Lecture 16:
Artificial Neural Networks (ANNs)
Back Propagation

Sabah Sayed

Department of Computer Science
Faculty of Computers and Artificial Intelligence
Cairo University
Egypt

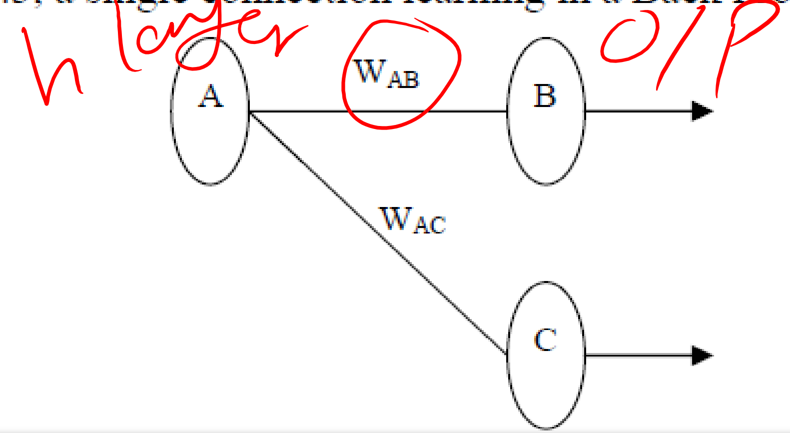
Back Propagation

- 1986: Most important multi-layer ANN learning algorithm (ANN weight update)
- The global error is backward propagated to network nodes, weights are modified proportional to their contribution.

Back Propagation Learning Algorithm for a single connection

- Initially we will look at one connection W_{AB}, between a neuron in the output layer and one in the hidden layer

Figure 3.3, a single connection learning in a Back Propagation network.



Back Propagation Learning Algorithm for a single connection

- **Step 1:** First apply the inputs to the network and work out the output.
- **Step 2:** Compute Mean Square Error :

$$E_p = \frac{1}{2}\sum_{k=1}^{n}(Target_k - Output_k)^2$$

$\boxed{Sig}$ $\frac{1}{1+e^{-x}}$

If Ep<= acceptable value **then** stop
Else go to step 3

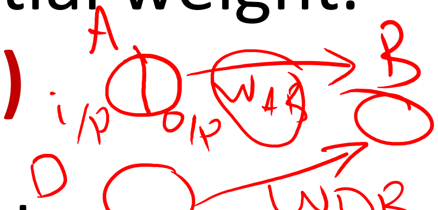
- **Step 3:** Next work out the error for neuron B. The error is *What you want – What you actually get:*
  Error$_B$ = Output$_B$ (1-Output$_B$)(Target$_B$ – Output$_B$)

  Output$_B$ (1-Output$_B$) is the derivative of the sigmoid function
- **Similarly , calculate error for all output neurons (1→n)**

Back Propagation Learning Algorithm for a single connection

- <u>**Step 4:**</u> Change the weight. Let W^{+}_{AB} be the new (trained) weight and W_{AB} be the initial weight.

$$\mathbf{W^{+}_{AB} = W_{AB} + (Error_{B} \times Output_{A})}$$



→ Note that weights associated with larger output values (from hidden layer, i.e. Neuron A) will receive bigger changes than those associated with lower output values .

→ We update all the weights in the output layer this way.

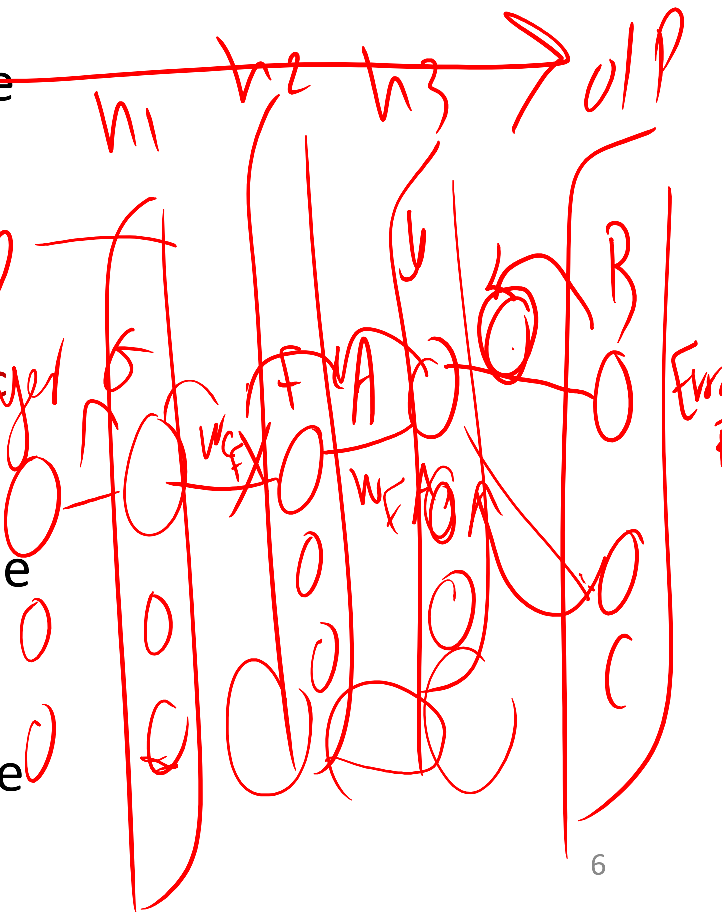
Back Propagation Learning Algorithm for a single connection

- <u>**Step 5:**</u> Calculate the Errors for the hidden layer neurons.
- Unlike the output layer we can't calculate these directly (because we don't have a Target).
- So we ***Back Propagate*** them from the output layer (hence the name of the algorithm).

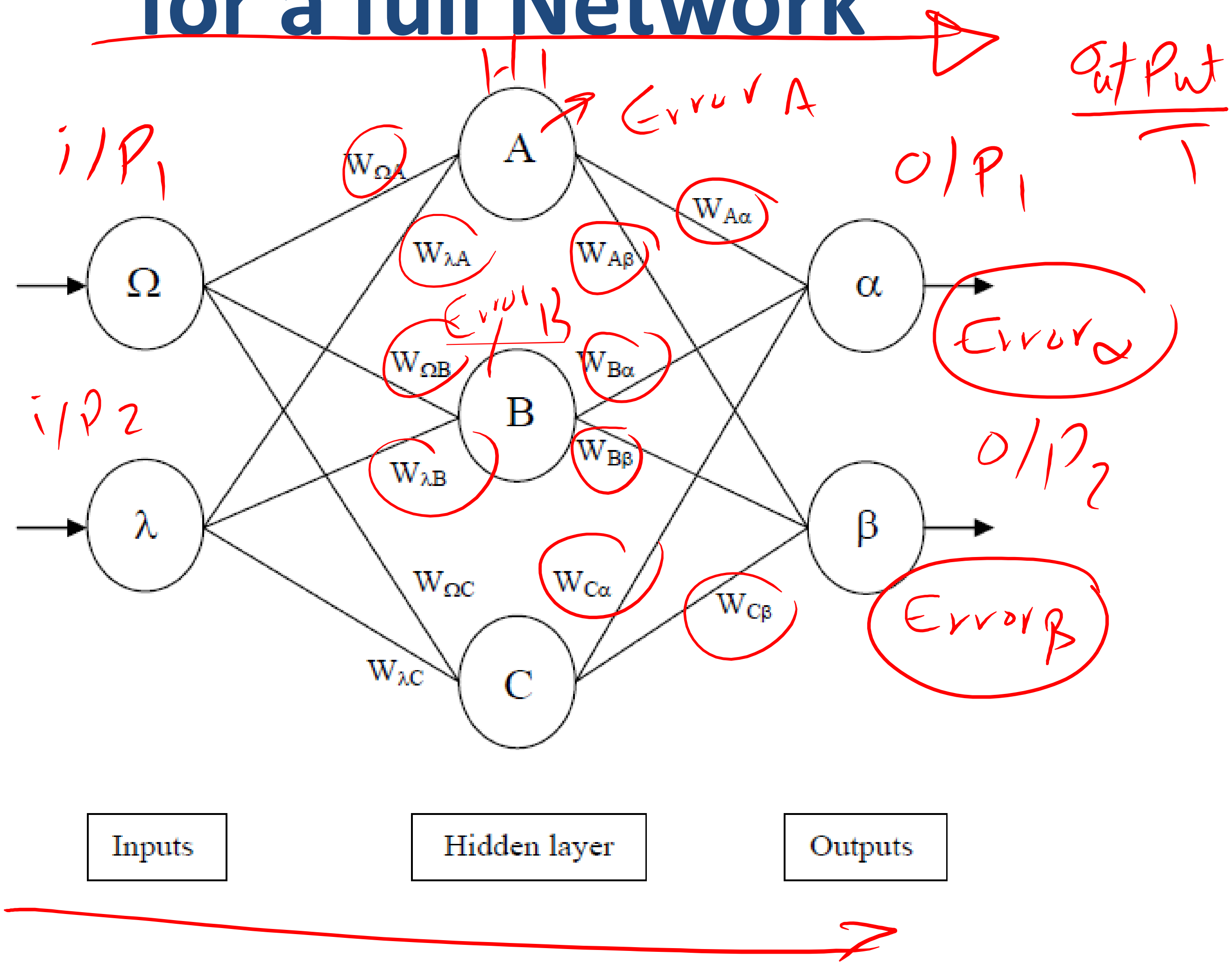
$Error_A = Output_A\ (1 - Output_A)(Error_B\ W_{AB} + Error_C\ W_{AC})$

$$\delta_A = out_A(1 - out_A)(\delta_B W_{AB} + \delta_C W_{AC})$$

- We calculate all hidden neurons errors the same way (1→l)
- Having obtained the Error for the hidden layer neurons now proceed as in step 4 to change the hidden layer weights.



Back Propagation Learning Algorithm for a full Network



Back Propagation Learning Algorithm for a full network

1. Calculate errors of output neurons

$$\delta_\alpha = out_\alpha\,(1 - out_\alpha)\,(Target_\alpha - out_\alpha)$$
$$\delta_\beta = out_\beta\,(1 - out_\beta)\,(Target_\beta - out_\beta)$$

2. Change output layer weights

$$W^+_{A\alpha} = W_{A\alpha} + \eta\delta_\alpha\, out_A \qquad W^+_{A\beta} = W_{A\beta} + \eta\delta_\beta\, out_A$$
$$W^+_{B\alpha} = W_{B\alpha} + \eta\delta_\alpha\, out_B \qquad W^+_{B\beta} = W_{B\beta} + \eta\delta_\beta\, out_B$$
$$W^+_{C\alpha} = W_{C\alpha} + \eta\delta_\alpha\, out_C \qquad W^+_{C\beta} = W_{C\beta} + \eta\delta_\beta\, out_C$$

3. Calculate (back-propagate) hidden layer errors

$$\delta_A = out_A\,(1 - out_A)\,(\delta_\alpha W_{A\alpha} + \delta_\beta W_{A\beta})$$
$$\delta_B = out_B\,(1 - out_B)\,(\delta_\alpha W_{B\alpha} + \delta_\beta W_{B\beta})$$
$$\delta_C = out_C\,(1 - out_C)\,(\delta_\alpha W_{C\alpha} + \delta_\beta W_{C\beta})$$

4. Change hidden layer weights

$$W^+_{\lambda A} = W_{\lambda A} + \eta\delta_A\, in_\lambda \qquad W^+_{\Omega A} = W^+_{\Omega A} + \eta\delta_A\, in_\Omega$$
$$W^+_{\lambda B} = W_{\lambda B} + \eta\delta_B\, in_\lambda \qquad W^+_{\Omega B} = W^+_{\Omega B} + \eta\delta_B\, in_\Omega$$
$$W^+_{\lambda C} = W_{\lambda C} + \eta\delta_C\, in_\lambda \qquad W^+_{\Omega C} = W^+_{\Omega C} + \eta\delta_C\, in_\Omega$$

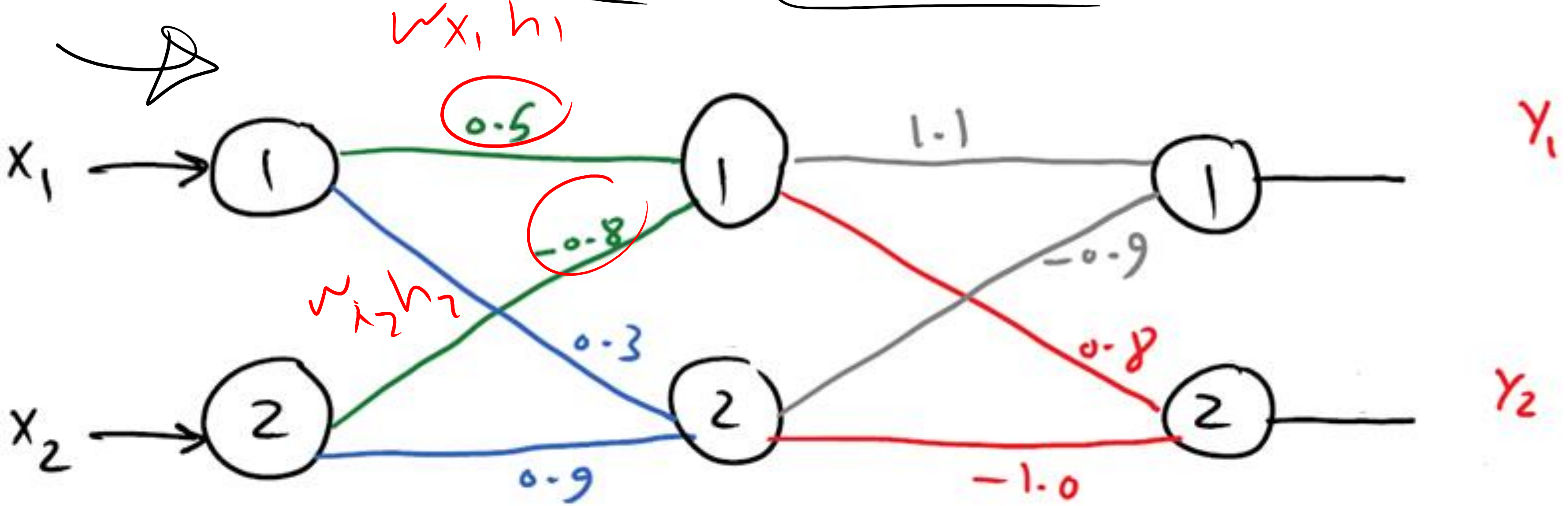
The constant η (called the learning rate, and nominally equal to one) is put in to speed up or slow down the learning if required.

Back Propagation Learning Algorithm for a full network- Example

Assume that the neurons have a Sigmoid activation function and $\eta = 0.5$
Where the dataset contains only 1 record :

X1	X2	Y1	Y2
1	3	0.9	0.1

(i) Perform a forward pass on the network.
(ii) Perform a reverse pass (training) once.
(iii) Perform a further forward pass and comment on the result



$$\boxed{\text{Step 1}}$$

$h_1\,in = 1 \times 0.5 + \boxed{3} \times (-0.8) = 1.9$ Feed forward pass

$$h_1\,out = \frac{1}{1+e^{-1.9}} = 0.13$$

$h_2\,in = 1 \times 0.3 + 3 \times 0.9 = 3.0$

$$h_2\,out = \frac{1}{1+e^{-3.0}} = 0.95$$

$Y_1\,in = 0.13 \times 1.1 + 0.95 \times (-0.9) = -0.712$

$$Y_1\,out = \frac{1}{1+e^{-(-0.712)}} = \boxed{0.329}$$

$Y_2\,in = 0.13 \times 0.8 + 0.95 * (-1.0) = -0.846$

$$Y_2\,out = \frac{1}{1+e^{-(-0.846)}} = \boxed{0.3}$$

Step 7 (ii)

$$Error_{y_1} = out_{y_1}(1-out_{y_1})(T_{y_1}-out_{y_1})$$

$$= 0.329(1-0.329)(0.9-0.329) = 0.126$$

$$Error_{y_2} = 0.3(1-0.3)(0.1-0.3) = -0.042$$

update weights

$$W_{h_1y_1}^{(t+1)} = W_{h_1y_1}^{(t)} + \eta * Error_{y_1} * h_{1out}$$

$$= 1.1 + [0.5 * 0.126 * 0.13] = 1.10819$$

$$W_{h_2y_1}(t+1) = W_{h_2y_1}(t) + \eta * Error_{y_1} * h_{2out}$$

$$= -0.9 + [0.5 * 0.126 * 0.95] = -0.84$$

$$W_{h_1y_2}(t+1) = 0.8 + [0.5 * (-0.042) * 0.13] = 0.797$$

$$W_{h_2y_2}(t+1) = (-1.0) + [0.5 * (-0.042) * 0.95] = -1.019$$

$$\text{Error}_m = 0.13(1-0.13) \times \left[\text{Error}_{y_1} \times \overset{\text{old}}{W_{hy_1}} + \text{Error}_{y_2} \times \overset{\text{old}}{W_{hy_2}}\right]$$

$$= 0.13(1-0.13) \times \left[0.126 \times 1.1 + (-0.042) \times 0.8\right]$$

$$= 0.01188$$

$$\text{Error}_{h_2} = 0.95(1-0.95) \times \left[0.126 \times (-0.9) + (-0.042) \times (-1.0)\right] = -0.00339$$

$$W_{x_1h_1}(t+1) = W_{x_1h_1}(t) + \eta\left[\text{Error}_{h_1} \times X_1\right]$$

$$= 0.5 + \left[0.5 \times 0.01188 \times 1\right]$$

$$= 0.5059$$

$W_{x_2h_1} = -0.8 + [0.5 \times (0.61189) \times 3] \simeq -0.78$

$W_{x_1h_2} = 0.3 + [0.5 \times (-0.00339) \times 1] = 0.298$

$W_{x_2h_2} = 0.~~9~~ + [0.5 \times (-0.00339) \times 3] = 0.8949$

stop back Propagation

(iii)

~~next feed forward pass~~

$h_{1in} = 1 \times 0.5059 + 3 \times (-0.78) = -1.83$

$h_{1out} = 0.138$

$h_{2in} = 1 \times 0.298 + 3 \times 0.8949 = 2.98$

$h_{2out} = \frac{1}{1 + e^{-h_{2in}}} = 0.951$

$y_{1in} = 0.138 \times 1.10819 + 0.951 \times (-0.84) = -0.645$

$y_{1out} = \frac{1}{1 + e^{-y_{1in}}} = 0.344$ $(t = 1)$

$T_{y_1} = 0.9$

$\left(\frac{\text{out } y}{(t = 2)}\right) = 0.32 \; (t = 0)$

$y_{2in} = 0.138 \times 0.797 + 0.95 \times (-1.019)$

$= -0.859$

$$\boxed{MSE = \frac{1}{2} \sum_{k=1}^{n} (T_k - out_k)^2}$$

$y_{2out} = \dfrac{1}{1+e^{-(-0.859)}} \rightarrow 0.297$

T_{y_2} $Out_{y_2}(t=0)$ $out_{y_2}(t=1)$

$\boxed{0.05}$

0.1 $0.1 - 0.3$ 0.3 0.297

0

T_{y_1} $out_{y_1}(t=0)$ $out_{y_1}(t=1)$

0.9 0.32 0.344

Derivation of Sigmoid function

Let's denote the sigmoid function as $\sigma(x) = \frac{1}{1+e^{-x}}$.

The derivative of the sigmoid is $\frac{d}{dx}\sigma(x) = \sigma(x)(1-\sigma(x))$.

Here's a detailed derivation:

$$
\begin{aligned}
\frac{d}{dx}\sigma(x) &= \frac{d}{dx}\left[\frac{1}{1+e^{-x}}\right] \\
&= \frac{d}{dx}\left(1+\mathrm{e}^{-x}\right)^{-1} \\
&= -(1+e^{-x})^{-2}(-e^{-x}) \\
&= \frac{1 * e^{-x}}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{e^{-x}}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \frac{(1+e^{-x})-1}{1+e^{-x}} \\
&= \frac{1}{1+e^{-x}} \cdot \left(\frac{1+e^{-x}}{1+e^{-x}} - \frac{1}{1+e^{-x}}\right) \\
&= \frac{1}{1+e^{-x}} \cdot \left(1 - \frac{1}{1+e^{-x}}\right) \\
&= \sigma(x) \cdot (1-\sigma(x))
\end{aligned}
$$