assignment10

December 6, 2018

1 This is assignment10

2 Name:PENG CIYUAN

3 Student ID:2018220161

4 Link:https://github.com/pcyyyy/assignment10.git

5 Getting train data and test data

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
from scipy import signal
from sklearn.metrics import confusion_matrix
file_data_train = "mnist_train.csv"
file_data_test  = "mnist_test.csv"

h_data_train    = open(file_data_train, "r")
h_data_test     = open(file_data_test, "r")

data_train      = h_data_train.readlines()
data_test       = h_data_test.readlines()

h_data_train.close()
h_data_test.close()

size_row    = 28    # height of the image
size_col    = 28    # width of the image

num_train   = len(data_train)   # number of training images
num_test    = len(data_test)    # number of testing 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)

#
# 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_train    = np.empty((size_row * size_col, num_train), dtype=float)
list_label_train    = np.empty(num_train, dtype=int)

list_image_test     = np.empty((size_row * size_col, num_test), dtype=float)
list_label_test     = np.empty(num_test, dtype=int)

count = 0

for line in data_train:

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

    list_label_train[count]     = label
    list_image_train[:, count]  = im_vector



    count += 1

count = 0

for line in data_test:

    line_data   = line.split(',')
    label       = line_data[0]
```

```
            im_vector    = np.asfarray(line_data[1:])
            im_vector    = normalize(im_vector)

            list_label_test[count]      = label
            list_image_test[:, count]   = im_vector

            count += 1
```

6 define function to get R

```
In [2]: def Matrix(p):
            R=[]
            for i in range(p):
                r = np.random.normal(0,1,784)
                R.append(r)
            R= np.array(R)
            R = R.T
            return R
```

7 Compute the weight A,B

```
In [3]: R = Matrix(27)
        trainX = np.matmul(list_image_train.T,R)
        trainY = list_label_train
        A1=np.linalg.inv((trainX.T).dot(trainX))
        A2=(trainX.T).dot(trainY)
        A=(A1).dot(A2)
        RR = Matrix(50)
        trainX1 = np.matmul(list_image_train.T,RR)
        trainY1 = list_label_train
        B1=np.linalg.inv((trainX1.T).dot(trainX1))
        B2=(trainX1.T).dot(trainY1)
        B=(B1).dot(B2)
```

8 Get classification result

```
In [4]: testX = np.matmul(list_image_test.T,R)
        result =np.dot(testX,A)
        testX1 = np.matmul(list_image_test.T,RR)
        result1 =np.dot(testX1,B)
```

9 Present confusion matrix M

```
In [5]: def resultandY(result, list_label_test):
            m = []
```

```
            Y = []
            for i in range(len(result)):
                m.append(abs(int(result[i])))
                Y.append(list_label_test[i])
            for i in range(len(result)):
                for j in range(10,19):
                    if result[i] == j:
                        result[i] = j-10
            return Y, m
        Y, result = resultandY(result,list_label_test)
        M = confusion_matrix(Y,result)
        print(M)
        Y, result1 = resultandY(result1,list_label_test)
        M1 = confusion_matrix(Y,result1)
        print(M1)
```

```
[[218 230 262 162  67  30   7   4   0   0   0]
 [ 59 222 415 288 103  36   9   2   1   0   0]
 [114 165 227 224 168  79  42  10   3   0   0]
 [ 50 125 198 204 196 125  64  29   9   7   3]
 [  1  15  64 120 212 236 182  97  42  11   2]
 [ 20  49 114 218 222 149  82  28   7   3   0]
 [ 36  63 140 212 238 185  62  16   5   1   0]
 [  3   8  37  95 156 243 226 158  67  25  10]
 [  4  13  42  94 166 171 208 141  92  24  19]
 [  1   7  32  90 161 244 231 144  76  21   2]
 [  0   0   0   0   0   0   0   0   0   0   0]]
[[330 259 189 104  60  25   6   5   2   0   0   0   0   0]
 [137 341 365 196  67  19   7   3   0   0   0   0   0   0]
 [127 190 246 231 138  66  25   6   1   1   1   0   0   0]
 [ 35 117 265 279 152  91  41  17   6   4   3   0   0   0]
 [  2  12  38 108 186 302 202  83  40   6   3   0   0   0]
 [ 19  62 125 208 209 158  71  26  13   0   1   0   0   0]
 [ 17  48  83 150 189 192 156  90  30   3   0   0   0   0]
 [  2  16  49 111 168 209 200 155  75  29  14   0   0   0]
 [  1  13  38  73 169 172 211 152  84  40  15   4   1   1]
 [  1   5  22  41  67 185 255 219 130  66  12   6   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

10 Present the F1 score

```
In [7]: for i in range(10):
            n = 0
```

```
            k = 0
            FN = 0
            while n<9:
                FN = FN+M[i][n]
                n+=1
            FN = FN - M[i][i]
            TP = M[0][0]
            FP = 0
            while k<9:
                FP = FP+M[k][i]
                k+=1
            FP = FP - M[k][i]
            P = TP/ (TP + FP)
            R =  TP/(TP + FN)
            F1 = (2 * P * R) / (P + R)

        print(F1)
```

```
0.3004824259131634
```