4.1.3 Back propagation

The determination of the error is a recursive process which start with the o/p units and the error is back propagated to the I/p units. Therefore the rule is called error Back propagation (EBP) or simply Back Propagation (BP). The weight is changed exactly in the same form of the standard DR

$$\Delta w_{ij} = \xi \; \delta_j \, x_i$$

$$\Rightarrow \quad w_{ij}(t+1) = w_{ij}(t) \; + \; \xi \; \delta_j \, x_i$$

There are two other equations that specify the error signal. If a unite is an o/p unit, the error signal is given by:-

$$\delta = (d_j - y_j) \; f_j(\text{net } j)$$

$$\text{Where} \quad \text{net } j = \sum w_{ij} \, x_i + \theta$$

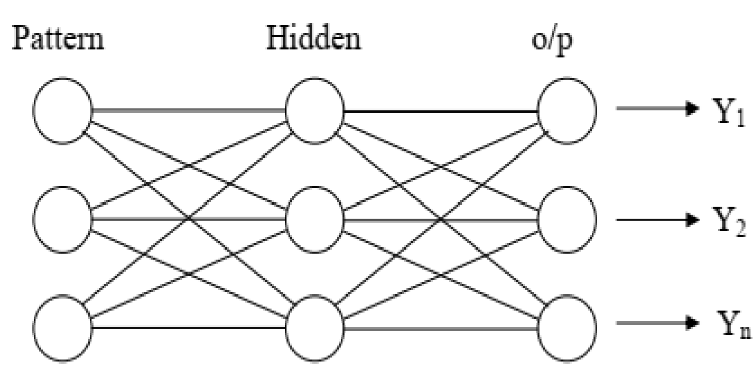
The GDR minimize the squares of the differences between the actual and the desired o/p values summed over the o/p unit and all pairs of I/p and o/p vectors. The rule minimize the overall error $E = \sum E_p$ by implementing a gradient descent in E: - where, $E_p = 1/2 \sum_j (d_j - y_j)^2$.

The BP consists of two phases:-

1- Forward Propagation:-

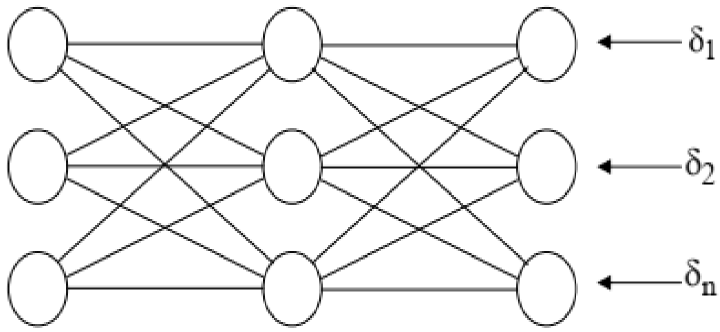
During the forward phase, the I/p is presented and propagated towards the o/p.

المرحلة الأولى



2- Backward Propagation:-

During the backward phase, the ***errors*** are formed at the o/p and propagated towards the I/p



3- Compute the error in the hidden layer.

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

$f' = y(1-y)$

Equation is can rewrite as:-

$\delta_j = y(1-y)(d_j - y_j)$

The error signal for hidden units for which there is no specified target (desired o/p) is determined recursively in terms of the error signals of the units to which it directly connects and the weights of those connections:-

That is

$$\delta_j = f'(net_j)\sum_k \delta_k w_{ik}$$

Or

$$\delta_j = y_j(1-y_j)\sum_k \delta_k w_{ik}$$

B.P learning is implemented when hidden units are embedded between input and output units.

4.1.3.1 Back propagation training algorithm

Training a network by back propagation involves three stages:-

1-the feed forward of the input training pattern

2-the back propagation of the associated error

3-the adjustment of the weights

let n = number of input units in input layer,

let p = number of hidden units in hidden layer

let m = number of output units in output layer

let V_{ij} be the weights between i/p layer and the hidden layer,

let W_{ij} be the weights between hidden layer and the output layer,

we refer to the i/p units as X_i , i=1, 2, ….,n. and we refer to the hidden units as Z_j , j=1,….,p. and we refer to the o/p units as y_k, k=1,….., m.

δ_{1j} is the error in hidden layer,

δ_{2k} is the error in output layer,

ζ is the learning rate

$\propto$ is the momentum coefficient (learning coefficient, $0.0 < \propto < 1.0$,

y_k is the o/p of the net (o/p layer),

Z_j is the o/p of the hidden layer,

X_i is the o/p of the i/p layer.

η is the learning coefficient.

The algorithm is as following :-

Step 0 : initialize weights (set to small random value).

Step 1 : while stopping condition is false do steps 2-9

Step 2: for each training pair, do steps 3-8

Feed forward :-

Step 3:- Each i/p unit (X_i) receives i/p signal X_i & broad casts this signal to all units in the layer above (the hidden layer)

Step 4:- Each hidden unit (Z_j) sums its weighted i/p signals,

$$Z-inj = Vaj + \sum_{i=1}^{n} x_i v_{ij} \quad (Vaj \ \text{is abias})$$

and applies its activation function to compute its output signal (the activation function is the binary sigmoid function),

$$Z_j f(Z-inj) = 1 \ / \ (1+\exp-(Z-inj))$$

and sends this signal to all units in the layer above (the o/p layer).

Step 5:- Each output unit (Yk)sums its weighted i/p signals,

$$y-ink = wok + \sum_{j=1}^{p} Zjwjk \quad (\text{where } wok \text{ is abias})$$

and applies its activation function to compute its output signal.

$$y_k = f(y-ink) = 1/(1+\exp-(y-ink)$$

back propagation of error:-

<u>step 6</u> : Each output unit (y_k , k= 1 tom) receive a target pattern corresponding to the input training pattern, computes its error information term and calculates its weights correction term used to update W_{jk} later,

$$\delta_{2k} = y_k(1-y_k)*(T_k - y_k),$$

where T_k is the target pattern & k=1 to m .

<u>step 7</u> : Each hidden unit (Z_j, j= 1 top) computes its error information term and calculates its weight correction term used to update Vij later,

$$\delta_{1j} = Zj*(1-Zj)*\sum_{k=1}^{m}\delta 2kWjk$$

Update weights and bias :-

<u>step 8</u>: Each output unit (y_k, k =1 tom) updates its bias and weights:

$$Wjk(new) = \eta * \delta 2k * Zj + \propto *[Wjk(dd)],$$

j= 1 to p

Each hidden unit (Z_j, j= 1 to p) update its bias and weights:

$$Vij(new) = \eta * \delta 1j * Xi + \propto [vij(dd)],$$

I = 1 to n

<u>Step 9</u> : Test stopping condition.

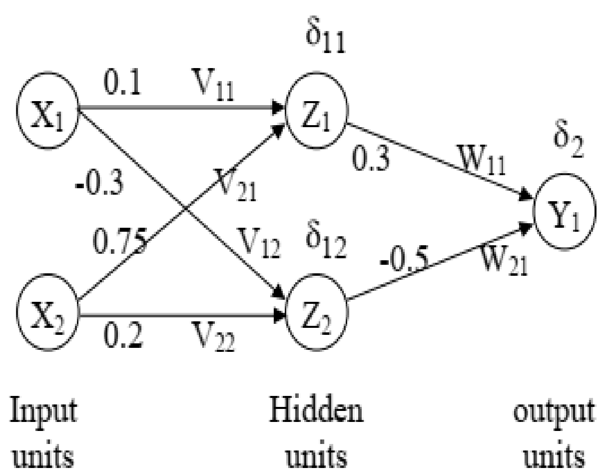
EX6

Suppose you have BP- ANN with 2-input , 2-hiddden , 1-output nodes with sigmoid function and the following matrices weight, trace with 1-iteration.

$$V = \begin{bmatrix} 0.1 & -0.3 \\ 0.75 & 0.2 \end{bmatrix} \qquad w = \begin{bmatrix} 0.3 & -0.5 \end{bmatrix}$$

Where $\propto = 0.9, \quad \eta = 0.45, \quad x = (1,0)$, and $T_k = 1$

Solution:-



1-Forword phase :-

$$Z-in1 = X_1V_{11} + X_2V_{21} = 1*0.1 + 0*0.75 = 0.1$$
$$Z-in2 = X_1V_{12} + X_2V_{22} = 1*-0.3 + 0*0.2 = -0.3$$
$$Z_1 = f(Z-in1) = 1/(1+\exp-(Z-in1)) = 0.5$$
$$Z_2 = f(Z-in2) = 1/(1+\exp-(Z-in2)) = 0.426$$
$$y-in1 = Z_1W_{11} + Z_2W_{21}$$
$$= 0.5*0.3 + 0.426*(-0.5) = -0.063$$
$$y_1 = f(y-in1) = 1/(1+\exp-(y-in1) = 0.484$$

2-Backward phase :-

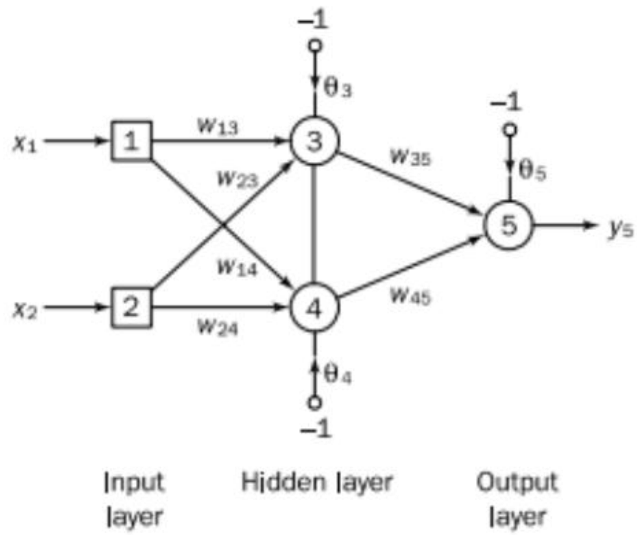
$$\delta_2 k = yk(1-yk) * (Tk - yk)$$
$$\delta_{21} = 0.484(1-0.484) * (1-0.484)0.129$$
$$\delta_{1j} = Z_j * (1-Z_j) * \sum_{k=1}^{m} \delta_{2k} W_{jk}$$
$$\delta_{11} = Z_1(1-Z_1) * (\delta_{21} W_{11})$$
$$= 0.5\,(1 - 0.5) * (0.129 * 0.3) = 0.0097$$
$$\delta_{12} = Z_2(1-Z_2) * (\delta_{21} W_{21})$$
$$= 0.426(1-0.426) * (0.129 * (-0.5)) = -0.015$$

3-Update weights:-

$$W_{jk}(new) = \eta * \delta_{2k} * Z_j + \propto * [W_{jk}(old)]$$
$$W_{11} = \eta * \delta_{21} * Z_1 + \propto * [W_{11}(old)]$$
$$= 0.45 * 0.129 * 0.5 + 0.9 * 0.3 = 0.299$$
$$W_{21} = \eta * \delta_{21} * Z_2 + \propto * [W_{21}(old)]$$
$$= 0.45 * 0.129 * 0.426 + 0.9 * -0.5 = -0.4253$$
$$V_{ij}(new) = \eta * \delta_{1j} * X_i + \propto * [V_{ij}(old)]$$
$$V_{11} = \eta * \delta_{11} * X_1 + \propto * [V_{11}(old)]$$
$$= 0.45 * 0.0097 * 1 + 0.9 * 0.1 = 0.0944$$
$$V_{12} = \eta * \delta_{12} * X_1 + \propto * [V_{12}(old)]$$
$$= 0.45 * 0.0158 * 1 + 0.9 * -0.3 = -0.2771$$
$$V_{21} = \eta * \delta_{11} * X_2 + \propto * [V_{21}(old)]$$
$$= 0.45 * 0.0097 * 0 + 0.9 * 0.75 = 0.675$$
$$V_{22} = \eta * \delta_{12} * X_2 + \propto * [V_{22}(old)]$$
$$= 0.45 * -0.0158 * 0 + 0.9 * 0.2 = 0.18$$

$$\therefore V = \begin{bmatrix} 0.0944 & -0.2771 \\ 0.675 & 0.18 \end{bmatrix} \qquad W = [0.299 \quad -0.4253]$$

As an example, we may consider the three-layer back-propagation network shown in Figure below. Suppose that the network is required to perform logical operation Exclusive-OR.



The effect of the threshold applied to a neuron in the hidden or output layer is represented by its weightθ, connected to a fixed input equal to 1. The initial weights and threshold levels are set randomly as follows:

$w_{13} = 0.5$, $w_{14} = 0.9$, $w_{23} = 0.4$, $w_{24} = 1.0$, $w_{35} = -1.2$, $w_{45} = 1.1$,
$\theta_3 = 0.8$, $\theta_4 = -0.1$ and $\theta_5 = 0.3$.

Consider a training set where inputs $x1$ and $x2$ are equal to 1 and desired output $y_{d,5}$is 0. The actual outputs of neurons 3 and 4 in the hidden layer are calculated as

$$y_3 = sigmoid\,(x_1 w_{13} + x_2 w_{23} - \theta_3) = 1/[1 + e^{-(1\times 0.5 + 1\times 0.4 - 1\times 0.8)}] = 0.5250$$

$$y_4 = sigmoid\,(x_1 w_{14} + x_2 w_{24} - \theta_4) = 1/[1 + e^{-(1\times 0.9 + 1\times 1.0 + 1\times 0.1)}] = 0.8808$$

Now the actual output of neuron 5 in the output layer is determined as

$$y_5 = sigmoid\,(y_3 w_{35} + y_4 w_{45} - \theta_5) = 1/[1 + e^{-(-0.5250\times 1.2 + 0.8808\times 1.1 - 1\times 0.3)}] = 0.5097$$

Thus, the following error is obtained:

$$e = y_{d,5} - y_5 = 0 - 0.5097 = -0.5097$$

The next step is weight training. To update the weights and threshold levels in our network, we propagate the error, e, from the output layer backward to the input layer.

First, we calculate the error gradient for neuron 5 in the output layer

$$\delta_5 = y_5(1 - y_5)e = 0.5097 \times (1 - 0.5097) \times (-0.5097) = -0.1274$$

Then we determine the weight corrections assuming that the learning rate parameter, α, is equal to 0.1:

$$\Delta w_{35} = \alpha \times y_3 \times \delta_5 = 0.1 \times 0.5250 \times (-0.1274) = -0.0067$$
$$\Delta w_{45} = \alpha \times y_4 \times \delta_5 = 0.1 \times 0.8808 \times (-0.1274) = -0.0112$$
$$\Delta \theta_5 = \alpha \times (-1) \times \delta_5 = 0.1 \times (-1) \times (-0.1274) = 0.0127$$

Next we calculate the error gradients for neurons 3 and 4 in the hidden layer:

$$\delta_3 = y_3(1 - y_3) \times \delta_5 \times w_{35} = 0.5250 \times (1 - 0.5250) \times (-0.1274) \times (-1.2) = 0.0381$$
$$\delta_4 = y_4(1 - y_4) \times \delta_5 \times w_{45} = 0.8808 \times (1 - 0.8808) \times (-0.1274) \times 1.1 = -0.0147$$

We then determine the weight corrections

$$\Delta w_{13} = \alpha \times x_1 \times \delta_3 = 0.1 \times 1 \times 0.0381 = 0.0038$$
$$\Delta w_{23} = \alpha \times x_2 \times \delta_3 = 0.1 \times 1 \times 0.0381 = 0.0038$$
$$\Delta \theta_3 = \alpha \times (-1) \times \delta_3 = 0.1 \times (-1) \times 0.0381 = -0.0038$$
$$\Delta w_{14} = \alpha \times x_1 \times \delta_4 = 0.1 \times 1 \times (-0.0147) = -0.0015$$
$$\Delta w_{24} = \alpha \times x_2 \times \delta_4 = 0.1 \times 1 \times (-0.0147) = -0.0015$$
$$\Delta \theta_4 = \alpha \times (-1) \times \delta_4 = 0.1 \times (-1) \times (-0.0147) = 0.0015$$

At last, we update all weights and threshold levels in our network:

$$w_{13} = w_{13} + \Delta w_{13} = 0.5 + 0.0038 = 0.5038$$
$$w_{14} = w_{14} + \Delta w_{14} = 0.9 - 0.0015 = 0.8985$$
$$w_{23} = w_{23} + \Delta w_{23} = 0.4 + 0.0038 = 0.4038$$
$$w_{24} = w_{24} + \Delta w_{24} = 1.0 - 0.0015 = 0.9985$$
$$w_{35} = w_{35} + \Delta w_{35} = -1.2 - 0.0067 = -1.2067$$
$$w_{45} = w_{45} + \Delta w_{45} = 1.1 - 0.0112 = 1.0888$$
$$\theta_3 = \theta_3 + \Delta \theta_3 = 0.8 - 0.0038 = 0.7962$$
$$\theta_4 = \theta_4 + \Delta \theta_4 = -0.1 + 0.0015 = -0.0985$$
$$\theta_5 = \theta_5 + \Delta \theta_5 = 0.3 + 0.0127 = 0.3127$$

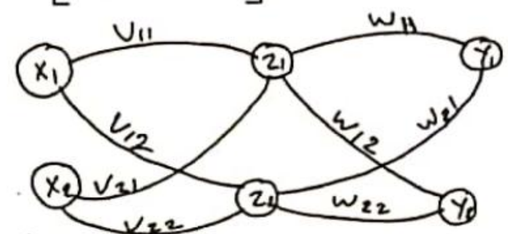
The training process is repeated until the sum of squared errors is less than 0.001

Can we now draw decision boundaries constructed by the multilayer network for operation Exclusive-OR?

Q/ Suppose you have BP- ANN with 2-input, 2-hiddden , 2-output nodes with **Hyperbolic function** and the following matrices weight, trace with 1-iteration.

Where $\alpha = 0.72$, $\eta = 0.34$, $x = (1,0)$, and $T_k = 1$

$$V = \begin{bmatrix} 0.2 & -0.3 \\ -0.1 & 0.4 \end{bmatrix} \qquad W = \begin{bmatrix} 0.1 & -0.3 \\ 0.7 & 0.8 \end{bmatrix}$$



① Forward phase :-

$$Z\_in1 = X_1 V_{11} + X_2 V_{21}$$
$$= (1 * 0.2) + (0 * (-0.1))$$
$$= 0.2$$

$$Z\_in2 = X_1 V_{12} + X_2 V_{22}$$
$$= (1 * (-0.3)) + (0 * (0.4))$$
$$= -0.3$$

$$Z_1 = \frac{e^{z_1} - e^{-z_1}}{e^{z_1} + e^{-z_1}} = \frac{e^{0.2} - e^{-0.2}}{e^{0.2} + e^{-0.2}}$$

$$\boxed{Z_1 = 0.197}$$

$$Z_2 = \frac{e^{z_2} - e^{-z_2}}{e^{z_2} + e^{-z_2}} = \frac{e^{-0.3} - e^{-(-0.3)}}{e^{-0.3} + e^{-(-0.3)}}$$

$$\boxed{Z_2 = -0.291}$$

$$Y\_in1 = Z_1 W_{11} + Z_2 W_{21}$$
$$= [0.197 * 0.1] + [-0.291 * 0.7]$$
$$= -0.184$$

$$Y\_in2 = Z_1 W_{12} + Z_2 W_{22}$$
$$= [0.197 * (-0.3)] + [-0.291 * 0.8]$$
$$= -0.291$$

$$Y_1 = \frac{e^{-0.184} - e^{-(-0.184)}}{e^{-0.184} + e^{-(-0.184)}} =$$

$$\boxed{Y_1 = -0.181}$$

$$Y_2 = \frac{e^{-0.291} - e^{-(-0.291)}}{e^{-0.291} + e^{-(-0.291)}}$$

$$\boxed{Y_2 = -0.283}$$

② Backward Phase :-

$\delta y_1 = y_1(1-y_1) * (T_K - y_1)$

$= -0.181(1-(-0.181)) * (1-(-0.181))$

$\boxed{\delta y_1 = -0.252}$

$\delta y_2 = y_2(1-y_2) * (T_K - y_2)$

$= -0.283(1+0.283) * (1+0.283)$

$\boxed{\delta y_2 = -0.465}$

$\delta z_1 = z_1(1-z_1) * [(\delta y_1 * w_{11}) + (\delta y_2 * w_{21})]$

$= 0.197(1-0.197) * [(-0.252 * 0.1) + (-0.465 * 0.7)]$

$= 0.158 * [-0.0252 + (-0.3255)]$

$\boxed{\delta z_1 = -0.0553}$

$\delta z_2 = z_2(1-z_2) * [(\delta y_1 * w_{12}) + (\delta y_2 * w_{22})]$

$= -0.291(1-(-0.291)) * [(-0.252) * (-0.3) + ((-0.465) * 0.8)]$

$\boxed{\delta z_2 = 0.111}$

③ Update weights :-

$w_{11new} = \eta * \delta y_1 * z_1 + \alpha * [w_{11old}]$

$= [0.34 * (-0.25) * 0.197] + [0.72 * (0.1)]$

$\boxed{w_{11new} = 0.055}$

$w_{12new} = \eta * \delta y_2 * z_2 + \alpha * [w_{12old}]$

$= 0.34 * (-0.46) * (-0.291) + (0.72 * (-0.3))$

$\boxed{w_{12new} = -0.246}$

$w_{21new} = \eta \delta y_1 z_2 + \alpha [w_{21old}]$

$= [0.34 * (-0.25) * (-0.291)] + [0.72 * 0.7]$

$\boxed{w_{21new} = 0.528}$

$w_{22new} = \eta \delta y_2 z_2 + \alpha [w_{22old}]$

$= [0.34 * (-0.46) * (-0.291)] + [0.72 * 0.8]$

$\boxed{w_{22new} = 0.621}$

$V_{11new} = \eta * \delta z_1 * X_1 + \alpha V_{11old}$

$= (0.34 * (-0.55) * 1) + (0.72 * 0.2)$

$\boxed{V_{11new} = -0.043}$

$V_{12new} = \eta \delta z_2 * X_1 + \alpha V_{12old}$

$= (0.34 * 0.11 * 1) + (0.72 * (-0.3))$

$\boxed{V_{12new} = -0.178}$

$V_{21new} = \eta \delta z_1 * X_2 + \alpha V_{21old}$

$= (0.34 * (-0.055) * 0 + (0.72 * (-0.1))$

$\boxed{V_{21new} = -0.072}$

$V_{22new} = \eta \delta z_2 * X_2 + \alpha V_{22old}$

$= 0 + 0.72 * 0.4$

$\boxed{V_{22new} = 0.288}$