Artificial Intelligence
&
Neural Networks

CSE-351

Lecture-13

Md. Mahbubur Rahman
Assistant Professor & Coordinator, Bangladesh
University of Business & Technology

Content

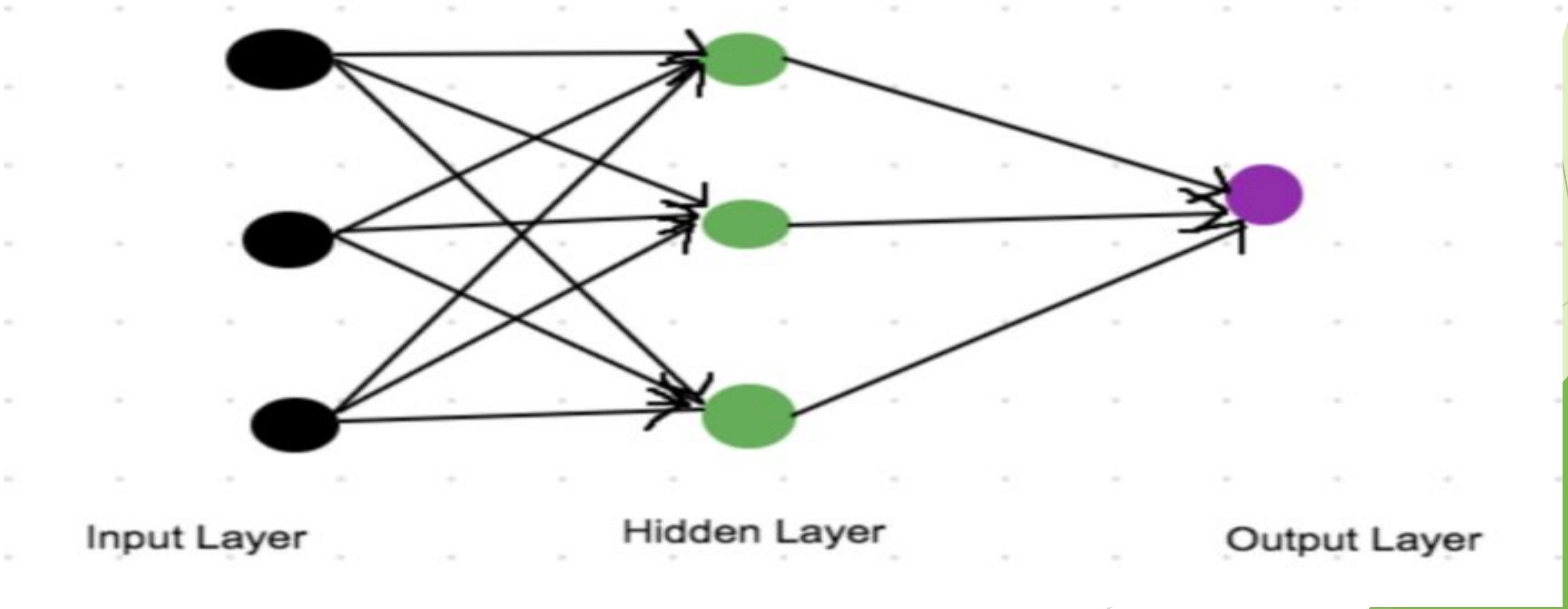
Artificial Neural Networks

- Networks

Neural Network Basics

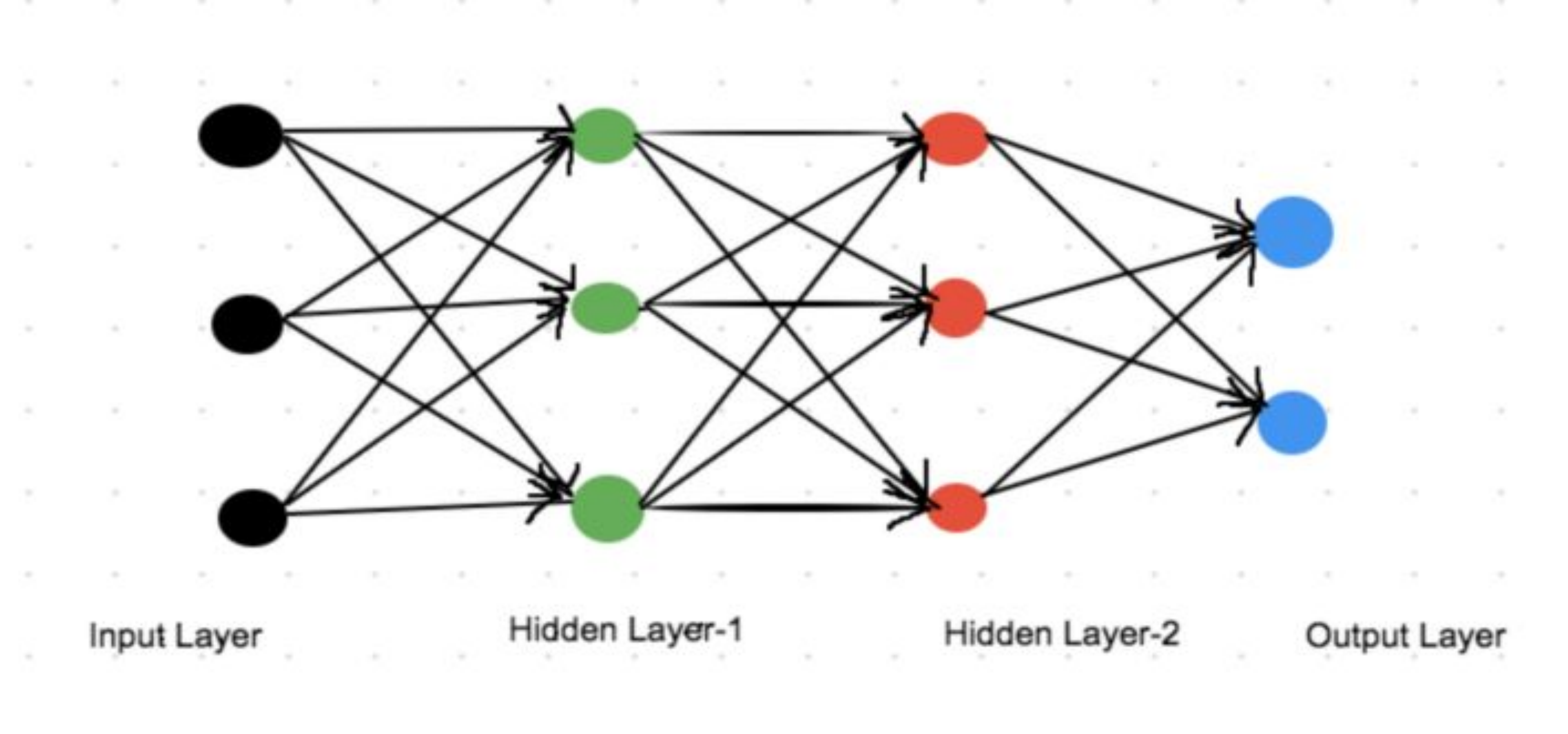
A general structure of a neural network will have three layers -

1. Input Layer, which has input nodes
2. Hidden Layer, which has hidden nodes
3. Output Layer, which has output nodes



Neural Network Basics

There could be multiple hidden layers and in that case it is known as Multi Layer Perceptron (MLPs) which look like this.



Neural Network

Before going to depth we should know….

- Forward-propagation
- Backward-propagation or Back propagation
- Error and Gradient-decent

Forward-Propagation at ANN

As the name suggests, the input data is fed in the forward direction through the network.

Each hidden layer accepts the input data, processes it as per the activation function and passes to the successive layer.

In order to generate some output, the input data should be fed in the forward direction only.

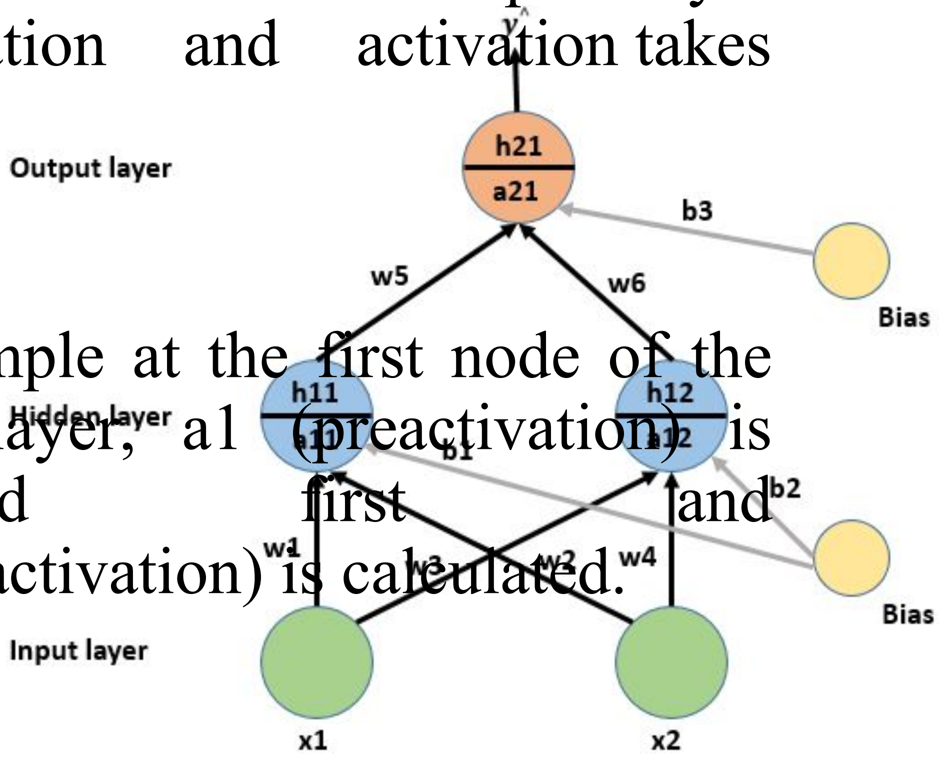
The data should not flow in reverse direction during output generation otherwise it would form a cycle and the output could never be generated. Such network configurations are known as ***feed-forward network***.

The *feed-forward network* helps in *forward propagation*.

Forward-Propagation at ANN

During forward propagation at each node of hidden and output layer preactivation and activation takes place.

For example at the first node of the hidden layer, a1 (preactivation) is calculated first and then h1(activation) is calculated.



Forward-Propagation at ANN

a1 is a weighted sum of inputs. Here, the weights are randomly generated.

a1 = w1*x1 + w2*x2 + b1 = 1.76* 0.88 + 0.40*(-0.49) + 0 = 1.37 approx. and **h1** is the value of activation function applied on **a1**.

$$h1 = \frac{1}{1+e^{-a1}} = 0.8 \text{ approx.}$$

Similarly

a2 = w3*x1 + w4*x2 + b2 = 0.97 *0.88 + 2.24 *(- 0.49)+ 0 = -2.29 approx and

$$h2 = \frac{1}{1+e^{-a2}} = 0.44 \text{ approx.}$$

Forward-Propagation at ANN

Back-propagation

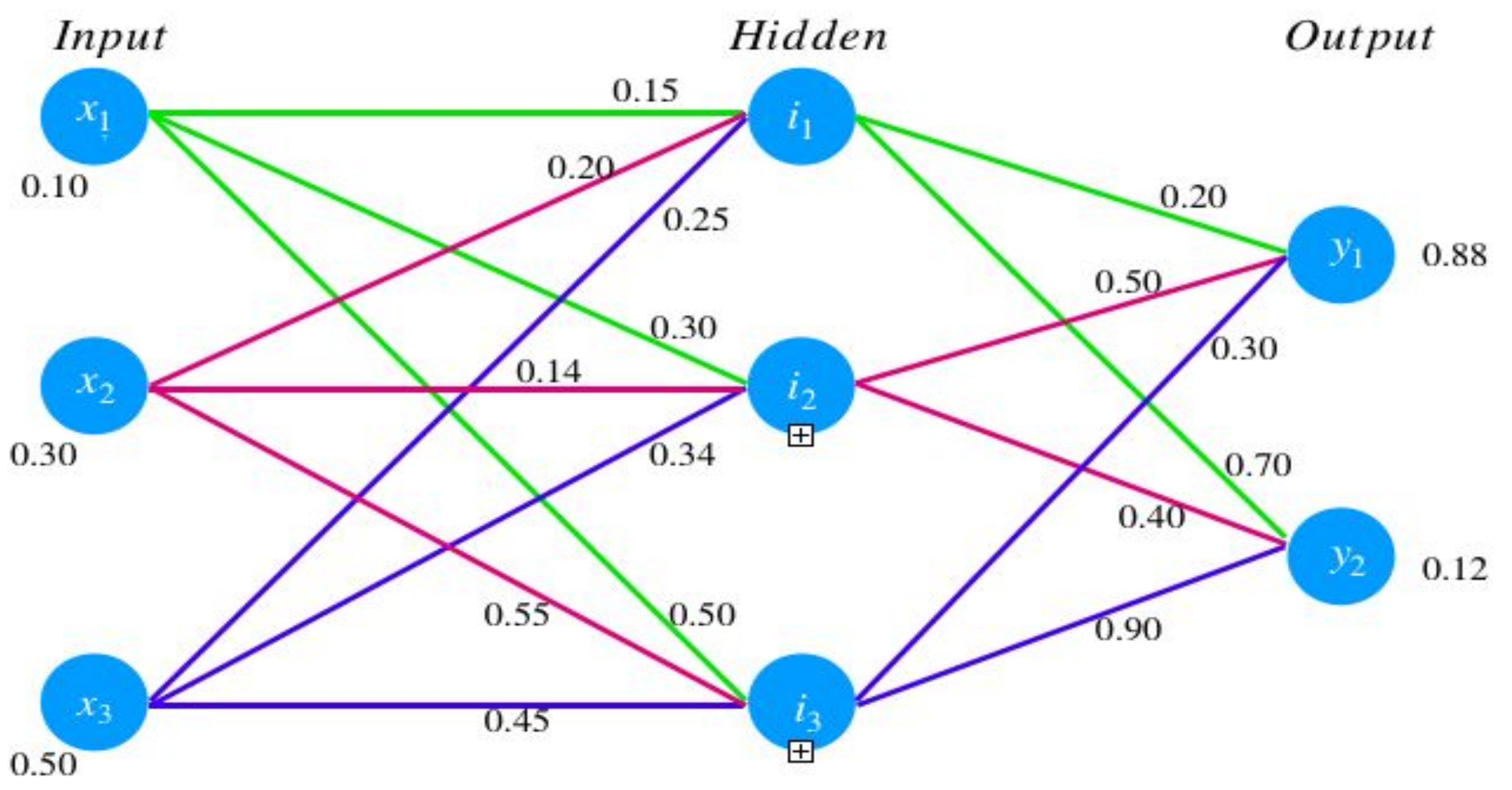
Back-propagation is a supervised learning algorithm, for training Multi-layer Perceptron's (Artificial Neural Networks).

The Back-propagation algorithm looks for the minimum value of the error function in weight space using a technique called the delta rule or gradient descent. The weights that minimize the error function is then considered to be a solution to the learning problem.

Allows the information to go back from the cost backward through the network in order to compute the gradient. Therefore, loop over the nodes starting at the final node in reverse topological order to compute the derivative of the final node output with respect to each edge's node tail. Doing so will help us know who is responsible for the most error and change the parameters in that direction.

Forward and Backward propagation by example

Suppose a neural network look like as-

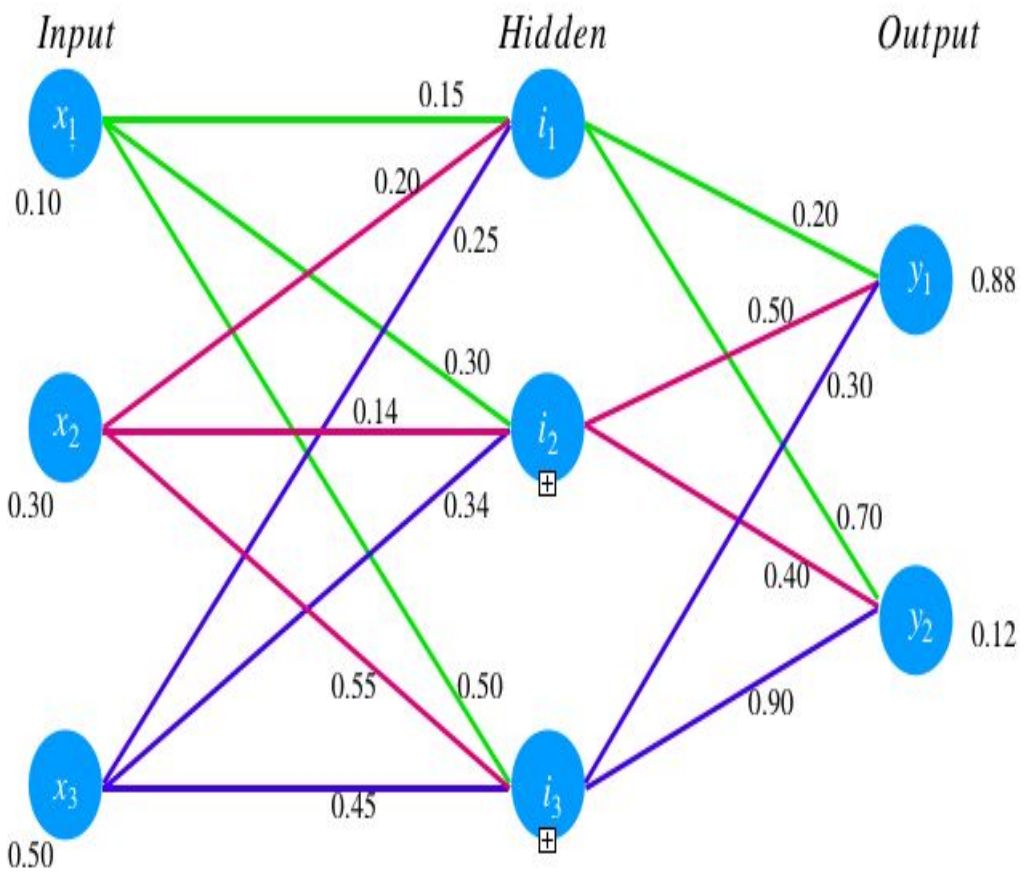


Forward and Backward propagation by example

Here,

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.10 \\ 0.30 \\ 0.50 \end{bmatrix}$$

$$W_1 = \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} = \begin{bmatrix} 0.15 & 0.20 & 0.25 \\ 0.30 & 0.14 & 0.34 \\ 0.50 & 0.55 & 0.45 \end{bmatrix}$$



Forward and Backward propagation by example

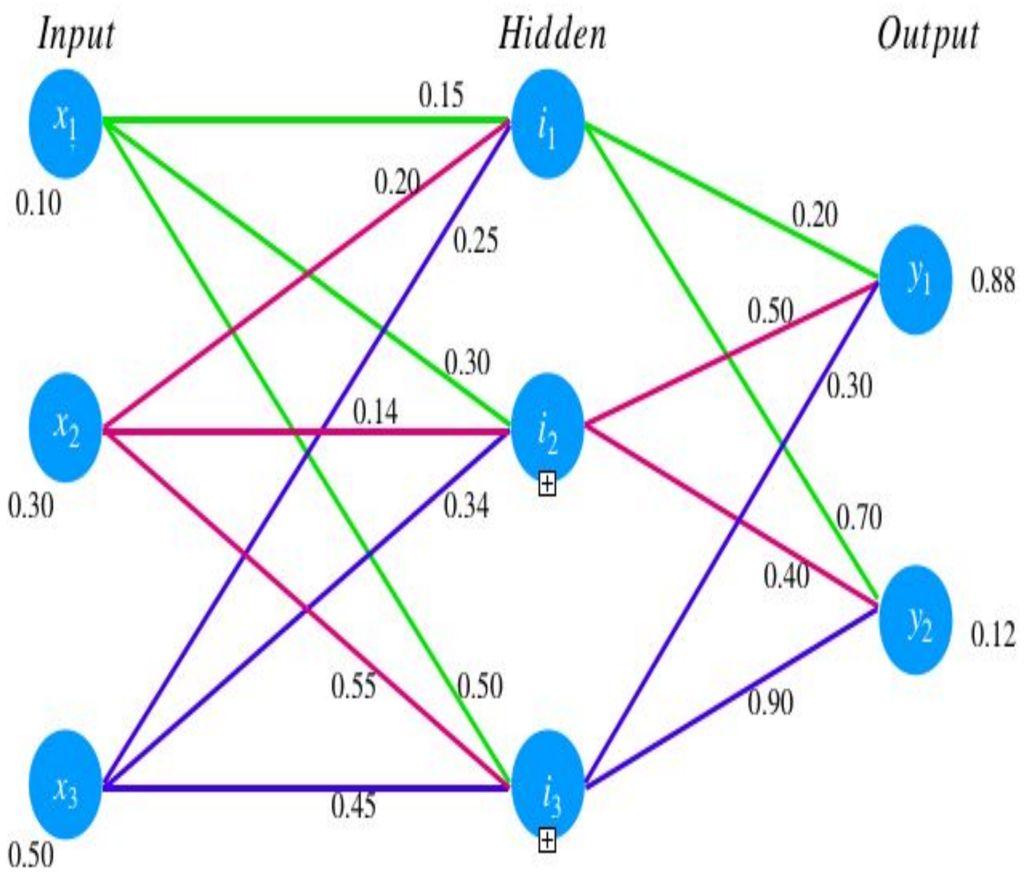
Here,

$$W_2 = \begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{13} & w_{14} & w_{15} \end{bmatrix} = \begin{bmatrix} 0.20 & 0.50 & 0.30 \\ 0.70 & 0.40 & 0.90 \end{bmatrix}$$

$$b_1 = \begin{bmatrix} .25 \\ .25 \\ .25 \end{bmatrix}$$

$$b_2 = \begin{bmatrix} .12 \\ .12 \end{bmatrix}$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0.88 \\ 0.12 \end{bmatrix}$$



Forward and Backward propagation by example

Logistic function:

$Z = WX + b$

Here,

W = weight matrix

X = input vector

b = bias vector

Logistic function:

$$a = \frac{1}{1 + e^{-z}}$$

Forward and Backward propagation by example

Hidden layer calculation will be-

$$Z_1 = W_1X + b1$$

$$Z_1 = \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_1 \\ b_1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.15 & 0.20 & 0.25 \\ 0.30 & 0.14 & 0.34 \\ 0.50 & 0.55 & 0.45 \end{bmatrix} * \begin{bmatrix} 0.10 \\ 0.30 \\ 0.50 \end{bmatrix} + \begin{bmatrix} 0.25 \\ 0.25 \\ 0.25 \end{bmatrix}$$

$$= \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} w_1x_1 + w_2x_2 + w_3x_3 + b_1 \\ w_4x_1 + w_5x_2 + w_6x_3 + b_1 \\ w_7x_1 + w_8x_2 + w_9x_3 + b_1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.45 \\ 0.492 \\ 0.69 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} \frac{1}{1+e^{-z_1}} \\ \frac{1}{1+e^{-z_2}} \\ \frac{1}{1+e^{-z_3}} \end{bmatrix} = \begin{bmatrix} \frac{1}{1+e^{-0.45}} \\ \frac{1}{1+e^{-0.492}} \\ \frac{1}{1+e^{-0.69}} \end{bmatrix} = \begin{bmatrix} 0.61063923 \\ 0.62057747 \\ 0.66596693 \end{bmatrix}$$

Forward and Backward propagation by example

Output layer calculation will be-

$$Z_2 = W_2 A_1 + b_2$$

$$Z_2 = \begin{bmatrix} w_{10} & w_{11} & w_{12} \\ w_{13} & w_{14} & w_{15} \end{bmatrix} * \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} + \begin{bmatrix} b_2 \\ b_2 \end{bmatrix}$$

$$Z_2 = \begin{bmatrix} 0.20 & 0.50 & 0.30 \\ 0.70 & 0.40 & 0.90 \end{bmatrix} * \begin{bmatrix} 0.61063923 \\ 0.62057747 \\ 0.66596693 \end{bmatrix} + \begin{bmatrix} 0.12 \\ 0.12 \end{bmatrix}$$

$$Z_2 = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} w_{10}\, a_1 + w_{11}\, a_2 + w_{12}\, a_3 + b_2 \\ w_{13}\, a_1 + w_{14}\, a_2 + w_{15}\, a_3 + b_2 \end{bmatrix}$$

$$Z_2 = \begin{bmatrix} 0.75220666 \\ 1.39504869 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} \dfrac{1}{1 + e^{-z_1}} \\ \dfrac{1}{1 + e^{-z_2}} \end{bmatrix} = \begin{bmatrix} \dfrac{1}{1 + e^{-0.75220666}} \\ \dfrac{1}{1 + e^{-1.39504869}} \end{bmatrix} = \begin{bmatrix} 0.67965933 \\ 0.80139701 \end{bmatrix}$$

Forward and Backward propagation by example

- Here original output is: [0.88, 0.12]
- but we got: [0.67965933, 0.80139713].
- Now we should calculate the error.