Checkpoint 5: Neural Network
MSAI 339
Prof. Jennie Rogers
November 15, 2018

Group 5
Jack R
Quincia Hu
Ikhlas A

Purpose

The purpose of this checkpoint is to build a model for a prediction of some property from the TRR documents provided in the CPDP database.

Prediction From Proposal

Because the prediction from the proposal preceded all of the checkpoints, we decided to replace it with attempting to predict an event that could be a strong correlation to a filed allegation. Ultimately we chose to pursue “SubjectInjured” for our model. The reason is that the subject pertains to the person interacting with the officer. If the subject were to be injured in an interaction with an officer, bystanders could be more emotionally driven to file a complaint.

Binary Classification Neural Network

TensorFlow has a large variety of classifier settings in the neural network package. Because Subject Injured would be True or False, we chose to use the Binary Classification package. We first cleaned unnecessary or irrelevant data from the trr data, then one hot encoded the action response and charge table columns with the weather, lighting, and subject columns of the trr table. This ultimately provided 34 feature columns as input to our neural network. These columns were chosen to include any prejudice against a subject as well as any visual obscurities were caused due to the weather. A section of the image can be seen below.

	Indoor	Taser	FirearmUsed	OfficerInUniform	SubjectArmed	SubjectInjured	Lighting_DAYLIGHT	Lighting_GOOD ARTIFICIAL	Lighting_NIGHT
0	1	0	0	1	0	0	0	1	0
1	1	0	0	1	0	0	0	1	0
2	1	0	0	1	0	0	0	1	0
3	0	0	0	1	0	1	0	0	0
4	1	0	0	1	0	0	0	1	0
5	0	0	0	1	1	1	1	0	0
6	0	0	0	1	0	0	0	0	1
7	0	0	0	1	0	0	0	0	1
8	1	0	0	1	0	0	0	1	0
9	1	0	0	1	0	0	0	1	0
10	1	0	0	1	0	0	0	1	0
11	1	0	0	1	0	0	0	1	0

With this DataFrame stored in pandas, we used TensorFlow, Keras, and Sklearn to build the network to accept an input layer of 34 neurons. The model code can be seen below.

```
y = df['SubjectInjured']
x = df.drop(['SubjectInjured'],axis=1)
x_train = (x.iloc[:,:].values).astype('int32')
x_train = x_train.reshape(x_train.shape[0],x_train.shape[1])
y_train = (y.iloc[:].values).astype('int32')
y_train = y_train.reshape(y_train.shape[0],1)
x_train,x_val,y_train,y_val = train_test_split(x_train,y_train,test_size=0.30,random_state=42)
feature_count = x_train.shape[1]
label_count = y_train.shape[1]
epochs = 100
learning_rate = 0.1
hidden_layers = feature_count - 1
cost_history = np.empty(shape=[1],dtype=float)

X = tf.placeholder(tf.float32,[None,feature_count])
Y = tf.placeholder(tf.float32,[None,label_count])

is_training = tf.Variable(True,dtype=tf.bool)
initializer = tf.contrib.layers.xavier_initializer()
input_layer = tf.layers.dense(X,hidden_layers,activation=tf.nn.relu,kernel_initializer=initializer)

output_layer = tf.layers.dense(input_layer,label_count,activation=None)

cross_entropy = tf.nn.sigmoid_cross_entropy_with_logits(labels=Y,logits=output_layer)
predicted = tf.nn.sigmoid(output_layer)
correct_pred = tf.equal(tf.round(predicted),Y)
accuracy = tf.reduce_mean(tf.cast(correct_pred,tf.float32))
cost = tf.reduce_mean(cross_entropy)
optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate).minimize(cost)
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    for step in range(epochs + 1):
        sess.run(optimizer,feed_dict={X:x_train,Y:y_train})
        loss,_,acc = sess.run([cost,optimizer,accuracy], feed_dict={X:x_train,Y:y_train})
        cost_history = np.append(cost_history,acc)
```

Results

Training with the settings above showed a training accuracy of 73.16% and a testing accuracy of 73.22%. Because some features had imbalanced values, we decided to remove different classes of weather and lighting as well as a few other redundant binary attributes. This ultimately brought our results up to 74% training and 75% testing accuracies. An example output can be seen below.

```
Step:   100     Loss: 0.556     Acc:73.16%
Test Accuracy:  (0.73222226, array([[0.],
       [0.],
       [0.],
       ...,
       [0.],
       [0.],
       [0.]], dtype=float32))
```

Analysis and Reflection

We learned a few things from our results as well as this checkpoint. First, we noticed that more features does not mean better results. We were able to cut training time in half by removing binary features that were redundant or imbalanced. In fact, the accuracy increased when we removed the features. Pertaining to the data, we realized that there was potential in this prediction, especially if we spent more time on incorporating the crime and settlement data.

The feature we chose, Subject Injured, did not provide as beneficial results as hoped. That being said, the model was able to get 75% accuracy with no hidden layers or activation layers. In fact, when incorporating dropout layers and hidden layers, the accuracy decreased by over 20%. This makes us think that sometimes a simpler solution is better.

All of our feature experimentation surrounded the sub categories of the action responses, charges, weapon discharge tables. Our initial thought would be that using solely weapon discharge and action responses, we would be able to identify the force the officer uses against the subject as well as any interaction they have. This surmounted to 200 features and increased the training time by a large amount, and decreased accuracy to 50%. Being no more accurate than a coin flip, we decided to shrink the features to visual and subject description. These categorical features provided the best results with the binary classification model we used. In future work, we would develop the model further by testing more combinations of activation and hidden layers.