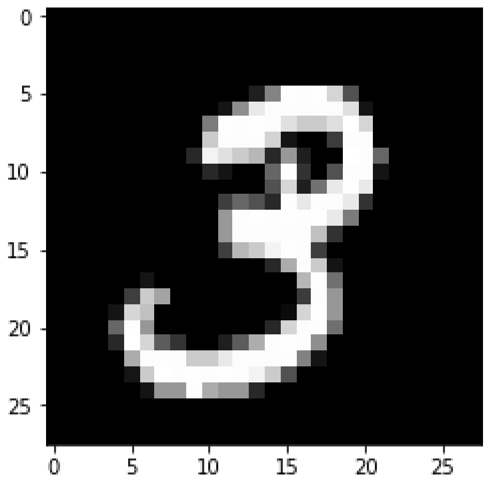
In [1]:
```
#-----------------------------------------------------------------
#  Author:        TFT
#  Written:       05/02/2018
#  Last updated:  05/02/2018
#
#
#  TFT Machine Learning
#  Assignment  Week 3-1
#
#-----------------------------------------------------------------
```

In [2]:
```
import numpy as np
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
import time

# download the dataset
# each image is 28 x 28
mnist = input_data.read_data_sets('data',one_hot=True)
```

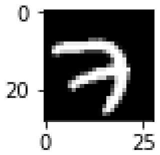
```
Extracting data/train-images-idx3-ubyte.gz
Extracting data/train-labels-idx1-ubyte.gz
Extracting data/t10k-images-idx3-ubyte.gz
Extracting data/t10k-labels-idx1-ubyte.gz
```

In [3]:
```
print(mnist.train.num_examples)
print(mnist.validation.num_examples)
print(mnist.test.num_examples)
print(mnist.train.images.shape)
print(mnist.train.labels.shape)
img = mnist.train.images[1]
label = mnist.train.labels[1]
print(img.shape)
print(label.shape)
print(label)
print(np.argmax(label))

import matplotlib.pyplot as plt
%matplotlib inline
plt.imshow(img.reshape([28,28]), cmap = "gray")
plt.show()
```

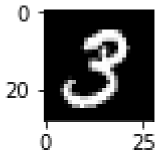
```
55000
5000
10000
(55000, 784)
(55000, 10)
(784,)
(10,)
[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
3
```



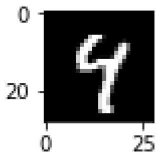
In [4]:
```python
for i in range(10):
    img = mnist.train.images[i]
    label = mnist.train.labels[i]
    plt.figure(figsize=(1,1))
    plt.imshow(img.reshape([28, 28]), cmap = "gray")
    plt.show()
    cls = np.argmax(label)
    print("Ground Truth: %d" % cls)
```



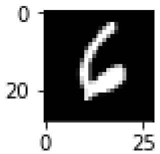

```
Ground Truth: 7
```



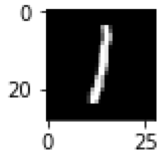

```
Ground Truth: 3
```



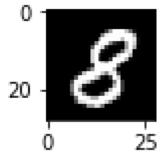

```
Ground Truth: 4
```



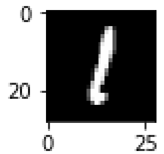

```
Ground Truth: 6
```



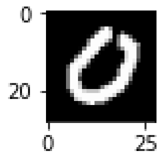
Ground Truth: 1



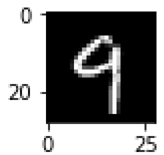
Ground Truth: 8



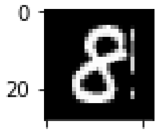
Ground Truth: 1



Ground Truth: 0



Ground Truth: 9



```
Ground Truth: 8
```

In [5]:

```
# 1. Create the model (build the compuation graph)
## Hyperparameters
batch_size = 256
n_input = 784 # 784 = 28 x 28
n_hidden = 256
n_classes = 10
learning_rate = 0.725

## Model input
x = tf.placeholder(tf.float32, [None, n_input])

## Hidden layer
W1 = tf.Variable(tf.random_normal([n_input, n_hidden], stddev=0.1))
b1 = tf.Variable(tf.zeros([n_hidden]))
h1 = tf.matmul(x, W1) + b1
h1 = tf.nn.relu(h1)

## Hidden layer
#W2 = tf.Variable(tf.random_normal([n_hidden, n_hidden], stddev=0.1))
#b2 = tf.Variable(tf.zeros([n_hidden]))
#h2 = tf.matmul(h1, W2) + b2
#h2 = tf.nn.relu(h2)

## Output layer
W_out = tf.Variable(tf.random_normal([n_hidden, n_classes], stddev=0.1))
b_out = tf.Variable(tf.zeros([n_classes]))
y_pred = tf.matmul(h1, W_out) + b_out

#W1 = tf.Variable(tf.random_normal([n_input, n_classes], stddev=0.1))
#b1 = tf.Variable(tf.zeros([n_classes]))
#y_pred = tf.matmul(x, W1) + b1


## Define loss and optimizer
y_gt = tf.placeholder(tf.float32, [None, n_classes])
loss = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits_v2(logits=y_pred, labels=y_gt, name='loss'))

## Train (update model parameters)
optimizer = tf.train.GradientDescentOptimizer(learning_rate)
#optimizer = tf.train.MomentumOptimizer(learning_rate, 0.9)
#optimizer = tf.train.AdamOptimizer()
```

```
train_step = optimizer.minimize(loss)

## Compute Accuracy
cls_pred = tf.argmax(y_pred, axis = 1)
cls_gt = tf.argmax(y_gt, axis = 1)
correct_prediction = tf.equal(cls_pred, cls_gt)
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

In [6]:

```
# 2. Train
## Initialize
sess = tf.InteractiveSession()
tf.global_variables_initializer().run()
max_iter = 2000
for iter in range(max_iter):
    batch_x, batch_y = mnist.train.next_batch(batch_size)
    sess.run(train_step, feed_dict = {x: batch_x, y_gt: batch_y})
    if iter % 100 == 0:
        train_loss = sess.run(loss, feed_dict = {x: batch_x, y_gt: batch_y})
        train_accuracy = sess.run(accuracy, feed_dict = {x: batch_x, y_gt: batch_y})

        validation_x = mnist.validation.images
        validation_y = mnist.validation.labels
        validation_accuracy = sess.run(accuracy, {x: validation_x, y_gt: validation_y})

        print("iter step %d, loss %f, training accuracy %f, validation accuracy %f" %
              (iter, train_loss, train_accuracy, validation_accuracy))
```

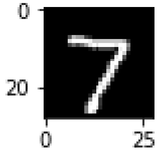
```
iter step 0, loss 2.573666, training accuracy 0.398438, validation accuracy 0.288800
iter step 100, loss 0.234085, training accuracy 0.929688, validation accuracy 0.934200
iter step 200, loss 0.130955, training accuracy 0.957031, validation accuracy 0.949600
iter step 300, loss 0.062175, training accuracy 0.984375, validation accuracy 0.958000
iter step 400, loss 0.095586, training accuracy 0.976562, validation accuracy 0.964200
iter step 500, loss 0.090505, training accuracy 0.984375, validation accuracy 0.966800
iter step 600, loss 0.063141, training accuracy 0.984375, validation accuracy 0.970000
iter step 700, loss 0.072876, training accuracy 0.984375, validation accuracy 0.969400
iter step 800, loss 0.037901, training accuracy 0.992188, validation accuracy 0.973600
iter step 900, loss 0.031467, training accuracy 1.000000, validation accuracy 0.976000
iter step 1000, loss 0.030806, training accuracy 0.996094, validation accuracy 0.976200
iter step 1100, loss 0.022884, training accuracy 1.000000, validation accuracy 0.976600
iter step 1200, loss 0.020247, training accuracy 1.000000, validation accuracy 0.976400
iter step 1300, loss 0.021146, training accuracy 0.996094, validation accuracy 0.975200
iter step 1400, loss 0.023413, training accuracy 0.996094, validation accuracy 0.977200
iter step 1500, loss 0.016684, training accuracy 1.000000, validation accuracy 0.980200
iter step 1600, loss 0.012238, training accuracy 1.000000, validation accuracy 0.979400
iter step 1700, loss 0.018151, training accuracy 0.996094, validation accuracy 0.977400
iter step 1800, loss 0.014926, training accuracy 1.000000, validation accuracy 0.978000
iter step 1900, loss 0.018532, training accuracy 0.996094, validation accuracy 0.978400
```

In [7]:

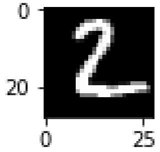
```
# 3. Test the trained model
train_x = mnist.train.images
train_y = mnist.train.labels
train_accuracy = sess.run(accuracy, {x: train_x, y_gt: train_y})

validation_x = mnist.validation.images
validation_y = mnist.validation.labels
validation_accuracy = sess.run(accuracy, {x: validation_x, y_gt: validation_y})

test_x = mnist.test.images
test_y = mnist.test.labels
test_accuracy = sess.run(accuracy, {x: test_x, y_gt: test_y})

print("train accuray: %f" % train_accuracy)
print("validation accuray: %f" % validation_accuracy)
print("test accuray: %f" % test_accuracy)
```

```
train accuray: 0.993527
validation accuray: 0.979600
test accuray: 0.977800
```

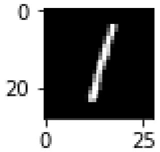
In [8]:
```
for i in range(10):
    img = mnist.test.images[i]
    label = mnist.test.labels[i]
    plt.figure(figsize=(1,1))
    plt.imshow(img.reshape([28, 28]), cmap = "gray")
    plt.show()
    cls = np.argmax(label)
    print("Ground Truth: %d" % cls)

    pred_label = sess.run(y_pred, feed_dict = {x: img.reshape([1, -1])})
    pred_cls = np.argmax(pred_label)
    print("Model prediction: %s" % pred_cls)
```



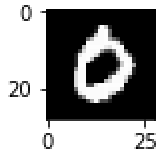

```
Ground Truth: 7
Model prediction: 7
```



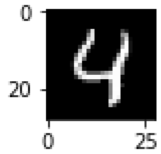

```
Ground Truth: 2
Model prediction: 2
```



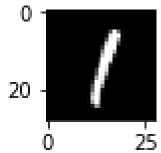

```
Ground Truth: 1
Model prediction: 1
```



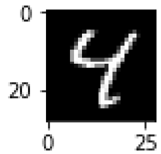

```
Ground Truth: 0
Model prediction: 0
```



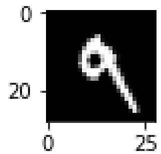

```
Ground Truth: 4
Model prediction: 4
```




```
Ground Truth: 1
Model prediction: 1
```

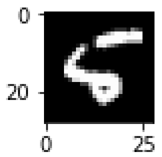



```
Ground Truth: 4
Model prediction: 4
```

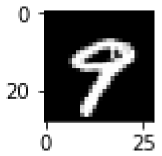



```
Ground Truth: 9
```

```
Model prediction: 9
```



```
Ground Truth: 5
Model prediction: 5
```



```
Ground Truth: 9
Model prediction: 9
```