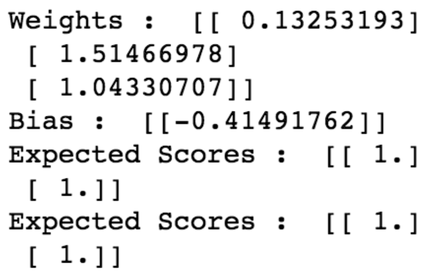
ANLY 535 TF_FFNN Report

– Xiaowei Wan

This report is to describe two basic feed forward neural networks(FFNNs) using TensorFlow deep learning in Python.
Given training data - inputs and outputs, predict color is Red or Blue(output) based on three color channels, RGB(inputs) values.
Output only have two values (red or blue), it is classification problem, will only use inputs and outputs, no hidden layers required. In order to train the model and predict the value, we need to add weights and bias into for neural network parameters.
Steps to predict this model:

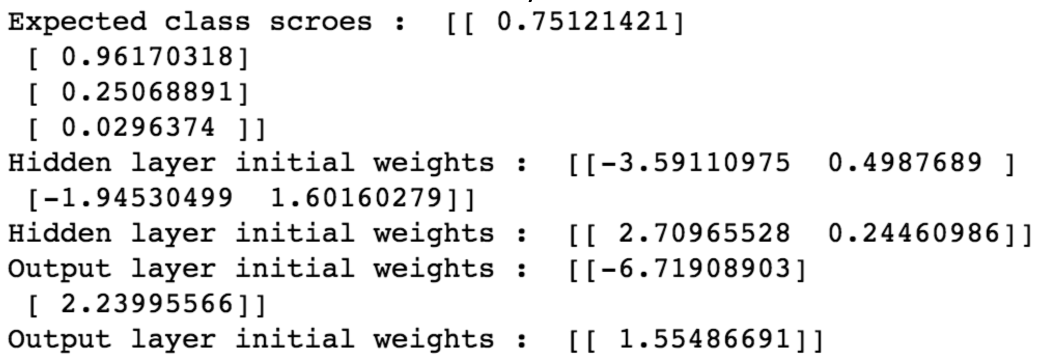
- Preparing training dataset (inputs and outputs), e.g. classify RED/BLUE -> [1, 0].
- Preparing neural network parameters (weights and bias) using TensorFlow Variables. Weights and bias need to be updated in order to get the best value for model. Set initial values.
- Preparing inputs of the activation function, the result is the summation of individual input-weight multiplications and plus bias. Here, code shows using TensorFlow matmul API call.
- Activation for output layer neuron, activation used to merge all inputs, weights and bias into a single value, here using sigmoid activation function, it represents the expected class score of the current input. In this case, sigmoid is sufficient.
- Predict error after the network being trained, we need to minimize the predict error during training, here we use gradient descent optimizer, Gradient descent will try to find the relationship between each parameter and the prediction error to know how each parameter affects the error, thus get the optimization value. Here using learning rate as 0.05, each step, it will add 0.05, and every time will decrease the error.
- Create TensorFlow Session and initializing the variables (here weights and bias), the training loop here will iterate 10000 times and at each iteration, the gradient Descent optimizer will generate new values for the parameters that decreases the error.
- Now that we have trained model, we need to test the Trained Neural Network. The result: (test: [[248, 80, 68],[0, 0, 255]] and [[255, 100, 50],[30, 50, 255]]



```
Weights :  [[ 0.13253193]
 [ 1.51466978]
 [ 1.04330707]]
Bias :  [[-0.41491762]]
Expected Scores :  [[ 1.]
 [ 1.]]
Expected Scores :  [[ 1.]
 [ 1.]]
```

After we get the accurate training model and results, we can apply the model to new inputs and predict the new outputs.

The second case added one more hidden layer. Below is the result of 2nd case:



```
Expected class scroes :  [[ 0.75121421]
 [ 0.96170318]
 [ 0.25068891]
 [ 0.0296374 ]]
Hidden layer initial weights :  [[-3.59110975  0.4987689 ]
 [-1.94530499  1.60160279]]
Hidden layer initial weights :  [[ 2.70965528  0.24460986]]
Output layer initial weights :  [[-6.71908903]
 [ 2.23995566]]
Output layer initial weights :  [[ 1.55486691]]
```