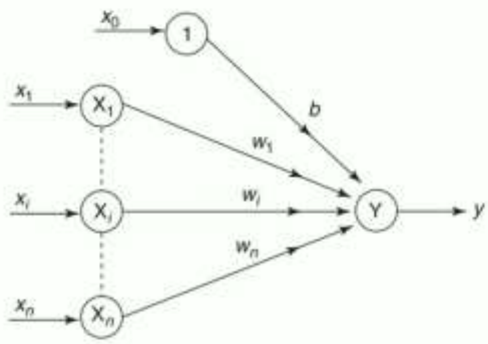
Perceptron Training Algorithm - Single Output Class

- Initialize the weights and the bias. Also initialize the learning rate ά (0< ά ≤ 1). T
- Until the final stopping condition is false.
  - for each training pair indicated by s:t.
    - Set each input unit i = 1 to n: $x_i = s_i$
    - Calculate the output of the network.

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i \qquad y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \le y_{in} \le \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$



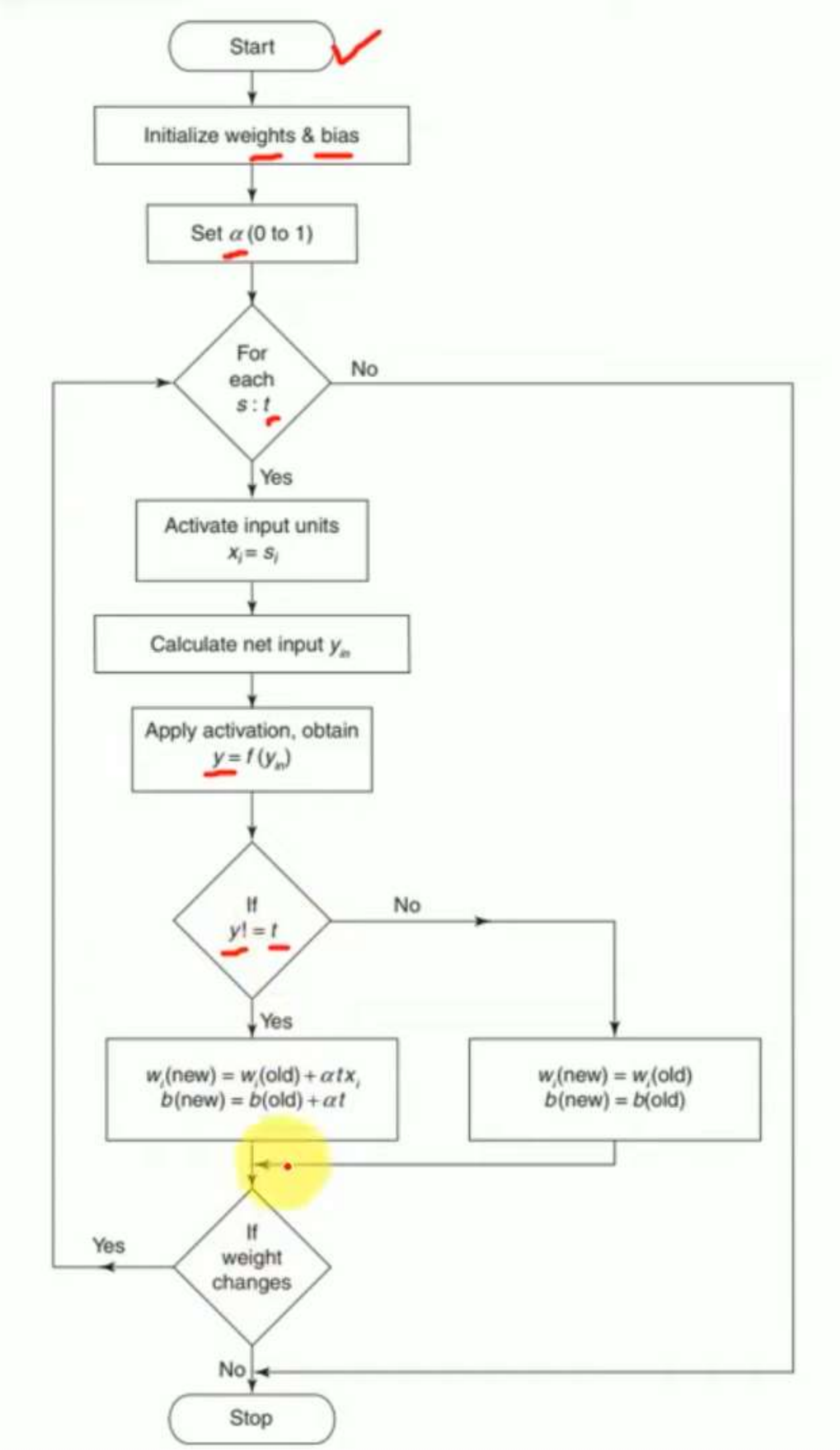
    - Weight and bias adjustment:

If $y \neq t$, then

$$w_i(\text{new}) = w_i(\text{old}) + \alpha t x_i$$
$$b(\text{new}) = b(\text{old}) + \alpha t$$

else we have

$$w_i(\text{new}) = w_i(\text{old})$$
$$b(\text{new}) = b(\text{old})$$


Start
Initialize weights & bias
Set α (0 to 1)
For each s : t
No
Yes
Activate input units
x_i = s_i
Calculate net input y_in
Apply activation, obtain
y = f (y_in)
If y! = t
No
Yes
w_i(new) = w_i(old) + α t x_i
b(new) = b(old) + α t
w_i(new) = w_i(old)
b(new) = b(old)
If weight changes
Yes
No
Stop

Perceptron Training Algorithm - Multiple Output Class

- Initialize the weights and the bias. Also initialize the learning rate ά (0< ά ≤ 1).
- Until the final stopping condition is false.
  - for each training pair indicated by s : t.
    - Set each input unit i = 1 to n: $x_i = s_i$
    - Calculate the output of the network.

$$y_{inj} = b_j + \sum_{i=1}^{n} x_i w_{ij} \qquad y_j = f(y_{inj}) = \begin{cases} 1 & \text{if } y_{inj} > \theta \\ 0 & \text{if } -\theta \le y_{inj} \le \theta \\ -1 & \text{if } y_{inj} < -\theta \end{cases}$$

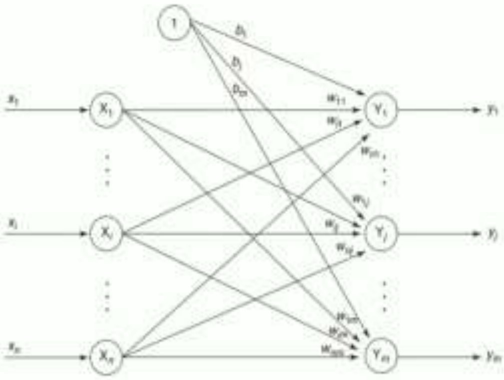
    - Weight and bias adjustment:

If $t_j \neq y_j$, then

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha t_j x_i$$
$$b_j(\text{new}) = b_j(\text{old}) + \alpha t_j$$

else, we have

$$w_{ij}(\text{new}) = w_{ij}(\text{old})$$
$$b_j(\text{new}) = b_j(\text{old})$$



Perceptron Network Testing Algorithm

- The initial weights to be used here are taken from the training algorithms (the final weights obtained during training).
- For each input vector X to be classified, perform the following
  - Calculate the net input of the unit.
  - Obtain the response of output unit.

$$y_{in} = \sum_{i=1}^{n} x_i w_i$$

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \le y_{in} \le \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

Perceptron Learning Rule

- In case of the perceptron learning rule, the learning signal is the difference between the calculated output and actual (target) output of a neuron.
- The output "y" is obtained on the basis of the net input calculated and activation function being applied over the net input.

$$y_{in} = b + \sum_{i=1}^{n} x_i w_i$$

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > \theta \\ 0 & \text{if } -\theta \le y_{in} \le \theta \\ -1 & \text{if } y_{in} < -\theta \end{cases}$$

If $y \neq t$, then

$$w(\text{new}) = w(\text{old}) + \alpha t x \quad (\alpha - \text{learning rate})$$

else, we have

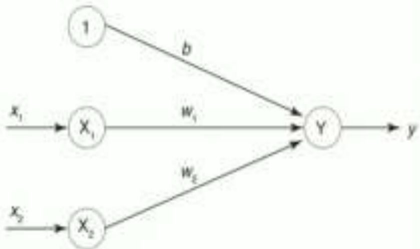
$$w(\text{new}) = w(\text{old})$$

- Weights are updated using the formula

AND function using Perceptron Rule Solved Example

- The perceptron network, which uses perceptron learning rule, is used to train the AND function.
- The input patterns are presented to the network one by one.
- When all the four input patterns are presented, then one epoch is said to be completed.
- The initial weights and threshold are set to zero.
- The learning rate a is set equal to 1.

x_1	x_2	t
1	1	1
1	−1	−1
−1	1	−1
−1	−1	−1



AND function using Perceptron Rule Solved Example

$$y_{in} = b + x_1 w_1 + x_2 w_2$$

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } y_{in} = 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

$\Delta w_1 = \alpha t x_1;$

$\Delta w_2 = \alpha t x_2;$

$\Delta b = \alpha t$

Input		Target	Net input	Calculated output	Weight changes			Weights		
x_1	x_2	(t)	(y_{in})	(y)	Δw_1	Δw_2	Δb	w_1 (0	w_2 0	b 0)
EPOCH-1										
1	1	1	0	0	1	1	1	1	1	1
1	−1	−1	1	1	−1	1	−1	0	2	0
−1	1	−1	2	1	+1	−1	−1	1	1	−1
−1	−1	−1	−3	−1	0	0	0	1	1	−1

AND function using Perceptron Rule Solved Example

$1 + 1 + (-1)$

$$y_{in} = b + x_1 w_1 + x_2 w_2$$

$0 + 1 \times 0 + 1 \times 0 = 0$

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } y_{in} = 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

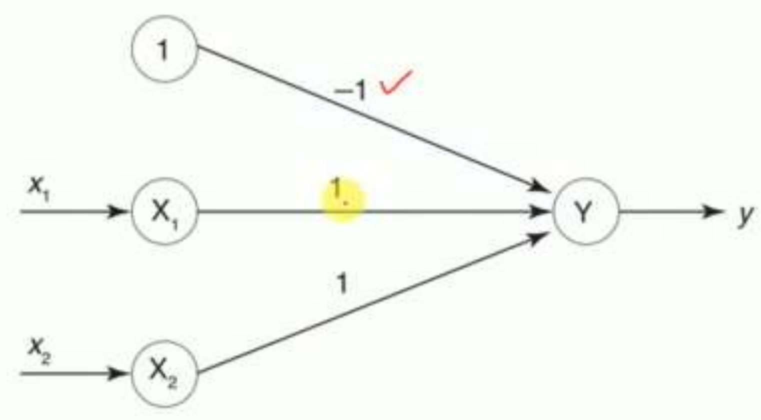
$\Delta w_1 = \alpha t x_1;$ $\alpha = 1$

$\Delta w_2 = \alpha t x_2;$

$\Delta b = \alpha t$

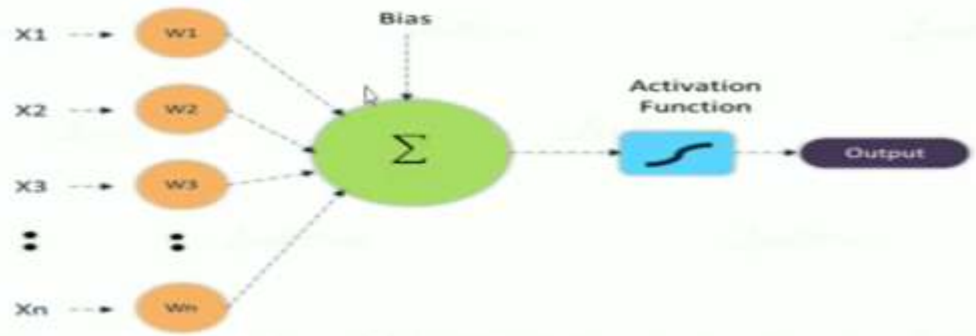
Input		Target	Net input	Calculated output	Weight changes			Weights		
x_1	x_2	(t)	(y_{in})	(y)	Δw_1	Δw_2	Δb	w_1 (0	w_2 0	b 0)
EPOCH-1										
1	1	1	0	0	1	1	1	1	1	1
1	−1	−1	1	1	−1	1	−1	0	2	0
−1	1	−1	2	1	+1	−1	−1	1	1	−1
−1	−1	−1	−3	−1	0	0	0	1	1	−1
EPOCH-2										
1	1	1	1	1	0	0	0	1	1	−1
1	−1	−1	−1	−1	0	0	0	1	1	−1
−1	1	−1	−1	−1	0	0	0	1	1	−1
−1	−1	−1	−3	−1	0	0	0	1	1	−1

AND function using Perceptron Rule Solved Example



WHAT IS BIAS..??

- **Bias** is one of the important terminologies.
- Often we add bias while creating any model in the artificial neural network.
- In a Neural network, **increase in weight** increases the **steepness of activation function.**
- Whereas **bias is used to delay the triggering of the activation function.**



Schematic Representation of a Neuron in a Neural Network

Perceptron Network (Rule) Solved Example

- Find the weights required to perform the following classification using perceptron network.
- The vectors (1, 1, 1, 1) and (- 1, 1 - 1, - 1) are belonging to the class 1, vectors (1, 1, 1, - 1) and (1, - 1, - 1, 1) are belonging to the class -1.
- Assume learning rate as 1
- and Initial weights as 0.

Input					Target
x_1	x_2	x_3	x_4	b	(t)
1	1	1	1	1	1
−1	1	−1	−1	1	1
1	1	1	−1	1	−1
1	−1	−1	1	1	−1

Perceptron Network (Rule) Solved Example

$$y_{in} = b + x_1 w_1 + x_2 w_2 + x_3 w_3 + x_4 w_4$$

$$y = f(y_{in}) = \begin{cases} 1 & \text{if } y_{in} > 0 \\ 0 & \text{if } y_{in} = 0 \\ -1 & \text{if } y_{in} < 0 \end{cases}$$

$$\Delta w_1 = \alpha t x_1;$$
$$\Delta w_2 = \alpha t x_2;$$
$$\Delta w_3 = \alpha t x_3;$$
$$\Delta w_4 = \alpha t x_4;$$
$$\Delta b = \alpha t$$

Inputs				Target	Net input	output	Weight changes					Weights				
$(x_1$	x_2	x_3	x_4	(t)	(y_{in})	(y)	$(\Delta w_1$	Δw_2	Δw_3	Δw_4	$\Delta b)$	w_1 (0	w_2 0	w_3 0	w_4 0	b 0)
EPOCH-1																
(1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1
(-1	1	-1	-1	1	-1	-1	-1	1	-1	-1	1	0	2	0	0	2
(1	1	1	-1	-1	4	1	-1	-1	-1	1	-1	-1	1	-1	1	1
(1	-1	-1	1	-1	1	1	-1	1	1	-1	-1	-2	2	0	0	0
EPOCH-2																
(1	1	1	1	1	0	0	1	1	1	1	1	-1	3	1	1	1
(-1	1	-1	-1	1	3	1	0	0	0	0	0	-1	3	1	1	1
(1	1	1	-1	-1	4	1	-1	-1	-1	1	-1	-2	2	0	2	0
(1	-1	-1	1	-1	-2	-1	0	0	0	0	0	-2	2	0	2	0
EPOCH-3																
(1	1	1	1	1	2	1	0	0	0	0	0	-2	2	0	2	0
(-1	1	-1	-1	1	2	1	0	0	0	0	0	-2	2	0	2	0
(1	1	1	-1	-1	-2	-1	0	0	0	0	0	-2	2	0	2	0
(1	-1	-1	1	-1	-2	-1	0	0	0	0	0	-2	2	0	2	0

```python
import numpy as np
x1=np.array([0,0,1,1])
x2=np.array([0,1,0,1])
y=np.array([0,1,1,1])
epochs=int(input('Enter the epochs:'))
bias=1
l=0.4
a=[]
a.append(float(input('Enter the Weights for bias :')))
a.append(float(input('Enter the weights for x1 :')))
a.append(float(input('Enter the weights for x2 :')))
w=np.array(a)
n=x1.shape[0]
for k in range(epochs):
    for i in range(n):
        f=x1[i]*w[1]+x2[i]*w[2]+bias*w[0]
        y_out=(f>0).astype('int')
        error=y[i]-y_out
        if error!=0:
            w[1]=w[1]+l*error*x1[i]
            w[2]=w[2]+l*error*x2[i]
            w[0]=w[0]+l*error*bias
        print('-----------------------------------------')
        print('Updated Weights after',i+1,'input instance')
        print('x1 weight ',w[1])
        print('x2 weight ',w[2])
        print('bias weight',w[0])
print('-----------------------------------------------------')
print('Final Weights after',epochs,'epoch(s)')
print('Updated weight for x1 :',w[1])
print('Updated weight for x2 :',w[2])
print('Updated weight for bias :',w[0])
```

Step 2 : Initialize the network parameters

(i) Epochs (training iterations)

(ii) bias input (bias = 1)

(iii) learning rate (0 to 1)

(iv) Input weights & bias weights (w_1, w_2, w_0)

Step 3 : Start the training process

For Each iteration (Epochs)

For Each input instances

3.1 Compute the Summation

$f(x) = x_1 * w_1 + x_2 * w_2 + bias * w_0$

3.2 Apply activation function

(unit step, sigmoid)

$$y_{out} = \begin{cases} 1 & f(x) > 0 \\ 0 & f(x) <= 0 \end{cases}$$

3.3 Update input weights & bias weights

$Error = y - y_{out}$

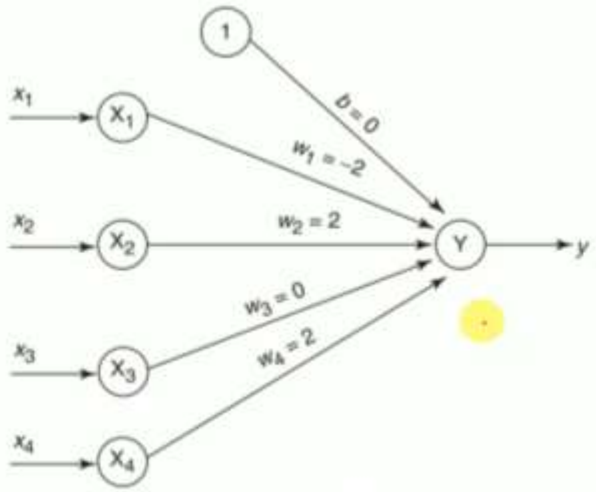
$w_1 = w_1 + (l) * Error * x_1$

$\uparrow$ learning rate $\qquad$ $\uparrow$ input

$w_2 = w_2 + (l) * error * x_2$

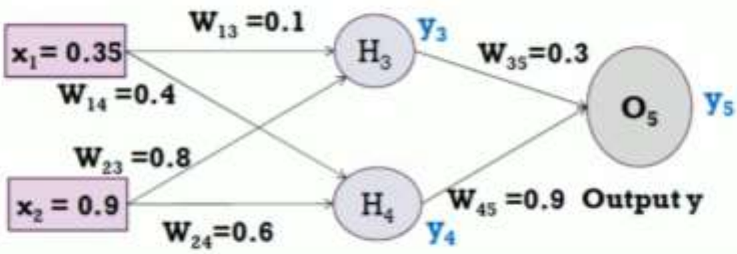
$w_0 = w_0 + (l) * error * bias$

Perceptron Network (Rule) Solved Example

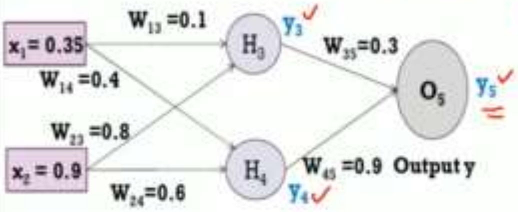


Back Propagation Solved Example - 1

- 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



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

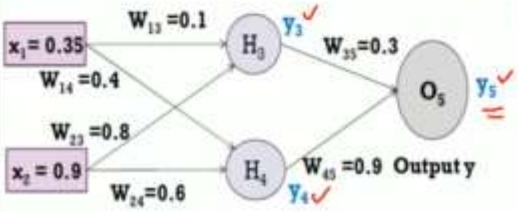
$$a_j = \sum_j (w_{i,j} * x_i) \qquad yj = F(aj) = \frac{1}{1+e^{-a_j}}$$

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

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

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

Back Propagation Solved Example - 1



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

$0.5 - 0.69$

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

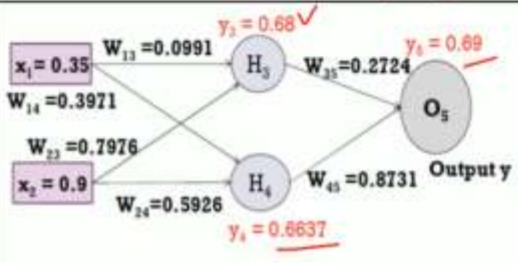
$$a_j = \sum_j (w_{i,j} * x_i) \qquad yj = F(aj) = \frac{1}{1+e^{-a_j}}$$

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

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

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

Back Propagation Solved Example - 1



- 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 Solved Example - 1

- Each weight changed by:

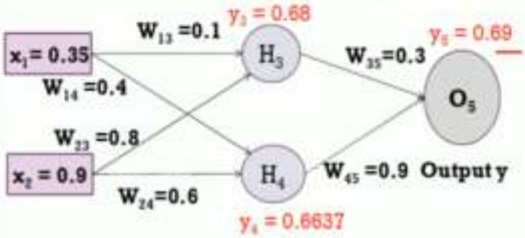
$$\Delta w_{ji} = \eta \delta_j o_i$$

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

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

- 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:

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

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

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

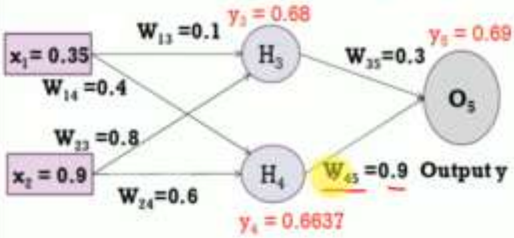
$$\delta_j = o_j(1-o_j)(t_j - o_j) \quad \text{if } j \text{ is an output unit}$$

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

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

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

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:

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

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

Compute new weights

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

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

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

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

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

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

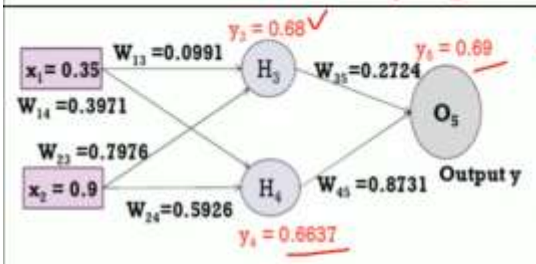
$w_{14}\text{(new)} = \Delta w_{14} + w_{14}\text{(old)} = -0.00287 + 0.4 = 0.3971$

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.

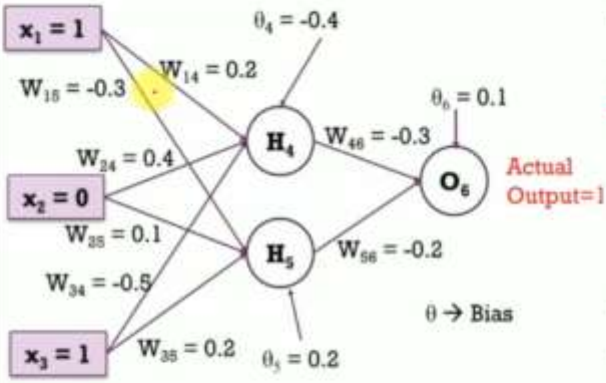
$$a_j = \sum_j (w_{i,j} * x_i) \qquad yj = F(aj) = \frac{1}{1 + e^{-a_j}}$$

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

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

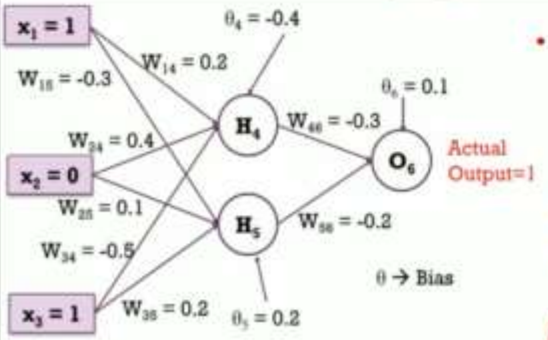
$a_3 = (w_{35} * y_3) + (w_{45} * y_4)$
$= (0.2724 * 0.6797) + (0.8731 * 0.6620) = 0.7631$
$y_5 = f(a_3) = 1/\,(1 + e^{-0.7631}) =$ **0.6820** (Network Output)

Back Propagation Solved Example - 2



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 Solved Example - 2



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

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

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

$a_4 = (w_{14} * x_1) + (w_{24} * x_2) + (w_{34} * x_3) + \theta_4$
$= (0.2 * 1) + (0.4 * 0) + (-0.5 * 1) + (-0.4) = -0.7$
$O(H_4) = y_4 = f(a_4) = 1/(1 + e^{0.7}) = 0.332$

$a_5 = (w_{15} * x_1) + (w_{25} * x_2) + (w_{35} * x_3) + \theta_5$
$= (-0.3 * 1) + (0.1 * 0) + (0.2 * 1) + (0.2) = 0.1$
$O(H_5) = y_5 = f(a_5) = 1/(1 + e^{-0.1}) = 0.525$

$a_6 = (w_{46} * H_4) + (w_{56} * H_5) + \theta_6$
$= (-0.3 * 0.332) + (-0.2 * 0.525) + 0.1 = -0.105$
$O(O_6) = y_6 = f(a_6) = 1/(1 + e^{0.105}) = \mathbf{0.474}$

Back Propagation Solved Example - 2

- Each weight changed by:

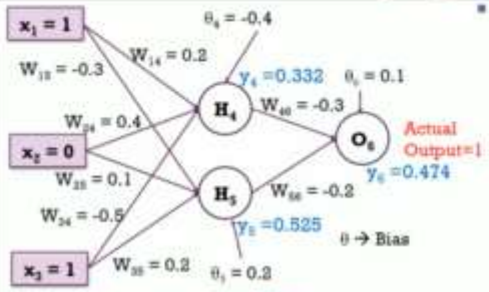
$$\Delta w_{ji} = \eta \delta_j o_i$$

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

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

- 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 - 2



- Backward Pass: Compute δ4, δ5 and δ6.

For output unit:

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

For hidden unit:

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

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

Compute new weights

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

$\Delta w_{46} = \eta\ \delta_6\ y_4 = 0.9 * 0.1311 * 0.332 = 0.03917$ ✓
$w_{46}\ (\text{new}) = \Delta w_{46} + w_{46}(\text{old}) = 0.03917 + (-0.3) = -0.261$

$\Delta w_{14} = \eta\ \delta_4\ x_1 = 0.9 * -0.0087 * 1 = -0.0078$
$w_{14}\ (\text{new}) = \Delta w_{14} + w_{14}(\text{old}) = -0.0078 + 0.2 = 0.192$

Back Propagation Solved Example - 2

- 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

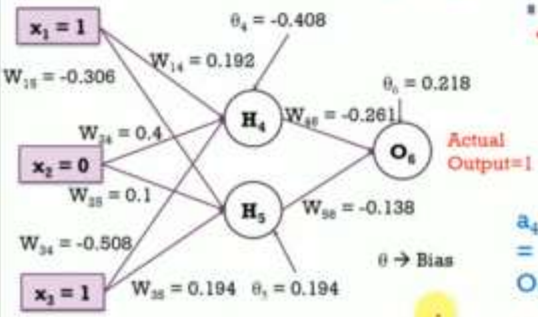
Back Propagation Solved Example - 2

- 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

1

Back Propagation Solved Example - 2



Error $= y_{target} - y_6 = 0.485$

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

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

$a_4 = (w_{14} * x_1) + (w_{24} * x_2) + (w_{34} * x_3) + \theta_4$
$= (0.192 * 1) + (0.4 * 0) + (-0.508 * 1) + (-0.408) = -0.724$
$O(H_4) = y_4 = f(a_4) = 1/(1 + e^{0.724}) = 0.327$

$a_5 = (w_{15} * x_1) + (w_{25} * x_2) + (w_{35} * x_3) + \theta_5$
$= (-0.306 * 1) + (0.1 * 0) + (0.194 * 1) + (0.194) = 0.082$
$O(H_5) = y_5 = f(a_5) = 1/(1 + e^{-0.082}) = 0.520$

$a_6 = (w_{46} * H_4) + (w_{56} * H_5) + \theta_6$
$= (-0.261 * 0.327) + (-0.138 * 0.520) + 0.218 = 0.061$
$O(O_6) = y_6 = f(a_6) = 1/(1 + e^{-0.061}) = \mathbf{0.515}$ (Network Output)