In [1]:
```
from __future__ import division, print_function, absolute_import

# Import MNIST data
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("/tmp/data/", one_hot=False)

import tensorflow as tf
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import PIL
```

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

In [2]:
```
# Training Parameters
learning_rate = 0.001
num_steps = 5000
batch_size = 128

# Network Parameters
num_input = 784 # MNIST data input (img shape: 28*28)
num_classes = 10 # MNIST total classes (0-9 digits)
dropout = 0.25 # Dropout, probability to drop a unit
```

In [3]:
```
# Create the neural network
def conv_net(x_dict, n_classes, dropout, reuse, is_training):

    # Define a scope for reusing the variables
    with tf.variable_scope('ConvNet', reuse=reuse):
        # TF Estimator input is a dict, in case of multiple inputs
        x = x_dict['images']

        # MNIST data input is a 1-D vector of 784 features (28*28 pixels)
        # Reshape to match picture format [Height x Width x Channel]
        # Tensor input become 4-D: [Batch Size, Height, Width, Channel]
        x = tf.reshape(x, shape=[-1, 28, 28, 1])

        # Convolution Layer with 32 filters and a kernel size of 5
```

```
        conv1 = tf.layers.conv2d(x, 32, 5, activation=tf.nn.relu)
        # Max Pooling (down-sampling) with strides of 2 and kernel size of 2
        conv1 = tf.layers.max_pooling2d(conv1, 2, 2)

        # Convolution Layer with 64 filters and a kernel size of 3
        conv2 = tf.layers.conv2d(conv1, 64, 3, activation=tf.nn.relu)
        # Max Pooling (down-sampling) with strides of 2 and kernel size of 2
        conv2 = tf.layers.max_pooling2d(conv2, 2, 2)

        # Flatten the data to a 1-D vector for the fully connected layer
        fc1 = tf.contrib.layers.flatten(conv2)

        # Fully connected layer (in tf contrib folder for now)
        fc1 = tf.layers.dense(fc1, 1024)
        # Apply Dropout (if is_training is False, dropout is not applied)
        fc1 = tf.layers.dropout(fc1, rate=dropout, training=is_training)

        # Output layer, class prediction
        out = tf.layers.dense(fc1, n_classes)

    return out
```

In [4]:
```
# Define the model function (following TF Estimator Template)
def model_fn(features, labels, mode):

    # Build the neural network
    # Because Dropout have different behavior at training and prediction time, we
    # need to create 2 distinct computation graphs that still share the same weights.
    logits_train = conv_net(features, num_classes, dropout, reuse=False, is_training=Tru
    logits_test = conv_net(features, num_classes, dropout, reuse=True, is_training=False

    # Predictions
    pred_classes = tf.argmax(logits_test, axis=1)
    pred_probas = tf.nn.softmax(logits_test)

    # If prediction mode, early return
    if mode == tf.estimator.ModeKeys.PREDICT:
        return tf.estimator.EstimatorSpec(mode, predictions=pred_classes)

    # Define loss and optimizer
    loss_op = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(
        logits=logits_train, labels=tf.cast(labels, dtype=tf.int32)))
    optimizer = tf.train.AdamOptimizer(learning_rate=learning_rate)
    train_op = optimizer.minimize(loss_op, global_step=tf.train.get_global_step())

    # Evaluate the accuracy of the model
    acc_op = tf.metrics.accuracy(labels=labels, predictions=pred_classes)
```

```
            # TF Estimators requires to return a EstimatorSpec, that specify
            # the different ops for training, evaluating, ...
            estim_specs = tf.estimator.EstimatorSpec(
              mode=mode,
              predictions=pred_classes,
              loss=loss_op,
              train_op=train_op,
              eval_metric_ops={'accuracy': acc_op})

            return estim_specs
```

```
In [5]: # Build the Estimator
        model = tf.estimator.Estimator(model_fn)
```

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: C:\Users\tom13\AppData\Local\Temp\
INFO:tensorflow:Using config: {'_keep_checkpoint_every_n_hours': 10000, '_task_type': 'worker',
```

```
In [6]: # Define the input function for training
        input_fn = tf.estimator.inputs.numpy_input_fn(
            x={'images': mnist.train.images}, y=mnist.train.labels,
            batch_size=batch_size, num_epochs=None, shuffle=True)
        # Train the Model
        model.train(input_fn, steps=num_steps)
```

```
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Saving checkpoints for 1 into C:\Users\tom13\AppData\Local\Temp\tmp97az0nju\mode
INFO:tensorflow:step = 1, loss = 2.31252
INFO:tensorflow:global_step/sec: 62.4813
INFO:tensorflow:step = 101, loss = 0.165548 (1.602 sec)
INFO:tensorflow:global_step/sec: 67.4517
INFO:tensorflow:step = 201, loss = 0.149662 (1.485 sec)
INFO:tensorflow:global_step/sec: 67.1394
INFO:tensorflow:step = 301, loss = 0.208731 (1.487 sec)
INFO:tensorflow:global_step/sec: 67.1479
INFO:tensorflow:step = 401, loss = 0.0409326 (1.490 sec)
INFO:tensorflow:global_step/sec: 68.0719
INFO:tensorflow:step = 501, loss = 0.0213496 (1.470 sec)
INFO:tensorflow:global_step/sec: 68.771
INFO:tensorflow:step = 601, loss = 0.037192 (1.452 sec)
INFO:tensorflow:global_step/sec: 67.3723
INFO:tensorflow:step = 701, loss = 0.0822325 (1.485 sec)
INFO:tensorflow:global_step/sec: 68.6787
INFO:tensorflow:step = 801, loss = 0.0128403 (1.455 sec)
INFO:tensorflow:global_step/sec: 68.8452
INFO:tensorflow:step = 901, loss = 0.0358137 (1.455 sec)
INFO:tensorflow:global_step/sec: 68.75
INFO:tensorflow:step = 1001, loss = 0.0192664 (1.453 sec)
```

```
INFO:tensorflow:global_step/sec: 68.5087
INFO:tensorflow:step = 1101, loss = 0.00774879 (1.459 sec)
INFO:tensorflow:global_step/sec: 67.7028
INFO:tensorflow:step = 1201, loss = 0.0851327 (1.477 sec)
INFO:tensorflow:global_step/sec: 67.7039
INFO:tensorflow:step = 1301, loss = 0.0553374 (1.479 sec)
INFO:tensorflow:global_step/sec: 64.6995
INFO:tensorflow:step = 1401, loss = 0.0355084 (1.547 sec)
INFO:tensorflow:global_step/sec: 67.4277
INFO:tensorflow:step = 1501, loss = 0.0163228 (1.481 sec)
INFO:tensorflow:global_step/sec: 65.4895
INFO:tensorflow:step = 1601, loss = 0.00959715 (1.528 sec)
INFO:tensorflow:global_step/sec: 67.7365
INFO:tensorflow:step = 1701, loss = 0.00657101 (1.477 sec)
INFO:tensorflow:global_step/sec: 68.2635
INFO:tensorflow:step = 1801, loss = 0.00808421 (1.464 sec)
INFO:tensorflow:global_step/sec: 68.3004
INFO:tensorflow:step = 1901, loss = 0.00447089 (1.465 sec)
INFO:tensorflow:global_step/sec: 66.2228
INFO:tensorflow:step = 2001, loss = 0.00395705 (1.510 sec)
INFO:tensorflow:global_step/sec: 67.0613
INFO:tensorflow:step = 2101, loss = 0.0143454 (1.492 sec)
INFO:tensorflow:global_step/sec: 66.8144
INFO:tensorflow:step = 2201, loss = 0.00251255 (1.496 sec)
INFO:tensorflow:global_step/sec: 67.2051
INFO:tensorflow:step = 2301, loss = 0.00249033 (1.487 sec)
INFO:tensorflow:global_step/sec: 67.9835
INFO:tensorflow:step = 2401, loss = 0.0498065 (1.470 sec)
INFO:tensorflow:global_step/sec: 68.2512
INFO:tensorflow:step = 2501, loss = 0.00226328 (1.467 sec)
INFO:tensorflow:global_step/sec: 67.8173
INFO:tensorflow:step = 2601, loss = 0.000694454 (1.473 sec)
INFO:tensorflow:global_step/sec: 67.6461
INFO:tensorflow:step = 2701, loss = 0.0344697 (1.480 sec)
INFO:tensorflow:global_step/sec: 67.8946
INFO:tensorflow:step = 2801, loss = 0.00430426 (1.472 sec)
INFO:tensorflow:global_step/sec: 69.0382
INFO:tensorflow:step = 2901, loss = 0.033511 (1.447 sec)
INFO:tensorflow:global_step/sec: 67.9999
INFO:tensorflow:step = 3001, loss = 0.00375906 (1.470 sec)
INFO:tensorflow:global_step/sec: 68.4807
INFO:tensorflow:step = 3101, loss = 0.00979484 (1.462 sec)
INFO:tensorflow:global_step/sec: 67.3055
INFO:tensorflow:step = 3201, loss = 0.0192151 (1.488 sec)
INFO:tensorflow:global_step/sec: 68.755
INFO:tensorflow:step = 3301, loss = 0.00091159 (1.452 sec)
INFO:tensorflow:global_step/sec: 68.2027
INFO:tensorflow:step = 3401, loss = 0.00265687 (1.466 sec)
```

```
INFO:tensorflow:global_step/sec: 66.981
INFO:tensorflow:step = 3501, loss = 0.00322506 (1.493 sec)
INFO:tensorflow:global_step/sec: 67.9271
INFO:tensorflow:step = 3601, loss = 0.00257395 (1.475 sec)
INFO:tensorflow:global_step/sec: 68.4105
INFO:tensorflow:step = 3701, loss = 0.0113146 (1.459 sec)
INFO:tensorflow:global_step/sec: 68.9221
INFO:tensorflow:step = 3801, loss = 0.0119776 (1.453 sec)
INFO:tensorflow:global_step/sec: 68.3297
INFO:tensorflow:step = 3901, loss = 0.000739706 (1.464 sec)
INFO:tensorflow:global_step/sec: 67.7096
INFO:tensorflow:step = 4001, loss = 0.00202849 (1.474 sec)
INFO:tensorflow:global_step/sec: 68.4309
INFO:tensorflow:step = 4101, loss = 0.00123262 (1.462 sec)
INFO:tensorflow:global_step/sec: 68.589
INFO:tensorflow:step = 4201, loss = 0.0213804 (1.460 sec)
INFO:tensorflow:global_step/sec: 68.3938
INFO:tensorflow:step = 4301, loss = 0.00818979 (1.463 sec)
INFO:tensorflow:global_step/sec: 68.1871
INFO:tensorflow:step = 4401, loss = 0.0192472 (1.465 sec)
INFO:tensorflow:global_step/sec: 67.4095
INFO:tensorflow:step = 4501, loss = 0.009068 (1.483 sec)
INFO:tensorflow:global_step/sec: 68.0209
INFO:tensorflow:step = 4601, loss = 0.00785339 (1.470 sec)
INFO:tensorflow:global_step/sec: 68.466
INFO:tensorflow:step = 4701, loss = 0.00145256 (1.461 sec)
INFO:tensorflow:global_step/sec: 68.8636
INFO:tensorflow:step = 4801, loss = 0.0157578 (1.452 sec)
INFO:tensorflow:global_step/sec: 68.8196
INFO:tensorflow:step = 4901, loss = 0.0353847 (1.454 sec)
INFO:tensorflow:Saving checkpoints for 5000 into C:\Users\tom13\AppData\Local\Temp\tmp97az0nju\m
INFO:tensorflow:Loss for final step: 0.024679.
```

Out[6]: <tensorflow.python.estimator.estimator.Estimator at 0x226833dcd30>

In [7]:
```
# Evaluate the Model
# Define the input function for evaluating
input_fn = tf.estimator.inputs.numpy_input_fn(
    x={'images': mnist.test.images}, y=mnist.test.labels,
    batch_size=batch_size, shuffle=False)
# Use the Estimator 'evaluate' method
model.evaluate(input_fn)
```

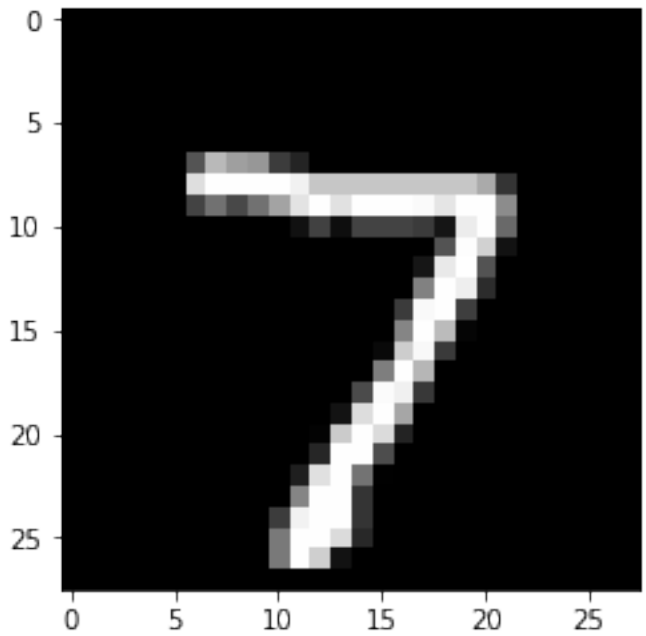
```
INFO:tensorflow:Starting evaluation at 2018-01-21-16:56:42
INFO:tensorflow:Restoring parameters from C:\Users\tom13\AppData\Local\Temp\tmp97az0nju\model.ck
INFO:tensorflow:Finished evaluation at 2018-01-21-16:56:43
INFO:tensorflow:Saving dict for global step 5000: accuracy = 0.989, global_step = 5000, loss = 0
```

```
Out[7]: {'accuracy': 0.98900002, 'global_step': 5000, 'loss': 0.052190915}
```

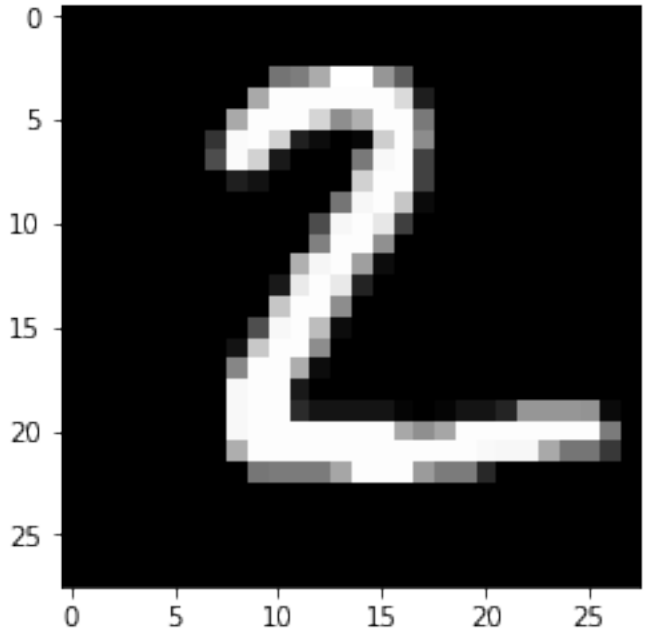
```
In [8]: # Predict single images
        n_images = 5
        # Get images from test set
        test_images = mnist.test.images[:n_images]
        # Prepare the input data
        input_fn = tf.estimator.inputs.numpy_input_fn(
            x={'images': test_images}, shuffle=False)
        # Use the model to predict the images class
        preds = list(model.predict(input_fn))

        # Display
        for i in range(n_images):
            plt.imshow(np.reshape(test_images[i], [28, 28]), cmap='gray')
            plt.show()
            print("Model prediction:", preds[i])
```

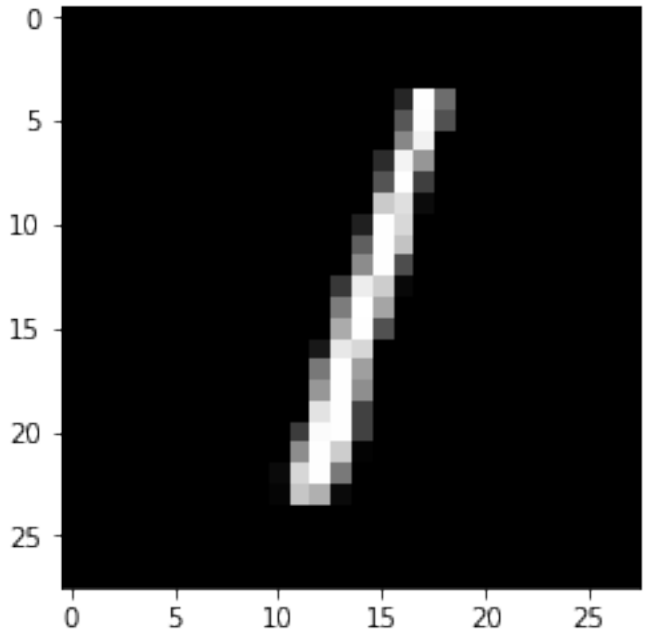
```
INFO:tensorflow:Restoring parameters from C:\Users\tom13\AppData\Local\Temp\tmp97az0nju\model.ck
```



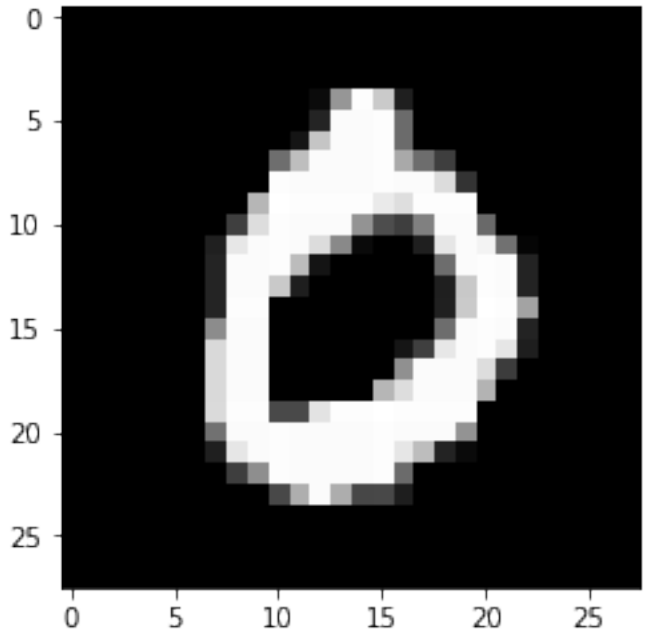

```
Model prediction: 7
```



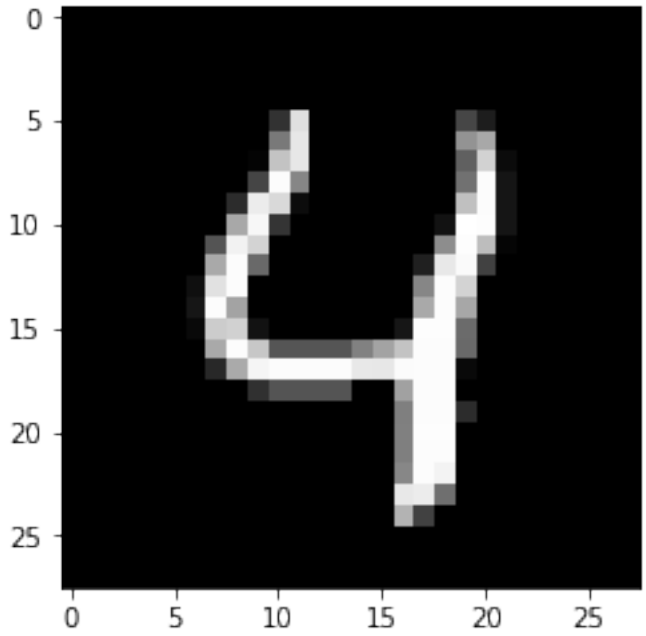

```
Model prediction: 2
```



```
Model prediction: 1
```




```
Model prediction: 0
```



```
Model prediction: 4
```