Introduction

Logistic regression on MNIST dataset using tensorflow. MNIST dataset contain handwritten number. we will use logistic regression on this dataset to train the machine and then predict the output with test data. here the data will be image of size 28*28 which is converted into gray scale. we are using small size image as we will have the data which is important to train machine as well as we will reduce the time required to train the machine.

Objectives

Objective of this program is to train the machine so it can identify the hand written number(single digits).

Approaches/Methods

We will be using logistic regression to train the machine.

Model:

pred = tf.nn.softmax(tf.matmul(x, W) + b)

softmax method will perform the following operation on it.

softmax = tf.exp(logits) / tf.reduce_sum(tf.exp(logits), dim)

dim – it tell the dimension on which softmax would be performed on. The default value to this argument is -1.

Logits – it is a non empty tensor.it can be only from this types—float64,float32,half.

Workflow

1.import data from tensorflow library example

    the data is releated to hand writen numbers

2. setting up the parameters

3.creating placeholder for input

    1. mnist data image of shape 28*28=784

    2. the result 0-9 digits recognition

4. Set model weights

5. Construct model

    tf.nn.softmax(tf.matmul(x, W) + b)

6. Minimize error using cross entropy

7. Gradient Descent with learning rate .01

8.Initialize the variables (i.e. assign their default value)

9. Start training

    1. Run the initializer

    2. Training cycle

    3.Loop over all batches (batch size=100)

    4. Run optimization op (backprop) and cost op (to get loss value)

    5. Compute average loss

    6. Display logs per epoch step

10. Test model

11. Calculate accuracy

Datasets

MNIST dataset contain handwritten number. we will use logistic regression on this dataset to train the machine and then predict the output with test data. here the data will be image of size 28*28 which is converted into gray scale.

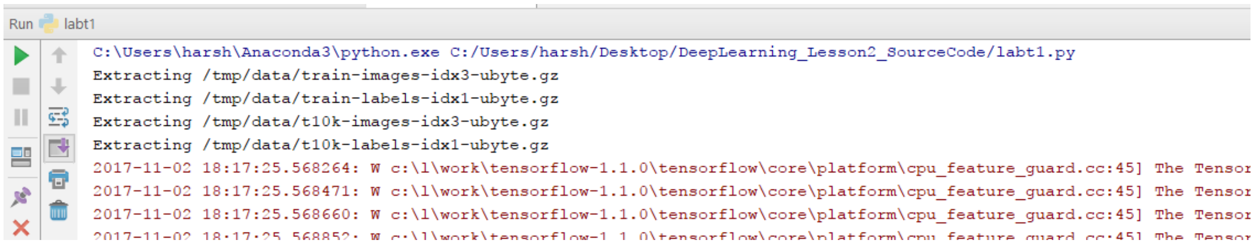
Here the data contain training set of 60000 example.
And test set of 10000 example.

****

Test_x=t10k-images-idx3-ubyte.gz --- it contain test set images of size 1648 kbytes.
test_y=t10k-labels-idx1-ubyte.gz ---- it contain test set labels of size 4 kbytes.
****

Train_x=train-images-idx3-ubyte.gz --- it contain training set images of size 9912 kbytes.
train_y=train-labels-idx1-ubyte.gz ---it contain training set labels of size 28 kbytes.

data initialization:



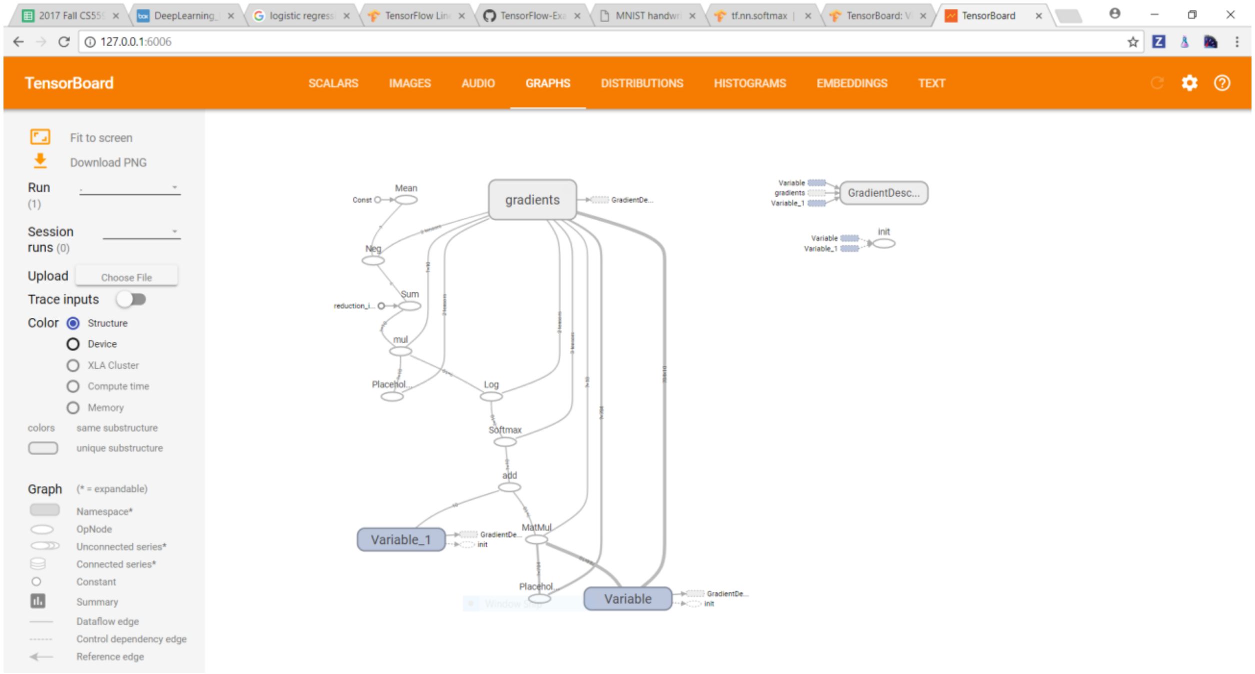
```
C:\Users\harsh\Anaconda3\python.exe C:/Users/harsh/Desktop/DeepLearning_Lesson2_SourceCode/labt1.py
Extracting /tmp/data/train-images-idx3-ubyte.gz
Extracting /tmp/data/train-labels-idx1-ubyte.gz
Extracting /tmp/data/t10k-images-idx3-ubyte.gz
Extracting /tmp/data/t10k-labels-idx1-ubyte.gz
2017-11-02 18:17:25.568264: W c:\l\work\tensorflow-1.1.0\tensorflow\core\platform\cpu_feature_guard.cc:45] The Tensor
2017-11-02 18:17:25.568471: W c:\l\work\tensorflow-1.1.0\tensorflow\core\platform\cpu_feature_guard.cc:45] The Tensor
2017-11-02 18:17:25.568660: W c:\l\work\tensorflow-1.1.0\tensorflow\core\platform\cpu_feature_guard.cc:45] The Tensor
2017-11-02 18:17:25.568852: W c:\l\work\tensorflow-1.1.0\tensorflow\core\platform\cpu_feature_guard.cc:45] The Tensor
```

Parameters

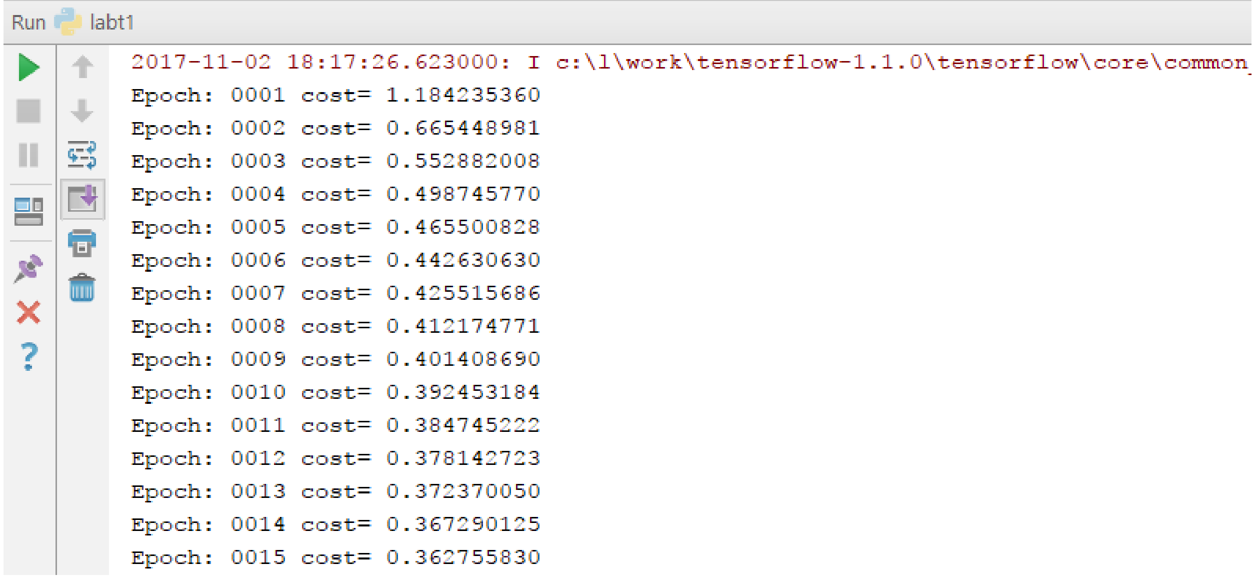
We are using learning rate at 0.01.
We are using the training loops equal to 25.
We will be using a batch size of 100.
We will display each epoch.

Evaluation & Discussion

Graphs:

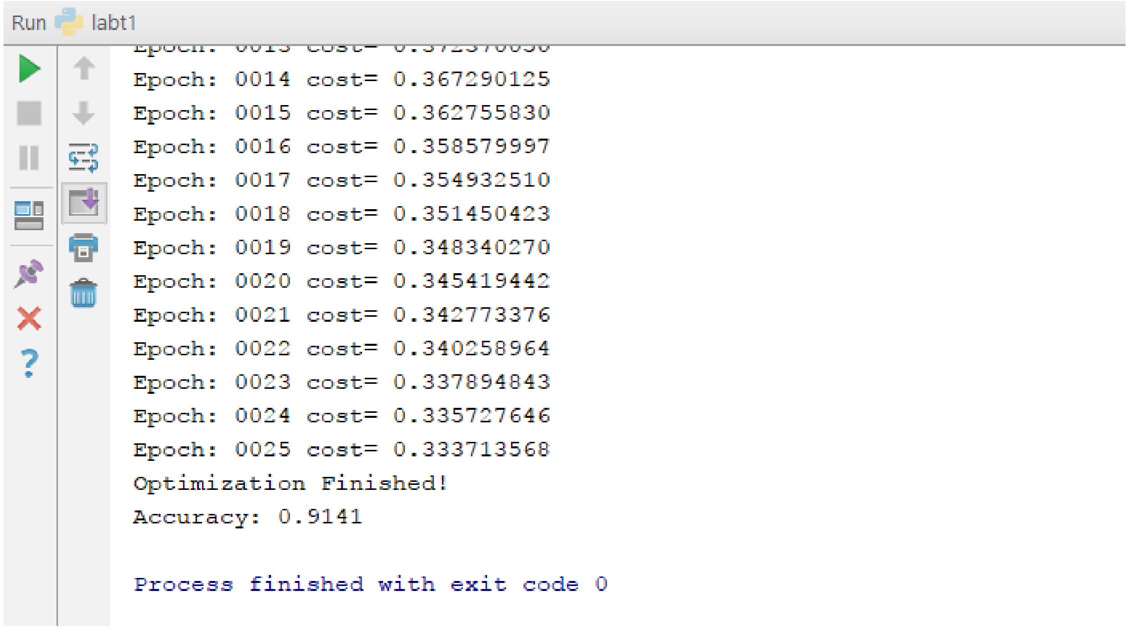


Output:



```
2017-11-02 18:17:26.623000: I c:\l\work\tensorflow-1.1.0\tensorflow\core\common
Epoch: 0001 cost= 1.184235360
Epoch: 0002 cost= 0.665448981
Epoch: 0003 cost= 0.552882008
Epoch: 0004 cost= 0.498745770
Epoch: 0005 cost= 0.465500828
Epoch: 0006 cost= 0.442630630
Epoch: 0007 cost= 0.425515686
Epoch: 0008 cost= 0.412174771
Epoch: 0009 cost= 0.401408690
Epoch: 0010 cost= 0.392453184
Epoch: 0011 cost= 0.384745222
Epoch: 0012 cost= 0.378142723
Epoch: 0013 cost= 0.372370050
Epoch: 0014 cost= 0.367290125
Epoch: 0015 cost= 0.362755830
```

Accuracy:



```
Epoch: 0013 cost= 0.372370050
Epoch: 0014 cost= 0.367290125
Epoch: 0015 cost= 0.362755830
Epoch: 0016 cost= 0.358579997
Epoch: 0017 cost= 0.354932510
Epoch: 0018 cost= 0.351450423
Epoch: 0019 cost= 0.348340270
Epoch: 0020 cost= 0.345419442
Epoch: 0021 cost= 0.342773376
Epoch: 0022 cost= 0.340258964
Epoch: 0023 cost= 0.337894843
Epoch: 0024 cost= 0.335727646
Epoch: 0025 cost= 0.333713568
Optimization Finished!
Accuracy: 0.9141

Process finished with exit code 0
```

Conclusion

We are getting a accuracy of 0.91 which is not consider very good.as human normally has an accuracy of 0.95-0.96 so if we can get the accuracy more than 0.97 the model we are using is best.to get a accuracy around .97 we can go for **Convolutional nets.**