# Lecture 16:
# Artificial Neural Networks (ANNs)
# Back Propagation

Sabah Sayed

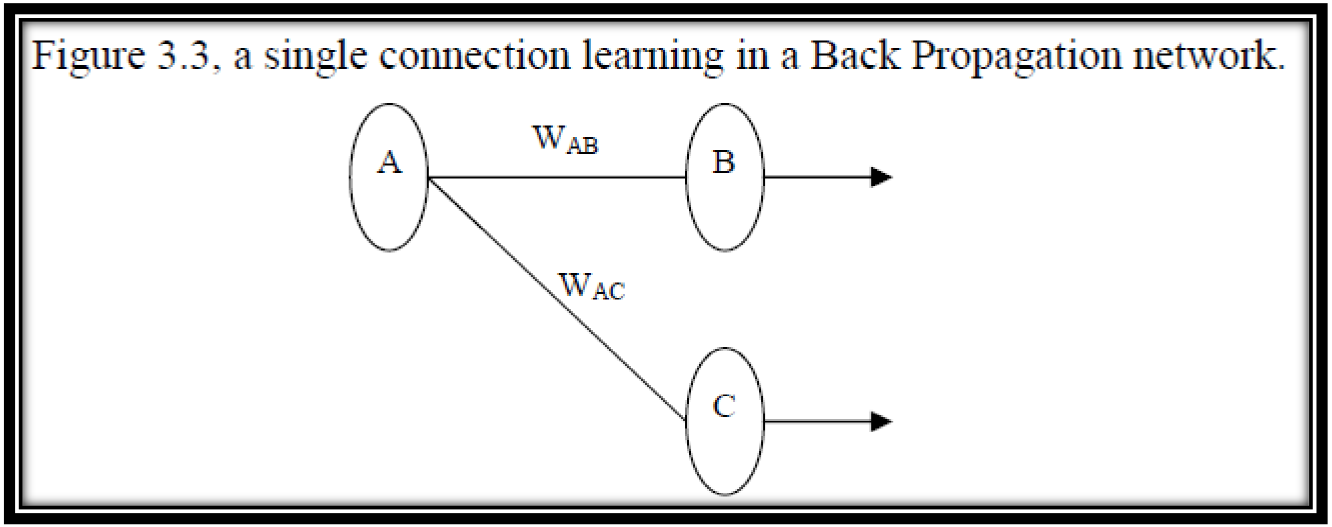
*Department of Computer Science*
*Faculty of Computers and Artificial Intelligence*
*Cairo University*
*Egypt*

# 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



# 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 :

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

**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

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

$$W^{+}_{AB} = W_{AB} + (Error_B \times Output_A)$$

→ 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

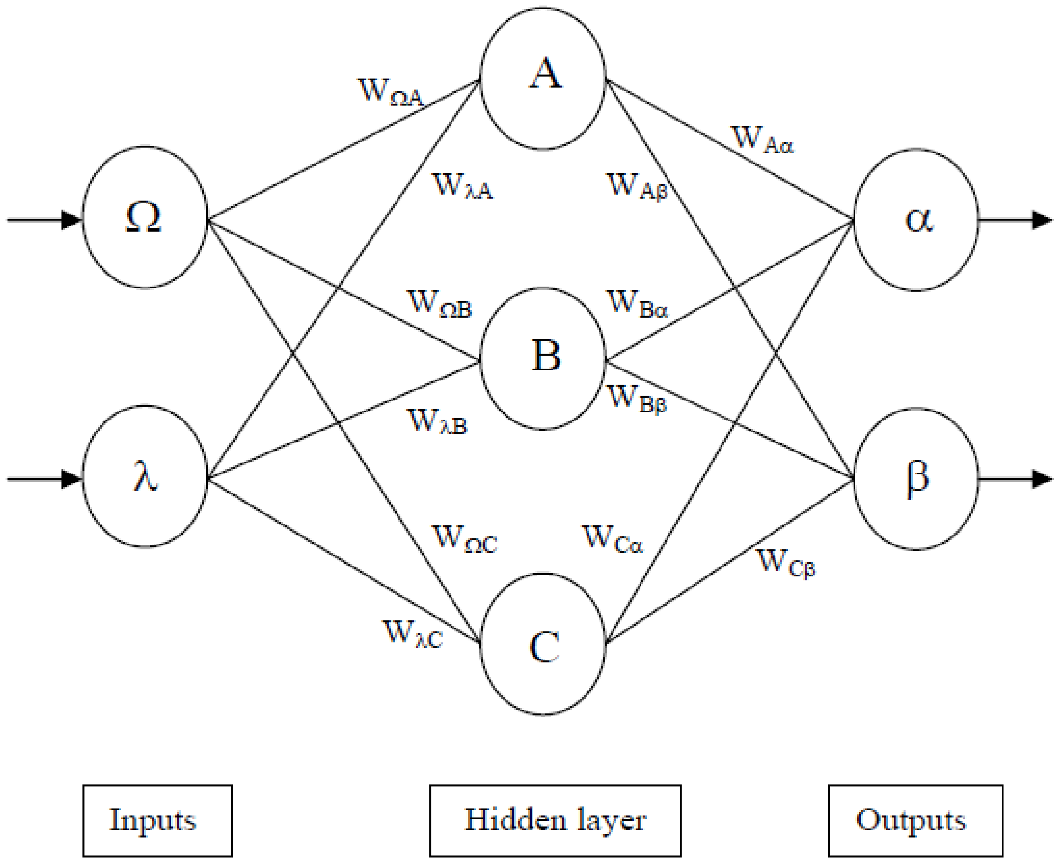
- **Step 5:** Calculate the Errors for the hidden layer neurons.
- Unlike the output layer we can't calculate these directly (because we don't have a Target).
- So we ***Back Propagate*** them from the output layer (hence the name of the algorithm).

**$\text{Error}_A$ = $\text{Output}_A$ (1 - $\text{Output}_A$)($\text{Error}_B$ $W_{AB}$ + $\text{Error}_C$ $W_{AC}$)**

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

- 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

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

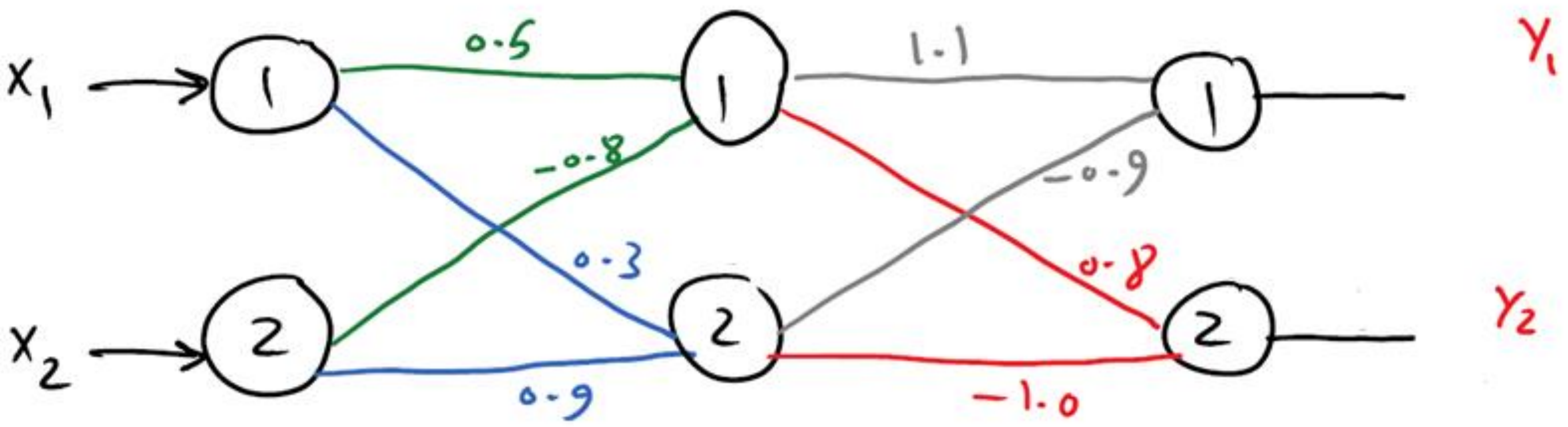
The constant $\eta$ (called the learning rate, and nominally equal to one) is put in to speed up or slow down the learning if required.

# 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



# 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:

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