COSC4337_103-TF-Neural-Network

1 First Neurons

[31]:
```
import numpy as np
import tensorflow as tf
```

** Set Random Seeds for same results **

[32]:
```
np.random.seed(101)
tf.set_random_seed(101)
```

** Data Setup **

Setting Up some Random Data for Demonstration Purposes

[33]:
```
rand_a = np.random.uniform(0,100,(5,5))
rand_a
```

[33]:
```
array([[51.63986277, 57.06675869,  2.84742265, 17.15216562, 68.52769817],
       [83.38968626, 30.69662197, 89.36130797, 72.15438618, 18.99389542],
       [55.42275911, 35.2131954 , 18.18924027, 78.56017619, 96.54832224],
       [23.23536618,  8.35614337, 60.35484223, 72.89927573, 27.62388285],
       [68.53063288, 51.78674742,  4.84845374, 13.78692376, 18.69674261]])
```

[34]:
```
rand_b = np.random.uniform(0,100,(5,1))
rand_b
```

[34]:
```
array([[99.43179012],
       [52.06653967],
       [57.87895355],
       [73.48190583],
       [54.19617722]])
```

[35]:
```
# CONFIRM SAME  RANDOM NUMBERS (EXECUTE SEED IN SAME CELL!) Watch video for␣
↪explanation
np.random.seed(101)
rand_a = np.random.uniform(0,100,(5,5))
rand_b = np.random.uniform(0,100,(5,1))
```

1.0.1 Placeholders

[36]:
```
a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)
```

1.0.2 Operations

[37]:
```
add_op = a+b # tf.add(a,b)
mult_op = a*b #tf.multiply(a,b)
```

1.0.3 Running Sessions to create Graphs with Feed Dictionaries

[38]:
```
with tf.Session() as sess:
    add_result = sess.run(add_op,feed_dict={a:rand_a,b:rand_b})
    print(add_result)

    print('\n')

    mult_result = sess.run(mult_op,feed_dict={a:rand_a,b:rand_b})
    print(mult_result)
```

```
[[151.07166  156.49855  102.27921  116.58396  167.95949 ]
 [135.45622   82.76316  141.42784  124.22093   71.06043 ]
 [113.30171   93.09215   76.06819  136.43912  154.42728 ]
 [ 96.71727   81.83804  133.83675  146.38118  101.10579 ]
 [122.72681  105.982925  59.044632  67.9831    72.89292 ]]


[[5134.644   5674.25     283.12433 1705.4707  6813.8315 ]
 [4341.8125  1598.267   4652.734   3756.8293   988.94635]
 [3207.8113  2038.1029  1052.7742  4546.9805  5588.1157 ]
 [1707.379    614.02527 4434.989   5356.7773  2029.8555 ]
 [3714.0984  2806.6438   262.76764  747.19855 1013.292  ]]
```

1.1 Example Neural Network

[39]:
```
n_features = 10
n_dense_neurons = 3
```

[40]:
```
# Placeholder for x
x = tf.placeholder(tf.float32,(None,n_features))
```

[41]:
```
# Variables for w and b
b = tf.Variable(tf.zeros([n_dense_neurons]))
```

```
W = tf.Variable(tf.random_normal([n_features,n_dense_neurons]))
```

** Operation Activation Function **

[42]:
```
xW = tf.matmul(x,W)
```

[43]:
```
z = tf.add(xW,b)
```

[44]:
```
# tf.nn.relu() or tf.tanh()
a = tf.sigmoid(z)
```

** Variable Intializer! **

[45]:
```
init = tf.global_variables_initializer()
```

[46]:
```
with tf.Session() as sess:
    sess.run(init)

    layer_out = sess.run(a,feed_dict={x : np.random.random([1,n_features])})
```

[47]:
```
print(layer_out)
```

```
[[0.94208264 0.55631936 0.20048709]]
```

We still need to finish off this process with optimization! Let's learn how to do this next.

1.2 Full Network Example

Let's work on a regression example, we are trying to solve a very simple equation:

y = mx + b

y will be the y_labels and x is the x_data. We are trying to figure out the slope and the intercept for the line that best fits our data!

1.2.1 Artifical Data (Some Made Up Regression Data)

[48]:
```
x_data = np.linspace(0,10,10) + np.random.uniform(-1.5,1.5,10)
```

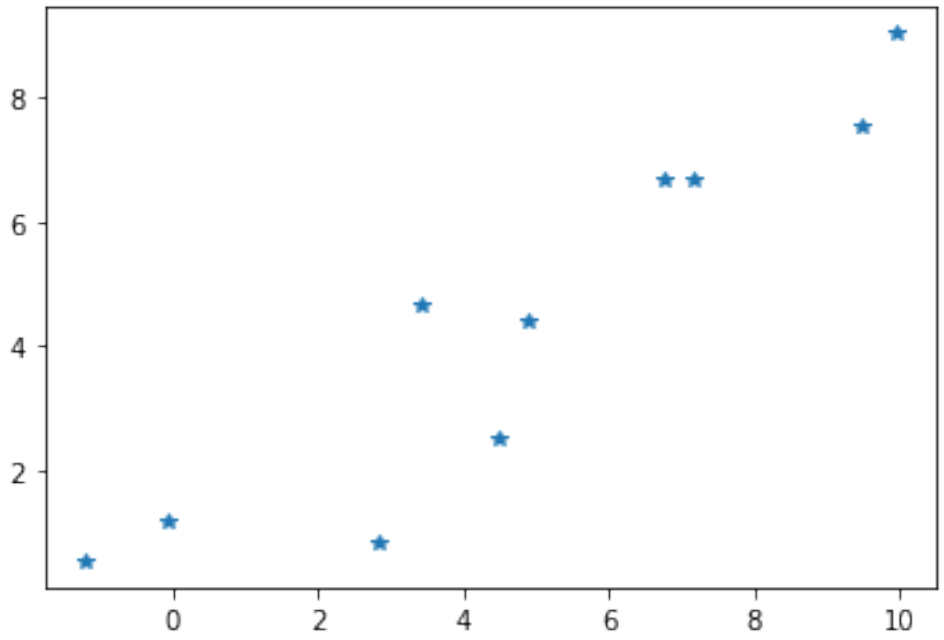
[49]:
```
x_data
```

[49]:
```
array([-1.20856056, -0.08034641,  2.82674411,  4.50477294,  3.42312535,
        4.88227319,  7.18414126,  6.77068715,  9.4930023 ,  9.96290567])
```

[50]:
```
y_label = np.linspace(0,10,10) + np.random.uniform(-1.5,1.5,10)
```

[51]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.plot(x_data,y_label,'*')
```

[51]: [<matplotlib.lines.Line2D at 0x1f5cff09cc8>]



** Variables **

[52]:
```
np.random.rand(2)
```

[52]: array([0.44236813, 0.87758732])

[53]:
```
m = tf.Variable(0.44)
b = tf.Variable(0.87)
```

1.2.2 Cost Function

[54]:
```
error = 0

for x,y in zip(x_data,y_label):

    y_hat = m*x + b  #Our predicted value

    error += (y-y_hat)**2 # The cost we want to minimize (we'll need to use an␣
↪optimization function for the minimization!)
```

1.2.3 Optimizer

[55]:
```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001)
train = optimizer.minimize(error)
```

1.2.4 Initialize Variables

[56]:
```
init = tf.global_variables_initializer()
```

1.2.5 Create Session and Run!

[57]:
```
with tf.Session() as sess:

    sess.run(init)

    epochs = 100

    for i in range(epochs):

        sess.run(train)


    # Fetch Back Results
    final_slope , final_intercept = sess.run([m,b])
```

[58]:
```
final_slope
```

[58]:
```
0.7535087
```

[59]:
```
final_intercept
```

[59]:
```
0.83729255
```

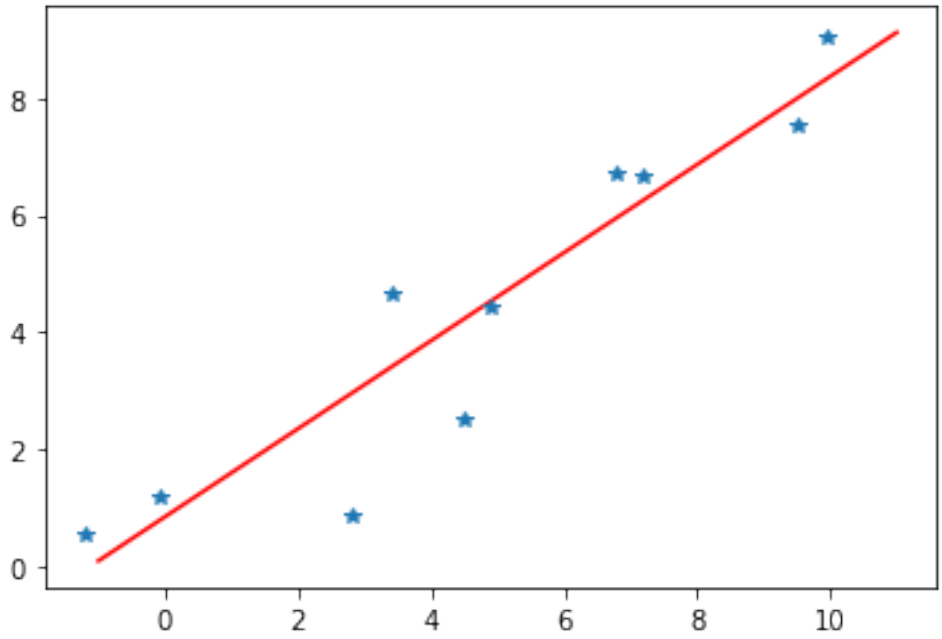
1.2.6 Evaluate Results

[60]:
```
x_test = np.linspace(-1,11,10)
y_pred_plot = final_slope*x_test + final_intercept

plt.plot(x_test,y_pred_plot,'r')

plt.plot(x_data,y_label,'*')
```

[60]:
```
[<matplotlib.lines.Line2D at 0x1f5d00c59c8>]
```



2 Great Job!

[ ]: