ARTIFICIAL NEURAL NETWORKS
BACKPROPAGATION EXAMPLE

BACKPROPAGATION(*training_examples*, η, n_{in}, n_{out}, n_{hidden})

Each training example is a pair of the form $\langle \vec{x}, \vec{t} \rangle$, *where* $\vec{x}$ *is the vector of network input values, and* $\vec{t}$ *is the vector of target network output values.*

η *is the learning rate (e.g., .05).* n_{in} *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_{in} inputs, n_{hidden} hidden units, and n_{out} output units.
- Initialize all network weights to small random numbers (e.g., between −.05 and .05).
- Until the termination condition is met, Do
  - For each $\langle \vec{x}, \vec{t} \rangle$ in *training_examples*, Do

    Propagate the input forward through the network:

    1. Input the instance $\vec{x}$ 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 output unit k, calculate its error term δ_k

    $$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k) \tag{T4.3}$$

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

    $$\delta_h \leftarrow o_h(1 - o_h) \sum_{k \in outputs} w_{kh}\delta_k \tag{T4.4}$$

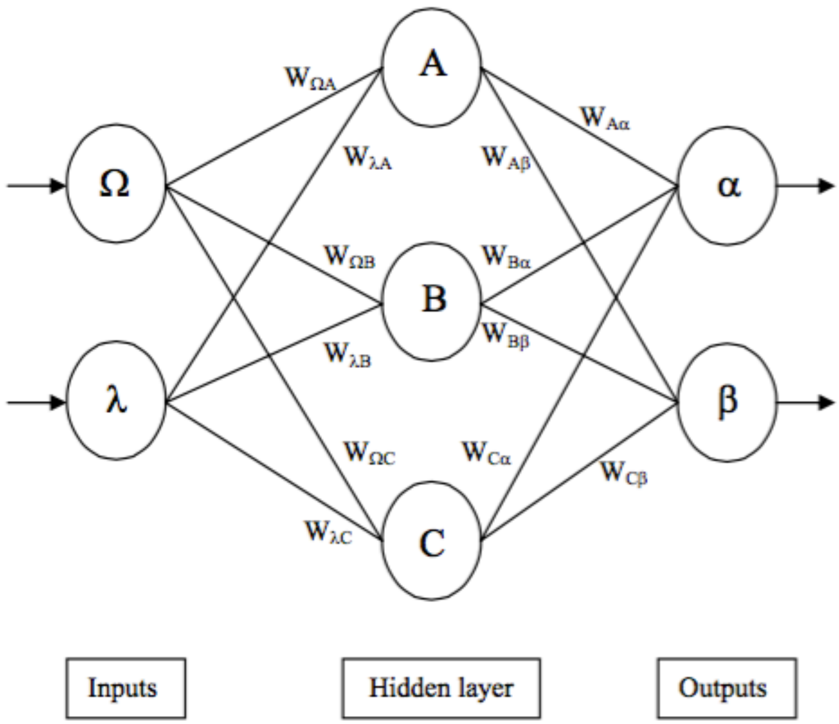
    4. Update each network weight w_{ji}

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

    where

    $$\Delta w_{ji} = \eta\, \delta_j\, x_{ji} \tag{T4.5}$$

Steps:



1. Assume random weights, apply inputs to network and evaluate the output – remember this is an initial estimate.

2. Calculate δ of output neurons –

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

3. Calculate δ of hidden units –

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

4. Change output layer weights, such as $W_{A\alpha}$, etc

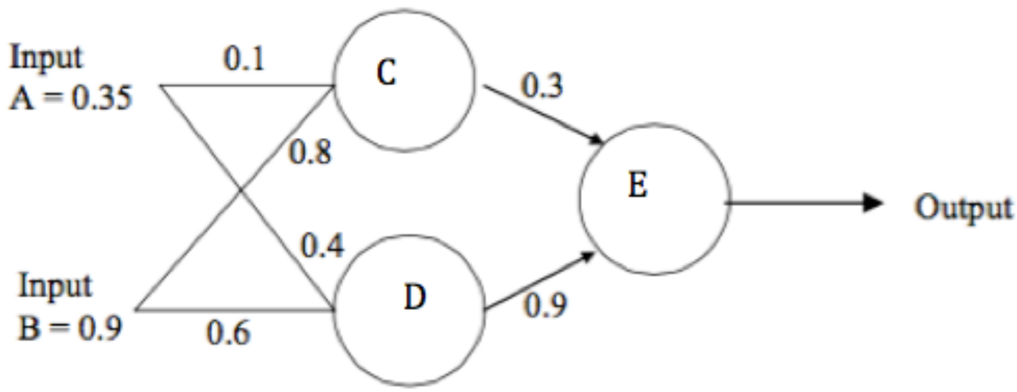
$$W^+_{A\alpha} = W_{A\alpha} + \eta\delta_\alpha\,out_A \qquad W^+_{A\beta} = W_{A\beta} + \eta\delta_\beta\,out_A$$
$$W^+_{B\alpha} = W_{B\alpha} + \eta\delta_\alpha\,out_B \qquad W^+_{B\beta} = W_{B\beta} + \eta\delta_\beta\,out_B$$
$$W^+_{C\alpha} = W_{C\alpha} + \eta\delta_\alpha\,out_C \qquad W^+_{C\beta}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

5. Change hidden layer weights:

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

η is called the learning factor (you can assume η =1, if no information provided).

For the neural net shown below:



Assuming the neurons (C, D) have a sigmoid activation function and the learning factor η = 1. Perform the following:

a. One forward pass on the network
b. Given that the target output of E = 0.5, perform a reverse pass (training pass) once.
c. With the training of step b, perform one more forward pass and see if the error goes down.

Steps:

1. Given the weights above, calculate the output values for C, D, and E:

Input to C = $net_c = \Sigma x_i w_i$ = 0.1x0.35 + 0.9x0.8 = 0.755

Output (using sigmoid activation) of C = $1/(1+ e^{-0.755})$ = 0.68

Input to D =

Output of D =

Input to E =

Output of E =

This is important and referred to as o

2. Output Error (δ_E) = (t-o)(1-o)o

t=target which is 0.5

3. Errors for hidden layer:

$\delta_C = o_C * (1 - o_C) * W_{CE} * \delta_E =$

$\delta_C = o_D * (1 - o_D) * W_{DE} * \delta_E =$

4. Change output layer weights:

$W_{CE} = W_{CE} + \delta_E * out_C =$

$W_{DE} = W_{DE} + \delta_E * out_D =$

Use out_C and out_D that you calculated in part 1

5. New hidden layer weight:

$W_{AC} = W_{AC} + (\delta_C * Input_A) =$

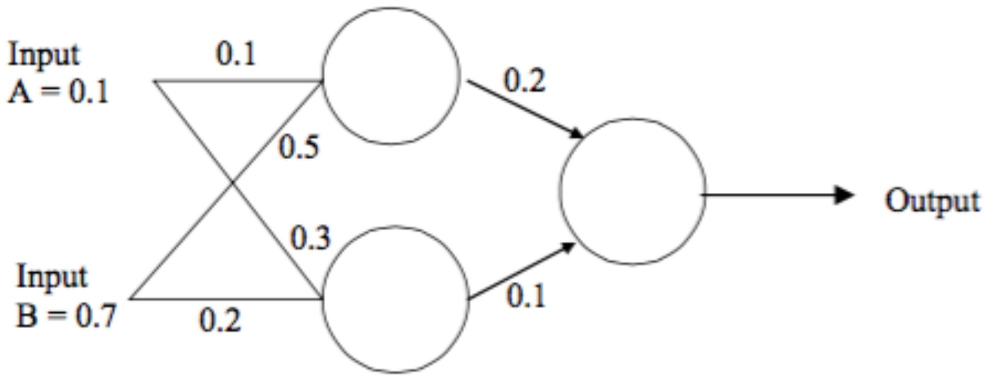
$W_{BC} = W_{BC} + (\delta_C * Input_B) =$

$W_{AD} = W_{AD} + (\delta_D * Input_A) =$

$W_{BD} = W_{BD} + (\delta_D * Input_B) =$

Re-compute value of error (δ)

2. Run one pass of the algorithm on the neural network below. The target of the output =1 and assume the learning rate=1



3. Run the Backpropagation algorithm on the following data. Stop when you obtain convergence.