Imports and data input

```
In [22]: import tensorflow as tf
```

```
In [23]: from tensorflow.examples.tutorials.mnist import input_data
```

```
In [24]: 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
```

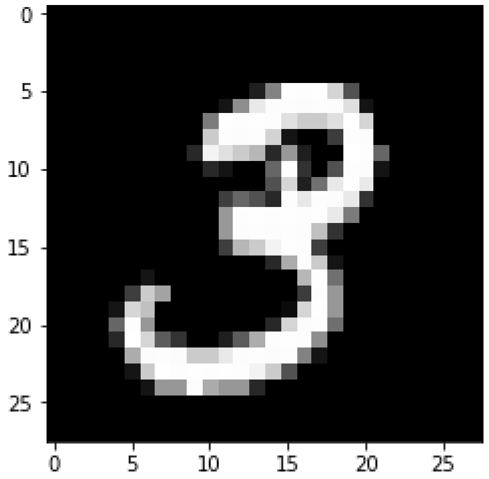
Visualizing example of image

```
In [25]: import matplotlib.pyplot as plt
```

```
In [26]: #reshape pixels into the 28x28 image
         im = mnist.train.images[1].reshape(28,28)
```

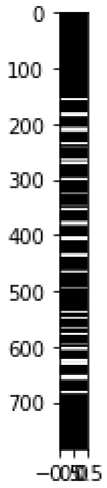
```
In [27]: #The CNN version of this analysis will view the image like this.
         plt.imshow(mnist.train.images[1].reshape(28,28),cmap='gist_gray')

Out[27]: <matplotlib.image.AxesImage at 0x18b01eda630>
```



In [28]:
```
#visualize all 784 in a column which is how this more simple, Softmax version will
plt.imshow(mnist.train.images[1].reshape(784,1),cmap='gist_gray',aspect=0.02)
```

Out[28]:
```
<matplotlib.image.AxesImage at 0x18b0cd3dfd0>
```



Placeholders

In [29]:
```
x = tf.placeholder(tf.float32,shape=[None,784])
y_true = tf.placeholder(tf.float32, [None,10])
```

In [30]:
```
# 784 due to number of pixels, 10 due to possible numbers (0-9)
W = tf.Variable(tf.zeros([784,10]))
b = tf.Variable(tf.zeros([10]))
```

In [31]:
```
# Graph
y = tf.matmul(x,W) + b
```

Loss Function and Optimizer

In [32]:
```
#Softmax is being used to estimate the probability that a image is any one of the
#Then the highest scoring number will be selected
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(labels=y_tr
```

In [33]:

```
#Create gradient Descent Optimizer
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.5)
train = optimizer.minimize(cross_entropy)
```

Start Session

In [34]:

```
init = tf.global_variables_initializer()
```

In [35]:

```
with tf.Session() as sess:

    sess.run(init)

    for step in range(100):

        #use batch sizes of 100 to train the model
        batch_x, batch_y = mnist.train.next_batch(batch_size=100)

        #feed in x and y data
        sess.run(train,feed_dict={x:batch_x,y_true:batch_y})

    # Predict label and compare it to true label to calculate accuracy
    correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_true,1))

    acc = tf.reduce_mean(tf.cast(correct_prediction,tf.float32))

    print(sess.run(acc,feed_dict={x:mnist.test.images,y_true:mnist.test.labels}))
```

```
0.8859
```

In [ ]: