mnist-graph-linear.py

```
# Learning with a linear model

import numpy as np
import tensorflow as tf

# supress warnings
old_v = tf.logging.get_verbosity()
tf.logging.set_verbosity(tf.logging.ERROR)

from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets('MNIST_data', one_hot=True)
tf.logging.set_verbosity(old_v)

# MNIST_data is a collection of 2D gray level images.
# Each image is a picture of  a digit from 0..9
# Each image is of size 28 x 28 pixels

sess = tf.InteractiveSession()

# xi is an image of size n. yi is the N labels of the image
# X is mxn. Row xi of X is an image
# Y is mxN. Row yi of Y is the labels of xi
X = tf.placeholder(tf.float32, shape=[None, 784])
Y = tf.placeholder(tf.float32, shape=[None, 10])

# consider the linear model: W^T xi + b = yi. Here W is nxN, b is Nx1.
# In matrix form:  X W + B = Y, where B is  mxN. B = 1 b^T

W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))

sess.run(tf.global_variables_initializer())

# the linear regression model: (b is one row. It will be replicated as needed.)

predicted_Y = tf.matmul(X,W) + b

# means squared error loss is very bad
# loss = tf.losses.mean_squared_error(labels = Y, predictions = predicted_Y)
loss = tf.nn.softmax_cross_entropy_with_logits_v2(labels=Y, logits=predicted_Y)
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(loss)

for i in range(1000):
  batch = mnist.train.next_batch(100)
  train_step.run(feed_dict={X: batch[0], Y: batch[1]})

correct_prediction = tf.equal(tf.argmax(predicted_Y,1), tf.argmax(Y,1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

print(accuracy.eval(feed_dict={X: mnist.test.images, Y: mnist.test.labels}))
```