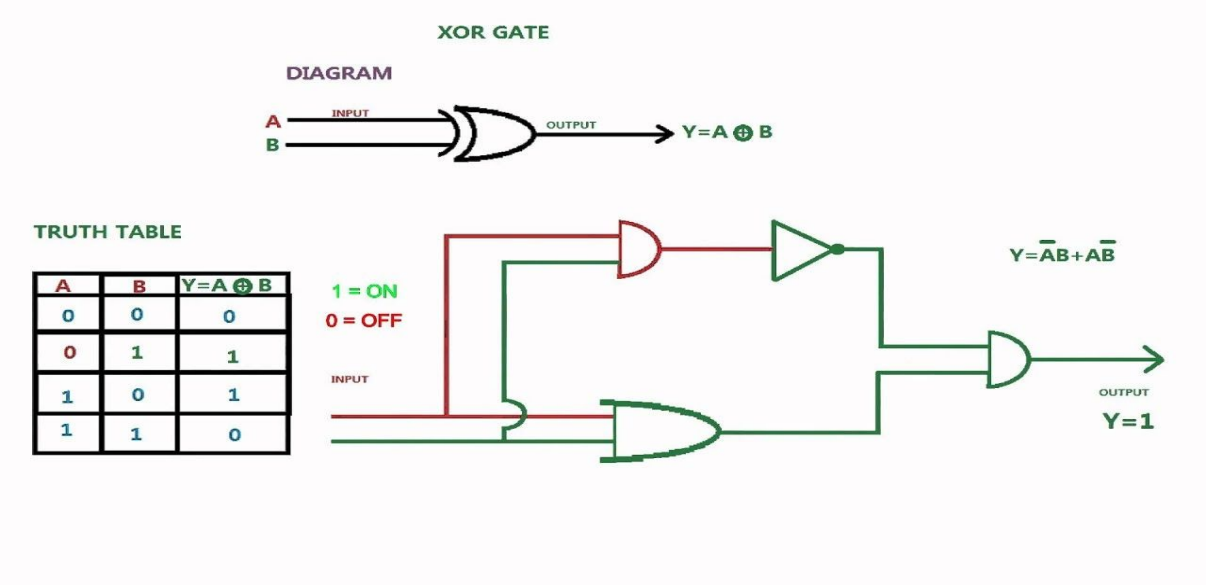
XOR-With-Numpy-Neural-Network

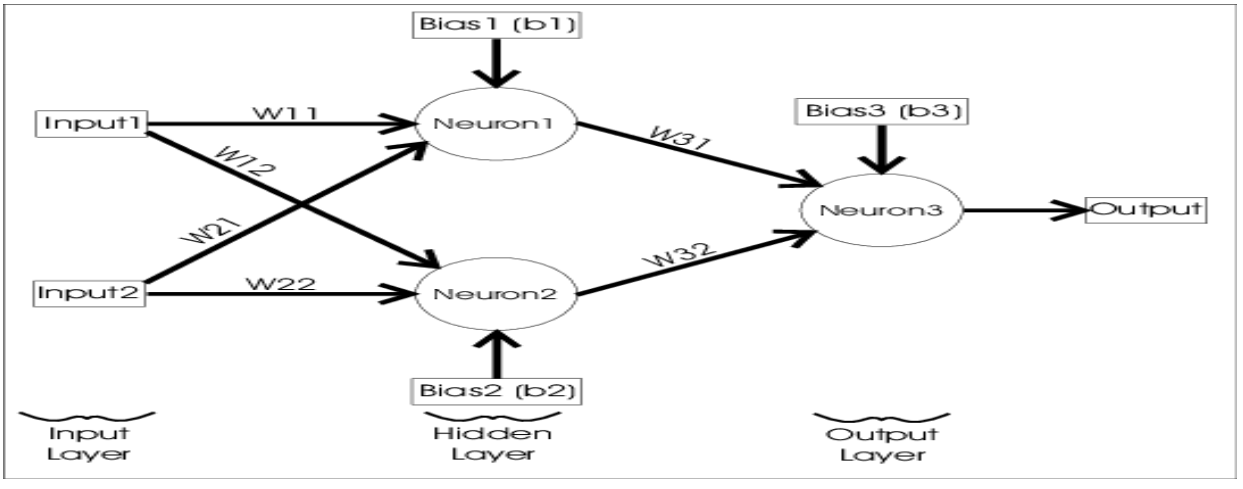
What is XOR Gate?

XOR gate (sometimes EOR, or EXOR and pronounced as Exclusive OR) is a digital logic gate that gives a true (1 or HIGH) output when the number of true inputs is odd. An XOR gate implements an exclusive or; that is, a true output results if one, and only one, of the inputs to the gate is true. If both inputs are false (0/LOW) or both are true, a false output results. XOR represents the inequality function, i.e., the output is true if the inputs are not alike otherwise the output is false



THE NEURAL NETWORK MODEL

The neural network needs to produce two different decision planes to linearly separate the input data based on the output patterns. This is achieved by using the concept of *hidden layers*. The neural network will consist of one input layer with two nodes (X1,X2); one hidden layer with two nodes (since two decision planes are needed); and one output layer with one node (Y). Hence, the neural network looks like this:



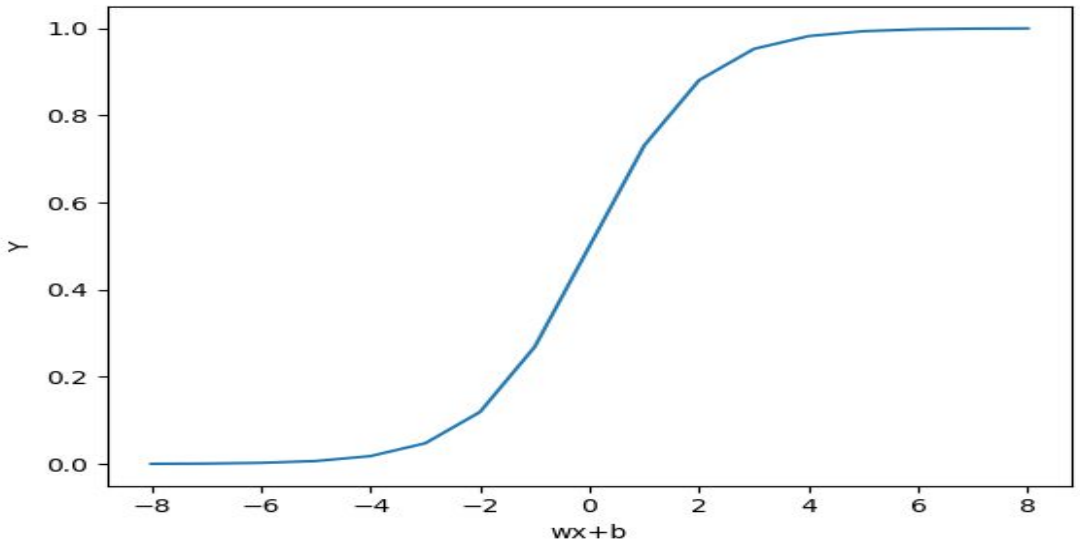
THE SIGMOID NEURON

To implement an XOR gate, I will be using a Sigmoid Neuron as nodes in the neural network. The characteristics of a Sigmoid Neuron are:

1. Can accept real values as input.

2. The value of the activation is equal to the weighted sum of its inputs i.e. ∑wi xi

3. The output of the sigmoid neuron is a function of the sigmoid function, which is also known as a logistic regression function. The sigmoid function is a continuous function which outputs values between 0 and 1:



‘

THE LEARNING ALGORITHM

The information of a neural network is stored in the interconnections between the neurons i.e. the weights. A neural network learns by updating its weights according to a learning algorithm that helps it converge to the expected output. The learning algorithm is a principled way of changing the weights and biases based on the loss function.

1. Initialize the weights and biases randomly.

2. Iterate over the data
   i. Compute the predicted output using the sigmoid function
   ii. Compute the loss using the square error loss function
   iii. W(new) = W(old) — α ∆W
   iv. B(new) = B(old) — α ∆B
3. Repeat until the error is minimal

This is a fairly simple learning algorithm consisting of only arithmetic operations to update the weights and biases. The algorithm can be divided into two parts: the *forward pass* and the *backward pass* also known as *"backpropagation."*

Working

The Neural Network has 3 layers: input layer with 2 neurons, hidden layer with 2 neurons, and output layer with 1 neuron. The first phase of neural network is training in which the possible values of (X,Y) goes as input [(0,0),(0,1),(1,0),(1,1)] with their respective answers [0,1,1,0]. The neural network modifies it's weights & biases according to the answers. After completion of training the neural network is ready to predict the answers. Lets show its step wise:

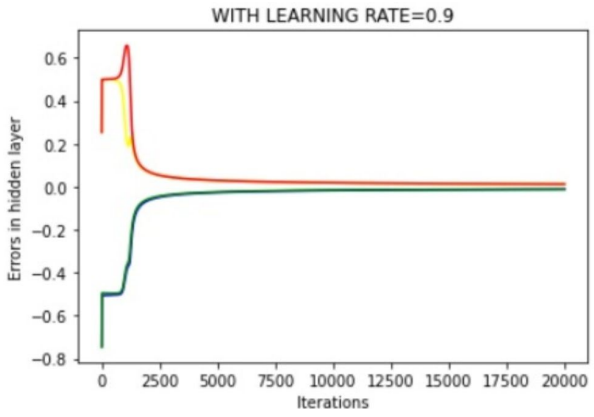
---

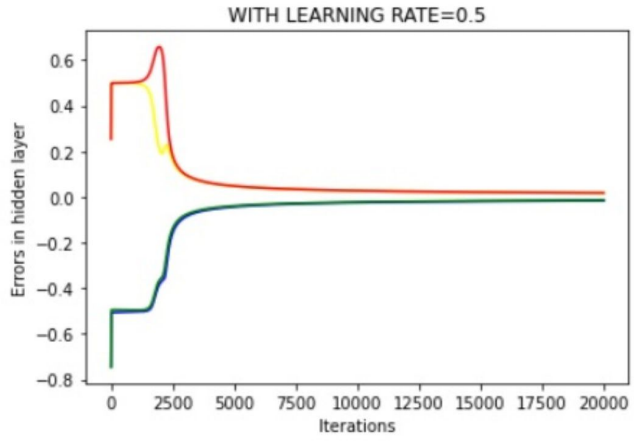
1) First we give XOR inputs for training data
2) We give randomize weights to all inputs and after that we take dot product of all inputs data to their respective weights and also add bias to it.
3) After that same process is done for hidden neurons nd finally we got our output.
4) After that we calculate errors using backpropogation
5) And update the weights using gradient descent
6) And repeating it for 20,000 times to make it more accurate

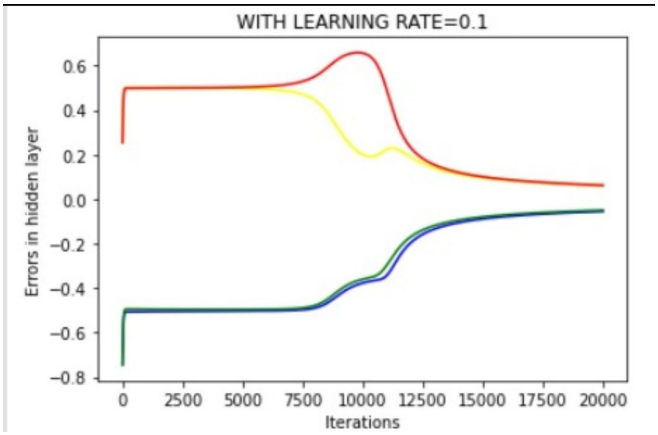
Complete Code:

https://github.com/nagarwal1510/1st-assignment-intelligence-agents/blob/master/neural%20network/xor%20neural%20network.ipynb

Graphs for Different Learning Rate:







It clearly shows that with high Learning rate, error decreases more rapidly.

Reference:

1) **The Nature of code: https://youtu.be/qWK7yW8oS0I**
2) **Wikipedia XOR Gate**
3) **https://towardsdatascience.com/implementing-the-xor-gate-using-backpropagation-in-neural-networks-c1f255b4f20d**