Module 2.2 : MLP & RBF Learning

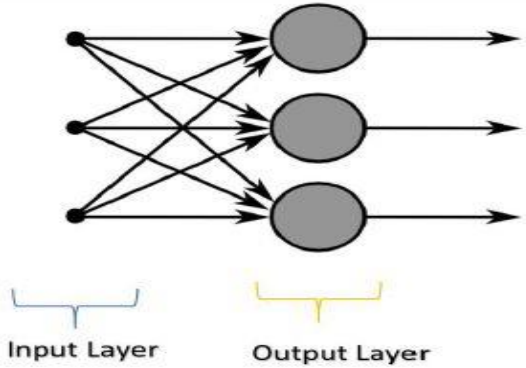
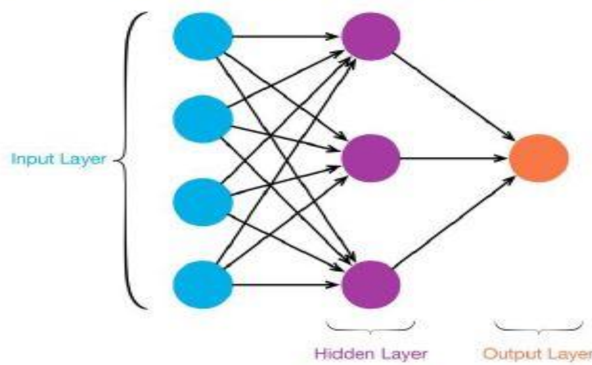
Limitation of Perceptron

Perceptron networks have several limitations.

- First, the output values of a perceptron can take on only one of two values (0 or 1) due to the hard-limit transfer function.
- Second, perceptron can only classify linearly separable sets of vectors.
- Networks with more than one perceptron can be used to solve more difficult problems

A Multi Layer Perceptron (MLP) contains one or more hidden layers (apart from one input and one output layer). While a single layer perceptron can only learn linear functions, a multi layer perceptron can also learn non – linear functions

Single layer Vs Multilayer Perceptron

Single Layer Feed-Forward Neural Network	Multi Layer Feed-Forward Neural Network
Layer is formed by taking processing element & combining it with other processing element.	It is formed by interconnection of several layers.
Input & output are linked with each other.	There are multiple layers between input & output layers which are known as hidden layers.
Inputs are connected to the processing nodes with various weights resulting series of output one per node.	Input layers receives input & buffers input signal, output layer generates output.
Zero hidden layers are present.	Zero to several hidden layers are in a network.
Not efficient in certain areas.	More the hidden layers, more the complexity of networks, but efficient output is produced.
	

Multilayer Perceptron

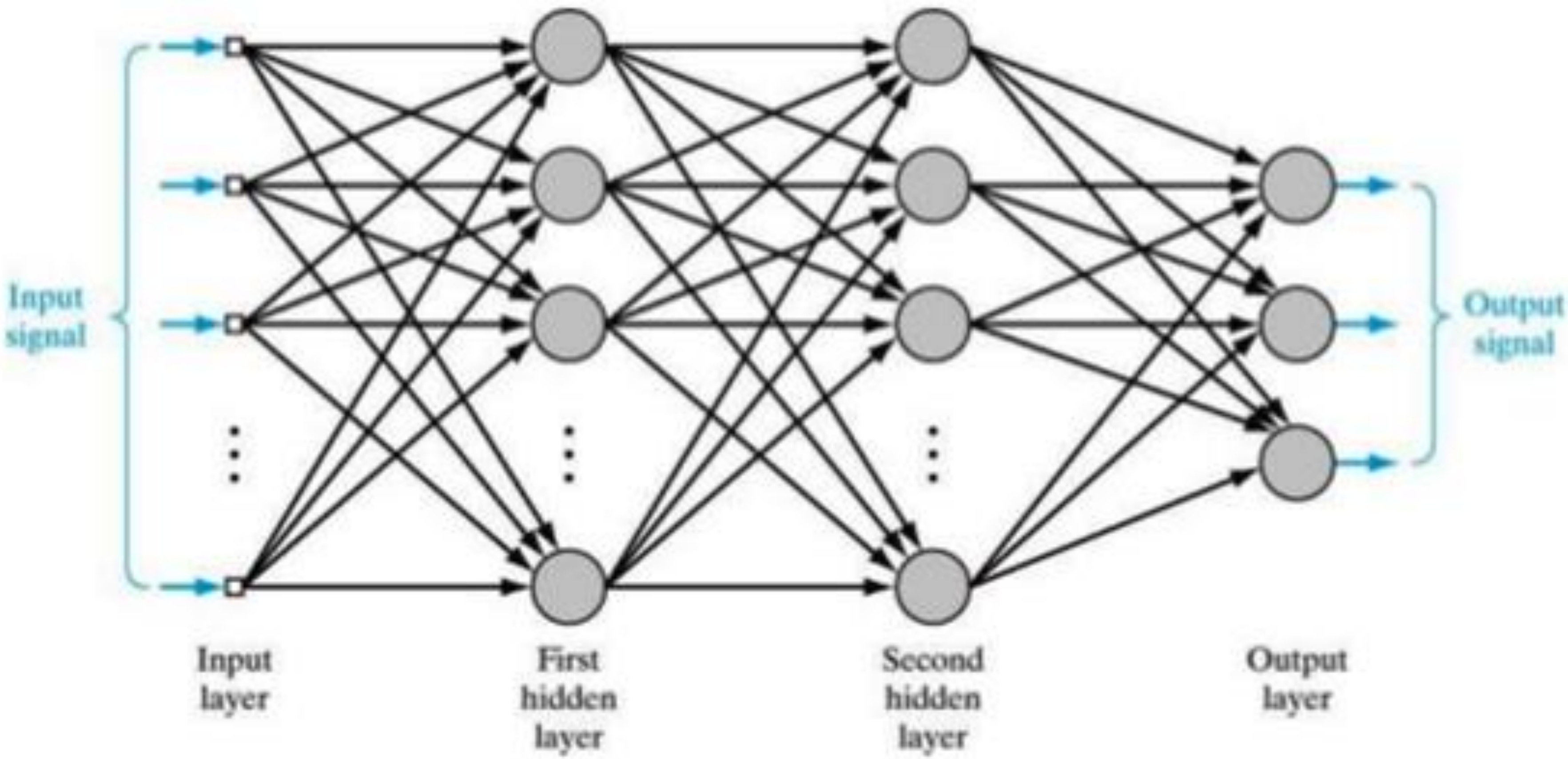
The multilayer perceptron (MLP) is proposed to overcome the limitations of the perceptron

- That is, building a network that can ***solve nonlinear*** problems.

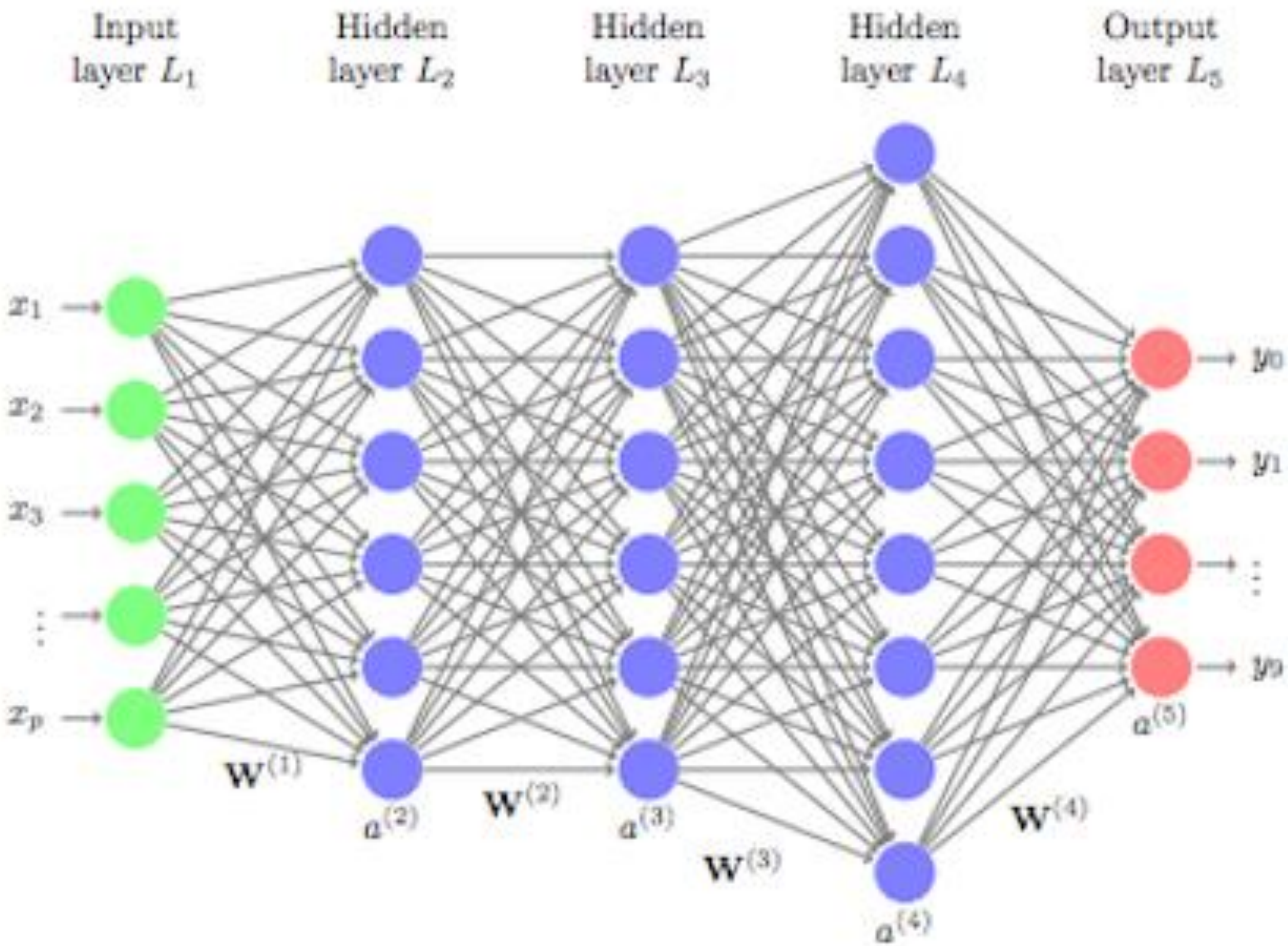
The basic features of the multilayer perceptrons:

- Each neuron in the network includes a nonlinear activation function that is *differentiable*.
- The network contains one or more layers that are *hidden* from both the input and output nodes.
- The network exhibits a high degree of *connectivity*.

Multilayer Perceptron

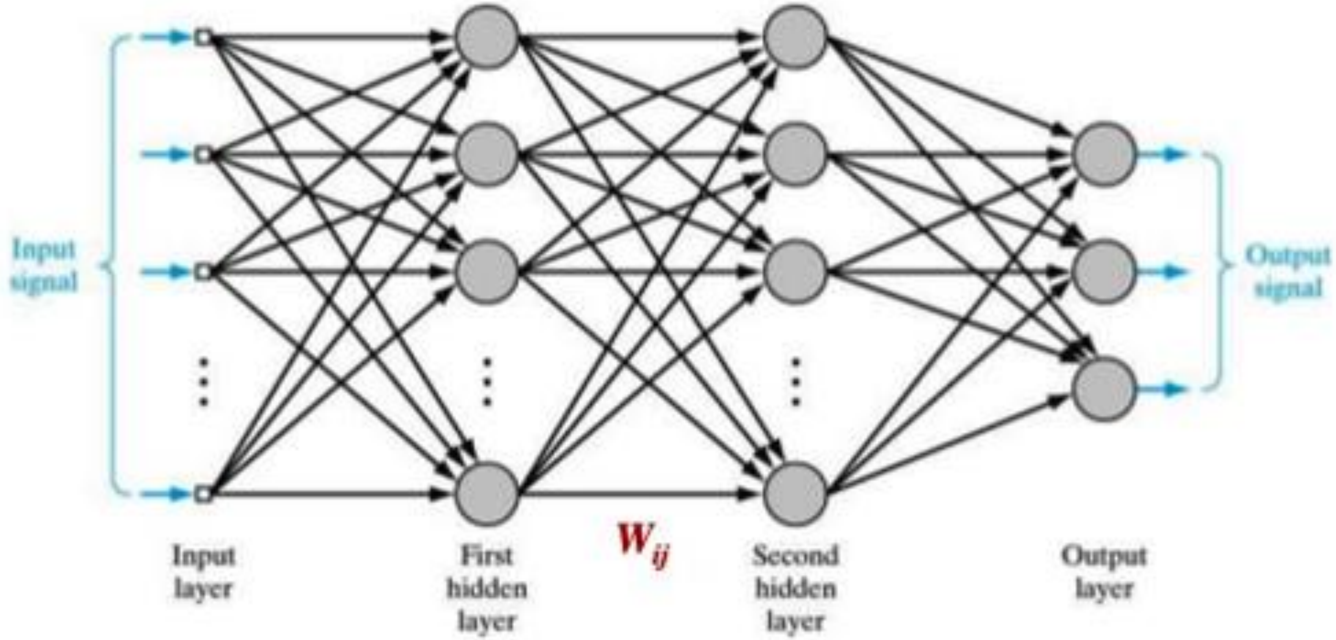


Multilayer Perceptron



Multilayer Perceptron

Weight Dimensions



If network has n units in layer i, m units in layer $i+1$, then the weight matrix W_{ij} will be of dimension $m \times (n+1)$.

Multilayer Perceptron

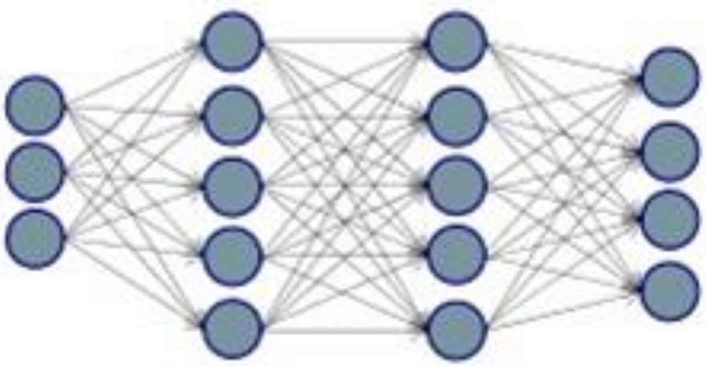
Number of neuron in the output layer

Pedestrian

Car

Motorcycle

Truck



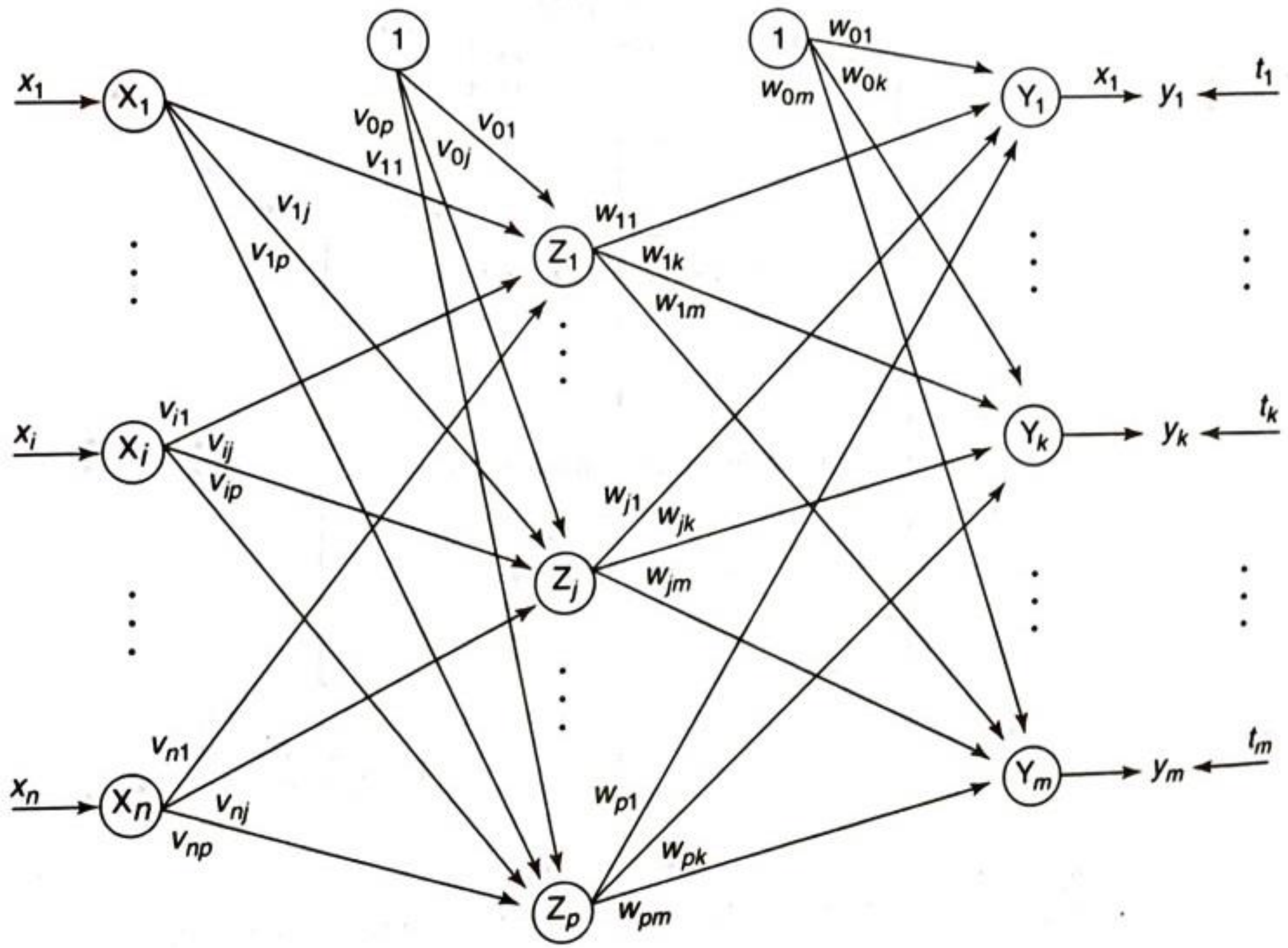
$$\begin{array}{cccc} \begin{bmatrix}1\\0\\0\\0\end{bmatrix} & \begin{bmatrix}0\\1\\0\\0\end{bmatrix} & \begin{bmatrix}0\\0\\1\\0\end{bmatrix} & \begin{bmatrix}0\\0\\0\\1\end{bmatrix} \\ \textbf{Pedestrain} & \textbf{Car} & \textbf{Moto} & \textbf{Truck} \end{array}$$

Multilayer Perceptron

- **Function of the Hidden neurons**
  - The ***hidden neurons*** play a critical role in the operation of a multilayer perceptron; they act as ***feature detectors***.
  - The ***nonlinearity*** transform the input data into a ***feature space*** in which data ***may be separated easily***.

- **Credit Assignment Problem**
  - Is the problem of assigning a ***credit*** or a ***blame*** for *overall outcomes* to the internal decisions made by the computational units of the distributed learning system.
  - The ***error-correction learning*** algorithm is easy to use for training single layer perceptrons. But its ***not*** easy to use it for a ***multilayer*** perceptrons,
    - the ***backpropagation*** algorithm ***solves*** this problem.

Back Propagation



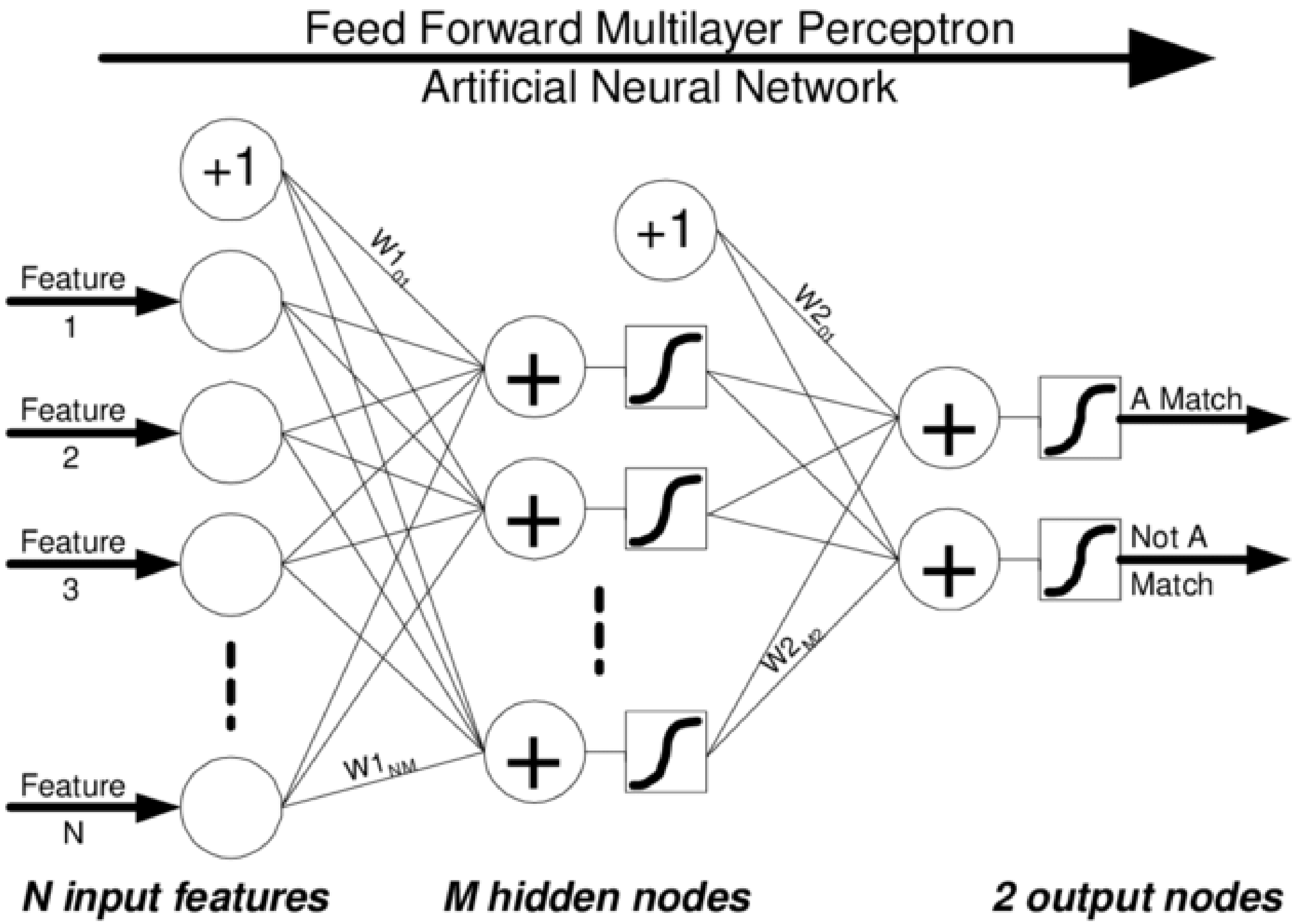
Back Propagation Learning Rule

Training of BPN is done in three phases

1. Feedforward of the i/p training pattern, it propagates through the network layer by layer ,until it reaches the output
2. Calculation of error and back-propogation of the error through the network layer by layer
3. Weight Updation-Similar to LMS (instant error and gradient descent method)

Testing Computation of feedforward phase only

Phase-1 Feed-forward



Phase-1 Feed-forward

Step 0: Initialize weights and learning rate (take some small random values).

Step 1: Perform Steps 2–9 when stopping condition is false.

Step 2: Perform Steps 3–8 for each training pair.

Feed-forward phase (Phase I):

Step 3: Each input unit receives input signal x_i and sends it to the hidden unit ($i = 1$ to n).

Step 4: Each hidden unit $z_j (j = 1$ to $p)$ sums its weighted input signals to calculate net input:

$$z_{inj} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

Calculate output of the hidden unit by applying its activation functions over z_{inj} (binary or bipolar sigmoidal activation function):

$$z_j = f(z_{inj})$$

and send the output signal from the hidden unit to the input of output layer units.

Step 5: For each output unit y_k ($k = 1$ to m), calculate the net input:

$$y_{ink} = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$$

Activation function at o/p $\quad y_k = f(y_{ink})$

BPN Activation Function

Differentiability **is the only** ***requirement*** **that an activation function has to satisfy in the BP Algoruthm.**

- This is required to compute the δ for each neuron.

Sigmoidal functions are commonly used, since they satisfy such a condition:

- Logistic Function

$$\boxed{\varphi(v) = \frac{1}{1+\exp(-av)}, \qquad a > 0} \Rightarrow \boxed{\varphi'(v) = \frac{a\exp(-av)}{1+\exp(-av)} = a\varphi(v)[1-\varphi(v)]}$$

- Hyperbolic Tangent Function

$$\boxed{\varphi(v) = a\tanh(bv), \qquad a, b > 0} \Rightarrow \boxed{\varphi'(v) = \frac{b}{a}[a-\varphi(v)][a+\varphi(v)]}$$

Sigmoidal Activation Function

- *Bipolar sigmoid function*: This function is defined as

$$f(x) = \frac{2}{1+e^{-\lambda x}} - 1 = \frac{1-e^{-\lambda x}}{1+e^{-\lambda x}}$$

where λ is the steepness parameter and the sigmoid function range is between -1 and $+1$. The derivative of this function can be

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

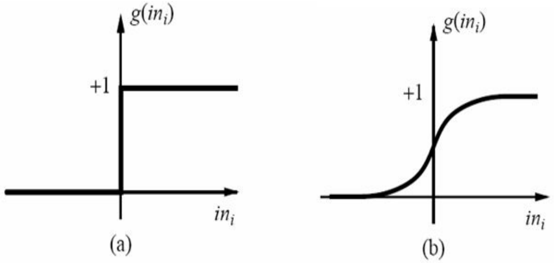
The bipolar sigmoidal function is closely related to hyperbolic tangent function, which is written as

$$h(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = \frac{1-e^{-2x}}{1+e^{-2x}}$$

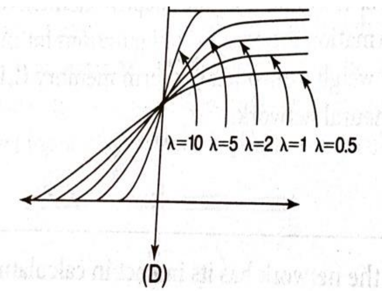
The derivative of the hyperbolic tangent function is

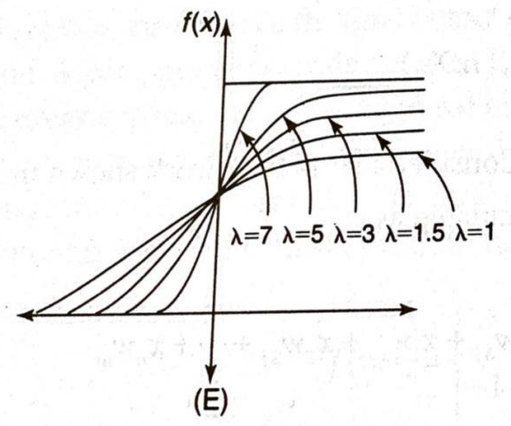
$$h'(x) = [1+h(x)][1-h(x)]$$

If the network uses a binary data, it is better to convert it to bipolar form and use the bipolar sigmoidal act
function or hyperbolic tangent function.



step function sigmoid function

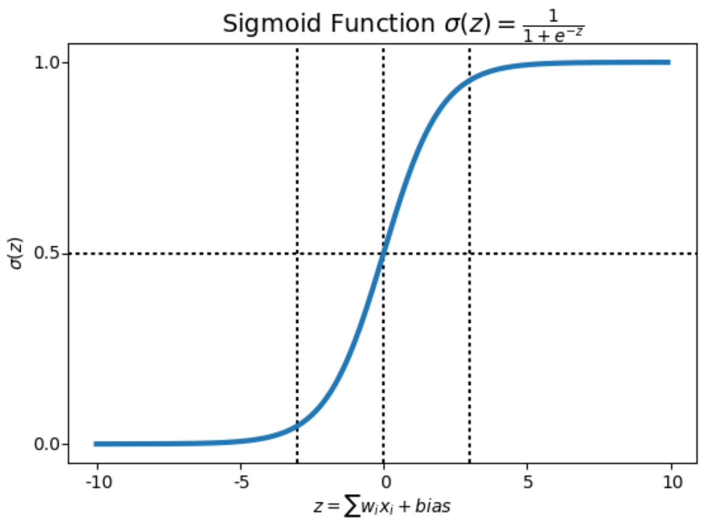


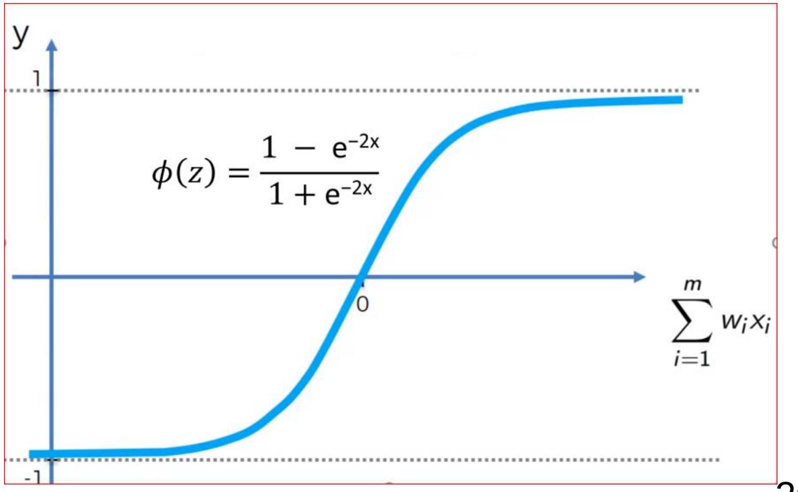


Sigmoidal Activation Function

Commonly used activation functions are binary sigmoidal and bipolar sigmoidal
These functions are used in the BPN because of following characteristics

- Continuity
- Differentiability
- Non decreasing monotony
- The range of binary Sigmoidal is from 0 to 1
- And for Bipolar from -1 to +1

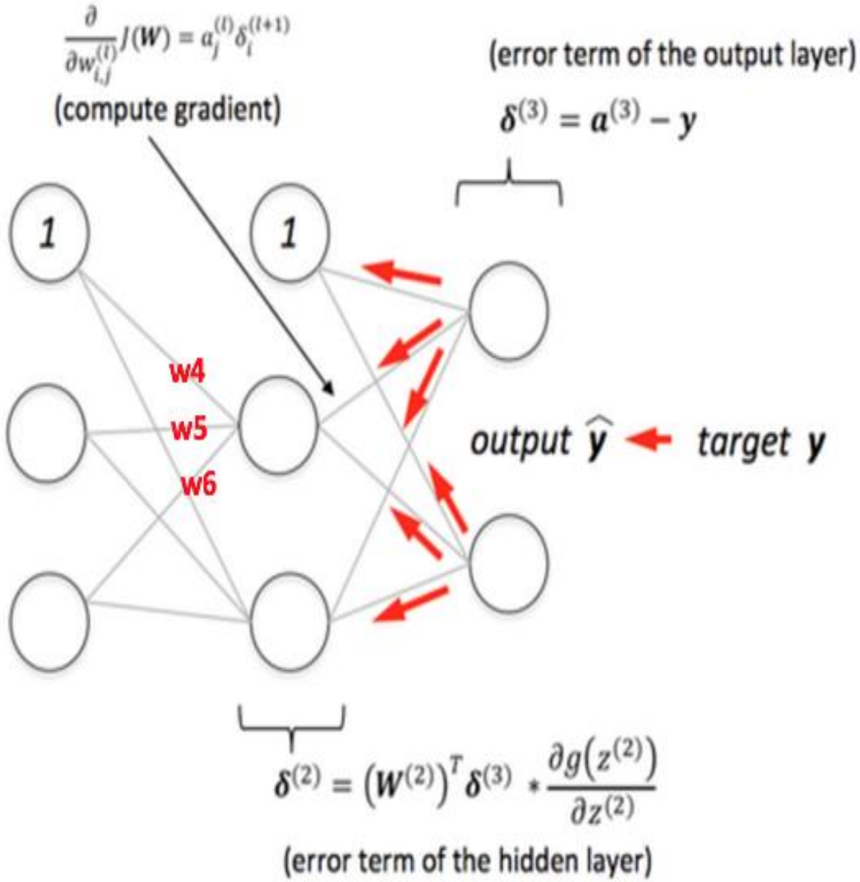




Phase-I1 Back Propagation of Error

Phase 2 & 3

Backpropagation
+
Weights Adjusted



Phase-I1 Back Propagation of Error

Back-propagation of error (Phase II):

Step 6: Each output unit $y_k (k = 1 \text{ to } m)$ receives a target pattern corresponding to the input training pattern and computes the error correction term:

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

$$\Delta w_{jk} = \alpha \delta_k z_j; \quad \Delta w_{0k} = \alpha \delta_k$$

Also, send δ_k to the hidden layer backwards.

Step 7: Each hidden unit $(z_j, j = 1 \text{ to } p)$ sums its delta inputs from the output units:

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

The term δ_{inj} gets multiplied with the derivative of $f(z_{inj})$ to calculate the error term:

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

The derivative $f'(z_{inj})$ can be calculated as discussed in Section 2.3.3 depending on whether binary or bipolar sigmoidal function is used. On the basis of the calculated δ_j, update the change in weights and bias:

$$\Delta v_{ij} = \alpha \delta_j x_i; \quad \Delta v_{0j} = \alpha \delta_j$$

Weight and bias updation (Phase III):

Phase-I1I Weight & Bias update

Phase III

Step 8: Each output unit (y_k, $k = 1$ to m) updates the bias and weights:

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

$$w_{0k}(\text{new}) = w_{0k}(\text{old}) + \Delta w_{0k}$$

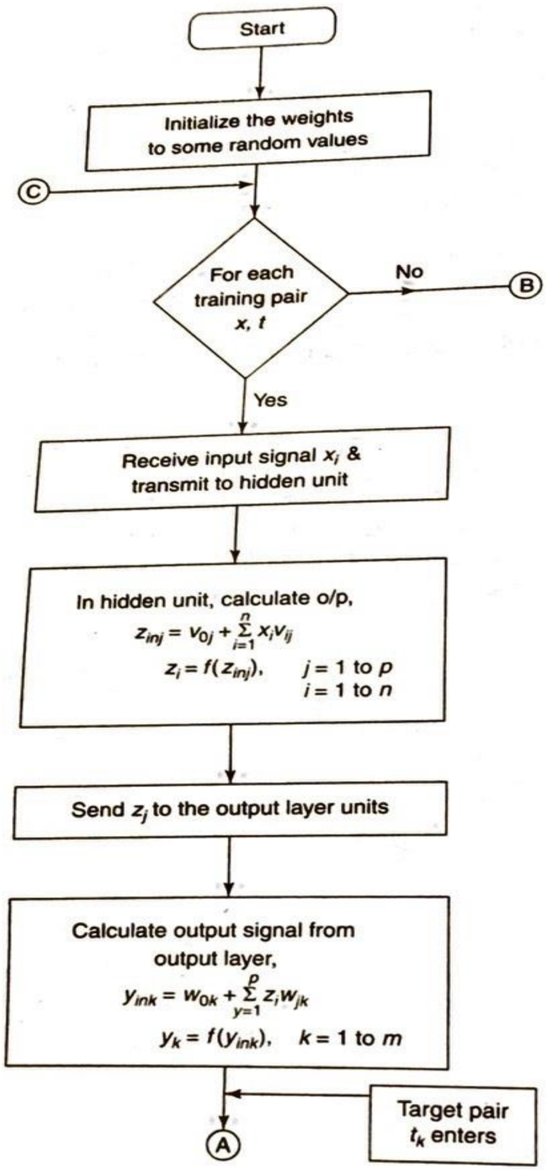
Each hidden unit (z_j, $j = 1$ to p) updates its bias and weights:

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

$$v_{0j}(\text{new}) = v_{0j}(\text{old}) + \Delta v_{0j}$$

Step 9: Check for the stopping condition. The stopping condition may be certain number of epochs reached or when the actual output equals the target output.

Training Process(Flow chart)



x = input training vector $(x_1, ..., x_i, ..., x_n)$

t = target output vector $(t_1, ..., t_k, ..., t_m)$

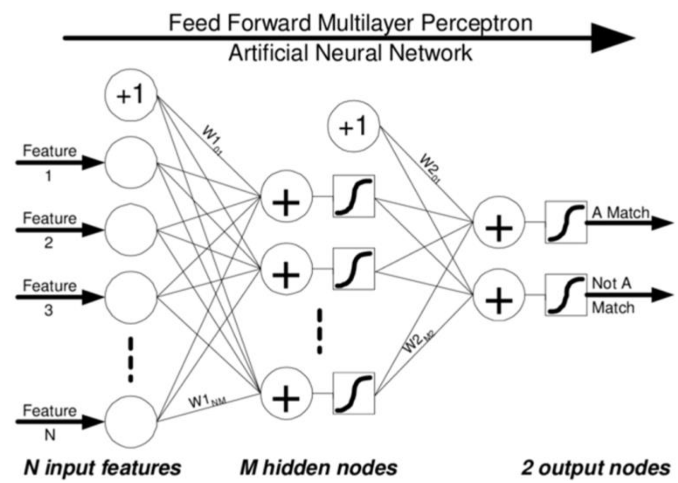
α = learning rate parameter

x_i = input unit i. (Since the input layer uses identity activation function, the input and output signals here are same.)

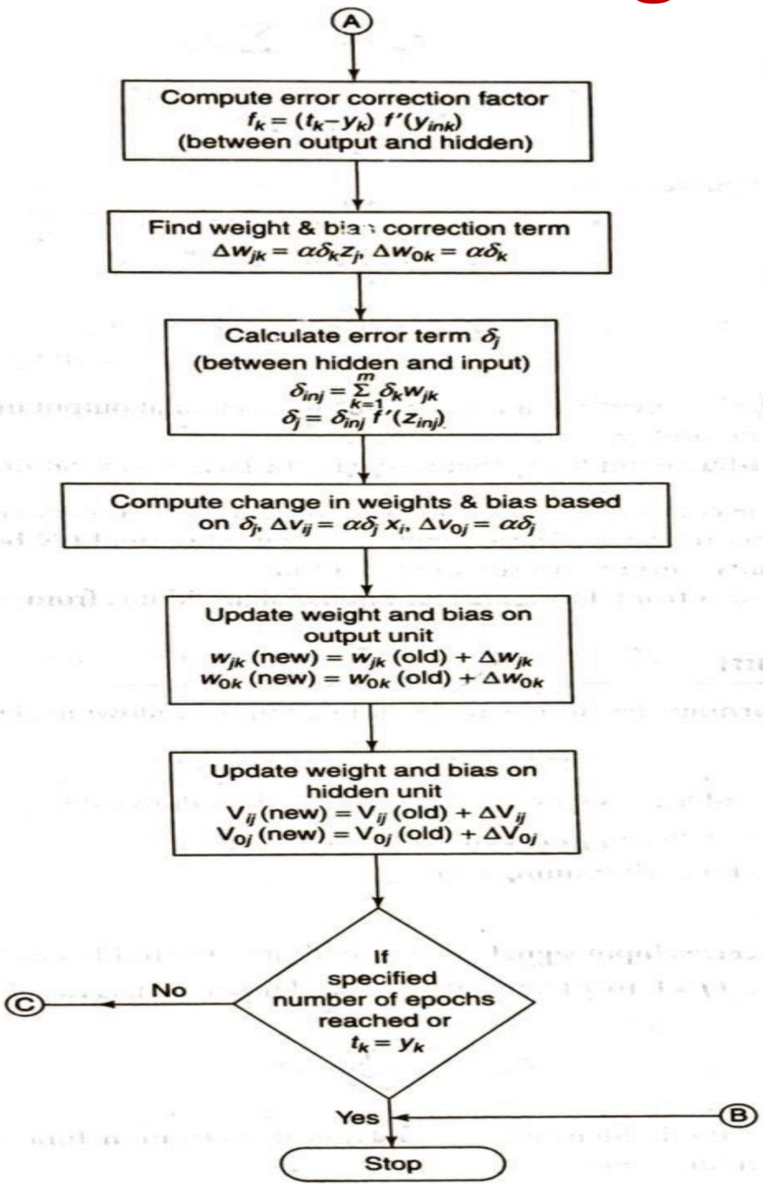
v_{0j} = bias on jth hidden unit

w_{0k} = bias on kth output unit

z_j = hidden unit j. The net input to z_j is



Training Process(Flow chart)



x = input training vector $(x_1, \ldots, x_i, \ldots, x_n)$

t = target output vector $(t_1, \ldots, t_k, \ldots, t_m)$

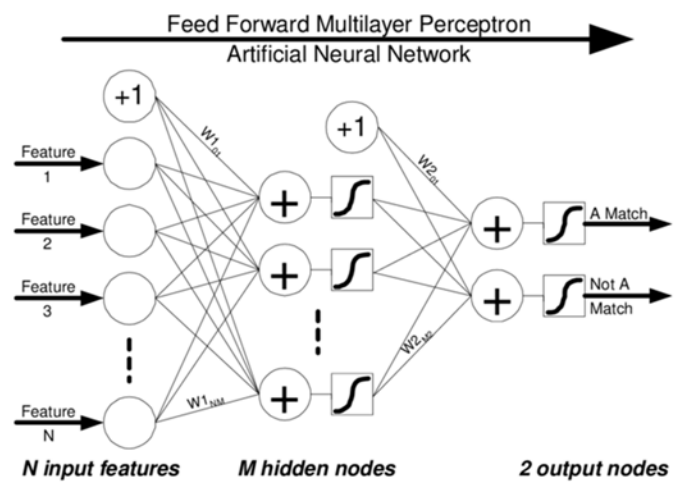
α = learning rate parameter

x_i = input unit i. (Since the input layer uses identity activation function, the input and output signals here are same.)

v_{0j} = bias on jth hidden unit

w_{0k} = bias on kth output unit

z_j = hidden unit j. The net input to z_j is



Training Process(Flow chart)

$$z_{inj} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

and the output is

$$z_j = f\,(z_{inj})$$

y_k = output unit k. The net input to y_k is

$$y_{ink} = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$$

and the output is

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

δ_k = error correction weight adjustment for w_{jk} that is due to an error at output unit y_k, which is back-propagated to the hidden units that feed into unit y_k

δ_j = error correction weight adjustment for v_{ij} that is due to the back-propagation of error to the hidden unit z_j.

Testing Process

Step 0: Initialize the weights. The weights are taken from the training algorithm.

Step 1: Perform Steps 2–4 for each input vector.

Step 2: Set the activation of input unit for x_i ($i = 1$ to n).

Step 3: Calculate the net input to hidden unit x and its output. For $j = 1$ to p,

$$z_{inj} = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}$$

$$z_j = f(z_{inj})$$

Step 4: Now compute the output of the output layer unit. For $k = 1$ to m,

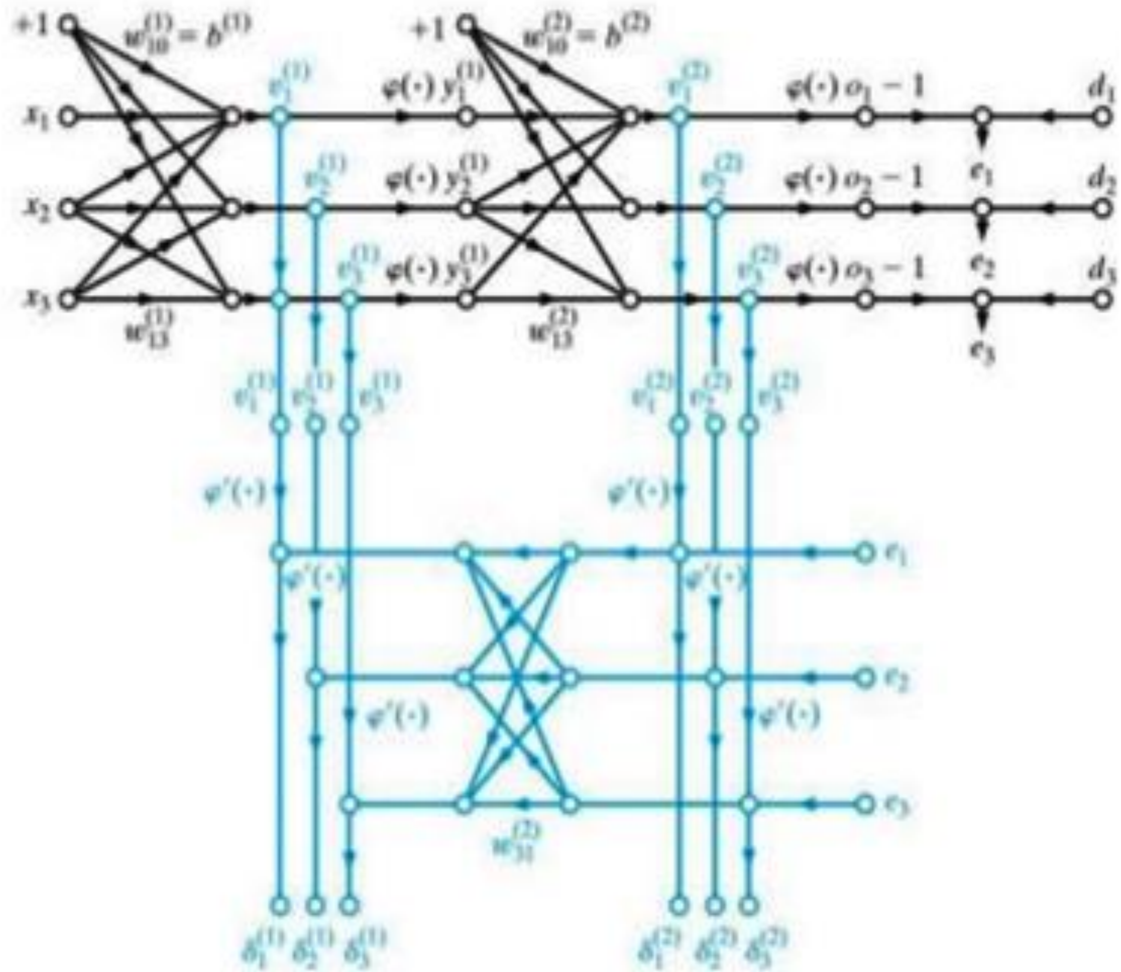
$$y_{ink} = w_{0k} + \sum_{j=1}^{p} z_j w_{jk}$$

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

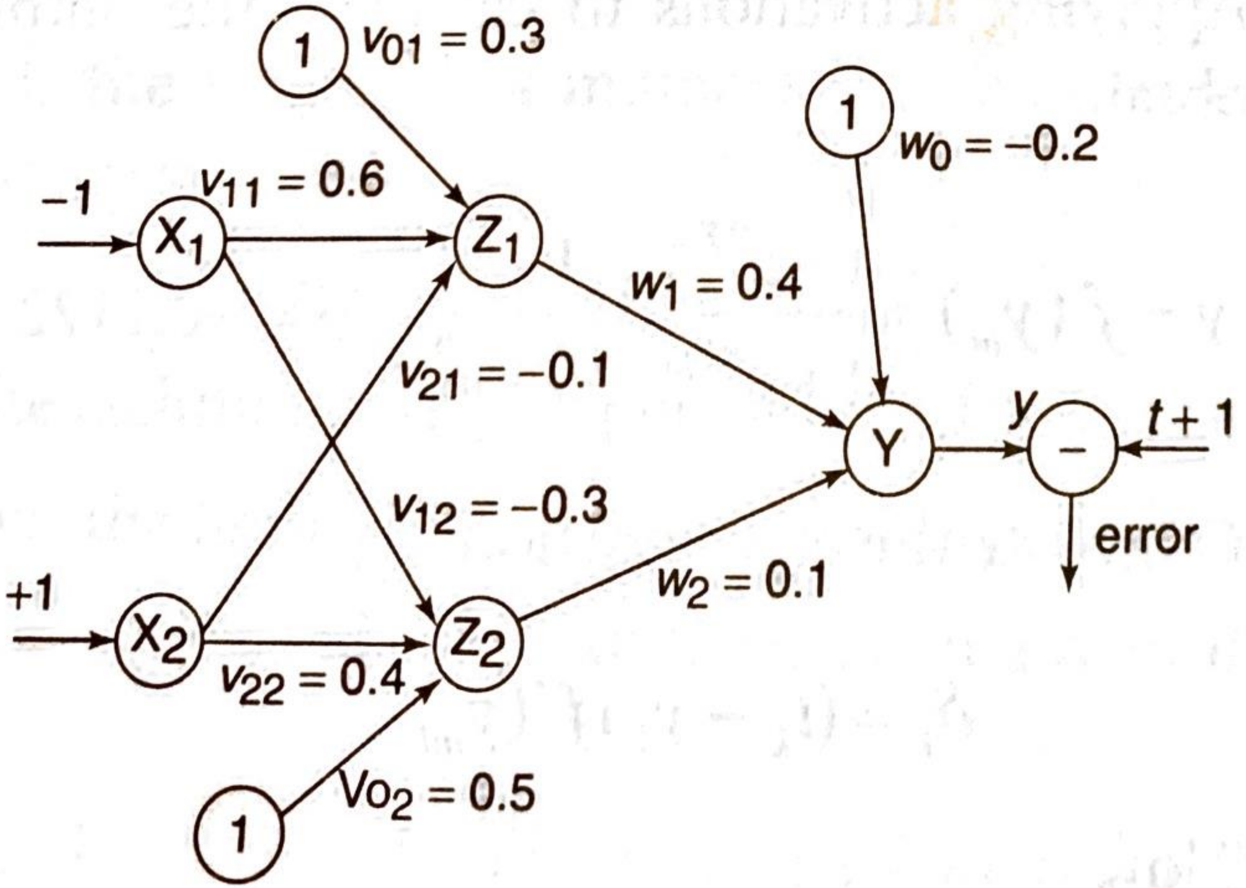
Use sigmoidal activation functions for calculating the output.

Summary of Back Propagation

1. ***Initialization***
2. ***Presentation of training example***
3. ***Forward computation***
4. ***Backward computation***
5. ***Iteration***



Example



presented with the input pattern [− 1, 1] and the target output is + 1. Use a learning rate of $\alpha = 0.25$ and bipolar sigmoidal activation function.

Solution: The initial weights are $[v_{11} v_{21} v_{01}] = [0.6 \; -0.1 \; 0.3]$, $[v_{12} \; v_{22} \; v_{02}] = [-0.3 \; 0.4 \; 0.5]$ and $[[w_1 \; w_2 \; w_0] = [0.4 \; 0.1 \; -0.2]$, and the learning rate is $\alpha = 0.25$.

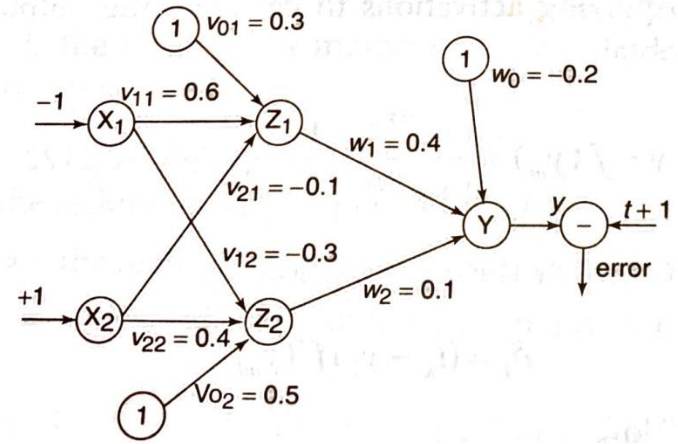
Activation function used is binary sigmoidal activation function and is given by

$$f(x) = \frac{2}{1+e^{-x}} - 1 = \frac{1-e^{-x}}{1+e^{-x}}$$

Phase 1

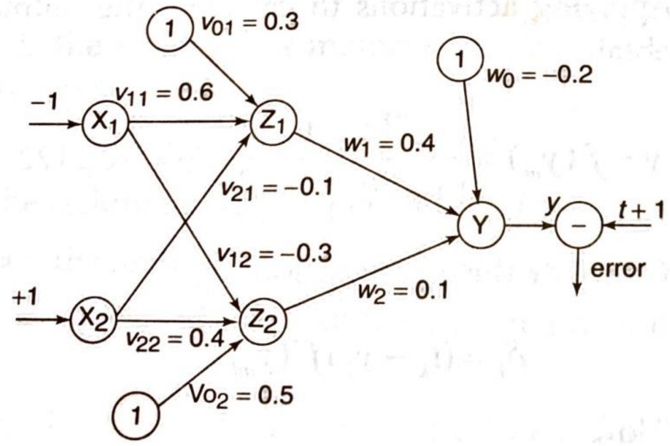
Activation function used is binary sigmoidal activation function and is given by

$$f(x) = \frac{2}{1+e^{-x}} - 1 = \frac{1-e^{-x}}{1+e^{-x}}$$



Given the input sample $[x_1, x_2] = [-1, 1]$ and target $t = 1$:

Phase 1



Calculate the net input: For z_1 layer

$$z_{in1} = v_{01} + x_1 v_{11} + x_2 v_{21}$$
$$= 0.3 + (-1) \times 0.6 + 1 \times -0.1 = -0.4$$

For z_2 layer

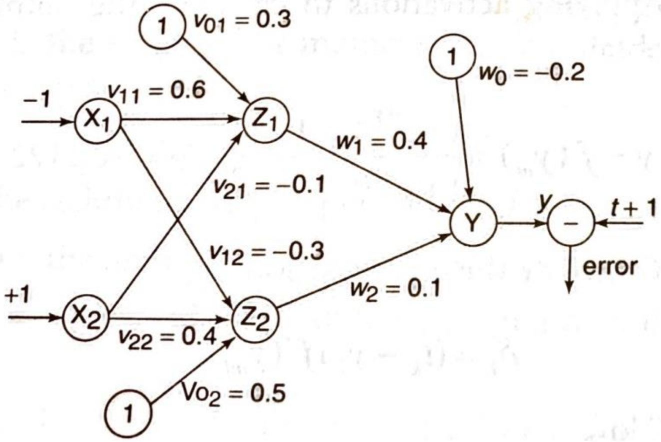
$$z_{in2} = v_{02} + x_1 v_{12} + x_2 v_{22}$$
$$= 0.5 + (-1) \times -0.3 + 1 \times 0.4 = 1.2$$

Phase 1

Applying activation to calculate the output, we obtain

$$z_1 = f(z_{in1}) = \frac{1-e^{-z_{in1}}}{1+e^{-z_{in1}}} = \frac{1-e^{0.4}}{1+e^{0.4}} = -0.1974$$

$$z_2 = f(z_{in2}) = \frac{1-e^{-z_{in1}}}{1+e^{-z_{in2}}} = \frac{1-e^{-1.2}}{1+e^{-1.2}} = 0.537$$



Sigmoidal Activation Function

- *Bipolar sigmoid function*: This function is defined as

$$f(x) = \frac{2}{1+e^{-\lambda x}} - 1 = \frac{1-e^{-\lambda x}}{1+e^{-\lambda x}}$$

where λ is the steepness parameter and the sigmoid function range is between −1 and +1. The derivative of this function can be

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

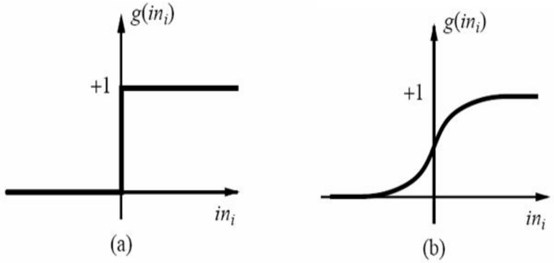
The bipolar sigmoidal function is closely related to hyperbolic tangent function, which is written as

$$h(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = \frac{1-e^{-2x}}{1+e^{-2x}}$$

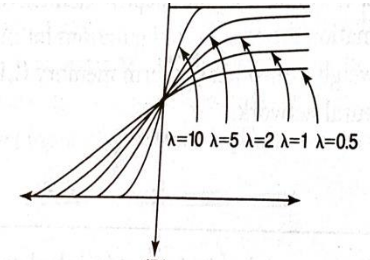
The derivative of the hyperbolic tangent function is

$$h'(x) = [1+h(x)][1-h(x)]$$

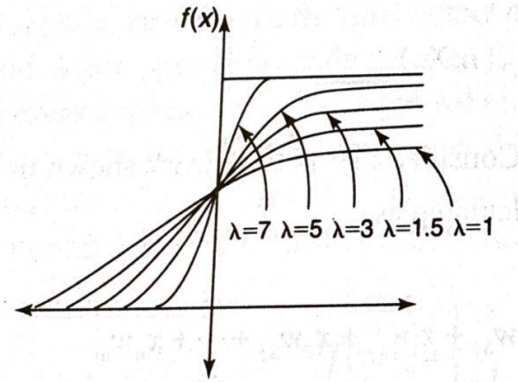
If the network uses a binary data, it is better to convert it to bipolar form and use the bipolar sigmoidal act function or hyperbolic tangent function.



step function sigmoid function



(D)



(E)

Phase 1

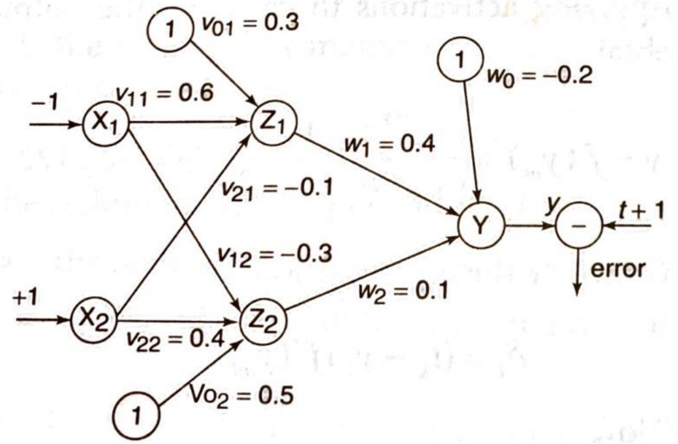
Calculate the net input entering the output layer.

For y layer

$$y_{in} = w_0 + z_1 w_1 + z_2 w_2$$
$$= -0.2 + (-0.1974) \times 0.4 + 0.537 \times 0.1$$
$$= -0.22526$$

Applying activations to calculate the output, we obtain

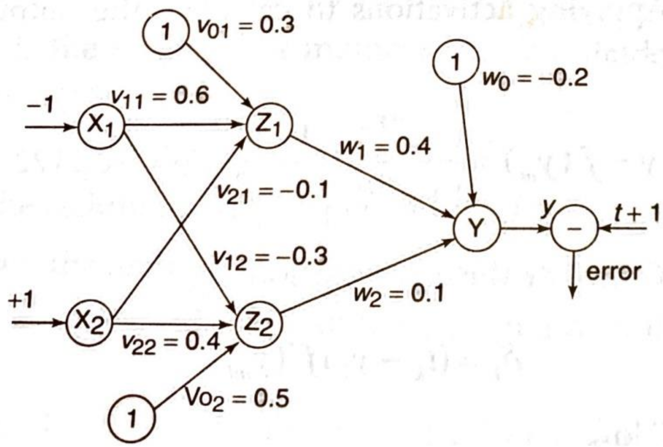
$$y = f(y_{in}) = \frac{1 - e^{-y_{in}}}{1 + e^{-y_{in}}} = \frac{1 - e^{0.22526}}{1 + e^{0.22526}} = -0.1122$$



Phase 1

Applying activations to calculate the output, we obtain

$$y = f(y_{in}) = \frac{1-e^{-y_{in}}}{1+e^{-y_{in}}} = \frac{1-e^{0.22526}}{1+e^{0.22526}} = -0.1122$$



Compute the error portion δ_k:

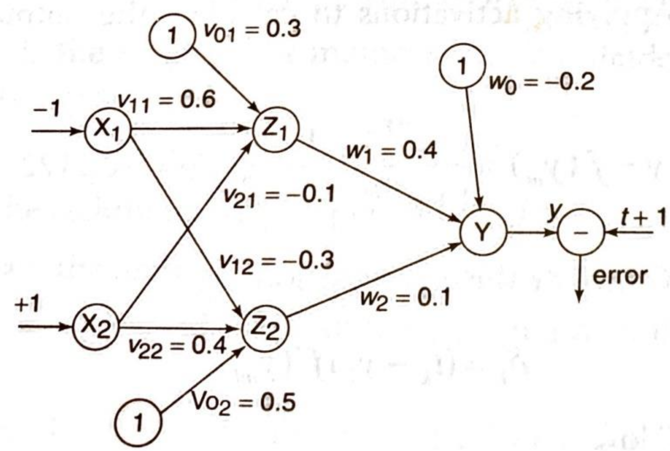
$$\delta_k = (t_k - y_k) f'(y_{ink})$$

Now

$$f'(y_{in}) = 0.5[1 + f(y_{in})][1 - f(y_{in})]$$
$$= 0.5\,[1 - 0.1122]\,[1 + 0.1122] = 0.4937$$

This implies

$$\delta_1 = (1 + 0.1122)(0.4937) = 0.5491$$



Sigmoidal Activation Function

- *Bipolar sigmoid function*: This function is defined as

$$f(x) = \frac{2}{1+e^{-\lambda x}} - 1 = \frac{1-e^{-\lambda x}}{1+e^{-\lambda x}}$$

where λ is the steepness parameter and the sigmoid function range is between -1 and $+1$. The derivative of this function can be

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

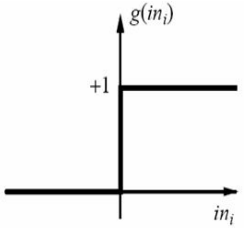
The bipolar sigmoidal function is closely related to hyperbolic tangent function, which is written as

$$h(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = \frac{1-e^{-2x}}{1+e^{-2x}}$$

The derivative of the hyperbolic tangent function is

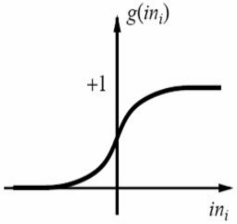
$$h'(x) = [1+h(x)][1-h(x)]$$

If the network uses a binary data, it is better to convert it to bipolar form and use the bipolar sigmoidal act function or hyperbolic tangent function.



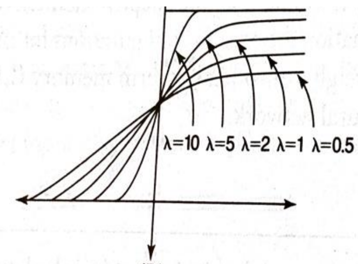
(a)

step function

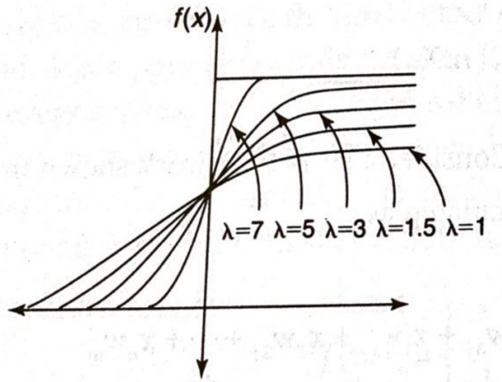


(b)

sigmoid function



(D)



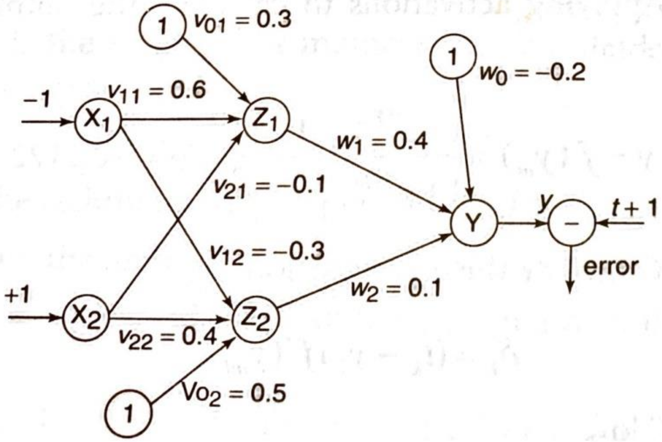
(E)

Phase II

Find the changes in weights between hidden and output layer:

$$\Delta w_1 = \alpha\delta_1 z_1 = 0.25 \times 0.5491 \times -0.1974$$

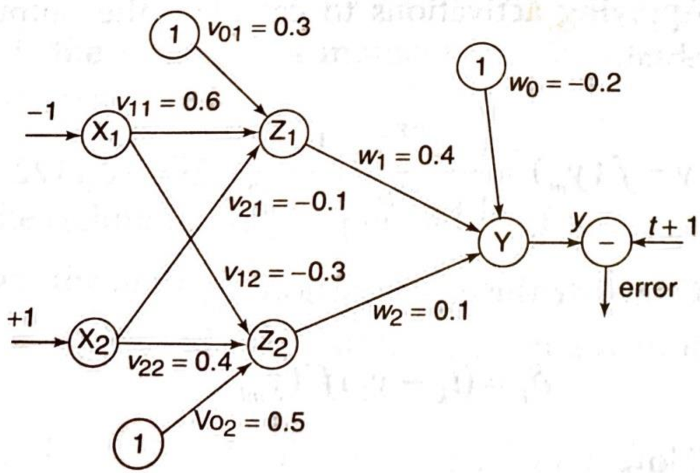
$$= -0.0271$$



Phase II

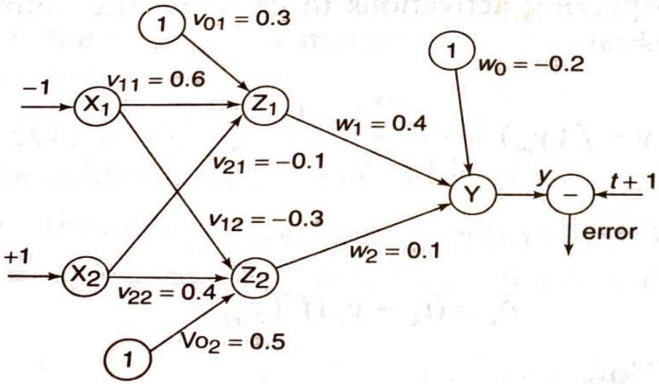
$$\Delta w_2 = \alpha\delta_1 z_2 = 0.25 \times 0.5491 \times 0.537 = 0.0737$$

$$\Delta w_0 = \alpha\delta_1 = 0.25 \times 0.5491 = 0.1373$$



Phase II

- Compute the error portion δ_j between input and hidden layer ($j = 1$ to 2):



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

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

$$\delta_{inj} = \delta_1 w_{j1} \quad [\because \text{ only one output neuron}]$$

$$\Rightarrow \delta_{in1} = \delta_1 w_{11} = 0.5491 \times 0.4 = 0.21964$$

$$\Rightarrow \delta_{in2} = \delta_2 w_{21} = 0.5491 \times 0.1 = 0.05491$$

$$\text{Error, } \delta_1 = \delta_{in1} f'(z_{in1})$$

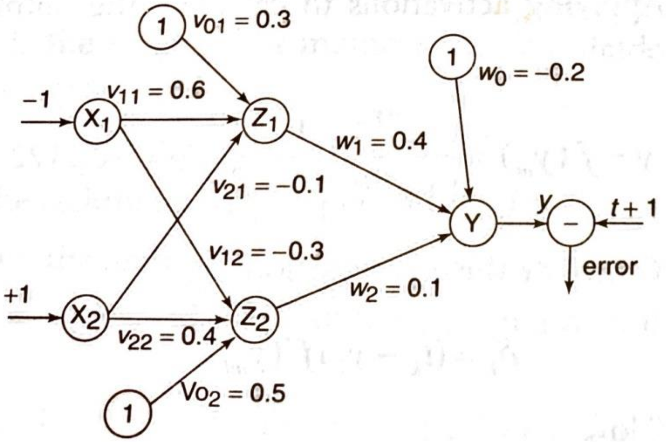
$$= 0.21964 \times 0.5 \times (1+0.1974)(1-0.1974)$$

$$= 0.1056$$

Phase II

$$\text{Error, } \delta_1 = \delta_{in1} f'(z_{in1})$$
$$= 0.21964 \times 0.5 \times (1+0.1974)(1-0.1974)$$
$$= 0.1056$$

$$\text{Error, } \delta_2 = \delta_{in2} f'(z_{in2})$$
$$= 0.05491 \times 0.5 \times (1-0.537)(1+0.537)$$
$$= 0.0195$$



Phase III

Now find the changes in weights between input and hidden layer:

$$\Delta v_{11} = \alpha\delta_1 x_1 = 0.25 \times 0.1056 \times -0.0264$$

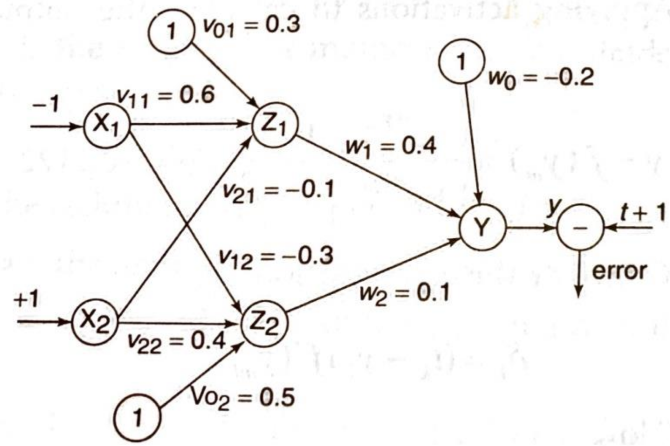
$$\Delta v_{21} = \alpha\delta_1 x_2 = 0.25 \times 0.1056 \times 1 = 0.0264$$

$$\Delta v_{01} = \alpha\delta_1 = 0.25 \times 0.1056 = 0.0264$$

$$\Delta v_{12} = \alpha\delta_2 x_1 = 0.25 \times 0.0195 \times -1 = 0.0049$$

$$\Delta v_{22} = \alpha\delta_2 x_2 = 0.25 \times 0.0195 \times 1 = 0.0049$$

$$\Delta v_{02} = \alpha\delta_2 = 0.25 \times 0.0195 = 0.0049$$



Phase III

$$v_{11}(\text{new}) = v_{11}(\text{old}) + \Delta v_{11} = 0.6 - 0.0264 = 0.5736$$

$$v_{12}(\text{new}) = v_{12}(\text{old}) + \Delta v_{12} = -0.3 - 0.0049 = -0.3049$$

$$v_{21}(\text{new}) = v_{21}(\text{old}) + \Delta v_{21} = -0.1 + 0.0264 = -0.0736$$

$$v_{22}(\text{new}) = v_{22}(\text{old}) + \Delta v_{22} = 0.4 + 0.0049 = 0.4049$$

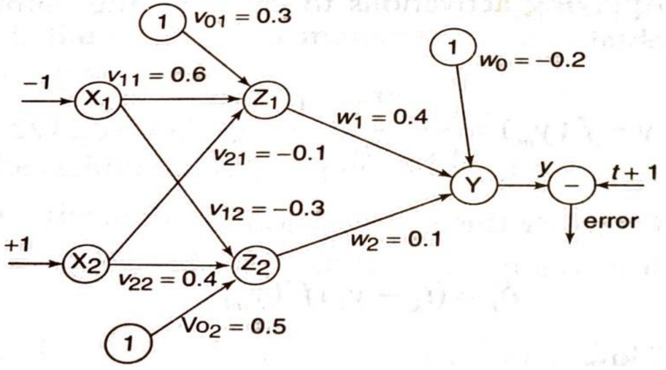
$$w_{1}(\text{new}) = w_{1}(\text{old}) + \Delta w_{1} = 0.4 - 0.0271 = 0.3729$$

$$w_{2}(\text{new}) = w_{2}(\text{old}) + \Delta w_{2} = 0.1 + 0.0737 = 0.1737$$

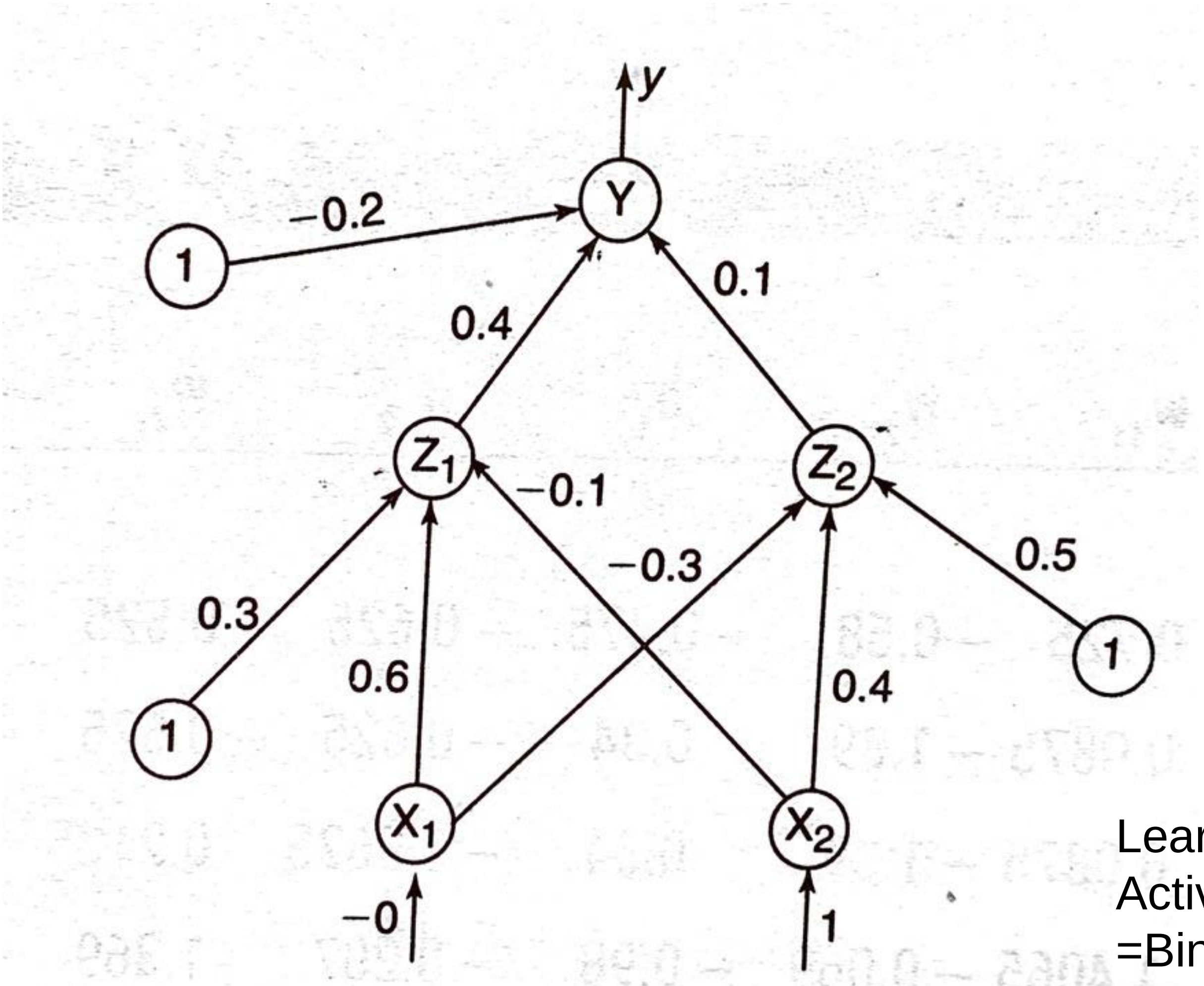
$$v_{01}(\text{new}) = v_{01}(\text{old}) + \Delta v_{01} = 0.3 + 0.0264 = 0.3264$$

$$v_{02}(\text{new}) = v_{02}(\text{old}) + \Delta v_{02} = 0.5 + 0.0049 = 0.5049$$

$$w_{0}(\text{new}) = w_{0}(\text{old}) + \Delta w_{0} = -0.2 + 0.1373 = -0.0627$$



Example



I/p=0,1
T=1

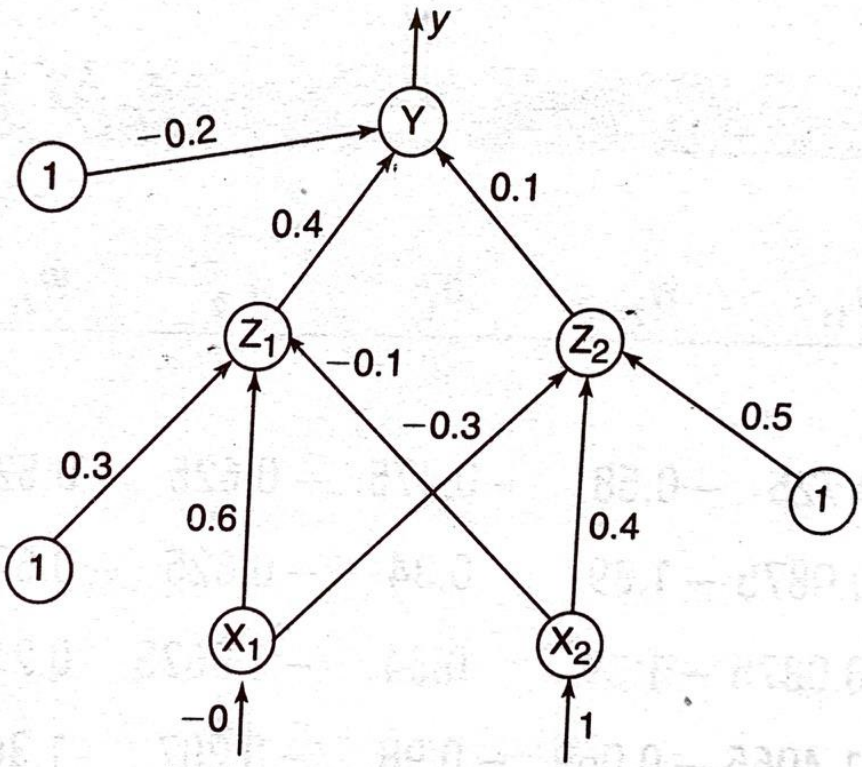
Learning rate =0.25
Activation function
=Binary Sigmoidal

Calculate the net input for Z1&Z2

$$z_{in1} = v_{01} + x_1 v_{11} + x_2 v_{21}$$
$$= 0.3 + 0 \times 0.6 + 1 \times -0.1 = 0.2$$

For z_2 layer

$$z_{in2} = v_{02} + x_1 v_{12} + x_2 v_{22}$$
$$= 0.5 + 0 \times -0.3 + 1 \times 0.4 = 0.9$$

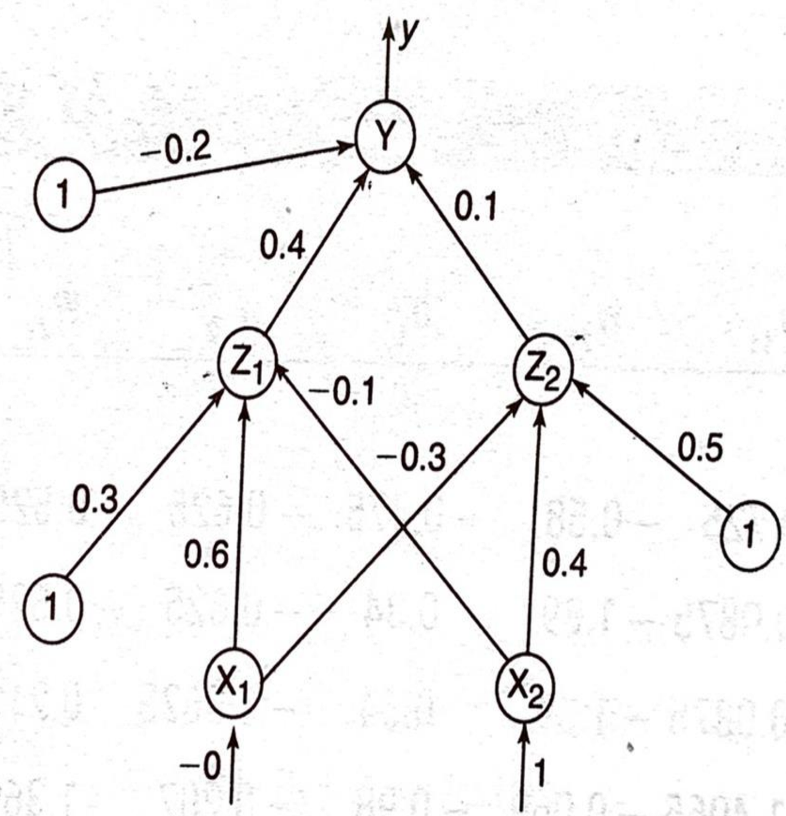


Apply Activation Function

Applying activation to calculate the output, we obtain

$$z_1 = f(z_{in1}) = \frac{1}{1+e^{-z_{in1}}} = \frac{1}{1+e^{-0.2}} = 0.5498$$

$$z_2 = f(z_{in2}) = \frac{1}{1+e^{-z_{in2}}} = \frac{1}{1+e^{-0.9}} = 0.7109$$



Calculate the net input to o/p layer

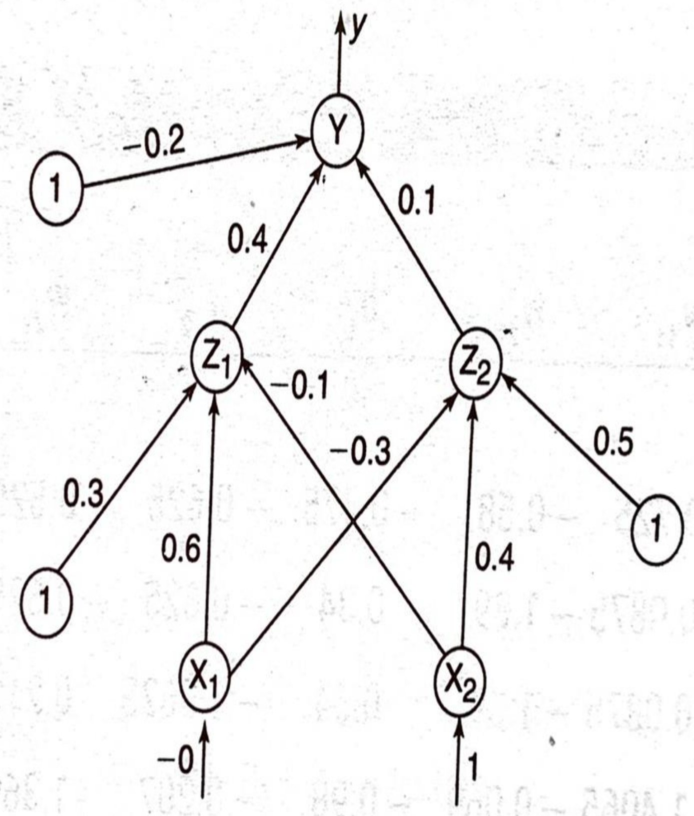
- Calculate the net input entering the output layer.

For y layer

$$y_{in} = w_0 + z_1 w_1 + z_2 w_2$$
$$= -0.2 + 0.5498 \times 0.4 + 0.7109 \times 0.1$$
$$= 0.09101$$

Applying activations to calculate the output, we obtain

$$y = f(y_{in}) = \frac{1}{1+e^{-y_{in}}} = \frac{1}{1+e^{-0.09101}} = 0.5227$$



Compute the Error function

- Compute the error portion δ_k :

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

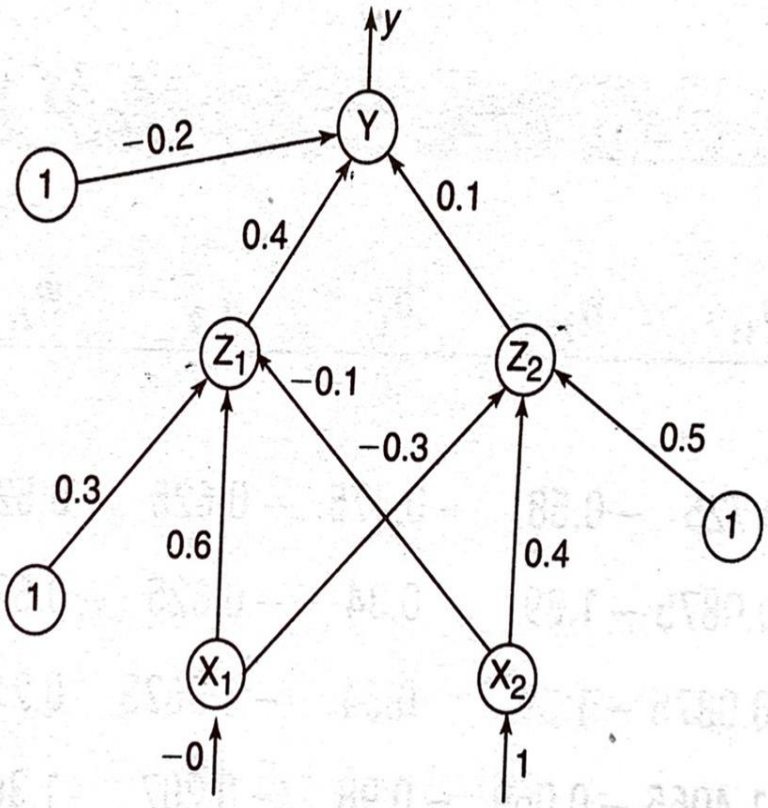
Now

$$f'(y_{in}) = f(y_{in})[1 - f(y_{in})] = 0.5227[1 - 0.5$$

$$f'(y_{in}) = 0.2495$$

This implies

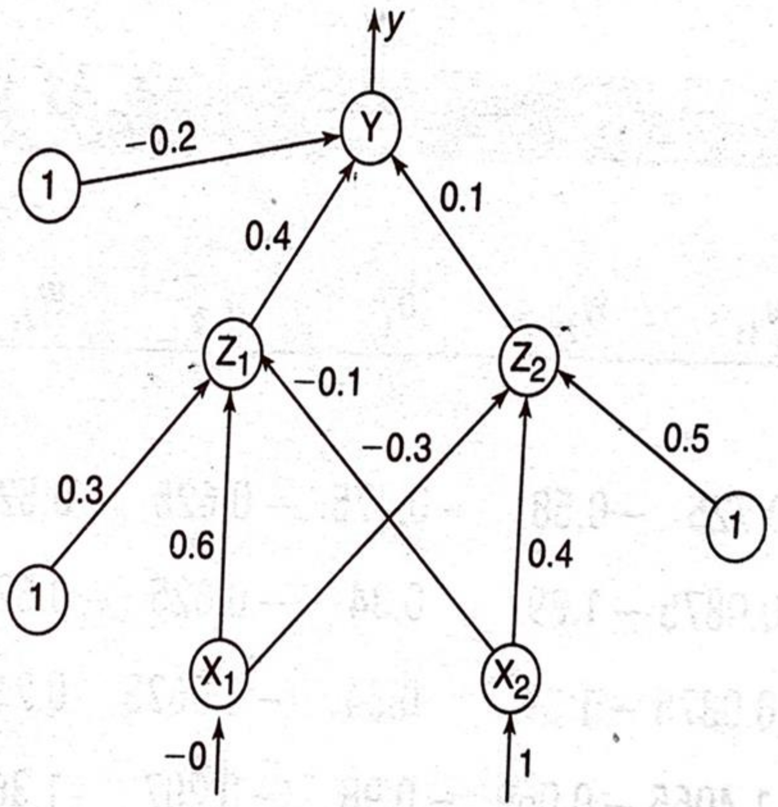
$$\delta_1 = (1 - 0.5227)(0.2495) = 0.1191$$



Find the weight change H&O

Find the changes in weights between hidden output layer:

$$\Delta w_1 = \alpha\delta_1 z_1 = 0.25\times 0.1191\times 0.5498$$
$$= 0.0164$$



$$\Delta w_2 = \alpha\delta_1 z_2 = 0.25\times 0.1191\times 0.7109$$
$$= 0.02117$$
$$\Delta w_0 = \alpha\delta_1 = 0.25\times 0.1191 = 0.02978$$

Compute Error (I&H)

- Compute the error portion δ_j between input and hidden layer ($j = 1$ to 2):

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

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

$$\delta_{inj} = \delta_1 w_{j1} \quad [\because \text{ only one output neuron}]$$

$$\Rightarrow \delta_{in1} = \delta_1 w_{11} = 0.1191 \times 0.4 = 0.04764$$

$$\Rightarrow \delta_{in2} = \delta_1 w_{21} = 0.1191 \times 0.1 = 0.01191$$

Error, $\delta_1 = \delta_{in1} f'(z_{in1})$

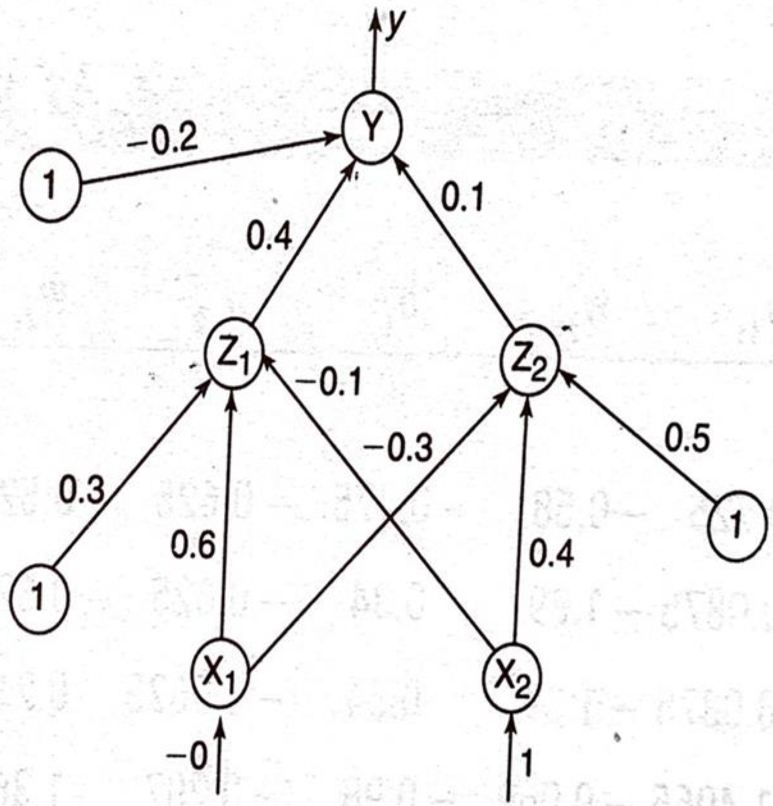
$$f'(z_{in1}) = f(z_{in1})[1 - f(z_{in1})]$$
$$= 0.5498[1 - 0.5498] = 0.2475$$

$$\delta_1 = \delta_{in1} f'(z_{in1})$$
$$= 0.04764 \times 0.2475 = 0.0118$$

Error, $\delta_2 = \delta_{in2} f'(z_{in2})$

$$f'(z_{in2}) = f(z_{in2})[1 - f(z_{in2})]$$
$$= 0.7109[1 - 0.7109] = 0.2055$$

$$\delta_2 = \delta_{in2} f'(z_{in2})$$
$$= 0.01191 \times 0.2055 = 0.00245$$



Change in weight (I&H)

Now find the changes in weights between input and hidden layer:

$$\Delta v_{11} = \alpha\delta_1 x_1 = 0.25 \times 0.0118 \times 0 = 0$$

$$\Delta v_{21} = \alpha\delta_1 x_2 = 0.25 \times 0.0118 \times 1 = 0.00295$$

$$\Delta v_{01} = \alpha\delta_1 = 0.25 \times 0.0118 = 0.00295$$

$$\Delta v_{12} = \alpha\delta_2 x_1 = 0.25 \times 0.00245 \times 0 = 0$$

$$\Delta v_{22} = \alpha\delta_2 x_2 = 0.25 \times 0.00245 \times 1 = 0.0006125$$

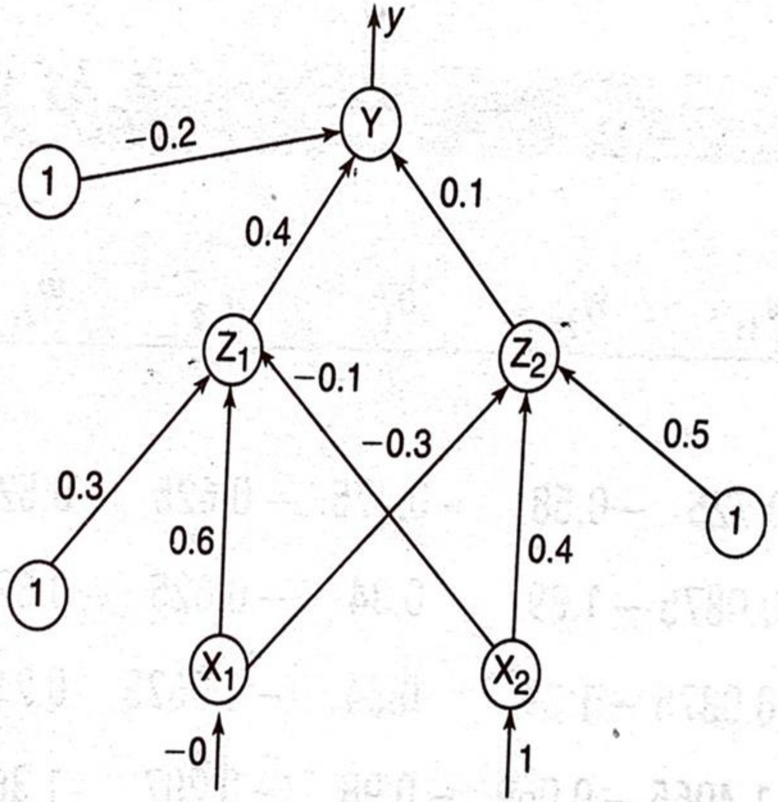
$$\Delta v_{02} = \alpha\delta_2 = 0.25 \times 0.00245 = 0.0006125$$

- Compute the final weights of the network:

$$v_{11}(\text{new}) = v_{11}(\text{old}) + \Delta v_{11} = 0.6 + 0 = 0.6$$

$$v_{12}(\text{new}) = v_{12}(\text{old}) + \Delta v_{12} = -0.3 + 0 = -0.3$$

$$v_{21}(\text{new}) = v_{21}(\text{old}) + \Delta v_{21}$$



Change in weight (I&H)

$$= -0.1 + 0.00295 = -0.09705$$

$$v_{22}(\text{new}) = v_{22}(\text{old}) + \Delta v_{22}$$
$$= 0.4 + 0.0006125 = 0.4006125$$

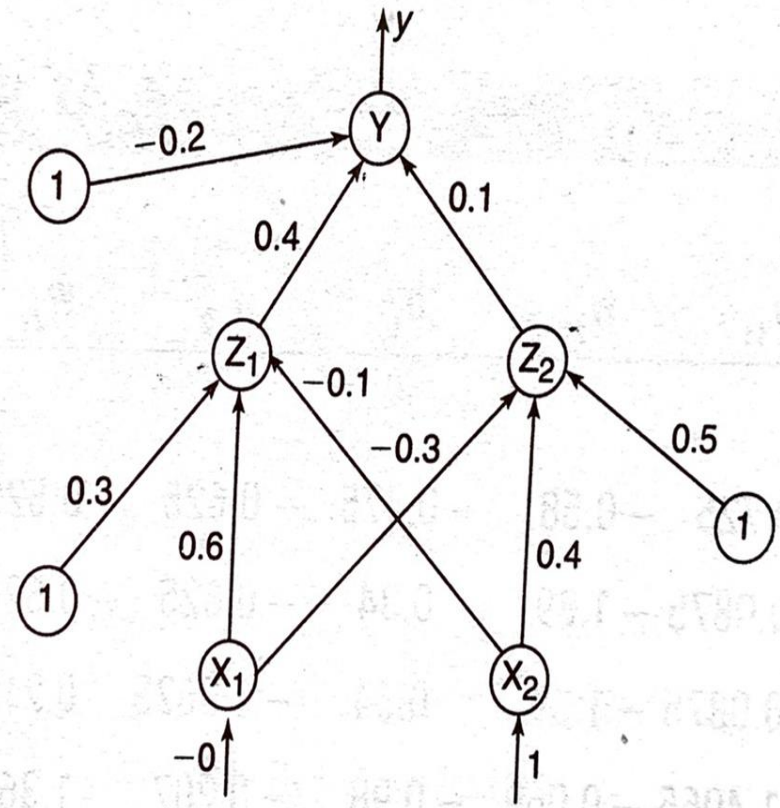
$$w_1(\text{new}) = w_1(\text{old}) + \Delta w_1 = 0.4 + 0.0164$$
$$= 0.4164$$

$$w_2(\text{new}) = w_2(\text{old}) + \Delta w_2 = 0.1 + 0.02117$$
$$= 0.12117$$

$$v_{01}(\text{new}) = v_{01}(\text{old}) + \Delta v_{01} = 0.3 + 0.00295$$
$$= 0.30295$$

$$v_{02}(\text{new}) = v_{02}(\text{old}) + \Delta v_{02}$$
$$= 0.5 + 0.0006125 = 0.5006125$$

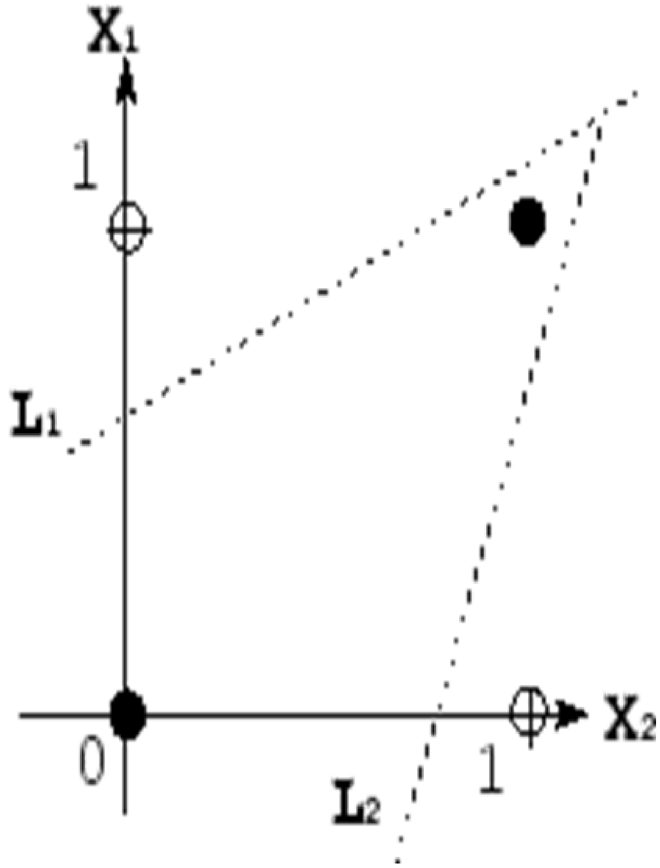
$$w_0(\text{new}) = w_0(\text{old}) + \Delta w_0 = -0.2 + 0.02978$$
$$= -0.17022$$



XOR

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	0

$$Y = X_1 \oplus X_2$$



MCQ

What is back propagation?

a) It is another name given to the curvy function in the perceptron

b) It is the transmission of error back through the network to adjust the inputs

c) It is the transmission of error back through the network to allow weights to be adjusted so that the network can learn

d) None of the mentioned

Answer: c

Having multiple perceptrons can actually solve the XOR problem satisfactorily: this is because each perceptron can partition off a linear part of the space itself, and they can then combine their results.

a) True – this works always, and these multiple perceptrons learn to classify even complex problems

b) False – perceptrons are mathematically incapable of solving linearly inseparable functions, no matter what you do

c) True – perceptrons can do this but are unable to learn to do it – they have to be explicitly hand-coded

d) False – just having a single perceptron is enough

Answer: c

- The network that involves backward links from output to the input and hidden layers is called __________
  a) Self organizing maps
  b) Perceptrons
  c) Recurrent neural network
  d) Multi layered perceptron

- Answer: c
  Explanation: RNN (Recurrent neural network) topology involves backward links from output to the input and hidden layers.

- What kind of signal is used in speech recognition?
  a) Electromagnetic signal
  b) Electric signal
  c) Acoustic signal
  d) Radar

- Answer: c
  Explanation: Acoustic signal is used to identify a sequence of words uttered by a speaker.

- What is used to initiate the perception in the environment?
  a) Sensor
  b) Read
  c) Actuators
  d) None of the mentioned

- Answer: a
  Explanation: A sensor is anything that can record some aspect of the environment.

- What is a perception check?
  a) a cognitive bias that makes us listen only to information we already agree with
  b) a method teachers use to reward good listeners in the classroom
  c) any factor that gets in the way of good listening and decreases our ability to interpret correctly
  d) a response that allows you to state your interpretation and ask your partner whether or not that interpretation is correct

- Answer: d

Which of the following is not the promise of artificial neural network?

a) It can explain result

b) It can survive the failure of some nodes

c) It has inherent parallelism

d) It can handle noise

Answer: a