Lab 4. Perceptrons and Neural Networks

In this lab, we will practice building Perceptrons to calculate various logical operations. We will also build a simple neural network to extend the limits of the Perceptron.

# Part 1. Modeling a perceptron in TensorFlow.

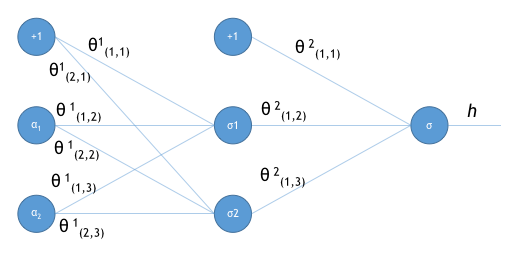
Objective: For our first objective, take the Perceptron example and extend it to support both the OR and the NAND boolean operations.

1. The current example only supports the AND operation. Extend the supplied code to support both the OR and the NAND boolean functions.
2. Re-write the perceptron as a class. Pass in the labels during class initialization. Hint: Look at both the

# Part 2. Model the XOR function

We’ve seen that the perceptron only supports a binary activation function (The Heaviside step function). Marvin Minsky et. al wrote a book in 1969 titled “*Perceptrons*” that proved that the single node Perceptron could not solve the XOR problem. This is because it is not possible to create a linear decision boundary that can pass through the XOR classes. Because of this, we will require a multi-layer perceptron with two nodes in the hidden layer to model the XOR function.

1. Create a multi-layer perceptron class that creates a simple neural network to model the XOR function. Use the TensorFlow sigmoid function tf.nn.sigmoid()for the activation function for the neural network. The diagram for this network should look something like this:



1. Create summaries for all operations and variables and display the graph with TensorBoard.

# Part 3. Creating a deep neural network

Objective: Use TensorFlow to model a deep neural network.

Recall that a Deep Neural Network is one with more than one hidden layer. Here we’re going to take our previous Lab that used a simple logistic regression model to classify our banking data and re-write it using a DNN.

1. Create a DNN. The DNN should have two hidden layers. The number of nodes in each layer is your choice, but should probably should be between 5 and 10 nodes deep.
2. As before, make sure that you include TensorBoard operations to be able to visualize your data.
3. The input data will remain the same. Additionally, the features that you chose in the earlier lab should not change.
4. Use the tf.nn.relu() method for the activation function for the hidden layers.
5. Compare the accuracy as well as the amount of time it takes for both the logistic regression model and the DNN model.