hw2_p1

February 16, 2018

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

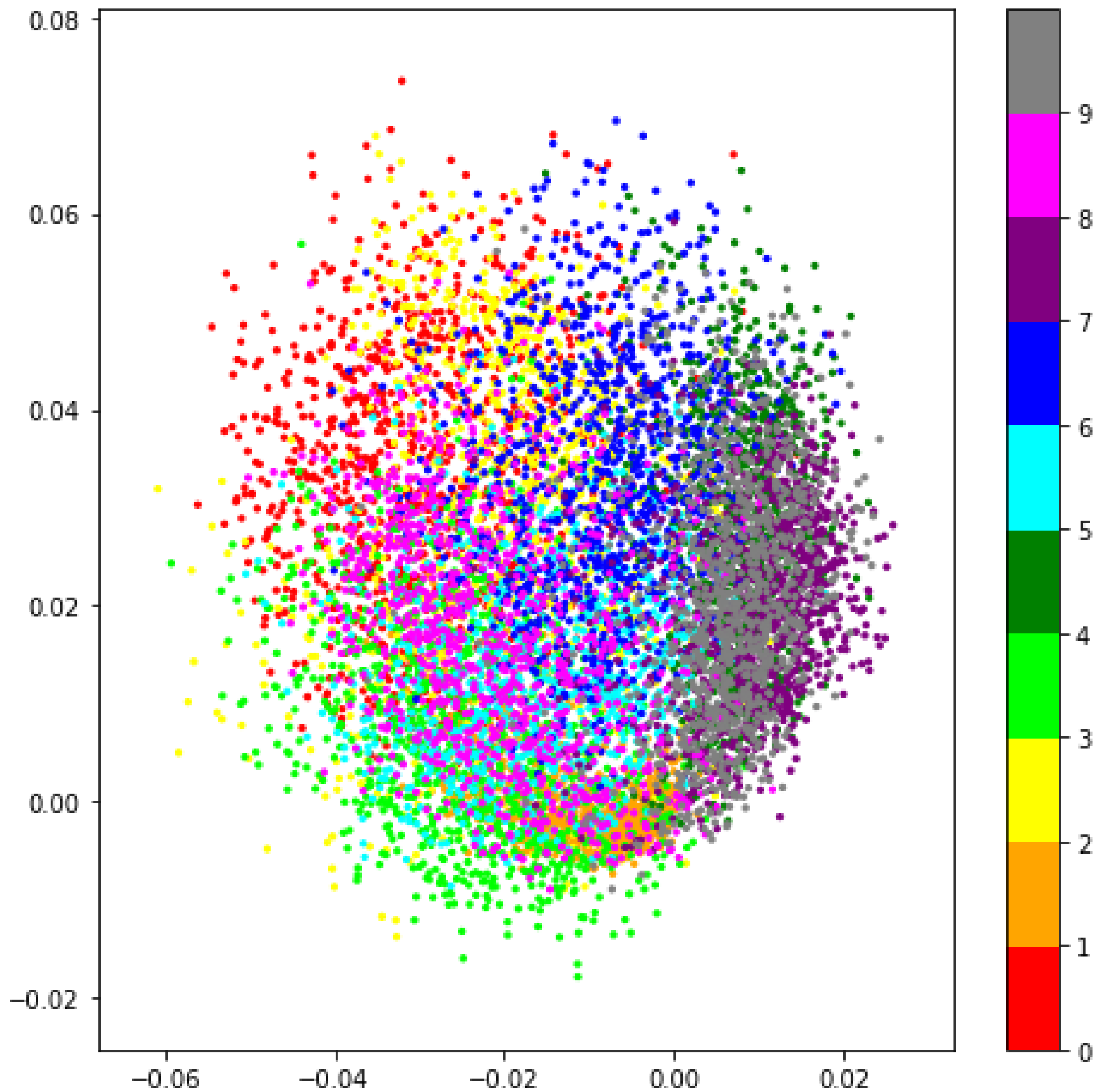
In [2]:
```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=False)
print(mnist.train.images.shape, mnist.train.labels.shape)
```

```
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 showimage(image, label):
    plt.gray()
    plt.imshow(image.reshape(28, 28))
    plt.show()
    print(label)

def extract_target_data(X, Y, target, num):
    p = (Y == target)
    x_target = X[p,:]
    y_target = Y[p]
    y_target = np.expand_dims(y_target, axis=1)
    return x_target[:num,:], y_target[:num,:]

# computer euclidean distance matrix
def euclidean_distance_matrix(x):
    r = tf.reduce_sum(x*x, 1)
    r = tf.reshape(r, [-1, 1])
    distance_mat = r - 2*tf.matmul(x, tf.transpose(x)) + tf.transpose(r)
    #return tf.sqrt(distance_mat)
    return distance_mat
```

```
In [4]: train_images, train_labels = extract_target_data(mnist.train.images, mnist.train.labels,

        # get 1000 data from each category
        for i in range(1, 10):
            cur_train_images, cur_train_labels = extract_target_data(mnist.train.images, mnist.t
            train_images = np.concatenate((train_images, cur_train_images), axis=0)
            train_labels = np.concatenate((train_labels, cur_train_labels), axis=0)

        train_images /= 255.
        #print(train_images.shape, train_labels.shape)

In [5]: distance_mat = euclidean_distance_matrix(train_images)
        with tf.Session() as sess:
            M = sess.run(distance_mat)

In [6]: with tf.device('/device:GPU:0'):
            x = tf.placeholder(tf.float32, shape=[None, 784], name='x')
            w = tf.get_variable('w', shape=[784, 2], initializer=tf.contrib.layers.xavier_initia
            b = tf.get_variable('b', shape=[2], initializer=tf.zeros_initializer())

            z = tf.matmul(x, w) + b
            M_ = euclidean_distance_matrix(z)
            # MSE cost function
            cost = tf.reduce_mean(tf.square(M_ - M))
            optimizer = tf.train.AdamOptimizer().minimize(cost)

In [7]: epoch = 1000
        config = tf.ConfigProto()
        config.gpu_options.allow_growth = True
        with tf.Session(config=config) as sess:
            sess.run(tf.global_variables_initializer())
            for i in range(epoch):
                _, epoch_cost, transform_mat = sess.run([optimizer, cost, w], feed_dict = {x:tra

In [8]: emb_images = np.dot(train_images, transform_mat)
        print(emb_images.shape)

(10000, 2)


In [9]: x = emb_images[:,0]
        y = emb_images[:,1]
        label = train_labels
        colors = ['red','orange','yellow','lime','green','cyan','blue','purple','magenta','grey'

        fig = plt.figure(figsize=(8,8))
        plt.scatter(x, y, c=label, cmap=matplotlib.colors.ListedColormap(colors),s=5)

        cb = plt.colorbar()
```

```
loc = np.arange(0,max(label),max(label)/float(len(colors)))
cb.set_ticks(loc)
cb.set_ticklabels(list(range(10)))
```



We can see from the image above, the datapoints of same category are clustering together. This infers the embedding, even though with only 2 dimensions, perserve essential distance information from 786 dimensional matrix and are able to dissimilate from other categories.