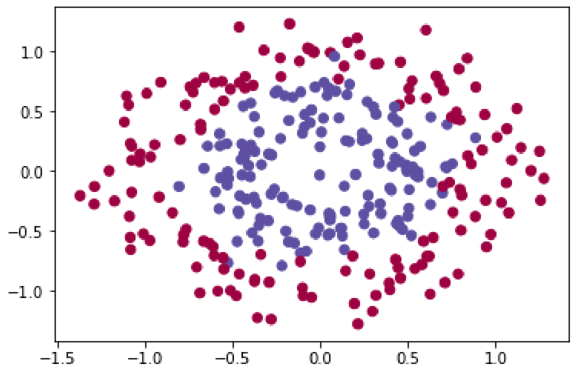
Neural Networks v's Logistic Regression

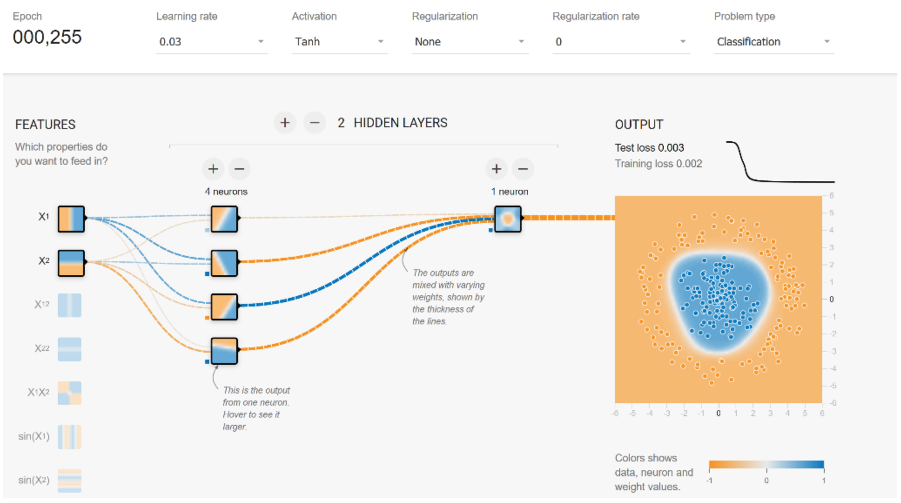
Question – Building a neural network with a single hidden layer.

In this exercise we are going to address a classification problem referred to as the circles problem. In the image depicted below we see a scatter plot of the dataset. The objective of the exercise is to build a neural network model that can differentiate between the two classes with a reasonable level of accuracy.



Step 1.

Before we move to the code go the TensorFlow playground. Select the circles dataset and configure the following neural network. The hidden layer should consist of four neurons and the output layer should consist of just one neuron. Select a learning rate of 0.3. Now build the model and try to reproduce what you see below.



Epoch 000,255
Learning rate 0.03
Activation Tanh
Regularization None
Regularization rate 0
Problem type Classification

FEATURES
Which properties do you want to feed in?

2 HIDDEN LAYERS

4 neurons

1 neuron

The outputs are mixed with varying weights, shown by the thickness of the lines.

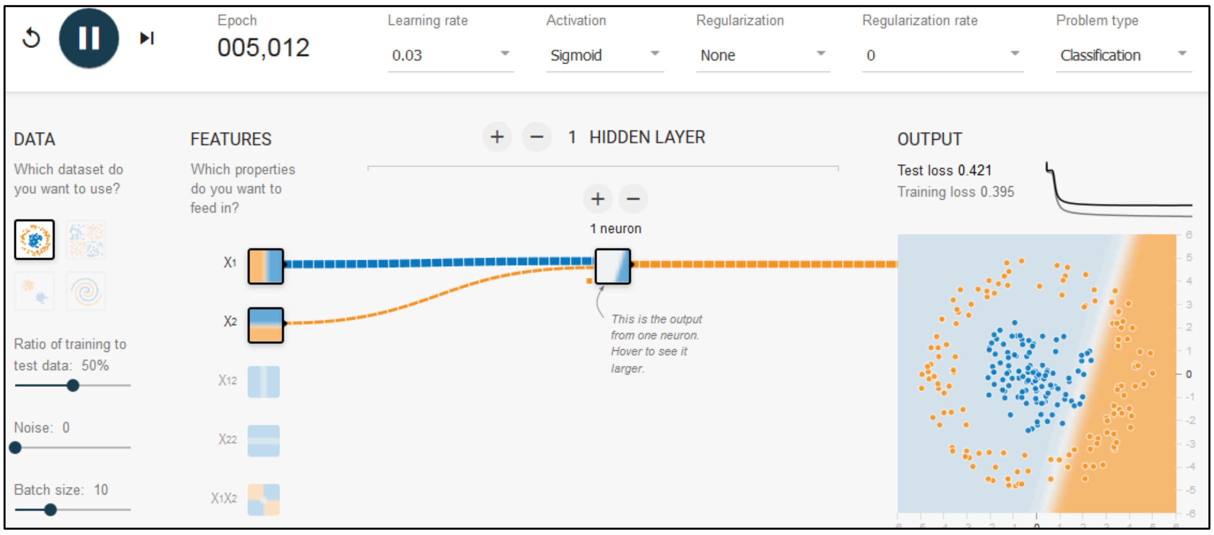
This is the output from one neuron. Hover to see it larger.

OUTPUT
Test loss 0.003
Training loss 0.002

Colors shows data, neuron and weight values.

Step 2.

Next reconfigure your network so that it contains just one neuron with a Sigmoid activation function (basic Logistic regression). The dataset is not linearly separable, so logistic regression doesn't perform well.


Epoch 005,012
Learning rate 0.03
Activation Sigmoid
Regularization None
Regularization rate 0
Problem type Classification

DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10

FEATURES
Which properties do you want to feed in?

1 HIDDEN LAYER
1 neuron
This is the output from one neuron. Hover to see it larger.

OUTPUT
Test loss 0.421
Training loss 0.395

Step 3. In this week's unit you will find a python file containing the code for this exercise. At the start of the exercise you will notice that we generate a noisy version of the circles dataset. See image of dataset above. Once the dataset is generated we run a logistic regression model using Sci-kit Learn and assess its accuracy. You should observe a poor level of accuracy of approx. 0.5.

Step 4: Next we build a neural network to try to solve this problem. The network will consist of two layers (similar to what we saw above). The hidden layer will consists of 4 neurons with Tanh activation functions and the output layer will consist of a single neuron with a Sigmoid activation function. The following are the general high level steps involved in building and training our neural network.

A) In the function runNeuralNetwork we set up the configuration of our neural network. Notice we specify the number of hidden neurons at 4, the number of output units as 1 and the learning rate as 0.3.

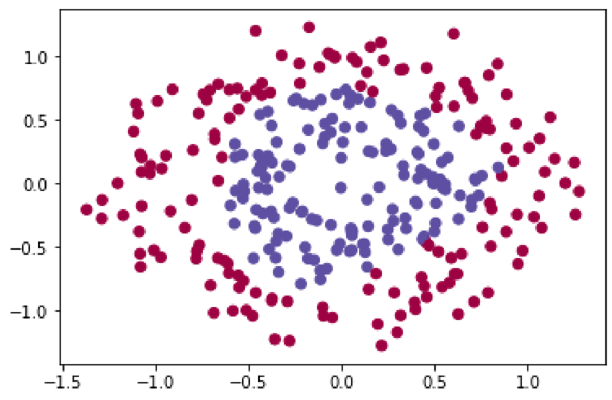
   Your first task involves initializing the weights and biases of the network to small random numbers. Remember the weigh matrices should be 2D arrays. Currently we specify the dimensions of the W1 and W2 matrices as 1*1. Your first task is to specify the correct shape of the W1 matrix (the matrix of weights feeding into the first layer of the network) and the W2 matrix (the matrix of weights feeding into the second layer). You should also specify the correct shape of b1 (the column vector containing the biases for the first layer) and b2 (containing the bias value for the second layer). Notice we can just set the biases to 0.

```
W1 = np.random.randn(1, 1) * 0.01
W2 = np.random.randn(1, 1) * 0.01
b1 = np.zeros(shape=(0, 1))
b2 = np.zeros(shape=(0, 1))
```

B) Next we call the gradient descent function and this iterates for 5000 iterations. Each time it iterates it calls the function **forwardPass(W1, b1, W2, b2, X).** This function takes in the weights and bias matrix for the hidden layer (W1 and b1 respectively) as well as the weights and bias matrix for the final layer (W2, and b2). You should implement this function using vector multiplications. You should calculate the output of the first layer of neurons $H^{[1]}$ and the output of the second layer of neurons $H^{[2]}$ and return both from the function.

C) In the gradient descent function please complete the gradient descent update rule (line 94-97).

D) Once you complete the code above you should obtain a level of accuracy of approximately 94%. Remember this is on the training set only. We have not tested it on a validation set for this problem. However, you should be able to see from the final scatter plot (which shows the classes predicted by the neural network that it is able to able to deal with this non-linear decision boundary). The predicted y values are illustrated in the scatter plot below.



1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0