20160040_assignment 10

June 6, 2019

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import copy
from sklearn.metrics import confusion_matrix

#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()

#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 [2]:
```
#
# make a matrix each column of which represents an images in a vector form
#
list_image  = np.zeros((num_image, size_row * size_col), dtype=float)
list_label  = np.zeros(num_image, dtype=int)

count = 0
for line in data:
    #the number of lables is at the front. so split and put it into lable value.
    line_data   = line.split(',')
    list_label[count] = line_data[0]
    list_image[count] = np.asfarray(line_data[1:])
    count += 1
```

In [32]:
```
#my file data path
 file_data = "C:\\Users\\recognize_data\\mnist_test.csv"
```

```
    handle_file = open(file_data, "r")

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

    t_list_image  = np.zeros((t_num_image, size_row * size_col), dtype=float)
    t_list_label  = np.zeros(t_num_image, dtype=int)

    t_num_image = len(data)
    count = 0
    for line in data:
        #the number of lables is at the front. so split and put it into lable value.
        line_data   = line.split(',')
        t_list_label[count] = line_data[0]
        t_list_image[count] = np.asfarray(line_data[1:])
        count += 1
```

In [4]:
```
#make matrix
matrix = np.zeros((num_image, size_row * size_col+1), dtype=float)
for i in range(num_image):
    for j in range(size_row * size_col+1):
        if(j == 0):
            matrix[i,j] = 1
        else:
            matrix[i,j] = list_image[i,j-1]
```

In [5]:
```
# assign y value
def assign_y_value(index):
    y = np.zeros((num_image,1), dtype=float)
    count = 0
    for i in list_label:
        if(i == index):
            y[count] = 1
        else:
            y[count] = -1
        count += 1
    return y
```

1 Compute an optimal model parameter using the training dataset(seta)

In [6]:
```
def make_seta(y):
    seta = np.zeros((size_row * size_col+1, 1), dtype=float)
    values = copy.deepcopy(np.linalg.pinv((np.mat(matrix.T)*np.mat(matrix)))*np.mat(mat
    seta = np.ravel(values)
```

```
            seta = np.reshape(np.array(seta),(size_row*size_col +1,1))

            return seta

        def estimation(seta):

            estimation = np.zeros((num_image,1), dtype=float)
            for i in range(num_image):
                for j in range(size_row * size_col+1):
                    estimation[i] += matrix[i][j]*seta[j]
            return estimation
```

```
In [7]: y_values = np.zeros((10, num_image,1), dtype=float)
        setas = np.zeros((10, size_row * size_col+1, 1), dtype=float)

        for i in range(10):
            y_values[i] = copy.deepcopy(assign_y_value(i))
            setas[i] = copy.deepcopy(make_seta(y_values[i]))
```

```
In [8]: estimations = np.zeros((10, num_image,1), dtype=float)
        matrix = np.reshape(np.array(matrix),(num_image,size_row*size_col +1))
        for i in range(10):
            estimations[i] = copy.deepcopy(estimation(setas[i]))
```

```
In [9]: esti_label = np.zeros((num_image,1), dtype=int)
        for i in range(num_image):
            esti_label[i] = np.argmax(estimations[:,i])
```

2 Compute TPR, ERR- Traning dataset

```
In [10]: c_matrix = np.zeros((10,10), dtype = int)
         c_matrix = copy.deepcopy(confusion_matrix(list_label,esti_label))
         print(c_matrix)
```

```
[[5682    7   18   14   24   43   64    4   61    6]
 [   2 6548   40   15   19   31   14   12   55    6]
 [  99  264 4792  149  108   11  234   91  192   18]
 [  42  167  176 5158   32  125   56  115  135  125]
 [  10   99   42    6 5212   50   39   23   59  302]
 [ 164   95   28  432  105 3991  192   36  235  143]
 [ 108   74   61    1   70   90 5476    0   35    3]
 [  55  189   37   47  170    9    2 5426   10  320]
 [  75  493   63  226  105  221   56   20 4412  180]
 [  68   60   20  117  371   12    4  492   38 4767]]
```

```
In [11]: tpr_cnt = 0;
         err_cnt = 0;
```

```
        for i in range(10):
            for j in range(10):
                if(i==j):
                    tpr_cnt += c_matrix[i][j]

                else:
                    err_cnt += c_matrix[i][j]


        tpr_rate = tpr_cnt/num_image
        err_rate = err_cnt/num_image
```

3 Result

```
In [12]: print("true positive rate:", tpr_rate)
         print("error rate:", err_rate)
```

```
true positive rate: 0.8577333333333333
error rate: 0.14226666666666668
```

4 test

```
In [29]: t_estimations = np.zeros((10, num_image,1), dtype=float)
         for k in range(10):
             for i in range(t_num_image):
                 for j in range(size_row * size_col+1):
                     if(j == 0):
                         t_estimations[k][i] += setas[k][j]
                     else:
                         t_estimations[k][i] += t_list_image[i][j-1]*setas[k][j]
```

```
In [30]: t_esti_label = np.zeros((t_num_image,1), dtype=int)

         for i in range(t_num_image):
             t_esti_label[i] = np.argmax(t_estimations[:,i])
```

5 Compute TPR, ERR - Testing dataset

```
In [33]: t_c_matrix = copy.deepcopy(confusion_matrix(t_list_label,t_esti_label))
```

```
In [34]: t_tpr_cnt = 0;
         t_err_cnt = 0;
         for i in range(10):
             for j in range(10):
                 if(i==j):
                     t_tpr_cnt += t_c_matrix[i][j]
```

```
                else:
                    t_err_cnt += t_c_matrix[i][j]


        t_tpr_rate = t_tpr_cnt/t_num_image
        t_err_rate = t_err_cnt/t_num_image
```

In [36]:
```
print(t_c_matrix)
```

```
[[ 944    0    1    2    2    7   14    2    7    1]
 [   0 1107    2    2    3    1    5    1   14    0]
 [  18   54  813   26   15    0   42   22   37    5]
 [   4   17   23  880    5   17    9   21   22   12]
 [   0   22    6    1  881    5   10    2   11   44]
 [  23   18    3   72   24  659   23   14   39   17]
 [  18   10    9    0   22   17  875    0    7    0]
 [   5   40   16    6   26    0    1  884    0   50]
 [  14   46   11   30   27   40   15   12  759   20]
 [  15   11    2   17   80    1    1   77    4  801]]
```

6 Result

In [35]:
```
print("true positive rate:", t_tpr_rate)
print("error rate:", t_err_rate)
```

```
true positive rate: 0.8603
error rate: 0.1397
```

In [ ]:

In [ ]: