Section B

SECTION-B

NN

Que-8: Define Single & Multi layer Perceptron and their algorithm to classify

* Classification Models :- A Neural Network classifies a given object presented to it according to the o/p ~~fired~~ activation. For binary outputs, 1 corresponds to one class & 0 corresponds to the other. 6

2 types of classification models are discussed below :

1) Single layer perceptron
2) Multilayer 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 on it.

Example. for e.g, how do you teach a child to recognize what a chair is? You show him examples telling him "This is a chair, that one is not a chair" until the child learns the concept of what a chair is. This is exactly the idea behind the perceptron.

Single layer perceptron :- A single layer perceptron consists of an input layer & an output layer. An output unit will assume the value 1 if the sum of weighted inputs is greater than its threshold i.e

$$\sum w_{ji} x_i > \theta_j \qquad w_{ji} \text{ weight from } i \text{ to } j$$

$x_i \rightarrow$ input
$\theta_j \rightarrow$ threshold

There are 2 classes A & B. If $\sum w_{ji} x_i > \theta_j$ then object will be classified as class A otherwise class B. Explain

Linear separability → As discussed in notes of Unit-4

* The Perceptron Algorithm

1) Weight Initialization:
set all weights & node thresholds to small random numbers.

2) Calculation of Activation.
The activation level O_j of an output unit

$$O_j = F_h(W_{ji} X_i - \theta_j)$$

F_h = Hard limiting fun.

$$F_h(a) = \begin{cases} 1 & \text{if } a > 0 \\ 0 & \text{Else} \end{cases}$$

3) Weight Training:

a) Adjust weight $\boxed{W_{ji}(t+1) = W_{ji}(t) + \Delta W_{ji}}$

$W_{ji}(t)$ → weight from unit i to unit j at time t

ΔW_{ji} → weight adjustment.

b) Weight change may be computed by Delta learning Rule

$$\Delta W_{ji} = \eta \delta_j X_i$$

δ_j → Error at unit j $\qquad \delta_j = T_j - O_j$

T_j → desired output activation

O_j → actual " " at output unit j.

c) Repeat the iterations will convergence.

* Limitations :- First, the output values of a perceptron can take on only one of two values (T or F).

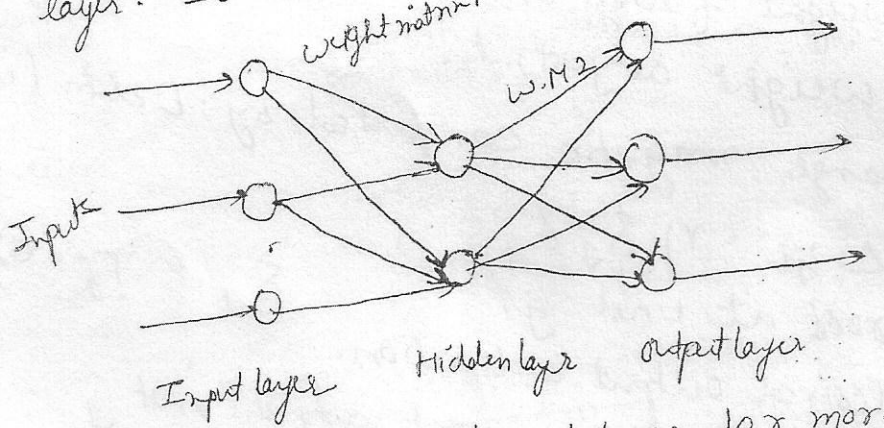
→ Second perceptrons can only classify linearly separable sets of vectors.

→ If a straight line or plane can be drawn t separate the input vectors into their correct categ the input vectors are linearly separable & the perception will find the solution If the vector are not linearly separable learning will never reac a point where all vectors are classified properly.

→ The most famous inability to solve problems with linearly nonseparable is boolean exclusive-or problem for these multilayer perceptron is used

* Multilayer Perceptron :- An MLP is a network of simpl neurons called perceptrons. It was introduced by Rosenbla in 1958

→ It is a feedforward neural N/w with at least one hidd layer. It can deal with nonlinear classification problem



Neuron layer → 1 input layer, 1 or more hidden layer, 1 output layer

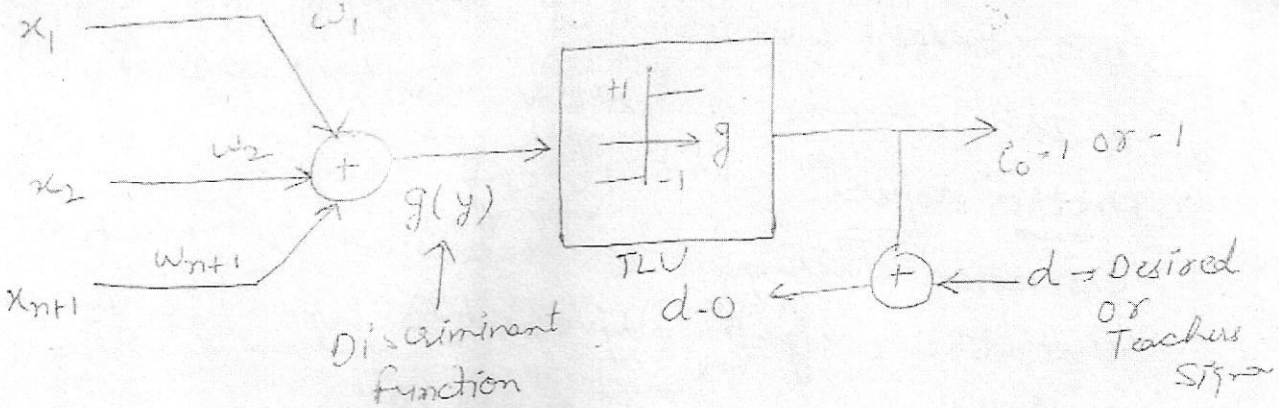
Input value type → Binary

Activation Func. → Sigmoid

Learning Method → Supervised

Learning algorithm → Delta learning Rule, Back propagation

* Training & Classification using the discrete perceptron



Linear Dichotomizer using TLU Based Perceptron

TLU → Threshold Logical Unit.

Dichotomizer: Classifier that divides the pattern space into 2 classes (Di means 2). i.e x_1 & x_2 only.

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

→ The N/w learns from the experience by comparing the targeted correct response with the actual response to clarify structure & the weight is adjused after each incorrect response based on error value generated

→ In this discrete perceptron concept the formula for adjused weights is

$$w' = w \pm Cy$$

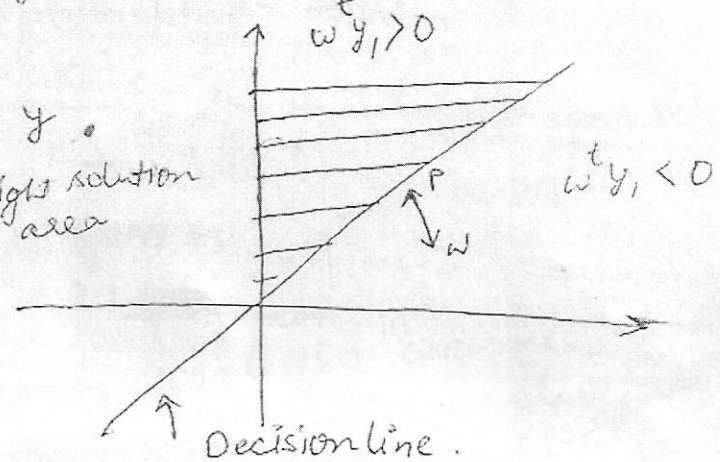
C → constant & $C > 0$ called correction increment

+ve → signal for pattern class 1, -ve → for class 2

If a correct classification takes place under the rule, no adjustment of weight is made, the 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 setup size is meaningful controlled.



Let p is distance of a point w' from the plane $w^t y = 0$ in (n+1) dimensional Euclidean space & it is calculated acc. to formula..

$$p = \frac{\pm w'^t y}{\|y\|}$$

p → distance & is always a non-negative scalar.

∴ can be written as

$$p = \frac{w'^t y}{\|y\|}$$

Now C must be selected such that corrected weight vector w^2 based on $w' = w \pm cy$ dislocated on the decision hyperplane $w'^t y = 0$ which is the decision hyperplane used for particular correction step

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

$$(w' \pm cy)^t y = 0$$

$$\therefore \quad (w'^t + cy^t) y = 0$$

$$w'^t y \pm cy^t y = 0$$

$$cy^t y = \mp w'^t y$$

$$c = \mp \frac{w'^t y}{y^t y}$$

Since C is correction constant & is positive.

$$c = \frac{|w'^t y|}{y^t y}$$

* Single Discrete Perceptron Training Algorithm :

Given that are P training pairs.

$\{x_1, d_1, x_2, d_2 \cdots x_p, d_p\}$ where

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

Input vector $y_i = \begin{bmatrix} x_i \\ 1 \end{bmatrix}$ for $i = 1, 2, \cdots P$

Step 1 $c > 0$ is chosen

Step 2 weights are initialized at w at small random values & $k \leftarrow 1$, $p \leftarrow 1$, $E \leftarrow 0$ where k is training step.

p is step counter within the training step or cycle.

Step 3 Training cycle starts.

Input is presented & output is computed.

$y \leftarrow y_p$, $d \leftarrow d_p$, $o \leftarrow \text{sgn}(w^t y)$

Step 4 Weights are updated

$$w \leftarrow w + \frac{1}{2} c(d-o) y$$

Step 5 Cycle error is computed

$$E \leftarrow \frac{1}{2}(d-o)^2 + E$$ Previous cycle error

Step 6 If $p < P$ then $p \leftarrow p+1$, $k \leftarrow k+1$ & go to step 3, otherwise go to step 7.

Step 7 Training is completed. If $E = 0$, terminate the training; output weight & k.

If $E > 0$, then $E \leftarrow 0$, $p \leftarrow 1$, new training step or iteration starts, starting from step 3.

* Single Layer Continuous Perceptron Networks for Linear separable classification (SCPTA).

Given P training pairs

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

Input vectors are used

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

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

$p \rightarrow$ step counter

$k \rightarrow$ training step

1. choose $\eta > 0$, λ (steepness constant) = 1,
$E_{max} > 0$.

2. Weight are initialized at w at small random value, counters & error are initialized.
$k \leftarrow 1$, $p \leftarrow 1$, $E \leftarrow 0$

3. Training starts → input y is presented & o/p is computed i.e
$y \leftarrow y_p$, $d \leftarrow d_p$, $o \leftarrow f(w^t y)$

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

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

6. ~~Error~~ If $p < P$, then $p \leftarrow p+1$, $k \rightarrow k+1$ & repeat step 3, otherwise go to step 7.

7. 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 & go to step 3.

Que 9 Define the terms → Classification, Model and Decision Region. (2017)

Unit - 2

Ans Classification Model, Features & Decision Region :-

1) Basic Meaning of Classifier :- Assume that a set of 8 points a_0 - a_7 in three dimensional space is available where

$a_0 = (-1, -1, -1)$
$a_1 = (-1, -1, 1)$
$a_2 = (-1, 1, -1)$
$a_3 = (-1, 1, 1)$
$a_4 = (1, -1, -1)$
$a_5 = (1, -1, 1)$
$a_6 = (1, 1, -1)$
$a_7 = (1, 1, 1)$

Elements of this set needs to be classified into two categories :-

(1) first category can be set of points with two or more +ve ones.

(2) Second category contains all the remaining points that do not belong to first category.

∴ First category consists of a_3, a_5, a_6, a_7 & remaining points lie in second category.

This is the function of classifier that is to classify to divide the pattern in diffuent categories

→ One of the most important application of ANN pattern classification :

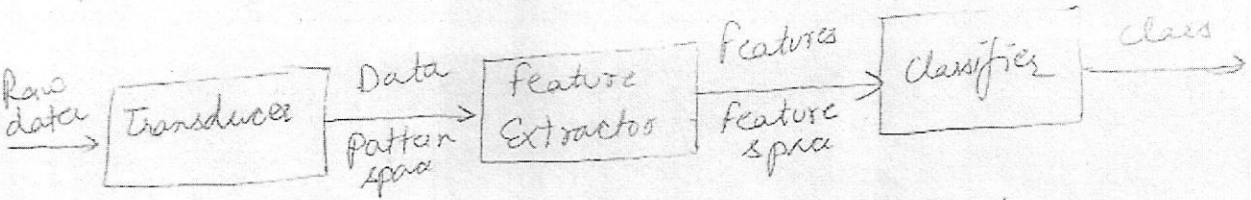
Pattern :- Pattern is the quantitative description of an object, event

Pattern can be classified as spatial or temporal patt

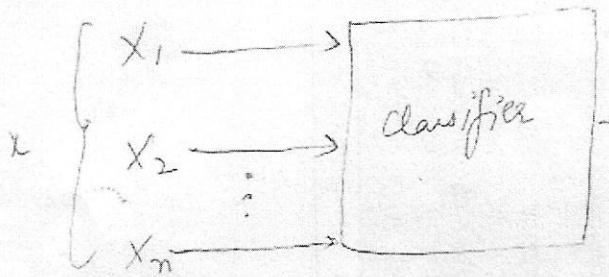
1) Spatial :- Ex are pictures, whether maps video go

2) Temporal :- Ex are speech signal, ECG etc. These patte involve the ordered sequence of data appea in time.

* Pattern Classification :- The main goal of pattern classification is to assign a physical object, event or phenomenon to one of the prespecified classes (categories) e.g. We classify various objects into different categories like living, non-living things etc.



a) Block diagram of classification sys.



b) General diagram of pattern classifier.

→ Transducer generally convert one form of energy to another form of energy. Here the input is raw data space $x = [x_1, x_2 -- x_n]$ & o/p is transducer is data in pattern space that belong to a certain category that is to $= 1, 2 -- R$. converted data at the o/p can be composed compressed data is called features.

→ Classifier at the end, convert or map the feature to the particular class.

3.3 LINEAR SEPARABILITY

The concept of linear separability in neural networks has been borrowed from perception in living beings (humans). In psychology, perception means mental organization and interpretation of sensory information. Perception is influenced by

(1) Intensity and physical dimensions of the stimulus
(2) Effects of preceeding stimulation
(3) Past experience
(4) Attention factors such as readiness to respond to stimulus, motivation and emotional state.

Patterns are perceived in visual organisation according to :

(1) The nearness to each other.

(2) Similarity between them.

(3) The tendency to perceive complete figures.

(4) The ability to distinguish important figures from background.

Sensory organs merely gather the signals whereas the stimuli are act perceived in the brain through :

(1) *Perceptual constancy.* It is the tendency of brain to interpret one object in the same manner regardless of such variations as distance, angle of sight or brightness.

(2) *Selective attention.* It is the tendency of the brain to focus on a limited number of stimuli and ignore those that are considered less important.

(3) *Depth perception.* It is produced by a variety of visual cues indicating perspective and by a slight disparity in the images of an object on the two retinas.

Now we will see how this concept of linearity is extended to linear operators used extensively in neural networks. Our aim in training a neural network is that it will respond with the desired classification when presented with an input pattern that it was trained on. Alternatively, it should respond when presented with an input that is sufficiently similar to one of the training patterns. The desired response from the output is a *'yes'* if the input pattern is a member of its class and a *'no'* if it is not. A 'yes' response is represented by an output signal of 1, a 'no' by an output signal of –1 in case of bipolar signals. Since we want one of the two responses, the activation function may be taken to be a step function. The value of function is 1 if the net input is positive and –1 if the net input is negative. The induced local field of the neuron for m-dimensional space is

$$v = \sum_{i=1}^{m} w_i x_i + b \qquad \ldots(1)$$

where b is the externally applied bias.

Hence, the goal of the network becomes to classify the set of externally applied stimuli $x_1, x_2, \ldots, x_m$ into one of the two classes. The decision rule for the classification is to assign the point represented by the inputs $x_1, x_2, \ldots, x_m$ to class I if the output y is +1 and to class II if it is –1. One can easily visualize that the boundary between the region where $v > 0$ and the region where $v < 0$ which we call the ***decision boundary.*** It is determined by the relation

$$\sum_{i=1}^{m} w_i x_i + b = 0 \qquad \ldots(2)$$

Depending on the number of input units in the network, Eq. (2) can represent a line, a plane or a hyperplane.

If there are weights (and a bias) so that all the training input vectors for which the correct response is +1 lie on one side of the decision boundary and all the training input vectors for which the correct response is –1 lie on the other side of the decision boundary, we say that the problem is ***"linearly separable"***, *i.e.*, separable by a linear combination of inputs.

Let us now consider the case of two input variables x_1 and x_2 for which the decision boundary takes the form of a straight line as shown in fig. (2). A point (x_1, x_2) that lies above the boundary line is assigned to class I and a point (x_1, x_2) that lies below the boundary line is assigned to class II. The choice of the sign for bias b determines which side of the separating line corresponds to a +1 response and which side to a −1 response.

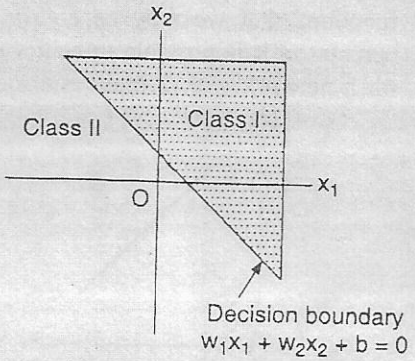


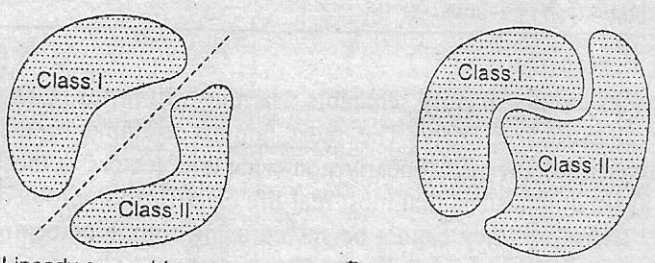
Fig. (2) Classification problem in two-dimensional input space.

Uptil now, we have considered problems with two input dimensions (x_1, x_2). If there is only one input dimension (x), then the two-class problem can be solved using a perception if and only if there is some value x_0 of x such all samples of one class occur for $x > x_0$ and all samples of other class occur for $x < x_0$ as shown in Fig. (3).

Fig. (3) Classification problem in one dimensional input space.

Similarly, if there are three input dimensions, a two class problem can be solved using a perceptron if and only if there is a plane that separates samples of different classes. Co-efficients of terms in the equation of the plane correspond to the weights of the perceptron.

For the perceptron to function properly, the two classes must be *linearly separable*. In other words, the patterns to be classified must be sufficiently separated from each other to ensure that the decision surface consists of a hyperplane. Refer to Fig. (4) which shows the case of a two-dimensional perceptron. In Fig. (4a), the two classes are sufficiently separated from each other so that we can draw a hyperplane (which is a straight line in this case) as the decision boundary.



(a) Linearly separable patterns (b) Non-linearly separable patterns

Fig. (4)

In Fig. (4b), they become non-linearly separable as no straight line can possibly separate samples belonging to the two classes. This type of situation is beyond the computing capability of perceptron. Most real-life classification problems are linearly non-separable. Hence they cannot be solved using perceptrons. This is the fundamental limitation of simple perceptron.

For simplicity sake, let us now consider a neutral network with 2 inputs and 1 output. Since there are two possible responses for each input there are 2^4 different functions that we may be able to train a very simple network to perform. Using perceptron it is possible to achieve linear separability functions as shown in Fig. (5) for a network with 2 input and 1 output.

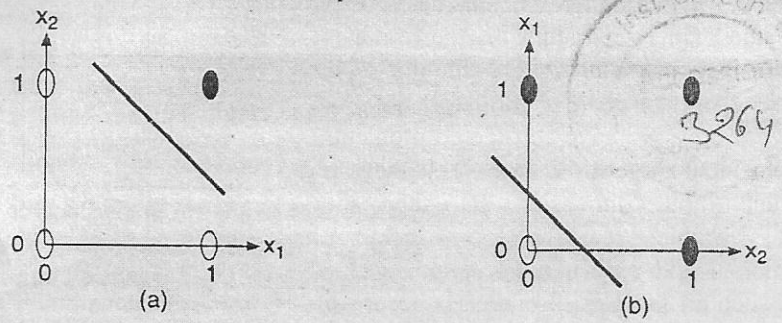


Fig. (5)

It can be seen that this is equivalent to the AND/OR logic gates as shown in Fig. (6).

x_1	x_2	*Output*
0	0	0
0	1	0
1	0	0
1	1	1

(a) AND logic gate

x_1	x_2	*Output*
0	0	0
0	1	1
1	0	1
1	1	1

(b) OR logic gate

Fig. (6)

Note : We have included a bias input in the examples-discussed. This is because if a bias weight is not included, the decision boundary will be forced to go through the origin. Sometimes, a third input component that is always 1 is also given (as in examples discussed after this article).

Limitation of Perceptron

If the classes are not linearly separable, learning will never reach a point where all inputs are classified properly. The most famous example of the perceptron's inability to solve problems with linearly non-separable vectors is the boolean XOR problem. It is again emphasized that most real-life classification problems are linearly non-separable. Therefore, they cannot be solved using simple perceptrons.

Example 1. *Obtain the decision boundary for binary AND function using Hebb rule after first training pair. The given binary inputs and targets are :*

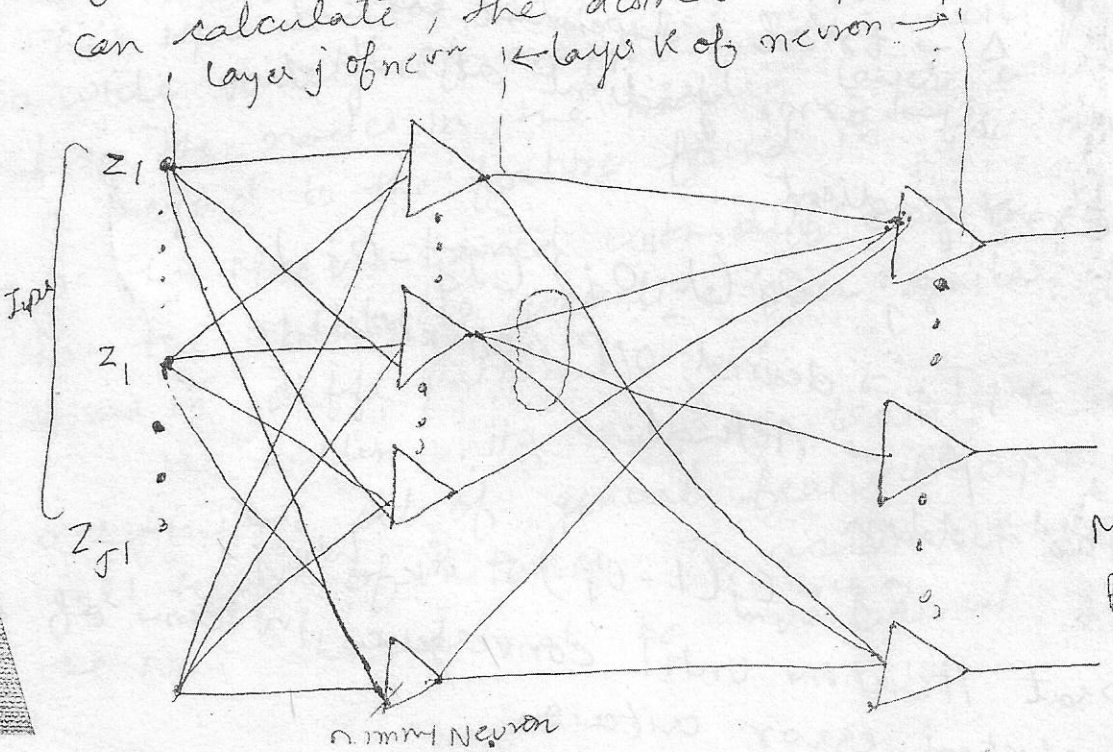
INPUT	*TARGET*
(x_1 x_2 1)	
(1 1 1)	1
(1 0 1)	0
(0 1 1)	0
(0 0 1)	0

* Neural Network learning :- The neural network has also been called the "connectionist". It contains a large no. of neuron like processing elements & a large no. of weighted connections b/w the elements. The weights on the connection encode the knowledge of a N/w. The various learning algorithm are

1. Backpropagation (Generalized Delta learning Rule)
2. Reinforcement learning
3. ART Networks

1. Backpropagation (Generalized Delta learning Rule) :-

- Backpropagation Network is Multilayer feedforward N/w with diff. transfer function in ANN & a more powerful learning rule. This learning rule is called "Backpropagation" which is a kind of gradient descent technique with backward error propagation.
- It is a Supervised learning method & is an implementation of Delta Rule. It requires a teacher that knows, or can calculate, the desired output for any given input



Backpropagation Algorithm

1) Weight Initialization – Set all the weights & na threshold to small random nos.

2) Calculation of Activation

The activation level O_j of hidden & o/p unit is determined by

$$O_j = F\left(\sum w_{ji} O_i - \theta_j\right)$$

$\theta_j \rightarrow$ threshold

$F \rightarrow$ sigmoid func.

$$F(a) = \frac{1}{(1+e^{-a})}$$

a = weight & input.

3) Weight Training :- a) Adjust weight by

$$w_{ji}(t+1) = w_{ji}(t) + \Delta w_{ji}$$

b) Weight change is computed by

$$\Delta w_{ji} = \eta \delta_j O_i.$$

$\Delta \rightarrow$ trial independent learning rate.

$\delta \rightarrow$ Error gradient at unit j.

c) Error gradient

$$\delta_j = O_j(1-O_j)(T_j - O_j)$$

$T_j \rightarrow$ desired o/p activation

$O_j \rightarrow$ Actual

d) for hidden

$$\delta_j = O_j(1-O_j)\sum \delta_k w_{kj}.$$

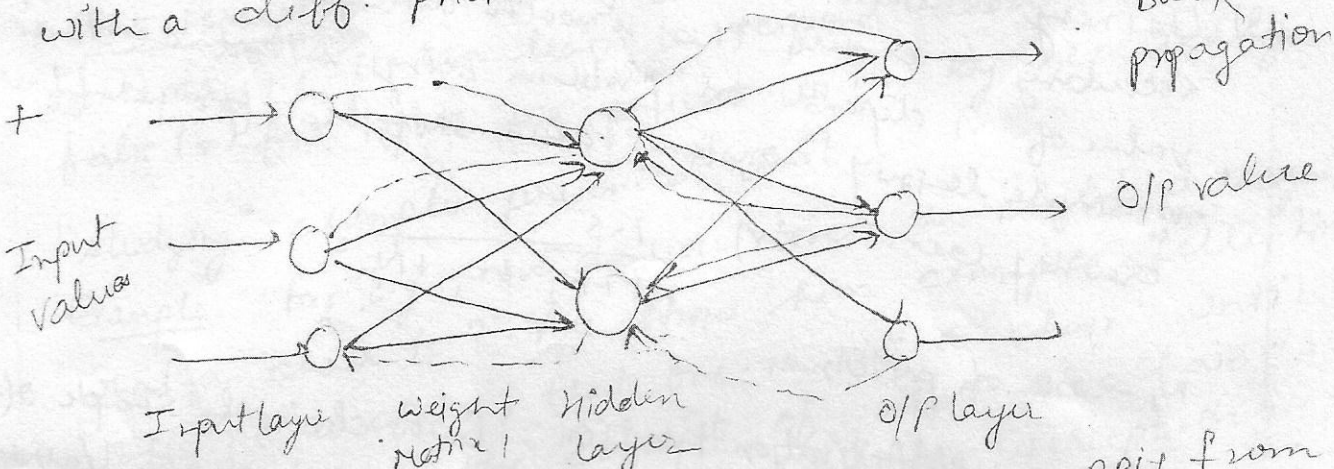
e) Repeat iteration until convergence in terms of the selected error criterion.

Some situation where a BP NN might be good idea. (8)

→ A large amount of input/output data is available, but you are not sure how to relate it to the o/p

→ The soln to the problem may change over time, within the bounds of the given input & output parameters

→ Most common application is in image processing. Some eg would be: identifying hand-written characters, matching a photograph ob a person's face with a diff. photo in dB.



→ The backpropagation network learn a mapping from a set of input pattern to a set of o/p patterns. This N/w can be designed & trained to accomplish a wide variety of mappings. This ability comes from the nodes in the hidden layer which learn to respond to the feature found in the input pattern

→ As the N/w is trained with diff. examples the N/w has the ability to generalize over similar features found in diff. pattern.

→ The hidden units must be trained to extract a sufficient set of general features applicable to all types of instances. To achive this goal, the N/w should not be overtrained.

Factors Affecting Backpropagation Training :-

1) Initialization of Weights - The initialization affects the complete solution. If all the weights start out with equal weight values & if the sol. requires that unequal weights be developed, the N/w may not be trained properly. There are 2 methods for training an ANN
   - → Self-organizing ANN
   - → Back propagation ANN

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

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

$N_1 \rightarrow$ no. of patterns

3) Steepness of Activation function :- Both choice & shape of the activation fun. strongly affect the speed of N/w learning

$$f(net) = \frac{2}{(1+\exp(-\lambda net))} - 1$$

Then $f'(net) = 2\lambda \exp(-\lambda net) / (1+\exp(-\lambda net))^2$

It reaches max value of $\frac{1}{2}\lambda$ at net = 0

Generally λ is kept at standard value of 1 & to control the learning speed using learning constant

4) N/w Size :- The N/w should have such a size as to capture the structure of the data & to model the problem. With some prior knowledge about the problem structure, one can form a good estimate of the proper N/w size

5) Threshold value :- A neuron generates the O/P if the weighted sum of the input exceeds the threshold value