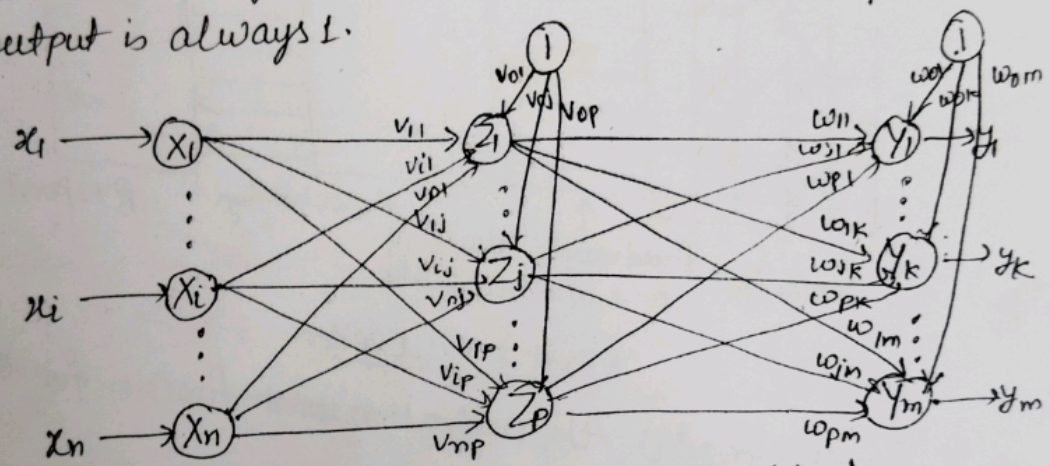
UNIT-2

III Neural-Networks-II (Back Propagation network)

Architecture: A back propagation neural network is a multilayer, feedforward neural network with an input layer, an output layer and a hidden layer. The neuron in the hidden and output layer have biases (similar to weights) which are connections from units whose output is always 1.



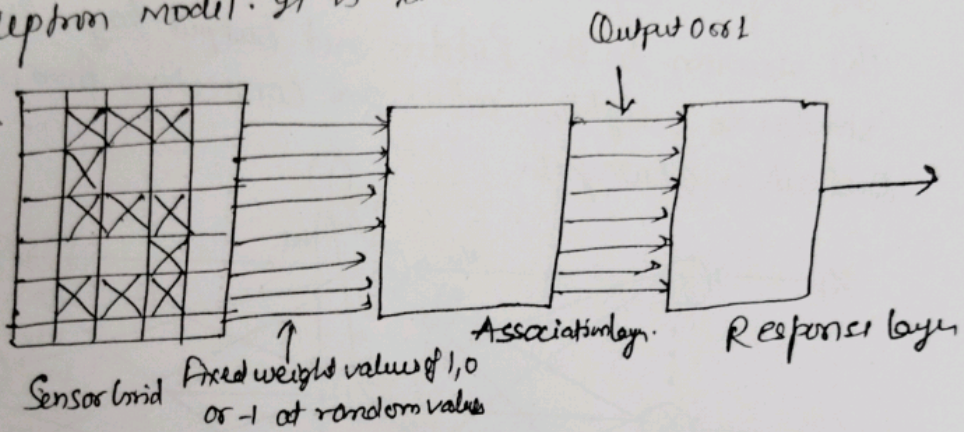
Architecture of Backpropagation Network -

The inputs are fed to the backpropagation net and the output obtained from it could be either binary (0,1) or bipolar (-1, +1). The activation function could be any function that increases monotonically and is also differentiable. The backpropagation implements the generalized delta rule. It is a gradient descendent method which minimizes the total squared error of the output of the network. Back Propagation training takes place in three stages.

(i) Feedforward of the input training pattern
(ii) Backpropagation of the associated error
(iii) Weight adjustment.

Activation function used in back propagation nets are:
(i) Binary Sigmoidal which has a range [0,1] and $y = \frac{1}{1+e^{-x}}$
(ii) Bipolar Sigmoidal which has a range [-1,+1] $y = \frac{2}{1+e^{-x}} - 1$

epic) Perceptron Model: First perceptron model was developed by Rosenblatt and was known as Mark1 perceptron model. It is shown in fig.



Mark 1 Perceptron.

In this model the activation ~~unit~~ function for the response unit is:

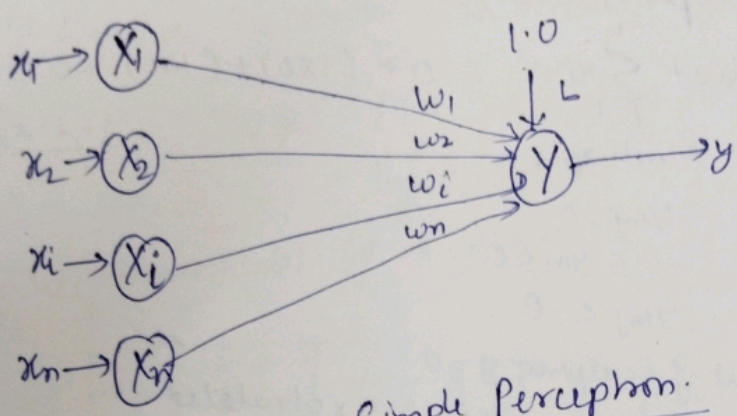
$$y = f(y_{in})$$

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

The weights that are connected between the association layer and response layer are adjusted during training. After presenting each training input vector, the net calculates the net input to the perceptron and then computes response of the perceptron. The output signal is compared with the target value to determine whether an error has occured. If an error has occured, then only the weights on the connections from units that sent a non-zero signal to the output unit will be adjusted.

ie - $w_i(new) = w_i(old) + \alpha x_i t$

where t is the target value 1 or -1 and α is the learning rate.



Simple Perceptron.

Topic-2

Perceptron Learning Algorithm.

Step-1: Initialize weights and biases (for simplicity weight with 0 & bias with 1)
Set learning rate α ($0 < \alpha \leq 1$)

Step-2. While stopping cond^n is false, do step 3 to 7.

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

Step-4: Set the input activations. $x_i = s_i$

Step-5. Compute net input to the perceptron and output response of the perceptron

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

$$y = \begin{cases} 1 & y_{in} > \theta \\ 0 & -\theta \leq y_{in} \leq \theta \\ -1 & y_{in} < \theta \end{cases}$$

Step-6. Update the bias and weights if the target is not equal to the output response.

if $t \neq y$
if $x_i \neq 0$
$w_i(new) = w_i(old) + \alpha x_i t$ $\quad i = 1, 2, 3 \cdots n$
else no change in weights
$b(new) = b(old) + \alpha t$

Step-7. Test for stopping cond^n. If no weight change in step 3 stop else continue.

Q. Develop a Perceptron network to implement an AND function.

Sol^{n}. For the first input sample $x_0=1$, $x_1=1$, $x_2=1$, $t=1$, the net input to the output neuron y_{in} is:

$$y_{in} = w_0 + \sum_{i=1}^{n} x_i w_i = 0 + (1\times 0) + (1\times 0) = 0$$

Applying activation function,

$$y = \begin{cases} 1 & y_{inj} > 0 \\ 0 & -0 \le y_{in} \le 0 \\ -1 & y_{inj} < -0 \end{cases}$$

Since the output of perceptron, $y=0$

Since $y \neq t$, the new weights are calculated as:

$w_0(new) = w_0(old) + \alpha \cdot t = 0 + (1\times 1) = 1$

$w_1(new) = w_1(old) + \alpha x_1 t = 0 + (1\times 1\times 1) = 1$

$w_2(new) = w_2(old) + \alpha x_2 t = 0 + (1\times 1\times 1) = 1$

The weight matrix after presenting the first sample, $W = [1\ 1\ 1]$.

For the second i/p sample, $x_0=1$, $x_1=-1$, $x_2=1$; $t=-1$, the net i/p to the o/p neuron is:

$$y_{in} = w_0 + \sum_{i=1}^{n} x_i w_i = 1 + (-1\times 1) + (1\times 1) = 1 \Rightarrow y = 1$$

Since $y \neq t$, the new weights are calculated as:

$w_0(new) = 1 + (1\times -1) = 0$

$w_1(new) = 1 + (1\times -1\times -1) = 2$

$w_2(new) = 1 + (1\times 1\times -1) = 0$

The weight matrix after presenting the second sample, $W = [0\ 2\ 0]$

For the third i/p sample, $x_0=1$, $x_1=1$, $x_2=-1$, $t=-1$, the net i/p to the output neuron, y_{in} is:

$$y_{in} = w_0 + \sum_{i=1}^{n} x_i w_i = 0 + (1\times 2) + (-1\times 0) = 2 \Rightarrow y = 1$$

Since $y \neq t$, the new weights are calculated as:

$w_0(new) = 0 + (1\times -1) = -1$

$w_1(new) = 2 + (1\times 1\times -1) = 1$

$w_2(new) = 0 + (1\times -1\times -1) = 1$

The weight matrix after presenting third sample, $W = [-1\ 1\ 1]$

For the forth i/p sample $x_0=1$, $x_1=-1$, $x_2=-1$, $t=-1$ the net i/p.

$y_{in} = -1 + (-1\times 1) + (-1\times 1) = -3 \Rightarrow y = -1$ Since $y=t$, no change in weight occurs. So $W = [-1\ 1\ 1]$

Back Propagation Algorithm.

Feedforward

Step-1. Initialize the weights.

Step-2 While stopping condition is false, execute steps 3 to 10.

Step-3. For each training pair $x:t$, do steps 4 to 9.

Step-4. Each input unit $X_i, i=1,2,\cdots,n$ receives the input signal, x_i and broadcasts it to the next layer.

Step-5. For each hidden layer neuron denoted as $Z_j, j=1,2,\cdots p$

$$Z_{inj} = v_{oj} + \sum_i x_i v_{ij}$$

$$Z_j = f(Z_{inj})$$

Broadcast z_j to the next layer.

Step-6. For each output neuron $Y_k, k=1,2\cdots m$

$$y_{ink} = w_{ok} + \sum_j z_j w_{jk}$$

$$y_k = f(y_{ink})$$

Backpropagation of error

Step-7. Compute δ_k for each output neuron, Y_k.

$$\delta_k = (t_k - y_k) f'(y_{ink})$$

$$\Delta w_{jk} = \alpha \delta_k z_j$$

$$\Delta w_{ok} = \alpha \delta_k \quad \text{since } z_0 = 1$$

Step-8 For each hidden neuron,

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk} \qquad j = 1,2,\cdots,p$$

$$\delta_j = \delta_{inj} f'(z_{inj})$$

$$\Delta v_{ij} = \alpha \delta_j x_0$$

$$\Delta v_{oj} = \alpha \delta_j$$

Weight update

Step-9. Update weights

$$w_{jk}(new) = w_{jk}(old) + \Delta w_{jk}$$

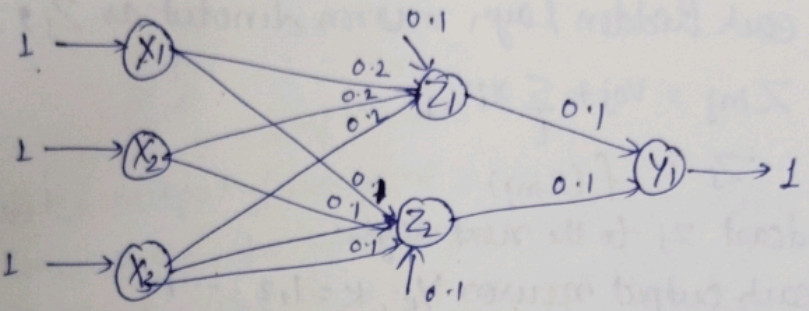
$$v_{ij}(new) = v_{ij}(old) + \Delta v_{ij}$$

Step-10. Test for stopping condition.

for ex:-

Q. Find the new weights when the net shown below is presented the input pattern (1 1 1) and the target output is 1. Use a learning rate of 0.1 and bipolar sigmoidal activation function. The bias is set to 1.

Sol^{n}. Activation function: $f(x) = \frac{2}{1+e^{-x}} - 1$ and $f'(x) = 0.5(1+f(x))(1-f(x))$



$$Z_{in1} = V_{01} + \sum_{i=1}^{3} x_i V_{i1} = 0.1 + 1\times0.2 + 1\times0.2 + 1\times0.2 = 0.7$$

$$Z_1 = f(Z_{in1}) = \frac{2}{1+e^{-0.7}} - 1 = 0.336$$

$$Z_{in2} = 0.1 + 0.1 + 0.1 + 0.1 = 0.4$$

$$Z_2 = f(Z_{in2}) = \frac{2}{1+e^{-0.4}} - 1 = 0.1974$$

$$y_{in} = 0.1(0.336) + 0.1(0.2) = 0.0536$$

$$Y_1 = f(y_{in}) = \frac{2}{1+e^{-0.0536}} - 1 = 0.02678$$

$$\delta_1 = (t - y_1) f'(y_{in})$$
$$= (1 - 0.02678) * 0.5 * (1 + 0.0268) * (1 - 0.0268)$$
$$= 0.4863$$

$$\delta_{in1} = \delta_1 w_{11} = 0.4866 * 0.1 = 0.04863$$

$$\delta = \delta_{in1} f'(z_{in1}) = 0.04866 * 0.5 * (1+0.336) * (1-0.336)$$
$$= 0.0216$$

$$\Delta V_{11} = \alpha * \delta * x_1 = 0.1 * 0.0216 * 1 = 0.00216$$
$$\Delta V_{21} = \alpha * \delta * x_2 = 0.1 * 0.0216 * 1 = 0.00216$$
$$\Delta V_{31} = \alpha * \delta * x_3 = 0.1 * (0.0216 * 1 = 0.00216$$
$$\Delta V_{01} = \alpha * \delta = 0.1 * 0.0216 = 0.00216$$

$$V_{11}(new) = V_{11}(old) + \Delta V_{11} = 0.2 + 0.00216 = 0.20216$$
$$V_{21}(new) = V_{21}(old) + \Delta V_{21} = 0.2 + 0.00216 = 0.20216$$

V_{31} (new) = V_{31} (old) + ΔV_{01} = 0.1 + 0.00216 = 0.10216

Thus new weights are $V_{11} = V_{21} = 0.20216$, $V_2 = 0.10216$

And $V_{01} = 0.10216$

Topics

Factors affecting Backpropagation technique-

① Choice of initial weights and Bias Random Initialization-
The choice of initial weights will influence whether the net reaches a global (or only a local) minimum of error and if so, how fast it converges. Since weight updation depends on the derivation so it is important to select initial weight such that neither activation function nor its derivative reaches to zero. Weights can be initialized within the range (-1, +1) or (-0.5, +0.5)

② Frequency of Weight Updates There are two pattern of learning per-pattern and per-Epoch (An epoch is one cycle through the entire set of training vectors). The training time can be reduced by involving different process for each input seprately.

③ Choice of learning Rule: A large learning rate leads to rapid learning but the weights may oscillate, while lower learning rate lead to slower learning. There are various method for adapting learning rate which are given as -

(i) A simple heuristic way suggests starting with a large learning rate and steadily decreasing it. Changes to the weight vector must be small in order to reduce oscillations or any divergence.

(ii) Another heuristic suggestion is to increase the learning rate in order to improve performance and to decrease the learning rate in order to worsen the performance.

④ Number of hidden Layers and Nodes: For a neural net with more than one hidden layer the calculation performed for a single layer is repeated for all the layers and are finally summed up. A three layer perceptron with $n(2n+1)$ nodes can be used to create any continuous likelihood fn required in a classifier.

⑤ Number of Training Pair: The enough training pairs for a network is given by the condition

$$\frac{W}{P} = e$$

Where P is the no. of training patterns.
W is the no. of weights to be trained.
e — expected ratio

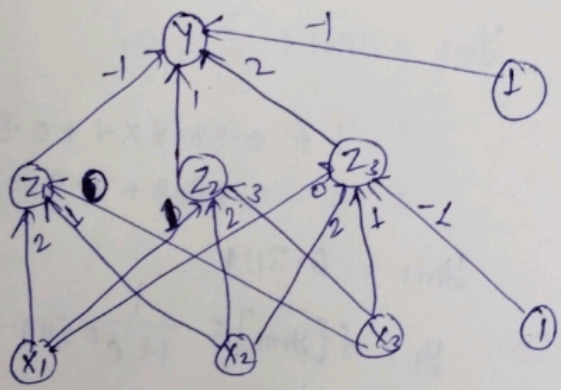
Application of Backpropagation:

① Image Processing
② Gesture Recognition
③ Online Motor fault Detection
④ Short-term load forecasting.

Effect of learning coefficient α: The learning coefficient cannot be negative because this would cause the change of weight vector to move away from ideal weight vector position. If the learning coefficient is zero, no learning take place and hence, the learning coefficient must be positive. If learning coefficient is equal to 2 then the NW is unstable and if learning coefficient is greater than 1, the weight vector will overshoot from its ideal position and oscillate. Hence the learning coefficient must

be between zero and one. Larger values for the learning coefficient are used when input data patterns are close to the ideal otherwise small values are used. If the nature of the input pattern is not known, it is better to use moderate values.

Q- Find the new weights when the n/w illustrated in fig is presented the input pattern [0.6 0.8 0] and the target output is 0.9. Use learning rate $\alpha = 0.3$ and use binary sigmoidal activation f^n.



Sol^n. Step 1. Initialize the weight and bias

$W = [-1 \; 1 \; 2]$, $W_0 = [-1]$

$$V = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 2 \\ 0 & 3 & 1 \end{bmatrix} \qquad V_0 = [0 \;\; 0 \;\; -1]$$

Step 2. for each training pair $X = [0.6 \;\; 0.8 \;\; 0]$

Step 3. for each training pair $X = [0.6 \;\; 0.8 \;\; 0]$

$t = [0.9]$

Feed Forward Stage-

Step-4 $Z_{in1} = V_{01} + \sum_{i=1}^{3} x_i V_{i1} = V_{01} + x_1 V_{11} + x_2 V_{21} + x_3 V_{31}$

$= 0 + 0.6 \times 2 + 0.8 \times 1 + 0 \times 0$

$= 1.2 + 0.8 = 2$

$$Z_{in2} = V_{02} + \sum_{i=1}^{3} x_i V_{i2} = V_{02} + x_1 V_{12} + x_2 V_{22} + x_3 V_{32}$$
$$= 0 + 0.6 + 0.8 \times 2 + 0 \times 3 = 2.2$$

$$Z_{in3} = V_{03} + \sum_{i=1}^{3} x_i V_{i3} = V_{03} + x_1 V_{13} + x_2 V_{22} + x_3 V_{23}$$
$$= -1 + 0.6 \times 0 + 0.8 \times 2 + 0 \times 1 = -1 + 1.6 = 0.6$$

$$Z_1 = f(Z_{in1}) = \frac{1}{1+e^{-2}} = 0.8808$$
$$Z_2 = f(Z_{in2}) = \frac{1}{1+e^{-2.2}} = 0.9002$$
$$Z_3 = f(Z_{in3}) = \frac{1}{1+e^{-0.6}} = 0.646$$

Step-5 - Calculate Y_{ink},

$$Y_{ink} = W_{0k} + \sum_{j=1}^{p} Z_j W_{jk}$$

$$y_{in1} = w_{01} + \sum_{j=1}^{3} z_j w_{j1}$$
$$= -1 + 0.8808 \times -1 + 0.9002 \times 1 + 0.646 \times 2$$
$$= -1 - 0.8808 + 0.9002 + 1.292$$
$$y_{in1} = 0.3114$$

So. $y_1 = f[y_{in1}] = \frac{1}{1+e^{-0.3114}} = 0.5772$

Back propagation of error.

Step-6. Calculate error information term δ_k.

$$\delta_k = (t_k - y_k) f'(y_{ink})$$
$$\delta_1 = (t_1 - y_1) f'(y_{in1})$$

for binary sigmoidal fn -

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

So $f'(y_{in1}) = f(y_{in1})(1 - f(y_{in1}))$
$$= 0.5772 \times (1 - 0.5772) = 0.2440$$

So. $\delta_1 = (0.9 - 0.5772)(0.2440) = 0.0708$

Step-7. Back propagation to the first layer ($i = 1, 2, 3$) we have.

to calculate δ_{inj}:

$$\delta_{inj} = \sum_{k=1}^{m} \delta_k w_{jk}$$

$\delta_{in1} = \delta_1 w_{11} \Rightarrow 0.0788 \times -1 = -0.0788$

$\delta_{in2} = \delta_1 w_{21} = 0.0788 \times 1 = 0.0788$

$\delta_{in3} = \delta_1 w_{31} = 0.0788 \times 2 = 0.1576$

to calculate error term in hidden layer:

$$\Delta = \delta_{inj} f'(z_{inj})$$

So $\Delta_1 = \delta_{in1} f'(z_{in1})$

$= \delta_{in1} \cdot f(z_{in1})(1 - f(z_{in1}))$

$= (-0.0788)(0.8808)(1 - 0.8808)$

$= -0.0083$

$\Delta_2 = \delta_{in2} f'(z_{in2})$

$= (0.0788)(0.9002)(1 - 0.9002) = 0.0071$

$\Delta_3 = \delta_{in3} f'(z_{in3})$

$= (0.1576)(0.646)(1 - 0.646) = 0.0361$

Step 8. Weight update.

$$\Delta v_{ij} = \alpha \Delta_j x_i$$

$x = [0.6 \quad 0.8 \quad 0]$

$\alpha = 0.3$

$\Delta v_{11} = \alpha \Delta_1 x_1 = 0.3 \times (-0.0083) \times 0.6 = -0.0015$

$\Delta v_{12} = \alpha \Delta_2 x_1 = 0.3 \times 0.0071 \times 0.6 = 0.0013$

$\Delta v_{13} = \alpha \Delta_3 x_1 = 0.3 \times 0.0361 \times 0.6 = 0.0065$

$\Delta v_{21} = \alpha \Delta_1 x_2 = 0.3 \times (-0.0083) \times 0.8 = -0.002$

$\Delta v_{22} = \alpha \Delta_2 x_2 = 0.3 \times 0.0071 \times 0.8 = 0.0017$

$\Delta v_{23} = \alpha \Delta_3 x_2 = 0.0087$

$\Delta v_{31} = \alpha \Delta_1 x_3 = 0$

$\Delta v_{32} = \alpha \Delta_2 x_3 = 0$

$\Delta v_{33} = \alpha \Delta_3 x_3 = 0$

$\Delta V_{01} = 0.3 \times -0.0083 = -0.0025$

$\Delta V_{02} = \alpha \Delta 2 = 0.3 \times 0.0071 \Rightarrow 0.0021$

$\Delta V_{03} = \alpha \Delta_3 = 0.3 \times 0.0361 = 0.0108$

$V_{new} = V_{old} + \Delta V_1$

$V_{11}(new) = V_{11}(old) + \Delta V_{11} = 2 - 0.0015 = 1.9985$

$V_{12}(new) = V_{12}(old) + \Delta V_{12} = 1 + 0.0013 = 1.0013$

$V_{13}(new) = V_{13}(old) + \Delta V_{13} = 0 + 0.065 = 0.065$

$V_{21}(new) = V_{21}(old) + \Delta V_{21} = 1 - 0.002 = 0.998$

$V_{22}(new) = V_{22}(old) + \Delta V_{22} = 2 + 0.0017 = 2.0017$

$V_{23}(new) = V_{23}(old) + \Delta V_{23} = 2 + 0.0087 = 2.0087$

$V_{31}(new) = V_{31}(old) + \Delta V_{31} = 0 + 0 = 0$

$V_{32}(new) = V_{32}(old) + \Delta V_{32} = 3 + 0 = 3$

$V_{33}(new) = V_{33}(old) + \Delta V_{33} = 1 + 0 = 1$

$$\therefore V = \begin{bmatrix} 1.9985 & 1.0013 & 0.085 \\ 0.998 & 2.0017 & 2.0087 \\ 0 & 3 & 1 \end{bmatrix}$$

$\Delta w_{ok} = \alpha \delta_k z_j$

$\Delta w_{11} = \alpha \delta_1 z_1 = 0.3 \times 0.0788 \times 0.8808 = 0.0208$

$\Delta w_{12} = \alpha \delta_1 z_2 = 0.3 \times 0.0788 \times 0.9002 = 0.0212$

$\Delta w_{13} = \alpha \delta_1 z_3 = 0.3 \times 0.0788 \times 0.646 = 0.0153$

$\Delta w_{11}(new) = w_{11}(old) + \Delta w_{11} = 1 + 0.0208 = 0.9792$

$\Delta w_{12}(new) = w_{12}(old) + \Delta w_{12} = 1 + 0.0212 = 1.0212$

$\Delta w_{13}(new) = w_{13}(old) + \Delta w_{13} = 2 + 0.0153 = 2.0153$

$\Delta w_{01} = \alpha \delta_1 = 0.3 \times 0.0788 = 0.02364$

$w = [0.9792 \quad 1.0212 \quad 2.0153]$

$$R \circ S = \begin{matrix} x_1 \\ x_2 \\ x_3 \end{matrix} \begin{matrix} \begin{matrix} z_1 & z_2 & z_3 \end{matrix} \\ \begin{bmatrix} 0.5 & 0.4 & 0.5 \\ 0.5 & 0.8 & 0.9 \\ 0.6 & 0.6 & 0.7 \end{bmatrix} \end{matrix}$$

$$\mu_{R \circ S}(x_1, z_1) = \max\left(\min\left(\overset{x_1 y_1}{0.5}, \overset{y_1 z_1}{0.6}\right), \min\left(\overset{x_1 y_2}{0.1}, \overset{y_2 z_2}{0.5}\right)\right)$$
$$= \max(0.5, 0.1) = 0.5$$

$$\mu_{R \circ S}(x_1, z_2) = \max(\min(x_1 y_1, y_1 z_2), \min(x_1 y_2, y_2 z_2))$$
$$= \max(\min(0.5, 0.9), \min(0.1, 0.8))$$
$$= \max(0.4, 0.1) = 0.4$$

And so on.

Q. Consider a set $P = \{P_1, P_2, P_3, P_4\}$ of four varieties of Paddy plants, set $D = \{D_1, D_2, D_3, D_4\}$ of the various diseases affecting the plants and $S = \{S_1, S_2, S_3, S_4\}$ be the common symptoms of the diseases.
Let $\tilde{R}$ be a relation on $P \times D$ and $\tilde{S}$ be a relation on $D \times S$

$$\tilde{R} = \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{matrix} \begin{matrix} \begin{matrix} D_1 & D_2 & D_3 & D_4 \end{matrix} \\ \begin{bmatrix} 0.6 & 0.6 & 0.9 & 0.8 \\ 0.1 & 0.2 & 0.9 & 0.8 \\ 0.9 & 0.3 & 0.4 & 0.8 \\ 0.9 & 0.8 & 0.1 & 0.2 \end{bmatrix} \end{matrix} \qquad \tilde{S} = \begin{matrix} D_1 \\ D_2 \\ D_3 \\ D_4 \end{matrix} \begin{matrix} \begin{matrix} S_1 & S_2 & S_3 & S_4 \end{matrix} \\ \begin{bmatrix} 0.1 & 0.2 & 0.7 & 0.9 \\ 1 & 1 & 0.4 & 0.6 \\ 0 & 0 & 0.5 & 0.9 \\ 0.9 & 1 & 0.8 & 0.2 \end{bmatrix} \end{matrix}$$

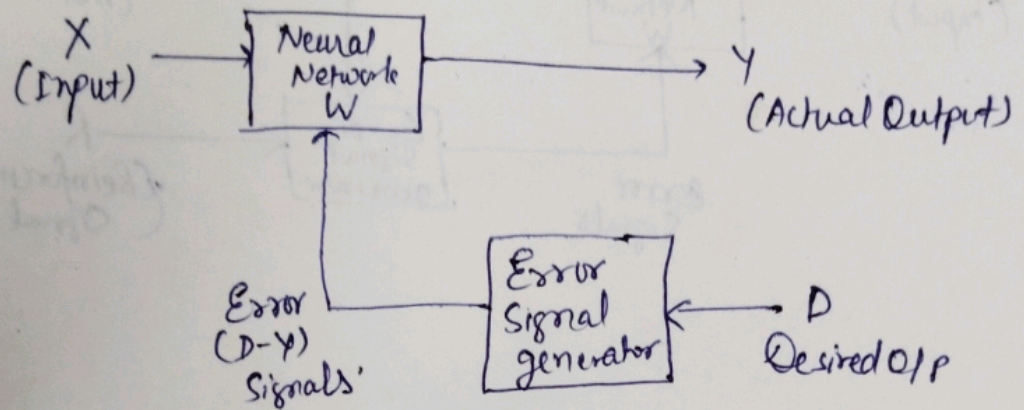
Obtain the association of the plants with the different symptoms of the diseases using max-min composition.

Sol^n.

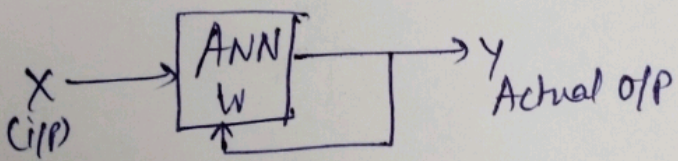
$$R \circ S = \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{matrix} \begin{matrix} \begin{matrix} S_1 & S_2 & S_3 & S_4 \end{matrix} \\ \begin{bmatrix} 0.8 & 0.8 & 0.8 & 0.9 \\ 0.8 & 0.8 & 0.8 & 0.9 \\ 0.8 & 0.8 & 0.8 & 0.9 \\ 0.8 & 0.8 & 0.7 & 0.9 \end{bmatrix} \end{matrix}$$

Learning → The main property of an ANN is its capability to learn. Learning or training is a process by means of which a neural network adapts itself to a stimulus by making proper parameter adjustments, resulting in the production of desired response.

① Supervised Learning.



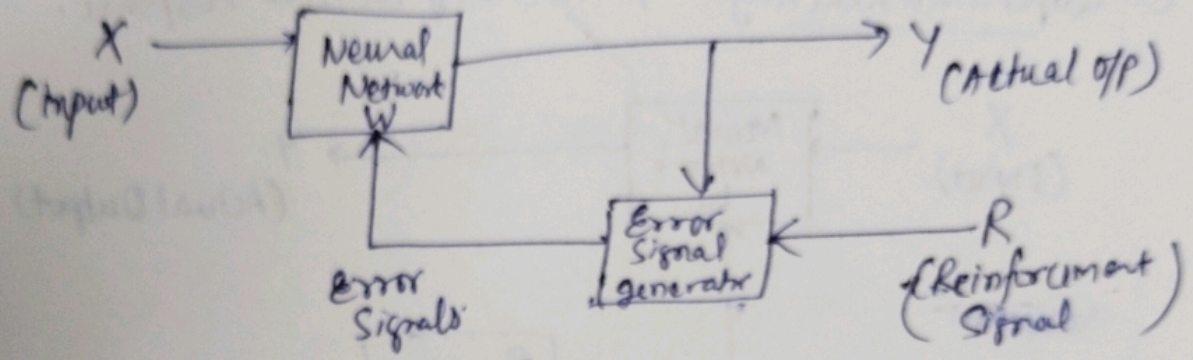
② Unsupervised Learning. Forms cluster based on similarity & dissimilarity. (Tadpole) (Fish) (automatically). If a input does not belong to any of the cluster then a new cluster is formed.



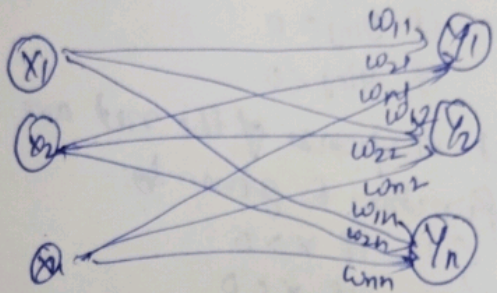
③ Reinforcement Learning. This learning process is similar to supervised learning. In the case of supervised learning, the correct target output values are known for each input pattern. But in some cases, less information might be available.

for example. the network might be told that its actual output is only "50%" correct or so. Thus, here only critic information is available, not the exact information. The learning based on this critic information is called

reinforcement learning and the feedback unit is called reinforcement signal.



Autoassociative Net - For an autoassociative net, the training input and target output vectors should be identical. The process of training is often called storing the vectors, which may be binary or bipolar. A stored vector can be retrieved from distorted partial (noisy) input, if the input is sufficiently similar to it. The performance of the net is judged by its ability to reproduce a stored pattern from noisy input;



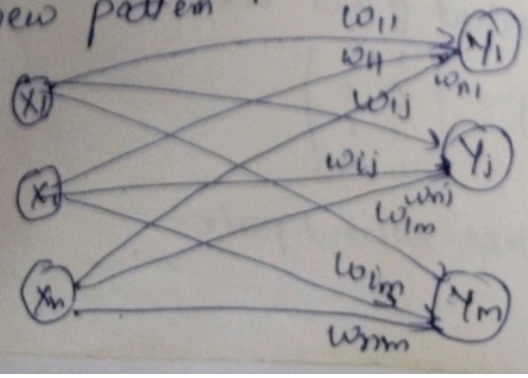
Associative network

In practice weights are adjusted using the formula

$$w = \sum_{p=1}^{P} s^T(p) s(p)$$

Application - An autoassociative neural net can be used to determine whether an input vector "known" or "unknown".

Heteroassociative Memory Neural N/w - Associative memory neural n/w are nets in which the weights are determined in such a way that the net can store a set of P pattern associations. Each association is a pair of vectors $(s(p), t(p))$ with $p = 1, 2, \cdots, P$. Each vector $s(p)$ has n-tuples and the vector $t(p)$ has m types. The weights may be found using Hebb rule or Delta rule. The net will find an appropriate output vector that corresponds to an input vector x, that may be either one of the stored pattern $s(p)$ or a new pattern.



Algo - Step-1. Initialize weights using either Hebb rule or Delta (Delta rule)

Step-2 For each input vector, do Step 3 to 5.

Step-3 Set the activations for input layer units equal to (current) input vector x_i

Step-4 Compute net input to output units.

$$y_{inj} = \sum_i x_i w_{ij}$$

Step-5 Determine the activation of the output units

$$y_j = \begin{cases} 1 & \text{if } y_{inj} > 0 \\ 0 & \text{if } y_{inj} = 0 \\ -1 & \text{if } y_{inj} < 0 \end{cases}$$

Note - if the target response of the net are binary, a suitable activation function is given by:

$$f(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \le 0 \end{cases}$$

For eg. - Train a heteroassociative net by outer product rule for input row vectors $s = (x_1, x_2, x_3, x_4)$ to output row vectors $t = (t_1, t_2)$:

	s_1	s_2	s_3	s_4	t_1	t_2
s	1	0	0	0	t	1	0
s	1	1	0	0	t	1	0
s	0	0	0	1	t	0	1
s	0	0	1	1	t	0	1

Soln. The weight matrix to store first s:t pair is given by outer product of the input vector,

$s = (1\ 0\ 0\ 0)$ & $t = (1, 0)$

The outer product of this vector pair is given by,

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}_{4\times1} \begin{bmatrix} 1 & 0 \end{bmatrix}_{1\times2} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Similarly to store second pair -

Hebb Learning Extra Detail. 1949

Hebb's original statement was:

"When an axon of cell A is near enough to excite a cell B and repeatdly or persistently takes place in firing it, some growth process or metabdic change takes place in one or both cells such that A's efficiency, as one of the cell firing B, is increased."

Hebb Net. This net consists of a single layer feedforward neural network. This net has a fixed bias value of 1). The input and the output patterns must be of bipolar form.

Linear Seprability - If it is possible to find the weights so that all of the training input vectors for which the correct response is 1 lie on one side of the boundary, ~~then the problem is~~ and all of the training patters for which the correct response is -1 lie on the either side of the boundary, then the problem is linearly seprable otherwise the problem is linearly non separable.

Delta Rule - Delta Rule is also reffered to as least mean square rule. This rule is applicable for the supervised training of neural network, according to this rule, the change in weight is directly proportional to the error signal $(t_j - y_i)$ and the input. The error signal is equal to the difference b/w desired output value and

actual output value of the neuron.

The change in weight vector, ΔW_j can be represented as:

$$\Delta W_j = c(t_j - y_j) X^T$$

where c - Learning constant
t_j - desired value at the j^{th} output neuron.
y_j - Computed output value at the j^{th} o/p neuron.