DEEP LEARNING LAB ASSIGNMENT – 1

Introduction: The given task is to implement Logistic Regression using Tensorflow and display the graph on Tensorboard using Python. Firstly Logistic Regression is similar to the linear regression but we use logistic regression for categorical data. Logistic Regression will be best fit if the predictors are either Qualitative or Quantitative and response is Qualitative. Logistic regression produces linear boundaries similar to the linear regression. No assumption on the predictors is made by logistic regression.

Objectives: In this task, we implement the logistic regression on the MNIST dataset which is a collection of the 28X28 pixel bounding black and white images. We implement the logistic regression on this dataset using tensorflow and write the graph on tensorboard using python. We change the hyperparameters to check the accuracy and costs of the model.

Approaches:

To implement the logistic regression to our dataset, i have chosen softmax function. We used softmax to build our model as we have multiple classes in our data. Softmax is a generalized version of logisitc regression used for multi-class classification. Then we use the reduce_mean function to obtain the cost for each epochs for the whole batch

Workflow:

1. First import the data from the tensorflow.examples.tutorials.mnist and read using read_input_data function.

2. We then use the placeholders for the graph on the tensorboard.

3. Construct our model using softmax function.

4. We use Cross entropy to minimize the error.

5. Declare the hyper parameters Learning rate and Epochs.

6. Using the Gradient Descent Optimizer to optimize the cost obtained.

7. We then train our model. In this section we use the epochs also can be referred as iterations. We create a batch and train our model based on that batch for iterations of the size of epochs.

8. We write the graph in the graph directory using the tf.summary.filewriter fucntion to get the graph from tensorboard.

9. Then we test the data using the tf.equal function and compare the real output with the predicted output. We use Accuracy function to get the accuracy of the obtained output in the final.

Dataset:

MNIST Dataset is a 28X28 pixel bounding box with black and white images. It is normalize data from the NSIT database. It has 60,000 training and 10,000 testing images. It has the gray scale images. It is widely used for training and testing purposes in the Machine Learning.

Parameters:

We used softmax in the data to construct our model. We get accuracy using the accuracy function. Batch_size is chosen low to get the cost minimized. Overfitting of the data should be avoided. We use one_hot for converting the binary classes to the categorical values.

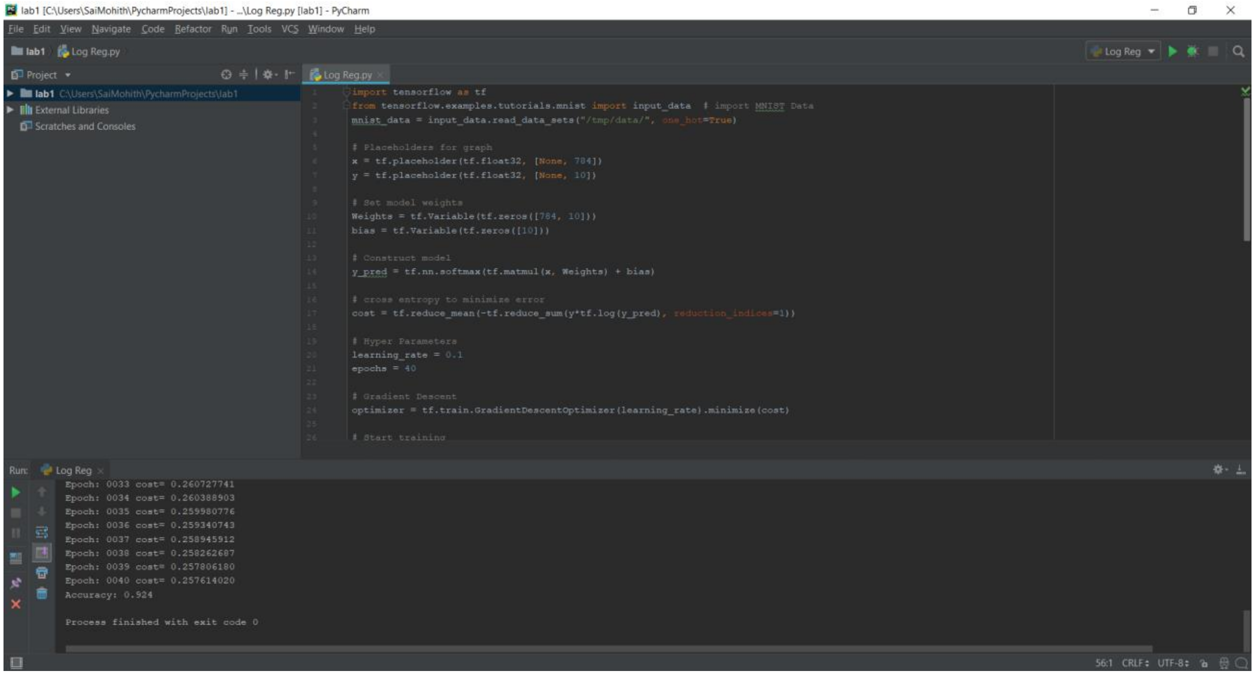
Evaluation:

Code:

```
   import tensorflow as tf
```

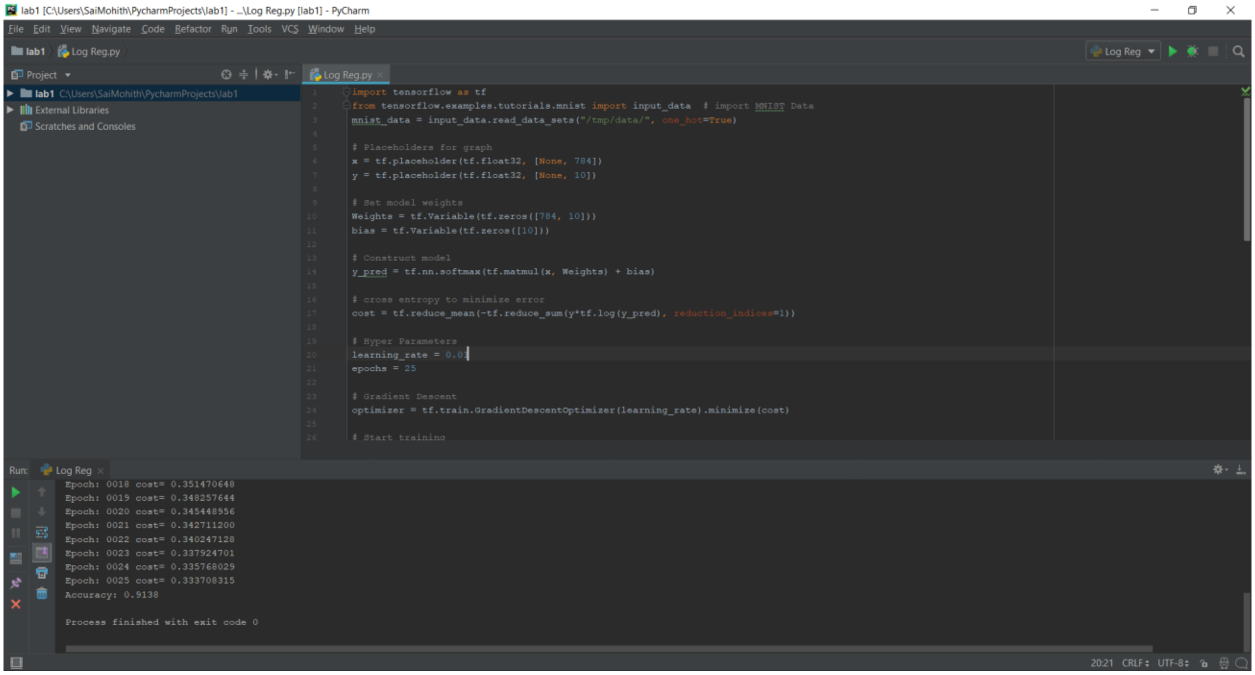
```
    from tensorflow.examples.tutorials.mnist import input_data  # import MNIST Data
    mnist_data = input_data.read_data_sets("/tmp/data/", one_hot=True)

    # Placeholders for graph
    x = tf.placeholder(tf.float32, [None, 784])
    y = tf.placeholder(tf.float32, [None, 10])

    # Set model weights
    Weights = tf.Variable(tf.zeros([784, 10]))
    bias = tf.Variable(tf.zeros([10]))

    # Construct model
    y_pred = tf.nn.softmax(tf.matmul(x, Weights) + bias)

    # cross entropy to minimize error
    cost = tf.reduce_mean(-tf.reduce_sum(y*tf.log(y_pred), reduction_indices=1))

# Hyper Parameters
learning_rate = 0.1
epochs = 40

# Gradient Descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

# Start training
with tf.Session() as sess:
    # Run the Global Initializer to initialize all the values
```

```
sess.run(tf.global_variables_initializer())
# Write the graph in the log_reg directory
writer = tf.summary.FileWriter('./graphs/log_reg', sess.graph)
# Start training cycle
for epoch in range(epochs):
    AverageCost = 0.
    Batch_tot = int(mnist_data.train.num_examples/100)
    # This loops over all batches
    for i in range(Batch_tot):
        batch_xs, batch_ys = mnist_data.train.next_batch(batch_size=100)
        # Run optimization op (backprop) and cost op (to get loss value)
        _, c = sess.run([optimizer, cost], feed_dict={x: batch_xs, y: batch_ys})
        # Compute average loss
        AverageCost += c / Batch_tot
    # Display logs per epoch step
    if (epoch+1) % 1 == 0:
        print("Epoch:", '%04d' % (epoch+1), "cost=", "{:.9f}".format(AverageCost))

writer.close()
w, b = sess.run([Weights, bias])

# Run the test model
correct_prediction = tf.equal(tf.argmax(y_pred, 1), tf.argmax(y, 1))

# Calculate accuracy
accu = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
print("Accuracy:", accu.eval({x: mnist_data.test.images, y: mnist_data.test.labels}))
```


lab1 [C:\Users\SaiMohith\PycharmProjects\lab1] - ...\Log Reg.py [lab1] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
lab1
Log Reg.py
Project
lab1 C:\Users\SaiMohith\PycharmProjects\lab1
External Libraries
Scratches and Consoles
Log Reg
Run:
56:1 CRLF UTF-8

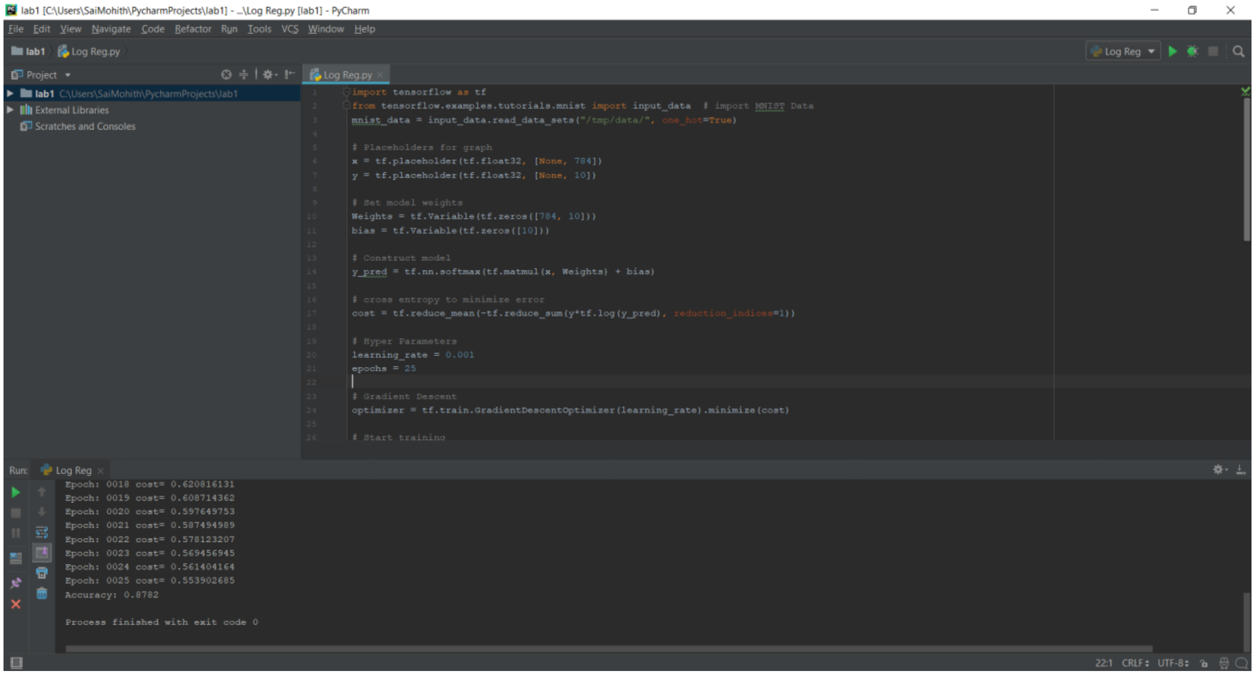
```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data  # import MNIST Data
mnist_data = input_data.read_data_sets("/tmp/data/", one_hot=True)

# Placeholders for graph
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])

# Set model weights
Weights = tf.Variable(tf.zeros([784, 10]))
bias = tf.Variable(tf.zeros([10]))

# Construct model
y_pred = tf.nn.softmax(tf.matmul(x, Weights) + bias)

# cross entropy to minimize error
cost = tf.reduce_mean(-tf.reduce_sum(y*tf.log(y_pred), reduction_indices=1))

# Hyper Parameters
learning_rate = 0.1
epochs = 40

# Gradient Descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

# Start training
```

```
Epoch: 0033 cost= 0.260727741
Epoch: 0034 cost= 0.260388903
Epoch: 0035 cost= 0.259980776
Epoch: 0036 cost= 0.259340743
Epoch: 0037 cost= 0.258945912
Epoch: 0038 cost= 0.258262687
Epoch: 0039 cost= 0.257806180
Epoch: 0040 cost= 0.257614020
Accuracy: 0.924

Process finished with exit code 0
```


lab1 [C:\Users\SaiMohith\PycharmProjects\lab1] - ...\Log Reg.py [lab1] - PyCharm
File Edit View Navigate Code Refactor Run Tools VCS Window Help
lab1
Log Reg.py
Project
lab1 C:\Users\SaiMohith\PycharmProjects\lab1
External Libraries
Scratches and Consoles
Log Reg
Run:
20:21 CRLF UTF-8

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data  # import MNIST Data
mnist_data = input_data.read_data_sets("/tmp/data/", one_hot=True)

# Placeholders for graph
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])

# Set model weights
Weights = tf.Variable(tf.zeros([784, 10]))
bias = tf.Variable(tf.zeros([10]))

# Construct model
y_pred = tf.nn.softmax(tf.matmul(x, Weights) + bias)

# cross entropy to minimize error
cost = tf.reduce_mean(-tf.reduce_sum(y*tf.log(y_pred), reduction_indices=1))

# Hyper Parameters
learning_rate = 0.01
epochs = 25

# Gradient Descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

# Start training
```

```
Epoch: 0018 cost= 0.351470648
Epoch: 0019 cost= 0.348257644
Epoch: 0020 cost= 0.345448956
Epoch: 0021 cost= 0.342711200
Epoch: 0022 cost= 0.340247128
Epoch: 0023 cost= 0.337924701
Epoch: 0024 cost= 0.335768029
Epoch: 0025 cost= 0.333708315
Accuracy: 0.9138

Process finished with exit code 0
```

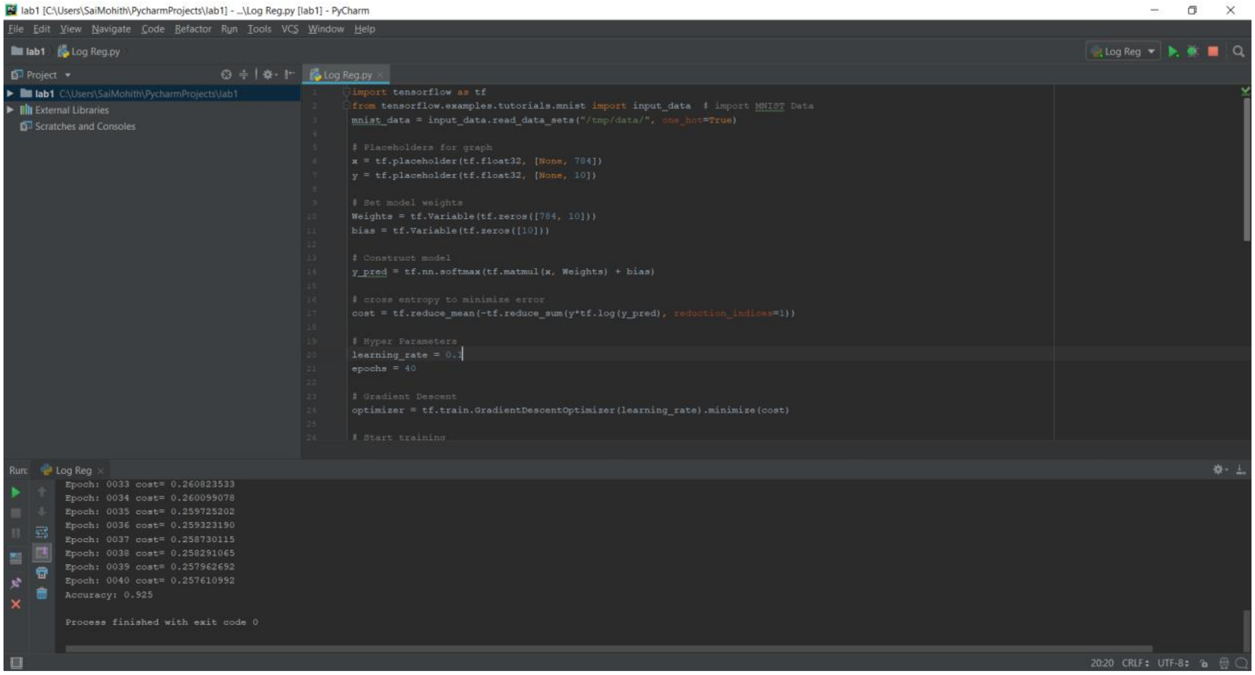


lab1 [C:\Users\SaiMohith\PycharmProjects\lab1] - ...\Log Reg.py [lab1] - PyCharm

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data  # import MNIST Data
mnist_data = input_data.read_data_sets("/tmp/data/", one_hot=True)

# Placeholders for graph
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])

# Set model weights
Weights = tf.Variable(tf.zeros([784, 10]))
bias = tf.Variable(tf.zeros([10]))

# Construct model
y_pred = tf.nn.softmax(tf.matmul(x, Weights) + bias)

# cross entropy to minimize error
cost = tf.reduce_mean(-tf.reduce_sum(y*tf.log(y_pred), reduction_indices=1))

# Hyper Parameters
learning_rate = 0.001
epochs = 25

# Gradient Descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

# Start training
```

```
Epoch: 0018 cost= 0.620816131
Epoch: 0019 cost= 0.608714362
Epoch: 0020 cost= 0.597649753
Epoch: 0021 cost= 0.587494989
Epoch: 0022 cost= 0.578123207
Epoch: 0023 cost= 0.569456945
Epoch: 0024 cost= 0.561404164
Epoch: 0025 cost= 0.553902685
Accuracy: 0.8782

Process finished with exit code 0
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help
lab1 Log Reg.py
Project
lab1 C:\Users\SaiMohith\PycharmProjects\lab1
External Libraries
Scratches and Consoles
Log Reg.py
Run: Log Reg
22:1 CRLF UTF-8

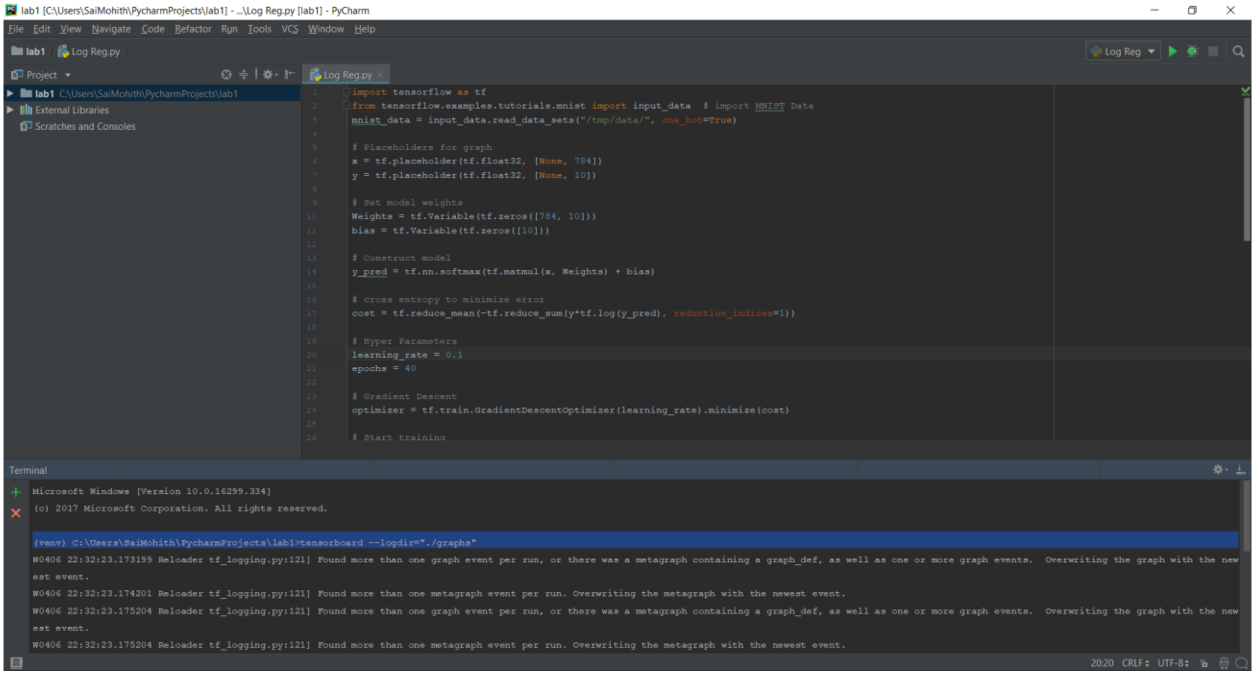


lab1 [C:\Users\SaiMohith\PycharmProjects\lab1] - ...\Log Reg.py [lab1] - PyCharm

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data  # import MNIST Data
mnist_data = input_data.read_data_sets("/tmp/data/", one_hot=True)

# Placeholders for graph
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])

# Set model weights
Weights = tf.Variable(tf.zeros([784, 10]))
bias = tf.Variable(tf.zeros([10]))

# Construct model
y_pred = tf.nn.softmax(tf.matmul(x, Weights) + bias)

# cross entropy to minimize error
cost = tf.reduce_mean(-tf.reduce_sum(y*tf.log(y_pred), reduction_indices=1))

# Hyper Parameters
learning_rate = 0.1
epochs = 40

# Gradient Descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

# Start training
```

```
Epoch: 0033 cost= 0.260823533
Epoch: 0034 cost= 0.260099078
Epoch: 0035 cost= 0.259725202
Epoch: 0036 cost= 0.259323190
Epoch: 0037 cost= 0.258730115
Epoch: 0038 cost= 0.258291065
Epoch: 0039 cost= 0.257962692
Epoch: 0040 cost= 0.257610992
Accuracy: 0.925

Process finished with exit code 0
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help
lab1 Log Reg.py
Project
lab1 C:\Users\SaiMohith\PycharmProjects\lab1
External Libraries
Scratches and Consoles
Log Reg.py
Run: Log Reg
20:20 CRLF UTF-8

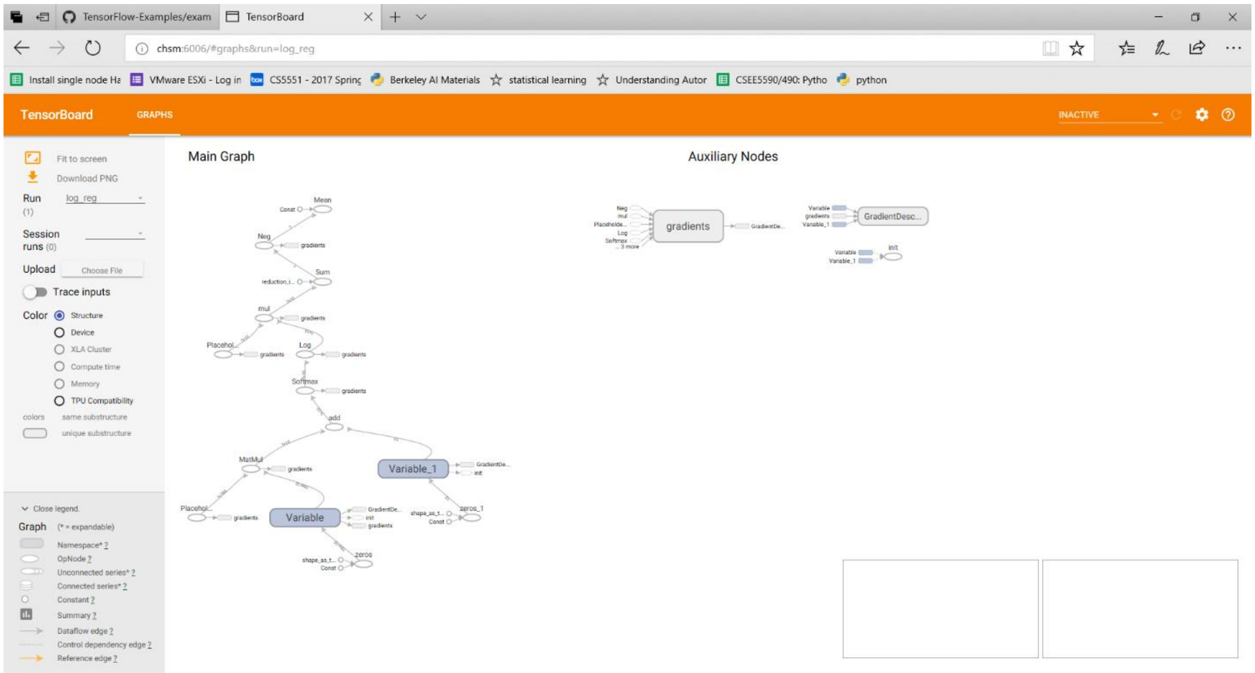


lab1 [C:\Users\SaiMohith\PycharmProjects\lab1] - ...\Log Reg.py [lab1] - PyCharm

```python
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data  # import MNIST Data
mnist_data = input_data.read_data_sets("/tmp/data/", one_hot=True)

# Placeholders for graph
x = tf.placeholder(tf.float32, [None, 784])
y = tf.placeholder(tf.float32, [None, 10])

# Set model weights
Weights = tf.Variable(tf.zeros([784, 10]))
bias = tf.Variable(tf.zeros([10]))

# Construct model
y_pred = tf.nn.softmax(tf.matmul(x, Weights) + bias)

# cross entropy to minimize error
cost = tf.reduce_mean(-tf.reduce_sum(y*tf.log(y_pred), reduction_indices=1))

# Hyper Parameters
learning_rate = 0.1
epochs = 40

# Gradient Descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

# Start training
```

```
Microsoft Windows [Version 10.0.16299.334]
(c) 2017 Microsoft Corporation. All rights reserved.

(venv) C:\Users\SaiMohith\PycharmProjects\lab1>tensorboard --logdir="./graphs"
W0406 22:32:23.173199 Reloader tf_logging.py:121] Found more than one graph event per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  Overwriting the graph with the new
est event.
W0406 22:32:23.174201 Reloader tf_logging.py:121] Found more than one metagraph event per run. Overwriting the metagraph with the newest event.
W0406 22:32:23.175204 Reloader tf_logging.py:121] Found more than one graph event per run, or there was a metagraph containing a graph_def, as well as one or more graph events.  Overwriting the graph with the new
est event.
W0406 22:32:23.175204 Reloader tf_logging.py:121] Found more than one metagraph event per run. Overwriting the metagraph with the newest event.
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help
lab1
Log Reg.py
Log Reg
Project
lab1 C:\Users\SaiMohith\PycharmProjects\lab1
External Libraries
Scratches and Consoles
Terminal
20:20 CRLF UTF-8



chsm:6006/#graphs&run=log_reg

First one shows the accuracy of 92.4 when learning rate is 0.1 but as we decrease the learning rate, we see the accuracy drops to 91.3 and 87.82. We see the accuracy is high when the learning rate is high and epochs are high. The accuracy falls down as the learning rate and epochs decrease. Learning rate have the major impact on the accuracy.

Conclusion:

Therefore the accuracy is seems to be 92.4% when the learning rate is 0.1. This seems to be a good fit the data.

References:

https://en.wikipedia.org/wiki/MNIST_database

https://www.kdnuggets.com/2016/02/scikit-flow-easy-deep-learning-tensorflow-scikit-learn.html

https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/

https://github.com/niyamatalmass/Logistic-regression-using-iris-and-mnist-data-set/blob/master/mnist_logistic_regression.ipynb

https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard

https://www.tensorflow.org/api_docs/python/tf/nn/softmax