hw2_p4

February 16, 2018

In [1]:
```
import numpy as np
import tensorflow as tf
%matplotlib inline
import matplotlib
import matplotlib.pyplot as plt
```

In [2]:
```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=False)
print(mnist.train.images.shape, mnist.train.labels.shape)
```

```
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
(55000, 784) (55000,)
```

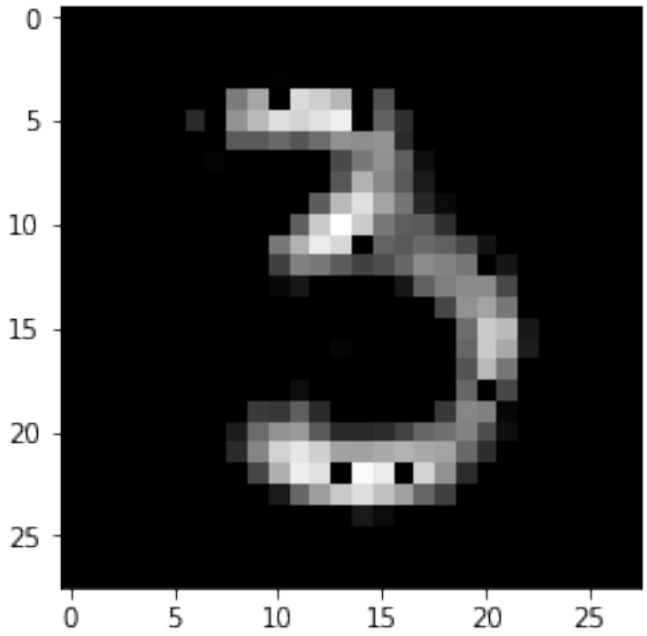
In [3]:
```
def show_image(image):
    plt.gray()
    plt.imshow(image.reshape(28, 28))
    plt.show()

# add gaussian noise to image -- mean = 0, variance = 0.1
def add_noise(image):
    mu, sigma = 0, 0.1
    gauss = np.random.normal(mu, sigma, 28*28)
    return image + gauss
```

In [4]:
```
# This is an encoder
# 2 convolutional layers and 1 fully connected layer
def encoder(x):
    x_image = tf.reshape(x, [-1, 28, 28, 1])

    conv1 = tf.contrib.layers.conv2d(x_image, 16, [3,3], stride=2, padding='VALID')

    conv2 = tf.contrib.layers.conv2d(conv1, 32, [3,3], stride=2, padding='VALID')

    pool2_flat = tf.reshape(conv2, [-1, 6*6*32])
```

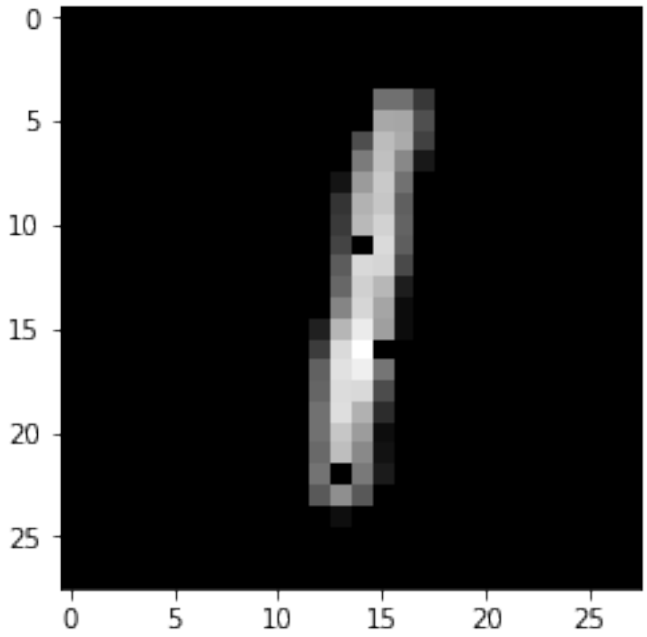
```
        fc = tf.contrib.layers.fully_connected(pool2_flat, 100)

        return fc

    # This is a decoder -- well commented!
    # 1 fully connected layer, 2 convolutional layers and 1 fc layer
    def decoder(x):
        fc = tf.contrib.layers.fully_connected(x, 6*6*32)
        fc_layer = tf.reshape(fc, [-1, 6, 6, 32])

        deconv1 = tf.contrib.layers.conv2d_transpose(fc_layer, 16, [3,3], stride=2, padding=
        deconv2 = tf.contrib.layers.conv2d_transpose(deconv1, 1, [3,3], stride=2, padding='V
        deconv2_flat = tf.reshape(deconv2, [-1, 27*27])
        fc = tf.contrib.layers.fully_connected(deconv2_flat, 28*28)
        return fc
```

In [5]:
```
with tf.device('/device:GPU:0'):
    # noisy image
    x = tf.placeholder(tf.float32, shape=[None, 784], name='x')
    # origin image (before adding noise)
    y = tf.placeholder(tf.float32, shape=[None, 784], name='y')

# reconstruct images
pred = decoder(encoder(x))
# MSE cost function
#compute difference between reconstructed images and imges before adding noise
cost = tf.reduce_mean(tf.squared_difference(y, pred))
optimizer = tf.train.AdamOptimizer(1e-4).minimize(cost)
```

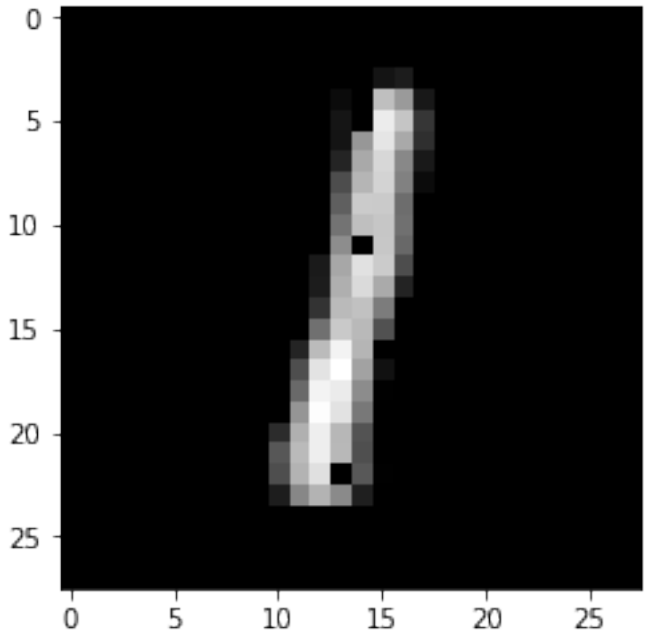
In [6]:
```
epoch = 15000
batch_size = 16
config = tf.ConfigProto()
config.gpu_options.allow_growth = True

sess = tf.Session(config=config)
sess.run(tf.global_variables_initializer())

for i in range(epoch):
    batch_x, _ = mnist.train.next_batch(batch_size)
    _, batch_cost = sess.run([optimizer, cost], feed_dict = {x:add_noise(batch_x), y:bat

    # show reconstructed images after every 5000 iterations of training
    if (i+1) % 5000 == 0:
        batch_x, _ = mnist.test.next_batch(1)
        noise_image = add_noise(batch_x)
        gen_image, test_cost = sess.run([pred,cost], feed_dict = {x:noise_image, y:batch
        #show_image(noise_image)
        show_image(gen_image)
        print('%d epochs-- train cost: %f    test cost: %f' % (i+1, batch_cost, test_cos
```




```
5000 epochs-- train cost: 0.015710    test cost: 0.017580
```



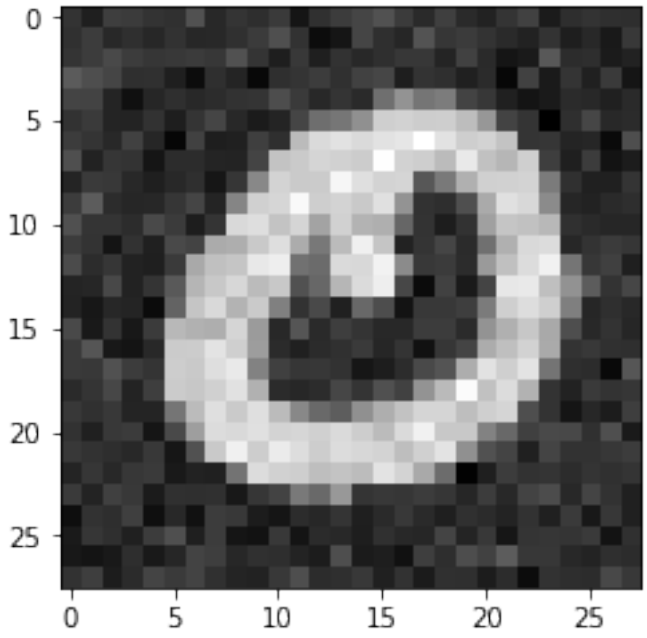
```
10000 epochs-- train cost: 0.013543    test cost: 0.005779
```



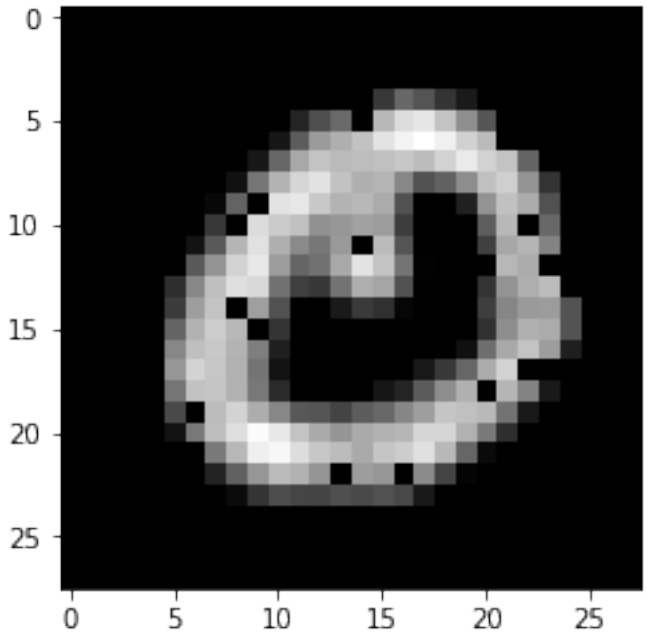

```
15000 epochs-- train cost: 0.014058    test cost: 0.006856
```

```
In [7]: # Testing -- reconstruct images from test set
        batch_x, _ = mnist.test.next_batch(1)
        noise_image = add_noise(batch_x)
        gen_image, test_cost = sess.run([pred,cost], feed_dict = {x:noise_image, y:batch_x})
        print('Noisy image')
        show_image(noise_image)
        print('Generated image')
        show_image(gen_image)
```

```
Noisy image
```

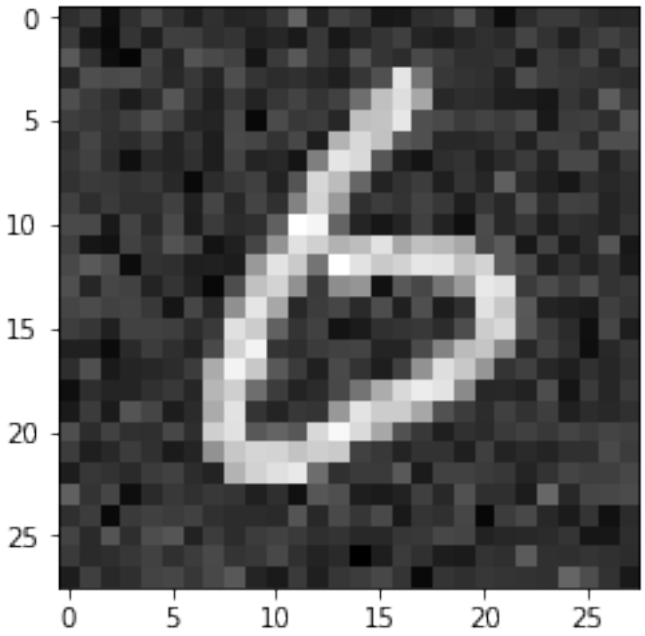


Generated image

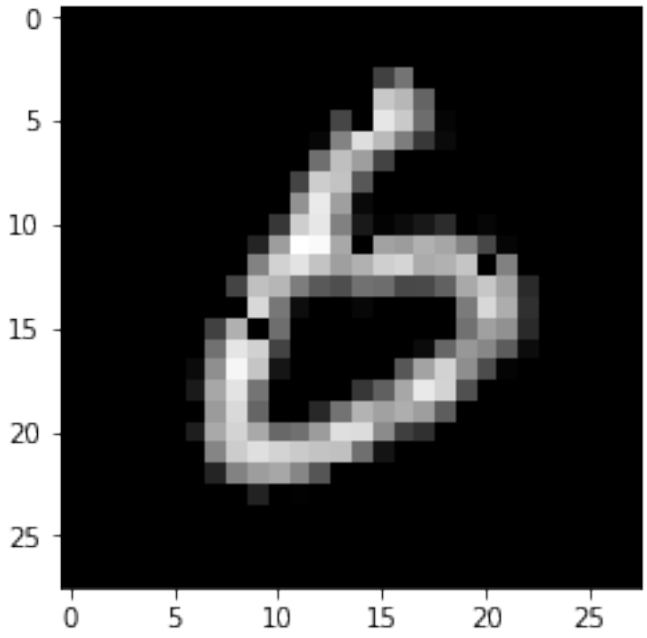


In [8]:
```
# Testing -- reconstruct images from test set
batch_x, _ = mnist.test.next_batch(1)
noise_image = add_noise(batch_x)
gen_image, test_cost = sess.run([pred,cost], feed_dict = {x:noise_image, y:batch_x})
print('Noisy image')
show_image(noise_image)
print('Generated image')
show_image(gen_image)
```

```
Noisy image
```




```
Generated image
```


0
5
10
15
20
25
0
5
10
15
20
25