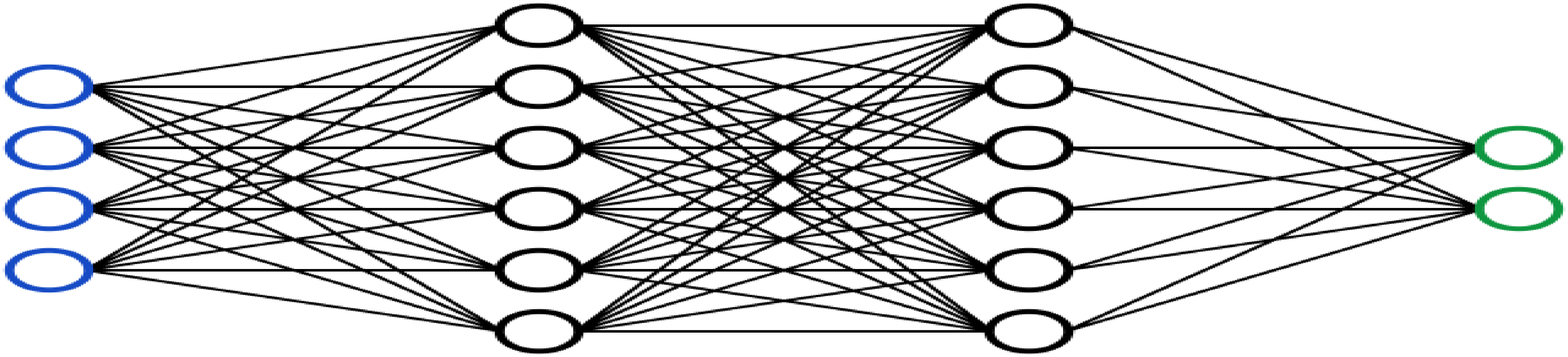


Artificial Neural Network (ANN)

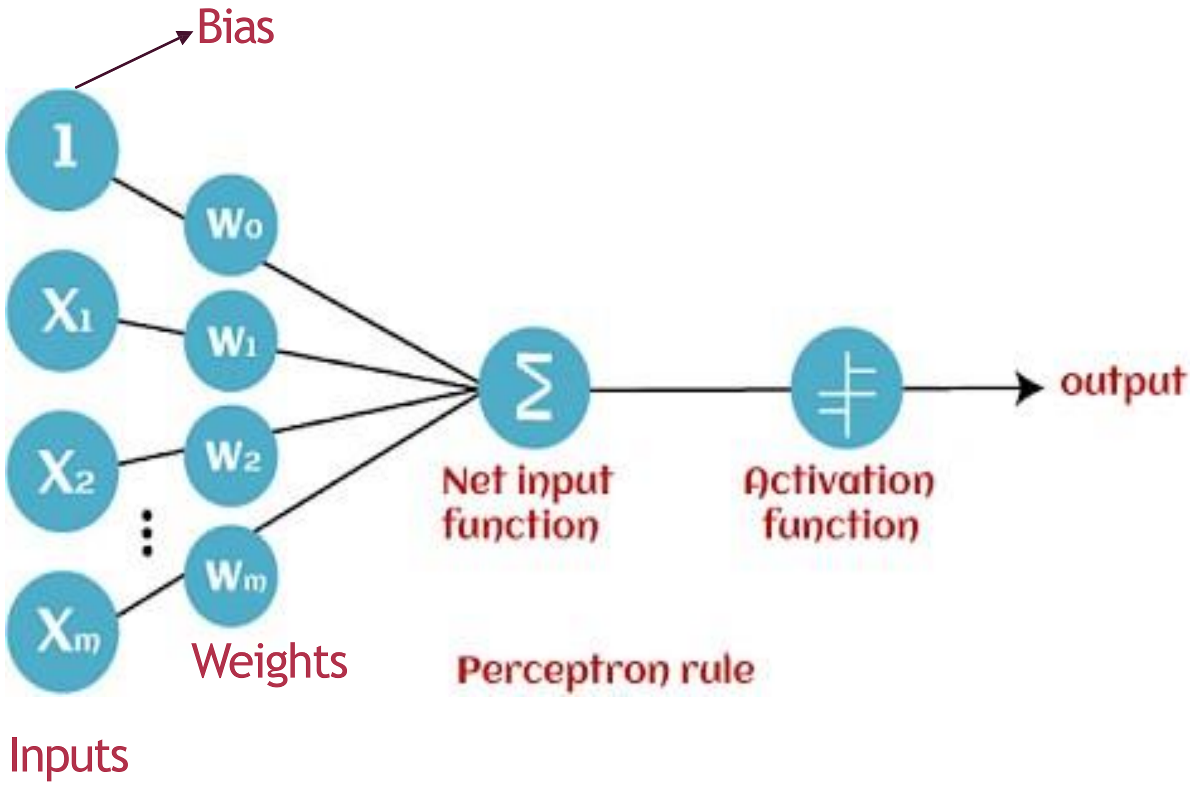
Lecture 5

Dina El-Manakhly, Ph. D.

dina_almnakhly@science.suez.edu.eg

Single-layer perceptron (Revision)

- The single-layer perceptron is one of the simplest forms of artificial neural networks (ANNs). It consists of a single layer of artificial neurons, also known as perceptron, which process inputs and produce outputs.
- Each input is associated with a weight, and the perceptron computes a weighted sum of its inputs, applies a bias term, and then passes the result through an activation function. The output of the perceptron is typically binary, representing a decision boundary.
- Single-layer perceptron have limitations in their ability to model complex relationships between inputs and outputs, as they can only represent linear decision boundaries.



Single-layer perceptron (Revision)

Mathematically, the output y of a single-layer perceptron with n input features $(x_1, x_2, \dots, x_n)$ can be expressed as:

$$y = f(w_1x_1 + w_2x_2 + \dots + w_nx_n + b)$$

where:

- $w_1, w_2, \dots, w_n$ are the weights associated with each input,
- $x_1, x_2, \dots, x_n$ are the input features,
- b is the bias term,
- $f()$ is the activation function.

Neural network training steps

1. Weight initialization
2. Input application
3. Sum of inputs –weight product
4. Activation functions
5. Weight adaptation
6. Back to step3

If the predicted output is incorrect, the weights are adjusted. This update process continues until the predicted output matches the target output for all training examples.

How to update weights in SLP ?

- Single layer perceptron
  - Using perceptron learning algorithm
  - Using delta rule

Weight adaptation : Delta rule

- If the predicted output Y is not the same as the desired output d, then weights are to be adapted according to the following equation:

$$W(n+1) = W(n) + \eta[d(n) - Y(n)]X(n)$$

Where

$$W(n) = [b(n), W_1(n), W_2(n), W_3(n), \ldots, W_m(n)]$$

Learning Rate η $\quad$ $0 \leq \eta \leq 1$

SLP using delta rule

Calculate y =
1 if net >= threshold,
0 if net < threshold

Threshold should be given. If not, assume random threshold

Here we assume threshold = 0.1 → net < threshold

x1	x2	bias	w1	w2	w_bias	net	y	t	
0	0	1	0.1	0.2	-0.2	-0.2	0	0	Error= 0
0	1	1	0.1	0.2	-0.2	0	0	1	Error= 1
1	0	1	0.1	0.3	-0.1	0	0	1	Error= 1
1	1	1	0.2	0.3	0	0.5	1	1	Error= 0

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta(t - o)x_i$$

learning rate, target value, perceptron output, input value

Same example using delta rule

Assume learning rate = 0.1

Use these as initial weights for next epoch

SLP using delta rule

Next epoch:

x1	x2	bias	w1	w2	w_bias	net	y	t	
0	0	1	0.2	0.3	0	0	0	0	Error= 0
0	1	1	0.2	0.3	0	0.3	1	1	Error= 0
1	0	1	0.2	0.3	0	0.2	1	1	Error= 0
1	1	1	0.2	0.3	0	0.5	1	1	Error= 0

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \underbrace{\eta}_{\text{learning rate}}(\underbrace{t}_{\text{target value}} - \underbrace{o}_{\text{perceptron output}})\underbrace{x_i}_{\text{input value}}$$

Same example using delta rule

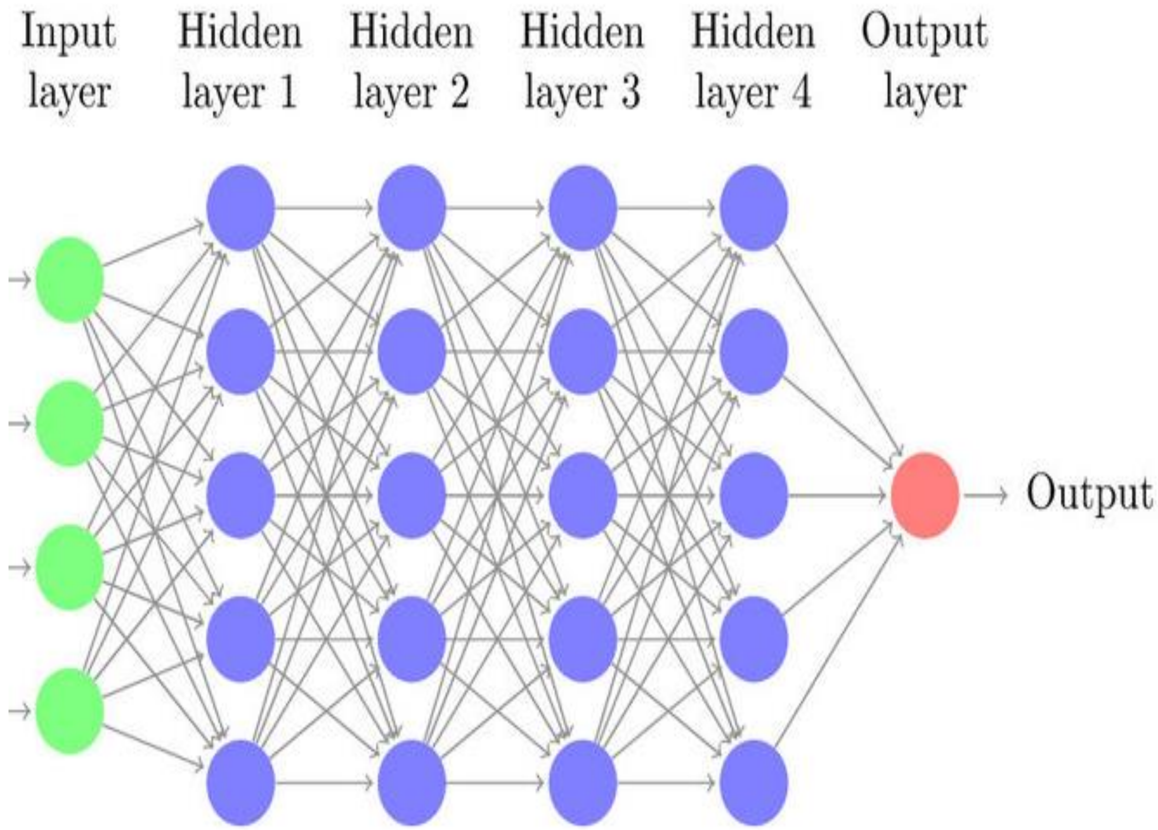
Assume learning rate = 0.1

Multi-layer perceptron

- Multi-layer perception is also known as MLP. It is fully connected dense layers, which transform any input dimension to the desired dimension.
  - "input dimension" refers to the number of features or variables present in the input data.

  For example, if you have a dataset with 100 samples and each sample has 10 features, the input dimension would be 10.

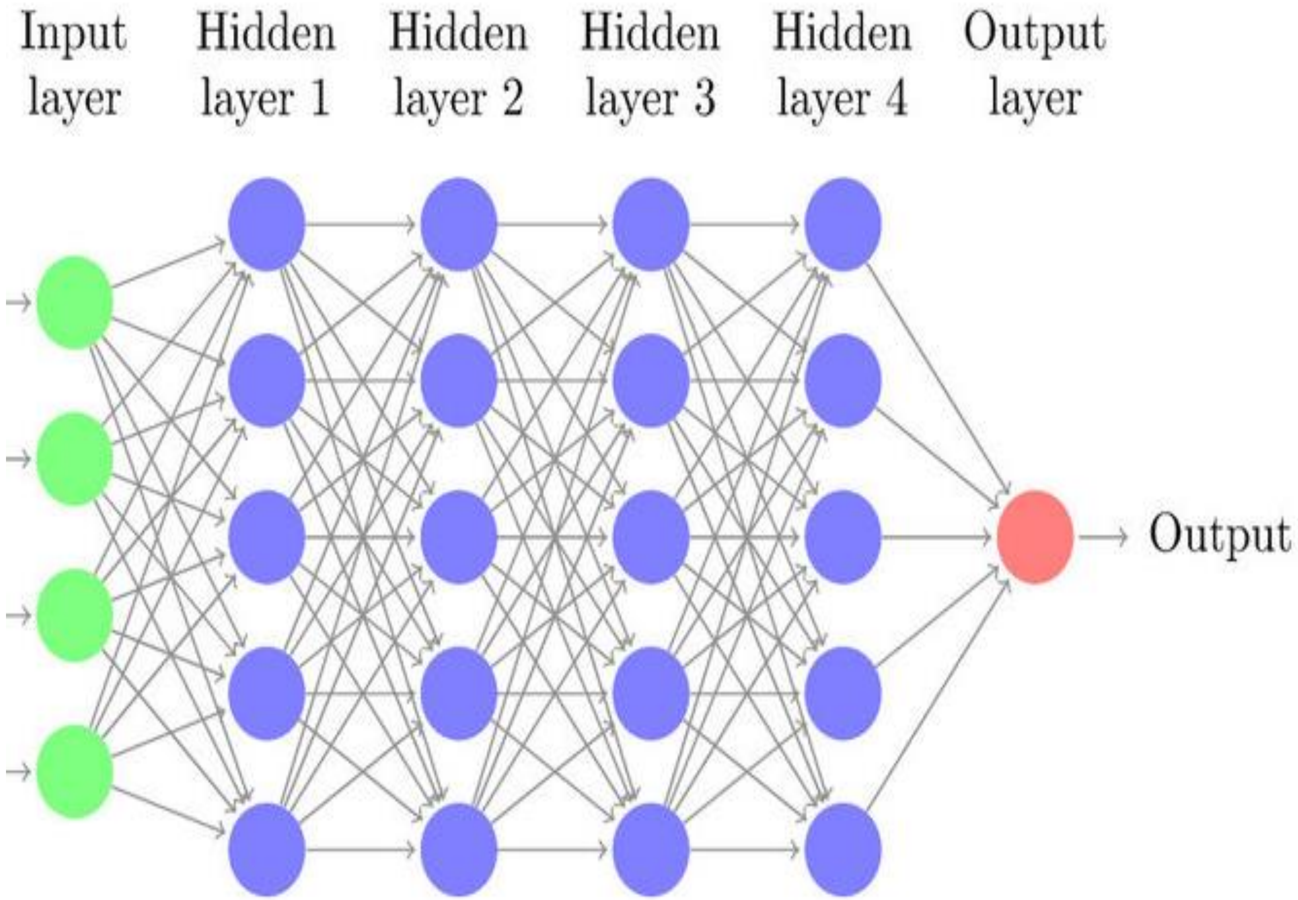
  - "desired dimension" refers to the number of output units or the dimensionality of the output space required for a specific task. This could be the number of classes in a classification task, the number of predicted values in a regression task, or any other desired output format.



Multi-layer perceptron

- A multi-layer perception is a neural network that has multiple layers. To create a neural network we combine neurons together so that the outputs of some neurons are inputs of other neurons

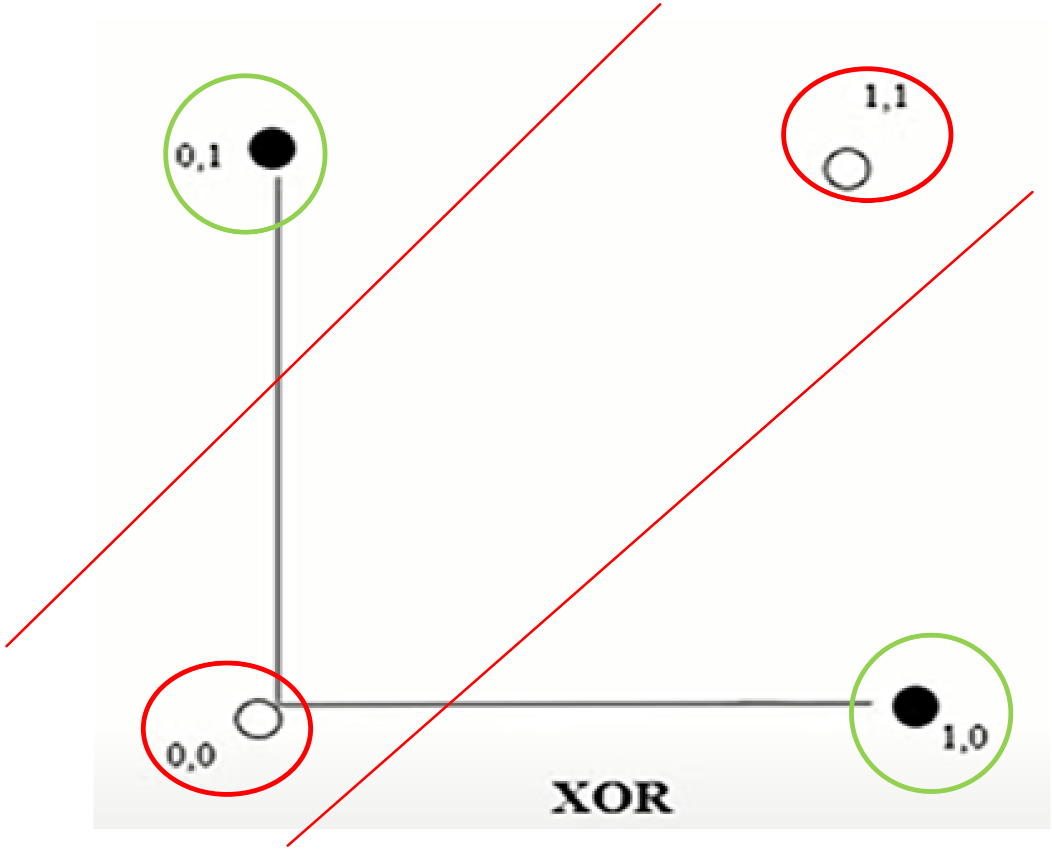
- A multi-layer perceptron has one input layer and for each input, there is one neuron(or node), it has one output layer with a single node for each output and it can have any number of hidden layers and each hidden layer can have any number of nodes.



XOR-Gate with multilayer perceptron

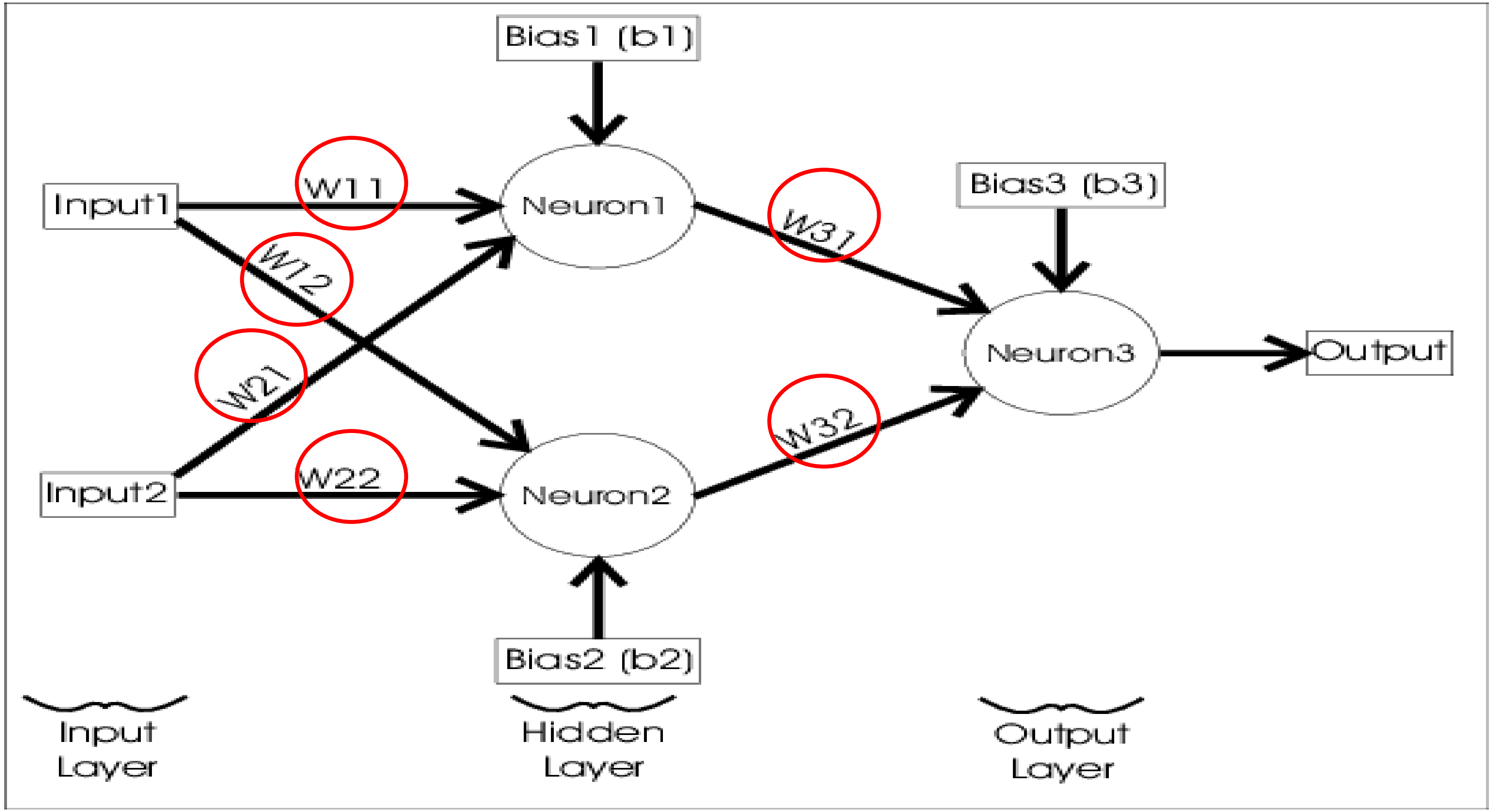
EX-OR (X-OR) Gate Truth Table

Inputs		Output X = A ⊕ B
A	B	
0	0	0
0	1	1
1	0	1
1	1	0



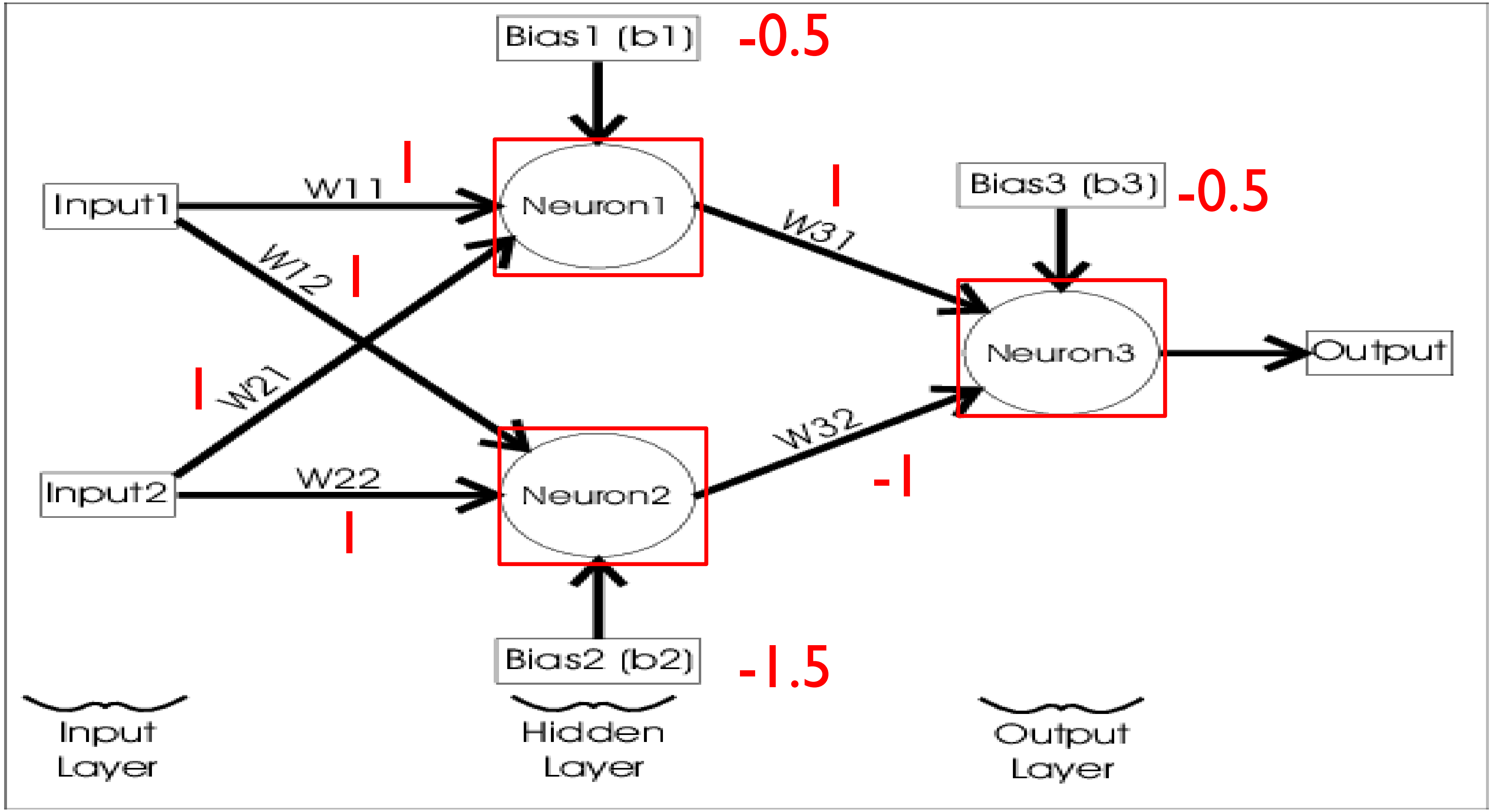
Non linearly separable

XOR-Gate with multilayer perceptron

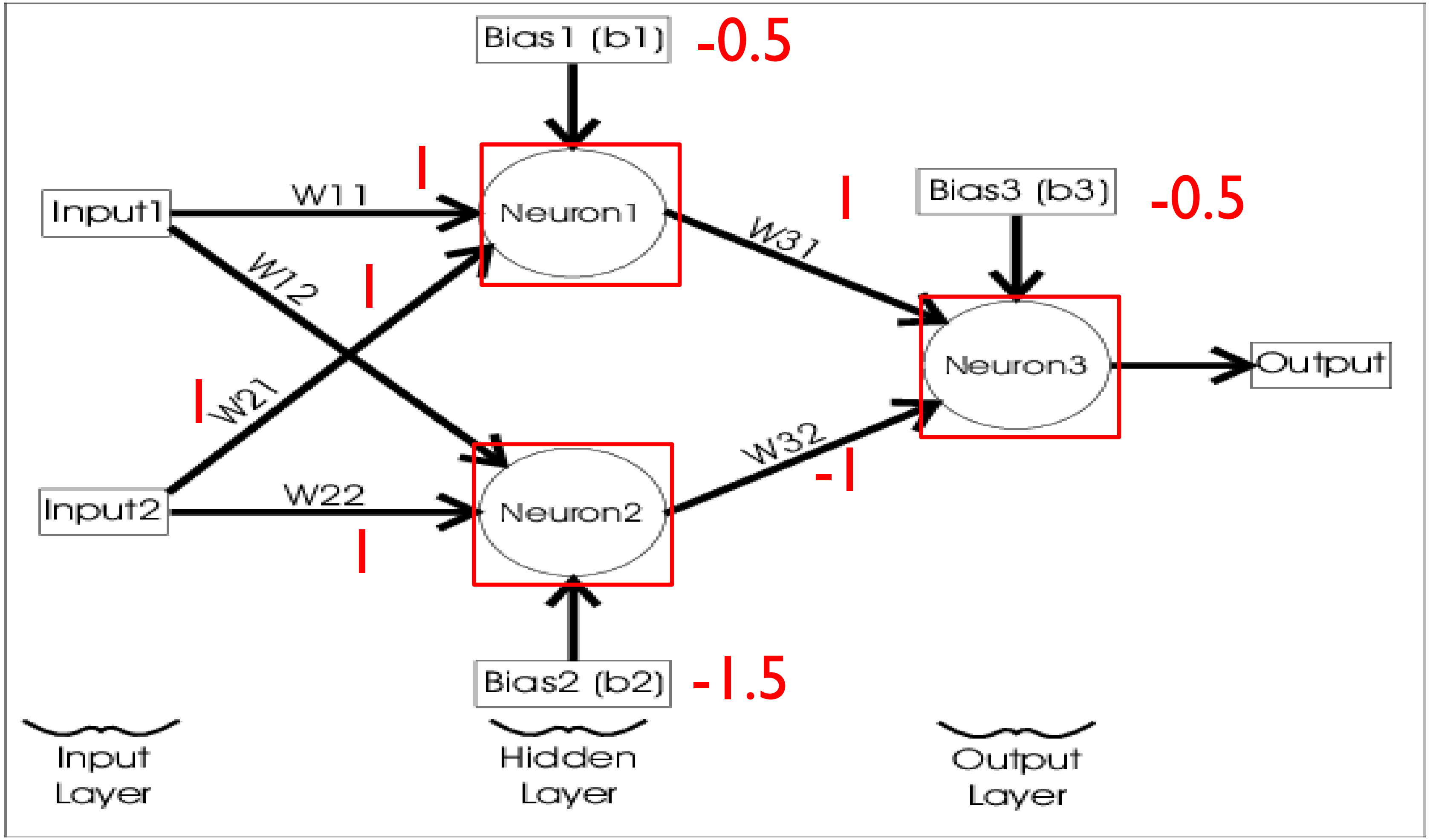


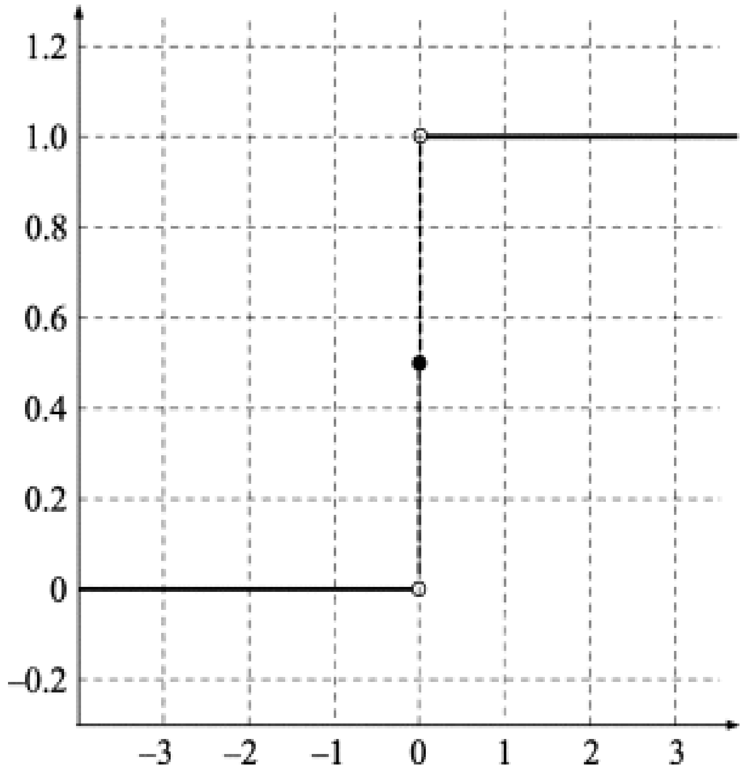
We want to get outputs as shown in the truth table.

XOR-gate with multilayer perceptron



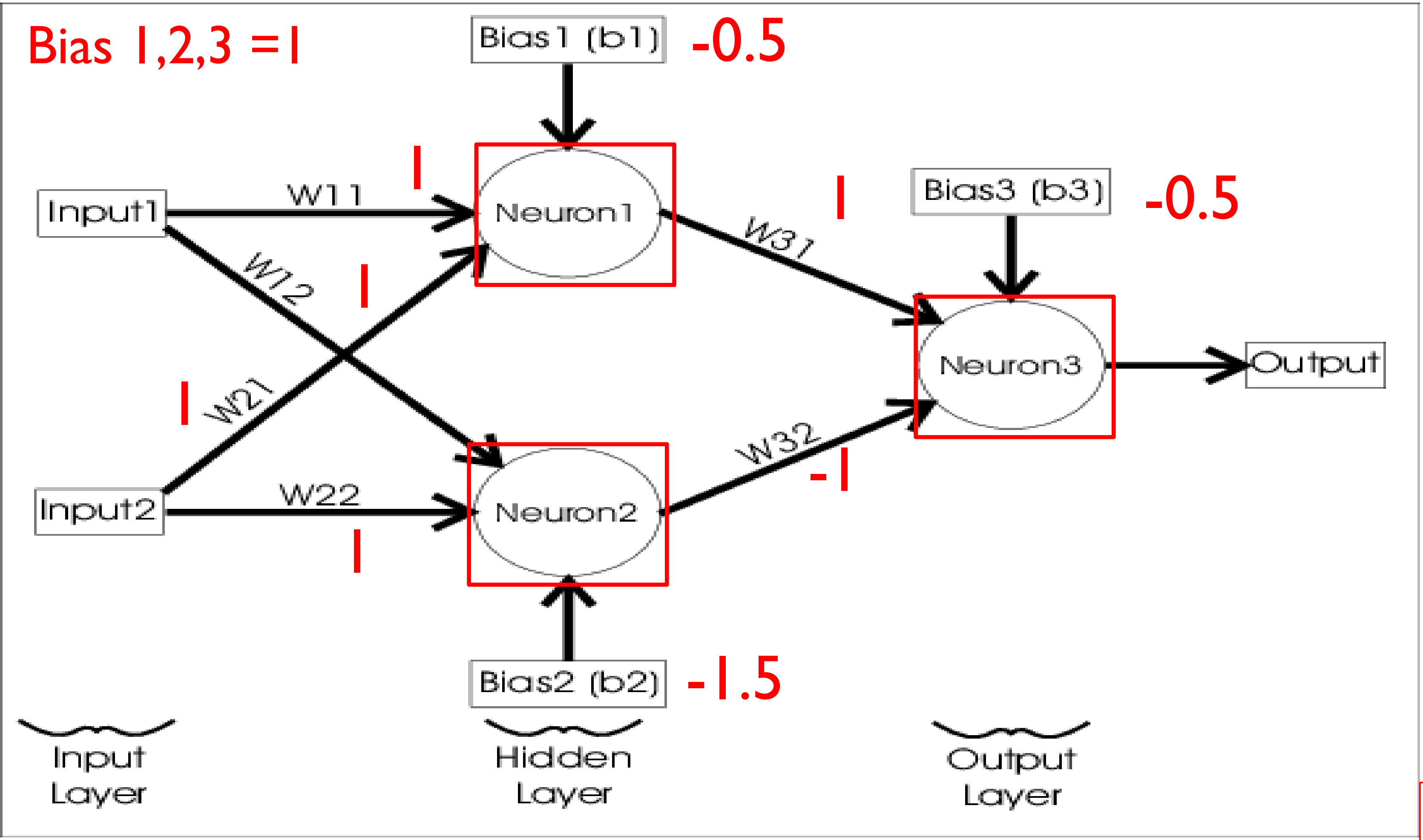
Calculation of XOR gate output





Step function

Calculation of XOR gate output



1- The XOR gate truth table says, if X1 = 0 and X2 =0 ,the output should be 0

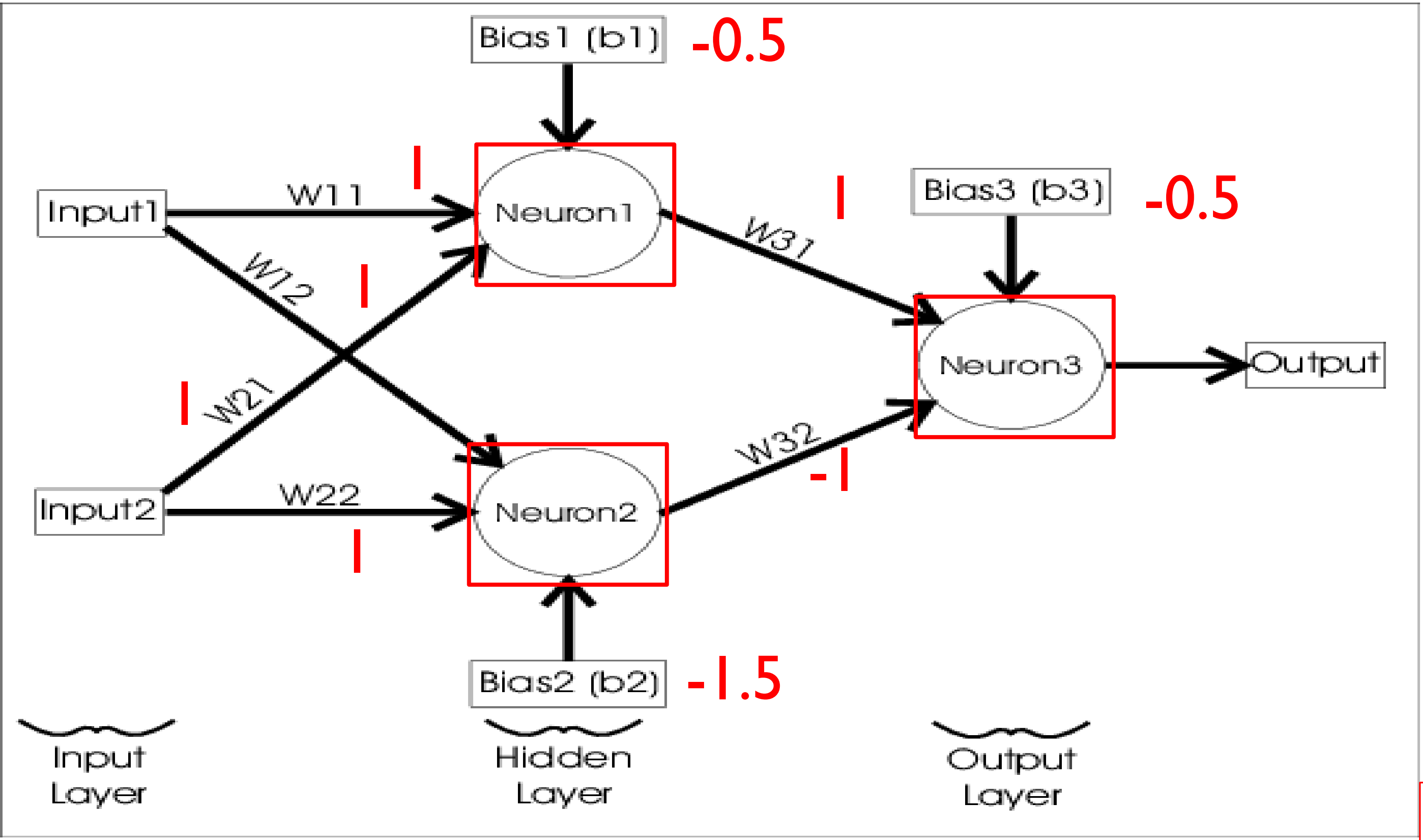
2- For hidden layer neuron Neuron 1=
*Input1 * w11 + input2 * w21 + bias1* b1=*
*0*1 + 0*1 + 1*(-0.5) = -0.5*
Stepfunction(-0.5)=0

3- For hidden layer neuron Neuron 2=
*Input1 * w12 + input2 * w22 + bias2* b2=*
*0*1 + 0*1 + 1*(-1.5) = -1.5*
Stepfunction(-1.5)=0

4- For Neuron 3=
*N1 * w31 + N2 * w32 + bias3* b3=*
*0*1 + 0*(-1) + 1*(-0.5) = -0.5*
Stepfunction(-0.5)=0

5- Matched with XOR truth table first row.

Calculation of XOR gate output



1- The XOR gate truth table says, if X1 = 0 and X2 =1 ,the output should be 0

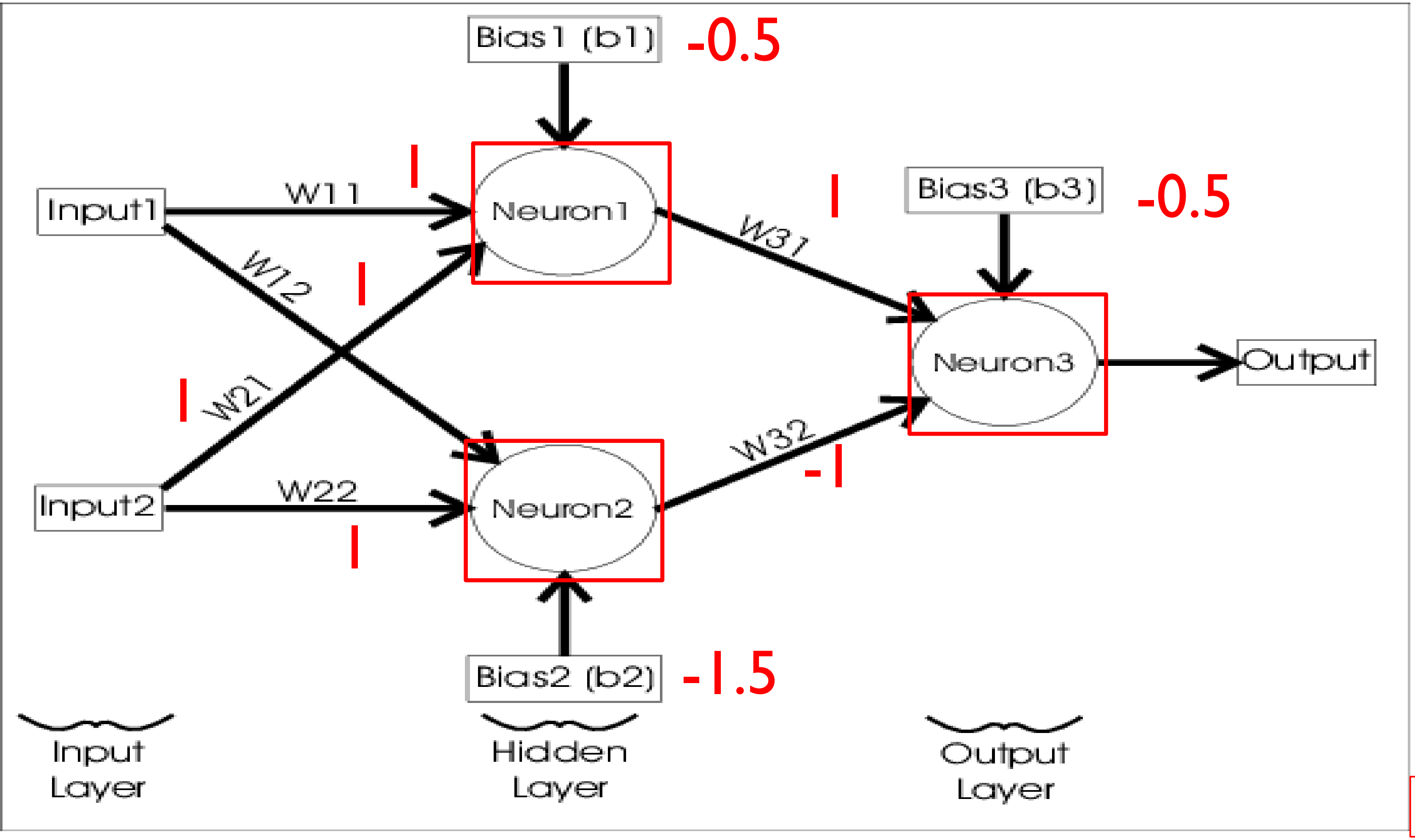
2- For hidden layer neuron Neuron 1=
*Input1 * w11 + input2 * w21 + bias1* b1=*
*0*1 + 1*1 + 1*(-0.5) = 0.5*
Stepfunction(0.5)=1

3- For hidden layer neuron Neuron 2=
*Input1 * w12 + input2 * w22 + bias2* b2=*
*0*1 + 1*1 + 1*(-1.5) = -0.5*
Stepfunction(-0.5)=0

4- For Neuron 3=
*N1 * w31 + N2 * w32 + bias3* b3=*
*1*1 + 0*(-1) + 1*(-0.5) = 0.5*
Stepfunction(0.5)=1

5- Matched with XOR truth table second row.

Calculation of XOR gate output



1- The XOR gate truth table says, if X1 = 1 and X2 =0 ,the output should be 0

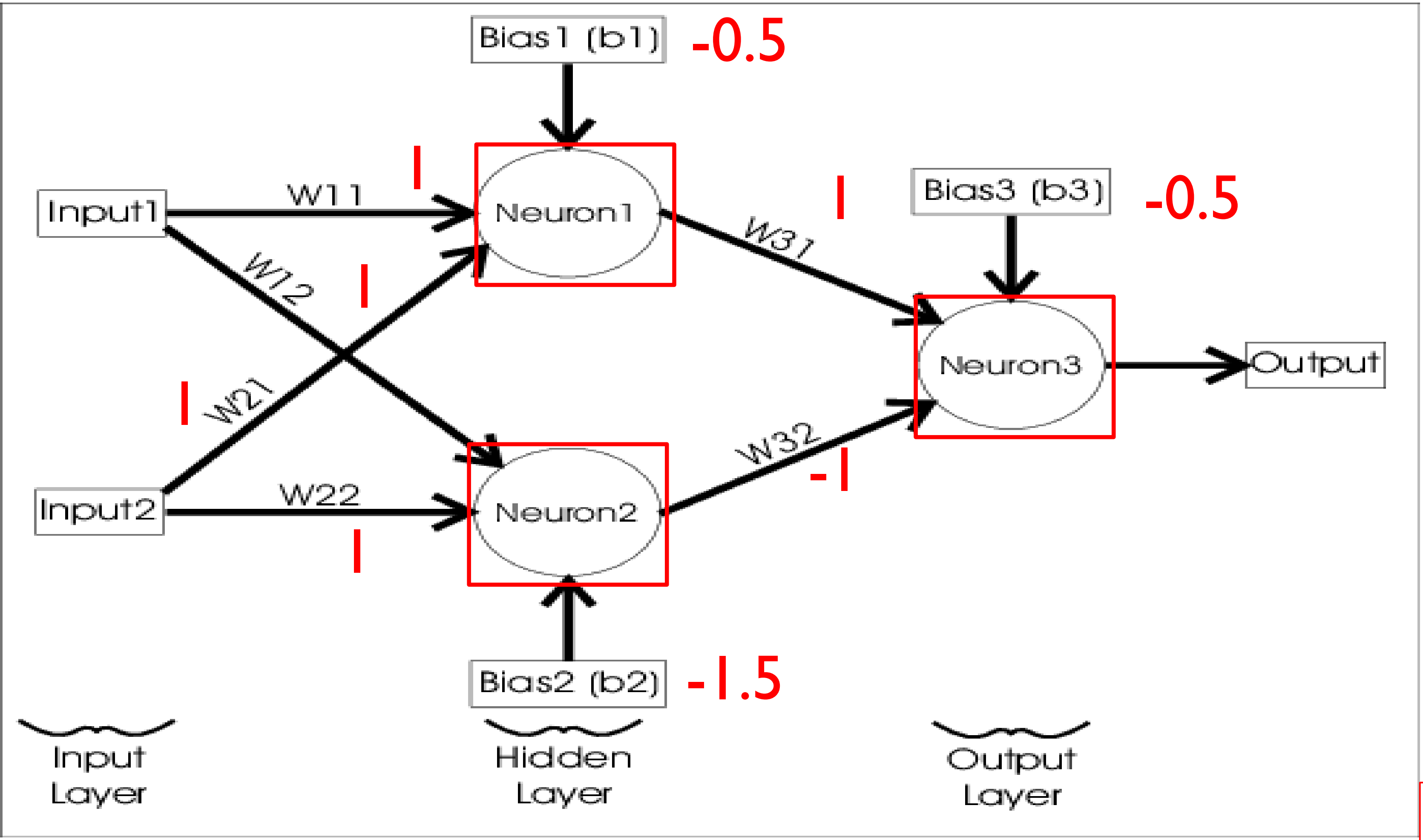
2- For hidden layer neuron Neuron 1=
*Input1 * w11 + input2 * w21 + bias1* b1=*
*1*1 + 0*1 + 1*(-0.5) = 0.5*
Stepfunction(0.5)=1

3- For hidden layer neuron Neuron 2=
*Input1 * w12 + input2 * w22 + bias2* b2=*
*1*1 + 0*1 + 1*(-1.5) = -0.5*
Stepfunction(-0.5)=0

4- For Neuron 3=
*N1 * w31 + N2 * w32 + bias3* b3=*
*1*1 + 0*(-1) + 1*(-0.5) = 0.5*
Stepfunction(0.5)=1

5- Matched with XOR truth table third row.

Calculation of XOR gate output



1- The XOR gate truth table says, if X1 = 1 and X2 =1 ,the output should be 0

*2- For hidden layer neuron Neuron 1= Input1 * w11 + input2 * w21 + bias1* b1= 1*1 + 1*1 + 1*(-0.5) = 1.5 Stepfunction(1.5)=1*

*3- For hidden layer neuron Neuron 2= Input1 * w12 + input2 * w22 + bias2* b2= 1*1 + 1*1 + 1*(-1.5) = 0.5 Stepfunction(0.5)=1*

*4- For Neuron 3= N1 * w31 + N2 * w32 + bias3* b3= 1*1 + 1*(-1) + 1*(-0.5) = -0.5 Stepfunction(-0.5)=0*

5- Matched with XOR truth table fourth row.