Neural Network

Lecture 07

School of Energy Science & Engineering

Neural Network

- A **neural network** is a computational model inspired by the way biological neural networks in the human brain process information.
- It consists of interconnected layers of nodes, known as neurons, which work together to recognize patterns, learn from data, and make decisions.
- Each connection between neurons has a weight that adjusts as the network learns, allowing the network to modify its behaviour based on the input data it receives.

Purpose

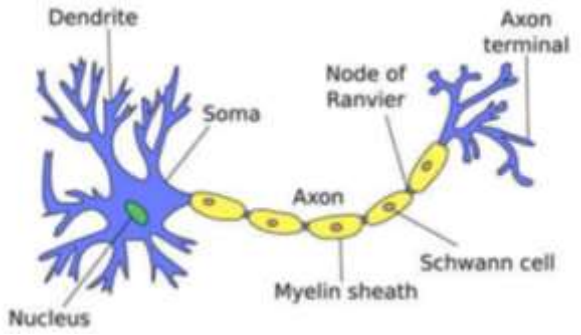
Pattern Recognition: Identify patterns in data, such as recognizing objects in images or detecting anomalies in datasets.

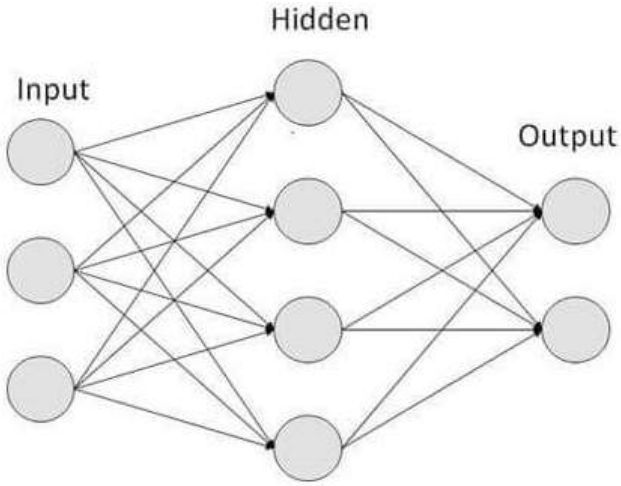
Classification: Assign input data into predefined categories, such as spam detection in emails.

Regression: Predict continuous values, like forecasting stock prices or housing values.

Optimization: Solve complex optimization problems, such as in resource allocation or scheduling.

Data Generation: Create new data that resembles a given dataset, as in generating realistic images or text





Neural Network

Real World Applications:

- **Computer Vision:** Used in facial recognition, object detection, and autonomous vehicles.
- **Natural Language Processing (NLP):** Powers applications like language translation, sentiment analysis, and chatbots.
- **Healthcare:** Assists in diagnosing diseases, predicting patient outcomes, and personalizing treatment plans.
- **Finance:** Used for fraud detection, algorithmic trading, and credit scoring.

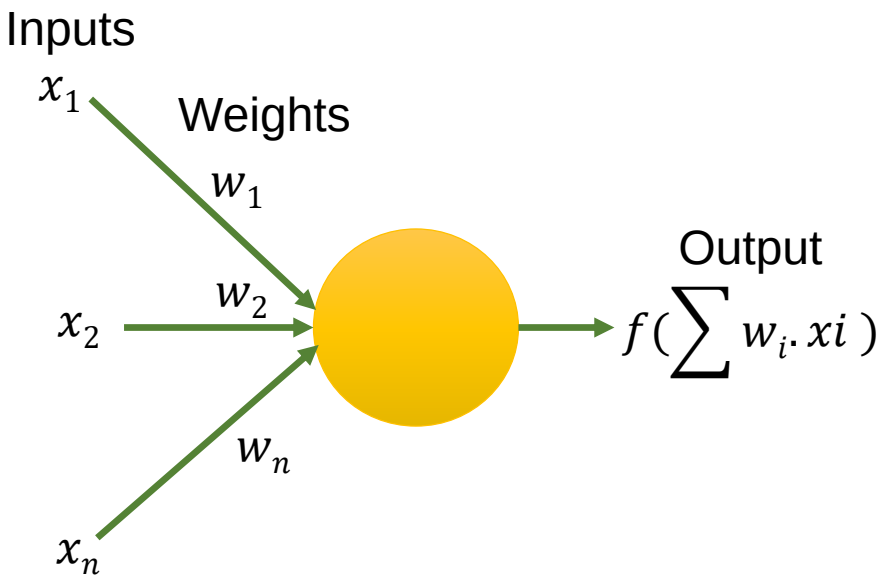
Basic Neuron Model In A Feedforward Network

- **Layers:**
- **Input Layer:** Receives the initial data.
- **Hidden Layers:** Intermediate layers where computations are performed.
- **Output Layer:** Produces the final result or prediction.

Inputs x_i arrive through pre-synaptic connections

- **Weights:** Values assigned to the connections between neurons that determine the strength of the signal being passed. Synaptic efficacy is modeled using real **weights** w_i.

- The response of the neuron is a **nonlinear function** ***f*** of its weighted inputs



Activation Function to Neuron's Output

A mathematical function applied to each neuron's output to introduce non-linearity, enabling the network to solve complex problems.

- The response function is normally nonlinear
- Samples include
  - Sigmoid

$$f(x) = \frac{1}{1 + e^{-\lambda x}}$$

  - Piecewise linear

$$f(x) = \begin{cases} x, if\ x \geq \theta \\ 0, if\ x < \theta \end{cases}$$

Backpropagation Preparation

- **Training Set**

A collection of input-output patterns that are used to train the network.

- **Testing Set**

A collection of input-output patterns that are used to assess network performance.

- **Learning Rate-η**

A scalar parameter, analogous to step size in numerical integration, used to set the rate of adjustments.

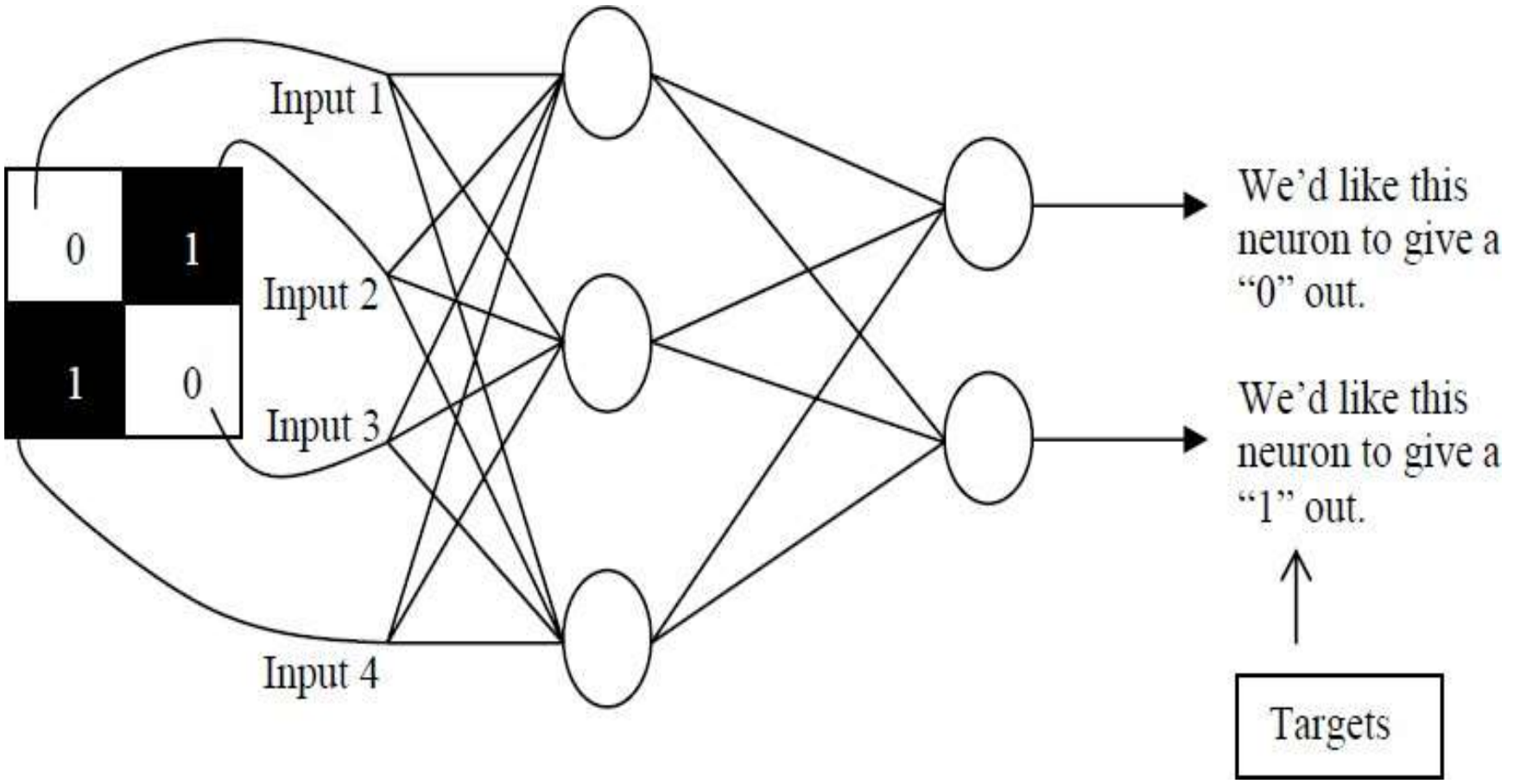
Network Error

- Total-Sum-Squared-Error (TSSE)

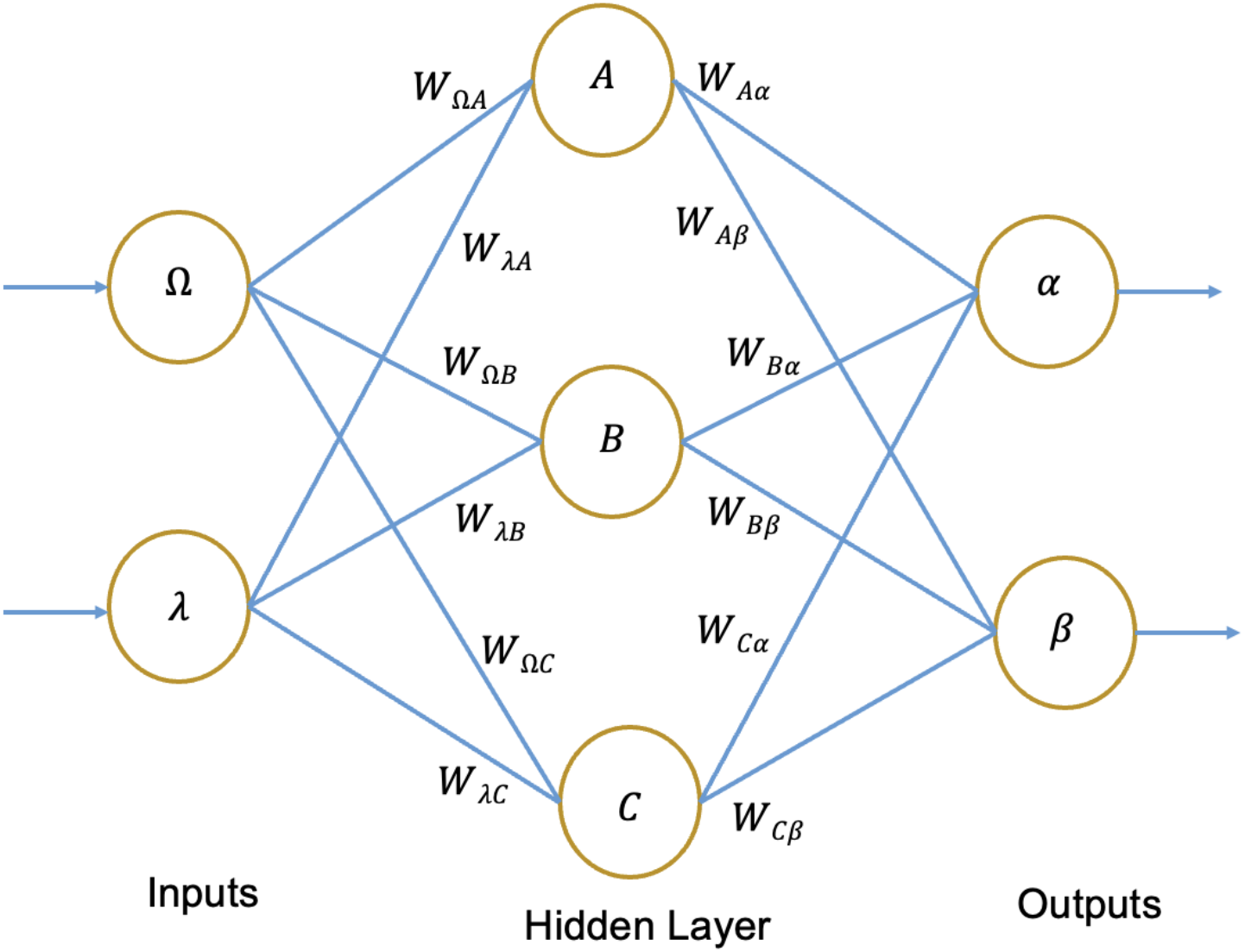
$$TSSE = \frac{1}{2} \sum_{patterns} \sum_{outputs} (desired - actual)^2$$

- Root-Mean-Squared-Error (RMSE)

$$RMSE = \sqrt{2 * \frac{TSSE}{\#patterns * \#outputs}}$$


Input 1
Input 2
Input 3
Input 4
0
1
1
0
We'd like this neuron to give a "0" out.
We'd like this neuron to give a "1" out.
Targets

A reverse pass of Back Propagation



Pseudo Code

1. Apply the inputs to the network and work out the output – remember this initial output could be anything, as the initial weights were random numbers.

2. Next work out the error for neuron B. The error is *What you want – What you actually get*, in other words:

$$Error_B = Output_B(1 - Output_B)(\text{Target}_B - Output_B)$$

The "*Output(1-Output)*" term is necessary in the equation because of the Sigmoid Function – if we were only using a threshold neuron it would just be *(Target – Output)*.

3. Change the weight. Let W^+_{AB} be the new (trained) weight and W_{AB} be the initial weight.

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

Notice that it is the output of the connecting neuron (neuron A) we use (not B). We update all the weights in the output layer in this way.

Pseudo Code

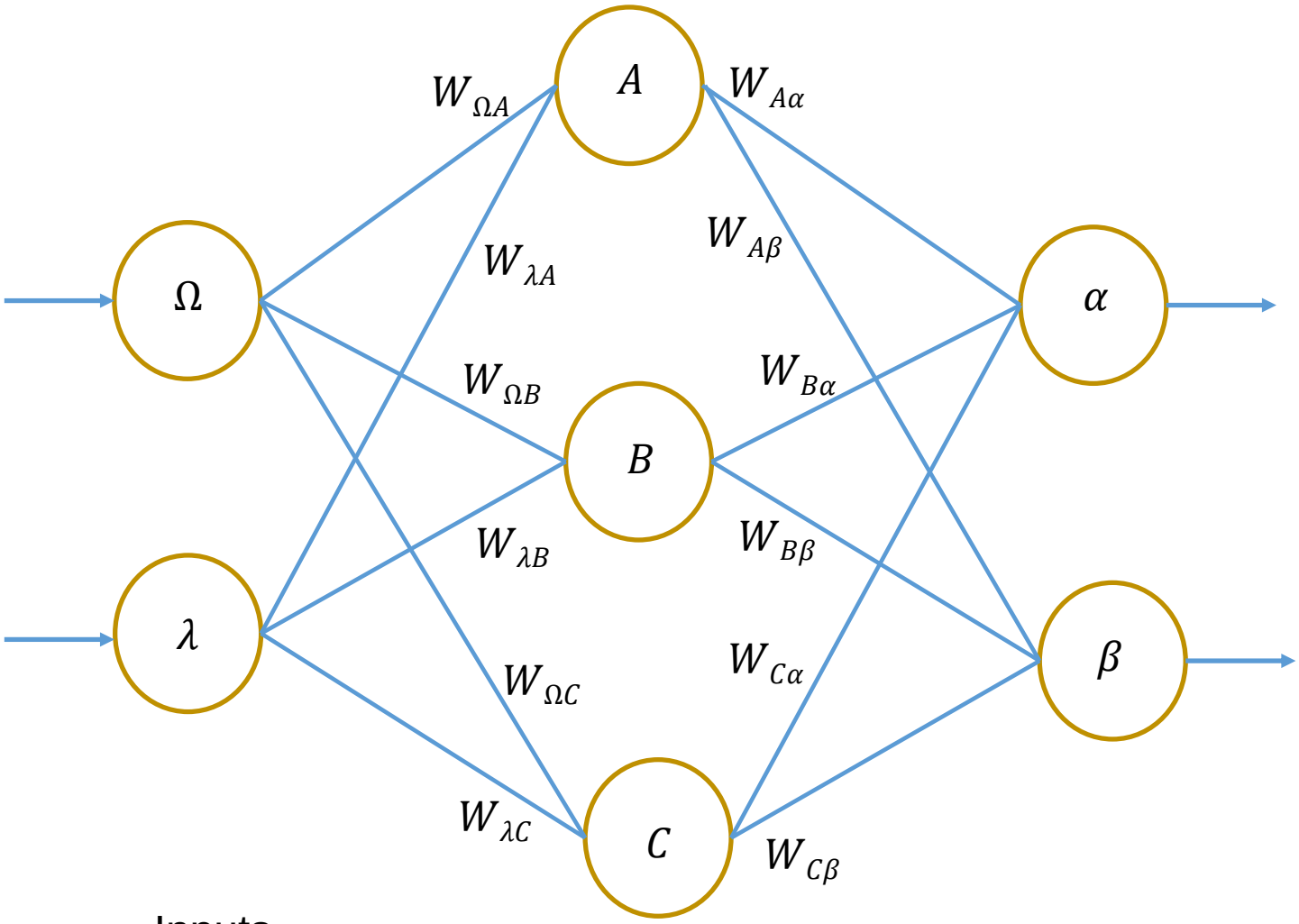
4.Calculate the Errors for the hidden layer neurons. Unlike the output layer we can't calculate these directly (because we don't have a Target), so we *Back Propagate* them from the output layer (hence the name of the algorithm). This is done by taking the Errors from the output neurons and running them back through the weights to get the hidden layer errors. For example if neuron A is connected as shown to B and C then we take the errors from B and C to generate an error for A.

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

Again, the factor "*Output (1 - Output)*" is present because of the sigmoid squashing function.

5.Having obtained the Error for the hidden layer neurons now proceed as in stage 3 to change the hidden layer weights. By repeating this method we can train a network of any number of layers.

A reverse pass of Back Propagation



Discriminative Linear Classification

1. Calculate errors of output neurons

$$\delta_\alpha = out_\alpha(1 - out_\alpha) * (\text{Target}_\alpha - out_\alpha)$$
$$\delta_\beta = out_\beta(1 - out_\beta) * (\text{Target}_\beta - out_\beta)$$

2. Change output layer weights

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

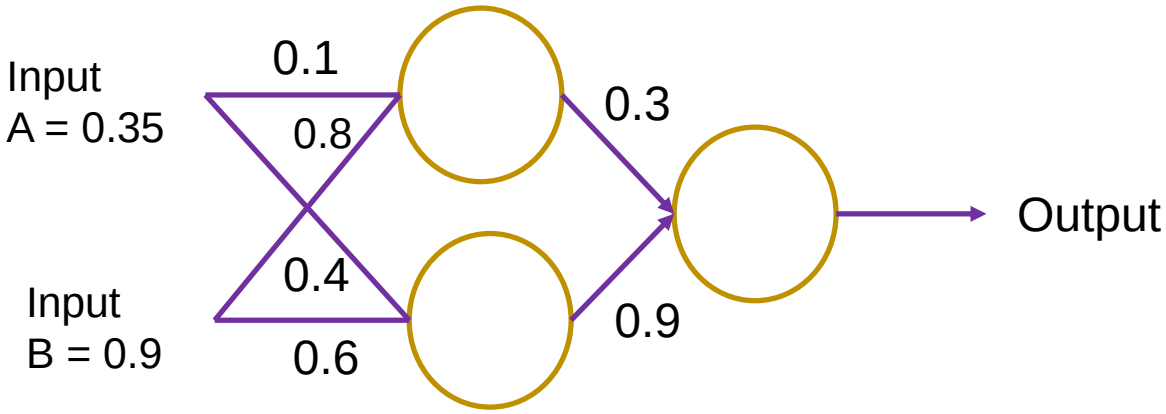
3. Calculate (back-propagate) hidden layer errors

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

4. Change hidden layer weights

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

Numerical



Assume that the neurons have a sigmoid activation function ($\lambda = 1$)

1. Perform a forward pass on the network.
2. Perform a reverse pass (training) once (target=0.5).
3. Perform a further forward pass and comment on the result.

(1) Input to top neuron = (0.35*0.1) + (0.9*0.8) = 0.755; Output = $\frac{1}{1+e^{-0.755}} = 0.68$

Input to bottom neuron= (0.9*0.6) + (0.35*0.4) = 0.68; Output = $\frac{1}{1+e^{-0.68}} = 0.6637$

Input to final neuron = (0.3*0.68) + (0.9*0.6637) = 0.80133; Output = $\frac{1}{1+e^{-0.80133}} = 0.69$

(2) Output error $\delta =$
$(target - output)(1 - output)output = (0.5 - 0.69)(1 - 0.69)0.69 = -0.0406$

New weights for output layer

$w1 \mathrel{+}= w1 + (\delta * input) = 0.3 + (-0.0406 * 0.68) = 0.272392$

$w2 \mathrel{+}= w2 + (\delta * input) = 0.9 + (-0.0406 * 0.6637) = 0.87305$

Errors for hidden layers:

$\delta 1 = \delta * w1 * (1 - \text{output1}) * \text{output1} = -0.0406 * 0.272392 * (1 - 0.68) * 0.68$
$= -2.406 * 10^{-3}$

$\delta 2 = \delta * w2 * (1 - \text{output2}) * \text{output2} = -0.0406 * 0.87305 * (1 - 0.6637) * 0.6637$
$= -7.916 * 10^{-3}$

$w5 \mathrel{+}= 0.4 + (-7.916 * 10^{-3} * 0.35) = 0.3972$

New hidden layer weights:

$w3 \mathrel{+}= 0.1 + (-2.406 * 10^{-3} * 0.35) = 0.09916$

$w4 \mathrel{+}= 0.8 + (-2.406 * 10^{-3} * 0.9) = 0.7978$

$w6 \mathrel{+}= 0.6 + (-7.916 * 10^{-3} * 0.9) = 0.5928$

(3) Input to top neuron = (0.35*0.09916) + (0.9*0.7978) = 0.752726

Output = $\frac{1}{1+e^{-0.752726}} = 0.6798$

Input to bottom neuron = (0.9*0.5928) + (0.35*0.3972) = 0.67254

Output,= $\frac{1}{1+e^{-0.67254}} = 0.6621$

Input to final neuron = (0.3*0.6798) + (0.9*0.6621) = 0.79983

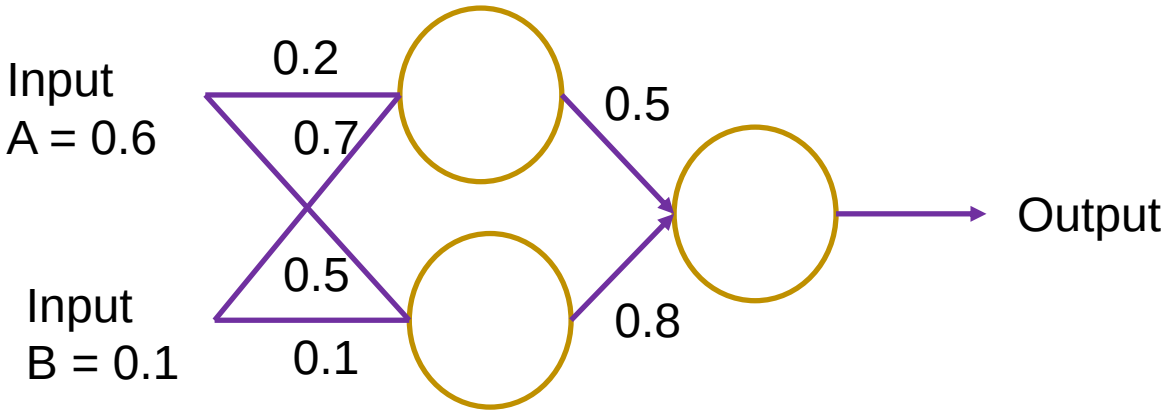
Output = $\frac{1}{1+e^{-0.79983}} = 0.68994$

Previous error = (target-output) = (0.5-0.69) = -0.19

Current error = (target-output) = (0.5 - 0.68994) = -0.18994

∴ Error reduced.

Numerical



Assume that the neurons have a sigmoid activation function

($\lambda = 2$)

1. Perform a forward pass on the network.
2. Perform a reverse pass (training) once (target=0.8).
3. Perform a further forward pass and calculate current error.