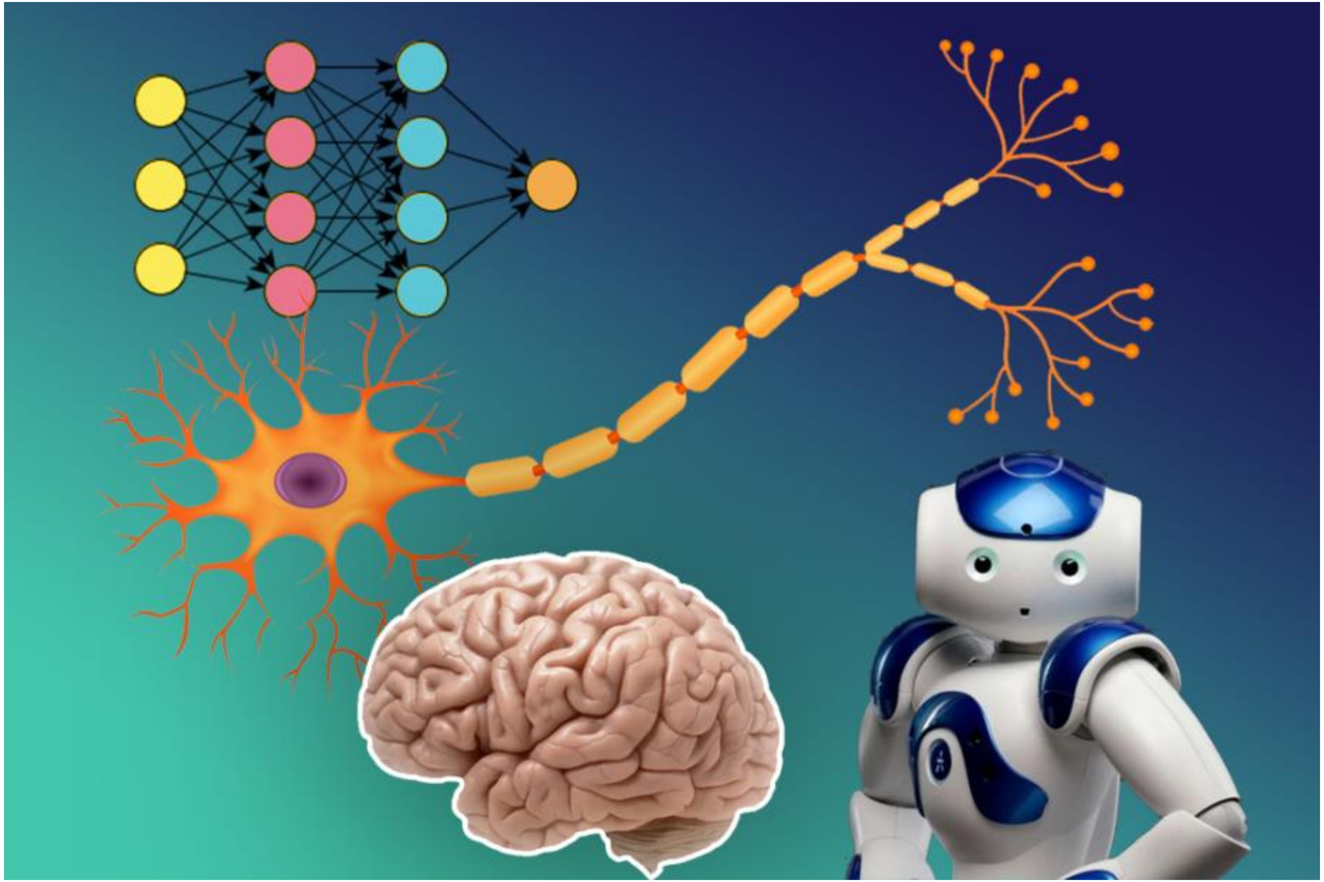
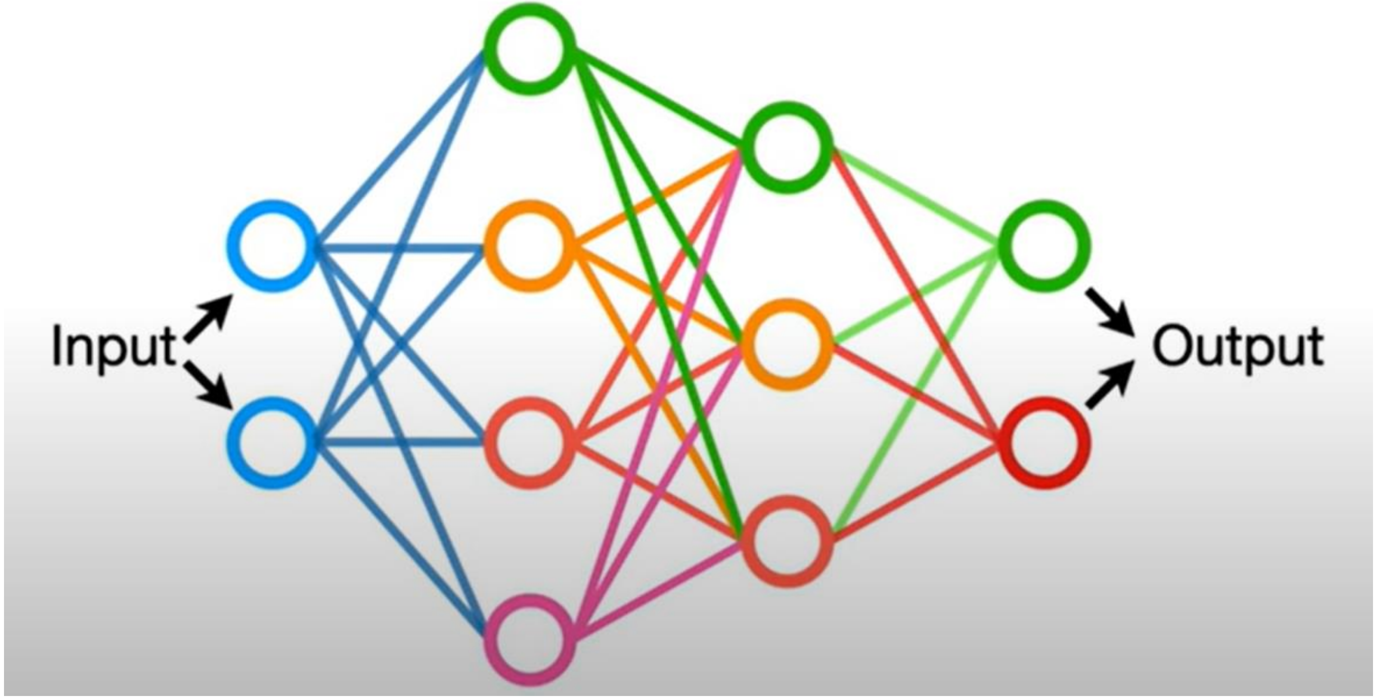
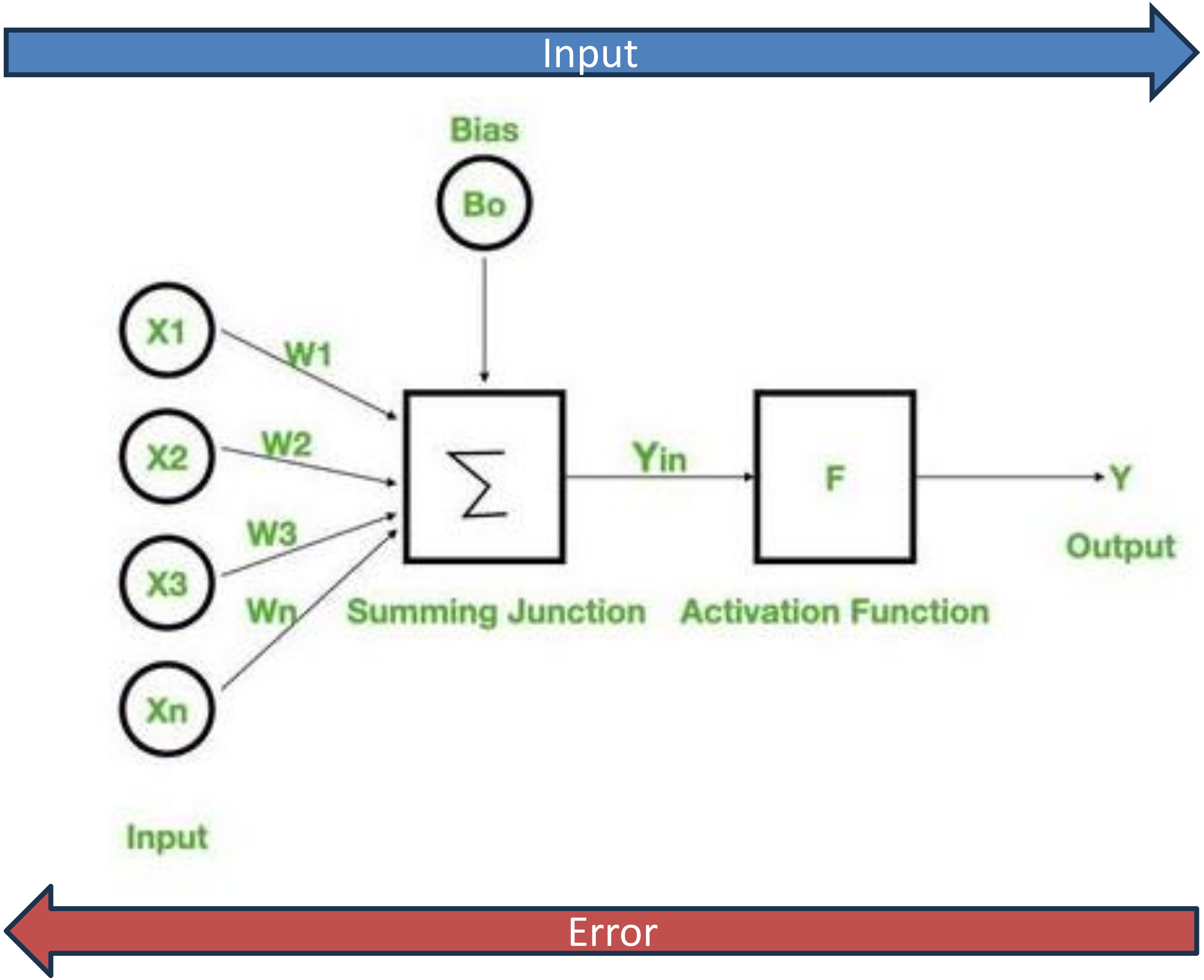
ANN

Dr. Mrunal Rane

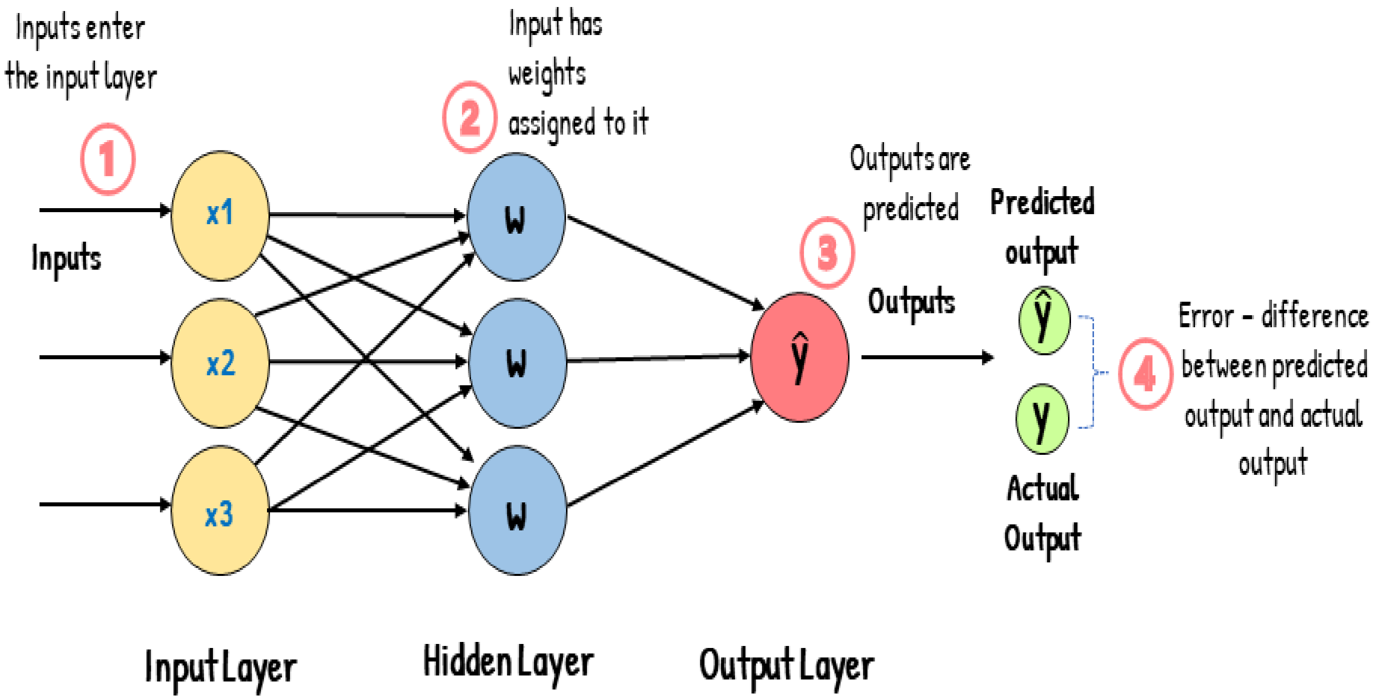


Structure

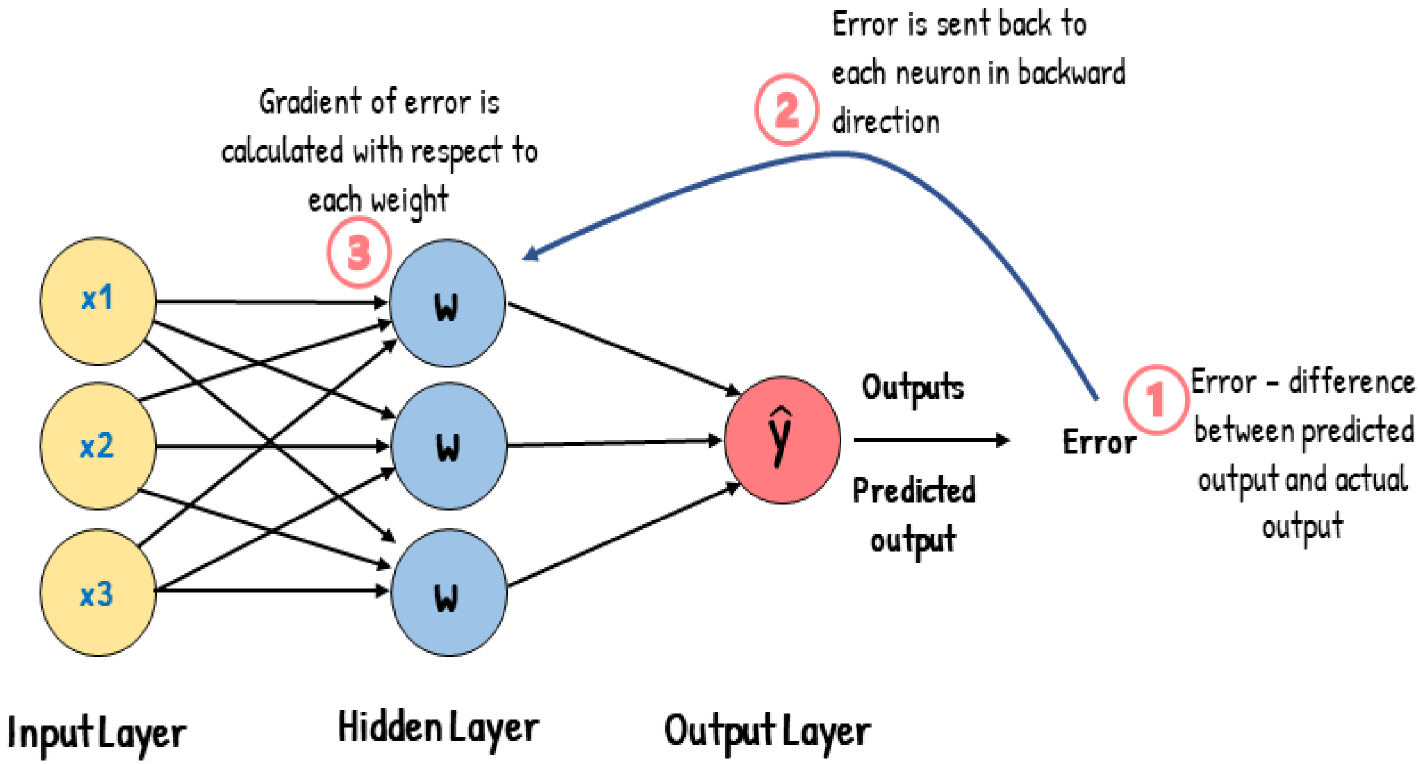



Input
Bias
Bo
X1
X2
X3
Xn
W1
W2
W3
Wn
Yin
F
Y
Output
Summing Junction
Activation Function
Input
Error

Feed-Forward NN



Backpropagation



Back Propagation

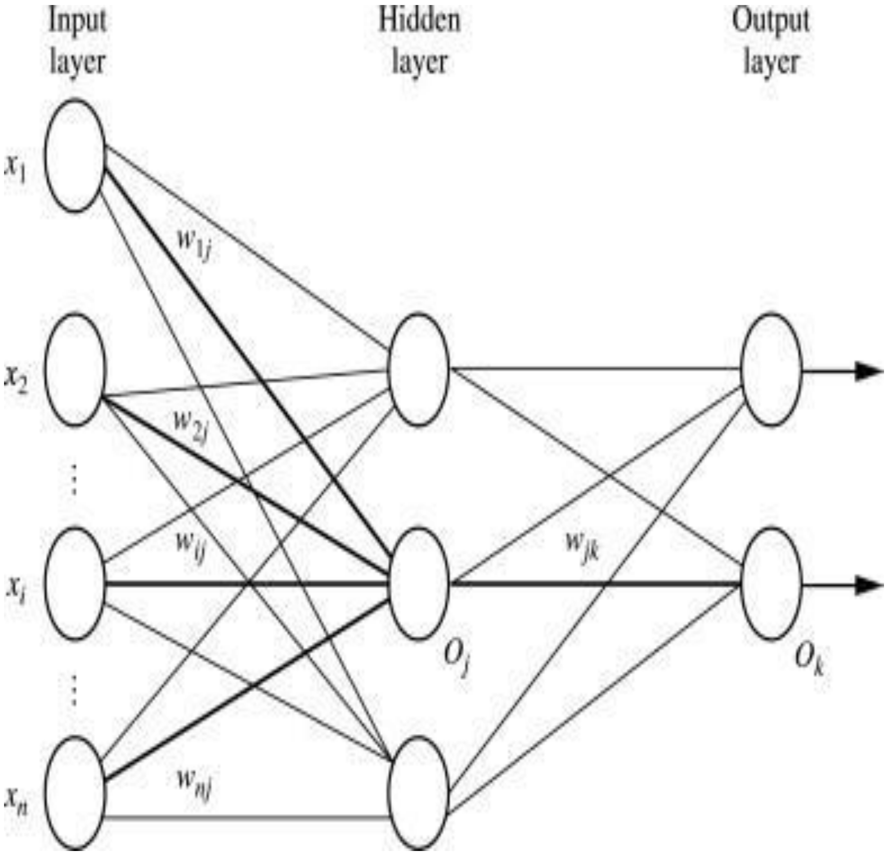
- **Features:**
  - it is the **gradient descent** method as used in the case of simple perceptron network with the differentiable unit.
  - it is different from other networks in respect to the **process by which the weights are calculated** during the learning period of the network.
- training is done in the three stages :
  - the **feed-forward** of input training pattern
  - the calculation and **backpropagation of the error**
  - **updation** of the weight

Algorithm

- Step 1: Inputs X, arrive through the preconnected path.
- Step 2: The input is modeled using true weights W. Weights are usually chosen randomly.
- Step 3: Calculate the output of each neuron from the input layer to the hidden layer to the output layer.
- Step 4: Calculate the error in the outputs

```
Backpropagation Error= Actual Output – Desired Output
```

- Step 5: From the output layer, go back to the hidden layer to adjust the weights to reduce the error.
- Step 6: Repeat the process until the desired output is achieved.



Parameters :

- x = inputs training vector x=(x_1,x_2,............x_n).
- t = target vector t=(t_1,t_2...............t_n).
- δ_k = error at output unit.
- δ_j = error at hidden layer.
- α = learning rate.
- V_{0j} = bias of hidden unit j.

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}

- 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

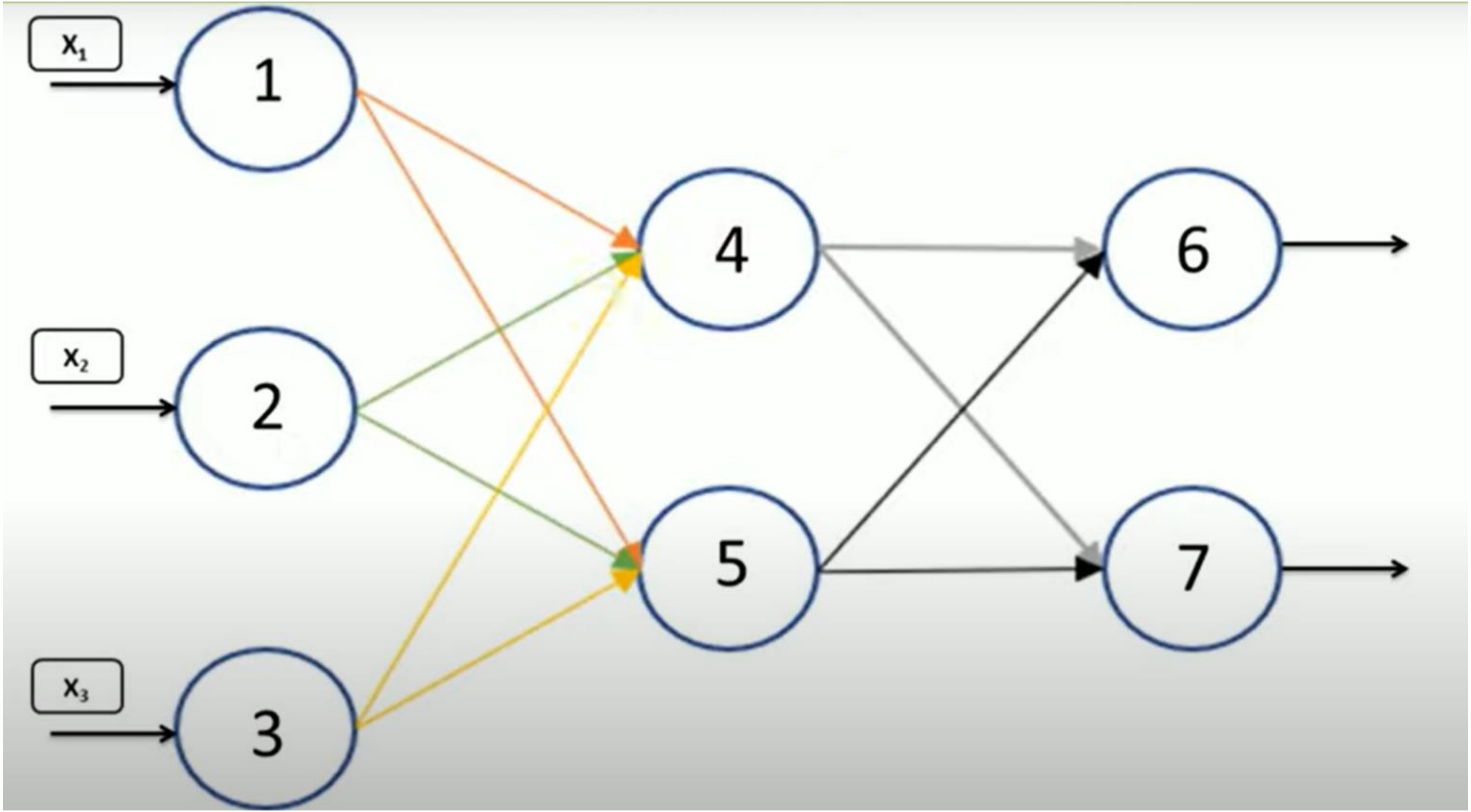
$$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\delta_k$$

4. Update each network weight w_{ji}

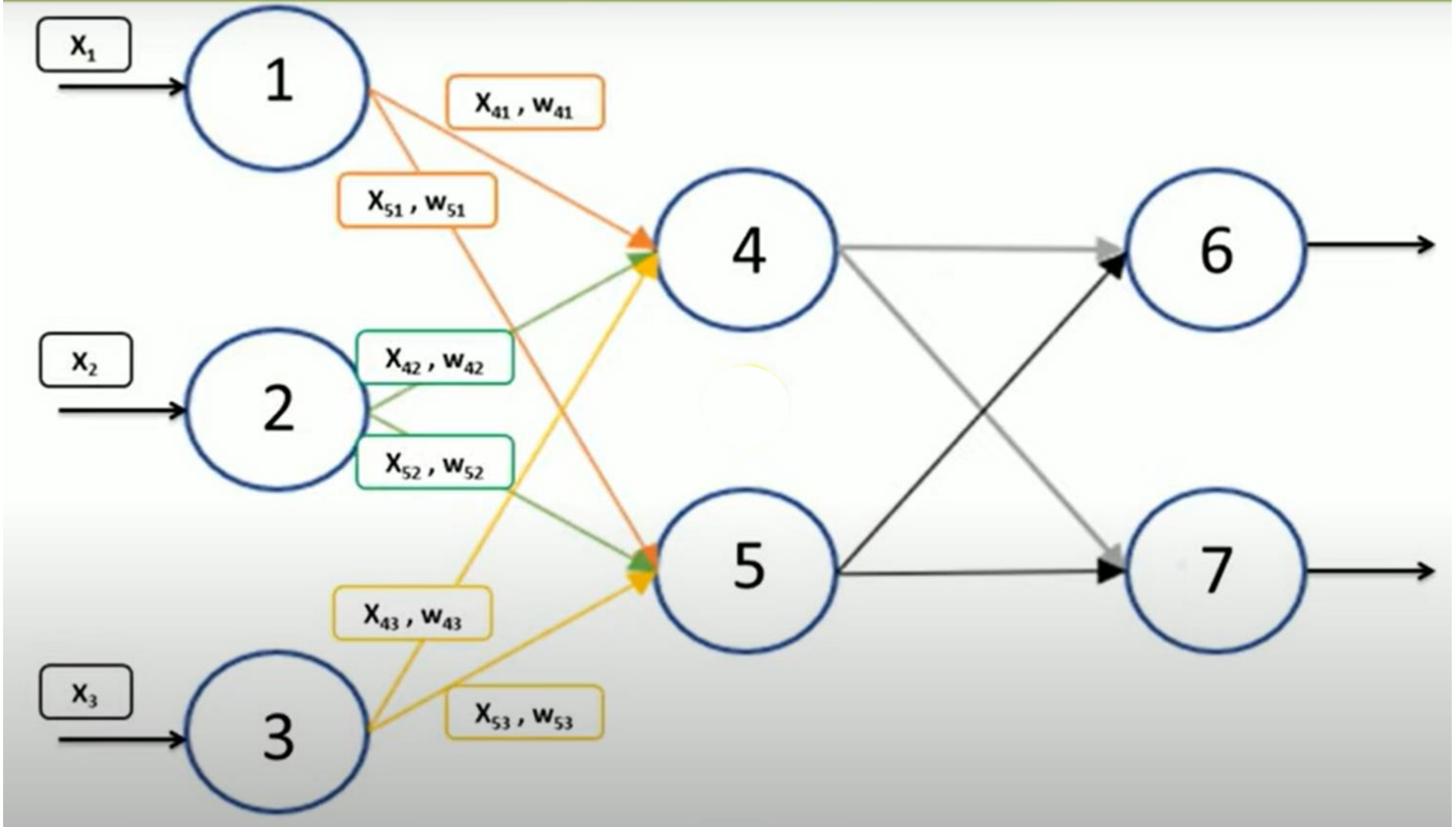
$$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$$

Where

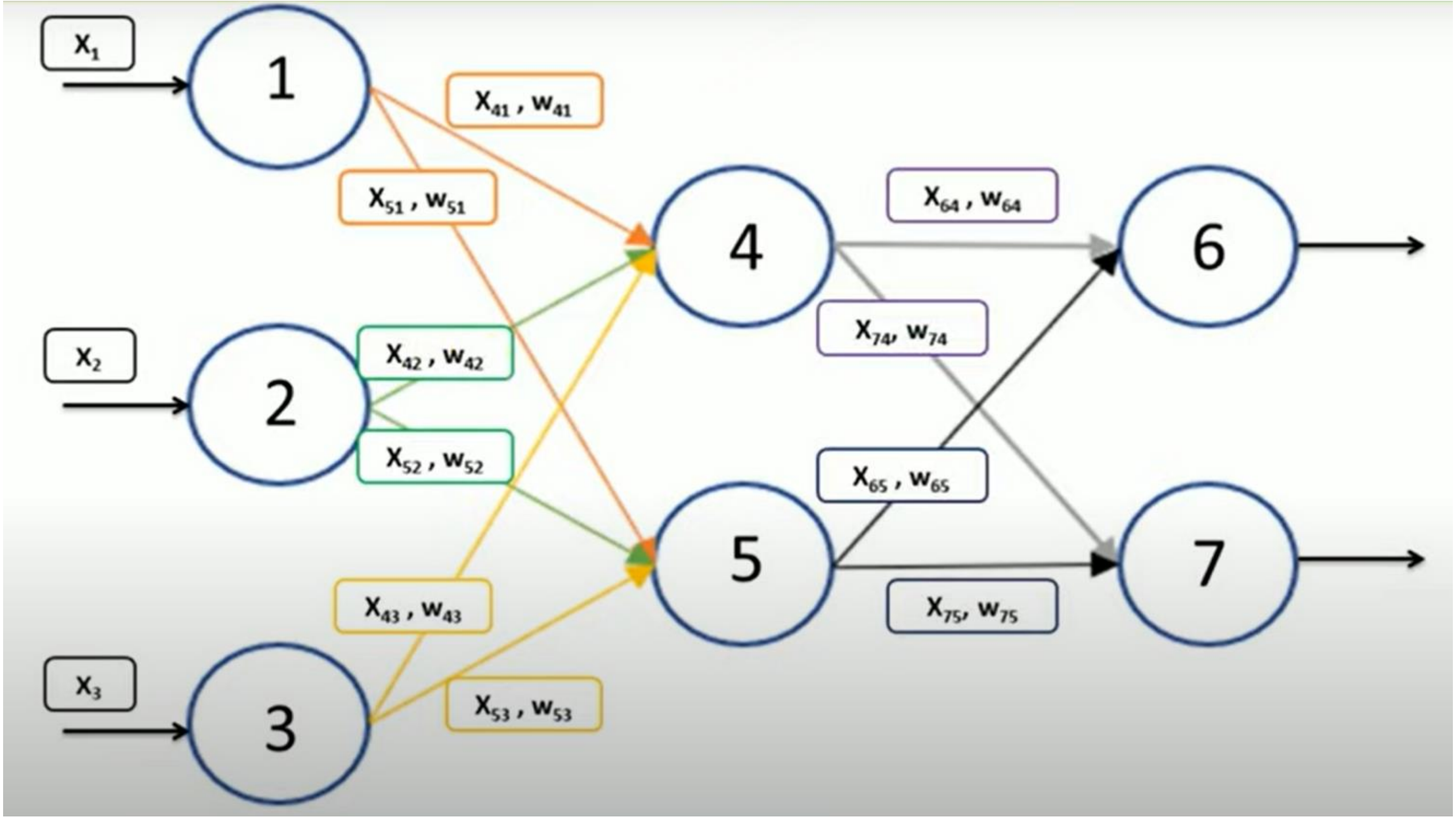
$$\Delta w_{ji} = \eta \delta_j x_{ji}$$



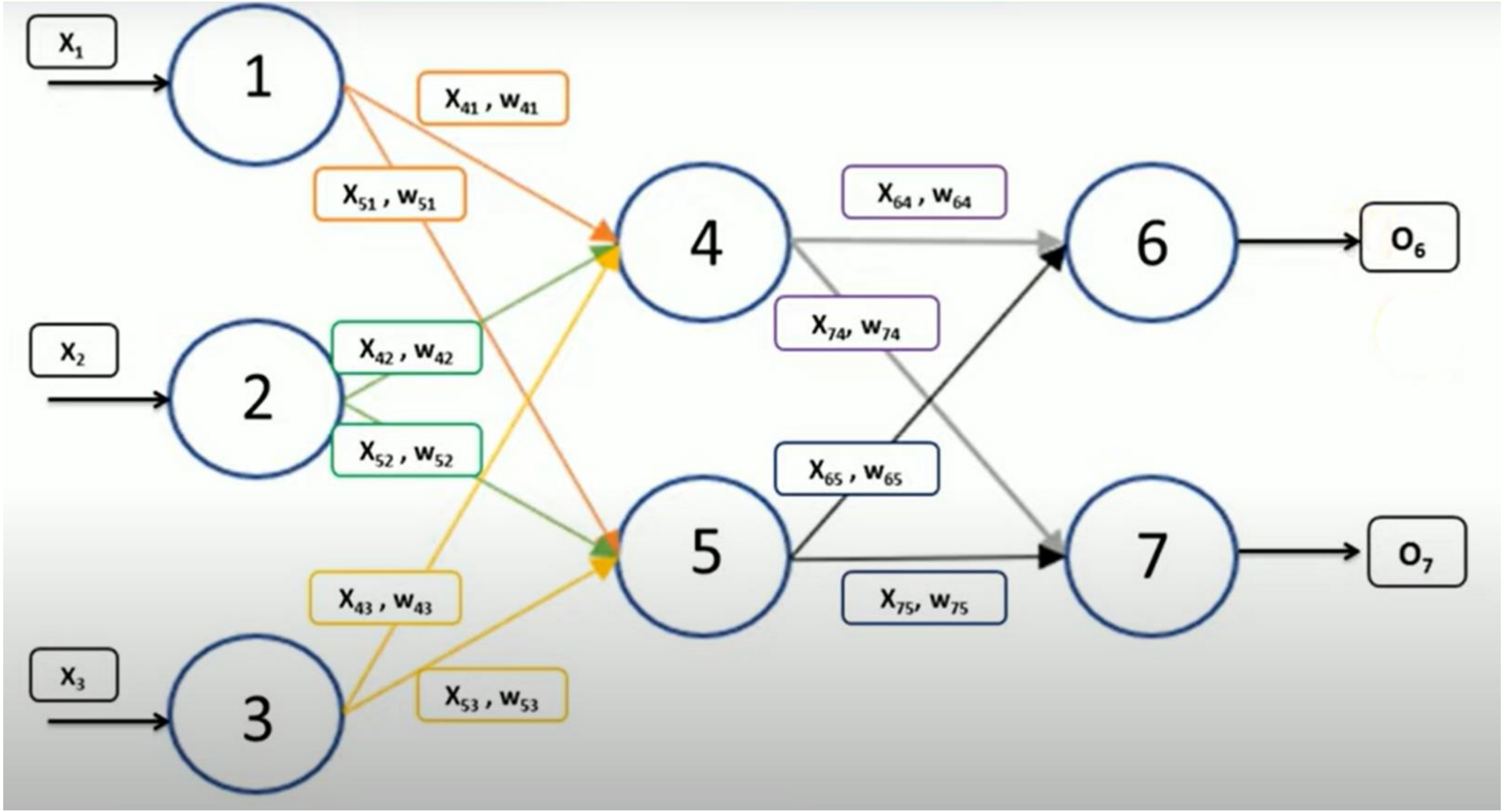
x_1
x_2
x_3
1
2
3
4
5
6
7



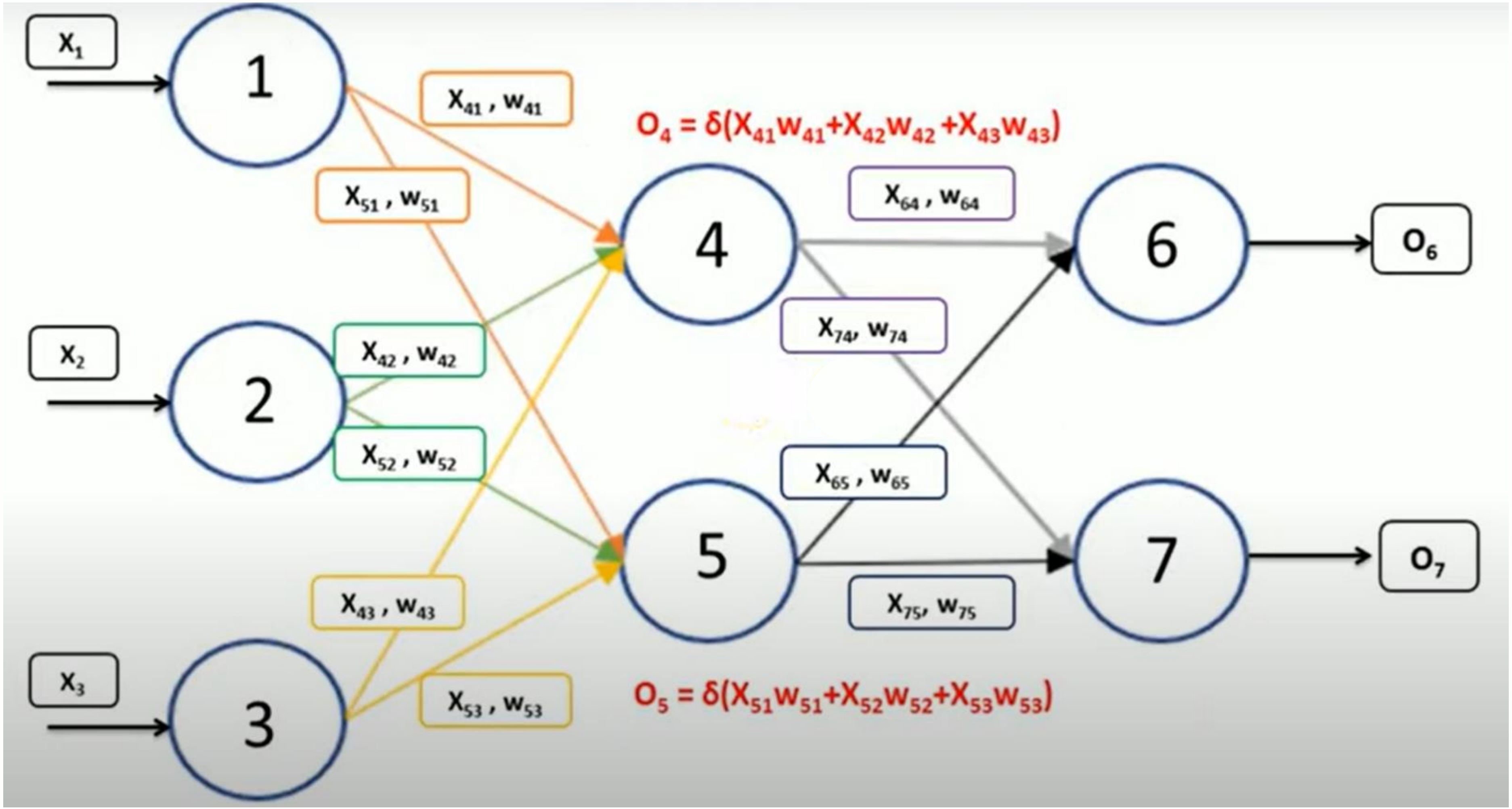
x1
x2
x3
1
2
3
4
5
6
7
X41 , w41
X51 , w51
X42 , w42
X52 , w52
X43 , w43
X53 , w53



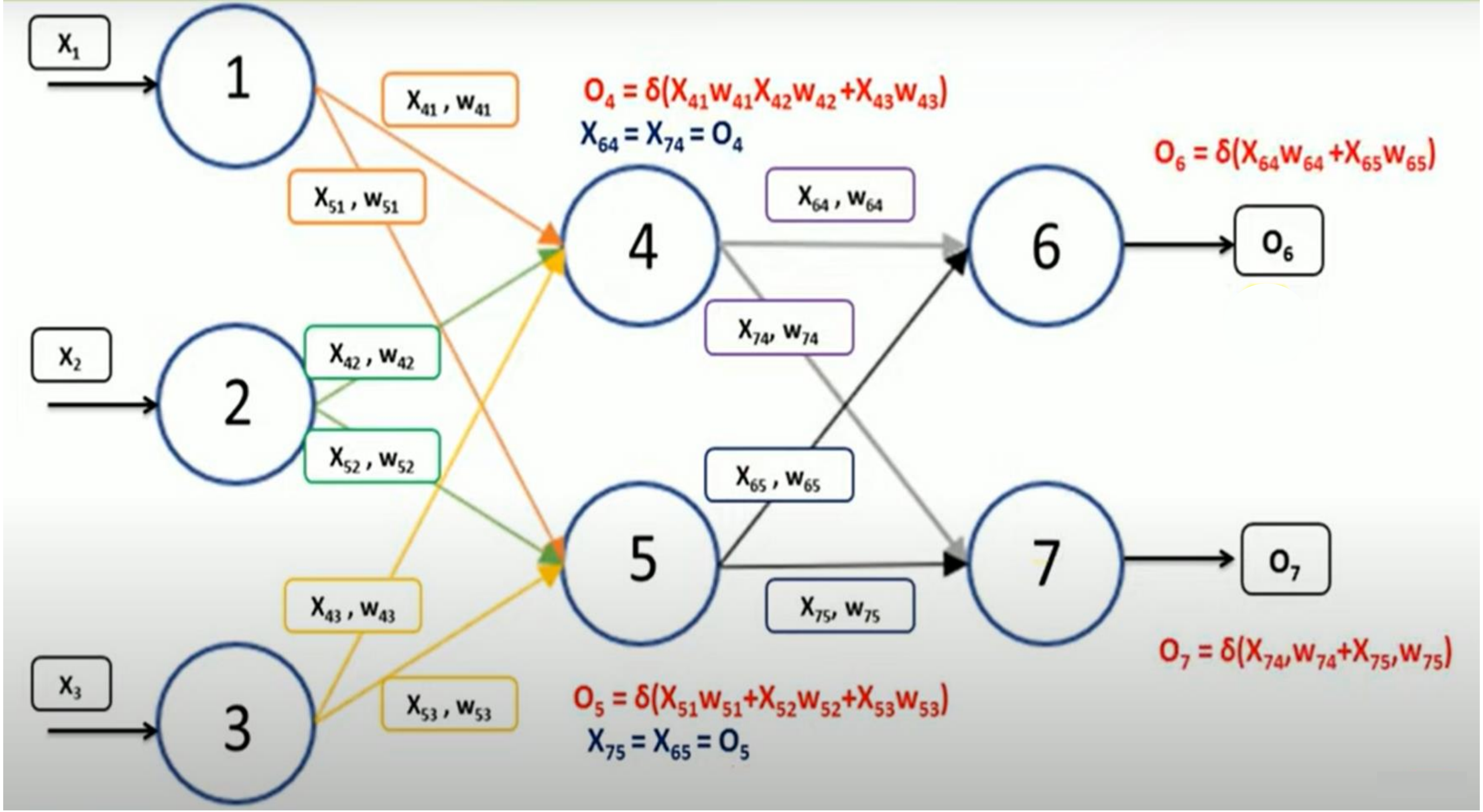
x_1
x_2
x_3
1
2
3
4
5
6
7
x_{41} , w_{41}
x_{51} , w_{51}
x_{42} , w_{42}
x_{52} , w_{52}
x_{43} , w_{43}
x_{53} , w_{53}
x_{64} , w_{64}
x_{74}, w_{74}
x_{65} , w_{65}
x_{75}, w_{75}



x_1
x_2
x_3
1
2
3
4
5
6
7
x_{41} , w_{41}
x_{51} , w_{51}
x_{42} , w_{42}
x_{52} , w_{52}
x_{43} , w_{43}
x_{53} , w_{53}
x_{64} , w_{64}
x_{74}, w_{74}
x_{65} , w_{65}
x_{75}, w_{75}
o_6
o_7

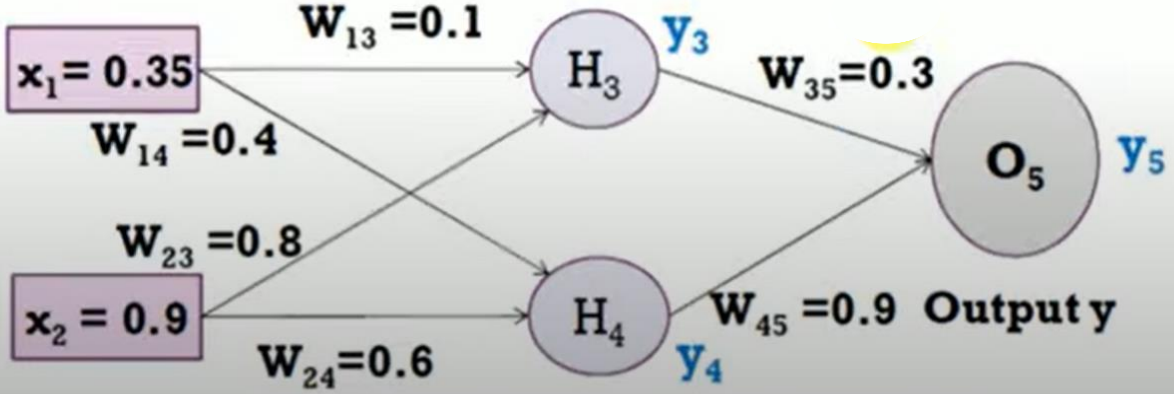


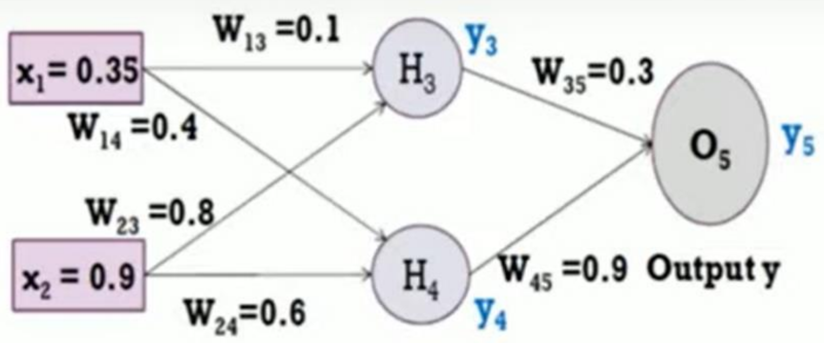
X1
X2
X3
1
2
3
4
5
6
7
X41 , w41
X51 , w51
X42 , w42
X52 , w52
X43 , w43
X53 , w53
X64 , w64
X74, w74
X65 , w65
X75, w75
O4 = δ(X41w41+X42w42+X43w43)
O5 = δ(X51w51+X52w52+X53w53)
O6
O7


X_1
X_2
X_3
1
2
3
4
5
6
7
X_{41}, w_{41}
X_{51}, w_{51}
X_{42}, w_{42}
X_{52}, w_{52}
X_{43}, w_{43}
X_{53}, w_{53}
$O_4 = \delta(X_{41}w_{41}X_{42}w_{42}+X_{43}w_{43})$
$X_{64} = X_{74} = O_4$
X_{64}, w_{64}
X_{74}, w_{74}
X_{65}, w_{65}
X_{75}, w_{75}
$O_5 = \delta(X_{51}w_{51}+X_{52}w_{52}+X_{53}w_{53})$
$X_{75} = X_{65} = O_5$
$O_6 = \delta(X_{64}w_{64}+X_{65}w_{65})$
$O_7 = \delta(X_{74},w_{74}+X_{75},w_{75})$
o_6
o_7

Problem

- 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.





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

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

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

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

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

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

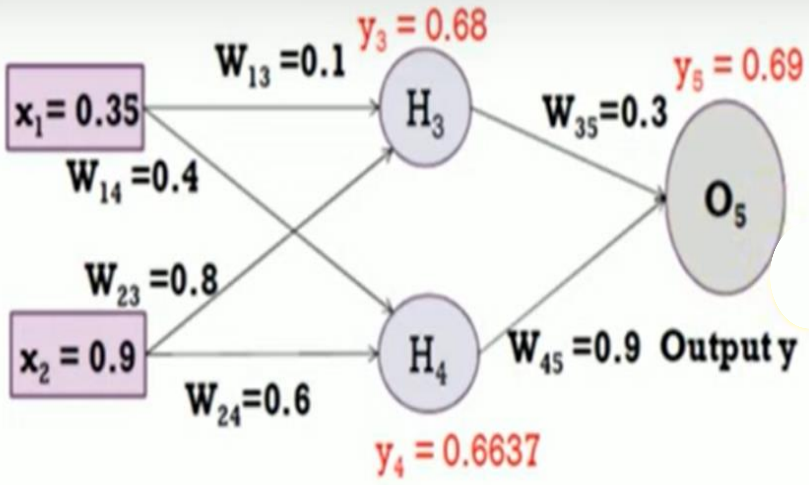
- Each weight changed by:

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

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

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

- 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



- Backward Pass: Compute δ3, δ4 and δ5.

For hidden unit:

$$\delta_3 = y_3(1-y_3)\, w_{35} * \delta_5$$
$$= 0.68*(1 - 0.68)*(0.3 * -0.0406) = -0.00265$$

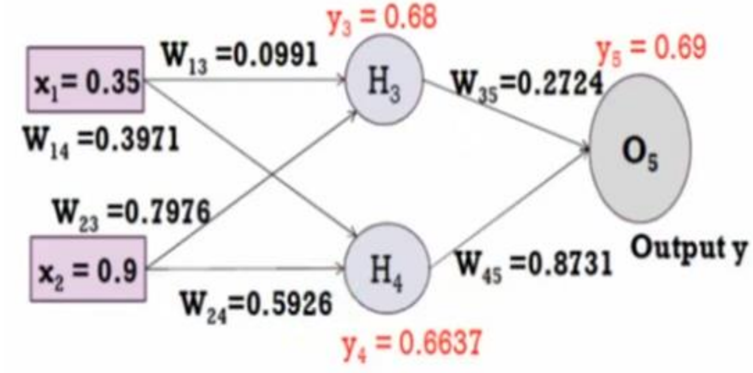
$$\delta_4 = y_4(1-y_4) w_{45} * \delta_5$$
$$= 0.6637*(1 - 0.6637)* (0.9 * -0.0406) = -0.0082$$

Compute new weights

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

Similarly update 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



- 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)**

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

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

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

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

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

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

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

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

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