assignment04

April 4, 2019

1 Load MNIST training dataset

In [1]:
```
#20160040 Ko Hoyun
#Visualize average images(L2-norm)
import matplotlib.pyplot as plt
import numpy as np
import random

#my file data path
file_data = "C:\\Users\\recognize_data\\mnist_train.csv"
handle_file = open(file_data, "r")

#read data with line
data = handle_file.readlines()
handle_file.close()
```

In [2]:
```
#image size
size_row = 28    # height of the image
size_col = 28    # width of the image

num_image = len(data)
count = 0     # count for the number of images
```

In [3]:
```
# normalize the values of the input data to be [0, 1]
def normalize(data):

    data_normalized = (data - min(data)) / (max(data) - min(data))

    return(data_normalized)

#function for squaring and
def square(x):
    d = x ** 2
    s = d
    return(s)

#l2 norm(squre was done)
```

```
        def l2_norm(x):

            s = np.sum(x)
            r = np.sqrt(s)

            return(r)

        #distance
        def distance(x,y):
            d = sum((x-y) ** 2)
            #s = np.sqrt(d)
            return(d)
```

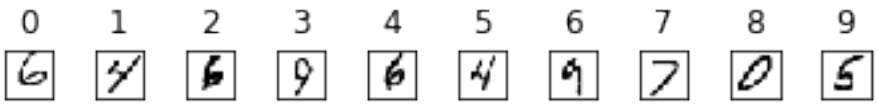
In [4]:
```
        # num of K
        k = 10
        #centroid images
        centroidImage  = [[0 for col in range(1)] for row in range(k)]

        for i in range(0,k):

            centroidImage[i][0] = np.asfarray(data[random.randrange(0,num_image)].split(',')[1
            centroidImage[i][0] = normalize(centroidImage[i][0])
```

In [5]:
```
        # insert centroid image value
        list_image  = np.empty((size_row * size_col, num_image), dtype=float)
        for i in range(0,k):
            list_image[:, i]    = centroidImage[i][0]
```

2 Visualize the centroid images.

In [7]:
```
        #
        # plot images
        #
        f1 = plt.figure(1)

        for i in range(0,10):

            label       = i
            centroidImage[i][0]   = list_image[:, i]
            im_matrix   = centroidImage[i][0].reshape((size_row, size_col))

            plt.subplot(10, 10, i+1)
            plt.title(label)
            plt.imshow(im_matrix, cmap='Greys', interpolation='None')

            frame   = plt.gca()
            frame.axes.get_xaxis().set_visible(False)
            frame.axes.get_yaxis().set_visible(False)
```

```
plt.show()
```




```
In [11]: def classify(a):
             return min(range(k),
                        key = lambda i:distance(a,centroidImage[i][0]))
```

```
In [23]: # make a matrix each column of which represents an images in a vector form
         list_image  = np.empty((size_row * size_col, num_image), dtype=float)
         list_label  = np.empty(num_image, dtype=int)

         #2-D array for putting square values of same group
         collect = [[0 for col in range(1)] for row in range(k)]
         #for average. divisor, number of kth element
         n = np.empty(k, dtype = int)

         for line in data:

             #the number of lables is at the front. so split and put it into lable value.
             line_data   = line.split(',')
             label       = line_data[0]

             #remain values put in the im_vector
             im_vector   = np.asfarray(line_data[1:])
             im_vector   = normalize(im_vector)

         #    list_label[count]       = label
         #    list_image[:, count]    = im_vector

             minvalue = classify(im_vector)
             collect[minvalue].append(im_vector)
             n[minvalue] += 1
             list_label[count] = minvalue

         for i in range(0,k):
             del collect[i][0]
         #    count += 1
```

3 energy function

$\sum_{k=1}^{K} \|x_i - c_{k_i}\|^2$

```
In [24]: energy = 0
         for i in range(0, num_image):
             energy += distance(collect[i], centroidImage[i][0])


         ---------------------------------------------------------------------------

         IndexError                                Traceback (most recent call last)

         <ipython-input-24-4476c124a1a3> in <module>()
           1 energy = 0
           2 for i in range(0, num_image):
         ----> 3     energy += distance(collect[i], centroidImage[i][0])


         IndexError: list index out of range


In [ ]: print('energy: ',energy)
```

4 Accuracy

$\frac{\sum_{k=1}^{K} m_k}{N}$

```
In [30]: # for most frequent number
         accuracy = 0
         from collections import Counter
         for i in range(0,k):
             cnt = Counter(collect[i])
             cnt.most_common(1)
             accuracy += cnt[0][0]


         ---------------------------------------------------------------------------

         TypeError                                 Traceback (most recent call last)

         <ipython-input-30-6fc82b862633> in <module>()
           3 from collections import Counter
           4 for i in range(0,k):
         ----> 5     cnt = Counter(collect[i])
           6     cnt.most_common(1)
           7     accuracy += cnt[0][0]


         C:\ProgramData\Anaconda3\lib\collections\__init__.py in __init__(*args, **kwds)
         564             raise TypeError('expected at most 1 arguments, got %d' % len(args))
         565         super(Counter, self).__init__()
```

```
--> 566         self.update(*args, **kwds)
    567
    568     def __missing__(self, key):


    C:\ProgramData\Anaconda3\lib\collections\__init__.py in update(*args, **kwds)
    651                     super(Counter, self).update(iterable) # fast path when counter
    652             else:
--> 653                 _count_elements(self, iterable)
    654         if kwds:
    655             self.update(kwds)


    TypeError: unhashable type: 'numpy.ndarray'
```

```
In [26]: print(accuracy)
```

```
0
```

```
In [ ]:
```