SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of AI &DS

Course Name – 19AD602 DEEP LEARNING

III Year / VI Semester

Unit 2-DEEP NETWORKS

Topic: Back propagation

Case Study

An e-commerce company uses a deep neural network to predict customer purchase behavior. The backpropagation algorithm updates the network's weights by minimizing the error between predicted and actual outcomes, improving the recommendation accuracy by 25%. This optimization enhanced user engagement and sales.

Back Propagation Algorithm

BACKPROPAGATION (training_example, η, n_{in}, n_{out}, n_{hidden})

- Each training example is a pair of the form (x, t), where (x) is the vector of network input values, and (t) is the vector of target network output values.
- η is the learning rate (e.g., 0.05).
- n_i, is the number of network inputs,
- n_{hidden} the number of units in the hidden layer, and
- n_{out} the number of output units.
- The input from unit i into unit j is denoted x_{ji}, and the weight from unit i to unit j is denoted w_{ji}

Back Propagation Algorithm

- Create a feed-forward network with n_i inputs, n_{hidden} hidden units, and n_{out} output units.
- Initialize all network weights to small random numbers
- Until the termination condition is met, Do
  - For each (x, t), in training examples, Do
    - Propagate the input forward through the network:

      1. Input the instance x, to the network and compute the output o_u of every unit u in the network.
    - Propagate the errors backward through the network

      2. For each network unit k, calculate its error term δ_k

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

      3. For each network unit h, calculate its error term δ_h

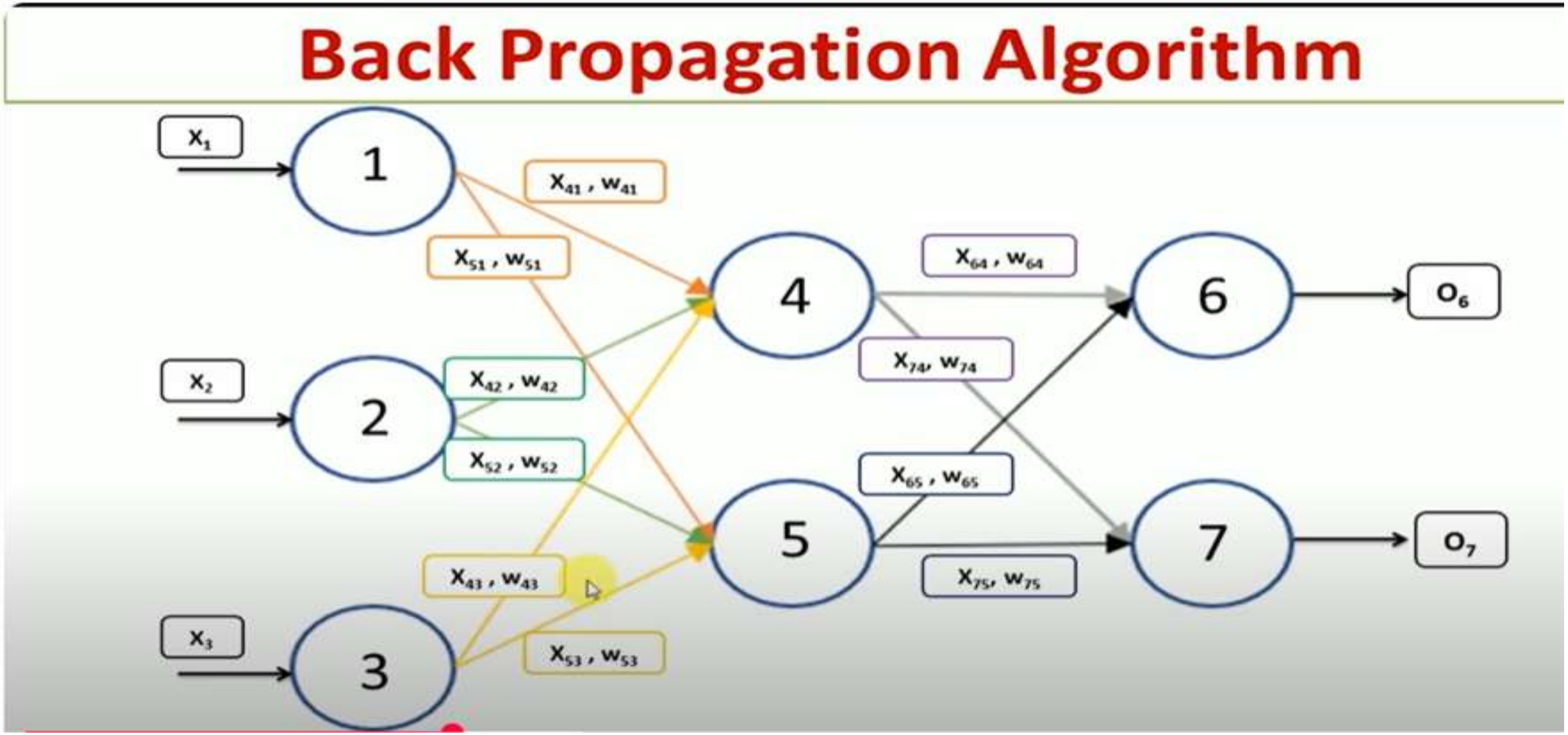
$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

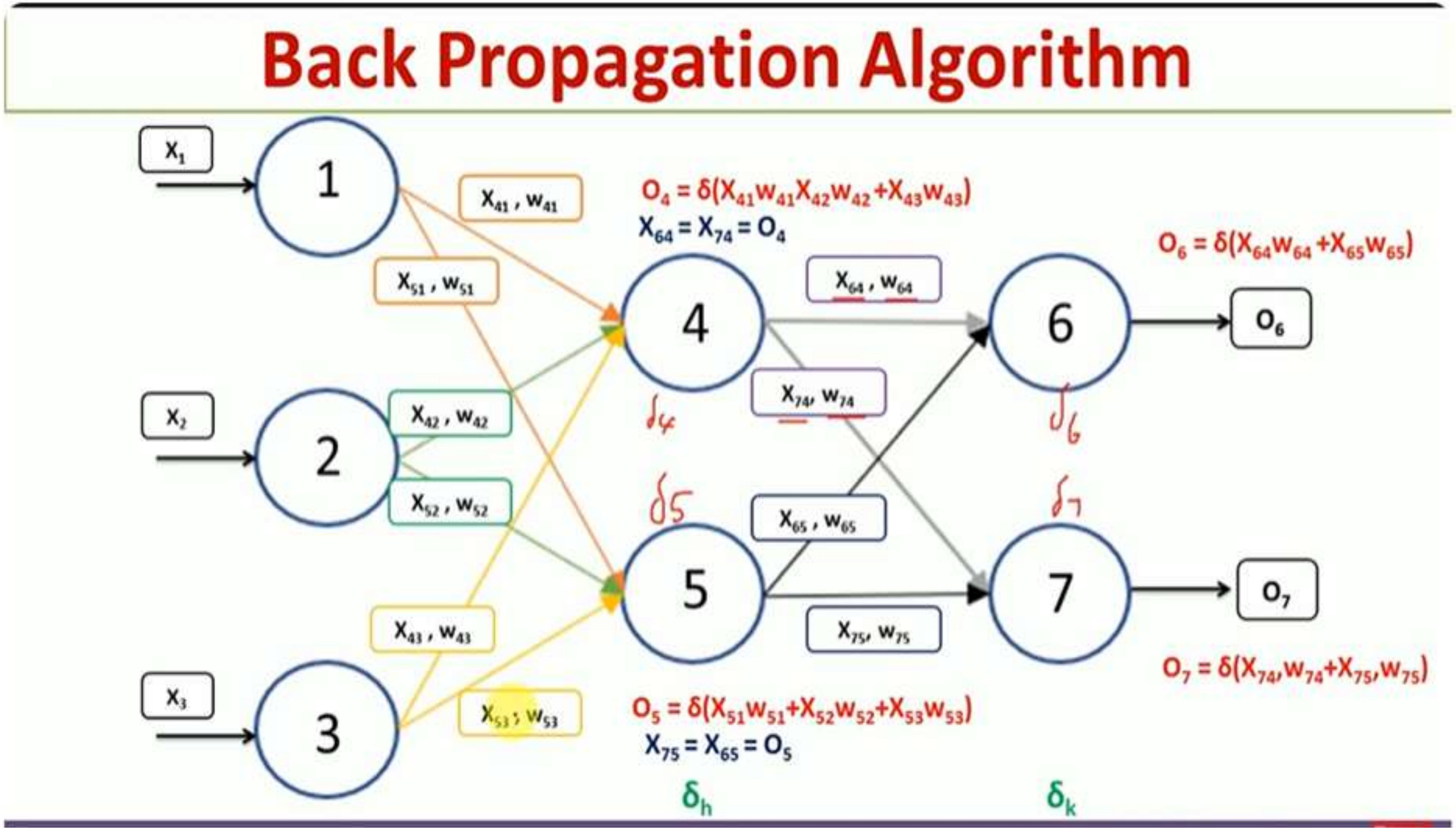
4. Update each network weight w_{ji}

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

Where

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


Back Propagation Algorithm
x1
x2
x3
1
2
3
4
5
6
7
x41 , w41
x51 , w51
x42 , w42
x52 , w52
x43 , w43
x53 , w53
x64 , w64
x74, w74
x65 , w65
x75, w75
o6
o7


Back Propagation Algorithm
X1
X2
X3
1
2
3
4
5
6
7
X41 , W41
X51 , W51
X42 , W42
X52 , W52
X43 , W43
X53 , W53
X64 , W64
X74, W74
X65 , W65
X75, W75
O6
O7
O4 = δ(X41W41X42W42+X43W43)
X64 = X74 = O4
O6 = δ(X64W64 +X65W65)
O5 = δ(X51W51+X52W52+X53W53)
X75 = X65 = O5
O7 = δ(X74,W74+X75,W75)
δh
δk

Derivation of Back Propagation Algorithm

- To derive the equation for updating weights in back propagation algorithm, we use Stochastic gradient descent rule.
- Stochastic gradient descent involves iterating through the training examples one at a time, for each training example **d** descending the gradient of the error ***Ed*** with respect to this single example.
- In other words, for each training example **d** every weight ***wji*** is updated by adding to it Δw_{ij}.
- That is,

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

Where

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

Derivation of Back Propagation Algorithm

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

Where

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

- where *Ed* is the error on training example **d**, that is half the squared difference between the target output and the actual output over all output units in the network,

$$E_d(\vec{w}) \equiv \frac{1}{2} \sum_{k \in outputs} (t_k - o_k)^2$$

- Here outputs is the set of output units in the network, $\mathbf{t_k}$ is the target value of unit k for training example **d**, and $\mathbf{o_k}$ is the output of unit k given training example **d**.

Derivation of Back Propagation Algorithm

Notation Used:

x_{ji} = the ith input to unit j

w_{ji} = the weight associated with the ith input to unit j

$net_j = \sum_i w_{ji} X_{ji}$ (the weighted sum of inputs for unit j)

o_j = the output computed by unit j

t_j = the target output for unit j

σ = the sigmoid function

outputs = the set of units in the final layer of the network

Downstream(j) = the set of units whose immediate inputs include the output of unit j

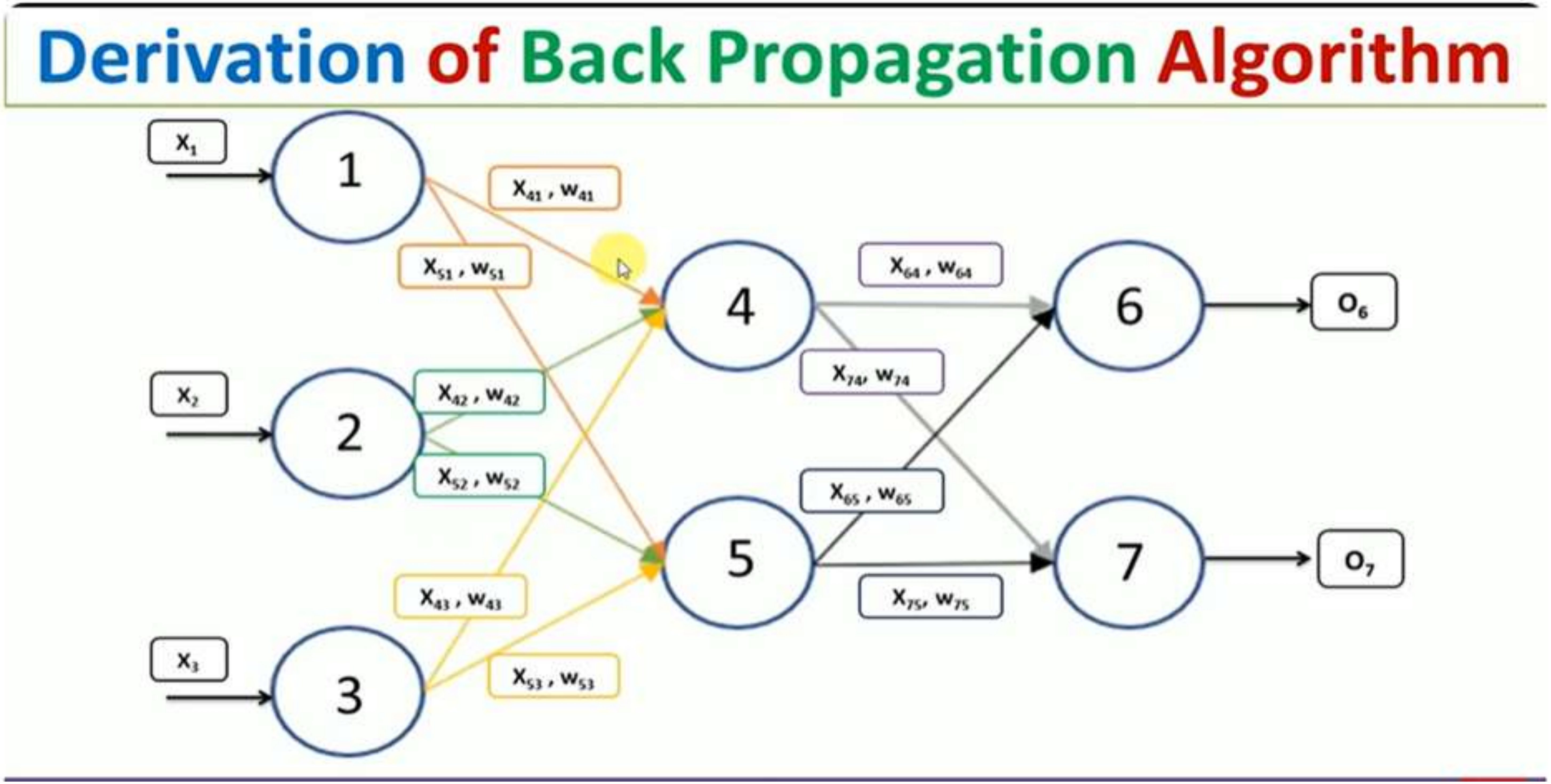
Derivation of Back Propagation Algorithm

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

Where

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

- To begin, notice that weight ***wji*** can influence the rest of the network only through ***netj.*** Therefore, we can use the chain rule to write,


Derivation of Back Propagation Algorithm
x1
x2
x3
1
2
3
4
5
6
7
X41 , W41
X51 , W51
X42 , W42
X52 , W52
X43 , W43
X53 , W53
X64 , W64
X74, W74
X65 , W65
X75, W75
O6
O7

Derivation of Back Propagation Algorithm

$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

Where

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

- To begin, notice that weight ***wji*** can influence the rest of the network only through ***netj.*** Therefore, we can use the chain rule to write,

$$\frac{\partial E_d}{\partial w_{ji}} = \frac{\partial E_d}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}}$$

$$= \frac{\partial E_d}{\partial net_j} x_{ji}$$

$$net_j = \sum_i w_{ji} x_{ji}$$

$$\frac{\partial net_j}{\partial w_{ji}} = x_{ji}$$

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

- Our remaining task is to derive a convenient expression for $\frac{\partial E_d}{\partial net_j}$

Derivation of 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.

Derivation of 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,

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

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

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

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

$$= -(t_j - o_j)$$

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

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

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

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

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

Derivation of Back Propagation Algorithm

Case 1: Training Rule for Output Unit Weights

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

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

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

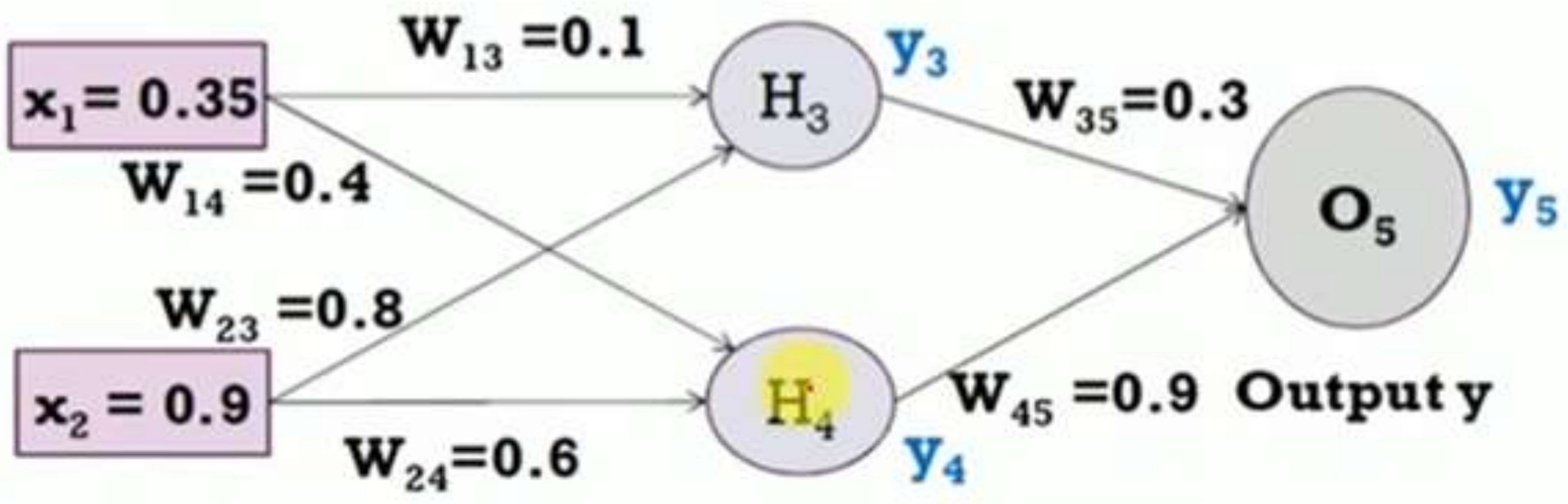
$$\Delta w_{ji} = \eta\, (t_j - o_j)\, o_j(1 - o_j) x_{ji}$$

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

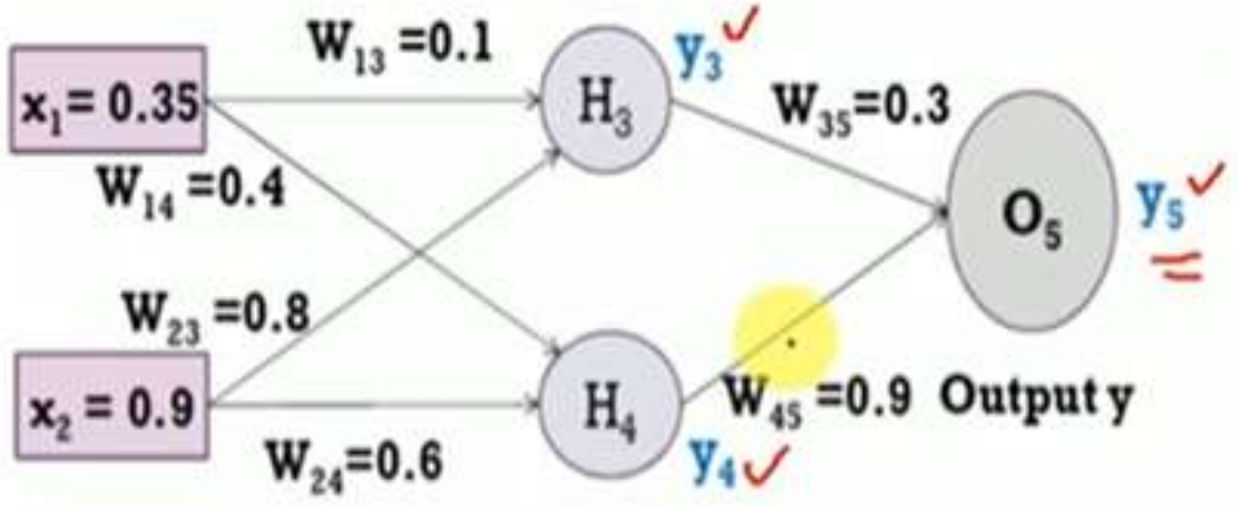
$$\Delta w_{ji} = \eta\, \delta_j\, x_{ji}$$

Back Propagation Solved Example - 1

- 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 0.5 and learning rate is 1. Perform another forward pass.



Back Propagation Solved Example - 1



$$Error = y_{target} - y_5 = -0.19$$

0.5 - 0.69

- Forward Pass: Compute output for y3, y4 and y5.

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

$$a_1 = (w_{13} * x_1) + (w_{23} * x_2)$$
$$= (0.1 * 0.35) + (0.8 * 0.9) = 0.755$$
$$y_3 = f(a_1) = 1/(1 + e^{-0.755}) = 0.68$$

$$a_2 = (w_{14} * x_1) + (w_{24} * x_2)$$
$$= (0.4 * 0.35) + (0.6 * 0.9) = 0.68$$
$$y_4 = f(a_2) = 1/(1 + e^{-0.68}) = 0.6637$$

$$a_3 = (w_{35} * y_3) + (w_{45} * y_4)$$
$$= (0.3 * 0.68) + (0.9 * 0.6637) = 0.801$$
$$y_5 = f(a_3) = 1/(1 + e^{-0.801}) = 0.69 \text{ (Network Output)}$$

Back Propagation Solved Example - 1

- Each weight changed by:

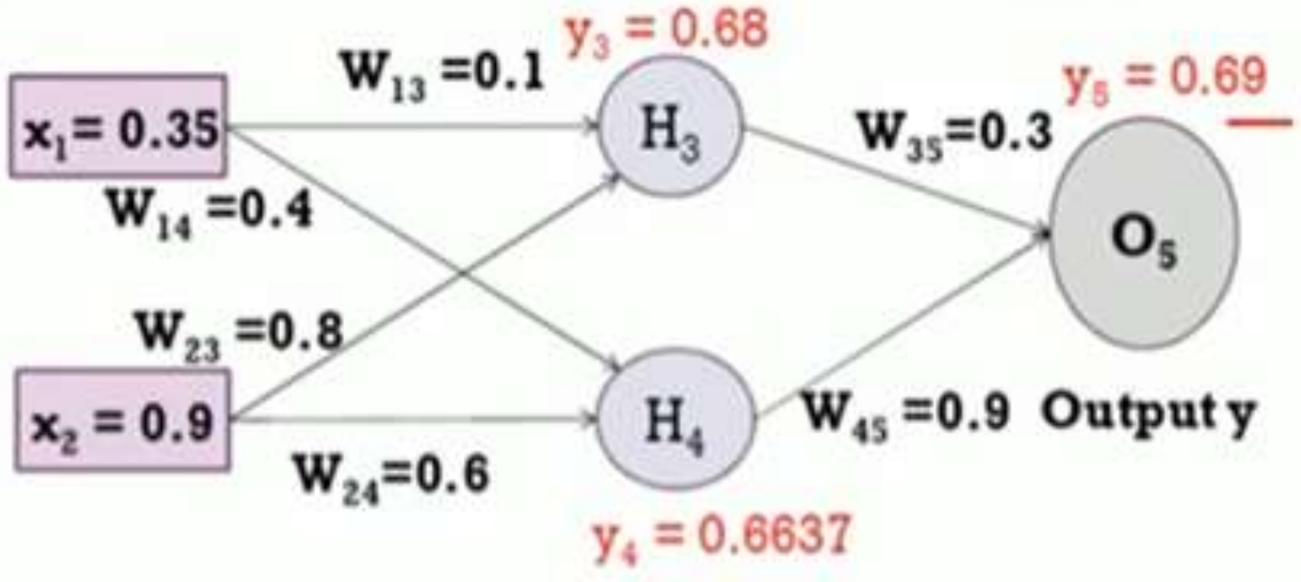
$$\Delta w_{ji} = \eta \delta_j o_i$$

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

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

- 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

Back Propagation Solved Example - 1



- Backward Pass: Compute δ3, δ4 and δ5.

For output unit:

$\delta_5 = y(1-y)\,(y_{target} - y)$

$= 0.69*(1-0.69)*(0.5-0.69) = -0.0406$

For hidden unit:

$\delta_3 = y_3(1-y_3)\, w_{35} * \delta_5$

$= 0.68*(1 - 0.68)*(0.3 * -0.0406) = -0.00265$

$\delta_4 = y_4(1-y_4) w_{45} * \delta_5$

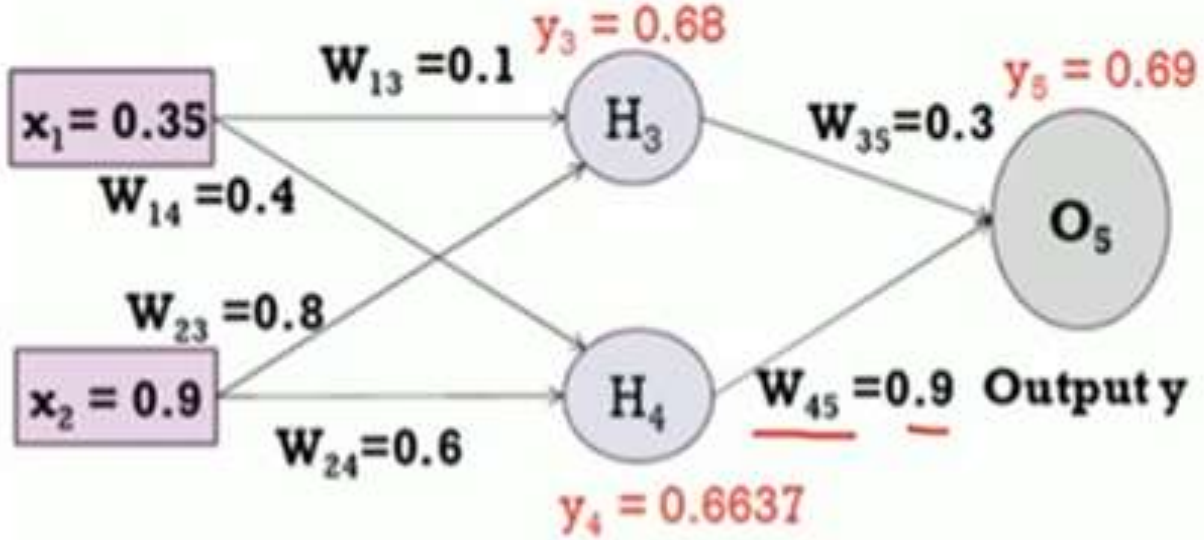
$= 0.6637*(1 - 0.6637)* (0.9 * -0.0406) = -0.0082$

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

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

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

Back Propagation Solved Example - 1



- Backward Pass: Compute δ3, δ4 and δ5.

For output unit:

$\delta_5 = y(1-y)(y_{target} - y)$

$= 0.69*(1-0.69)*(0.5-0.69) = -0.0406$

For hidden unit:

$\delta_3 = y_3(1-y_3)\, w_{35} * \delta_5$

$= 0.68*(1 - 0.68)*(0.3 * -0.0406) = -0.00265$

$\delta_4 = y_4(1-y_4) w_{45} * \delta_5$

$= 0.6637*(1 - 0.6637)* (0.9 * -0.0406) = -0.0082$

Compute new weights

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

$\Delta w_{45} = \eta\, \delta_5\, y_4 = 1 * -0.0406 * 0.6637 = -0.0269$

w_{45} (new) $= \Delta w_{45} + w_{45}$(old) $= -0.0269 + (0.9) = 0.8731$

$\Delta w_{14} = \eta\, \delta_4\, x_1 = 1 * -0.0082 * 0.35 = -0.00287$

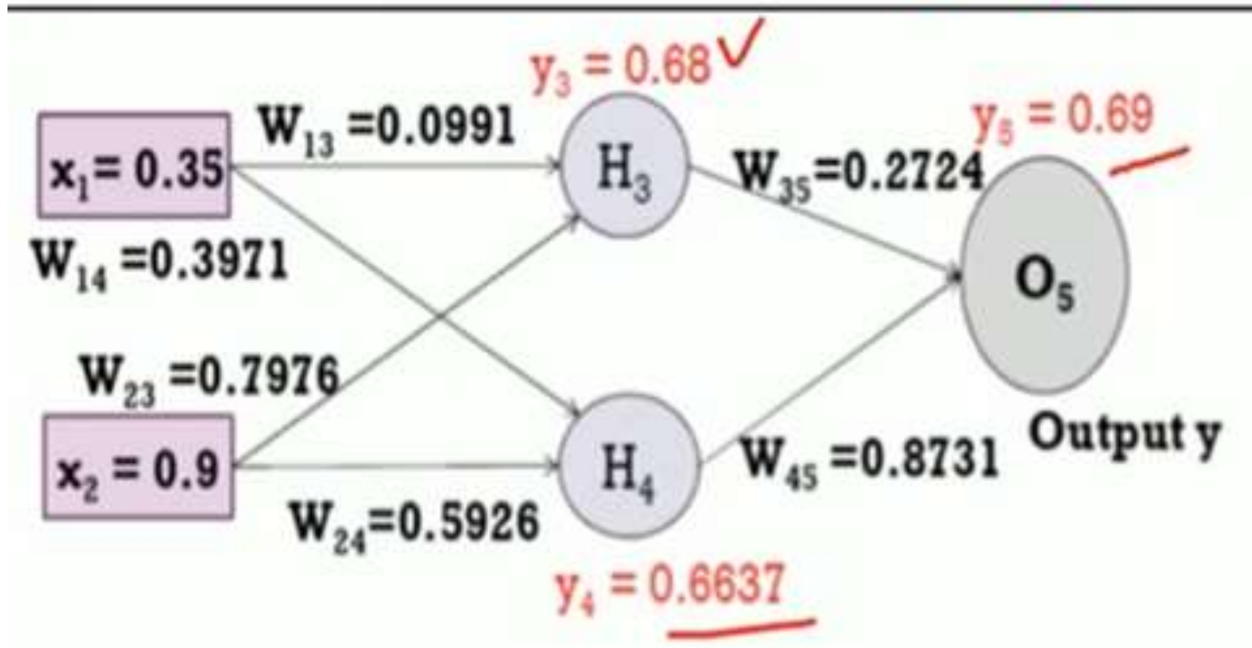
w_{14} (new) $= \Delta w_{14} + w_{14}$(old) $= -0.00287 + 0.4 = 0.3971$

Back Propagation Solved Example - 1

- Similarly, update all other weights

i	j	w_{ij}	δ_j	x_i	η	Updated w_{ij}
1	3	0.1	-0.00265	0.35	1	0.0991
2	3	0.8	-0.00265	0.9	1	0.7976
1	4	0.4	-0.0082	0.35	1	0.3971
2	4	0.6	-0.0082	0.9	1	0.5926
3	5	0.3	-0.0406	0.68	1	0.2724
4	5	0.9	-0.0406	0.6637	1	0.8731

Back Propagation Solved Example - 1



Error $= y_{target} - y_5 = -0.182$

- Forward Pass: Compute output for y3, y4 and y5.

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

$a_1 = (w_{13} * x_1) + (w_{23} * x_2)$
$= (0.0991 * 0.35) + (0.7976 * 0.9) = 0.7525$
$y_3 = f(a_1) = 1/(1 + e^{-0.7525}) = 0.6797$

$a_2 = (w_{14} * x_1) + (w_{24} * x_2)$
$= (0.3971 * 0.35) + (0.5926 * 0.9) = 0.6723$
$y_4 = f(a_2) = 1/(1 + e^{-0.6723}) = 0.6620$

$a_3 = (w_{35} * y_3) + (w_{45} * y_4)$
$= (0.2724 * 0.6797) + (0.8731 * 0.6620) = 0.7631$
$y_5 = f(a_3) = 1/(1 + e^{-0.7631}) = 0.6820$ (Network Output)

Activity

Train a deep neural network on a dataset (e.g., predicting housing prices) and visualize how backpropagation adjusts weights by tracking changes in loss over epochs. Compare results with and without proper weight updates.

THANK YOU