README.md

Facial recognition

For this problem tensorflow is used to train logistic regression model for facial recognition.

Data set

For this task Olivetti faces dataset is used. The original website describes dataset as

"There are ten different images of each of 40 distinct subjects. For some subjects, the images were taken at different times, varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images were taken against a dark homogeneous background with the subjects in an upright, frontal position (with tolerance for some side movement)."

There are 40 targets for which the classification is to be done.

Loading the data set

This data set is available in sklearn and for this task same is being used.

```
from sklearn.datasets import fetch_olivetti_faces

targets = data.target
features = data.images
encoder = OneHotEncoder()
targets = encoder.fit_transform(targets.reshape(-1, 1)).todense()
xTrain, xTest, yTrain, yTest = train_test_split(features, targets, test_size=0.1)
```

Training model

```
def logisticRegressionNetwork(inputDimension, outputDimension):
    X = tf.placeholder("float", [None, inputDimension])
    Y = tf.placeholder("float", [None, outputDimension])
    """Build Graph"""
    outputLayer = tf.nn.softmax(tf.matmul(X, tf.Variable(tf.zeros([inputDimension,
                          outputDimension])) +tf.Variable(tf.zeros([outputDimension])))
    return outputLayer, X, Y
```

Tensorflow uses graphs and the structure for same needs to be defined. The above method defines a network for logistic regression. At last layer softmax function is used so that multiclass classification can be performed.

```
def trainModel(session, network,
               xContainer, yContainer, xTrain, yTrain, xValidation, yValidation,
               learningRate, epochs):

    trainSetLoss = []
    validationSetAcc = []
    trainSetAcc = []

    lossFunction = tf.reduce_mean(-tf.reduce_sum(
                      yContainer * tf.log(network), reduction_indices=1))

    optimizer = tf.train.GradientDescentOptimizer(learning_rate = learningRate)
    trainOperation = optimizer.minimize(lossFunction)

    prediction = tf.equal(tf.argmax(network, 1), tf.argmax(yContainer, 1))
    accuracy = tf.reduce_mean(tf.cast(prediction, tf.float32))
    session.run(tf.global_variables_initializer())
```

```
    # Create a summary to monitor accuracy tensor
    tf.summary.scalar("Training_Accuracy", accuracy)
    tf.summary.scalar("Training_Loss", lossFunction)
    tf.summary.histogram("Acc_Hist", accuracy)
    tf.summary.histogram("Loss_Hist", lossFunction)
    merged_summary_op = tf.summary.merge_all()

    summary_writer = tf.summary.FileWriter("tensorboard/logs", graph=tf.get_default_graph())

    for i in range(0, epochs):
        _, loss, acc, summary = session.run([trainOperation, lossFunction, accuracy,
                                merged_summary_op], feed_dict={xContainer: xTrain,
                                                               yContainer: yTrain})
        summary_writer.add_summary(summary, i)
        if i%50 ==0:
            print("Training accuracy "+str(acc))

        trainSetLoss.append(loss)
        trainSetAcc.append(acc)
        loss, acc = session.run([lossFunction, accuracy],
                            feed_dict={xContainer: xValidation,
                                       yContainer: yValidation})
        if i%50 ==0:
            print("Test accuracy "+str(acc))

        validationSetAcc.append(acc)
    return trainSetLoss, trainSetAcc, validationSetAcc
```

The above method train the logistic regression model. Additionally it adds required entries in summary for tensorboard. For optimization process gradient descent optimizer is used.

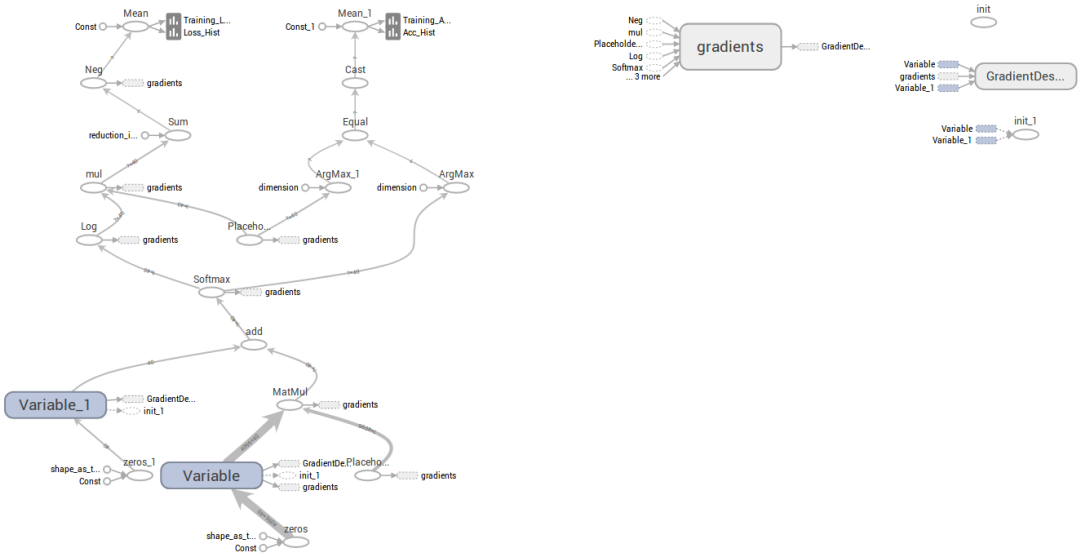
Execution

```
network, X, Y = logisticRegressionNetwork(4096, 40)
session = tf.Session()
trainSetLoss, trainSetAcc, validationSetAcc = trainModel(session, network, X, Y,
                                xTrainFlattened, yTrain, xTestFlattened, yTest, 0.07, 300)
```

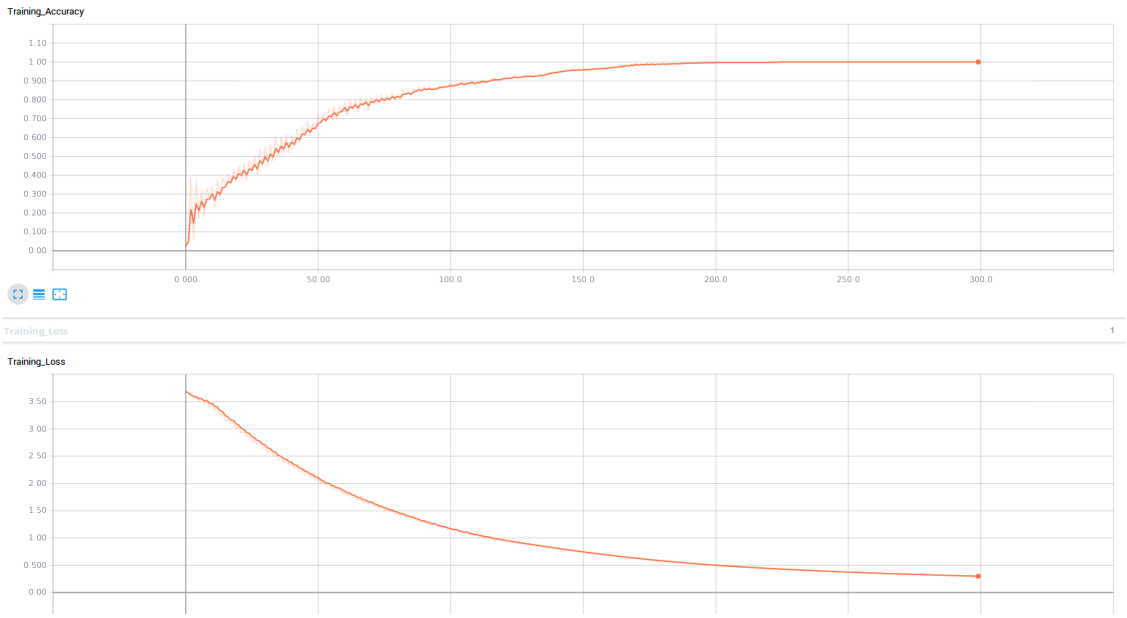
The algorithm is trained for 300 epochs following is the result.

Epoch	Train accuracy	Test accuracy
50	0.02	0
100	0.6	0.3
150	0.9	0.75
200	0.97	0.875
250	0.99	0.95
300	1	0.95

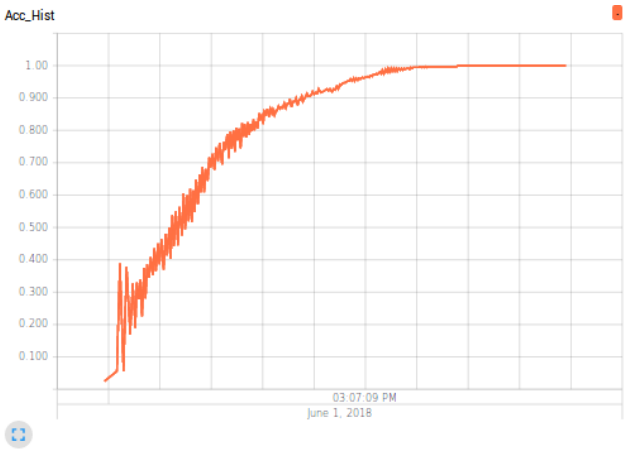
Graph structure

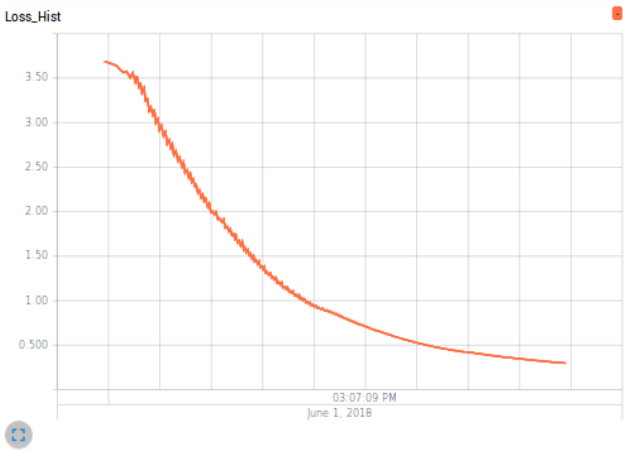


Training loss and accuracy

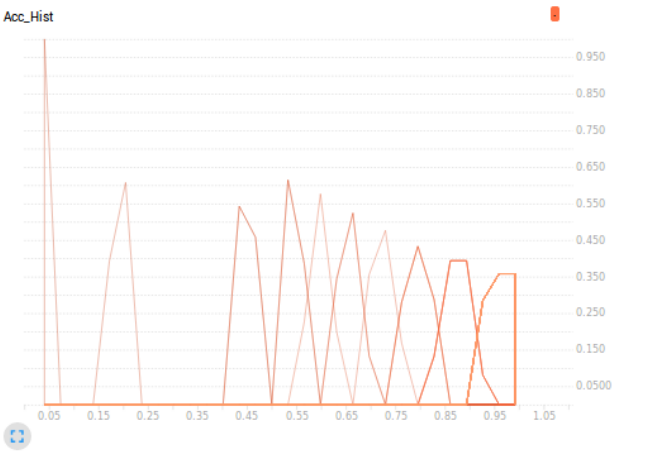


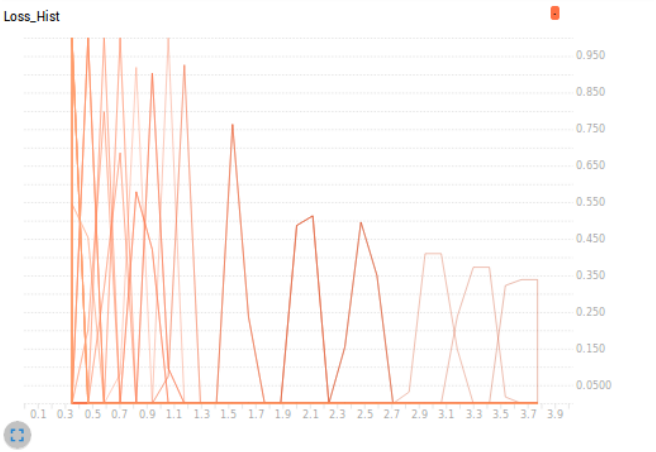
Loss distribution





Loss histogram


Acc_Hist
0.950
0.850
0.750
0.650
0.550
0.450
0.350
0.250
0.150
0.0500
0.05 0.15 0.25 0.35 0.45 0.55 0.65 0.75 0.85 0.95 1.05


Loss_Hist
Loss_Hist
0.950
0.850
0.750
0.650
0.550
0.450
0.350
0.250
0.150
0.0500
0.1 0.3 0.5 0.7 0.9 1.1 1.3 1.5 1.7 1.9 2.1 2.3 2.5 2.7 2.9 3.1 3.3 3.5 3.7 3.9