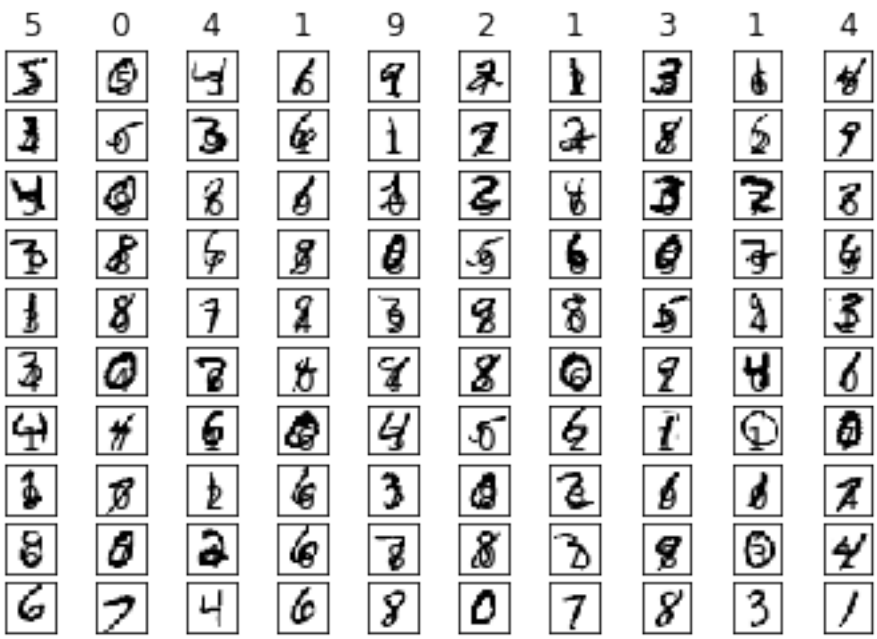
Assignment03_20133096_HyunjaeLee

March 26, 2019

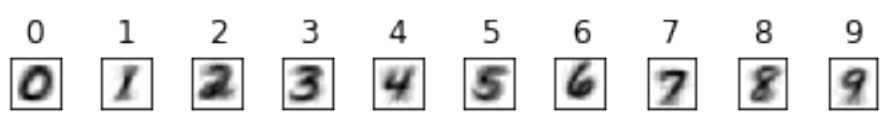
In [186]:
```
#20133096 Hyunjae Lee
'''
1. Write codes with detailed comments and present useful results at Jupyter Notebook.
2. Export the Jupyter Notebook file as a PDF file.
3. Submit the PDF file to Classroom.

[Visualize average images]

1. Load MNIST training dataset.
2. Compute the average images for each label (digit) based on L2-norm.
3. Visualize the average images.
'''

import matplotlib.pyplot as plt
import numpy as np

file_data                = "mnist_train.csv"
handle_file        = open(file_data, "r")
data                       = handle_file.readlines()
handle_file.close()

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

#
# 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)

#
```

```python
# example of distance function between two vectors x and y
#
def distance(x, y):

    d = (x - y) ** 2
    s = np.sum(d)
    # r = np.sqrt(s)

    return(s)

#
# 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)

for line in data:

    line_data   = line.split(',')
    label       = line_data[0]
    im_vector   = np.asfarray(line_data[1:])
    im_vector   = normalize(im_vector)

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

    count += 1

#
# plot first 100 images out of 10,000 with their labels
#
f1 = plt.figure(1)

for i in range(100):

    label       = list_label[i]
    im_vector   = list_image[:, i]
    im_matrix   = im_vector.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 [187]:
```
#
# Square
#
def square(data):
    data_squared = data ** 2
    return (data_squared)
#
# Root
#
def root(data):
    data_root = data ** 0.5
    return (data_root)



# create one list that contains 10 lists for each Integer
# and the other one that contains how many imaged each label has
N = [list() ] * 10
cnt = list()
# initialize two lists
for i in range(0,10):
    N[i] = [0.0]*784
    cnt.append(0)

count = 0
j = 0
```

```
# L2-norm : root(x1^2 + ... + xn^2) / n
# (1) x1^2 + ... + xn^2
for item in list_image:
    for j in range(0, 10):
        if(list_label[count] == j):
        # add square value of each element
            N[j] += square(list_image[:, count])
        # the number of 0 label image
            cnt[j] += 1
    count += 1

# (2) root(sum of square of each element) / n
j = 0
for j in range(0,10):
    N[j] = root(N[j])/cnt[j]

f2 = plt.figure(2)
j = 0
for j in range(0,10):
    im_matrix   = N[j].reshape((size_row, size_col))

    plt.subplot(10, 10, j+1)
    plt.title(j)
    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)
    j += 1
plt.show()
```



In [ ]: