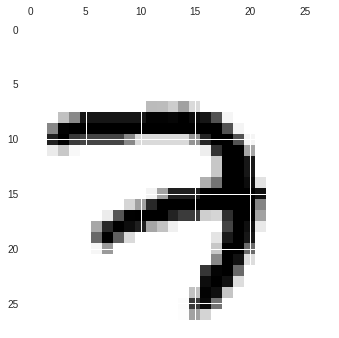
## LeNet for MNIST classification

import tensorflow as tf  
import numpy as np  
import matplotlib.pyplot as plt

from tensorflow.examples.tutorials.mnist import input\_data  
mnist = input\_data.read\_data\_sets('MNIST\_data', one\_hot=True)

Extracting MNIST\_data/train-images-idx3-ubyte.gz  
Extracting MNIST\_data/train-labels-idx1-ubyte.gz  
Extracting MNIST\_data/t10k-images-idx3-ubyte.gz  
Extracting MNIST\_data/t10k-labels-idx1-ubyte.gz

plt.matshow(mnist.train.images[0].reshape((28,28)))  
plt.show()



png

X = tf.placeholder(tf.float32, [None, 28, 28, 1], name='X')  
Y = tf.placeholder(tf.float32, [None, 10], name='Y')

W1 = tf.Variable(tf.truncated\_normal([5, 5, 1, 6], stddev=0.1))  
b1 = tf.Variable(tf.ones([6])/10)  
  
W2 = tf.Variable(tf.truncated\_normal([5, 5, 6, 16], stddev=0.1))  
b2 = tf.Variable(tf.ones([16])/10)

conv1 = tf.nn.relu(  
 tf.nn.conv2d(X, W1, strides=[1, 1, 1, 1], padding="SAME") + b1)  
  
pool1 = tf.nn.max\_pool(conv1,  
 ksize=[1, 2, 2, 1],  
 strides = [1, 2, 2, 1],  
 padding="SAME")  
  
conv2 = tf.nn.relu(  
 tf.nn.conv2d(pool1, W2, strides=[1, 1, 1, 1], padding="SAME") + b2)  
  
pool2 = tf.nn.max\_pool(conv2,  
 ksize=[1, 2, 2, 1],  
 strides = [1, 2, 2, 1],  
 padding="SAME")  
  
flatten = tf.reshape(pool2, [-1, 16\*7\*7])

W3 = tf.Variable(tf.truncated\_normal([16\*7\*7, 120], stddev=0.1))  
b3 = tf.Variable(tf.ones([120])/10)  
  
W4 = tf.Variable(tf.truncated\_normal([120, 84], stddev=0.1))  
b4 = tf.Variable(tf.ones([84])/10)  
  
W5 = tf.Variable(tf.truncated\_normal([84, 10], stddev=0.1))  
b5 = tf.Variable(tf.ones([10])/10)

# drop\_out1 = tf.nn.dropout(flatten, 0.4)  
  
fc1 = tf.nn.relu(tf.matmul(flatten, W3) + b3)  
  
# drop\_out2 = tf.nn.dropout(fc1, 0.4)  
  
fc2 = tf.nn.relu(tf.matmul(fc1, W4) + b4)  
  
  
Z = tf.matmul(fc2, W5) + b5  
  
Y\_ = tf.nn.softmax(Z)

cross\_entropy = tf.nn.softmax\_cross\_entropy\_with\_logits\_v2(logits=Z, labels=Y)  
  
loss\_objective = tf.reduce\_mean(cross\_entropy)\*100

learning\_rate = 0.001  
  
training = tf.train.GradientDescentOptimizer(  
 learning\_rate).minimize(loss\_objective)

is\_correct = tf.equal(tf.argmax(Y,1), tf.argmax(Y\_,1))  
  
accuracy = tf.reduce\_mean(tf.cast(is\_correct, tf.float32))

init = tf.global\_variables\_initializer()  
  
epochs = 5  
iterations = 500  
  
train\_err = []  
train\_acc = []  
  
test\_accuracy, test\_loss = 0, 99999  
  
with tf.Session() as sess:  
   
 sess.run(init)  
   
 for epoch in range(epochs):  
   
 print('\nEpoch : {0}\n'.format(epoch+1))  
   
 for i in range(iterations+1):  
   
 batch\_X, batch\_Y = mnist.train.next\_batch(100)  
 batch\_X = np.reshape(batch\_X, (-1, 28, 28, 1))  
 step = sess.run(training, feed\_dict={X: batch\_X, Y: batch\_Y})  
   
 if i % 100 == 0:  
  
 train\_accuracy, new\_err = sess.run([accuracy, loss\_objective],  
 feed\_dict={X: batch\_X, Y: batch\_Y})   
   
 train\_acc.append(train\_accuracy); train\_err.append(new\_err)  
   
 print('{2} Training Accuracy : {0} | Loss : {1}'.format(train\_accuracy,  
 new\_err, i))  
   
   
 batch\_X, batch\_Y = mnist.validation.images, mnist.validation.labels  
 batch\_X = np.reshape(batch\_X, (-1, 28, 28, 1))  
  
 val\_accuracy, val\_loss = sess.run([accuracy, loss\_objective],  
 feed\_dict={X: batch\_X, Y: batch\_Y})  
   
 print('\nValidation Accuracy : {0} | Loss : {1}'.format(val\_accuracy,  
 val\_loss))  
  
 batch\_X, batch\_Y = mnist.test.images, mnist.test.labels  
 batch\_X = np.reshape(batch\_X, (-1, 28, 28, 1))  
  
 test\_accuracy, test\_loss = sess.run([accuracy, loss\_objective],  
 feed\_dict={X: batch\_X, Y: batch\_Y})

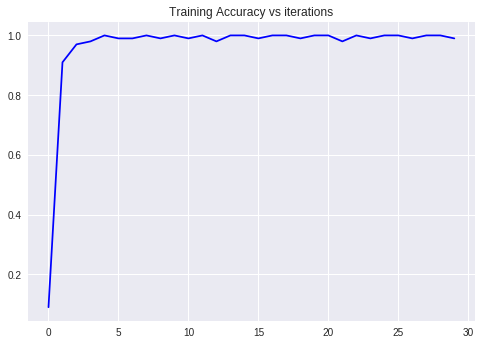
Epoch : 1  
  
0 Training Accuracy : 0.09000000357627869 | Loss : 230.72386169433594  
100 Training Accuracy : 0.9100000262260437 | Loss : 23.432987213134766  
200 Training Accuracy : 0.9700000286102295 | Loss : 15.549623489379883  
300 Training Accuracy : 0.9800000190734863 | Loss : 4.88357400894165  
400 Training Accuracy : 1.0 | Loss : 4.831546783447266  
500 Training Accuracy : 0.9900000095367432 | Loss : 3.250980854034424  
  
Epoch : 2  
  
0 Training Accuracy : 0.9900000095367432 | Loss : 3.1790456771850586  
100 Training Accuracy : 1.0 | Loss : 2.2342658042907715  
200 Training Accuracy : 0.9900000095367432 | Loss : 2.94486927986145  
300 Training Accuracy : 1.0 | Loss : 2.9101932048797607  
400 Training Accuracy : 0.9900000095367432 | Loss : 3.389744520187378  
500 Training Accuracy : 1.0 | Loss : 1.1675102710723877  
  
Epoch : 3  
  
0 Training Accuracy : 0.9800000190734863 | Loss : 6.132957935333252  
100 Training Accuracy : 1.0 | Loss : 1.1962836980819702  
200 Training Accuracy : 1.0 | Loss : 1.7145755290985107  
300 Training Accuracy : 0.9900000095367432 | Loss : 2.203922748565674  
400 Training Accuracy : 1.0 | Loss : 1.6431407928466797  
500 Training Accuracy : 1.0 | Loss : 1.2255125045776367  
  
Epoch : 4  
  
0 Training Accuracy : 0.9900000095367432 | Loss : 2.8778159618377686  
100 Training Accuracy : 1.0 | Loss : 0.9480134844779968  
200 Training Accuracy : 1.0 | Loss : 0.4046200215816498  
300 Training Accuracy : 0.9800000190734863 | Loss : 2.413910388946533  
400 Training Accuracy : 1.0 | Loss : 1.024740219116211  
500 Training Accuracy : 0.9900000095367432 | Loss : 2.3180315494537354  
  
Epoch : 5  
  
0 Training Accuracy : 1.0 | Loss : 0.917784571647644  
100 Training Accuracy : 1.0 | Loss : 1.3640785217285156  
200 Training Accuracy : 0.9900000095367432 | Loss : 1.9262890815734863  
300 Training Accuracy : 1.0 | Loss : 0.758085310459137  
400 Training Accuracy : 1.0 | Loss : 1.5455034971237183  
500 Training Accuracy : 0.9900000095367432 | Loss : 2.533975601196289  
  
Validation Accuracy : 0.9846000075340271 | Loss : 5.334902763366699

plt.plot(range(len(train\_err)), train\_err, color='red')  
plt.title('Training error vs iterations')  
plt.show()



png

plt.plot(range(len(train\_acc)), train\_acc, color='blue')  
plt.title('Training Accuracy vs iterations')  
plt.show()



png

print('\nTest Accuracy : {0} \nTest Loss : {1}'.format(test\_accuracy,  
 test\_loss))

Test Accuracy : 0.9833999872207642   
Test Loss : 4.8419623374938965