CHAPTER 2

ARTIFICIAL NEURAL NETWORKS: AN INTRODUCTION

"Principles of Soft Computing, 2nd Edition"

by S.N. Sivanandam & SN Deepa

Back Propagation Algorithm

To derive a convenient expression for $\frac{\partial E_d}{\partial net_j}$

We consider two cases in turn:

- Case 1, where unit j is an output unit for the network, and
- Case 2, where unit j is an internal unit of the network.

Back Propagation Algorithm

Case 1: Training Rule for Output Unit Weights

- Just as ***wji*** can influence the rest of the network only through net_j, net_j can influence the network only through ***oj***. Therefore, we can invoke the chain rule again to write,

$$\frac{\partial E_d}{\partial net_j} = \frac{\partial E_d}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

$$\frac{\partial E_d}{\partial o_j} = \frac{\partial}{\partial o_j} \frac{1}{2} (t_j - o_j)^2$$

$$= \frac{1}{2} 2(t_j - o_j) \frac{\partial (t_j - o_j)}{\partial o_j}$$

$$= -(t_j - o_j)$$

$$\frac{\partial o_j}{\partial (net_j)} = \frac{\partial \sigma(net_j)}{\partial (net_j)}$$

$$\frac{\partial \sigma(x)}{\partial (x)} = \sigma(x)\,(1 - \sigma(x))$$

$$= \sigma(net_j)(1 - \sigma(net_j))$$

$$= o_j\,(1 - o_j)$$

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)\, o_j(1 - o_j)$$

Back Propagation Algorithm

Case 1: Training Rule for Output Unit Weights

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)\, o_j(1 - o_j)$$

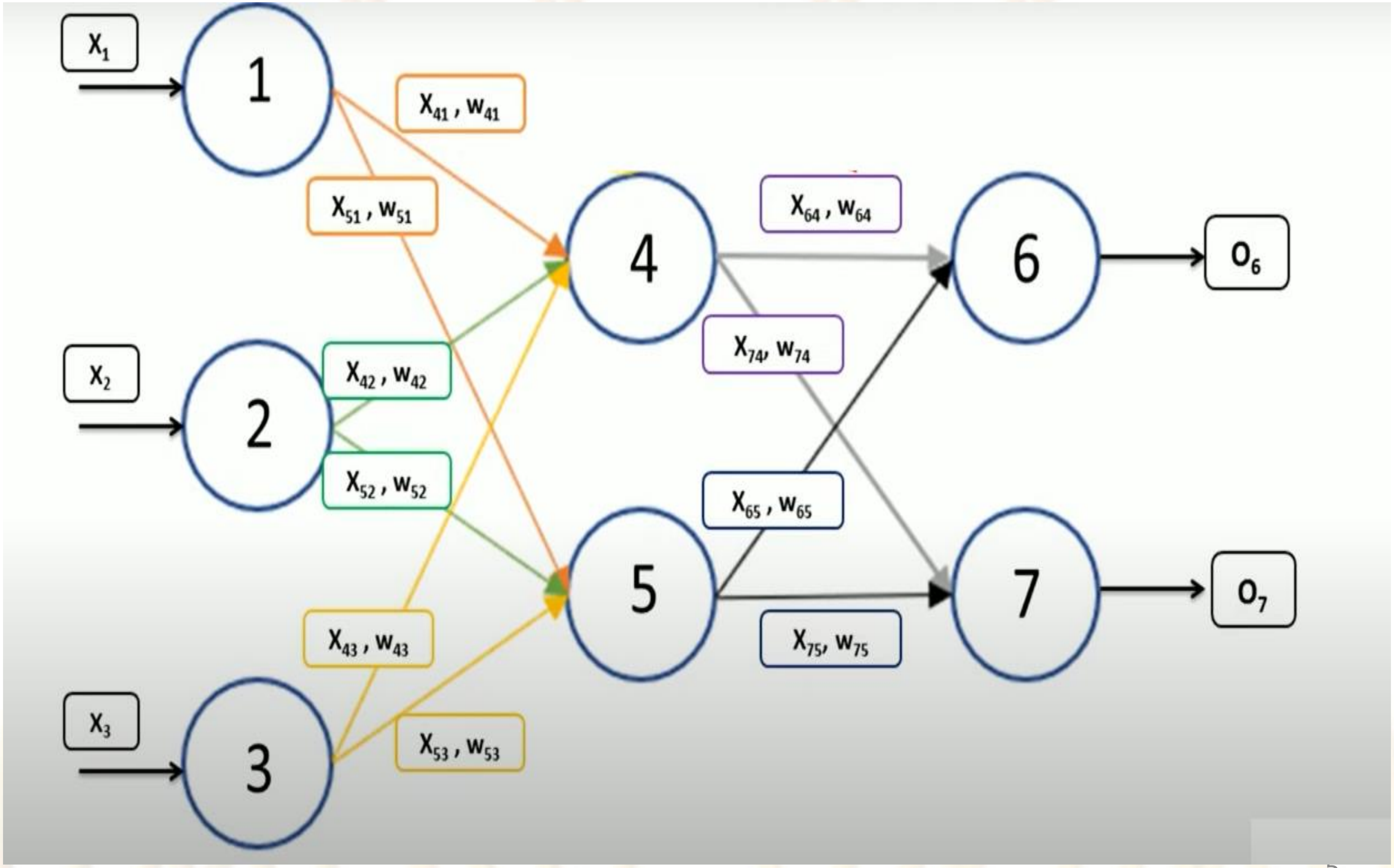
$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$$

$$\Delta w_{ji} = \eta\, \underline{(t_j - o_j)\, o_j(1 - o_j)} x_{ji}$$

$$\delta_j = (t_j - o_j)\, o_j(1 - o_j)$$

$$\Delta w_{ji} = \eta\, \delta_j x_{ji}$$

Back Propagation Algorithm



Ref- vtupulse.com

Back Propagation Algorithm

Case 2: Training Rule for Hidden Unit Weights

$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} \frac{\partial E_d}{\partial net_k} \frac{\partial net_k}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \frac{\partial net_k}{\partial o_j} \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \, w_{kj} \frac{\partial o_j}{\partial net_j}$$

$$= \sum_{k \in Downstream(j)} -\delta_k \, w_{kj} \, o_j(1 - o_j)$$

$$\frac{\partial E_d}{\partial net_j} = -(t_j - o_j)\, o_j(1 - o_j)$$

$$\delta_j = (t_j - o_j)\, o_j(1 - o_j)$$

$$\frac{\partial \, net_k}{\partial o_j} = \frac{\partial x_{kj} w_{kj}}{\partial o_j} = \frac{\partial o_j w_{kj}}{\partial o_j}$$

$$\frac{\partial o_j}{\partial (net_j)} = \frac{\partial \sigma(net_j)}{\partial (net_j)}$$

$$= \sigma(net_j)(1 - \sigma(net_j))$$

$$= o_j(1 - o_j)$$

Ref- vtupulse.com

Back Propagation Algorithm

Case 2: Training Rule for Hidden Unit Weights

$$\Delta w_{ji} = -\eta \frac{\partial E_d}{\partial net_j} x_{ji}$$

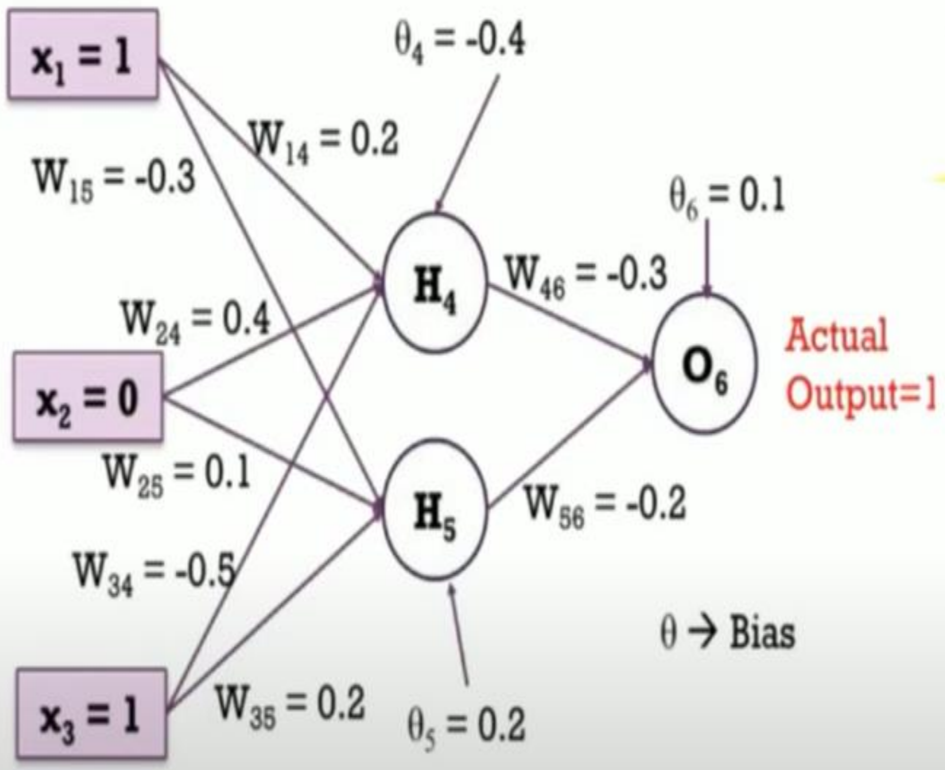
$$\frac{\partial E_d}{\partial net_j} = \sum_{k \in Downstream(j)} -\delta_k \, w_{kj} \, o_j(1 - o_j)$$

$$\Delta w_{ji} = \eta \, o_j(1 - o_j) \sum_{k \in Downstream(j)} \delta_k \, w_{kj} \, x_{ji}$$

$$\Delta w_{ji} = \eta \, \delta_j \, x_{ji}$$

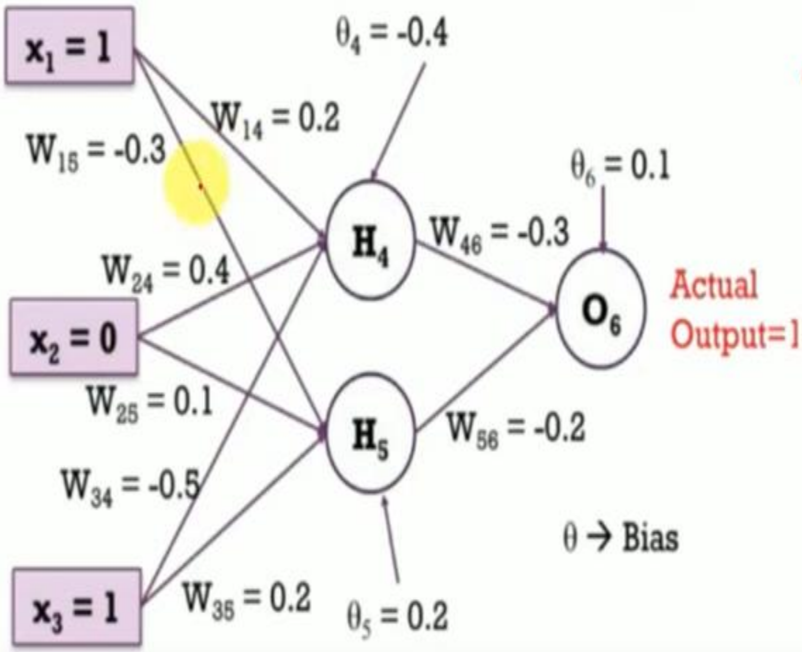
$$\delta_j = o_j(1 - o_j) \sum_{k \in Downstream(j)} \delta_k \, w_{kj}$$

Back Propagation Algorithm Example



Assume that the neurons have a sigmoid activation function, perform a forward pass and a backward pass on the network. Assume that the actual output of y is 1 and learning rate is 0.9. Perform another forward pass.

Back Propagation Algorithm Example



- Forward Pass: Compute output for y4, y5 and y6.

$$a_j = \sum_j (w_{i,j} * x_i) \qquad yj = F(aj) = \frac{1}{1+e^{-a_j}}$$

$a_4 = (w_{14} * x_1) + (w_{24} * x_2) + (w_{34} * x_3) + \theta_4$

$= (0.2 * 1) + (0.4 * 0) + (-0.5 * 1) + (-0.4) = -0.7$

$O(H_4) = y_4 = f(a_4) = 1/(1 + e^{0.7}) = 0.332$

$a_5 = (w_{15} * x_1) + (w_{25} * x_2) + (w_{35} * x_3) + \theta_5$

$= (-0.3 * 1) + (0.1 * 0) + (0.2 * 1) + (0.2) = 0.1$

$O(H_5) = y_5 = f(a_5) = 1/(1 + e^{-0.1}) = 0.525$

$a_6 = (w_{46} * H_4) + (w_{56} * H_5) + \theta_6$

$= (-0.3 * 0.332) + (-0.2 * 0.525) + 0.1 = -0.105$

$O(O_6) = y_6 = f(a_6) = 1/(1 + e^{0.105}) = \mathbf{0.474}$

$\text{Error} = y_{target} - y_6 = 0.526$

Back Propagation Algorithm Example

- Each weight changed by:

$$\Delta w_{ji} = \eta \delta_j o_i$$

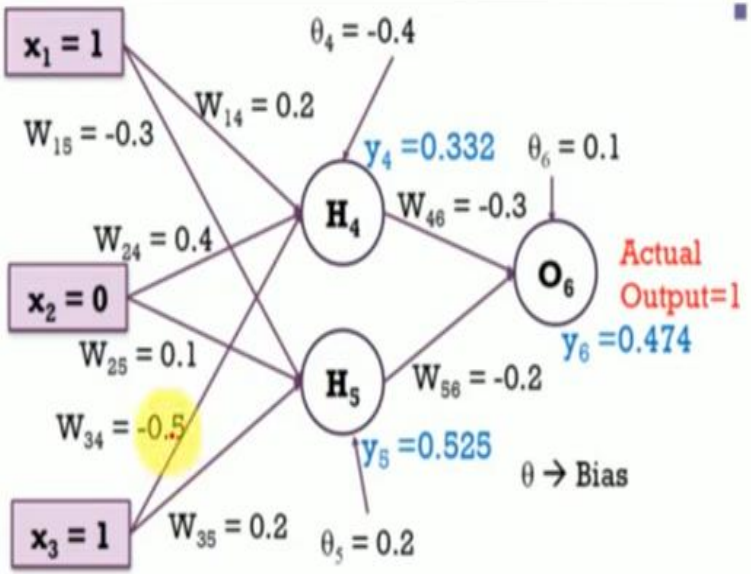
$$\delta_j = o_j(1 - o_j)(t_j - o_j) \qquad \text{if } j \text{ is an output unit}$$

$$\delta_j = o_j(1 - o_j)\sum_k \delta_k w_{kj} \qquad \text{if } j \text{ is a hidden unit}$$

- where η is a constant called the learning rate
- tj is the correct teacher output for unit j
- δj is the error measure for unit j

Ref- vtupulse.com

Back Propagation Algorithm Example



- Backward Pass: Compute δ4, δ5 and δ6.

For output unit:

$$\delta_6 = y_6(1-y_6)(y_{target} - y_6)$$
$$= 0.474*(1-0.474)*(1-0.474) = \underline{0.1311}$$

For hidden unit:

$$\delta_5 = y_5(1-y_5)\, w_{56} * \delta_6$$
$$= 0.525*(1 - 0.525)*(-0.2 * 0.1311) = \underline{-0.0065}$$

$$= y_4(1-y_4)\, w_{46} * \delta_6$$
$$= 0.332*(1 - 0.332)* (-0.3 * 0.1331) = \underline{-0.0087}$$

Compute new weights

$$\Delta w_{ji} = \eta \delta_j o_i$$

$\Delta w_{46} = \eta\, \delta_6\, y_4 = 0.9 * 0.1311 * 0.332 = 0.03917$ ✓

w_{46} (new) $= \Delta w_{46} + w_{46}$(old) $= 0.03917 + (-0.3) = \underline{-0.261}$

$\Delta w_{14} = \eta\, \delta_4\, x_1 = 0.9 * -0.0087 * 1 = \underline{-0.0078}$

w_{14} (new) $= \Delta w_{14} + w_{14}$(old) $= -0.0078 + 0.2 = \underline{0.192}$

Back Propagation Algorithm Example

- Similarly, update all other weights

i	j	w_{ij}	δ_j	x_i	η	Updated w_{ij}
4	6	-0.3	0.1311	0.332	0.9	-0.261
5	6	-0.2	0.1311	0.525	0.9	-0.138
1	4	0.2	-0.0087	1	0.9	0.192
1	5	-0.3	-0.0065	1	0.9	-0.306
2	4	0.4	-0.0087	0	0.9	0.4
2	5	0.1	-0.0065	0	0.9	0.1
3	4	-0.5	-0.0087	1	0.9	-0.508
3	5	0.2	-0.0065	1	0.9	0.194

Back Propagation Algorithm Example

- Similarly, update bais weights

θ_j	Previous θ_j	δ_j	η	Updated θ_j
Θ_6	0.1	0.1311	0.9	0.218
Θ_5	0.2	-0.0065	0.9	0.194
Θ_4	-0.4	-0.0087	0.9	-0.408