Tensorflow Multiple Layer Neural Network for Credit Card Fraud Detection

-- by Sijiang Du, May 2017

Summary

Multi-layer neoral network is trained by the data set of anonymized credit card transactions data. The raw data are labeled as two classes fraudulent or genuine

The challenge of this classification task is to identify a very small percentage of fraud cases in a large sample base. The total fraud transactions are less than 0.2% in total 284807 transaction records.

The tensorflow python program constructs a neural network model to detect the fraud transactions. The overall accuracy is measured. Credit card scoring and Kolmogorov Smirnov chart is being used to evaluate the model.

Preprocessing Credit Card Data Set

The following is the head of the csv file and one row data:

"Time","V1","V2","V3","V4","V5","V6","V7","V8","V9","V10","V11","V12","V13","V14","V15","V16","V17","V18","V19","V20","V21","V22","V23","V24","V25","V26","V27","V28","Amount","Class"

0,-1.3598071336738,-0.0727811733098497,2.53634673796914,1.37815522427443,-0.338320769942518,0.462387777762292,0.239598554061257,0.0986979012610507,0.363786969611213,0.0907941719789316,-0.551599533260813,-0.617800855762348,-0.991389847235408,-0.311169353699879,1.46817697209427,-0.470400525259478,0.207971241929242,0.025790580198591,0.403992960255733,0.251412098239705,-0.018306777944153,0.277837575558899,-0.110473910188767,0.0669280749146731,0.128539358273528,-0.189114843888824,0.133558376740387,-0.0210530534538215,149.62,"0"

The last column of the row is "Class", where "0" is for genuine (true) transaction and "1" is for fraud transaction. The python program replace the "Class" column with two fabricated columns to indicate the likelihood of being fraud or genuine:

"Fraud" column: 0 if "Class" is 0, 1 if "Class" 1

"True" column: 1 if "Class" is 0, 0 if "Class" 1

Therefore for the row value [0] -- > [0,1], and [1] -- > [1,0]

```
csv_r = csv.reader(data)
 for line in csv_r:
   if line[-1]=='0':
      line = line[:-1] + ['0','1']
   elif line[-1]=='1':
      line = line[:-1] + ['1','0']
```

The data set is divided to two parts: 80% for training set and 20% for testing set by "csv_partition_train_test(csv_file_name, 0.8)".

Tensorflow Input Pipeline

The 80% training data are furthermore separated to two sets and save in two files: one is for true and the other is for fraud. Each training step is done in a batch inputs. Half of the batch data are fraud and another half are genuine (true) transactions. E.g., for a batch size 128, the input tensor has 128 rows, 64 rows are for "true" set, and 64 are from "fraud" set. They are mixed in the batch randomly.

The tensorflow input pipeline organizes the feeding data by a list of files. And records can be repeatedly obtained for the number of epochs (or cycling forever) from an input queue after shuffling.

```
 with tf.name_scope('input_examples'):
    example_batch_train_true, label_batch_train_true = input_pipeline(tf.constant([FLAGS.data_dir+train_file_name_true]), round(batch_size/2))
    example_batch_train_fraud, label_batch_train_fraud = input_pipeline(tf.constant([FLAGS.data_dir+train_file_name_fraud]), batch_size-round(batch_size/2))
    example_batch_test_true, label_batch_test_true = input_pipeline(tf.constant([FLAGS.data_dir+test_file_name_true]), batch_size)
    example_batch_test_fraud, label_batch_test_fraud = input_pipeline(tf.constant([FLAGS.data_dir+test_file_name_fraud]), batch_size)
    example_batch_test_both, label_batch_test_both = input_pipeline(tf.constant([FLAGS.data_dir+test_file_name_both]), batch_size,1)
```

Multi-layer Graph

The are "layer_num" hidden layers plus one output layer. Each layer has "neuron_num" nodes. Activation function is tf.identity. The train( ) function is designed with parameters to specify the number of hidden layers and the number neurons in each layer. This is helpful to run experiments with variant range of sizes and depths.

E.g. Call "train(5,100)", means 5 hidden layers, each layer has 100 neurons. or, can put train(10, 2000) for a really large structure. The computation graph is constructed by while loop to generate multiple hidden layers.

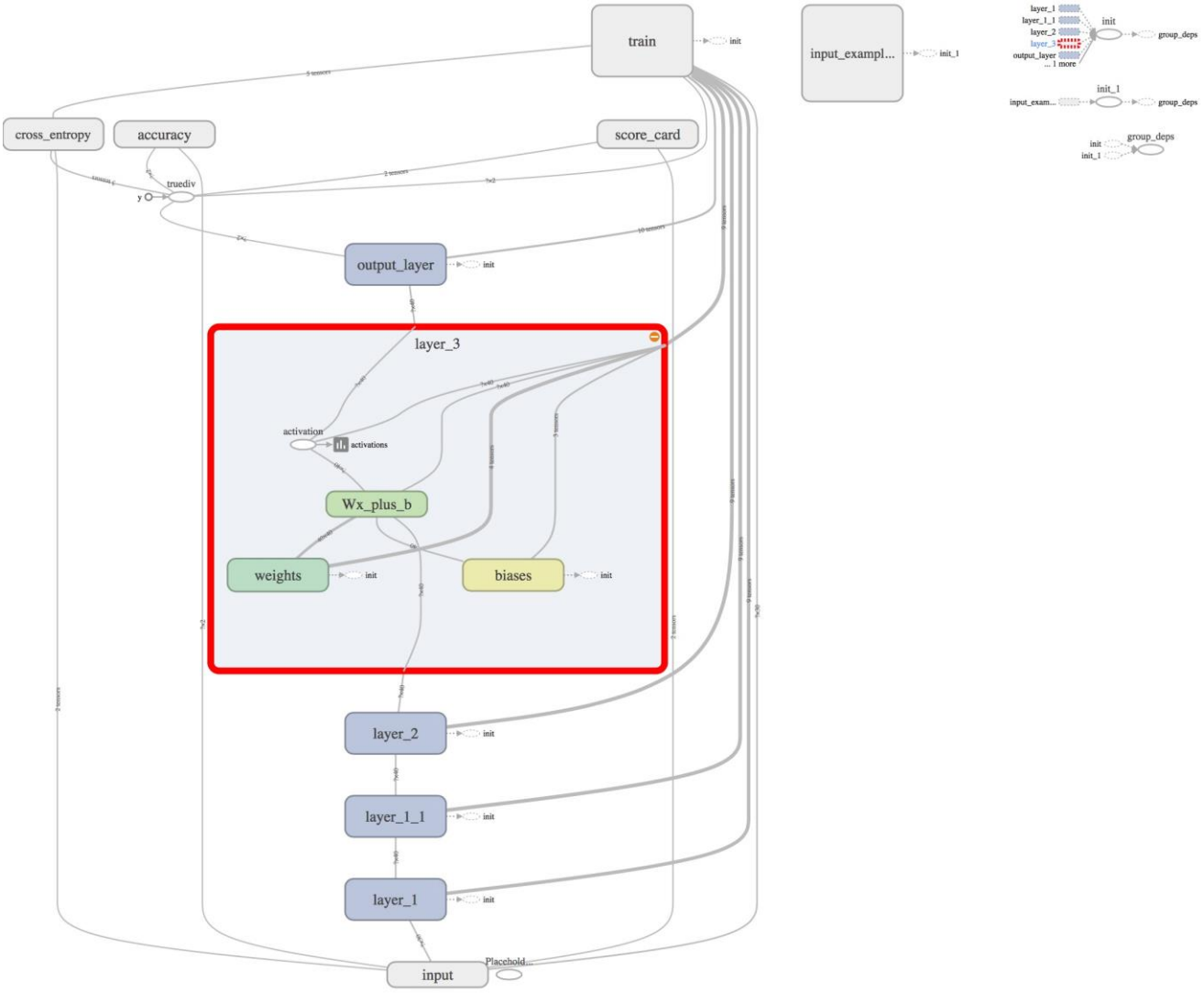

Figure 1. Neural Network Graphs Generated by Tensorboard

```
#Add layer_num layers of hidden layer

cur_layer = nn_layer(x, input_channel_num, neuron_num, 'layer_1', tf.identity, keep_prob)
```

```
for i in range(1,layer_num):
  cur_layer = nn_layer(cur_layer, neuron_num, neuron_num, 'layer_'+str(i), tf.identity, keep_prob)
# the last layer is the output layer
y = nn_layer(cur_layer, neuron_num, output_channel_num, 'output_layer', act=tf.identity)
# scale the activation down by the size of hidden layer
y = y/neuron_num
```

KS Graph to Evaluate Testing Results

Experiments show that 3 epochs are sufficient to settle the neuron model and achieve a stable accuracy. The testing result is evaluated by two measurements: overall accuracy (successful rate) and KS value by credit card scoring.

Using the KS value is a more appropriate approach to evaluate the performance because the percentage of fraud cases is too small. A high successful rate, for instance 99.0% does not necessarily prove a strong performance of fraud detection.

Construct the Kolmogorov Smirnov chart has these two parts:

- Scorecard: the trained model generate a scorecard for each test sample, which contains a score indicates how "creditable" this sample is. The scores of test cases are comparable and sortable. The test samples are sorted by their scorecard score values.
- Score: the value is computed from the two column values of the output layer. The first column value indicates the likelihood of being fraud and the second column value indicates the likelihood being genuine. The score is equal to the second column value minus the first column value.
- Cumulative Percentages: there two lines in the graph. The blue lines shows the cumulative percentages of genuine cases are predicted correctly. The red lines shows the cumulative percentages of fraud cases are predicted correctly.
- KS Value: max vertical distance in between blue and red lines.

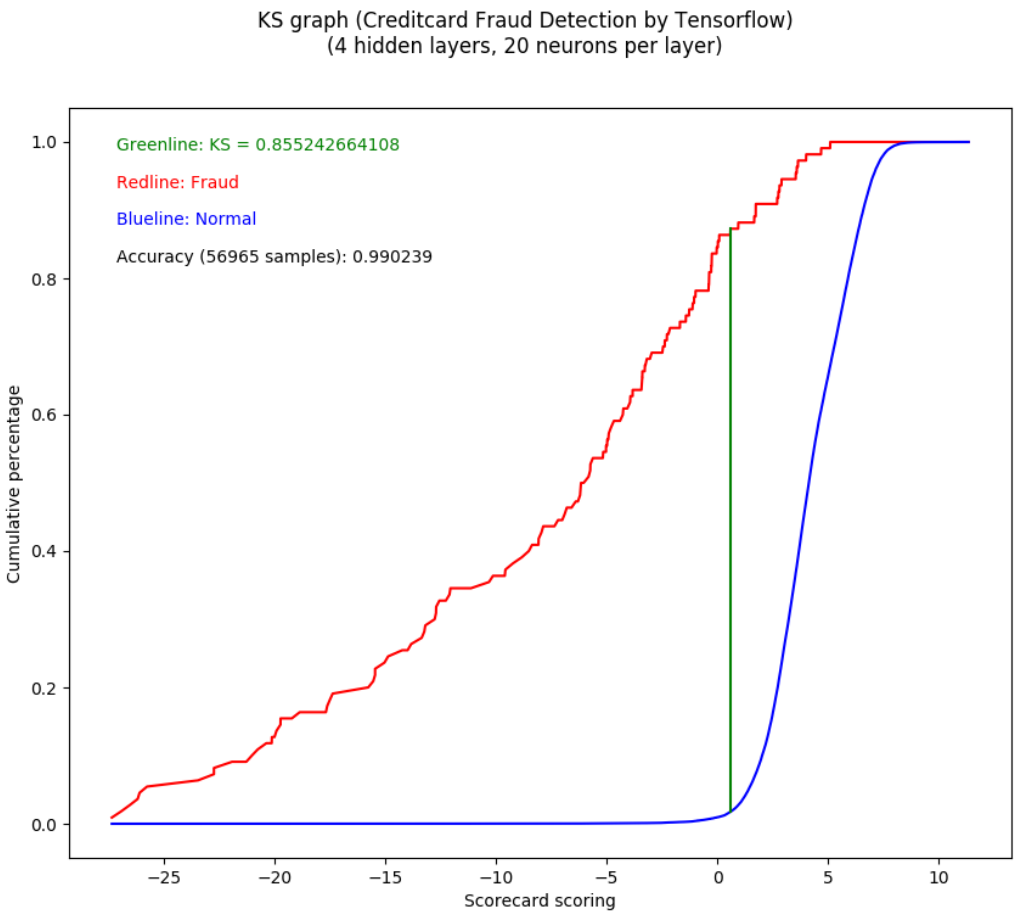

Figure 2. KS Graph shows the capability of fraud detection

The length of green line is the KS value

The overall successful rate of this 4 hidden layer neural network is 99.0%.Each hidden layer has 20 output channels (20 neurons).

The KS value (0.855 shows encouraging performance on the fraud detection.

The test are done for many sizes and depths model. However, larger and deeper models do not generate better result.

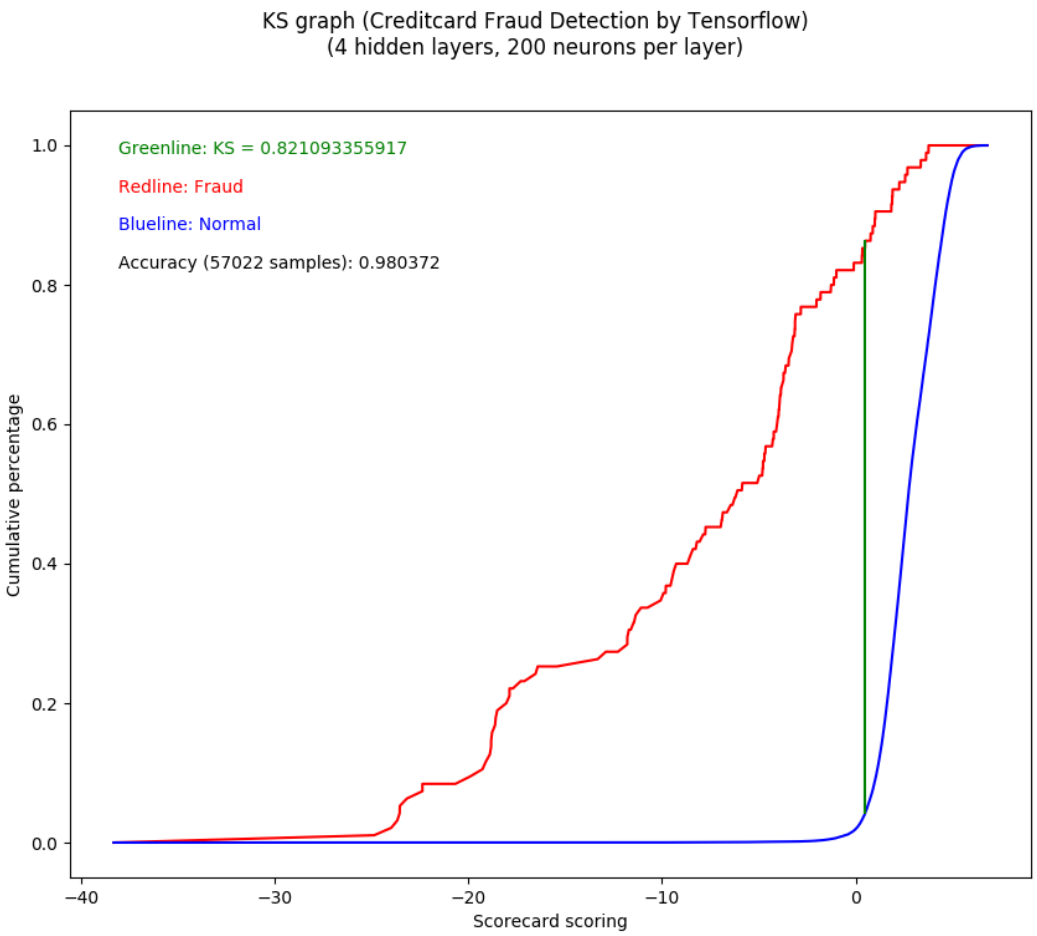

Figure 3. KS Graph show the result of 4 hidden layers, each layer has 200 neurons. It has almost same performance as the previous 4 layer 20 neuron model.

Amazing Result by 0-hidden Layer Test

Run "train(1,1)", the hidden lay becomes one channel data path direct to output. That means the input vector is able to generate output only by one step linear transform(weights matrix multiplication). The result has same performance as the 4 layer 200 neurons model and 4 layer 20 neurons model.

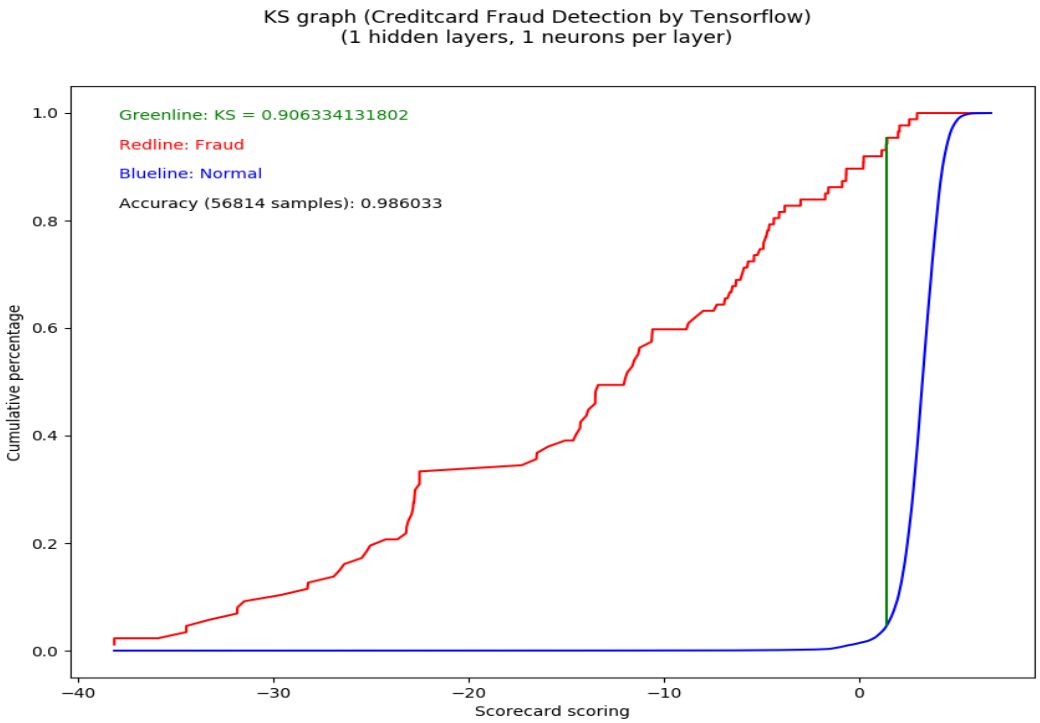

Figure 4. Test Result of One Layer Neural Network

30 channels input vector is directly map to output by one weight matrix. The result is as good as multi-layer models

Conclusion

The neuron network constructed by tensorflow is trainable by the credit card fraud detection data set. This indicates that there is obvious patterns to discover fraud or genuine cases

The training task by the data does not require the feature extraction from the data set.

The 0-hidden layer test result shows that the input data has a simple linear pattern to map to fraud or genuine classes

The KS graph is suitable to assess the performance.