hw2q1_multidimensional_scaling

February 14, 2018

In [1]:
```
import matplotlib
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import time
import sklearn
from tensorflow.examples.tutorials.mnist import input_data
from sklearn.metrics import euclidean_distances
```

In [2]:
```
#Read Data
mnist = input_data.read_data_sets(".",one_hot = True)
```

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

In [3]:
```
def TRAIN_SIZE(num):
    print ('Total Training Images in Dataset = ' + str(mnist.train.images.shape))
    print ('-----------------------------------------------------')
    x_train = mnist.train.images[:num,:]
    print ('x_train Examples Loaded = ' + str(x_train.shape))
    y_train = mnist.train.labels[:num,:]
    print ('y_train Examples Loaded = ' + str(y_train.shape))
    print('')
    return x_train, y_train

def display_digit(num):
    print(y_train[num])
    label = y_train[num].argmax(axis=0)
    image = x_train[num].reshape([28,28])
    plt.title('Example: %d  Label: %d' % (num, label))
    plt.imshow(image, cmap='binary')
    plt.show()

def squared_dist(A):
#     expanded_a = tf.expand_dims(A, 1)
```

```
        #     expanded_b = tf.expand_dims(A, 0)
        #     distances = tf.reduce_sum(tf.squared_difference(expanded_a, expanded_b), 2)
            r = tf.reduce_sum(A*A, 1)
            r = tf.reshape(r, [-1, 1])
            distances = r - 2*tf.matmul(A, tf.transpose(A)) + tf.transpose(r)
            return distances
```

In [4]:
```
sample_size = 10000
X_train, Y_train = TRAIN_SIZE(sample_size)
```

```
Total Training Images in Dataset = (55000, 784)
-------------------------------------------------
x_train Examples Loaded = (10000, 784)
y_train Examples Loaded = (10000, 10)
```

In [5]:
```
#Placeholder For Input Image
D = tf.placeholder(tf.float32, [None, None])

#Initialize X' weight0
X_proj = tf.get_variable('x_proj', [sample_size,2], initializer = tf.random_normal_initi
```

In [6]:
```
D_proj = squared_dist(X_proj)

#Loss Function
# loss = tf.reduce_sum(0.5*tf.square(D-D_proj))
loss = tf.reduce_mean(tf.square(D-D_proj))

#Define Optimizer
optimize = tf.train.AdamOptimizer(learning_rate=0.8).minimize(loss)
```

In [7]:
```
sess = tf.InteractiveSession()
tf.global_variables_initializer().run()

# sample_size = 100
num_iter = 150
loss_val = np.zeros(num_iter)
# X_train = X_train/255
D_train = euclidean_distances(X_train)**2
# D_train = euclidean_distances(X_train)

for i in range(num_iter):
    _, l, X_proj_train = sess.run([optimize, loss, X_proj], feed_dict = {D: D_train})
#     print(X_proj_train)
#     print(sess.run(D_proj))

    if i==(num_iter-1):
        X_proj_final = X_proj_train
```

```
            loss_val[i] = l

            print('epoch', i, 'loss:', loss_val[i])
```

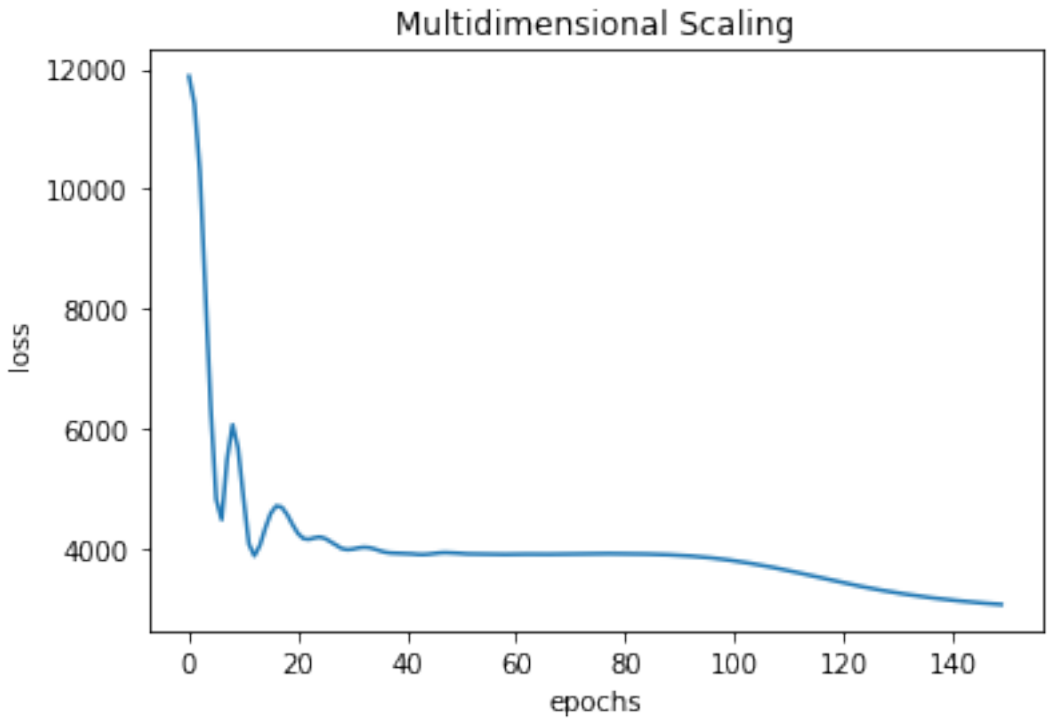
```
epoch 0 loss: 11881.0712890625
epoch 1 loss: 11423.41015625
epoch 2 loss: 10292.5615234375
epoch 3 loss: 8315.9296875
epoch 4 loss: 6344.02783203125
epoch 5 loss: 4823.84228515625
epoch 6 loss: 4484.501953125
epoch 7 loss: 5494.27880859375
epoch 8 loss: 6067.34521484375
epoch 9 loss: 5678.3759765625
epoch 10 loss: 4847.1142578125
epoch 11 loss: 4077.478759765625
epoch 12 loss: 3880.261474609375
epoch 13 loss: 4052.53955078125
epoch 14 loss: 4343.06201171875
epoch 15 loss: 4589.609375
epoch 16 loss: 4711.35693359375
epoch 17 loss: 4693.97802734375
epoch 18 loss: 4570.5693359375
epoch 19 loss: 4401.36767578125
epoch 20 loss: 4252.1240234375
epoch 21 loss: 4168.94921875
epoch 22 loss: 4156.25390625
epoch 23 loss: 4180.1953125
epoch 24 loss: 4193.85546875
epoch 25 loss: 4169.32470703125
epoch 26 loss: 4112.07861328125
epoch 27 loss: 4049.353515625
epoch 28 loss: 4003.20068359375
epoch 29 loss: 3983.74365234375
epoch 30 loss: 3991.379638671875
epoch 31 loss: 4011.103271484375
epoch 32 loss: 4023.525634765625
epoch 33 loss: 4017.76220703125
epoch 34 loss: 3994.924560546875
epoch 35 loss: 3965.05126953125
epoch 36 loss: 3940.4306640625
epoch 37 loss: 3927.657470703125
epoch 38 loss: 3924.955322265625
epoch 39 loss: 3924.896240234375
epoch 40 loss: 3920.978515625
epoch 41 loss: 3912.5810546875
epoch 42 loss: 3904.369384765625
epoch 43 loss: 3902.290771484375
```

```
epoch 44 loss: 3908.322265625
epoch 45 loss: 3919.338623046875
epoch 46 loss: 3929.5009765625
epoch 47 loss: 3934.173828125
epoch 48 loss: 3932.220947265625
epoch 49 loss: 3925.6083984375
epoch 50 loss: 3918.339111328125
epoch 51 loss: 3913.1953125
epoch 52 loss: 3911.254638671875
epoch 53 loss: 3910.821044921875
epoch 54 loss: 3909.976806640625
epoch 55 loss: 3907.68359375
epoch 56 loss: 3905.083251953125
epoch 57 loss: 3903.70947265625
epoch 58 loss: 3904.48583984375
epoch 59 loss: 3906.5517578125
epoch 60 loss: 3908.5166015625
epoch 61 loss: 3909.264892578125
epoch 62 loss: 3908.4248046875
epoch 63 loss: 3906.85595703125
epoch 64 loss: 3905.64013671875
epoch 65 loss: 3905.671630859375
epoch 66 loss: 3906.681640625
epoch 67 loss: 3907.93994140625
epoch 68 loss: 3909.205322265625
epoch 69 loss: 3910.433837890625
epoch 70 loss: 3911.881591796875
epoch 71 loss: 3913.8125
epoch 72 loss: 3915.720947265625
epoch 73 loss: 3917.25244140625
epoch 74 loss: 3918.055908203125
epoch 75 loss: 3918.000244140625
epoch 76 loss: 3917.563720703125
epoch 77 loss: 3917.0615234375
epoch 78 loss: 3916.568359375
epoch 79 loss: 3916.033203125
epoch 80 loss: 3915.186767578125
epoch 81 loss: 3914.2109375
epoch 82 loss: 3912.9697265625
epoch 83 loss: 3911.55419921875
epoch 84 loss: 3909.8876953125
epoch 85 loss: 3907.935546875
epoch 86 loss: 3905.431396484375
epoch 87 loss: 3902.283447265625
epoch 88 loss: 3898.3818359375
epoch 89 loss: 3893.902587890625
epoch 90 loss: 3888.794189453125
epoch 91 loss: 3883.24658203125
```

```
epoch 92 loss: 3877.147216796875
epoch 93 loss: 3870.11083984375
epoch 94 loss: 3862.382568359375
epoch 95 loss: 3853.851806640625
epoch 96 loss: 3844.526123046875
epoch 97 loss: 3834.411865234375
epoch 98 loss: 3823.057861328125
epoch 99 loss: 3810.840087890625
epoch 100 loss: 3797.934326171875
epoch 101 loss: 3784.199462890625
epoch 102 loss: 3769.807373046875
epoch 103 loss: 3754.731689453125
epoch 104 loss: 3739.11376953125
epoch 105 loss: 3722.978271484375
epoch 106 loss: 3706.30078125
epoch 107 loss: 3688.97021484375
epoch 108 loss: 3671.024169921875
epoch 109 loss: 3652.469482421875
epoch 110 loss: 3633.285400390625
epoch 111 loss: 3613.459716796875
epoch 112 loss: 3593.19140625
epoch 113 loss: 3572.601318359375
epoch 114 loss: 3552.14697265625
epoch 115 loss: 3531.32763671875
epoch 116 loss: 3510.9169921875
epoch 117 loss: 3490.660400390625
epoch 118 loss: 3470.5234375
epoch 119 loss: 3450.576171875
epoch 120 loss: 3430.781005859375
epoch 121 loss: 3411.646240234375
epoch 122 loss: 3392.946533203125
epoch 123 loss: 3374.72314453125
epoch 124 loss: 3356.779541015625
epoch 125 loss: 3339.3271484375
epoch 126 loss: 3322.60107421875
epoch 127 loss: 3306.300048828125
epoch 128 loss: 3290.5458984375
epoch 129 loss: 3275.457763671875
epoch 130 loss: 3260.658203125
epoch 131 loss: 3246.624267578125
epoch 132 loss: 3232.91259765625
epoch 133 loss: 3219.752197265625
epoch 134 loss: 3207.0517578125
epoch 135 loss: 3194.79345703125
epoch 136 loss: 3182.940673828125
epoch 137 loss: 3171.673583984375
epoch 138 loss: 3160.875732421875
epoch 139 loss: 3150.601806640625
```

```
epoch 140 loss: 3140.72509765625
epoch 141 loss: 3131.31689453125
epoch 142 loss: 3122.27197265625
epoch 143 loss: 3113.647705078125
epoch 144 loss: 3105.26806640625
epoch 145 loss: 3097.307373046875
epoch 146 loss: 3089.726806640625
epoch 147 loss: 3082.5625
epoch 148 loss: 3075.63330078125
epoch 149 loss: 3069.082275390625
```

In [8]:
```
plt.plot(np.arange(0,num_iter,1), loss_val)
plt.xlabel('epochs')
plt.ylabel('loss')
plt.title('Multidimensional Scaling')
plt.show()
```



In [9]:
```
x_coord = X_proj_final[:,0]
y_coord = X_proj_final[:,1]

color_list = ['C0','C1','C2','C3','C4','C5','C6','C7','C8','C9']
colors = []
```

```
for i in range(sample_size):
    colors.append(color_list[Y_train[i].argmax(axis=0)])

plt.figure(figsize = (6, 6))
plt.scatter(x_coord, y_coord, s=3, c=colors)
plt.show()
```