. : Neuron Weight Adjustment

Ans. (d) Auto associative memory : A n auto-associative memory, also know as auto-associative correlator, is used to retrieve a previously stored pattern that most closely resembles the current pattern.



Auto-associative memory

Section-A

Q.2. What are biological neurons ? How help in creating artifical neuron model. Also discuss the significance of such models. **(20)**

Ans. Biological Neuron : The elementary nerve cell called a neuron is the fundamental building block of biological neural network. The three main components of a biological neuron are :

1. A neuron cell body called *soma.*
2. Branching extensions called *dendrites* for receiving input, and
3. An *axon* that carries the neuron's output to the dendrites of other neurons.

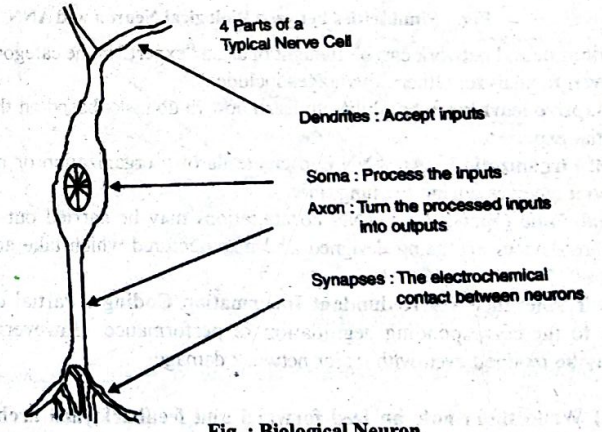


Fig. : Biological Neuron.

In the human brain, a typical neuron collects signals from others through a host of fine structures called *dendrites*. The neuron sends out spikes of electrical activity through a long, thin

stand known as an axon, which splits into thousands of branches. The axon-dendrite contact organ is called a synapse. At the end of each branch, a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes. The neuron is able to respond to the total of its inputs aggregated within a short time interval called period of latent summation.

ANN is defined as an interconnection of neurons such that the neuron outputs are connected, through the weights to all other neurons including themselves; both lag-free and delay connections are allowed.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

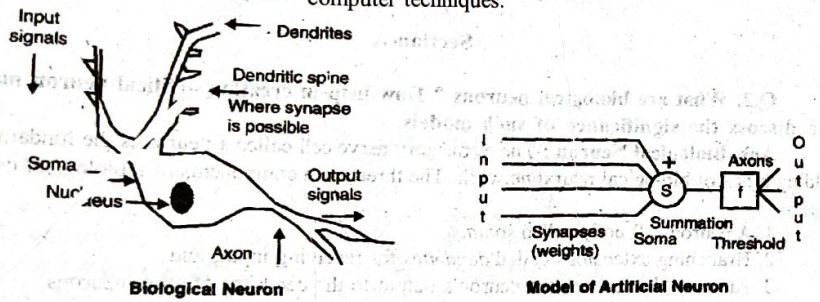


Fig. : Similarities between Biological Neuron and ANN

A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze. Other advantages include :

1. Adaptive learning : An ability to learn how to do tasks based on the data given for training or initial experience.

2. Self-Organization : An ANN can create its own organization or representation of the information it receives during learning time.

3. Real Time Operation : ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.

4. Fault Tolerance via Redundant Information Coding : Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

Q.3.(a) Write short note on feed forward and feedbackward architecture. (10)

Ans. Feedforward Network : Feedforward neural network is a biologically inspired classification algorithm. It consists of a (possibly large) number of simple neuron-like processing units, organized in layers. Every unit is a layer is connected with all the units in the previous

layer. These connections are not all equal; each connection may have a different strength or weight. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called nodes. Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called feedforward neural networks.

Fig.(1) : Block diagram of Feedforward Network

Let us consider the architecture of feedforward architecture of m neuron receiving n inputs as shown in fig. Its inputs and output vectors are respectively

$$O = [O_1 \quad O_2 \ldots O_m]^t$$
$$X = [x_1 \quad x_2 \ldots x_n]^t \qquad \ldots(i)$$

Weight w_{ij} connects the ith neuron with the jth output. Therefore the activation value for the ith neuron is

$$net_i = \sum_{j=1}^{n} W_{ij} x_j \qquad \text{for } i = 1, 2, \ldots, n \qquad \ldots(ii)$$

$$\text{Output is } O_i = f(W_i^t x) \qquad \text{for } i = 1, 2, \ldots m \qquad \ldots(iii)$$

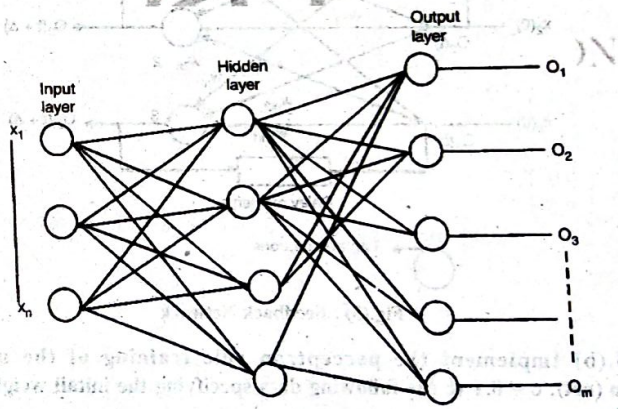


Fig.(2) : Feedforward Network

where weight vector W_i contains the weighting leading towards the ith output node and is defined as

$$W_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^t \qquad \ldots(iv)$$

Introducing the nonlinear matrix operator Γ the mapping of input space x to output space O implemented by

$$O = \Gamma (WX) \quad \ldots(v)$$

where W is called weight matrix.

Since there is no time delay between the input x and the output o, therefore we can write

$$O(t) = \Gamma [Wx(t)]$$

Feedback Network/Recurrent Network : Recurrent neural netwroks (*RNN*) have a closed loop in the network topology. They are developed to deal with the time varying or time-lagged patterns and are usable for the problems where the dynamics of the considered process is complex and the measured data is noisy. Specific groups of the units get the feedback signals from the previous time steps and these units are called context unit. The *RNN* can be either fully or partially connected. In a fully connected *RNN* all the hidden units are connected recurrently, whereas in a partially connected *RNN* the recurrent connections are omitted paratially. Examples of recurrent neural networks are Hopfield networks, Regressive networks, Jordan-Elman networks, and Brain-State-In-a-Box (BSB) networks.

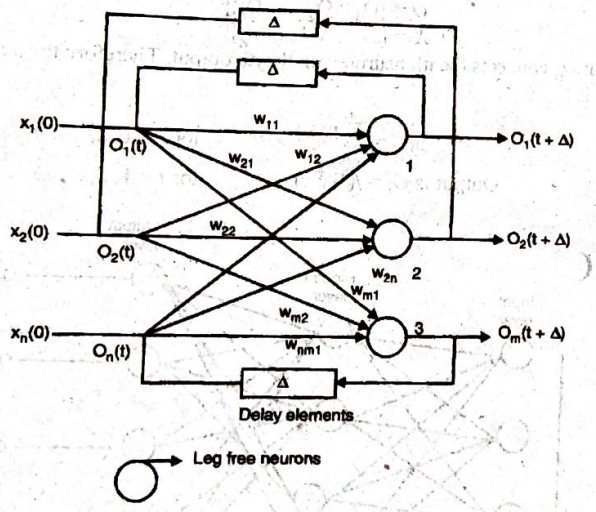


Fig.(3) : Feedback Network

Q.3.(b) Implement the perceptron rule training of the network using f (net) = sgn (net), c = 0.1 & the following data specifying the initail weights W′ and two training pairs.

$$w_0 = \begin{bmatrix} 1.0 \\ -1.0 \\ 0.0 \\ 0.5 \end{bmatrix}, d_0 = -1.0, d_1 = -1, d_2 = 1$$

d_0, d_1, d_2 are teacher's signal.

and

$$x_0 = \begin{bmatrix} 1.0 \\ -2.0 \\ 0.0 \\ -1.0 \end{bmatrix}, x_1 = \begin{bmatrix} 0.0 \\ 1.5 \\ -0.5 \\ -1.0 \end{bmatrix}, x_2 = \begin{bmatrix} -1.0 \\ 1.0 \\ 0.5 \\ -1.0 \end{bmatrix}$$

Ans.

$$z_0 = w_0^T x_0 = \begin{bmatrix} 1.0 & -1.0 & 0.0 & 0.5 \end{bmatrix} \begin{bmatrix} 1.0 \\ -2.0 \\ 0.0 \\ -1.0 \end{bmatrix} = +2.5$$

Note that sgn $(2.5) \neq d_0$. (Note c = 0.1). The weight update is evaluted as :

$$= d - a$$
$$= d_0 - f(w'_0 x_0)$$
$$= -1 - f(2.5)$$
$$= -1 - 1 \text{ [because in perceptron learing rule signum function is used and sgn } (2.5) = +1]$$

$$w_1 = w_0 + 0.1 (-1 - 1) x_0$$

$$w_1 = \begin{bmatrix} 1.0 \\ -1.0 \\ 0.0 \\ 0.5 \end{bmatrix} - 0.2 \begin{bmatrix} 1.0 \\ -2.0 \\ 0.0 \\ -1.0 \end{bmatrix} = \begin{bmatrix} 0.8 \\ -0.6 \\ 0.0 \\ 0.7 \end{bmatrix}$$

$$z_1 = w_1' x_1 = \begin{bmatrix} 0.0 & 1.5 & -0.5 & -1.0 \end{bmatrix} \begin{bmatrix} 0.8 \\ -0.6 \\ 0.0 \\ 0.7 \end{bmatrix} = -1.6$$

Since sgn $(-1.6) = d_1$ no weight update is required. Now

$$w_2 = w_1 + 0.1 (-1 + 1).x_1$$

$$w_2 = w = \begin{bmatrix} 0.8 \\ -0.6 \\ 0 \\ 0.7 \end{bmatrix}$$

$$x_2 = w_2^T x_2 = \begin{bmatrix} 0.8 & -0.6 & 0 & 0.7 \end{bmatrix} \begin{bmatrix} -1.0 \\ 1.0 \\ 0.5 \\ -1.0 \end{bmatrix} = 2.1$$

Note that sgn $(-2.1) \neq d_2$. The weight update is evaluated as :

$$w_3 = w_2 + 0.1 (1 + 1)x_2$$

$$w_3 = \begin{bmatrix} 0.8 \\ -0.6 \\ 0 \\ 0.7 \end{bmatrix} + 0.2 \begin{bmatrix} -1.0 \\ 1.0 \\ 0.5 \\ -1.0 \end{bmatrix} = \begin{bmatrix} 0.6 \\ -0.4 \\ 0.1 \\ 0.5 \end{bmatrix}$$

Section-B

Q.4.(a) What do you mean by discrete perceptron. Explain the use of discrete precceptron in single layer perceptron model. **(10)**

Ans. Discrete Perceptron Concept : the formula for adjusted weights is

$$w' = w \pm Cy$$

where C is constant, and it is greater than O (*i.e.* $C > 0$) and called correction increment.

here +ve sign is for undetected pattern of class 1

and –ve sign applies for undetected pattern of class 2.

If a correct classification takes place under this rule, no adjustment of weight is made, then new weight (w') = old weight (w).

This is one aspect of weight adjustment.

Another aspect is by using the geometrical relationship correction increment C is chosen in such a way that the weight adjustment setp size is meaningfully controlled.

Let p is distance of a point w' from the plane $w'y = 0$ in $(n + 1)$ dimensional. Euclidean space and it is calculated according to formula

$$p = \pm \frac{w^{1t} y}{\| y \|} \quad ...(i)$$

Since p is the distance and by definition of the distance p is always a non-negative scalar. ∴ can be written as

$$p = \frac{w^{1t} y}{\| y \|} \quad ...(ii)$$

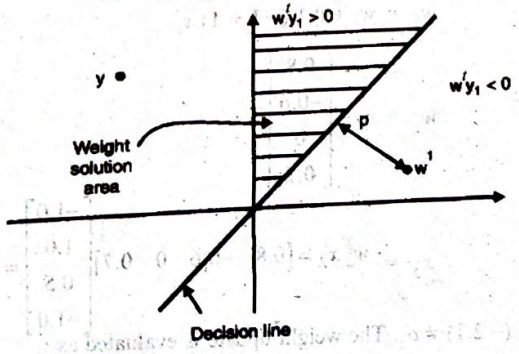


Fig.

Now C must be selected such that corrected weiht vector w^2 based on $w^1 = w \pm cy$ dislocated on the decision hyperplane $w^{1t} y = 0$ which is the decision hyperplane used for particular correction step.

$$\Rightarrow \quad w^{2t} y = 0$$

$$(w^1 \pm cy)^t y = 0$$

$$\therefore \quad (w^1 \pm cy^t) y = 0$$

$$w^{1t} y \pm cy^t y = 0 \quad ...(iii)$$

or $$c y^t y = \mp w^{1t} y$$

$$c = \mp \frac{w^{1t} y}{y^t y} \quad ...(iv)$$

Since c is correction constant and is positive

$$\therefore \quad c = \frac{|w^{1t} y|}{y^t y} \quad ...(v)$$

or length of weight adjustment vector cy can be expressed as

$$\| cy \| = \frac{|w^{1t} y|}{y^t y} \| y \| \quad ...(vi)$$

But $$y^t y = \| y \|^2$$

⇒ required distance p from the point w^1 to decision plane and expressed by (ii) is identical to length of the weight increment vector (vi).

∴ For this the correction increment c is therefore not constant and depends on current tracking pattern as expressed of (v).

Q.4.(b) Explain the delta learning rule for multiperceptron neural network with is equations. **(10)**

Ans. Delta learning rule for multiperceptron neural network (Multi layer perceptron) : A MLP (Multi layer perceptron) is a feedforward artificial neural network which is a modification of the linear perceptron using three of more layers of neurons with more power of distinguishing non-linear data. In MLP each neuron uses a non-linear activation function which is used to develop action potential for firing of neurons. The activation function used must be differentiable and normalizable e.g., tanh ( ) and sigmoids. Learning in MLP is done with the help of a teacher i.e., supervised learning. In this learning of perceptrons connection weights are updated after every process based on the error produced by the difference of desired response and actual response. One of the learning algorithms using Delta Learning rule is described below :

Algorithm : Given are P training pairs

$$\{x_1, d_1, x_2, d_2, \ldots x_p, d_p\}$$

where x_i is $(n * 1)$, d_i is $(R * 1)$

$$Y_i = \text{Augmented Input Vector} = \begin{bmatrix} x_i \\ 1 \end{bmatrix}$$

$i = 1, 2, \ldots, P$

No. of Classes/Categories = R

In the following steps, k denotes the training step and p denotes the step counter within the training cycle, c is the learning constant and E is error.

Step 1 : $c > 0$, E_{min} are chosen.

Step 2 : Initialize weights at very small values, w is $(n + 1) * 1$. Initialize $k = 1$, $p = 1$, $E = 0$.

Step 3 : The training cycle begins here. Input is given and output is calculated as

$$y = y_p, \quad d = d_p$$

$$O_i = f(w^t\ y) \quad \text{for} \quad i = 1, 2, ..., R$$

[f is the activation function]

Step 4 : Weights are updated as

$$w_i = w_i + \frac{1}{2} c(d_i - o_i)Y \quad \text{for } i = 1, 2, ..., R$$

Step 5 : Cycle error is computed as

$$E = 1/2\ (d_i - o_i)^2 + E \quad \text{for } i = 1, 2, ..., R$$

Step 6 : If $p < P$, then $P = p + 1$ and $k = k + 1$ and go to step 3, otherwise go to step 7.

Step 7 : The training cycle is completed. For $E = 0$, terminate the training session. Output weights and k are checked, otherwise if $E > 0$, then $E = 0$, $p = 1$ and enter the new training cycle by going to step 3.

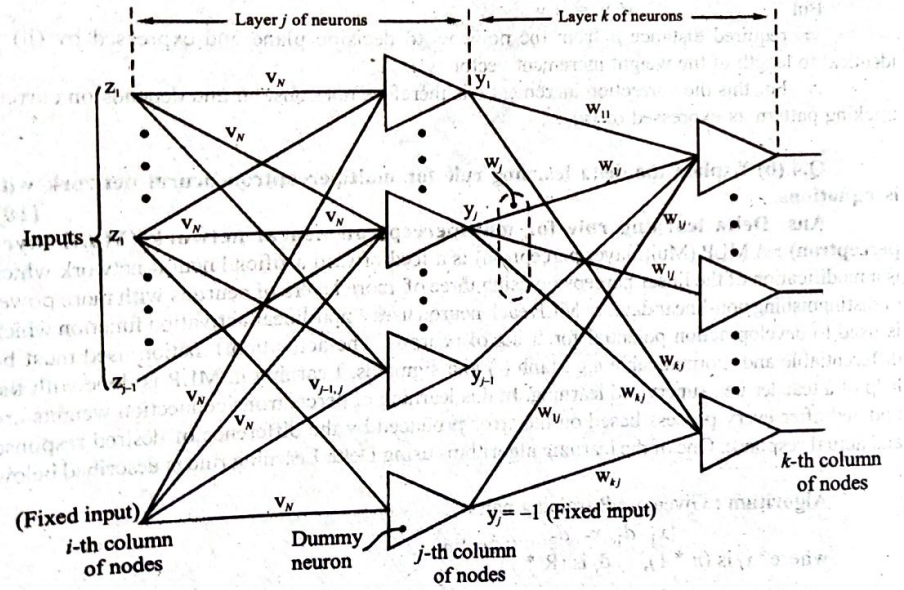


Fig. : Multilayer Feedforward Network

This algorithm is known as R-category Continuous Perceptron Training Algorithm because in this the network uses the linear classifier for categorizing a number of classes (more than 2).

In this algorithm it is assumed that all the classes/categories are linearly separable and there is a different function for each class. Both Single Layer continuous peceptron training algorithm and multilayer continuous perceptron training algorithm are ERROR CORRECTING ALGORITHMS.

Derivation of equation : Consider fig of multilayer feedforward network, the negative gradient decent formula for the hidden layer is

$$\Delta v_{ji} = -\eta \frac{\partial E}{\partial v_{ji}}, \quad \text{for } j = 1, 2, ..., J \text{ and } i = 1, 2, ..., I \qquad ...(i)$$

and

$$\frac{\partial E}{\partial v_{ji}} = \frac{\partial E}{\partial (net_j)} \cdot \frac{\partial (net_j)}{\partial v_{ji}} \qquad ...(ii)$$

Let us notice that the inputs to the layer are z_i, for $i = 1, 2, ..., I$. Based on relation

$$\text{Net} = \text{weight} * \text{input}$$

$$net_j = v_{ji} * Z_i$$

The second term in the product (ii) is equal to z_i, and the weight adjustment can be expressed as

$$\Delta v_{ji} = \eta \delta_{yj} z_i \qquad ...(iii)$$

where δ_{yj} is the error signal term of the hidden layer having output y. This error signal term is produced by the j^{th} neuron of the hidden layer, where $j = 1, 2, ..., J$. The error signal term is equal to

$$\delta_{yj} = -\frac{\partial E}{\partial (net_j)}, \quad \text{for } j = 1, 2, ..., J \qquad ...(iv)$$

In contrast to the output layer neurons' excitation net_k, which affected the k^{th} neuron output only, the net, contributes now to every error component in the error sum containing K terms. The error signal terms at the node j can be computed as

$$\delta_{yj} = -\frac{\partial E}{\partial_{yj}} \cdot \frac{\partial_{yj}}{\partial (net_j)} \qquad ...(v)$$

where

$$\frac{\partial E}{\partial_{yj}} = \frac{\partial}{\partial y_j} \left(\frac{1}{2} \sum_{k=1}^{k} \{d_k - f[net_k(y)]\}^2 \right) \qquad ...(vi)$$

The 2nd term of above equation (v) is

$$\frac{\partial y_j}{\partial (net_j)} = f_j'(net_j) \qquad ...(vii)$$

Equation (vi) will become (vii) after taking the derivative

$$\frac{\partial E}{\partial y_j} = -\sum_{k=1}^{k} (d_k - o_k) \frac{\partial}{\partial y_j} \{f[net_k(y)]\} \qquad ...(viii)$$

Calculation of the derivative in braces of expression (viii) becomes

$$\frac{\partial E}{\partial y_j} = -\sum_{k=1}^{K} (d_k - o_k) f'(net_k) \frac{\partial (net_k)}{\partial y_j} \qquad ...(ix)$$

Now after simplification of above equation, we get

$$\frac{\partial E}{\partial y_j} = -\sum_{k=1}^{K} \delta_{ok} w_{kj} \qquad ...(x)$$

Combining the result from (x) and (vii)

$$\delta_{jt} = f_j'(net_j)\sum_{k=1}^{K}\delta_{ok}w_{kj} \quad \text{for } j = 1, 2, \ldots, J \qquad \ldots(xi)$$

The weight adjustment in the hidden layer now becomes

$$\Delta v_{ji} = \eta f_j'(net_j)z_i\sum_{k=1}^{K}\delta_{ok}w_{kj}, \text{ for } j = 1, 2, \ldots, J \text{ and } i = 1, 2, \ldots, I. \ldots(xii)$$

The equation (xii) is called generalized Delta Rule.

Q.5.(a) Explain the linearly separable classification with suitable example. (10)

Ans. Linears Separability : When a linear hyperplane exists to place the instances of one class on one side and those of other class on the other side of plane. Then this is called *Linear Separability*. For example in case of OR gate we can draw a linear hyperplane to separate the inputs whose corresponding outputs are 1 on one side and the inputs whose corresponding outputs are 0 an other sidee of the plane.

Table : OR Gate

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	1

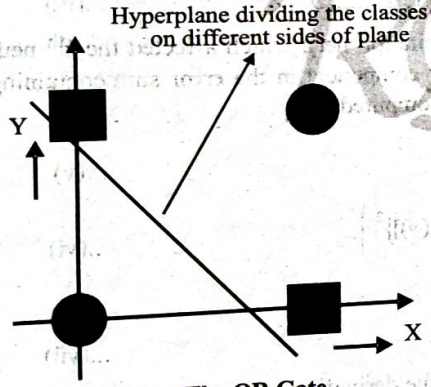


Fig. : The OR Gate

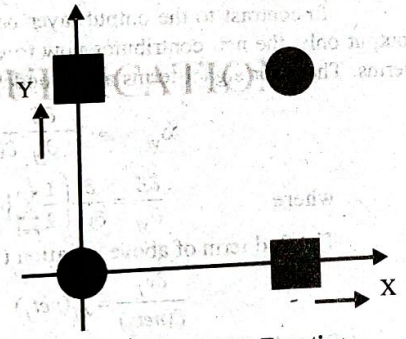


Fig. : X-OR Function

But all the classification problems are not linear separable. For example the EX-OR problem is not linear separable because EX-OR is non equality gate i.e.

Table: The EX-OR Gate

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	0

In this case we can't draw any hyperplane that divides the space so that different classes are on different sides. To cope up with a problem which is not linear separable, a Multilayer Perceptron is required.

Q.5.(b) Explain error back program training algorithm. (10)

Ans. The layer network is mapping the input vector z into the output vector o as follows :

$$o = N(z)$$

where N denotes a composite nonlinear matrix operator. For the two-layer net the mapping $z \rightarrow o$ can be represented as a mapping within a mapping, or

$$o = G[W\Gamma[Vz]]$$
$$\Gamma[Vz] = y$$

and it related to the hidden layer mapping $z \rightarrow y$. Note that the right arrows denote mapping of one space into another. Each of the mapping is performed by a single-layer of the layered network. The operator Γ is a nonlinear diagonal operator.

Algorithm :

Given are P training pairs

$$\{z_1, d_1, z_2, d_2, \ldots\ldots, z_p, d_p,\},$$

Where z_i is $(I \times 1)$, d_i is $(K \times 1)$, and $i = 1, 2, \ldots P$. Note that the Ith component of each x_i is of value – 1 since input vectors have been augmented. Size $J-1$ of the hidden layer having outputs y is selected. Note that the Jth component of y is of value – 1, since hidden layer outputs have also been augmented; y is $(J \times 1)$ and o is $(K \times 1)$.

Step 1 : $\eta > 0$. E_{max} chosen

Weights W and Y are initialized at small random values; W is $(K \times J)$, V is $(J \times I)$.

$$q \leftarrow 1, \quad p \leftarrow 1, E \leftarrow 0$$

Step 2 : Training step starts here.

Input is presented and the layers output computed.

$$z \leftarrow z_p, d \leftarrow d_p$$
$$y_j \leftarrow f(v_j^t z), \text{ for } j = 1, 2, \ldots\ldots, J$$

where v_j, a column vector is the Jth row of V and

$$o_k \leftarrow f(w_k^t y), \text{ for } k = 1, 2, \ldots\ldots, K$$

where w_k, a column vector, is the kth row of W.

Step 3 : Error value is computed

$$E \leftarrow \frac{1}{2}(d_k - o_k)^2 + E, \text{ for } k = 1, 2, \ldots\ldots, K$$

Step 4 : Error signal vectors δ_0 and δ_y of the both layers are computed.

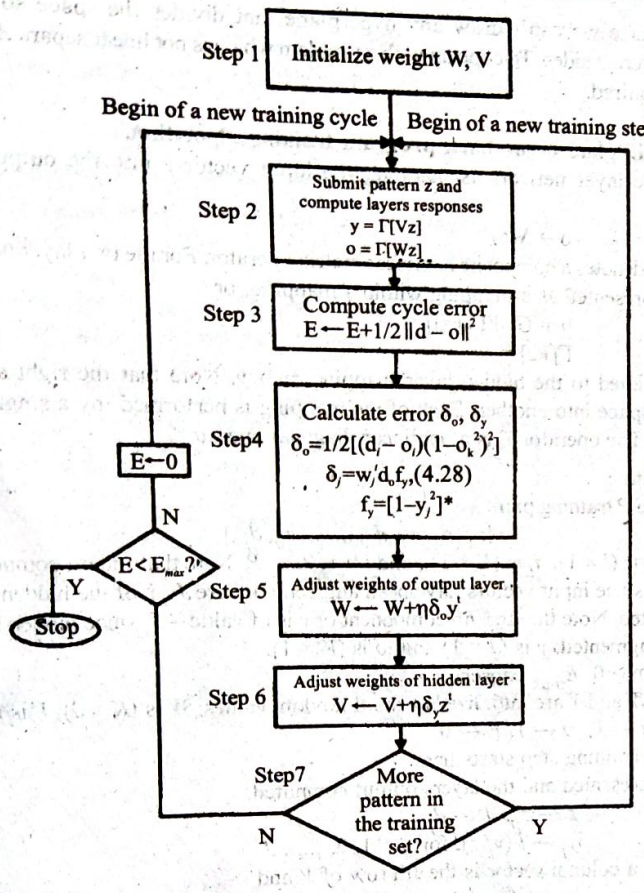
Vector δ_0 is $(K \times 1)$, δ_y is $(J \times 1)$.

The error signal terms of the output layer in this step are

$$\delta_{ok} = \frac{1}{2}(d_k - o_k)(1 - o_k^2), \text{for } k = 1, 2, \ldots\ldots, K$$

The error signal terms of the hidden layer in this step are

$$\delta_{yi} = \frac{1}{2}(1 - y_j^2)\sum_{k=1}^{k}\delta_{ok}w_{kj}, \text{for } j = 1, 2, \ldots\ldots, J$$



Step 5 : Output layer weights are adjusted :

$$w_{kj} \leftarrow w_{kj} + \eta\delta_{ok}y_j, \text{ for } k = 1, 2, \ldots K \text{ and } j = 1, 2, \ldots, J$$

Step 6 : Hidden layer weights are adjusted

$$v_{ji} \leftarrow v_{ji} + \eta\delta_{yj}z_i, \text{ for } j = 1, 2, \ldots J \text{ and } i = 1, 2, \ldots, I$$

Step 7 : If $p < P$ then $p \leftarrow p + 1$, $q \leftarrow q + 1$, and go to Step 2; otherwise, go to Step 8.

Step 8 : The training cycle is completed.

For $E < E_{max}$ terminate the training session. Output weights W, V, q and E.

If $E > E_{max}$ then $E \leftarrow 0$, $p \leftarrow 1$, and initiate the new training cycle by going to Step 2.

Section-C

Q.6. Explain various architechures of Hopfield network in detail. How learning process occurs in Hopfield network ? **(20)**

Ans. Hopfield Network : The Hopfield model was proposed by John Hopfield of the California Institute of Technology during the early 1980s. The dynamics of the Hopfield model is different from that of the linear associator model in that it computes its output recursively in time until the system becomes stable. Below is a Hopfield model with six units, where each node is connected to every other node in the network.

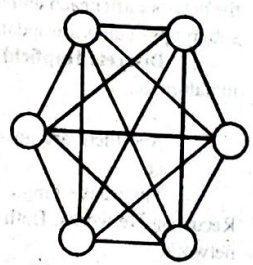


Fig. : Hopfield Model

Unlike the linear associator model which consists of two layers of processing units, one serving as the inputs layer while the other as the output layer, the Hopfield model consists of a single layer of processing elements where each unit is connected to every other unit in the network other than itself.

The connection weight matrix W of this type of network is square and symmetric, i.e., $w_{ij} = w_{ji}$ for $i, j = 1, 2 \ldots, m$. Each unit has an extra external input I_i. This extra input leads to a modification in the computation of the net input to the units :

$$\text{input}_j = \alpha \sum_{i=1}^{m} x_i w_{ij} + I_j$$

where $j = 1, 2, \ldots, m$.

Unlike the linear associator, the units in the Hopfield model act as both input and output units. But just like the linear associator, a single associated pattern pair is stored by computing the weight matrix as follows :

$$W_k = X_k$$
$$TY_k$$

where $Y_k = X_k$ to store p different associated pattern pairs. Since the Hopfield model is an autoassociative memory model, patterns, rather than associated pattern pairs, are stored in memory.

After encoding, the network can be used for decoding. Decoding in the Hopfield model is achieved by a collective and recursive relaxation search for a stored pattern given an initial stimulus pattern. Given an input pattern X, decoding is accomplished by computing the net input to the units and determining the output of those units using the output function to produce the pattern X'. The pattern X' is then fed back to the units as an input pattern to produce the pattern X''. The pattern X'' is again fedback to the units to produce the pattern X'''. The process is repeated until the network stabilizes on a stored pattern where further computations do not change the output of the units.

If the input pattern X is an incomplete pattern or if it contain some distortions, the stored pattern to which the network stabilizes is typically one that is most similar to X without the distortions. This feature is called *pattern completion* and is very useful in many image processing applications.

During decoding, there are several schemes that can be useed to update the output of the units. The updating schemes are *synchronous* (or parallel as termed in some literatures), *asynchronous* (or sequential), or a combination of the two (hybrid). Using the synchronous updating scheme, the output of the units are updated as a group prior to feeding the output back to the network. On the other hand, using the asynchronous updating scheme, the output of the

units are updated in some order (e.g. random or sequential) and the output are then fed back to the network after each unit update. Using the hybrid synchronous-asynchronous undating scheme, subgroups of units are updated synchronously while units in each subgroup updated asynchronously.

Discrete Hopfield Networks : The Hopfield network is fully connected and symmetric in nature i.e.,

$$w_{ij} = w_{ji}$$

And there are no self connections

i.e., $$w_{ii} = 0$$

They have single layer feedback network architecture i.e., why they are called as Recurrent Networks. Both binary (0, 1) and bipolar (–1, 1) values can be given as input to the network.

In this network, only one unit updates at a time. Here input pattern is applied to the network and network's output is computed. Now initial input pattern is removed and the output pattern is forced as input to the input layer through feedback interconnections and produces second updated output. This process continues until no new, updated responses are produced and network reaches an equilibrium/attractor state.

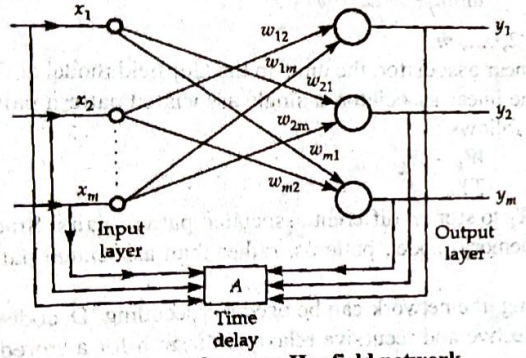


Fig. ; Asynchronous Hopfield network

Training Algorithm of Discrete Hopfield Net :

(a) For storing a set of binary input patterns $s(p), p = 1$ to P,

where $s(p) = (s_1(p), \ldots s_i(p), \ldots s_n(p)$,

The weight matrix is given by

$$w_{ij} = \sum_{p=1}^{P} [2s_i(p) - 1][2s_j(p) - 1] \qquad \text{for } i \neq j$$

(b) For storing a set of bipolar input patterns

$$s(p), p = 1 \text{ to } P,$$

where $s(p) = (s_1(p), \ldots s_i(p), \ldots s_n(p)$.

The weight matrix is given by

$$w_{ij} = \sum_{p=1}^{P} s_i(p) s_j(p) \qquad \text{for } i \neq j$$

and $w_{ij} = 0$ as weights have no self connections.

Q.7.(a) Design a bi-directional associative memory to encode the following pattern.

A_1 = 100001 B_1 = 11000
A_2 = 011001 B_2 = 10100
A_3 = 001011 B_3 = 01110
& check it for A_3

Ans. Consider $N = 3$ pattern pairs $(A_1, B_1), (A_2, B_2), (A_3, B_3)$ given by **(10)**

$A_1 = (1\ 0\ 0\ 0\ 0\ 1)$ $\quad B_1 = (1\ 1\ 0\ 0\ 0)$
$A_2 = (0\ 1\ 1\ 0\ 0\ 0)$ $\quad B_2 = (1\ 0\ 1\ 0\ 0)$
$A_3 = (0\ 0\ 1\ 0\ 1\ 1)$ $\quad B_3 = (0\ 1\ 1\ 1\ 0)$

Convert these three binary pattern to bipolar form replacing 0s by –1s.

$X_1 = (1\ -1\ -1\ -1\ -1\ \ 1)$ $\quad Y_1 = (1\ \ 1\ -1\ -1\ -1)$
$X_2 = (-1\ \ 1\ \ 1\ -1\ -1\ \ 1)$ $\quad Y_2 = (1\ -1\ \ 1\ -1\ -1)$
$X_3 = (-1\ -1\ \ 1\ -1\ \ 1\ \ 1)$ $\quad Y_3 = (-1\ \ 1\ \ 1\ \ 1\ -1)$

The correlation matrix M is calculated as 6×5 matrix

$$M = X_1^T Y_1 + X_2^T Y_2 + X_3^T Y_3 = \begin{bmatrix} 1 & 1 & -3 & -1 & 1 \\ 1 & -3 & 1 & -1 & 1 \\ -1 & -1 & 3 & 1 & -1 \\ -1 & -1 & -1 & 1 & 3 \\ -3 & 1 & 1 & 3 & 1 \\ -1 & 3 & -1 & 1 & -1 \end{bmatrix}$$

Suppose we start with $\alpha = X_3$, and we hope to retrieve the associated pair Y_3. The calculations for the retrieval of Y_3 yield :

$\alpha M = [-1\ -1\ \ 1\ -1\ \ 1\ \ 1]$ $(M) = (-6\ \ 6\ \ 6\ \ 6\ -6)$
$\Phi(\alpha M) = \beta' = (-1\ \ 1\ \ 1\ \ 1\ -1)$
$\beta' M^T = (-5\ \ -5\ \ \ 5\ \ -3\ \ \ 7\ \ \ 5)$
$\Phi(\beta' M^T) = (-1\ \ -1\ \ \ 1\ \ -1\ \ \ 1\ \ \ 1) = \alpha'$
$\alpha' M = (-1\ \ -1\ \ \ 1\ -1\ \ \ 1\ \ \ 1)$ $M = (-6\ \ \ 6\ \ \ 6\ \ \ 6\ \ -6)$
$\Phi(\alpha' M) = \beta'' = (-1\ \ \ 1\ \ \ 1\ \ \ 1\ \ \ 1\ \ -1)$

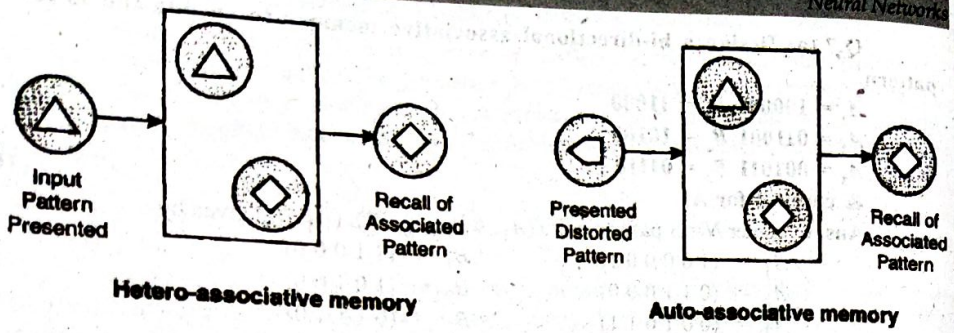
This retrieved pattern β' is same as Y_3.

Hence, $(\alpha_f, \beta_f) = (X_3, Y_3)$ is correctly recalled, a desired result.

Q.7.(b) Write two typess of associative memories. **(10)**

Ans. Auto Associative Memory : A n auto-associative memory, also know as auto-associative correlator, is used to retrieve a previously stored pattern that most closely resembles the current pattern.

Hetero Associative Memory : A hetero-associative memory, also known as hetero-associative correlator, is used to rerieve pattern in general, different from the input pattern not only in content but possibly also different in type and format.



Section-D

Q.8.(a) Describe unsupervised learing of clusters with suitable example. (10)

Ans. Unsupervised learning of clusters : Learning is based on clustering of input data. No prior knowledge is assumed to be available regarding an input membership in a particular class. Rather, gradually detected characteristics and a history of training will be used to assist the network in defining classes and possible boundaries between them.

Clustering : Clustering is understood to be the grouping of similar objects and separating of dissimilar ones. The objective of clustering neural networks is to categorize or cluster data. The classes must first be found from the correlations of an input data stream. Since the network actually deals with unlabeled data, the clustering should be followed by labeling clusters with appropriate category names or numbers. This process of providing the category of objects with a label is usually termed as calibration.

Although the algorithm won't have names to assign to these clusters, it can produce them and then use those clusters to assign new examples into one or the other of the clusters. This is a data-driven approach that can work well when there is sufficient data; for instance, social information filtering algorithms, such as those that Amazon.com use to recommend books, are based on the principle of finding similar groups of people and then assigning new users to groups. In some cases, such as with social information filtering, the information about other members of a cluster (such as what books they read) can be sufficient for the algorithm to produce meaningful results. In other cases, it may be the case that the clusters are merely a useful tool for a human analyst. Unfortunately, even unsupervised learning suffers from the problem of overfitting the training data. There's no silver bullet to avoiding the problem because any algorithm that can learn from its inputs needs to be quite powerful.

Suppose we are given a set of patterns without any information as to the number of classes that may be present in the set. The clustering problem in such a case is that of identifying the number of classes according to a certain criterion, and of assigning the membership of the patterns in these classes. The clustering technique presented below :

Assumptions : The number of classes is known a priori. The pattern set $\{x_1, x_2, \ldots\ldots, x_n\}$ is submitted to the input to determine decision functions required to identify possible clusters. Since no information is available from the teacher as far as the desired classifier's responses, we will use the similarity of incoming patterns as the criterion for clustering. To define a cluster, we need to establish a basis for assigning patterns to the domain of particular cluster.

How we will assign a pattern to domain of particular cluster? There are 2 rules for this.

(1) The most common similarity rule is to find the Euclidean distance between 2 patterns x for x_i defined as

$$|X - X_i| = [(X - X_i)^t (X - X_i)]^{1/2} \quad \ldots(i)$$

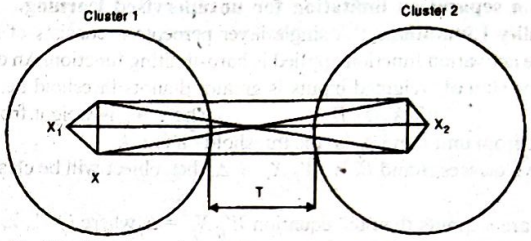


Fig.(a) : Measure of similarity between 2 clusters using distance

Smaller the distance, closer the patterns. Using (i) the distance between all the pairs are computed. Now how to distinguish the clusters? A distance T can then be chosen to discriminate clusters. The value T is understood as the maximum distance between patterns within a single cluster. Fig.(a) shows an example of two clusters with a T value chosen to be greater than the typical within-cluster distance but smaller than the between-cluster distance.

Let there are 2 patterns X_1 and X_2 and we have to find which pattern is similar to pattern X. Then using (1) distance is calculated. The value for which the distance is small, closer the pattern is with x.

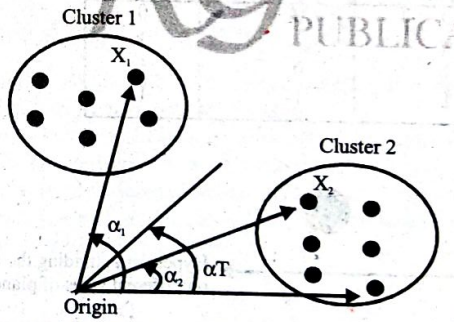


Fig.(b) : Measure of similarity between 2 clusters using normalized scalar product

(2) Another similarity rule is the cosine of the angle between x and x_i:

$$\cos \alpha = \frac{(X^t X_i)}{\| X \| . \| X_i \|} \quad \ldots(ii)$$

This rule is particularly useful when clusters develop along certain principal and different axes as shown in fig.(b). For $\cos \alpha_2 < \cos \alpha_1$ pattern x is more similar to x_2 than to x_1. It would thus be natural to group it with the second of the two apparent clusters. To facilitate this decision, the threshold angle α_t can be chosen to define the minimum angular cluster distance. It should be

noted, however, that the measure defined in equation (ii) should be used according to certain additional qualifications. If the angular similarity criterion equation (ii) is to be efficient, vectors x_1, x_2 and x should be of comparable, or better, identical lengths.

Q.8.(b) Explain separation limitation for unsupervised learning. **(10)**

Ans. Separability Limitations : A single layer perceptron consists of an input layer and an output layer. The activation function applied is hard-limiting function. An output unit will assume the value 1 if the sum of weighted inputs is greater than its threshold i.e.

$$W_{ji} X_i > W_j \qquad \text{where } W_{ji} \text{ is weight from } i \text{ to unit } j.$$

X_i is the input from unit i and Δ_j is the threshold on unit j.

Let there are two classes A and B. If $W_{ji} X_i > \Delta_j$ then object will be classified as Class A otherwise Class B.

Suppose there are n inputs then the equation $W_{ji} X_i = \Delta_j$ where $i = 1, 2, \ldots n$ form a hyperplane, dividing the space in two halves.

Linears Separability : When a linear hyperplane exists to place the instances of one class on one side and those of other class on the other side of plane. Then this is called *Linear Separability*. For example in case of OR gate we can draw a linear hyperplane to separate the inputs whose corresponding outputs are 1 on one side and the inputs whose corresponding outputs are 0 an other sidee of the plane.

Table : OR Gate

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	1

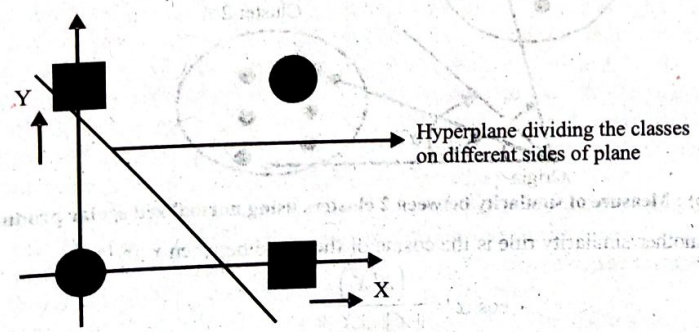


Fig. : The OR Gate

But all the classification problems are not linear separable. For example the EX-OR problem is not linear separable because EX-OR is non equality gate i.e.

Table: The EX-OR Gate

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	0

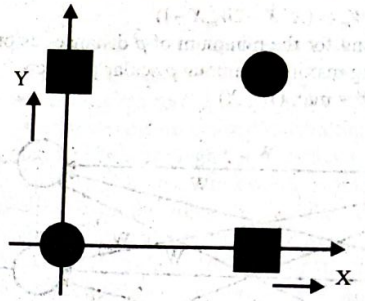


Fig. : X-OR Function

In this case we can't draw any hyperplane that divides the space so that different classes are on different sides. To cope up with a problem which is not linear separable, a Multilayer Perceptron is required.

Multilayer perceptron is a feedforward neural network with at least one hidden layer. It can deal with nonlinear classification problems.

Q.9. Describe Winner take all learning and the architecture of Kohonen networks. Illustrate it with the help of example. **(20)**

Ans. Winner-Take-All-Learning (Kohonen Network) :

(i) It consists of single layer of nodes pluse an input layer.

(ii) Each layer receives inputs from environment and from other nodes within the layer.

(iii) Important point about this network is that both weight vector and input vectors are normalized to a constant value before learning

$$\hat{w}_i = \frac{(W_i)}{|W_i|} \qquad \text{for } i = 1, 2, \ldots P$$

(iv) Each node computes by taking the dot product of its weight vector and input vector. $O_j = XW$, where O_j is output or activation level of unit j.

(v) Node with the largest activation level is declared the winner in the competition. This *wining* node is the only node that will generate an output signal, the activation level of other neurons is suppressed to zero. The Winning node with largest net_i is allowed to change the weight that is weight adjustment is for winning neuron only while weights of other neurons remain unaffected. The weight adjustment criteria for this mode of training is such that

$$|X - W_m| = \min\{|X - W_i|\} \qquad \ldots(i)$$

where $i = 1, 2, ..., P$ and m is winning neuron no.

min represents the minimum. As explained earlier also in clustering that smaller the distance more close the patterns are and finding the minimum of P distances corresponds to finding maximum among the p scalar products.

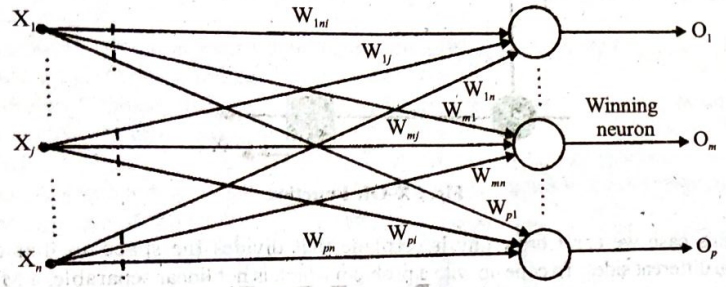
$$W_m^t X = \max (W_i^t) X \qquad \text{where } i = 1, 2, ..., P$$

The left hand side of above equation can be written as

$$| X - W_m | = (X^t X - 2W_m^t X - 1)^{1/2} \qquad ...(ii)$$

It is clear that searching for the minimum of p distances as on right hand side of (i) Corresponds to finding maximum among p scalar products

$$W_m{}^t X = \max (W_i^t X) \qquad ....(iii)$$



For $i = 1, 2, ... p$

After the winning neuron has been identified and declared a winner, its weights must be adjusted so that the distance is reduced in the current training step. Thus $|X - W_m|$ must be reduced, perferably along the gradient direction in the weight space w_{m1}, w_{m2}, ... w_{mn}

$$\nabla w_m \| x - w_m \|^2 = -2(x - w_m) \qquad ...(iv)$$

It seems reasonable to reward the weights of the $X - W_m$ winning neuron with an increment of weight in the negative gradient direction, thus in the direction $x - w_m$. We thus have

$$\nabla \hat{w}_m = \alpha (x - \Delta \hat{w}_m) \qquad ...(v)$$

where α is a small learning constant selected heuristically, usually between 0.1 and 0.7.

Learning according to equation (iv) and (v) is called Winner-take-All learning and it is a common competitive and unsupervised learning technique. The winning neuron with the largest net is rewarded with a weight adjustment, while the weights of others remain unaffected.

Another modification of the winner-take-all learning rule is that both the winner's and loser's weights are adjusted in proportion to their level of responses. This may be called leaky competitive learning and should provide more subtle learning in the cases for which clusters may be hard to distinguish.

NEURAL NETWORKS

Dec 2015

Paper Code: CSE-407-F

Note: ***Attempt five questions in all, selecting one question from each section. Q. No. 1 is compulsory.***

Q.1.(a) Differentiate supervised and unsupervised learning. **(5)**

Ans. Supervised Learning : In supervised learning, we assume that at each instant of time when the input is applied, the desired response d of the system is provided by the teacher. The network then processes the inputs and compares its resulting outputs against the desired output d. Errors are then propagated back through the system, causing the system to adjust the weights, which control the network. This process occurs over and over as the weights are continually decreases. Since we assume the adjustable weights, the teacher may implement a reward – and -punishment scheme to adapt the network's weight matrix W. The set of data, which enables the training, is called the "training set". During the training of a network the same set of data is processed many times as the connection weights are ever refined.

Unsupervised Learning : The other type of learning is called unsupervised learning. In unsupervised learning, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred toas self-organization or adaptation.

Q.1.(b) Explain learing factors. **(5)**

Ans. Learning factors :

1. *Initialization of Weights* : The weights of the network to be trained are initialized at small random values. The initialization affects the complete solution. If all the weights start out with equal weight values and if the soultion requries that unequal weights be developed, the network may not be trained properly. artificial neural networks typically start out with randomized weights for all their neurons.

If training continues beyond a certain low error level, it will result in undersirable drift of weights. This casues error to increase and the quality of mapping by the network decreases. Therefore the choice of initial weights is however only one of several factors affecting training of error towards an acceptable error minimization.

2. *Learning Constant* : Careful selection of step size (learning constant η) is often necessary to ensure the smooth convergence. However optimum value of η depends on problem being solved and there is no single learning constant value suitable for different training cases.

$$\eta = \frac{1.5}{N_1^2 + N_2^2 + N_n^2}$$

Where N_1 is the number of patterns of type 1 and there are total n different patterns.

The learning coefficient cant be –ve because this would cause the change of weight vector to move from ideal weight vector position. If the value of η is zero then, no learning takes place and hence the value of η must be +ve.

3. *Momentum Method* : The purpose of the momentum method is to accelerate the convergence of the error back-propagation learning algorithm. The method involves supplementing the current weight adjustments with a fraction of the most recent weight adjustment. This is usually done according to the formula

$$\Sigma W(t) = (-)\delta \Delta E/\Delta t + \alpha \Sigma W(t-1)$$

Where the arguments t and t – 1 are used to indicate the current and the most recent training step, respectively, and it is a user-selected positive momentum constant. The second term, indicating a scaled most recent adjustment of weights, is called the *momentum term*. If momentum is zero, then smoothening is minimum and the entire weight adjustment comes from newly calculated change if momentum is 1, new weight adjustment is ignored and the previous one is repeated. Between 0 and 1 is a region where the weight adjustment is smoothened by an amount proportional to the momentum factor. Typically, α is chosen between 0.1 and 0.8 to increase the speed of learning without leading to oscillations.

4. *Sigmoidal gain* : If sigmoidal function is selected, the input-output relationship of the neuron can be set as $O = 1/[1 + \exp(-\lambda(I + O))]$ where λ is a scaling factor known as sigmoidal gain.

In some problems, when the weights become large and force the neuron to operate in a region where the sigmoidal function is very flat, a better method of coping with the network paralysis is to adjust the sigmoidal gain.

By decreasing the scvaling factor, this activation function can be spread out on wide range so that training proceeds faster.

5. *Threshold value* : O in above equation is called as threshold value of a neuron, or bias or the noise factor. A neuron generates the output if the weighted sum of the input exceeds the threshold value.

Q.1.(c) What are separability limitations ? **(5)**

Ans. A single layer perceptron consists of an input layer and an output layer. The activation function applied is hard-limiting function. An output unit will assume the value 1 if the sum of weighted input is greater than its threshold. *i.e.*,

$W_{ji}X_i > W_j$ where W_{ji} is weight from unit i to unit j.

X_i is the input from unit i and Δ_j is the threshold on unit j

Let there are 2 classes A and B. If $W_{ji}X_i > \Delta_j$ where $i = 1, 2, \ldots n$

forms a hyperplane, dividing the space in 2 halves.

Linear separability : When a linear hyperplane exists to place the instances of one class on one side and those of other class on the other side of plane. Then this is called Linear Separability. For example in case of OR gate we can draw a linear hyperplane to separate the input whose corresponding outputs are 1 on one side and the inputs whose corresponding outputs are 0 an other side of the plane.

Table : OR Gate

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	1

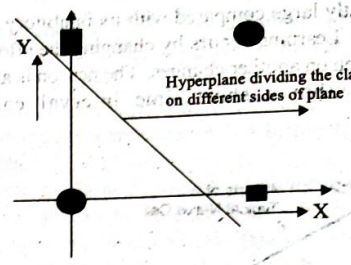


Fig. : The OR gate

But all the classification problems are not linear separable. Fox example the EX-OR problem is not linear separable because EX-OR is non equality gate.

Q.1.(d) Compare perceptron, delta, and winner take all learning rules. **(5)**

Ans. Comparision between perceptron, delta, and winner take all learning rules :

Learning rule	*Single weight adjustment* Δw_{ij}	*Initial weights*	*Learnings*	*Neuron Characteristics*	*Neuron Layer*
Perception	$c[d_i - \text{sgn}(w_i^t(x)]\ x_j$ $j = 1, 2, \ldots, n.$	Any	*S*	Binary bipolar, or Binary uni-polar	Neuron
Delta	$c(d_i - o_i)f'(\text{net}_i)x_j$ $j = 1, 2, \ldots, n$	Any	*S*	Continuous	Neuron
Winner-take-all	$\Delta w_{mj} = \alpha(x_j - w_{mj})$ m-winning neuron number. $j = 1, 2, \ldots, n$	Random Normalized	*U*	Conditions	Layer of p neurons.

Section – A

Q.2. Explain in detail the structure of biological neurons relevant to ANN. **(20)**

Ans. The elementary nerve cell called a neuron is the fundamental building block of biological neural network. The three main components of a biological neuron are :

1. A neuron cell body called *soma*.
2. Branching extensions called *dendrites* for receiving input, and
3. An *axon* that carries the neuron's output to the dendrites of other neurons.

In the human brain, a typical neuron collects signals from others through a host of fine structures called *dendrites*. The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. The axon-dendrite contact organ is called a synapse. At the end of each branch, a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives

excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes. The neuron is able to respond to the total of its inputs aggregated within a short time interval called period of latent summation.

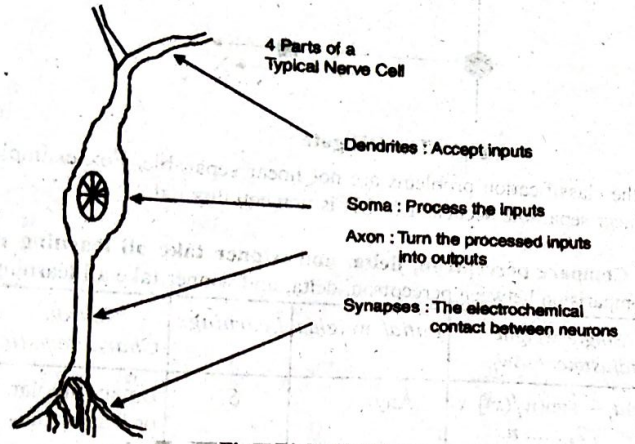


Fig. : Biological Neuron.

ANN is defined as an interconnection of neurons such that the neuron outputs are connected, through the weights to all other neurons including themselves; both lag-free and delay connections are allowed.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

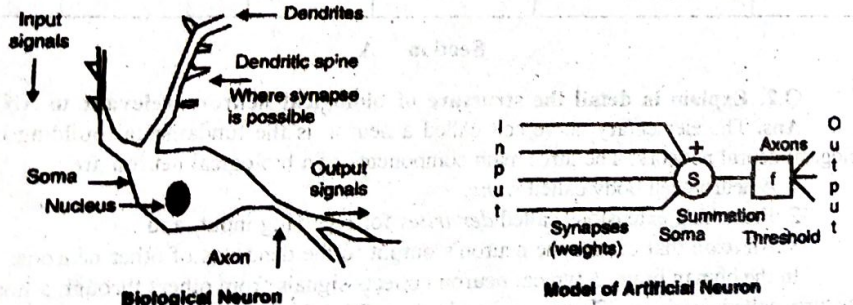


Fig. : Similarities between Biological Neuron and ANN

Q.3. Explain McCulloch-Pitts neuron model and use McCulloch-Pitts neuron to design logic networks of AND and XOR logic function. **(20)**

Ans. McCulloch-Pitts (MCP) Neuron Model : The early model of an artificial neuron is introduced by Warren McCulloch and Walter Pitts in 1943. The MCP neuron

(i) Has a number of inputs I_1, ... I_n. These inputs represent the incoming signals received from the neuron's synapses. So I_i can either be 1, which corresponds to the presence of an incoming signal from the ith connection, or I_i is 0, which corresponds to the absence of a signal from the ith connection.

(ii) The *MCP* neuron produces an output y, which can either be 1, corresponding to the neuron sending a signal, or it can be 0, corresponding to the neuron remaining at rest. Mathematically an *MCP* neuron is a function which takes an n-tuple of 1's and 0's and produces a 1 or a 0. In order to quantity the influence each synapse has on the *MCP* neuron, we assign a weight w_i to each input I_i.

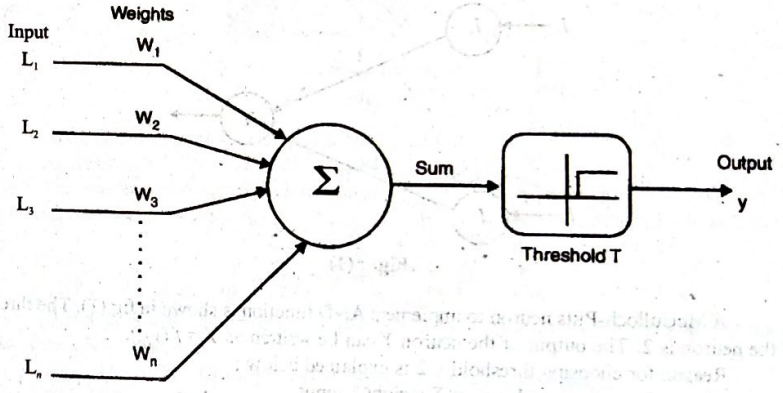


Fig.(a) : Symbolic Illustration of linear threshold gate

(iii) The weights are decimal numbers, and the size of the weight corresponds to the amount of influence the associated connection has on the *MCP* neuron. Positive weights correspond to excitatory synapses, and negative weights correspond to inhibitory synapses. Each input is multiplied by the corresponding weight, and these weighted inputs are then added together to produce

$$\text{Sum} = \sum_{i=1}^{N} I_i W_i \quad \text{...(i)}$$

$$y = f(\text{sum}) \quad \text{...(ii)}$$

where $w_i * I_i$ means w_i multiplied by I_i. This corresponds to the summing of the incoming signals that is assumed to be performed by a neuron. Finally, there is a threshold T, which is a decimal number that is compared to the weighted sum of the signals. This corresponds to the apparent threshold value that the sum of the received signals must exceed in order to cause a neuron to "fire". If $I_1 * w_1 + w_2 * I_2 + ...w_n * I_n >= T$, i.e. $w_1 * I_1 + w_2 * I_2 + ...w_n * I_n$ is greater than or equal to T, then the *MCP* neuron produces an output of 1. If $w_1 * I_1 + w_2 * I_2 +$

$.....w_n * I_n < T$, i.e. $w_1 * I_1 + w_2 * I_2 + ...w_n * I_n$ is less than T, then the *MCP* neuron produces a 0. So an *MCP* neuron is completely determined by its weights and threshold. By choosing various combinations of weights and threshold, we can produce several different *MCP* neurons.

AND logic function : The truth table the for AND gate is

I_1	I_2	Y (Output)
0	0	0
0	1	0
1	0	0
1	1	1

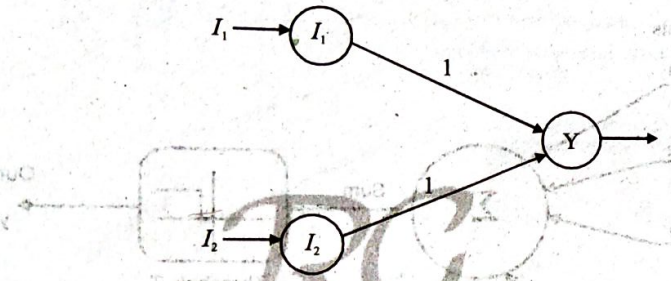


Fig. : (1)

A McCulloch-Pitts neuron to implement AND function is shown in fig.(1). The threshold of the neuron is 2. The output of the neuron Y can be written as $Y = f(y_{in})$.

Reason for choosing threshold = 2 is explained below :

The net input is given by $y_{in} = \Sigma$ weight * input

Since inputs are I_1, I_2 and the weights are 1 and 1 respectively, therefore

$$y_{in} = 1 * I_1 + 1 * I_2 = I_1 + I_2$$

From this the activation fucntions of the output neuron can be formed.

$$Y = f(y_{in}) = 1 \text{ if } y_{in} \geq 2$$
$$0 \text{ if } y_{in} < 2$$

where $Y_{in} = I_1 + I_2$

Now if

(a) $I_1 = I_2 = 1$

and $w_1 = w_2 = 1$ then $Y_{in} = 2$

$\therefore$ If $Th \neq 2$ then answer can not be obtained both inputs = 1. Threshold value can be chosen according to the applications.

and $Y = f(y_{in}) = 1$

(b) If $I_1 = 1$

and $I_2 = 0$ then $y_{in} = 1$

and $Y = f(y_{in}) = 0$. Since $y_{in} = 1 < 2$

Similarly for other combinations.

XOR Funcation : The truth table the for XOR gate is

X_1	X_2	Y (Output)
1	1	0
1	0	1
0	1	1
0	0	0

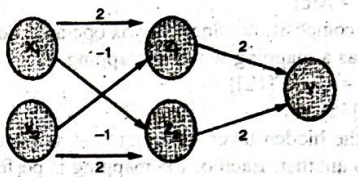


Fig. : (2)

XOR can be modeled using AND NOT and OR;

X_1 *XOR* $X_2 = (X_1$ AND NOT $X_2)$ OR $(X_2$ AND NOT $X_1)$

This explains the network shown above. The first layer performs the two AND NOT's and the second layer performs the *OR*. Both Z neurons and the Y neuron have a threshold of 2.

Section - B

Q.4. Explain the single layer continuous perceptron training algorithm for linearly separable classification. **(20)**

Ans. Algorithm of single continuous perceptron training :

Given, P training pairs

$\{x_1, d_1, x_2, d_2, ..., x_p\ d_p\}$

Augmented input vectors are used

$$y_i = \begin{bmatrix} x_i \\ 1 \end{bmatrix} \text{ for } i = 1, 2, ..., P$$

where x_i is $(n \times 1)$, d_i is (1×1), $i = 1, 2, ..., P$

p denotes step counter within training cycle and k denotes the training step.

(i) Choose $\eta > 0$, l(steepress constant) = 1, $E_{max} > 0$.

(ii) Weight are initialized at w at small random values, counters and error are initialized.

$$k \leftarrow 1, p \leftarrow 1, E \leftarrow 0.$$

(iii) Training starts $\rightarrow$ input y is presented and output is computed i.e.

$$y \leftarrow y_p, d \leftarrow dp, o \leftarrow f(w^t y)$$

(iv) Weights are updated $w^{k+1} \leftarrow w^k + \frac{1}{2}\eta(d^k - o^k)(1 - o^{k^2})y$

(v) Error E is computed $E^{k+1} \leftarrow \frac{1}{2}(d^k - o^k)^2 + E^k$.

(vi) If $p < P$, then $p \leftarrow p + 1$, $k \rightarrow k + 1$ and repeat step 3, otherwise go to step 7.

(vii) Training cycle is completed. If $E < E_{max}$ terminate the session.

If $E \geq E_{max}$ then $E \leftarrow 0$, $p \leftarrow 1$, start new training cycle and go to step 3.

Q.5. Explain error back propagation training algorithm. **(20)**

Ans. The layer network is mapping the input vector z into the output vector o as follows :

$$o = N(z)$$

where N denotes a composite nonlinear matrix operator. For the two-layer net the mapping $z \rightarrow o$ can be represented as a mapping within a mapping, or

$$o = G[W\Gamma[Vz]]$$

$$\Gamma[Vz] = y$$

and it related to the hidden layer mapping $z \rightarrow y$. Note that the right arrows denote mapping of one space into another. Each of the mapping is performed by a single-layer of the layered network. The operator Γ is a nonlinear diagonal operator.

Algorithm :

Given are P training pairs

$$\{z_1, d_1, z_2, d_2, ..., z_p, d_p,\},$$

Where z_i is $(I \times 1)$, d_i is $(K \times 1)$, and $i = 1, 2, ...P$. Note that the Ith component of each x_i is of value – 1 since input vectors have been augmented. Size $J-1$ of the hidden layer having outputs y is selected. Note that the Jth component of y is of value – 1, since hidden layer outputs have also been augmented; y is $(J \times 1)$ and o is $(K \times 1)$.

Step 1 : $\eta > 0$. E_{max} chosen

Weights W and Y are initialized at small random values; W is $(K \times J)$, V is $(J \times I)$.

$$q \leftarrow 1, \quad p \leftarrow 1, E \leftarrow 0$$

Step 2 : Training step starts here.

Input is presented and the layers output computed.

$$z \leftarrow z_p, d \leftarrow d_p$$

$$y_j \leftarrow f(v_j^t z), \text{ for } j = 1, 2, ..., J$$

where v_j, a column vector is the Jth row of V and

$$o_k \leftarrow f(w_k^t y), \text{ for } k = 1, 2, ..., K$$

where w_k, a column vector, is the kth row of W.

Step 3 : Error value is computed

$$E \leftarrow \frac{1}{2}(d_k - o_k)^2 + E, \text{ for } k = 1, 2, ...,K$$

Step 4 : Error signal vectors δ_0 and δ_y of the both layers are computed.

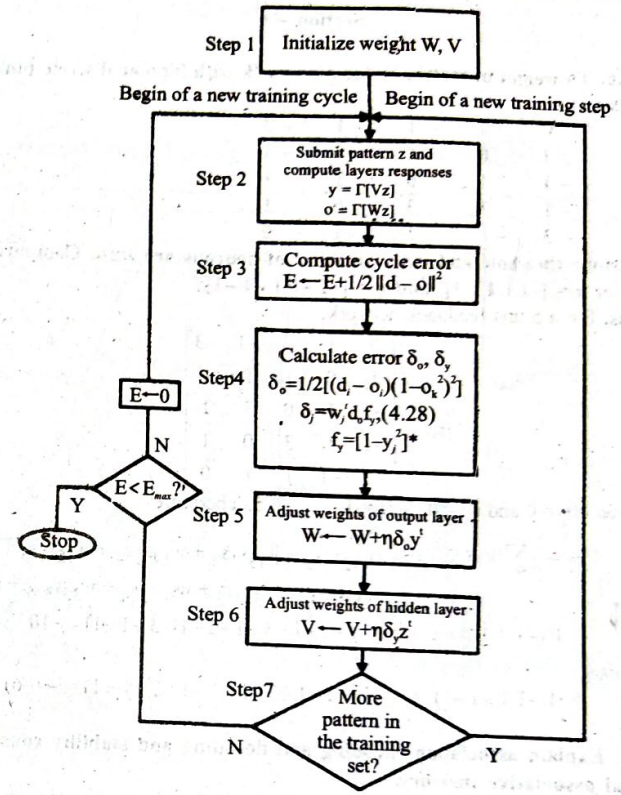
Vector δ_0 is $(K \times 1)$, δ_y is $(J \times 1)$.

The error signal terms of the output layer in this step are

$$\delta_{ok} = \frac{1}{2}(d_k - o_k)(1 - o_k^2), \text{for } k = 1, 2, ..., K$$

The error signal terms of the hidden layer in this step are

$$\delta_{yi} = \frac{1}{2}(1 - y_j^2)\sum_{k=1}^{k} \delta_{ok} w_{kj}, \text{for } j = 1, 2, ..., J$$



Step 5 : Output layer weights are adjusted :

$$w_{kj} \leftarrow w_{kj} + \eta\delta_{ok}y_j, \text{ for } k = 1, 2, ... K \text{ and } j = 1, 2, ..., J$$

Step 6 : Hidden layer weights are adjusted

$$v_{ji} \leftarrow v_{ji} + \eta\delta_{yj}z_i, \text{ for } j = 1, 2, ... J \text{ and } i = 1, 2, ..., I$$

Step 7 : If $p < P$ then $p \leftarrow p + 1$, $q \leftarrow q + 1$, and go to Step 2; otherwise, go to Step 8.

Step 8 : The training cycle is completed.

For $E < E_{max}$ terminate the training session. Output weights W, V, q and E.

If $E > E_{max}$ then $E \leftarrow 0$, $p \leftarrow 1$, and initiate the new training cycle by going to Step 2.

Section – C

Q.6. The weight of matrix W for a network with bipolar discrete binary neurous is given as :

$$\begin{matrix} & 1 & -1 & -1 & -3 \\ 1 & 0 & 1 & 1 & -1 \\ -1 & 1 & 0 & 3 & 1 \\ -1 & 1 & 3 & 0 & 1 \\ -3 & -1 & 1 & 1 & 0 \end{matrix}$$

Assume threshold and external input of neurous are zero. Compare the values of energy for $v = [-1\ 1\ 1\ 1\ 1]^t$ and $v = [-1\ -1\ 1\ -1\ -1]^t$. (20)

Ans. For a 5 unit Feedback network

$$W = \begin{bmatrix} 0 & 1 & -1 & -1 & -3 \\ 1 & 0 & 1 & 1 & -1 \\ -1 & 1 & 0 & 3 & 1 \\ -1 & 1 & 3 & 0 & 1 \\ -3 & -1 & 1 & 1 & 0 \end{bmatrix}$$

Given, $\theta_i = 0$ and $w_i = 0$, $\forall\ i = 1, 2, \ldots, 5$. Therefore

$$V = -\frac{1}{2}\sum w_{ij} S_i S_j = -(w_{12}s_1s_2 + w_{13}s_1s_3 + w_{14}s_1s_4 + w_{15}s_1s_5 + w_{23}s_2s_3 + w_{24}s_2s_4 + w_{25}s_2s_5 + w_{34}s_3s_4 + w_{35}s_3s_5 + w_{45}s_4s_5)$$

$$V(-1\ 1\ 1\ 1\ 1) = -(-1+1+1+3+1+1-1+3+1+1) = -10$$

Similary,

$$V(-1\ -1\ 1\ -1\ -1) = -(1+1-1-3-1+1-1-3-1+1) = -(-6) = 6$$

Q.7. Explain association encoding and decoding and stability consideration for bidirectional associative memory. (20)

Ans. Bidirectional Associative memory : Bidirectional associative memory is a heteroassociative, contant-addressable memory consisting of two layers. It uses the forward and background information flow to produce an associative search for stored stimulus-response association (kosko 1987,1988). Consider that stored in the memory are p vector association pairs known as

$$\left\{\left(a^{(1)}, b^{(1)}\right), \left(a^{(2)}, b^{(2)}\right), \ldots, \left(a^{(p)}, b^{(p)}\right)\right\} \quad \ldots(i)$$

When the memory neurons are activated, the network evolves to a stable state of two pattern reverberation, each pattern at output of one layer. The stable reverberation corresponds to a local energy minimum. The network's dynamics involves two layers of interaction. Becuase the memory processes information in time and involves bidirectional data flow, it differs in principle from a linear associator, although both networks are used to store assocation pairs. It also differs from the recurrent autoassociative memory in its update mode.

Association Encoding and Decoding : The coding of information (i) into the bidirectional associative memory is done using the customary outer product rule, or by adding p cross-correlation matrices. The formula for the weight matrix is

$$W = \sum_{i=1}^{p} a^{(i)} b^{(i)t} \quad \ldots(ii)$$

where $a^{(i)}$ and $b^{(i)}$ are bipolar binary vector, which are membars of the i'th pair.

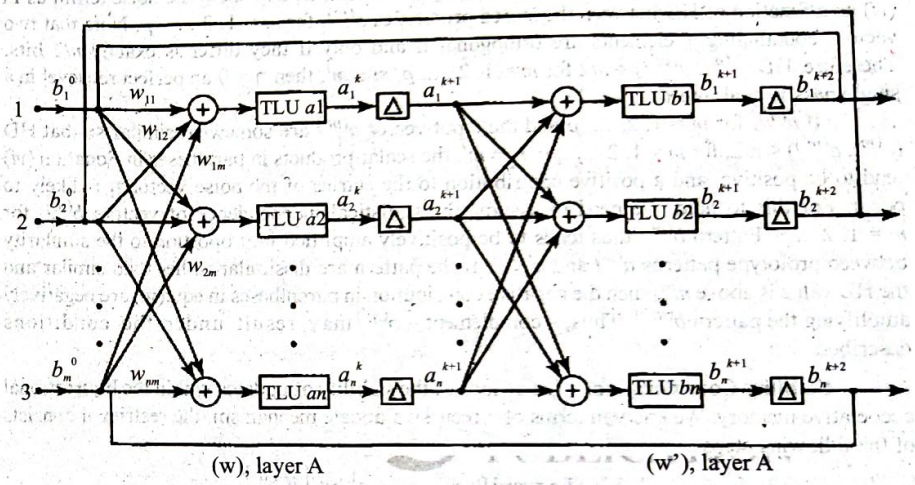


Fig. : Discrete-time bidirectional associative memory expanded diagram.

yielding the following weight values :

$$w_{ij} = \sum_{m=1}^{p} a_i^{(m)} b_j^{(m)} \quad \ldots(iii)$$

Suppose one of the stored pattern, $a(m')$, is presented to the memory. The retrieval proceeds as follows

$$b = \Gamma\left[\sum_{m=1}^{p}\left(b^{(m)} a^{(m)t}\right) a^{(m')}\right] \quad \ldots(iv)$$

which further reduces to

$$b = \Gamma\left[n b^{(m')} + \sum_{m \neq m'}^{p} b^{(m)} a^{(m)t} a^{(m')}\right] \quad \ldots(v)$$

The net_b vector inside brackets in Equation (v) contains a signle term $nb^{(m')}$ additive with the noise term η of value

$$\eta = \sum_{m \neq m'}^{p} b^{(m)}\left(a^{(m)t} a^{(m')}\right) \quad \ldots(vi)$$

Assuming temporarily the orthogonality of stored patterns $a^{(m)}$, for $m = 1. 2, \ldots, p$, the noise the term η reduces to zero. Therefore, immediate stabilization and exact association $b = b(m')$ occurs within only a single pass though layer B. If the input vector is a distorted version of pattern $a^{(m')}$, the stabilization at $b^{(m')}$ is not imminent, however, and depends on many factors such as the HD between the key vector and prototype vectors, as well as on the orthogonality or HD between vectors $b^{(i)}$, for $i = 1, 2, \ldots, p$.

To gain better insight into the memory performance, let us look at the noise term η as in (vi) as a function of HD between the stored prototypes $a^{(m)}$, for $m = 1, 2, \ldots, p$. Note that two vectors containging $\pm$ elements are orthogonal if and only if they differ is exactly $n/2$ bits. Therefore, HD $(a^{(m)}, a^{(m')}) = n/2$ for $m = 1, 2, \ldots, p$, $m \neq m'$, then $\eta = 0$ an perfect retrievel in a single pass is guaranteed.

If $a^{(m)}$, for $m = 1, 2, \ldots, p$ and the input vector $a^{(m')}$ are somewhat similar so that HD $(a^{(m)}, a^{(m')}) < n/2$, for $m = 1, 2, \ldots, p$, $m \neq m'$, the scalar products in paratheses in Equation (vi) tend to be positive, and a positive contribution to the entries of the noise vector η is likely to occur. For this to hold, we need to assume the statistical independence of vectors $b^{(m)}$, for $m = 1, 2, \ldots p$. Pattern $b^{(m')}$ thus tends to be positively amplified in proportion to the similarity between prototype patterns $a^{(m)}$ and $a^{(m')}$. If the pattern are dissimilar rather than similar and the HD value is above $n/2$, then the negative contribution in parentheses in equ.(vi) are negatively amplifying the pattern $b^{(m')}$. Thus, a complement $-b^{(m')}$ may result under the conditions described.

Stability Considerations : Let us look at the stability of updates within the bidirectional accociative memory. We know in terms of a recursive update mechanism, the restrieval consists of the following steps :

$$\begin{aligned}
&\text{First Forward Pass :} && a^1 = \Gamma\left[Wb^0\right] \\
&\text{First Backward Pass :} && b^2 = \Gamma\left[W^t a^1\right] \\
&\text{Second Forward Pass :} && a^3 = \Gamma\left[Wb^2\right] \\
&\qquad\qquad\qquad && \vdots \\
&k/2\text{'thBackward Pass :} && b^k = \Gamma\left[W^t a^{k-1}\right]
\end{aligned} \qquad \ldots(vii)$$

As the updates in (vii) continue and the memory comes to its equilibrium at the k'th step, we have $a^k \rightarrow b^{k+1} \rightarrow a^{k+2}$, and $a^{k+2} = a^k$. In such a case, the memory is said to be *bidirectionally stable*. This corresponds to the energy function reaching one of its minima after which any further decrease of its value is impossible. Let us propose the energy fucntion for minimization by this system in transition as

$$E(a,b) \overset{\Delta}{=} -\frac{1}{2}a^t Wb - \frac{1}{2}b^t W^t a \qquad \ldots(viii)$$

The reader may easily verify that this expression reduces to

$$E(a, b) = -a^t Wb \qquad \ldots(ix)$$

Let us evaluate the energy changes during a single pattern recall. The summary of thresholding bit updates for the outputs of layer A can be

$$\overset{\Delta a_i}{i=1,2,\ldots,n} = \begin{cases} 2 & > 0 \\ 0 \quad \text{for} \quad \sum_{j=1}^{m} w_{ij} b_j & = 0 \\ -2 & < 0 \end{cases} \qquad \ldots[(x)a]$$

and for the outputs of layer B they result as

$$\overset{\Delta b_j}{j=1,2,\ldots,n} = \begin{cases} 2 & > 0 \\ 0 \quad \text{for} \quad \sum_{i=1}^{m} w_{ij} a_i & = 0 \\ -2 & < 0 \end{cases} \qquad \ldots[(x)b]$$

The gradients of energy (ix) with respect to a and b can be computed, respectively, as

$$\nabla_a E(a,b) = -Wb \qquad \ldots[(xi)a]$$

$$\nabla_b E(a,b) = -W^t a \qquad \ldots[(xi)b]$$

The bitwise update expression (x) translate into the following energy changes due to the single bit increments Δa_i and Δb_j :

$$\Delta E a_i(a,b) = -\left(\sum_{j=1}^{m} w_{ij} b_j\right) \Delta a_i, \text{ for } i = 1,2, \ldots, n \qquad \ldots[(xii)a]$$

$$\Delta E b_j(a,b) = -\left(\sum_{i=1}^{m} w_{ij} a_i\right) \Delta b_j, \text{ for } j = 1,2, \ldots, m \qquad \ldots[(xii)b]$$

Inspecting the right sides of equations (xii) and comparing them with the ordinary update rules as in (x) lead to be consculsion that $\Delta E \leq 0$. As with recurrent autoassociative memory, the energy changes are nonpositive. Since E is a bounded function from below according to the following inequality :

$$E(a, b) \geq \sum_{i=1}^{n} \sum_{j=1}^{m} |w_{ij}|$$

then the memory converages to a stable point. The point is a local minimum of the energy function, and the memory is said to be bidirectionally stable. Moreover, no restrictions exits regarding the choice of matrix W, so any arbitrary real $n \times m$ matrix will result in bidirectionally stable memory.

Sectiion – D

Q.8. Explain in detail unsupervised learning of clusters. **(20)**

Ans. Unsupervised learning of clusters : Learning is based on clustering of input data. No prior knowledge is assumed to be available regarding an input membership in a particular class. Rather, gradually detected characteristics and a history of training will be used to assist the network in defining classes and possible boundaries between them.

Clustering : Clustering is understood to be the grouping of similar objects and separating of dissimilar ones. The objective of clustering neural networks is to categorize or cluster data. The classes must first be found from the correlations of an input data stream. Since the network

actually deals with unlabeled data, the clustering should be followed by labeling clusters with appropriate category names or numbers. This process of providing the category of objects with a label is usually termed as calibration.

Although the algorithm won't have names to assign to these clusters, it can produce them and then use those clusters to assign new examples into one or the other of the clusters. This is a data-driven approach that can work well when there is sufficient data; for instance, social information filtering algorithms, such as those that Amazon.com use to recommend books, are based on the principle of finding similar groups of people and then assigning new users to groups. In some cases, such as with social information filtering, the information about other members of a cluster (such as what books they read) can be sufficient for the algorithm to produce meaningful results. In other cases, it may be the case that the clusters are merely a useful tool for a human analyst. Unfortunately, even unsupervised learning suffers from the problem of overfitting the training data. There's no silver bullet to avoiding the problem because any algorithm that can learn from its inputs needs to be quite powerful.

Suppose we are given a set of patterns without any information as to the number of classes that may be present in the set. The clustering problem in such a case is that of identifying the number of classes according to a certain criterion, and of assigning the membership of the patterns in these classes. The clustering technique presented below :

Assumptions : The number of classes is known a priori. The pattern set $\{x_1, x_2, \ldots\ldots, x_n\}$ is submitted to the input to determine decision functions required to identify possible clusters. Since no information is available from the teacher as far as the desired classifier's responses, we will use the similarity of incoming patterns as the criterion for clustering. To define a cluster, we need to establish a basis for assigning patterns to the domain of particular cluster.

How we will assign a pattern to domain of particular cluster? There are 2 rules for this.

(1) The most common similarity rule is to find the Euclidean distance between 2 patterns x for x_i defined as

$$|X - X_i| = [(X - X_i)^t (X - X_i)]^{1/2} \quad \ldots(i)$$

Smaller the distance, closer the patterns. Using (i) the distance between all the pairs are computed. Now how to distinguish the clusters? A distance T can then be chosen to discriminate clusters. The value T is understood as the maximum distance between patterns within a single cluster. Fig.(a) shows an example of two clusters with a T value chosen to be greater than the typical within-cluster distance but smaller than the between-cluster distance.

Let there are 2 patterns X_1 and X_2 and we have to find which pattern is similar to pattern X. Then using (1) distance is calculated. The value for which the distance is small, closer the pattern is with x.

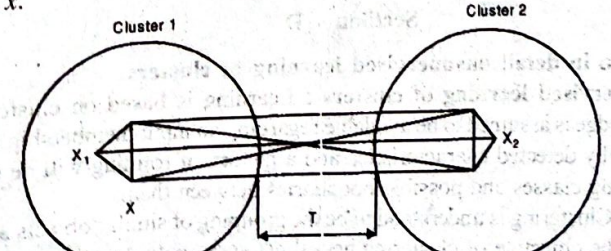


Fig.(a) : Measure of similarity between 2 clusters using distance

(2) Another similarity rule is the cosine of the angle between x and x_i:

$$\cos \alpha = \frac{(X' X_i)}{\| X \| . \| X_i \|} \quad \ldots(ii)$$

This rule is particularly useful when clusters develop along certain principal and different axes as shown in fig.(b). For $\cos \alpha_2 < \cos \alpha_1$ pattern x is more similar to x_2 than to x_1. It would thus be natural to group it with the second of the two apparent clusters. To facilitate this decision, the threshold angle α_t can be chosen to define the minimum angular cluster distance. It should be noted, however, that the measure defined in equation (ii) should be used according to certain additional qualifications. If the angular similarity criterion equation (ii) is to be efficient, vectors x_1, x_2 and x should be of comparable, or better, identical lengths.

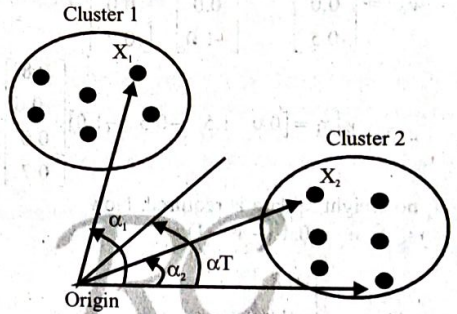


Fig.(b) : Measure of similarity between 2 clusters using normalized scalar product

Q.9. Implement the perceptron rule training of the network using f (net) = sgn (net), c = 0.1 & the following data specifying the initail weights W′ and two training pairs.

$$w_0 = \begin{bmatrix} 1.0 \\ -1.0 \\ 0.0 \\ 0.5 \end{bmatrix}, d_0 = -1.0, d_1 = -1, d_2 = 1$$

d_0, d_1, d_2 are teacher's signal.

and

$$x_0 = \begin{bmatrix} 1.0 \\ -2.0 \\ 0.0 \\ -1.0 \end{bmatrix}, x_1 = \begin{bmatrix} 0.0 \\ 1.5 \\ -0.5 \\ -1.0 \end{bmatrix}, x_2 = \begin{bmatrix} -1.0 \\ 1.0 \\ 0.5 \\ -1.0 \end{bmatrix} \quad \textbf{(20)}$$

Ans.

$$z_0 = w_0^T x_0 = [1.0 \quad -1.0 \quad 0.0 \quad 0.5] \begin{bmatrix} 1.0 \\ -2.0 \\ 0.0 \\ -1.0 \end{bmatrix} = +2.5$$

Note that sgn $(2.5) \neq d_0$. (Note c = 0.1). The weight update is evaluted as :

$$= d - a$$
$$= d_0 - f(w'_0 x_0)$$
$$= -1 - f(2.5)$$
$$= -1 - 1 \text{ [because in perceptron learing rule signum function is used and sgn } (2.5) = +1]$$

$$w_1 = w_0 + 0.1(-1-1)x_0$$

$$w_1 = \begin{bmatrix} 1.0 \\ -1.0 \\ 0.0 \\ 0.5 \end{bmatrix} - 0.2 \begin{bmatrix} 1.0 \\ -2.0 \\ 0.0 \\ -1.0 \end{bmatrix} = \begin{bmatrix} 0.8 \\ -0.6 \\ 0.0 \\ 0.7 \end{bmatrix}$$

$$z_1 = w_1^t x_1 = [0.0 \quad 1.5 \quad -0.5 \quad -1.0] \begin{bmatrix} 0.8 \\ -0.6 \\ 0.0 \\ 0.7 \end{bmatrix} = -1.6$$

Since sgn $(-1.6) = d_1$ no weight update is required. Now

$$w_2 = w_1 + 0.1(-1+1).x_1$$

$$w_2 = w = \begin{bmatrix} 0.8 \\ -0.6 \\ 0 \\ 0.7 \end{bmatrix}$$

$$x_2 = w_2^T x_2 = [0.8 \quad -0.6 \quad 0 \quad 0.7] \begin{bmatrix} -1.0 \\ 1.0 \\ 0.5 \\ -1.0 \end{bmatrix} = 2.1$$

Note that sgn $(-2.1) \neq d_2$. The weight update is evaluated as :

$$w_3 = w_2 + 0.1(1+1)x_2$$

$$w_3 = \begin{bmatrix} 0.8 \\ -0.6 \\ 0 \\ 0.7 \end{bmatrix} + 0.2 \begin{bmatrix} -1.0 \\ 1.0 \\ 0.5 \\ -1.0 \end{bmatrix} = \begin{bmatrix} 0.6 \\ -0.4 \\ 0.1 \\ 0.5 \end{bmatrix}$$

NEURAL NETWORKS

Dec - 2016
Paper Code : CSE-407-F

Note : ***Attempt five questions in all, selecting one question from each Section. Question No. 1 is compulsory. All questions carry equal marks.***

Q.1. Explain the following : **(20)**

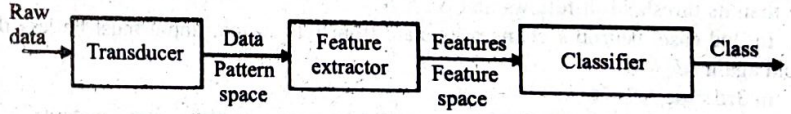
(a) Delta learning rule
(b) Pattern classification
(c) Stability.
(d) Recall mode.

Ans. (a) Delta learning rule : The Delta learning algorithm is used for automatic updating of connection weights when processing information in a connectionist or neural network. This Delta rule was originally formulated by Widrow and Hoff, but obtained its final name because it adjusts connection weights according to the difference between the actual output of a network and the intended output.

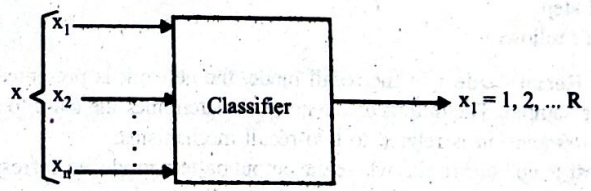
Thus, it basically compares the output of an artificial network with the output that it was intended to produce, and then adjusts the connections within the network so as to better produce the intended output.

Ans.(b)Pattern classification : The main goal of pattern classification is to assign a physical object, event or phenomenon one of prespecified classes (categories).

For example : We (human beings) classify various objects into different categories like living non-living things, plants, weather, voice etc. The interpretation of data has been learned as a result of repetitive inspection and classification of examples.



(a) Block diagram of classification system



(b) General diagram of pattern classifier

Fig.

When a person preceives a pattern, an inductive inference is made and the perception is associated with some general concepts derived froms persons past experience. The problem of

patterns classification may be regarded as one of discriminating the input data within object population via search for the different attributes among members of the population.

In some of the applications, the various classifying aids frame helpful because of the growing complexity of the human environment. Fig. shows the block diagram of recognition and classification system.

Applications of Pattern Classifier are

(i) Fingerprint Identification,

(ii) Radar and Signal Detection,

(iii) Speech Recognition etc.

Fig. shows the block diagram of recognition and classification system.

Ans.(c)Stability of Hopfield Nets : Selection of a Liapunor function as energy function of Hopfield net ensures stability.

Suppose neuron K changes its state from time t to time $t + 1$.

$\therefore$ Change in energy level

$$\Delta E = E(t-1) - E(t)$$

$$= -\left[\left(\frac{2}{2}\sum_i w_{ki} - V_i\right) + I_k - Q_k\right]\Delta V_k \text{ because for neuron k.}$$

$$\Delta V_k = V_k(t+1) - V_k(t)$$

Since a neuron assumes a state of either 0 or 1.

$\therefore$ 3 possible cases for ΔV_k

$$\Delta V_k = \begin{cases} 1 \\ -1 \\ 0 \end{cases}$$

$\therefore$ In the first case, neuron k changes its state from 0 to 1, $\therefore$ its total input must be greater than its threshold. It follows that $\Delta E < O$.

In 2nd case, neuron k changes its state from 1 to $\therefore$ total input must be less than its threshold again $\Delta E < O$.

In 3rd case $\Delta E = 0$

$\Rightarrow$ Energy level is always non-increasing (decreasing or remaining constant).

Whenever a neuron changes its state, the network will eventually reach a minimum energy state and step.

The result follows.

Ans.(d) Recall mode : In the recall mode, the network is presented with the feature vector of a single sample. The output of the network determines the class to which the sample belongs. *Hetero-association* is related to two recall mechanisms:

(i) Nearest-neighbour recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented.

(ii) Interpolative recall, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented. Yet another paradigm, which is a variant associative mapping is classification, *i.e.* when there is a fixed set of categories into which the input patterns are to be classified.

Section - A

Q.2.(i) Describe in detail about Biological Neurons. (10)

Ans. Biological Neuron : The elementary nerve cell called a neuron is the fundamental building block of biological neural network. The three main components of a biological neuron are :

1. A neuron cell body called *soma* .
2. Branching extensions called *dendrites* for receiving input, and
3. An *axon* that carries the neuron's output to the dendrites of other neurons.

In the human brain, a typical neuron collects signals from others through a host of fine structures called *dendrites*. The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. The axon-dendrite contact organ is called a synapse. At the end of each branch, a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes. The neuron is able to respond to the total of its inputs aggregated within a short time interval called period of latent summation.

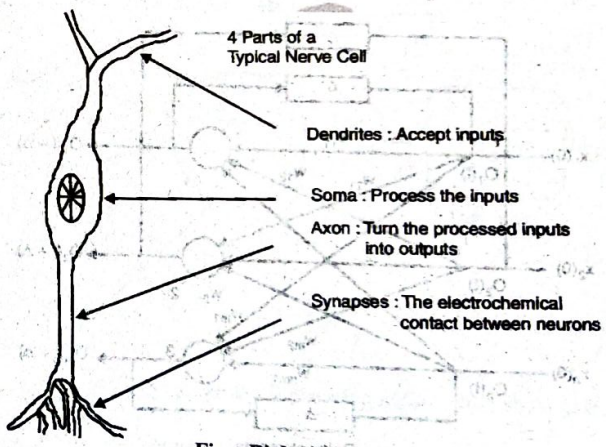


Fig. : Biological Neuron.

Q.2.(ii) What do you mean by Feed forward and Feedback Networks ? Explain. (10)

Ans. Feedforward Network : Feedforward neural network is a biologically inspired classification algorithm. It consists of a (possibly large) number of simple neuron-like processing units, organized in layers. Every unit is a layer is connected with all the units in the previous layer. These connections are not all equal; each connection may have a different strength or weight. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called nodes. Data enters at the inputs and passes through the

network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called feedforward neural networks.

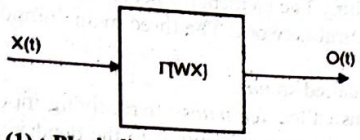


Fig.(1) : Block diagram of Feedforward Network

Feedback Network/Recurrent Network : Recurrent neural netwroks (*RNN*) have a closed loop in the network topology. They are developed to deal with the time varying or time-lagged patterns and are usable for the problems where the dynamics of the considered process is complex and the measured data is noisy. Specific groups of the units get the feedback signals from the previous time steps and these units are called context unit. The *RNN* can be either fully or partially connected. In a fully connected *RNN* all the hidden units are connected recurrently, whereas in a partially connected *RNN* the recurrent connections are omitted paratially. Examples of recurrent neural networks are Hopfield networks, Regressive networks, Jordan-Elman networks, and Brain-State-In-a-Box (BSB) networks.

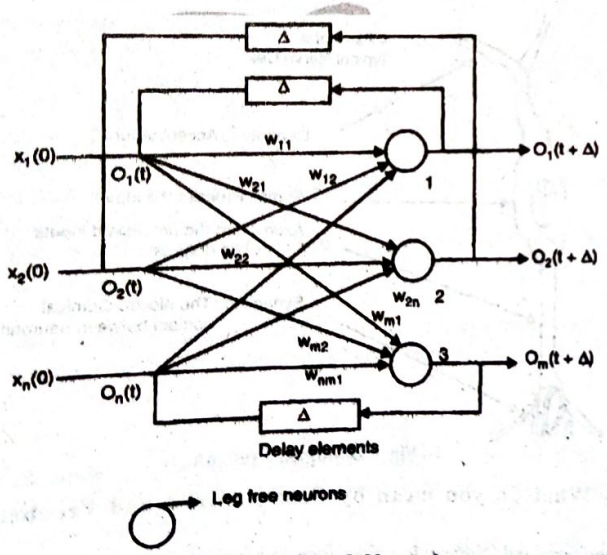


Fig.(3) : Feedback Network

Q.3 Define ANN. Describe in detail about different models of ANN. (20)

Ans. An artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. It is composed of a large number of highly interconnected processing elements(neurons) working in

parallel to solve specific problems. The weights on the connections encode the knowledge of a network. Each neuron has a local memory and the output of each neuron depends upon only the input signal arriving at the neuron and value in neuron's memory. ANNs, like people, learn by example.

Models of Artificial Neural Networks :

(1) McCulloch-Pitts (MCP) Neuron Model : The early model of an artificial neuron is introduced by Warren McCulloch and Walter Pitts in 1943. The MCP neuron

(i) Has a number of inputs I_1, ... I_n. These inputs represent the incoming signals received from the neuron's synapses. So I_i can either be 1, which corresponds to the presence of an incoming signal from the ith connection, or I_i is 0, which corresponds to the absence of a signal from the ith connection.

(ii) The *MCP* neuron produces an output y, which can either be 1, corresponding to the neuron sending a signal, or it can be 0, corresponding to the neuron remaining at rest. Mathematically an *MCP* neuron is a function which takes an n-tuple of 1's and 0's and produces a 1 or a 0. In order to quantity the influence each synapse has on the *MCP* neuron, we assign a weight w_i to each input I_i.

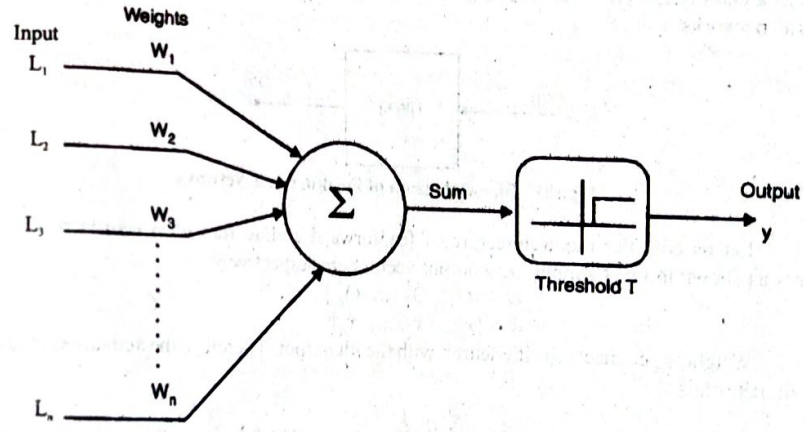


Fig.(a) : Symbolic Illustration of linear threshold gate

(iii) The weights are decimal numbers, and the size of the weight corresponds to the amount of influence the associated connection has on the *MCP* neuron. Positive weights correspond to excitatory synapses, and negative weights correspond to inhibitory synapses. Each input is multiplied by the corresponding weight, and these weighted inputs are then added together to produce

$$\text{Sum} = \sum_{i=1}^{N} I_i W_i \quad \text{...(i)}$$

$$y = f(\text{sum}) \quad \text{...(ii)}$$

where $w_i * I_i$ means w_i multiplied by I_i. This corresponds to the summing of the incoming signals that is assumed to be performed by a neuron. Finally, there is a threshold T, which is a decimal number that is compared to the weighted sum of the signals. This corresponds to the apparent threshold value that the sum of the received signals must exceed in order to cause a neuron to "fire". If $I_1 * w_1 + w_2 * I_2 + ...w_n * I_n >= T$, i.e. $w_1 * I_1 + w_2 * I_2 + ...w_n * I_n$ is greater than or equal to T, then the *MCP* neuron produces an output of 1. If $w_1 * I_1 + w_2 * I_2 + ...w_n * I_n < T$, i.e. $w_1 * I_1 + w_2 * I_2 + ...w_n * I_n$ is less than T, then the *MCP* neuron produces a 0. So an *MCP* neuron is completely determined by its weights and threshold. By choosing various combinations of weights and threshold, we can produce several different *MCP* neurons.

(2) Feedforward Network : Feedforward neural network is a biologically inspired classification algorithm. It consists of a (possibly large) number of simple neuron-like processing units, organized in layers. Every unit is a layer is connected with all the units in the previous layer. These connections are not all equal; each connection may have a different strength or weight. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called nodes. Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called feedforward neural networks.

Fig.(b) : Block diagram of Feedforward Network

Let us consider the architecture of feedforward architecture of m neuron receiving n inputs as shown in fig. Its inputs and output vectors are respectively

$$O = [O_1 \quad O_2 \quad .... \quad O_m]^t$$
$$X = [x_1 \quad x_2 \quad .... \quad x_n]^t \qquad ...(i)$$

Weight w_{ij} connects the ith neuron with the jth output. Therefore the activation value for the ith neuron is

$$\text{net}_i = \sum_{j=1}^{n} W_{ij} x_j \qquad \text{for } i = 1, 2, ..., n \qquad ...(ii)$$

$$\text{Output is } O_i = f(W_i^t x) \qquad \text{for } i = 1, 2, ... m \qquad ...(iii)$$

where weight vector W_i contains the weighting leading towards the ith output node and is defined as

$$W_i = [w_{i1}, w_{i2}, ..., w_{in}]^t \qquad ...(iv)$$

Introducing the nonlinear matrix operator Γ the mapping of input space x to output space O implemented by

$$O = \Gamma (WX) \qquad ...(v)$$

where W is called weight matrix.

Since there is no time delay between the input x and the output o, therefore we can write

$$O(t) = \Gamma [Wx(t)]$$

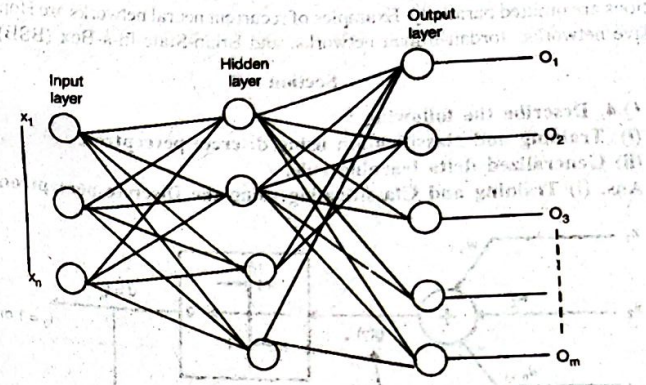


Fig.(c) : Feedforward Network

Feedback Network/Recurrent Network : Recurrent neural netwroks (*RNN*) have a closed loop in the network topology. They are developed to deal with the time varying or time-lagged patterns and are usable for the problems where the dynamics of the considered process is complex and the measured data is noisy. Specific groups of the units get the feedback signals from the previous time steps and these units are called context unit.

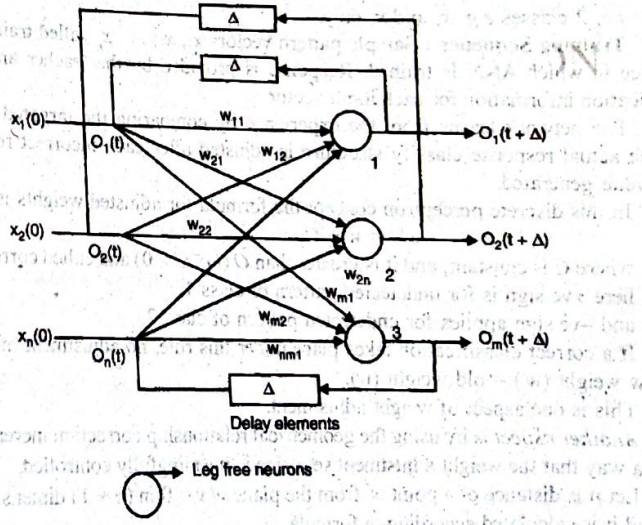


Fig.(d) : Feedback Network

The *RNN* can be either fully or partially connected. In a fully connected *RNN* all the hidden units are connected recurrently, whereas in a partially connected *RNN* the recurrent

connections are omitted paratially. Examples of recurrent neural networks are Hopfield networks, Regressive networks, Jordan-Elman networks, and Brain-State-In-a-Box (BSB) networks.

Section - B

Q.4. Describe the following :
(i) Training and classification using discrete perceptron. **(10)**
(ii) Generalized delta learning rule. **(10)**

Ans. (i) Training and Classificating using the Discrete perceptron :

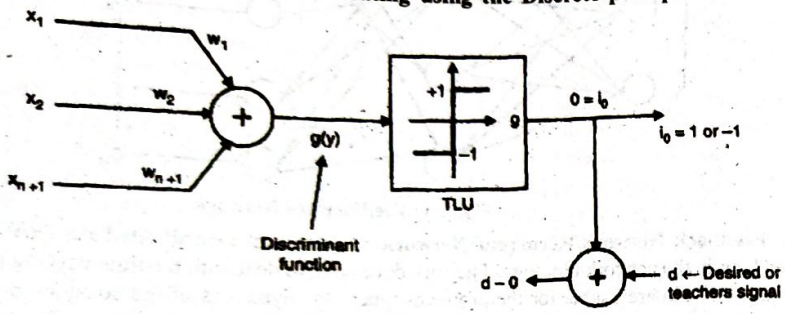


Fig. : Linear dichotomizer using TLU based perceptron.

Dichotomizer : Classifier that divides the pattern space into 2 classes. (D_i means 2). *i.e.* 2 classes *e.g.* x_1 and x_2 only.

Training Sequence : Sample pattern vectors $x_1, x_2, \ldots, x_p$ called training sequence *i.e.* sequence in which ANN is trained. Response is provided by the teacher and it specifies the classification information for each input vector.

The network learns from the experience by comparing the targated correct response with the actual response classify structure is adjusted after each incorrect response based on error value generated.

In this discrete perceptron concept the formula for adjusted weights is

$$w' = w \pm Cy$$

where C is constant, and it is greater than O (*i.e.* $C > 0$) and called correction increment.

here +ve sign is for undetected pattern of class 1

and –ve sign applies for undetected pattern of class 2.

If a correct classification takes place under this rule, no adjustment of weight is made, then new weight (w') = old weight (w).

This is one aspect of weight adjustment.

Another aspect is by using the geometrical relationship correction increment C is chosen in such a way that the weight adjustment setp size is meaningfully controlled.

Let p is distance of a point w' from the plane $w'y = 0$ in $(n + 1)$ dimensional. Euclidean space and it is calculated according to formula

$$p = \pm \frac{w^{1t} y}{\| y \|} \quad \ldots(i)$$

Since p is the distance and by definition of the distance p is always a non-negative scalar. ∴ can be written as

$$p = \frac{w^{1t} y}{\| y \|} \quad \ldots(ii)$$

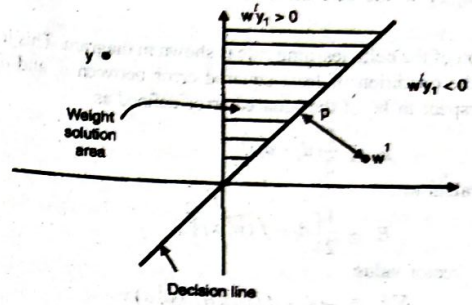


Fig.

Now C must be selected such that corrected weiht vector w^2 based on $w^1 = w \pm cy$ dislocated on the decision hyperplane $w^{1t} y = 0$ which is the decision hyperplane used for particular correction step.

$\Rightarrow$ $w^{2t} y = 0$

$(w^1 \pm cy)^t y = 0$

∴ $(w^1 \pm cy^t) y = 0$

$w^{1t} y \pm cy^t y = 0$

or $c\, y^t y = \mp w^{1t} y \quad \ldots(iii)$

$$c = \mp \frac{w^{1t} y}{y^t y} \quad \ldots(iv)$$

Since c is correction constant and is positive

$$\therefore \quad c = \frac{|w^{1t} y|}{y^t y} \quad \ldots(v)$$

or length of weight adjustment vector cy can be expressed as

$$\| cy \| = \frac{|w^{1t} y|}{y^t y} \| y \| \quad \ldots(vi)$$

But $y^t y = \| y \|^2$

$\Rightarrow$ required distance p from the point w^1 to decision plane and expressed by (ii) is identical to length of the weight increment vector (vi).

∴ For this the correction increment c is therefore not constant and depends on current tracking pattern as expressed of (v).

Ans.(ii) Delta learning rule : Delta learning rule is only valid for continuous activation functions. This rule is used in the supervised training mode.

The learning signal for this rule is called delta and is defined as follows :

$$r \cong \left[d_i - f(w_i^t x)\right] f'(w_i^t x) \quad \ldots(i)$$

The term $f'(w_i^t x)$ is the derivative of the activation function f (net) computed for net $= W_{ix}$.

The explanation of the delta learning rule is shown in diagram. This learning rule can be readily derived from the condition of least squared error between o_i and d_i. Calculating the gradient vector with respect to W_i of the squared error defined as

$$E \cong \frac{1}{2}(d_i - o_i)^2 \quad \ldots(ii)$$

which is equivalent to

$$E \cong \frac{1}{2}\left[d_i - f(w_i^t x)\right]^2 \quad \ldots(iii)$$

Error gradient vector value

$$\nabla E = -[d_i - f(w_i^t x)] f'(w_i^t x) x \quad \ldots(iv)$$

The components of the gradient vector are

$$\frac{\partial E}{\partial w_{ij}} = -\left[d_i - f(w_i^t x)\right] f'(w_i^t x) x_j \quad \ldots(v)$$

for $j = 1, 2, \ldots, n$

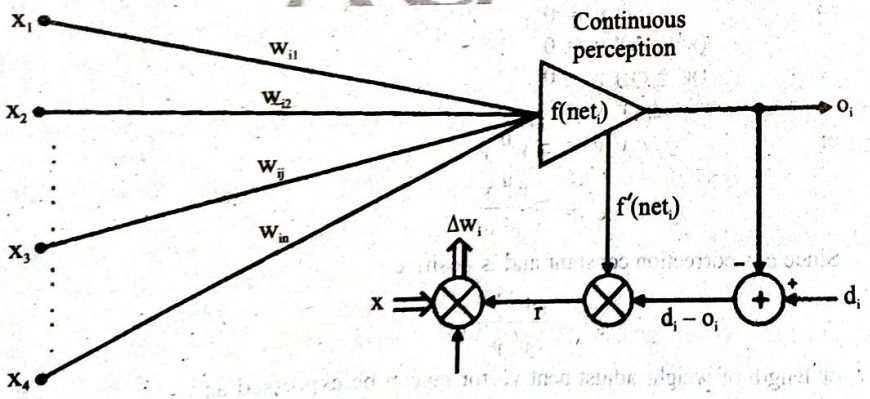


Fig.

Since the minimization of the error requires the weight changes to be in the negative gradient direction, we take

$$\Delta W_i = -\eta \nabla E \quad \ldots(vi)$$

where η is a +ve equation from (v) and (vi)

$$\Delta W_i = \eta\left[d_i - f(w_i^t x)\right] f'(w_i^t x) x \quad \ldots(vii)$$

or, for the single weight the adjustment becomes

$$\Delta w_{ij} = \eta(d_i - o_i) f'(\text{net}_i) x_j \quad \ldots(viii)$$

for $j = 1, 2, \ldots, n$

This delta rule was introduced by McClelland and Rumelhart in 1986. This rule parallels the discrete perceptron training rule.

This rule is also called as continuous perceptron training rule.

Q.5. Give a complete description about multi layer feed forward networks. (20)

Ans. 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. Recall is instantaneous in this type of network (we will discuss this later in the section on the uses of ANNs). These networks can be used to realize complex input/output mappings.

This type of network (Fig.) 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.

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).

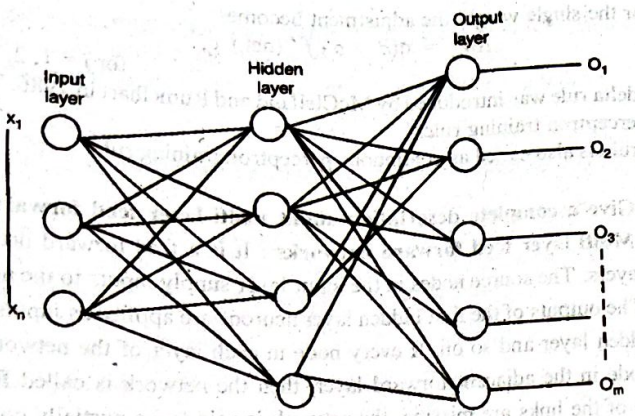


Fig. : A Multilayer Feedforward Network

Short characterization of feedforward networks :

(1) Typically, activation is fed forward from input to output through 'hidden layers', though many other architectures exist.

(2) Mathematically, they implement static input-output mappings.

(3) Most popular supervised training algorithm: backpropagation algorithm.

(4) Have proven useful in many practival application as approximators of nonlinear functions and as pattern classificators.

Section - C

Q.6. What do you understand by single layer feedback networks ? Explain. (20)

Ans. Single layer feedback network : Recurrent neural netwroks (*RNN*) have a closed loop in the network topology. They are developed to deal with the time varying or time-lagged patterns and are usable for the problems where the dynamics of the considered process is complex and the measured data is noisy. Specific groups of the units get the feedback signals from the previous time steps and these units are called context unit. The *RNN* can be either fully or partially connected. In a fully connected *RNN* all the hidden units are connected recurrently, whereas in a partially connected *RNN* the recurrent connections are omitted paratially. Examples of recurrent neural networks are Hopfield networks, Regressive networks, Jordan-Elman networks, and Brain-State-In-a-Box (BSB) networks.

All types of recurrent neural networks are normally trained with the backpropagation learning rule by minimizing the error by the gradient descent method. Mostly they use some computational units which are called associative memories or context units that can learn

associations among dissimilar binary objects, where a set of binary inputs is fed to a matrix of resistors, producing a set of binary outputs. The outputs are '1' if the sum of the inputs is above a given threshold, otherwise it is zero. The weights (which are binary) are updated by using very simple rules based on Hebbian learning. These are very simple devices with one layer of linear units that maps N inputs (a point in N dimensional space) onto M outputs (a point in M dimensional space). However, they remember the past events.

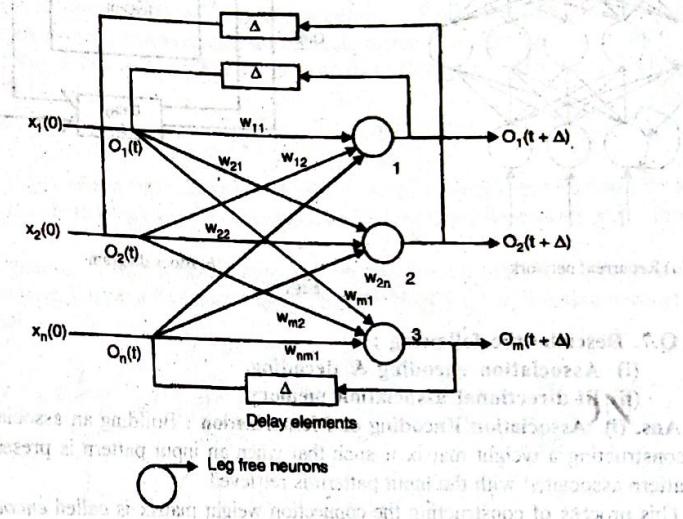


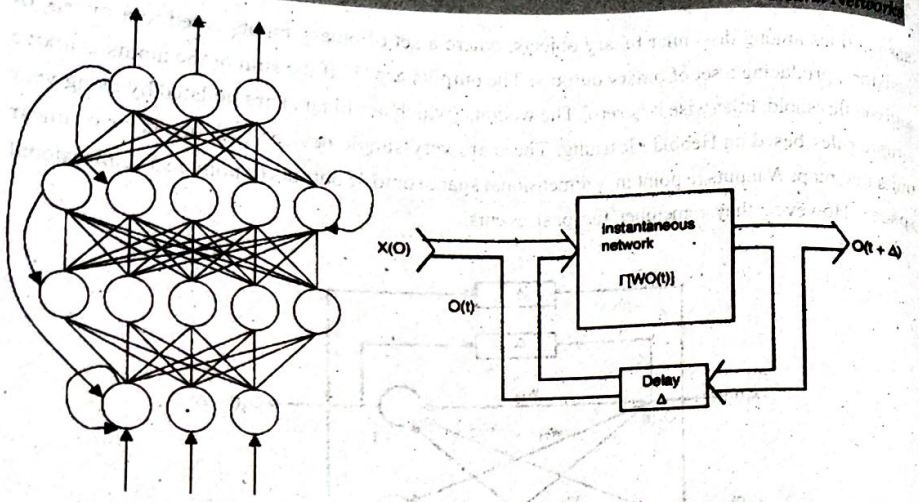
Fig. : Single layer Feedback Network

The time Δ elaased between t and $t + \Delta$ is introduced by the delay elements in the feedback loop. Using the notation introduced for feedforward networks, the mapping of $O(t)$ into $O(t + \Delta)$ can be written as

$$O(t + \Delta) = \Gamma [WO(t)]$$

Note that the input $x(t)$ is only needed to initialize this network so that $O(0) = x(0)$. The input is then removed and the system remains autonomous for $t > 0$.

Recurrent networks typically operate with discrete representation of data; they employ neurons with a hard-limiting activation function. A system with discrete time inputs and a discrete data representation is called an automaton.



(*a*) Recurrent network (*b*) Block diagram

Fig.

Q.7. Describe the following : **(20)**

(i) Association encoding & decoding.

(ii) Bi-directional association memory.

Ans. (i) Association Encoding or Memorization : Building an associative memory means, constructing a weight matrix w such that when an input pattern is presented, and the stored pattern associated with the input pattern is retrieved.

This process of constructing the connection weight matrix is called *encoding*. During encoding, for an associated pattern pair (x_k, y_k) the weight values of the correlation matrix w_k are computed as

$$(w_{ij})_k = (x_i)_k (y_j)_k \text{ where}$$

$(x_i)_k$ represents the ith component of pattern x_k, and

$(y_j)_k$ represents the jth component of pattern y_k,

for $i = 1, 2, \ldots\ldots, m$ and $j = 1, 2, \ldots\ldots, n$

Constructing of the connection weight matrix w is accomplished by summing up the individual correlation matrices w_k, i.e.,

$$w = \alpha \sum_{k=1}^{p} w_k$$

where α is the proportionality or normalizing constant.

Association Decoding (Retrieval or Recollection) : After memorization, the process of retrieving a stored pattern, given an input pattern, is called *decoding* .

Given an input pattern x, the decoding or recollection is accomplished by :

First compute the net input to the output units using

$$\text{Input } j = \sum_{i=1}^{m} x_i w_{ij}$$

where input j is weighted sum of the input or activation value of node j, for $j = 1, 2, \ldots\ldots, n$.

Then determine the units outputs using a bipolar output function ;

$$y_j = \begin{cases} +1 \text{ if input } j \geq \theta \\ -1 \text{ otherwise} \end{cases}$$

where θ_j is the threshold value of output neuron j.

Ans.(ii) Bi-directional Associative Memory : BAM (Bi-directional Associative Memory) is a hetero asosciative, content-addressable memory consisting of 2 layers.

The network structure of BAM is similar to that of linear associator but connections are bidirectional that is

$$w_{ij} = w_{ji} \quad \text{for} \begin{cases} i = 1, 2, \ldots\ldots, m \\ j = 1, 2, \ldots\ldots, n \end{cases}$$

i.e., BAM allows forward and backward flow of information between the layers. In BAM the units in both layers serve as both input and output units depending on the direction of propagation.

Propagating the signals from X layer to Y layer makes the units in X layer to act as input units while units in Y layer act as output units. Same is true for other direction because memory is bidirectional.

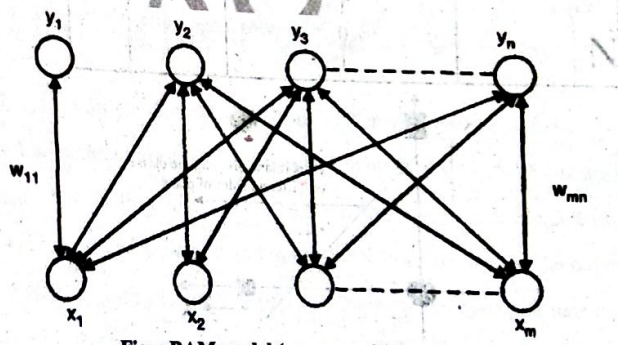


Fig. : BAM model (arrow are bidirectional).

BAM model can perform both auto and hetero associative recall of stored information.

Like is linear associator and hopfield network encoding in BAM can be carried out by using

$$w_k = x_k^T y_k \quad \text{(to store a single associated pattern pair)}$$

and $$w = \alpha \sum_{k=1}^{p} w_k \quad \text{(to store simultaneously several associated pattern pairs)}$$

This process is generally called storage or encoding.

Section - D

Q.8. Explain the following : (20)

(i) Seperability limitations.

(ii) Recall mode.

(iii) Initialization of weights.

Ans.(i) Separability limitations : A single layer perceptron consists of an input layer and an output layer. The activation function applied is hard-limiting function. An output unit will assume the value 1 if the sum of weighted input is greater than its threshold. *i.e.*,

$W_{ji}X_i > W_j$ where W_{ji} is weight from unit i to unit j.

X_i is the input from unit i and Δ_j is the threshold on unit j

Let there are 2 classes A and B. If $W_{ji}X_i > \Delta_j$ where $i = 1, 2, \ldots n$ forms a hyperplane, dividing the space in 2 halves.

Linear separability : When a linear hyperplane exists to place the instances of one class on one side and those of other class on the other side of plane. Then this is called Linear Separability. For example in case of OR gate we can draw a linear hyperplane to separate the input whose corresponding outputs are 1 on one side and the inputs whose corresponding outputs are 0 an other side of the plane.

Table : OR Gate

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	1

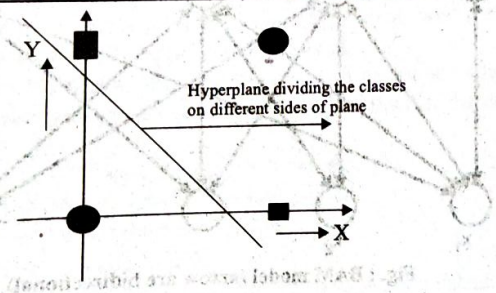


Fig. : The OR gate

But all the classification problems are not linear separable. Fox example the EX-OR problem is not linear separable because EX-OR is non equality gate.

Ans.(ii) Recall Mode : In the recall mode, the network is presented with the feature vector of a single sample. The output of the network determines the class to which the sample belongs. *Hetero-association* is related to two recall mechanisms:

(a) Nearest-neighbour recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented.

(b) Interpolative recall, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented. Yet another paradigm, which is a variant associative mapping is classification, *i.e.* when there is a fixed set of categories into which the input patterns are to be classified.

Ans.(iii) Initialization of Weights : The weights of the network to be trained are initialized at small random values. The initialization affects the complete solution. If all the weights start out with equal weight values and if the soultion requries that unequal weights be developed, the network may not be trained properly. artificial neural networks typically start out with randomized weights for all their neurons.

If training continues beyond a certain low error level, it will result in undersirable drift of weights. This casues error to increase and the quality of mapping by the network decreases. Therefore the choice of initial weights is however only one of several factors affecting training of error towards an acceptable error minimization.

Q.9. Give a complete description about self organizing networks. (20)

Ans. Self-organizing network : A Self-Organizing Network (SON) is an automation technology designed to make the planning, configuration, management, optimization and healing of mobile radio access networks simpler and faster. SON functionality and behavior has been defined and specified in generally accepted mobile industry recommendations produced by organizations such as 3GPP (3rd Generation Partnership Project) and the NGMN (Next Generation Mobile Networks).

SON has been codified within 3GPP Release 8 and subsequent specifications in a series of standards including 36.902, as well as public white papers outlining use cases from the NGMN. The first technology making use of SON features will be Long Term Evolution (LTE), but the technology has also been retro-fitted to older radio access technologies such as Universal Mobile

Telecommunications System (UMTS). The LTE specification inherently supports SON features like Automatic Neighbor Relation (ANR) detection, which is the 3GPP LTE Rel. 8 flagship feature.

Newly added base stations should be self-configured in line with a "plug-and-play" paradigm while all operational base stations will regularly self-optimize parameters and algorithmic behavior in response to observed network performance and radio conditions. Furthermore, self-healing mechanisms can be triggered to temporarily compensate for a detected equipment outage, while awaiting a more permanent solution.

Introduction of SON : Self-organizing Networks features are being introduced gradually with the arrival of new 4G systems in radio access networks, allowing for the impact of potential 'teething troubles' to be limited and gradually increasing confidence. Self-optimization mechanisms in mobile radio access networks can be seen to have some similarities to automated trading algorithms in financial markets. SON has also been retrofitted to existing 3G networks to help reduce cost and improve service reliability.

SON architectural types : Self-organizing networks are commonly divided into three major architectural types.

(i) Distributed SON : In this type of SON (D-SON), functions are distributed among the network elements at the edge of the network, typically the ENodeB elements. This implies a certain degree of localization of functionality and is normally supplied by the network equipment vendor manufacturing the radio cell.

(ii) Centralized SON : In centralized SON (C-SON), function is more typically concentrated closer to higher-order network nodes or the network OSS, to allow a broader overview of more edge elements and coordination of e.g. load across a wide geographic area. Due to the need to inter-work with cells supplied by different equipment vendors, C-SON systems are more typically supplied by 3rd parties.

(iii) Hybrid SON : Hybrid SON is a mix of centralized and distributed SON, combining elements of each in a hybrid solution.

SON sub-functions : Self-organizing network functionalities are commonly divided into three major sub-functional groups, each containing a wide range of decomposed use cases.

(i) Self-configuration functions : Self-configuration strives towards the "plug-and-play" paradigm in the way that new base stations shall automatically be configured and integrated into the network. This means both connectivity establishment, and download of configuration parameters are software. Self-configuration is typically supplied as part of the software delivery with each radio cell by equipment vendors. When a new base station is introduced into the network and powered on, it gets immediately recognized and registered by the network. The neighboring base stations then automatically adjust their technical parameters (such as emission power, antenna tilt, etc.) in order to provide the required coverage and capacity, and, in the same time, avoid the interference.

(ii) Self-optimization functions : Every base station contains hundreds of configuration parameters that control various aspects of the cell site. Each of these can be altered to change network behavior, based on observations of both the base station itself and measurements at the mobile station or handset. One of the first SON features establishes neighbor relations automatically (ANR) while others optimize random access parameters or mobility robustness in terms of handover oscillations. A very illustrative use case is the automatic switch-off of a percent of base stations during the night hours. The neighboring base station would then re-configure their parameters in order to keep the entire area covered by the signal. In case of a sudden growth in connectivity demand for any reason, the "sleeping" base stations "wake up" almost instantaneously. This mechanism leads to significant energy savings for operators.

(iii) Self-healing functions : When some nodes in the network become inoperative, self-healing mechanisms aim at reducing the impacts from the failure, for example by adjusting parameters and algorithms in adjacent cells so that other nodes can support the users that were supported by the failing node. In legacy networks, the failing base stations are at times hard to identify and a significant amount of time and resources is required to fix it. This function of SON

permits to spot such a failing base stations immediately in order to take further measures, and ensure no or insignificant degradation of service for the users.

Self-Protection functions : It is a proactive approach of a system for defending itself from the penetration of any unauthorised user in the system and from any active or passive attack. The main objectives of self-protection are to make the security of the system unbreakable and also make the data confidential and secure.

NEURAL NETWORKS

Dec - 2017
Paper Code:-CSE-407-F

Note : ***Attempt five questions in all, selecting one question from each Section. Question No. 1 is compulsory. All questions carry equal marks.***

Q.1.(a) What is Hebbian Learning Rule ? **(5)**

Ans. Hebbian Learning Rule : In Hebbian learning, weights between learning nodes are adjusted so that each weight bette represents the relationship between the nodes. Nodes which tend to be positive or negative at the same time will have strong positive weights while those which tend to be opposite will have strong negative weights. Nodes that are uncorrelated will have weights near zero. For example, if two nodes A and B are often simultaneously active, Hebbian learning will increase the connection strength between the two so that excitation of either one tends to cause excitation of the other. On the other hand, if nodes A and C were of opposite activations at all times, then Hebbian learning would gradually decrease the connection in between below zero so that an excited A or C would inhibit the other.

For this rule the learning signal is equal simple to the neuron's output

We have $$r = f(W_i^t x) \quad ...(i)$$

The increment ΔW_i of the weight vector becomes

$$\Delta W_i = c f(W_i^t x) x \quad ...(ii)$$

The single weight adjustment using the following increment :

$$\Delta W_{ij} = c f(W_i^t x_i) x_j \quad ...(iii)$$

This can be written as

$$\Delta Wi_j = c o_i x_j \text{ for } j = 1, 2, 3, ... n \quad ...(iv)$$

Hebbian learning has four features :

(1) First it is unsupervised;

(2) Second it is a local learning rule, meaning that it can be applied to a network in parallel;

(3) Third it is simple and therefore requires very little computation;

(4) Fourth it is biologically plausible.

Q.1.(b) What is Multi-Layer Forward Networks ? **(5)**

Ans. 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. Recall is instantaneous in this type of network (we will discuss this later in the section on the uses of ANNs). These networks can be used to realize complex input/output mappings.

This type of network (Fig.) 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.

Q.1.(c) What is Recurrent Auto Associative Memory ? **(5)**

Ans. In order to produce an improved association there is a need for suppressing the output noise at memory output in linear associative memory. This can be done by thresholding the output and by recycling of the output to input.

The repetitive process of recycling of the output to the input through the network can be performed by a Recurrent Neural Network (RNN).

RNN is similar to linear associative memory. But it has feedback is dynamical.

There are two modes of operation of updation.

(1) Under the asynchronous update mode, only neuron is allowed to compute or change state at a time and then all outputs are delayed by a time υ that is produced by unity delay element in the feedback loop.

(2) In synchronous mode

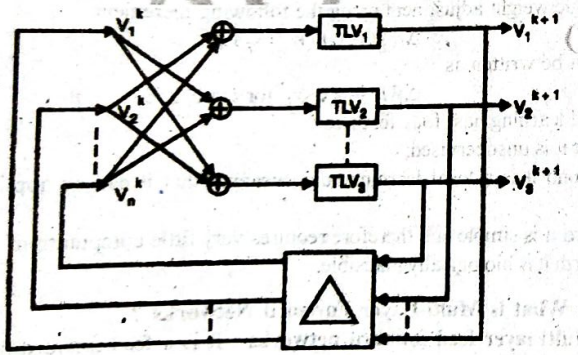


Fig. : Hop?eld model autoassociative memory (Recurrent autoassociative memory).

Explanation of figure dots represents the neurons, TLU threshold logic unit and these are used because associative memories are updated in discrete time.

$\Delta \rightarrow$ Unit delay in time.

Q.1.(d) What is clustering ? **(5)**

Ans. Clustering : Clustering is understood to be the grouping of similar objects and separating of dissimilar ones. The objective of clustering neural networks is to categorize or

cluster data. The classes must first be found from the correlations of an input data stream. Since the network actually deals with unlabeled data, the clustering should be followed by labeling clusters with appropriate category names or numbers. This process of providing the category of objects with a label is usually termed as calibration.

Section – A

Q.2.(a) Differentiate Biological Neurons and Artificial Neural Networks. **(10)**

Ans. Comparision between ANN and biological networks are as follows :

S.No.	ANN	Biological Networks
1	ANNs have usually been limited to 10000 units with hundreds of connections per unit.	The human brain is extremely large for a neural network containing 10^{11} neurons and there exists 10^{15} interconnections.
2	The memory has an organized behaviour and the data retrieved follows an ordered sequence.	The memory follows a pattern of distributed representation and data retrieval depends on retention capacity of the brain
3	Learning takes place by a formulated set of rules and hence the system is less faulty. The processing elements receive many signals and sum up the weighted inputs. These networks can be retrained in case of significant damage (i.e. loss of data).	Learning takes place arbitrarily and hence the system is bound for failure. Biological networks are fault tolerable and even in a traumatic loss other neurons can takes over the functions of damaged cells.
4	Redundancy can increase the reliability of the system, allowing it to function even when some of the neural units are destroyed.	Redundancy is used. When some cells die, the system appears to perform the same. It can counteract source of noise in biological systems.
5	The strength of neuron depends on whether It is active or inactive and has relatively simpler interconnections.	The strength of neuron depends on active chemicals present and connections are stronger or weaker as a result of structure layer rather than individual synapses.
6	ANNs focus on learning a single task.	Biological system have broad capabilities and can address many different types of tasks.
7	Artificial systems are slow to converge and usually requie hundreds or thousands of training presentations for learning to take place.	Biological systems have the property of being able to learn with as little as one training presentation. A face that is viewed once, can be recognized again.

Q.2.(b) Explain different models of ANNs. **(10)**

Ans. Models of Artificial Neural Networks :

(1) McCulloch-Pitts (MCP) Neuron Model : The early model of an artificial neuron

is introduced by Warren McCulloch and Walter Pitts in 1943. The MCP neuron

(i) Has a number of inputs I_1, ... I_n. These inputs represent the incoming signals received from the neuron's synapses. So I_i can either be 1, which corresponds to the presence of an incoming signal from the ith connection, or I_i is 0, which corresponds to the absence of a signal from the ith connection.

(ii) The *MCP* neuron produces an output y, which can either be 1, corresponding to the neuron sending a signal, or it can be 0, corresponding to the neuron remaining at rest. Mathematically an *MCP* neuron is a function which takes an n-tuple of 1's and 0's and produces a 1 or a 0. In order to quantity the influence each synapse has on the *MCP* neuron, we assign a weight w_i to each input I_i.

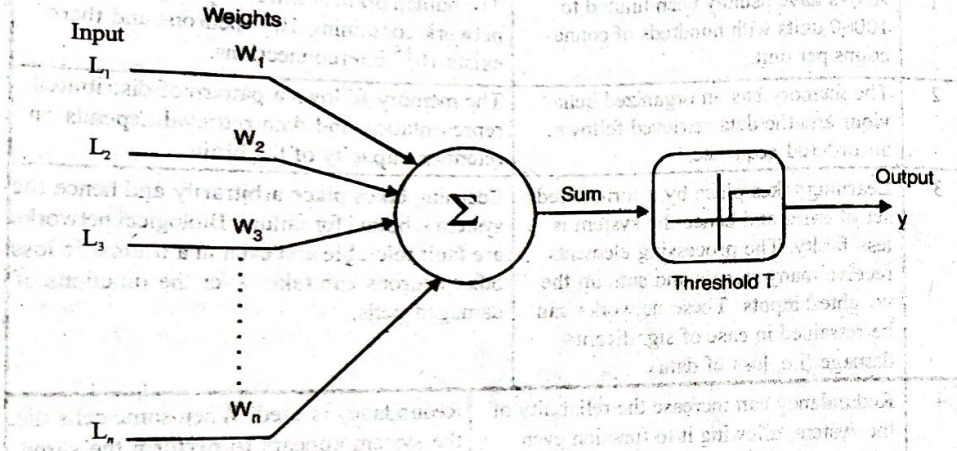


Fig.(a) : Symbolic Illustration of linear threshold gate

(iii) The weights are decimal numbers, and the size of the weight corresponds to the amount of influence the associated connection has on the *MCP* neuron. Positive weights correspond to excitatory synapses, and negative weights correspond to inhibitory synapses. Each input is multiplied by the corresponding weight, and these weighted inputs are then added together to produce

$$\text{Sum} = \sum_{i=1}^{N} I_i W_i \qquad ...(i)$$

$$y = f(\text{sum}) \qquad ...(ii)$$

where $w_i * I_i$ means w_i multiplied by I_i. This corresponds to the summing of the incoming signals that is assumed to be performed by a neuron. Finally, there is a threshold T, which is a decimal number that is compared to the weighted sum of the signals. This corresponds to the apparent threshold value that the sum of the received signals must exceed in order to cause a neuron to "fire". If $I_1 * w_1 + w_2 * I_2 + ...w_n * I_n >= T$, i.e. $w_1 * I_1 + w_2 * I_2 + ...w_n * I_n$ is greater than or equal to T, then the *MCP* neuron produces an output of 1. If $w_1 * I_1 + w_2 * I_2 + ...w_n * I_n$ is less than T, then the *MCP* neuron produces a 0. So an *MCP* neuron is completely determined by its weights and threshold. By choosing various combinations of weights and threshold, we can produce several different *MCP* neurons.

(2) Feedforward Network : Feedforward neural network is a biologically inspired classification algorithm. It consists of a (possibly large) number of simple neuron-like processing units, organized in layers. Every unit is a layer is connected with all the units in the previous layer. These connections are not all equal; each connection may have a different strength or weight. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called nodes. Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called feedforward neural networks.

Fig.(b) : Block diagram of Feedforward Network

Let us consider the architecture of feedforward architecture of m neuron receiving n inputs as shown in fig. Its inputs and output vectors are respectively

$$O = [O_1 \quad O_2 \quad .... \quad O_m]^t$$
$$X = [x_1 \quad x_2 \quad .... \quad x_n]^t \qquad ...(i)$$

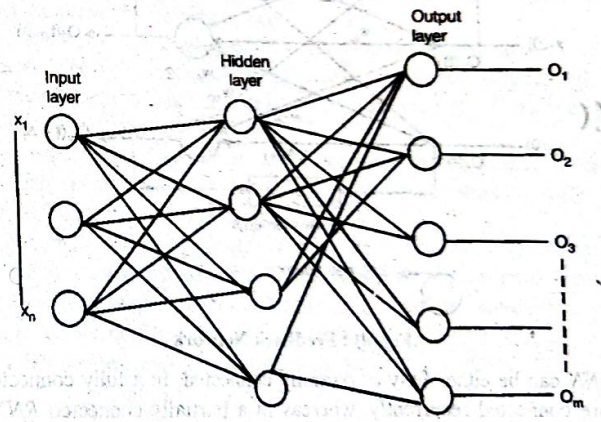


Fig.(c) : Feedforward Network

Weight w_{ij} connects the ith neuron with the jth output. Therefore the activation value for the ith neuron is

$$\text{net}_i = \sum_{j=1}^{n} W_{ij} x_j \qquad \text{for } i = 1, 2, ..., n \qquad ...(ii)$$

$$\text{Output is } O_i = f(W_i^t x) \qquad \text{for } i = 1, 2, ... m \qquad ...(iii)$$

where weight vector W_i contains the weighting leading towards the ith output node and is defined as

$$W_i = [w_{i1}, w_{i2}, \ldots, w_{in}]^t \quad \ldots(iv)$$

Introducing the nonlinear matrix operator Γ the mapping of input space x to output space O implemented by

$$O = \Gamma (WX) \quad \ldots(v)$$

where W is called weight matrix.

Since there is no time delay between the input x and the output o, therefore we can write

$$O(t) = \Gamma [Wx(t)]$$

Feedback Network/Recurrent Network : Recurrent neural netwroks (*RNN*) have a closed loop in the network topology. They are developed to deal with the time varying or time-lagged patterns and are usable for the problems where the dynamics of the considered process is complex and the measured data is noisy. Specific groups of the units get the feedback signals from the previous time steps and these units are called context unit.

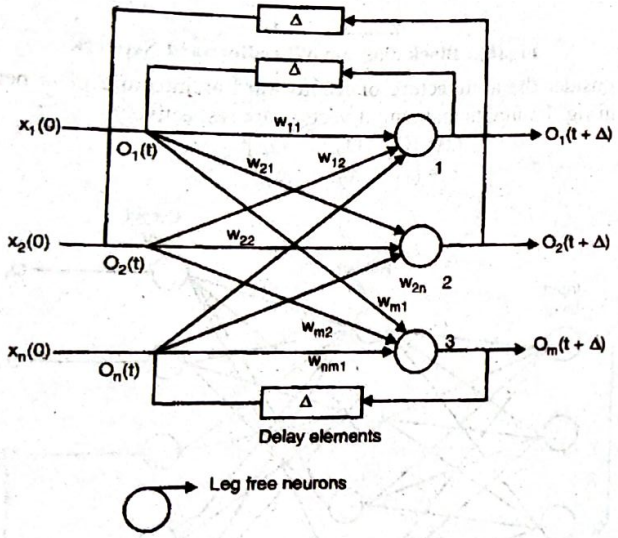


Fig.(d) : Feedback Network

The *RNN* can be either fully or partially connected. In a fully connected *RNN* all the hidden units are connected recurrently, whereas in a partially connected *RNN* the recurrent connections are omitted paratially. Examples of recurrent neural networks are Hopfield networks, Regressive networks, Jordan-Elman networks, and Brain-State-In-a-Box (BSB) networks.

Q.3. What is ANN ? What are the different Learning Rules used in ANN ? Explain in detail. **(20)**

Ans. An artificial Neural Network (ANN) is an information-processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. It is composed of a large number of highly interconnected processing elements(neurons) working in parallel to solve specific problems. The weights on the connections encode the knowledge of a

network. Each neuron has a local memory and the output of each neuron depends upon only the input signal arriving at the neuron and value in neuron's memory. ANNs, like people, learn by example.

Learning rule or **Learning process** is a method or a mathematical logic. It improves the Artificial Neural Network's performance and applies this rule over the network. Thus learning rules updates the weights and bias levels of a network when a network simulates in a specific data environment.

Applying learning rule is an iterative process. It helps a neural network to learn from the existing conditions and improve its performance.

Let us see different learning rules in the Neural network:

- **Hebbian learning rule** – It identifies, how to modify the weights of nodes of a network.
- **Perceptron learning rule** – Network starts its learning by assigning a random value to each weight.
- **Delta learning rule** – Modification in sympatric weight of a node is equal to the multiplication of error and the input.
- **Correlation learning rule** – The correlation rule is the supervised learning.
- **Outstar learning rule** – We can use it when it assumes that nodes or neurons in a network arranged in a layer.

Hebbian Learning Rule : The Hebbian rule was the first learning rule. In 1949 *Donald Hebb* developed it as learning algorithm of the unsupervised neural network. We can use it to identify how to improve the weights of nodes of a network.

The Hebb learning rule assumes that – If two neighbor neurons activated and deactivated at the same time. Then the weight connecting these neurons should increase. For neurons operating in the opposite phase, the weight between them should decrease. If there is no signal correlation, the weight should not change.

When inputs of both the nodes are either positive or negative, then a strong positive weight exists between the nodes. If the input of a node is positive and negative for other, a strong negative weight exists between the nodes.

At the start, values of all weights are set to zero. This learning rule can be used0 for both soft- and hard-activation functions. Since desired responses of neurons are not used in the learning procedure, this is the unsupervised learning rule. The absolute values of the weights are usually proportional to the learning time, which is undesired.

The Hebbian learning rule describes the formula as follows:

$$w_{ij} = x_i * x_j$$

Perceptron Learning Rule : As you know, each connection in a neural network has an associated weight, which changes in the course of learning. According to it, an example of supervised learning, the network starts its learning by assigning a random value to each weight. Calculate the output value on the basis of a set of records for which we can know the expected output value. This is the learning sample that indicates the entire definition. As a result, it is called a learning sample.

The network then compares the calculated output value with the expected value. Next calculates an error function ", which can be the sum of squares of the errors occurring for each

individual in the learning sample.

Computed as follows: $\sum_i \sum_j (E_{ij} - O_{ij})^2$

Perform the first summation on the individuals of the learning set, and perform the second summation on the output units. Eij and Oij are the expected and obtained values of the jth unit for the ith individual.

The network then adjusts the weights of the different units, checking each time to see if the error function has increased or decreased. As in a conventional regression, this is a matter of solving a problem of least squares.

Since assigning the weights of nodes according to users, it is an example of supervised learning.

Delta Learning Rule : Developed by *Widrow* and *Hoff*, the delta rule, is one of the most common learning rules. It depends on supervised learning.

This rule states that the modification in sympatric weight of a node is equal to the multiplication of error and the input.

In Mathematical form the delta rule is as follows:

$$\Delta W = \eta(t - y)x_i$$

For a given input vector, compare the output vector is the correct answer. If the difference is zero, no learning takes place; otherwise, adjusts its weights to reduce this difference. The change in weight from u_i to u_j is: $dw_{ij} = r^* a_i * e_j$.

where r is the learning rate, a_i represents the activation of u_i and e_j is the difference between the expected output and the actual output of u_j. If the set of input patterns form an independent set then learn arbitrary associations using the delta rule.

It has seen that for networks with linear activation functions and with no hidden units. The error squared vs. the weight graph is a paraboloid in n-space. Since the proportionality constant is negative, the graph of such a function is concave upward and has the least value. The vertex of this paraboloid represents the point where it reduces the error. The weight vector corresponding to this point is then the ideal weight vector.

We can use the delta learning rule with both single output unit and several output units.

While applying the delta rule assume that the error can be directly measured.

The aim of applying the delta rule is to reduce the difference between the actual and expected output that is the error.

Correlation Learning Rule : The correlation learning rule based on a similar principle as the Hebbian learning rule. It assumes that weights between responding neurons should be more positive, and weights between neurons with opposite reaction should be more negative. Contrary to the Hebbian rule, the correlation rule is the supervised learning. Instead of an actual the response, o_j, the desired response, d_j, uses for the weight-change calculation.

In Mathematical form the correlation learning rule is as follows:

$$\Delta W_{ij} = \eta x_i d_j$$

where d_j is the desired value of output signal. This training algorithm usually starts with the initialization of weights to zero.

Since assigning the desired weight by users, the correlation learning rule is an example of supervised learning.

Out Star Learning Rule : We use the Out Star Learning Rule when we assume that nodes or neurons in a network arranged in a layer. Here the weights connected to a certain node should be equal to the desired outputs for the neurons connected through those weights. The out start rule produces the desired response t for the layer of n nodes.

Apply this type of learning for all nodes in a particular layer. Update the weights for nodes are as in Kohonen neural networks.

In Mathematical form, express the out star learning as follows:

$$W_{ik} = \begin{cases} \eta(y_k - w_{jk}) & \text{if node } j \text{ wins the competition} \\ 0 & \text{if node } j \text{ losses the competition} \end{cases}$$

This is supervised training procedure because desired outputs must be known.

Section – B

Q.4.(a) Explain the single layer continuous perceptron training algorithm for linearly separable classification. **(10)**

Ans. Algorithm of single continuous perceptron training :

Given, P training pairs

$\{x_1, d_1, x_2, d_2, ..., x_p d_p\}$,

Augmented input vectors are used

$$y_i = \begin{bmatrix} x_i \\ 1 \end{bmatrix} \text{ for i} = 1, 2, ..., P$$

where x_i is $(n \times 1)$, d_i is (1×1), $i = 1, 2, ..., P$

p denotes step counter within training cycle and k denotes the training step.

(i) Choose $\eta > 0$, l(steepress constant) = 1, $E_{max} > 0$.

(ii) Weight are initialized at w at small random values, counters and error are initialized.

$$k \leftarrow 1, p \leftarrow 1, E \leftarrow 0.$$

(iii) Training starts → input y is presented and output is computed i.e.

$$y \leftarrow y_p, d \leftarrow dp, o \leftarrow f(w^t y)$$

(iv) Weights are updated : $w^{k+1} \leftarrow w^k + \frac{1}{2}\eta(d^k - o^k)(1 - o^{k^2})y$

(v) Error E is computed $E^{k+1} \leftarrow \frac{1}{2}(d^k - o^k)^2 + E^k$.

(vi) If $p < P$, then $p \leftarrow p + 1, k \rightarrow k + 1$ and repeat step 3, otherwise go to step 7.

(vii) Training cycle is completed. If $E < E_{max}$ terminate the session.

If $E \geq E_{max}$ then $E \leftarrow 0, p \leftarrow 1$, start new training cycle and go to step 3.

Q.4.(b) In which manner multilayer Perceptron Models differ from Single Layer Perceptron Model ? **(10)**

Ans. Single layer Perceptron : A single layer perceptron consists of an input layer and an output layer. The activation function applied is hard-limiting function. An output unit will assume the value 1 if the sum of weighted inputs is greater than its threshold i.e.

$\Sigma W_{ji} X_i > \theta_j$

where W_{ji} is weight from i to unit j.

X_i is the input from unit i and θ_j is the threshold on unit j.

Let there are 2 classes A and B. If $\Sigma W_{ji} X_i > \theta_j$ then object will be classified as Class A otherwise Class B.

Suppose there are n inputs then the equation $\Sigma W_{ji} X_i > \theta_j$ where $i = 1, 2, \ldots n$ form a hyperplane, dividing the space in 2 halves.

Multi layer perceptron : A MLP (Multi layer perceptron) is a feedforward artificial neural network which is a modification of the linear perceptron using three of more layers of neurons with more power of distinguishing non-linear data. In MLP each neuron uses a non-linear activation function which is used to develop action potential for firing of neurons. The activation function used must be differentiable and normalizable e.g., tanh ( ) and sigmoids. Learning in MLP is done with the help of a teacher i.e., supervised learning. In this learning of perceptrons connection weights are updated after every process based on the error produced by the difference of desired response and actual response.

Q.5. Write short note on :

(a) Delta Learning Rule. **(10)**

(b) Error Back-Propagation Training algorithm. **(10)**

Ans. (a) Delta learning rule : Delta learning rule is only valid for continuous activation functions. This rule is used in the supervised training mode.

The learning signal for this rule is called delta and is defined as follows :

$$r \cong \left[d_i - f(w_i^t x)\right] f'(w_i^t x) \qquad \ldots(i)$$

The term $f'(w_i^t x)$ is the derivative of the activation function f (net) computed for net $= W_{ix}$.

The explanation of the delta learning rule is shown in diagram. This learning rule can be readily derived from the condition of least squared error between o_i and d_i. Calculating the gradient vector with respect to W_i of the squared error defined as

$$E \cong \frac{1}{2}(d_i - o_i)^2 \qquad \ldots(ii)$$

which is equivalent to

$$E \cong \frac{1}{2}\left[d_i - f(w_i^t x)\right]^2 \qquad \ldots(iii)$$

Error gradient vector value

$$\nabla E = -[d_i - f(w_i^t x)] f'(w_i^t x) x \qquad \ldots(iv)$$

The components of the gradient vector are

$$\frac{\partial E}{\partial w_{ij}} = -\left[d_i - f(w_j^t x)\right] f'(w_i^t x) x_j \qquad \ldots(v)$$

for $j = 1, 2, \ldots, n$

Since the minimization of the error requires the weight changes to be in the negative gradient direction, we take

$$\Delta W_i = -\eta \nabla E \qquad \ldots(vi)$$

where η is a +ve equation from (v) and (vi)

$$\Delta W_i = \eta\left[d_i - f(w_i^t x)\right] f'(w_i^t x) x \qquad \ldots(vii)$$

or, for the single weight the adjustment becomes

$$\Delta w_{ij} = \eta(d_i - o_i) f'(\text{net}_i)\, x_j \qquad \ldots(viii)$$

for $j = 1, 2, \ldots, n$

This delta rule was introduced by McClelland and Rumelhart in 1986. This rule parallels the discrete perceptron training rule.

This rule is also called as continuous perceptron training rule.

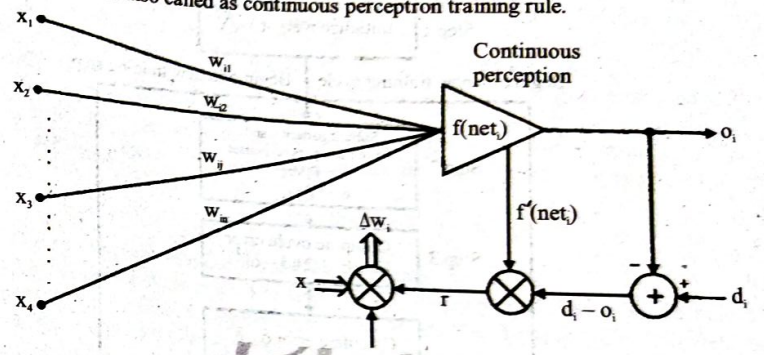


Fig.

Ans.(b)Error Back-Propagation Training algorithm : The layer network is mapping the input vector z into the output vector o as follows :

$$o = N(z)$$

where N denotes a composite nonlinear matrix operator. For the two-layer net the mapping $z \rightarrow o$ can be represented as a mapping within a mapping, or

$$o = G[W\Gamma[Vz]]$$

$$\Gamma[Vz] = y$$

and it related to the hidden layer mapping $z \rightarrow y$. Note that the right arrows denote mapping of one space into another. Each of the mapping is performed by a single-layer of the layered network. The operator Γ is a nonlinear diagonal operator.

Algorithm :

Given are P training pairs

$$\{z_1, d_1, z_2, d_2, \ldots\ldots, z_p, d_p,\},$$

Where z_i is $(l \times 1)$, d_i is $(K \times 1)$, and $i = 1, 2, \ldots P$. Note that the lth component of each x_i is of value – 1 since input vectors have been augmented. Size $J-1$ of the hidden layer having outputs y is selected. Note that the Jth component of y is of value – 1, since hidden layer outputs have also been augmented; y is $(J \times 1)$ and o is $(K \times 1)$.

Step 1 : $\eta > 0$. E_{max} chosen

Weights W and Y are initialized at small random values; W is $(K \times J)$, V is $(J \times I)$.

$$q \leftarrow 1,\ p \leftarrow 1, E \leftarrow 0$$

Step 2 : Training step starts here.
Input is presented and the layers output computed.

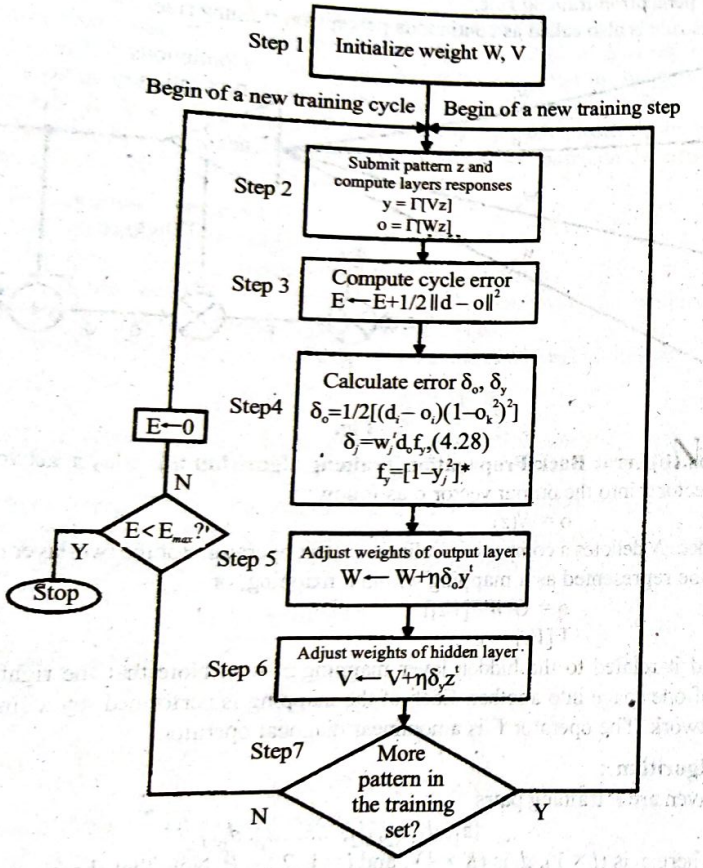
$$z \leftarrow z_p,\ d \leftarrow d_p$$

$$y_j \leftarrow f(v_j^t z),\ \text{for } j = 1, 2, \ldots, J$$

where v_j, a column vector is the Jth row of V and

$$o_k \leftarrow f(w_k^t y),\ \text{for } k = 1, 2, \ldots, K$$

where w_k, a column vector, is the kth row of W.



Step 3 : Error value is computed

$$E \leftarrow \frac{1}{2}(d_k - o_k)^2 + E,\ \text{for } k = 1, 2, \ldots, K$$

Step 4 : Error signal vectors δ_0 and δ_y of the both layers are computed.
Vector δ_0 is $(K \times 1)$, δ_y is $(J \times 1)$.

The error signal terms of the output layer in this step are

$$\delta_{ok} = \frac{1}{2}(d_k - o_k)(1 - o_k^2),\ \text{for } k = 1, 2, \ldots, K$$

The error signal terms of the hidden layer in this step are

$$\delta_{yi} = \frac{1}{2}(1 - y_j^2)\sum_{k=1}^{k} \delta_{ok} w_{kj},\ \text{for } j = 1, 2, \ldots, J$$

Step 5 : Output layer weights are adjusted :

$$w_{kj} \leftarrow w_{kj} + \eta\delta_{ok} y_j,\ \text{for } k = 1, 2, \ldots K \text{ and } j = 1, 2, \ldots, J$$

Step 6 : Hidden layer weights are adjusted

$$v_{ji} \leftarrow v_{ji} + \eta\delta_{yj} z_i,\ \text{for } j = 1, 2, \ldots J \text{ and } i = 1, 2, \ldots, I$$

Step 7 : If $p < P$ then $p \leftarrow p + 1$, $q \leftarrow q + 1$, and go to Step 2; otherwise, go to Step 8.
Step 8 : The training cycle is completed.
For $E < E_{max}$ terminate the training session. Output weights W, V, q and E.
If $E > E_{max}$ then $E \leftarrow 0$, $p \leftarrow 1$, and initiate the new training cycle by going to Step 2.

Section – C

Q.6.(a) Explain Hopfield Network in detail. **(10)**

Ans. Hopfield Network : The Hopfield model was proposed by John Hopfield of the California Institute of Technology during the early 1980s. The dynamics of the Hopfield model is different from that of the linear associator model in that it computes its output recursively in time until the system becomes stable. Below is a Hopfield model with six units, where each node is connected to every other node in the network.

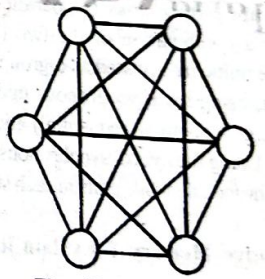


Fig. : Hopfield Model

Unlike the linear associator model which consists of two layers of processing units, one serving as the inputs layer while the other as the output layer, the Hopfield model consists of a single layer of processing elements where each unit is connected to every other unit in the network other than itself.

The connection weight matrix W of this type of network is square and symmetric, i.e., $w_{ij} = w_{ji}$ for $i, j = 1, 2 \ldots, m$. Each unit has an extra external input I_i. This extra input leads to a modification in the computation of the net input to the units :

$$\text{input}_j = \alpha\sum_{i=1}^{m} x_i w_{ij} + I_j$$

where $j = 1, 2, ..., m$.

Unlike the linear associator, the units in the Hopfield model act as both input and output units. But just like the linear associator, a single associated pattern pair is stored by computing the weight matrix as follows :

$$W_k = X_k$$
$$TY_k$$

where $Y_k = X_k$ to store p different associated pattern pairs. Since the Hopfield model is an autoassociative memory model, patterns, rather than associated pattern pairs, are stored in memory.

After encoding, the network can be used for decoding. Decoding in the Hopfield model is achieved by a collective and recursive relaxation search for a stored pattern given an initial stimulus pattern. Given an input pattern X, decoding is accomplished by computing the net input to the units and determining the output of those units using the output function to produce the pattern X'. The pattern X' is then fed back to the units as an input pattern to produce the pattern X''. The pattern X'' is again fedback to the units to produce the pattern X'''. The process is repeated until the network stabilizes on a stored pattern where further computations do not change the output of the units.

If the input pattern X is an incomplete pattern or if it contain some distortions, the stored pattern to which the network stabilizes is typically one that is most similar to X without the distortions. This feature is called *pattern completion* and is very useful in many image processing applications.

During decoding, there are several schemes that can be useed to update the output of the units. The updating schemes are *synchronous* (or parallel as termed in some literatures), *asynchronous* (or sequential), or a combination of the two (hybrid). Using the synchronous updating scheme, the output of the units are updated as a group prior to feeding the output back to the network. On the other hand, using the asynchronous updating scheme, the output of the units are updated in some order (e.g. random or sequential) and the output are then fed back to the network after each unit update. Using the hybrid synchronous-asynchronous undating scheme, subgroups of units are updated synchronously while units in each subgroup updated asynchronously.

Q.6.(b) What is Associative Memory ? Explain its various type in detail with suitable example. **(10)**

Ans. The block diagram of associative memory performing an associative mapping of an input vector X into an output vector V.

$$V = M[X] \quad ...(i)$$

$\therefore$ where M = denotes a general non-linear diagonal operator.

The different for each type of memory model used. For dynamic memories, M also involves time variable.

The computation of M is done with the help of "recording or storage algorithm". The mapping of V to X is called retrievel. The storage algorithm depends on whether an auto associative or a hetero-associative type of memory is designed.

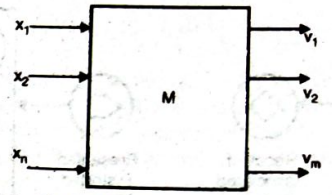


Fig.(a) : Block diagram of an associative memory

Assumption :

1. Certain prototype vector are stored in memory in such a way that once a key input has been applied, an output produced by the memory and associated with the key is the memory response.

2. Total stored pairs in the memory is P and the association are defined as

$$X^{(i)} \rightarrow V^{(i)} \quad \text{for } i = 1, 2, ..., P \quad ...(ii)$$

and $$V^{(i)} \neq X^{(i)} \quad \text{for } i = 1, 2, ..., P,$$

The network can be termed as hetero associative memory.

$\because$ in hetero associative memory, there is associations between pairs of 2ordered sets of vector $\{X^{(1)}, X^{(2)}, ..., X^{(P)},\}$ and $\{V^{(1)}, V^{(2)}, ..., V^{(P)},\}$ as explained earlier.

If $$X^{(i)} \rightarrow V^{(i)}$$
$$V^{(i)} = X^{(i)} \quad \text{for } i = 1, 2, ..., P,$$

Then memory is called auto associative memory.

In auto associative memory, there is association within only one set which is $\{X^{(1)}, X^{(2)}, ..., X^{(P)},\}$.

The realistic application of an auto associative mapping would be recovery of an undistorted prototype vector in response to the distorted prototype key vector.

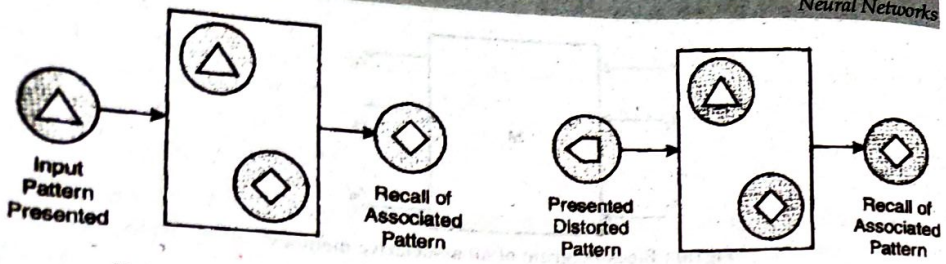
Associative Memory Classes : There are two classes of associative memory :

(i) Auto associative memory

(ii) Hetero associative memory.

(i) Auto Associative Memory : A n auto-associative memory, also know as auto-associative correlator, is used to retrieve a previously stored pattern that most closely resembles the current pattern.

(ii) Hetero Associative Memory : A hetero-associative memory, also known as hetero-associative correlator, is used to rerieve pattern in general, different from the input pattern not only in content but possibly also different in type and format.



Working of Associative Memories : An associative memory is a storehouse of associated patterns which are encoded in some form.

When the storehouse is triggered or excited with a pattern, then the associated pattern pair is recalled or appears at the output. The input could be an exact or distored or partial representation of a stored pattern.

Fig. shows in illustrates the working of an associated memory.

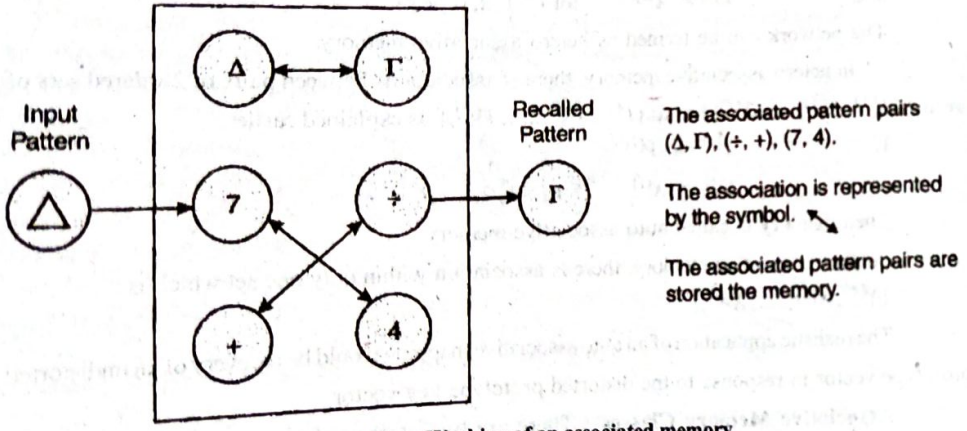


Fig. : Working of an associated memory.

When the memory is triggered with an input pattern say u then the associated pattern æ is retrieved automattically.

Q.7. Explain the Association, Encoding and Decoding and Stability consideration for Bidirectional Associative Memory in detail. **(20)**

Ans. Bidirectional Associative memory : Bidirectional associative memory is a heteroassociative, contant-addressable memory consisting of two layers. It uses the forward and background information flow to produce an associative search for stored stimulus-response association (kosko 1987,1988). Consider that stored in the memory are p vector association pairs known as

$$\left\{\left(a^{(1)},b^{(1)}\right),\left(a^{(2)},b^{(2)}\right),.....,\left(a^{(p)},b^{(p)}\right)\right\} \quad ...(i)$$

When the memory neurons are activated, the network evolves to a stable state of two pattern reverberation, each pattern at output of one layer. The stable reverberation corresponds to a local energy minimum. The network's dynamics involves two layers of interaction. Becuase the memory processes information in time and involves bidirectional data flow, it differs in principle from a linear associator, although both networks are used to store assocation pairs. It also differs from the recurrent autoassociative memory in its update mode.

Association Encoding and Decoding : The coding of information (i) into the bidirectional associative memory is done using the customary outer product rule, or by adding p cross-correlation matrices. The formula for the weight matrix is

$$W = \sum_{i=1}^{p} a^{(i)} b^{(i)t} \quad ...(ii)$$

where $a^{(i)}$ and $b^{(i)}$ are bipolar binary vector, which are membars of the i'th pair.

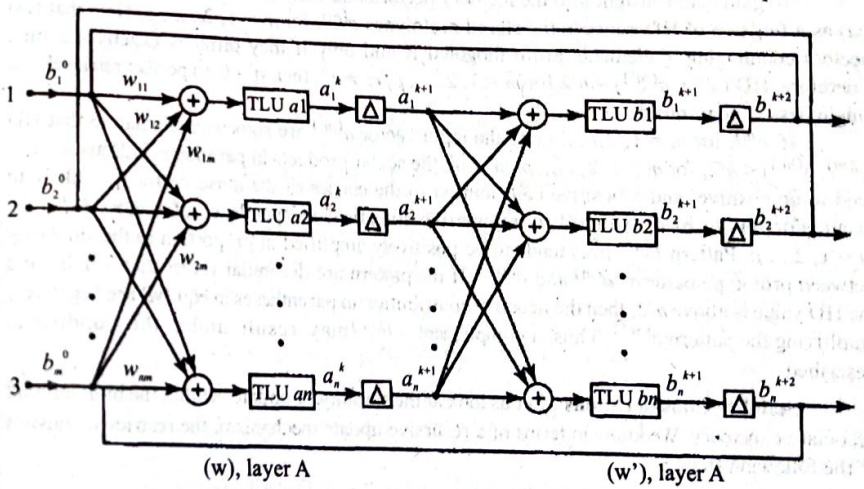


Fig. : Discrete-time bidirectional associative memory expanded diagram.

yielding the following weight values :

$$w_{ij} = \sum_{m=1}^{p} a_i^{(m)} b_j^{(m)} \quad ...(iii)$$

Suppose one of the stored pattern, $a(m')$, is presented to the memory. The retrieval proceeds as follows

$$b = \Gamma\left[\sum_{m=1}^{p}\left(b^{(m)}a^{(m)t}\right)a^{(m')}\right] \quad ...(iv)$$

which further reduces to

$$b = \Gamma\left[nb^{(m')} + \sum_{m \neq m'}^{p} b^{(m)} a^{(m)t} a^{(m')} \right] \quad ...(v)$$

The net$_b$ vector inside brackets in Equation (v) contains a signle term $nb^{(m')}$ additive with the noise term η of value

$$\eta = \sum_{m \neq m'}^{p} b^{(m)} \left(a^{(m)t} a^{(m')} \right) \quad ...(vi)$$

Assuming temporarily the orthogonality of stored patterns $a^{(m)}$, for $m = 1.\ 2, \ldots, p$, the noise the term η reduces to zero. Therefore, immediate stabilization and exact association $b = b(m')$ occurs within only a single pass though layer B. If the input vector is a distorted version of pattern $a^{(m')}$, the stabilization at $b^{(m')}$ is not imminent, however, and depends on many factors such as the HD between the key vector and prototype vectors, as well as on the orthogonality or HD between vectors $b^{(i)}$, for $i = 1, 2, \ldots, p$.

To gain better insight into the memory performance, let us look at the noise termη as in (*vi*) as a function of HD between the stored prototypes $a^{(m)}$, for $m = 1, 2, \ldots, p$. Note that two vectors containging $\pm$ elements are orthogonal if and only if they differ is exactly $n/2$ bits. Therefore, HD $(a^{(m)}, a^{(m')}) = n/2$ for $m = 1, 2, \ldots, p$, $m \neq m'$, then $\eta = 0$ an perfect retrievel in a single pass is guaranteed.

If $a^{(m)}$, for $m = 1, 2, \ldots, p$ and the input vector $a^{(m')}$ are somewhat similar so that HD $(a^{(m)}, a^{(m')}) < n/2$, for $m = 1, 2, \ldots, p$, $m \neq m'$, the scalar products in paratheses in Equation (*vi*) tend to be positive, and a positive contribution to the entries of the noise vector η is likely to occur. For this to hold, we need to assume the statistical independence of vectors $b^{(m)}$, for $m = 1, 2, \ldots p$. Pattern $b^{(m')}$ thus tends to be positively amplified in proportion to the similarity between prototype patterns $a^{(m)}$ and $a^{(m')}$. If the pattern are dissimilar rather than similar and the HD value is above $n/2$, then the negative contribution in parentheses in equ.(*vi*) are negatively amplifying the pattern $b^{(m')}$. Thus, a complement $-b^{(m')}$ may result under the conditions described.

Stability Considerations : Let us look at the stability of updates within the bidirectional accociative memory. We know in terms of a recursive update mechanism, the restrieval consists of the following steps :

$$\begin{aligned}
&\text{First Forward Pass :} && a^1 = \Gamma\left[Wb^0 \right] \\
&\text{First Backward Pass :} && b^2 = \Gamma\left[W^t a^1 \right] \\
&\text{Second Forward Pass :} && a^3 = \Gamma\left[Wb^2 \right] \\
& && \vdots \\
&k/2\text{'thBackward Pass :} && b^k = \Gamma\left[W^t a^{k-1} \right]
\end{aligned} \quad ...(vii)$$

As the updates in (vii) continue and the memory comes to its equilibrium at the k'th step, we have $a^k \rightarrow b^{k+1} \rightarrow a^{k+2}$, and $a^{k+2} = a^k$. In such a case, the memory is said to be

bidirectionally stable. This corresponds to the energy function reaching one of its minima after which any further decrease of its value is impossible. Let us propose the energy fucntion for minimization by this system in transition as

$$E(a,b) \overset{\Delta}{=} -\frac{1}{2} a^t W b - \frac{1}{2} b^t W^t a \quad ...(viii)$$

The reader may easily verify that this expression reduces to

$$E(a,\ b) = -a^t W b \quad ...(ix)$$

Let us evaluate the energy changes during a single pattern recall. The summary of thresholding bit updates for the outputs of layer A can be

$$\underset{i=1,2,\ldots,n}{\overset{\Delta a_i}{}} = \begin{cases} 2 & > 0 \\ 0 \quad \text{for} \quad \sum_{j=1}^{m} w_{ij} b_j & = 0 \\ -2 & < 0 \end{cases} \quad ...[(x)a]$$

and for the outputs of layer B they result as

$$\underset{j=1,2,\ldots,n}{\overset{\Delta b_j}{}} = \begin{cases} 2 & > 0 \\ 0 \quad \text{for} \quad \sum_{i=1}^{m} w_{ij} a_i & = 0 \\ -2 & < 0 \end{cases} \quad ...[(x)b]$$

The gradients of energy (*ix*) with respect to a and b can be computed, respectively, as

$$\nabla_a E(a,b) = -Wb \quad ...[(xi)a]$$

$$\nabla_b E(a,b) = -W^t a \quad ...[(xi)b]$$

The bitwise update expression (*x*) translate into the following energy changes due to the single bit increments Δa_i and Δb_j :

$$\Delta E a_i(a,b) = -\left(\sum_{j=1}^{m} w_{ij} b_j \right) \Delta a_i, \text{ for } i = 1,2, \ldots, n \quad ...[(xii)a]$$

$$\Delta E b_j(a,b) = -\left(\sum_{i=1}^{m} w_{ij} a_i \right) \Delta b_j, \text{ for } j = 1,2, \ldots, m \quad ...[(xii)b]$$

Inspecting the right sides of equations (*xii*) and comparing them with the ordinary update rules as in (*x*) lead to be consculsion that $\Delta E \leq 0$. As with recurrent autoassociative memory, the energy changes are nonpositive. Since E is a bounded function from below according to the following inequality :

$$E(a,\ b) \geq \sum_{i=1}^{n} \sum_{j=1}^{m} \left| w_{ij} \right|$$

then the memory converages to a stable point. The point is a local minimum of the energy function, and the memory is said to be bidirectionally stable. Moreover, no restrictions exits regarding the choice of matrix W, so any arbitrary real $n \times m$ matrix will result in bidirectionally stable memory.

Section – D

Q.8. Explain UN supervised learing of clusters in detail. **(20)**

Ans. Unsupervised learning of clusters : Learning is based on clustering of input data. No prior knowledge is assumed to be available regarding an input membership in a particular class. Rather, gradually detected characteristics and a history of training will be used to assist the network in defining classes and possible boundaries between them.

Clustering : Clustering is understood to be the grouping of similar objects and separating of dissimilar ones. The objective of clustering neural networks is to categorize or cluster data. The classes must first be found from the correlations of an input data stream. Since the network actually deals with unlabeled data, the clustering should be followed by labeling clusters with appropriate category names or numbers. This process of providing the category of objects with a label is usually termed as calibration.

Although the algorithm won't have names to assign to these clusters, it can produce them and then use those clusters to assign new examples into one or the other of the clusters. This is a data-driven approach that can work well when there is sufficient data; for instance, social information filtering algorithms, such as those that Amazon.com use to recommend books, are based on the principle of finding similar groups of people and then assigning new users to groups. In some cases, such as with social information filtering, the information about other members of a cluster (such as what books they read) can be sufficient for the algorithm to produce meaningful results. In other cases, it may be the case that the clusters are merely a useful tool for a human analyst. Unfortunately, even unsupervised learning suffers from the problem of overfitting the training data. There's no silver bullet to avoiding the problem because any algorithm that can learn from its inputs needs to be quite powerful.

Suppose we are given a set of patterns without any information as to the number of classes that may be present in the set. The clustering problem in such a case is that of identifying the number of classes according to a certain criterion, and of assigning the membership of the patterns in these classes. The clustering technique presented below :

Assumptions : The number of classes is known a priori. The pattern set $\{x_1, x_2, \ldots\ldots, x_n\}$ is submitted to the input to determine decision functions required to identify possible clusters. Since no information is available from the teacher as far as the desired classifier's responses, we will use the similarity of incoming patterns as the criterion for clustering. To define a cluster, we need to establish a basis for assigning patterns to the domain of particular cluster.

How we will assign a pattern to domain of particular cluster? There are 2 rules for this.

(1) The most common similarity rule is to find the Euclidean distance between 2 patterns x for x_i defined as

$$|X - X_i| = [(X - X_i)^t (X - X_i)]^{1/2} \quad \ldots(i)$$

Smaller the distance, closer the patterns. Using (i) the distance between all the pairs are computed. Now how to distinguish the clusters? A distance T can then be chosen to discriminate clusters. The value T is understood as the maximum distance between patterns within a single

cluster. Fig.(a) shows an example of two clusters with a T value chosen to be greater than the typical within-cluster distance but smaller than the between-cluster distance.

Let there are 2 patterns X_1 and X_2 and we have to find which pattern is similar to pattern X. Then using (1) distance is calculated. The value for which the distance is small, closer the pattern is with x.

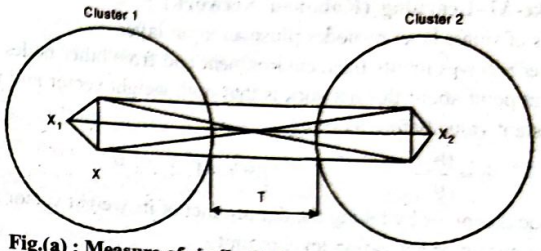


Fig.(a) : Measure of similarity between 2 clusters using distance

(2) Another similarity rule is the cosine of the angle between x and x_i:

$$\cos \alpha = \frac{(X^t X_i)}{\|X\| \cdot \|X_i\|} \quad \ldots(ii)$$

This rule is particularly useful when clusters develop along certain principal and different axes as shown in fig.(b). For $\cos \alpha_2 < \cos \alpha_1$ pattern x is more similar to x_2 than to x_1. It would thus be natural to group it with the second of the two apparent clusters. To facilitate this decision, the threshold angle α_t can be chosen to define the minimum angular cluster distance. It should be noted, however, that the measure defined in equation (ii) should be used according to certain additional qualifications. If the angular similarity criterion equation (ii) is to be efficient, vectors x_1, x_2 and x should be of comparable, or better, identical lengths.

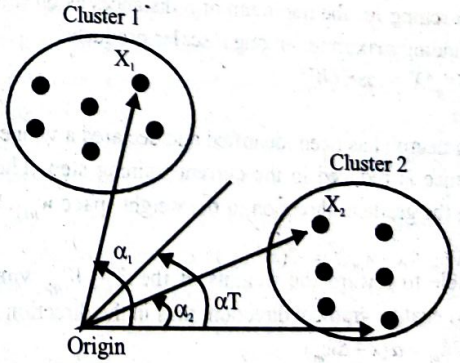


Fig.(b) : Measure of similarity between 2 clusters using normalized scalar product

Q.9.(a) What is Learning ? Explain. winner-Take-All Learning. **(10)**

Ans. Learning means how to adapt the changes so that system will work more effectively and efficiently. Different researchers give different statements about learning.

Marvin Minsky 1986 – "Learning is making useful changes in the working of our mind."

Ryszard Michalski 1986 – "Learning is constructing or modifying representations of what is being experienced."

Winner-Take-All-Learning (Kohonen Network) :

(i) It consists of single layer of nodes pluse an input layer.

(ii) Each layer receives inputs from environment and from other nodes within the layer.

(iii) Important point about this network is that both weight vector and input vectors are normalized to a constant value before learning

$$\hat{w}_i = \frac{(W_i)}{|W_i|} \qquad \text{for } i = 1, 2, \ldots P$$

(iv) Each node computes by taking the dot product of its weight vector and input vector. $O_j = XW$; where O_j is output or activation level of unit j.

(v) Node with the largest activation level is declared the winner in the competition. This *wining* node is the only node that will generate an output signal, the activation level of other neurons is suppressed to zero. The Winning node with largest net$_i$ is allowed to change the weight that is weight adjustment is for winning neuron only while weights of other neurons remain unaffected. The weight adjustment criteria for this mode of training is such that

$$|X - W_m| = \min\{|X - W_i|\} \qquad \ldots(i)$$

where $i = 1, 2, \ldots, P$ and m is winning neuron no.

min represents the minimum. As explained earlier also in clustering that smaller the distance more close the patterns are and finding the minimum of P distances corresponds to finding maximum among the p scalar products.

$$W_m^t X = \max (W_i^t) X \qquad \text{where } i = 1, 2, \ldots, P$$

The left hand side of above equation can be written as

$$|X - W_m| = (X^t X - 2W_m^t X - 1)^{1/2} \qquad \ldots(ii)$$

It is clear that searching for the minimum of p distances as on right hand side of (i) Corresponds to finding maximum among p scalar products

$$W_m{}^t X = \max (W_i^t X) \qquad \ldots(iii)$$

For $i = 1, 2, \ldots P$

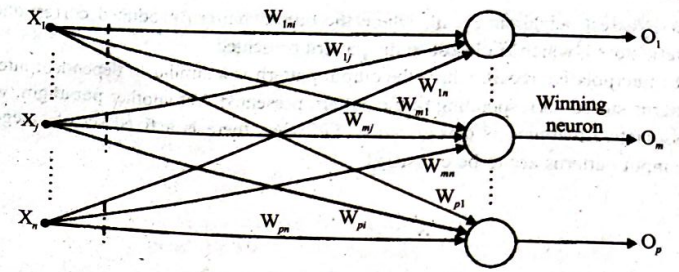
After the winning neuron has been identified and declared a winner, its weights must be adjusted so that the distance is reduced in the current training step. Thus $|X - W_m|$ must be reduced, perferably along the gradient direction in the weight space $w_{m1}, w_{m2}, \ldots\ldots\ldots w_{mn}$

$$\nabla w_m \| x - w_m \|^2 = -2(x - w_m) \qquad \ldots(iv)$$

It seems reasonable to reward the weights of the $X - W_m$ winning neuron with an increment of weight in the negative gradient direction, thus in the direction $x - w_m$. We thus have

$$\nabla \hat{w}_m = \alpha(x - \Delta \hat{w}_m) \qquad \ldots(v)$$

where α is a small learning constant selected heuristically, usually between 0.1 and 0.7.



Learning according to equation (iv) and (v) is called Winner-take-All learning and it is a common competitive and unsupervised learning technique. The winning neuron with the largest net is rewarded with a weight adjustment, while the weights of others remain unaffected.

Another modification of the winner-take-all learning rule is that both the winner's and loser's weights are adjusted in proportion to their level of responses. This may be called leaky competitive learning and should provide more subtle learning in the cases for which clusters may be hard to distinguish.

Q.9.(b) Write short note on Recall Mode for self organizing network. **(10)**

Ans. Recall Mode : The network trained in the winner-take-all mode responds instantaneously during feedforward recall at all p neuron outputs. The response is computed according to $y = M\,[WX]$, where M is non linear diagonal operator. Note that the layer now performs as a filter of the input vectors such that the largest output neuron is found as follows :

$$y_m = \max (y_1, y_2, \ldots, y_p)$$

and the input is identified as belonging to cluster m. In general, the neurons' activation functions should be continuous in this network. For some applications, however, $y_m = 1$ and $y_i = 0$, $i \neq m$ must be set in the recall mode of the clustering layer. In this way, for example, the weights of the following layer can be fanned out from the activated node of this previous layer while other nodes remain suppressed. Before a one-to-one vector-to-cluster mapping can be made after the network is trained in the unsupervised mode, it needs to be calibrated in a supervised environment. The calibration involves the teacher applying a sequence of p best matching class /cluster inputs and labeling the output nodes 1, 2, ..., p, respectively, according to their observed responses. Obviously, the calibrating labels assigned to the physical neurons of the layer would vary from training to training depending on the sequence of data within the training set, the training parameters, and the initial weights. Once the clustering network is labeled, it can perform as a cluster classifier in a local representation.

In the recall mode, the network is presented with the feature vector of a single sample. The output of the network determines the class to which the sample belongs. H*etero-association* is related to two recall mechanisms:

(a) Nearest-neighbour recall, where the output pattern produced corresponds to the input pattern stored, which is closest to the pattern presented.

(b) Interpolative recall, where the output pattern is a similarity dependent interpolation of the patterns stored corresponding to the pattern presented. Yet another paradigm, which is a variant associative mapping is classification, *i.e.* when there is a fixed set of categories into which the input patterns are to be classified.

NEURAL NETWORKS

Dec. - 2019

Paper Code:CSE-407-F

Note : ***Attempt five questions in all, selecting one question from each Section. Question No. 1 is compulsory. All questions carry equal marks.***

Q.1.(a) What is Linear associator ? Explain briefly.

Ans. Linear associator : The linear associator is one of the simplest and first studied associative memory model. Linear associator is a feedforward type network where the output is produced in a single feedforward computation.

In the fig., all the m input units are connected to all the n output units via the connection weight matrix $W = [w_{ij}]_{m \times n}$ where w_{ij} denotes the synaptic strength of the unidirectional connection from the ith input unit to the jth output unit.

It is the connection weight matrix that stores the p different associated pattern pairs $\{(X_k, Y_k) | k = 1, 2, \ldots\ldots p\}$ where $X_k \hat{I} \{-1,+1\}_m$ and $Y_k \hat{I} \{-1,+1\}_n$ in a distributed representation.

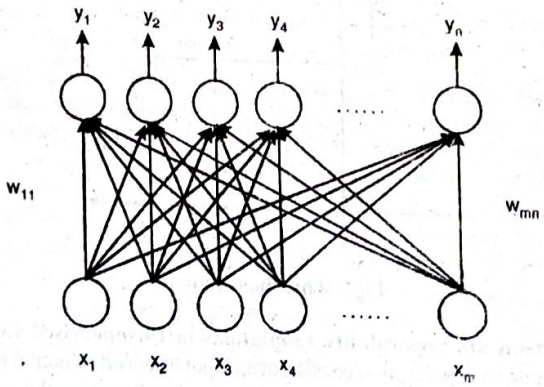


Fig. : Linear Associator

Q.1.(b) What do you mean by activation function and threshold activation function? **(5)**

Ans. Activation Functions : Activation Functions for the hidden units are needed to introduce non-linearity into the network. Without non-linearity hidden units would not make networks more powerful than just plain Perceptron(which do not have any hidden units, just input and output units). The reason is that a linear function of linear functions is again a linear function. However, it is the nonlinearity (i.e., the capability to represent non-linear functions) that makes multilayer networks so powerful. Almost any nonlinear function does the job, except for polynomials. The differential activation functions are useful for solving function which is bounded;

the sigmoid functions such as logistic and tanh and the Gaussian function are the most common choices. Functions such as tanh or arctan that produce both positive and negative values tend to yield faster training than functions that produce only positive values such as logistic, because of better numerical conditioning. For hidden units, sigmoid activation functions are difficult to train because the error function is stepwise constant, hence the gradient either does not exist or is zero, making it impossible to use back propagation or more efficient gradient-based training methods. Sigmoid units are easier to train that threshold units. With sigmoid units, a small change in the weights will usually produce a change in the outputs, which makes it possible to tell whether that change in the weights is good or bad. With threshold units, a small change in the weights will often produce no change in the outputs.

Threshold activation function : This function output is either 0 or 1 and given by the function as

$$\text{OUT} = 1 \text{ if SUM} > 0$$
$$= 0 \text{ if SUM} < 0$$

And its graphical representation is shown in fig.

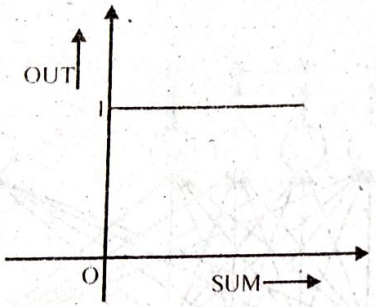


Fig. : Threshold Function

Q.1.(c) What are Separability Limitations in Unsupervised Learning?

Ans. A single layer perceptron consists of an input layer and an output layer. The activation function applied is hard-limiting function. An output unit will assume the value 1 if the sum of weighted input is greater than its threshold. *i.e.*,

$W_{ji}X_i > W_j$ where W_j is weight from unit i to unit j.

X_i is the input from unit i and Δ_j is the threshold on unit j

Let there are 2 classes A and B. If $W_{ji}X_i > \Delta_j$ where $i = 1, 2, \ldots n$ forms a hyperplane, dividing the space in 2 halves.

Linear separability : When a linear hyperplane exists to place the instances of one class on one side and those of other class on the other side of plane. Then this is called Linear Separability. For example in case of OR gate we can draw a linear hyperplane to separate the input whose corresponding outputs are 1 on one side and the inputs whose corresponding outputs are 0 an other side of the plane.

Table : OR Gate

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	1

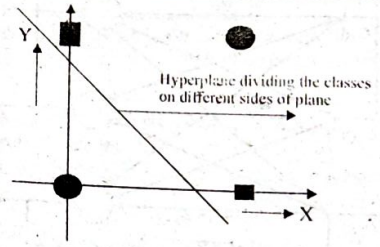


Fig. : The OR gate

But all the classification problems are not linear separable. Fox example the EX-OR problem is not linear separable because EX-OR is non equality gate.

Q.1.(d) Differentiate between Feedforward and Feedback Network.

Ans. Feedforward Network : Feedforward neural network is a biologically inspired classification algorithm. It consists of a (possibly large) number of simple neuron-like processing units, organized in layers. Every unit is a layer is connected with all the units in the previous layer. These connections are not all equal; each connection may have a different strength or weight. The weights on these connections encode the knowledge of a network. Often the units in a neural network are also called nodes. Data enters at the inputs and passes through the network, layer by layer, until it arrives at the outputs. During normal operation, that is when it acts as a classifier, there is no feedback between layers. This is why they are called feedforward neural networks.

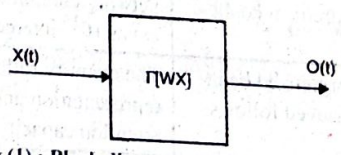


Fig.(1) : Block diagram of Feedforward Network

Feedback Network/Recurrent Network : Recurrent neural netwroks (*RNN*) have a closed loop in the network topology. They are developed to deal with the time varying or time-lagged patterns and are usable for the problems where the dynamics of the considered process is complex and the measured data is noisy. Specific groups of the units get the feedback signals from the previous time steps and these units are called context unit. The *RNN* can be either fully or partially connected. In a fully connected *RNN* all the hidden units are connected recurrently.

whereas in a partially connected *RNN* the recurrent connections are omitted paratially. Examples of recurrent neural networks are Hopfield networks, Regressive networks, Jordan-Elman networks, and Brain-State-In-a-Box (BSB) networks.

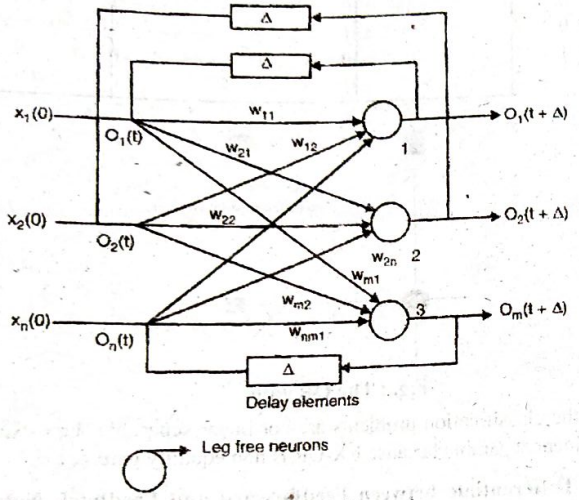


Fig.(2) : Feedback Network

SECTION - A

Q.2. How biological neural is different from the artificial naural network? Explain the architecture of biological neural networks. (20)

Ans. Comparision between **ANN and biological networks** are as follows :

S.No.	ANN	Biological Networks
1	ANNs have usually been limited to 10000 units with hundreds of connections per unit.	The human brain is extremely large for a neural network containing 10^{11} neurons and there exists 10^{15} interconnections.
2	The memory has an organized behaviour and the data retrieved follows an ordered sequence.	The memory follows a pattern of distributed representation and data retrieval depends on retention capacity of the brain
3	Learning takes place by a formulated set of rules and hence the system is less faulty. The processing elements receive many signals and sum up the weighted inputs. These networks can be retrained in case of significant damage (i.e. loss of data).	Learning takes place arbitrarily and hence the system is bound for failure. Biological networks are fault tolerable and even in a traumatic loss other neurons can takes over the functions of damaged cells.

4	Redundancy can increase the reliability of the system, allowing it to function even when some of the neural units are destroyed.	Redundancy is used. When some cells die, the system appears to perform the same. It can counteract source of noise in biological systems.
5	The strength of neuron depends on whether It is active or inactive and has relatively simpler interconnections.	The strength of neuron depends on active chemicals present and connections are stronger or weaker as a result of structure layer rather than individual synapses.
6	ANNs focus on learning a single task.	Biological system have broad capabilities and can address many different types of tasks.
7	Artificial systems are slow to converge and usually requie hundreds or thousands of training presentations for learning to take place.	Biological systems have the property of being able to learn with as little as one training presentation. A face that is viewed once, can be recognized again.

Architecture of biological neural networks : The elementary nerve cell called a neuron is the fundamental building block of biological neural network. The three main components of a biological neuron are :

1. A neuron cell body called *soma.*
2. Branching extensions called *dendrites* for receiving input, and
3. An *axon* that carries the neuron's output to the dendrites of other neurons.

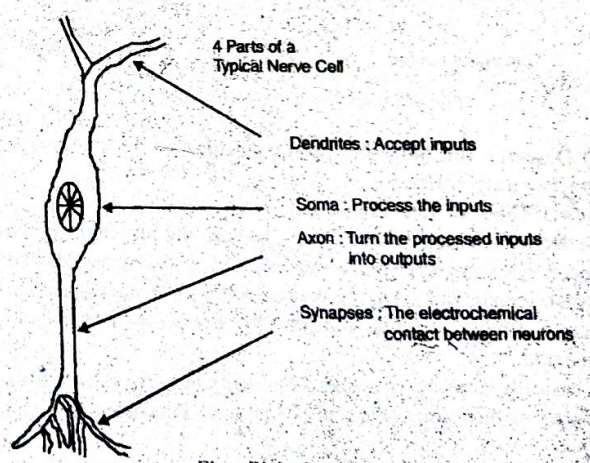


Fig. : Biological Neuron.

In the human brain, a typical neuron collects signals from others through a host of fine structures called *dendrites*. The neuron sends out spikes of electrical activity through a long, thin stand known as an axon, which splits into thousands of branches. The axon-dendrite contact

organ is called a synapse. At the end of each branch, a structure called a *synapse* converts the activity from the axon into electrical effects that inhibit or excite activity from the axon into electrical effects that inhibit or excite activity in the connected neurons. When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon. Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on another changes. The neuron is able to respond to the total of its inputs aggregated within a short time interval called period of latent summation.

ANN is defined as an interconnection of neurons such that the neuron outputs are connected, through the weights to all other neurons including themselves; both lag-free and delay connections are allowed.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

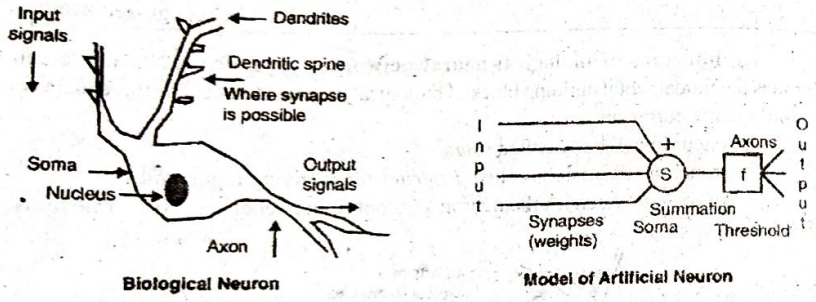


Fig. : Similarities between Biological Neuron and ANN

Q.3.(a) Describe the following learning Rules : **(10)**

(i) Delta learning Rule

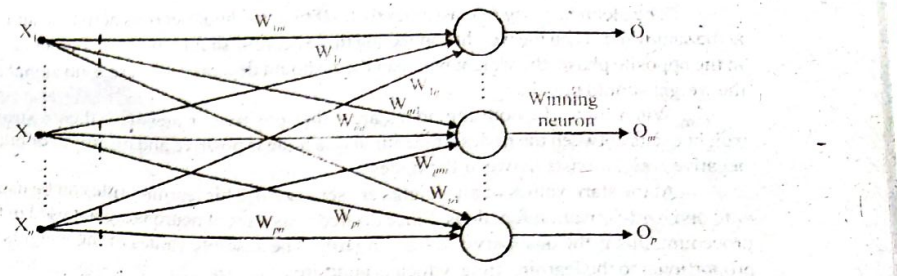
(ii) Winner Take All Learning Rule

Ans. (i) Delta learning Rule : The Delta learning algorithm is used for automatic updating of connection weights when processing information in a connectionist or neural network. This Delta rule was originally formulated by Widrow and Hoff, but obtained its final name because it adjusts connection weights according to the difference between the actual output of a network and the intended output.

Thus, it basically compares the output of an artificial network with the output that it was intended to produce, and then adjusts the connections within the network so as to better produce the intended output.

(ii) Winner Take All Learning Rule : (i) This rule is for unsupervised mode and it differ from all learning rules.

(ii) In this neurons are arranged in a layer of P units. This rule is an example of competitive learning and is used for unsuperbised training mode.



(iii) Learning is based on the fact that one of the neurons in the layer (suppose m) has maximum response due to input X, then that neuron is declared as Winner. As a result of this Winning event weight vector $w_m = [w_{m1}, w_{m2}, \ldots\ldots\ldots w_{mn}]^t$

(iv) The increment is the weight vector is computed as

$$\Delta w_m = \alpha (X - w_m) \quad \ldots(i)$$

for single weight adjustment for $i = 1, 2, \ldots\ldots, n$.

$\alpha > 0 \Rightarrow$ small learning constant

α decrease as learning constant increases.

(v) The winner selection is based on the criteria of maximum activation among all participating neurons in the competition :

$$\Delta w_{mi} X = \max (w_i^t x) \quad \text{where } i = 1, 2, \ldots\ldots P \quad \ldots(ii)$$

(vi) In this rule weights are initialized at random values.

Q.3.(b) Explain any three learning rules. **(10)**

Ans. Learning rule or **Learning process** is a method or a mathematical logic. It improves the Artificial Neural Network's performance and applies this rule over the network. Thus learning rules updates the weights and bias levels of a network when a network simulates in a specific data environment.

Applying learning rule is an iterative process. It helps a neural network to learn from the existing conditions and improve its performance.

Let us see different learning rules in the Neural network:

- **Hebbian learning rule** – It identifies, how to modify the weights of nodes of a network.
- **Perceptron learning rule** – Network starts its learning by assigning a random value to each weight.
- **Delta learning rule** – Modification in sympatric weight of a node is equal to the multiplication of error and the input.
- **Correlation learning rule** – The correlation rule is the supervised learning.
- **Outstar learning rule** – We can use it when it assumes that nodes or neurons in a network arranged in a layer.

Hebbian Learning Rule : The Hebbian rule was the first learning rule. In 1949 *Donald Hebb* developed it as learning algorithm of the unsupervised neural network. We can use it to identify how to improve the weights of nodes of a network.

The Hebb learning rule assumes that – If two neighbor neurons activated and deactivated at the same time. Then the weight connecting these neurons should increase. For neurons operating in the opposite phase, the weight between them should decrease. If there is no signal correlation, the weight should not change.

When inputs of both the nodes are either positive or negative, then a strong positive weight exists between the nodes. If the input of a node is positive and negative for other, a strong negative weight exists between the nodes.

At the start, values of all weights are set to zero. This learning rule can be used0 for both soft- and hard-activation functions. Since desired responses of neurons are not used in the learning procedure, this is the unsupervised learning rule. The absolute values of the weights are usually proportional to the learning time, which is undesired.

The Hebbian learning rule describes the formula as follows:

$$w_{ij} = x_i * x_j$$

Perceptron Learning Rule : As you know, each connection in a neural network has an associated weight, which changes in the course of learning. According to it, an example of supervised learning, the network starts its learning by assigning a random value to each weight. Calculate the output value on the basis of a set of records for which we can know the expected output value. This is the learning sample that indicates the entire definition. As a result, it is called a learning sample.

The network then compares the calculated output value with the expected value. Next calculates an error function ", which can be the sum of squares of the errors occurring for each individual in the learning sample.

Computed as follows: $\sum_i \sum_j \left(E_{ij} - O_{ij}\right)^2$

Perform the first summation on the individuals of the learning set, and perform the second summation on the output units. Eij and Oij are the expected and obtained values of the jth unit for the ith individual.

The network then adjusts the weights of the different units, checking each time to see if the error function has increased or decreased. As in a conventional regression, this is a matter of solving a problem of least squares.

Since assigning the weights of nodes according to users, it is an example of supervised learning.

Delta Learning Rule : Developed by *Widrow* and *Hoff*, the delta rule, is one of the most common learning rules. It depends on supervised learning.

This rule states that the modification in sympatric weight of a node is equal to the multiplication of error and the input.

In Mathematical form the delta rule is as follows:

$$\Delta W = \eta\left(t - y\right)x_i$$

For a given input vector, compare the output vector is the correct answer. If the difference is zero, no learning takes place; otherwise, adjusts its weights to reduce this difference. The change in weight from u_i to u_j is: $dw_{ij} = r^* a_i * e_j$.

where r is the learning rate, a_i represents the activation of u_i and e_j is the difference between the expected output and the actual output of u_j. If the set of input patterns form an independent set then learn arbitrary associations using the delta rule.

It has seen that for networks with linear activation functions and with no hidden units. The error squared vs. the weight graph is a paraboloid in n-space. Since the proportionality constant is negative, the graph of such a function is concave upward and has the least value. The vertex of this paraboloid represents the point where it reduces the error. The weight vector corresponding to this point is then the ideal weight vector.

We can use the delta learning rule with both single output unit and several output units.

While applying the delta rule assume that the error can be directly measured.

The aim of applying the delta rule is to reduce the difference between the actual and expected output that is the error.

SECTION - B

Q.4. Explain the linearly separable classification with suitable example. (20)

Ans. Linears Separability : When a linear hyperplane exists to place the instances of one class on one side and those of other class on the other side of plane. Then this is called *Linear Separability*. For example in case of OR gate we can draw a linear hyperplane to separate the inputs whose corresponding outputs are 1 on one side and the inputs whose corresponding outputs are 0 an other sidee of the plane.

Table : OR Gate

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	1

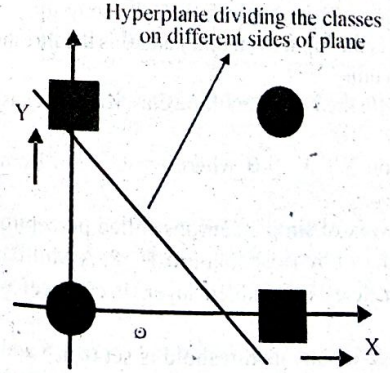


Fig. : The OR Gate

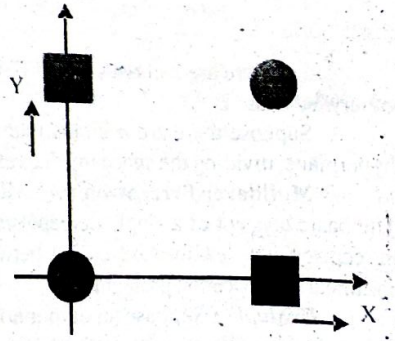


Fig. : X-OR Function

But all the classification problems are not linear separable. For example the EX-OR problem is not linear separable because EX-OR is non equality gate i.e.

Table: The EX-OR Gate

Input A	Input B	Output
0	0	0
0	1	1
1	0	1
1	1	0

In this case we can't draw any hyperplane that divides the space so that different classes are on different sides. To cope up with a problem which is not linear separable, a Multilayer Perceptron is required.

Q.5.(a) Discuss the Classification model, Features & Decision regions in detail. (10)

Ans. Classification Models : A Neural Network classifies a given object presented to it according to the output activation. For binary outputs, 1 corresponds to one class and 0 corresponds to the other. For continuous outputs between 0 and 1, 0.5 can be used as threshold to make a decision. In the case of multiple outputs, the object is assigned to the class corresponding to the output node with maximum activation. Two types of classification models are discussed below :

1. Single Layer Perceptron
2. Multilayer layer perceptron

The Perceptron : The perceptron is a program that learns concepts, i.e. it can learn to respond with **True**(1) or **False**(0) for inputs we present to it, by repeatedly "studying" examples presented to it.

Single Layer Perceptron : A single layer perceptron consists of an input layer and an output layer. The activation function applied is hard-limiting function. An output unit will assume the value if the sum of weighted inputs is greater than its threshold. i.e.

$\sum W_{ji} X_i > \theta_j$ where W_{ji} is weight from unit i to unit j.
X_i is the input from unit i and θ_j is the threshold on unit j

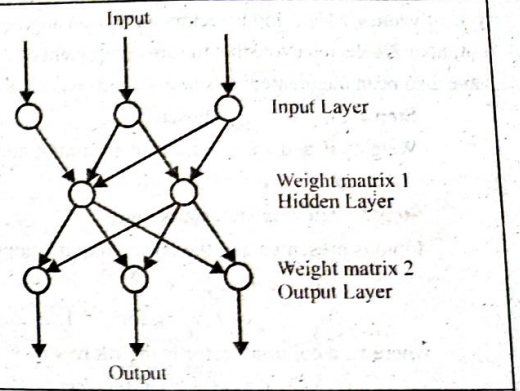
Let there are 2 classes A and B. If $\sum W_{ji} X_i > \theta_j$ then object will be classified as Class A otherwise Class B.

Suppose there are n inputs then the equation $\sum W_{ji} X_i > \theta_j$ where $i = 1, 2 \ldots n$ froms a hyperplane, dividing the space in 2 halves.

Multilayer Perceptron : An MLP is a network of simple neurons called perceptrons. The basic concept of a single perceptron was introduced by Rosenblatt in 1958. A Multilayer Perceptron is a feedforward neural network with at least one hidden layer. It can deal with nonlinear classification problems.

Example : Suppose an output node receives 2 inputs, its threshold is set to 1.5 and its input weights are both set to 1. Therefore when both the input units are active (i.e. 1) the output node will be active. In this case the output node performs a logical AND operation. On the other hand if threshold is set to 0.5, then any active input can activate the node. In this case the output node performs a logical OR operation. Thus by choosing a different set of weights and threshold, a node can implement a different logical operation.

Multilayer Perceptron Characteristics

Sample Structure	
Type	Feed-forward
Neuron Layers	1 input layer 1 or more hidden layers 1 output layer
Input value types	Binary
Activation Functions	Hard limiter/sigmoid
Learning Method	Supervised
Learning algorithm	Delta learning rule Back-propagation (mostly used)

Q.5.(b) Explain error back-propagation algorighm in details. (10)

Ans. The layer network is mapping the input vector z into the output vector o as follows :

$$o = N(z)$$

where N denotes a composite nonlinear matrix operator. For the two-layer net the mapping $z \rightarrow o$ can be represented as a mapping within a mapping, or

$$o = G[W\Gamma[Vz]]$$
$$\Gamma[Vz] = y$$

and it related to the hidden layer mapping $z \rightarrow y$. Note that the right arrows denote mapping of one space into another. Each of the mapping is performed by a single-layer of the layered network. The operator Γ is a nonlinear diagonal operator.

Algorithm :

Given are P training pairs

$$\{z_1, d_1, z_2, d_2, \ldots\ldots, z_p, d_p,\},$$

Where z_i is $(I \times 1)$, d_i is $(K \times 1)$, and $i = 1, 2, \ldots P$. Note that the Ith component of each x_i is of value – 1 since input vectors have been augmented. Size J –1 of the hidden layer having outputs y is selected. Note that the Jth component of y is of value – 1, since hidden layer outputs have also been augmented; y is $(J \times 1)$ and o is $(K \times 1)$.

Step 1 : $\eta > 0$. E_{max} chosen

Weights W and Y are initialized at small random values; W is $(K \times J)$, V is $(J \times I)$.

$$q \leftarrow 1, \quad p \leftarrow 1, E \leftarrow 0$$

Step 2 : Training step starts here.

Input is presented and the layers output computed.

$$z \leftarrow z_p, \; d \leftarrow d_p$$

$$y_j \leftarrow f(v_j^t z), \text{ for } j = 1, 2, \ldots\ldots, J$$

where v_j, a column vector is the Jth row of V and

$$o_k \leftarrow f(w_k^t y), \text{ for } k = 1, 2, \ldots\ldots, K$$

where w_k, a column vector, is the kth row of W.

Step 3 : Error value is computed

$$E \leftarrow \frac{1}{2}(d_k - o_k)^2 + E, \text{ for } k = 1, 2, \ldots\ldots, K$$

Step 4 : Error signal vectors δ_0 and δ_y of the both layers are computed.

Vector δ_0 is $(K \times 1)$, δ_y is $(J \times 1)$.

The error signal terms of the output layer in this step are

$$\delta_{ok} = \frac{1}{2}(d_k - o_k)(1 - o_k^2), \text{for } k = 1, 2, \ldots\ldots, K$$

The error signal terms of the hidden layer in this step are

$$\delta_{yj} = \frac{1}{2}(1 - y_j^2)\sum_{k=1}^{K} \delta_{ok} w_{kj}, \text{ for } j = 1, 2, \ldots\ldots, J$$

Step 5 : Output layer weights are adjusted :

$$w_{kj} \leftarrow w_{kj} + \eta \delta_{ok} y_j \text{ for } k = 1, 2, \ldots. K \text{ and } j = 1, 2, \ldots\ldots, J$$

Step 6 : Hidden layer weights are adjusted

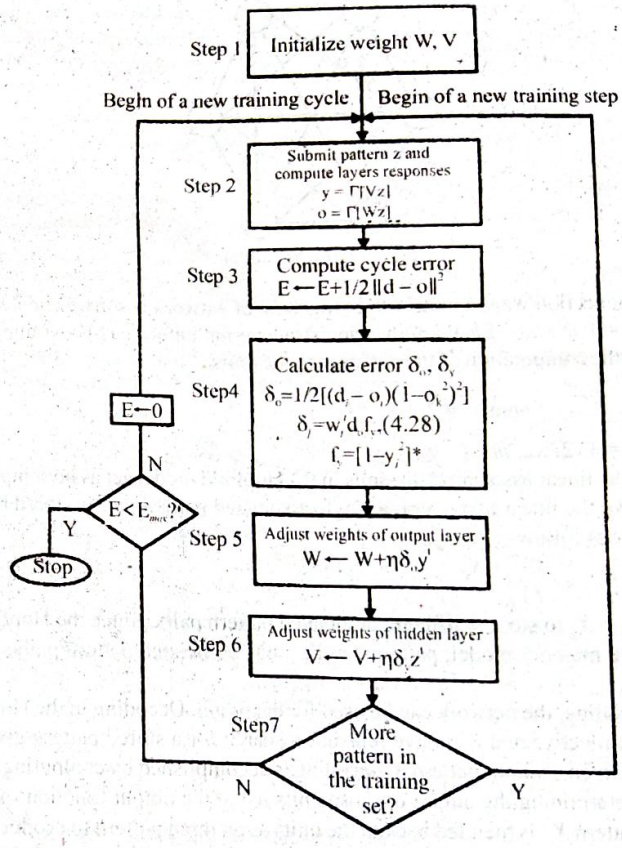
$$v_{ji} \leftarrow v_{ji} + \eta \delta_{yj} z_i \text{ for } j = 1, 2, \ldots. J \text{ and } i = 1, 2, \ldots\ldots, I$$

Step 7 : If $p < P$ then $p \leftarrow p + 1$, $q \leftarrow q + 1$, and go to Step 2; otherwise, go to Step 8.

Step 8 : The training cycle is completed.

For $E < E_{max}$ terminate the training session. Output weights W, V, q and E.

If $E > E_{max}$ then $E \leftarrow 0$, $p \leftarrow 1$, and initiate the new training cycle by going to Step 2.



SECTION - C

Q.6. Explain the various architectures of Hopfield network in detail. How learning process does occur in Hopfield network? (20)

Ans. Hopfield Network : The Hopfield model was proposed by John Hopfield of the California Institute of Technology during the early 1980s. The dynamics of the Hopfield model is different from that of the linear associator model in that it computes its output recursively in time until the system becomes stable. Below is a Hopfield model with six units, where each node is connected to every other node in the network.

Unlike the linear associator model which consists of two layers of processing units, one serving as the inputs layer while the other as the output layer, the Hopfield model consists of a single layer of processing elements where each unit is connected to every other unit in the network other than itself.

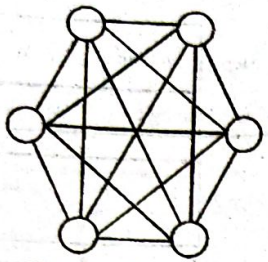


Fig. : Hopfield Model

The connection weight matrix W of this type of network is square and symmetric, i.e., $w_{ij} = w_{ji}$ for $i, j = 1, 2 \ldots, m$. Each unit has an extra external input I_i. This extra input leads to a modification in the computation of the net input to the units :

$$\text{input}_i = \alpha \sum_{i=1}^{m} x_i w_{ij} + I_i$$

where $j = 1, 2, \ldots, m$.

Unlike the linear associator, the units in the Hopfield model act as both input and output units. But just like the linear associator, a single associated pattern pair is stored by computing the weight matrix as follows :

$$W_k = X_k$$
$$TY_k$$

where $Y_k = X_k$ to store p different associated pattern pairs. Since the Hopfield model is an autoassociative memory model, patterns, rather than associated pattern pairs, are stored in memory.

After encoding, the network can be used for decoding. Decoding in the Hopfield model is achieved by a collective and recursive relaxation search for a stored pattern given an initial stimulus pattern. Given an input pattern X, decoding is accomplished by computing the net input to the units and determining the output of those units using the output function to produce the pattern X'. The pattern X' is then fed back to the units as an input pattern to produce the pattern X''. The pattern X'' is again fedback to the units to produce the pattern X'''. The process is repeated until the network stabilizes on a stored pattern where further computations do not change the output of the units.

If the input pattern X is an incomplete pattern or if it contain some distortions, the stored pattern to which the network stabilizes is typically one that is most similar to X without the distortions. This feature is called *pattern completion* and is very useful in many image processing applications.

During decoding, there are several schemes that can be useed to update the output of the units. The updating schemes are *synchronous* (or parallel as termed in some literatures), *asynchronous* (or sequential), or a combination of the two (hybrid). Using the synchronous updating scheme, the output of the units are updated as a group prior to feeding the output back to the network. On the other hand, using the asynchronous updating scheme, the output of the units are updated in some order (e.g. random or sequential) and the output are then fed back to

the network after each unit update. Using the hybrid synchronous-asynchronous undating scheme, subgroups of units are updated synchronously while units in each subgroup updated asynchronously.

Discrete Hopfield Networks : The Hopfield network is fully connected and symmetric in nature i.e.,

$$w_{ij} = w_{ji}$$

And there are no self connections

i.e., $$w_{ii} = 0$$

They have single layer feedback network architecture i.e., why they are called as Recurrent Networks. Both binary (0, 1) and bipolar (–1, 1) values can be given as input to the network.

In this network, only one unit updates at a time. Here input pattern is applied to the network and network's output is computed. Now initial input pattern is removed and the output pattern is forced as input to the input layer through feedback interconnections and produces second updated output. This process continues until no new, updated responses are produced and network reaches an equilibrium/attractor state.

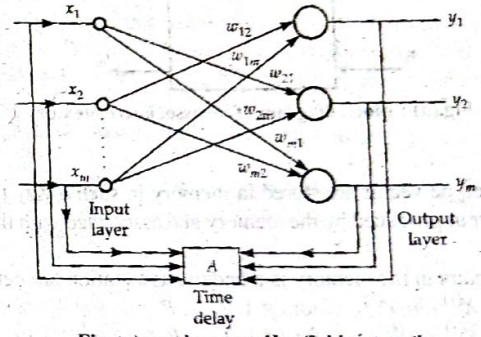


Fig. : Asynchronous Hopfield network

Training Algorithm of Discrete Hopfield Net :

(a) For storing a set of binary input patterns $s(p)$, $p = 1$ to P,

where $s(p) = (s_1(p), \ldots s_i(p), \ldots s_n(p)$,

The weight matrix is given by

$$w_{ij} = \sum_{p=1}^{P} [2s_i(p) - 1][2s_j(p) - 1] \qquad \text{for } i \neq j$$

(b) For storing a set of bipolar input patterns

$s(p)$, $p = 1$ to P,

where $s(p) = (s_1(p), \ldots s_i(p), \ldots s_n(p)$.

The weight matrix is given by

$$w_{ij} = \sum_{p=1}^{P} s_i(p) s_j(p) \qquad \text{for } i \neq j$$

and $w_{ij} = 0$ as weights have no self connections.

Q.7.(a) Explain the architecture of associative memory in detail. Also discuss its retrieval algorithm. (10)

Ans. The block diagram of associative memory performing an associative mapping of an input vector X into an output vector V.

$$\therefore \quad V = M[X] \qquad ...(i)$$

where M = denotes a general non-linear diagonal operator.

The different for each type of memory model used. For dynamic memories, M also involves time variable.

The computation of M is done with the help of "recording or storage algorithm". The mapping of V to X is called retrievel. The storage algorithm depends on whether an auto associative or a hetero-associative type of memory is designed.

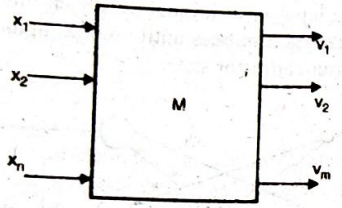


Fig.(a) : Block diagram of an associative memory

Assumption :

1. Certain prototype vector are stored in memory in such a way that once a key input has been applied, an output produced by the memory and associated with the key is the memory response.

2. Total stored pairs in the memory is P and the association are defined as

$$X^{(i)} \rightarrow V^{(i)} \quad \text{for } i = 1, 2, ..., P \qquad ...(ii)$$

and $$V^{(i)} \neq X^{(i)} \quad \text{for } i = 1, 2, ..., P,$$

The network can be termed as hetero associative memory.

∵ in hetero associative memory, there is associations between pairs of 2ordered sets of vector $\{X^{(1)}, X^{(2)}, ... X^{(P)},\}$ and $\{V^{(1)}, V^{(2)}, ... V^{(P)},\}$ as explained earlier.

If $$X^{(i)} \rightarrow V^{(i)}$$

$$V^{(i)} = X^{(i)} \quad \text{for } i = 1, 2, ..., P,$$

Then memory is called auto associative memory.

In auto associative memory, there is association within only one set which is $\{X^{(1)}, X^{(2)}, ... X^{(P)},\}$.

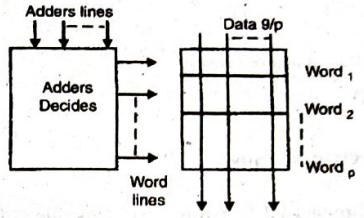
The realistic application of an auto associative mapping would be recovery of an undistorted prototype vector in response to the distorted prototype key vector.

Different Addressing Modes Which are Used for Memory Data Retrieval ;

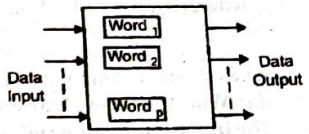
1. Address – addressable memory 2. Content – addressable memory

These two addressing modes are commonly used for memory data retrieval. Indigital computers, data can be accessed when their correct addresses in the memory are given as shown in fig.(b).

From the figure it is clear that data have input and output lines, a word line occesses and activates the entire row of binary cells containing word data bits. This activation takes place whenever binary address is decoded by address decoder. The addressed word can be either "read" or replaced during "write" operation.



(a) Address-addressable memory



(b) Content-addressable memory

Fig. : (b)

2nd type is *CAM* (content addressable memory) : In this words can be accessed based on the content of key vector. When the network excited with a portion of stored data $X(i)$, $i = 1, 2, ..., P$ the efficient response of auto associative network is the complete $X(i)$ vector.

Q.7.(b) Design a bi-directional associative memory to encode the following pattern : (10)

$A_1 = 100001$ $B_1 = 11000$

$A_2 = 011000$ $B_2 = 10100$

$A_3 = 001011$ $B_3 = 01110$

Ans. Consider $N = 3$ pattern pairs $(A_1, B_1), (A_2, B_2), (A_3, B_3)$ given by

$A_1 = (1\ 0\ 0\ 0\ 0\ 1)$ $B_1 = (1\ 1\ 0\ 0\ 0)$

$A_2 = (0\ 1\ 1\ 0\ 0\ 0)$ $B_2 = (1\ 0\ 1\ 0\ 0)$

$A_3 = (0\ 0\ 1\ 0\ 1\ 1)$ $B_3 = (0\ 1\ 1\ 1\ 0)$

Convert these three binary pattern to bipolar form replacing 0s by –1s.

$X_1 = (1\ -1\ -1\ -1\ -1\ 1)$ $Y_1 = (1\ 1\ -1\ -1\ -1)$

$X_2 = (-1\ 1\ 1\ -1\ -1\ 1)$ $Y_2 = (1\ -1\ 1\ -1\ -1)$

$X_3 = (-1\ -1\ 1\ -1\ 1\ 1)$ $Y_3 = (-1\ 1\ 1\ 1\ -1)$

The correlation matrix M is calculated as 6×5 matrix

$$M = X_1^T Y_1 + X_2^T Y_2 + X_3^T Y_3 = \begin{bmatrix} 1 & 1 & -3 & -1 & 1 \\ 1 & -3 & 1 & -1 & 1 \\ -1 & -1 & 3 & 1 & -1 \\ -1 & -1 & 1 & 1 & 3 \\ -3 & 1 & 1 & 3 & 1 \\ -1 & 3 & -1 & 1 & -1 \end{bmatrix}$$

Suppose we start with $\alpha = X_3$, and we hope to retrieve the associated pair Y_3. The calculations for the retrieval of Y_3 yield :

$\alpha M = [-1 \ -1 \ 1 \ -1 \ 1 \ 1] \ (M) = (-6 \ 6 \ 6 \ 6 \ -6)$

$\Phi(\alpha M) = \beta' = (-1 \ 1 \ 1 \ 1 \ -1)$

$\beta' M^T = (-5 \ -5 \ 5 \ -3 \ 7 \ 5)$

$\Phi(\beta' M^T) = (-1 \ -1 \ 1 \ -1 \ 1 \ 1) = \alpha'$

$\alpha' M = (-1 \ -1 \ 1 \ -1 \ 1 \ 1) \ M = (-6 \ 6 \ 6 \ 6 \ -6)$

$\Phi(\alpha' M) = \beta'' = (-1 \ 1 \ 1 \ 1 \ 1 \ -1)$

This retrieved pattern β' is same as Y_3.

Hence, $(\alpha_f, \beta_f) = (X_3, Y_3)$ is correctly recalled, a desired result.

SECTION - D

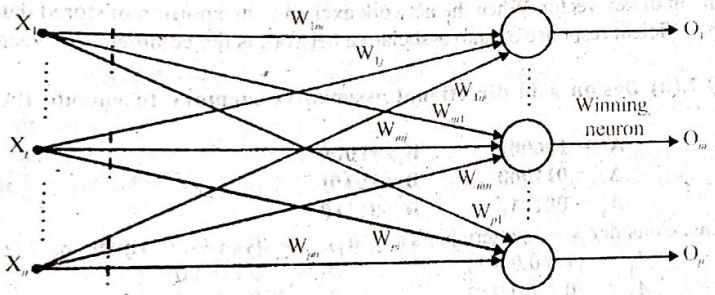
Q.8. Write a short note on : (20)

(a) Winner-take-all learning in unsupervised learning.

(b) Initialization of weights.

Ans. (a) Winner-take-all learning in unsupervised learning : (i) This rule is for unsupervised mode and it differ from all learning rules.

(ii) In this neurons are arranged in a layer of P units. This rule is an example of competitive learning and is used for unsuperbised training mode.



(iii) Learning is based on the fact that one of the neurons in the layer (suppose m) has maximum response due to input X, then that neuron is declared as Winner. As a result of this Winning event weight vector $w_m = [w_{m1}, w_{m2}, \ldots\ldots\ldots w_{mn}]^T$

(iv) The increment is the weight vector is computed as

$$\Delta w_m = \alpha (X - w_m) \qquad \ldots(i)$$

for single weight adjustment for $j = 1, 2, \ldots\ldots\ldots n$.

$\alpha > 0 \Rightarrow$ small learning constant

α decrease as learning constant increases.

(v) The winner selection is based on the criteria of maximum activation among all participating neurons in the competition :

$$\Delta w_{mi} \ X = \max \ (w_i^T x) \qquad \text{where } i = 1, 2, \ldots\ldots\ldots P \qquad \ldots(ii)$$

(vi) In this rule weights are initialized at random values.

(b) Initialization of weights : The weights of the network to be trained are initialized at small random values. The initialization affects the complete solution. If all the weights start out with equal weight values and if the soultion requries that unequal weights be developed, the

network may not be trained properly. artificial neural networks typically start out with randomized weights for all their neurons.

If training continues beyond a certain low error level, it will result in undersirable drift of weights. This casues error to increase and the quality of mapping by the network decreases. Therefore the choice of initial weights is however only one of several factors affecting training of error towards an acceptable error minimization.

Q.9. What is clustering? Explain unsupervised learning of clusters in detail.(20)

Ans. Clustering : Clustering is understood to be the grouping of similar objects and separating of dissimilar ones. The objective of clustering neural networks is to categorize or cluster data. The classes must first be found from the correlations of an input data stream. Since the network actually deals with unlabeled data, the clustering should be followed by labeling clusters with appropriate category names or numbers. This process of providing the category of objects with a label is usually termed as calibration.

Unsupervised learning of clusters : Learning is based on clustering of input data. No prior knowledge is assumed to be available regarding an input membership in a particular class. Rather, gradually detected characteristics and a history of training will be used to assist the network in defining classes and possible boundaries between them.

Clustering : Clustering is understood to be the grouping of similar objects and separating of dissimilar ones. The objective of clustering neural networks is to categorize or cluster data. The classes must first be found from the correlations of an input data stream. Since the network actually deals with unlabeled data, the clustering should be followed by labeling clusters with appropriate category names or numbers. This process of providing the category of objects with a label is usually termed as calibration.

Although the algorithm won't have names to assign to these clusters, it can produce them and then use those clusters to assign new examples into one or the other of the clusters. This is a data-driven approach that can work well when there is sufficient data; for instance, social information filtering algorithms, such as those that Amazon.com use to recommend books, are based on the principle of finding similar groups of people and then assigning new users to groups. In some cases, such as with social information filtering, the information about other members of a cluster (such as what books they read) can be sufficient for the algorithm to produce meaningful results. In other cases, it may be the case that the clusters are merely a useful tool for a human analyst. Unfortunately, even unsupervised learning suffers from the problem of overfitting the training data. There's no silver bullet to avoiding the problem because any algorithm that can learn from its inputs needs to be quite powerful.

Suppose we are given a set of patterns without any information as to the number of classes that may be present in the set. The clustering problem in such a case is that of identifying the number of classes according to a certain criterion, and of assigning the membership of the patterns in these classes. The clustering technique presented below :

Assumptions : The number of classes is known a priori. The pattern set $\{x_1, x_2, \ldots\ldots\ldots x_n\}$ is submitted to the input to determine decision functions required to identify possible clusters. Since no information is available from the teacher as far as the desired classifier's responses, we will use the similarity of incoming patterns as the criterion for clustering. To define a cluster, we need to establish a basis for assigning patterns to the domain of particular cluster.

How we will assign a pattern to domain of particular cluster? There are 2 rules for this.

(1) The most common similarity rule is to find the Euclidean distance between 2 patterns x for x_i defined as

$$|X - X_i| = [(X - X_i)^T (X - X_i)]^{1/2} \qquad \ldots(i)$$

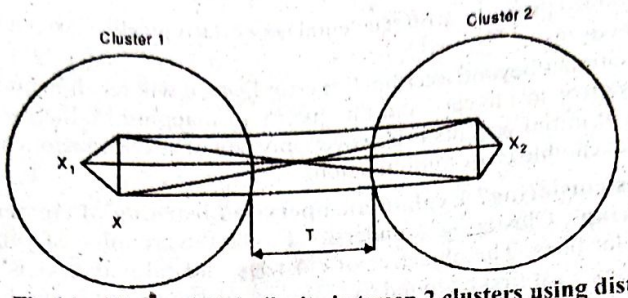


Fig.(a) : Measure of similarity between 2 clusters using distance

Smaller the distance, closer the patterns. Using (i) the distance between all the pairs are computed. Now how to distinguish the clusters? A distance T can then be chosen to discriminate clusters. The value T is understood as the maximum distance between patterns within a single cluster. Fig.(a) shows an example of two clusters with a T value chosen to be greater than the typical within-cluster distance but smaller than the between-cluster distance.

Let there are 2 patterns X_1 and X_2 and we have to find which pattern is similar to pattern X. Then using (1) distance is calculated. The value for which the distance is small, closer the pattern is with x.

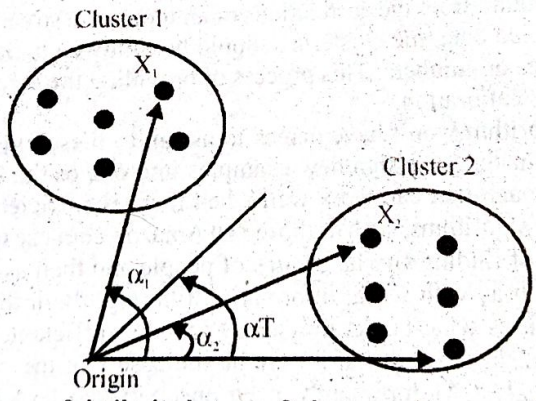


Fig.(b) : Measure of similarity between 2 clusters using normalized scalar product

(2) Another similarity rule is the cosine of the angle between x and x_i:

$$\cos \alpha = \frac{\left(X' X_i\right)}{\| X \| . \| X_i \|} \quad \ldots(ii)$$

This rule is particularly useful when clusters develop along certain principal and different axes as shown in fig.(b). For $\cos \alpha_2 < \cos \alpha_1$ pattern x is more similar to x_2 than to x_1. It would thus be natural to group it with the second of the two apparent clusters. To facilitate this decision, the threshold angle α_t can be chosen to define the minimum angular cluster distance. It should be noted, however, that the measure defined in equation (ii) should be used according to certain additional qualifications. If the angular similarity criterion equation (ii) is to be efficient, vectors x_1, x_2 and x should be of comparable, or better, identical lengths.