# Assignment 11

June 12, 2019

20163228 Yuseon Nam

[1]:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import confusion_matrix
import pandas as pd
import random
```

## 1 Read Files

[2]:

```
#
# Read Train File
#
size_row    = 28    # height of the image
size_col    = 28    # width of the image

train_file  = "mnist_train.csv"

handle_file = open(train_file, "r")
train_data  = handle_file.readlines()
handle_file.close()

train_num   = len(train_data)

train_list  = np.zeros((train_num, size_row * size_col), dtype=float)
train_a     = np.zeros((train_num, size_row * size_col + 1), dtype=float)
train_label = np.zeros((10, train_num), dtype=int)
train_original_label = np.zeros(train_num, dtype=int)

count = 0
label = 2
for line in train_data :
    line_data = line.split(',')
    train_original_label[count] = line_data[0]

    for i in range (10) :
        if (line_data[0] == str(i)) :
            train_label[i, count] = 1
        else :
            train_label[i, count] = -1

    # Image Vector
    im_vector = np.asfarray(line_data[1:])
    train_list[count, :] = im_vector

    train_a[count, 0]    = 1
    train_a[count, 1:]   = im_vector
    count += 1
```

[3]:

```
#
# Read Test File
#
test_file   = "mnist_test.csv"

handle_file = open(test_file, "r")
test_data   = handle_file.readlines()
handle_file.close()

test_num    = len(test_data)

test_list   = np.zeros((test_num, size_row * size_col), dtype=float)
test_a      = np.zeros((test_num, size_row * size_col + 1), dtype=float)
test_label = np.zeros((10, test_num), dtype=int)
test_original_label = np.zeros(test_num, dtype=int)

count = 0
label = -2
for line in test_data :
    line_data = line.split(',')
    test_original_label[count] = line_data[0]

    for i in range (10) :
        if (line_data[0] == str(i)) :
            test_label[i, count] = 1
        else :
            test_label[i, count] = -1

    im_vector = np.asfarray(line_data[1:])
    test_list[count, :] = im_vector

    test_a[count, 0]    = 1
    test_a[count, 1:]   = im_vector
    count += 1
```

## 2 Make Random Vector

[4]:

```
k = 1000
random_vector = np.zeros((k, size_row * size_col), dtype=float)

for i in range (k) :
    for j in range(size_row * size_col) :
        random_vector[i, j] = random.gauss(0, 1)
```

[5]:

```
def calculate_new_a(original_img) :
    num = len(original_img)
    new_a = np.zeros((num, k+1), dtype=float)

    new_a[:, 0] = 1
    new_a[:, 1:] = np.dot(original_img, random_vector.T)

    for i in range(num) :
        for j in range (1, k+1) :
            new_a[i, j] = max(new_a[i, j], 0)

    return new_a
```

[6]:

```
new_train_a = np.zeros((train_num, k+1), dtype=float)
new_train_a = calculate_new_a(train_list)
```

[7]:

```
new_test_a = np.zeros((test_num, k+1), dtype=float)
new_test_a = calculate_new_a(test_list)
```

## 3 Solve Least Square Problem

[8]:

```
def cal_least_square(image, label) :
    n = len(label)

    trans_image = image.T
    ata = np.dot(trans_image, image)

    re_label = label.reshape(n, 1)
    atb = np.dot(trans_image, re_label)

    mati_ata = np.linalg.pinv(ata)
    mat_atb  = np.asmatrix(atb)
    aia = np.dot(mati_ata, mat_atb)

    return aia
```

[9]:

```
theta = np.zeros((10, k + 1), dtype=float)
for i in range(10) :
    setaa    = cal_least_square(new_train_a, train_label[i, :])
    theta[i] = setaa.reshape(k+1)
```

# 4 Calculate y and label according to argmax

[10]:

```
train_y = np.zeros((train_num, 10), dtype=float)
test_y  = np.zeros((test_num, 10) , dtype=float)

for i in range (10) :
    train_y[:, i] = np.dot(new_train_a, theta[i])
    test_y[: , i] = np.dot(new_test_a , theta[i])
```

[11]:

```
train_pred = np.zeros(train_num, dtype=int)
test_pred  = np.zeros(test_num,  dtype=int)

for i in range (train_num) :
    train_pred[i] = train_y[i, :].argmax()

for i in range (test_num) :
    test_pred[i] = test_y[i, :].argmax()
```

# 5 Calculate Accuracy

## 5.1 Train Data

[12]:

```
train_result = confusion_matrix(train_original_label, train_pred)
print(train_result)
```

```
[[5789    1   10    6    9   14   39    6   42    7]
 [   1 6627   38   13    9   10   12   10   12   10]
 [  35   39 5577   50   30   11   36   63   98   19]
 [  17   28   93 5659    2  110   17   49   85   71]
 [   7   37   19    3 5530    9   44   15   25  153]
 [  27   22   13  116   24 5034   86   12   47   40]
 [  38   15    7    1   11   68 5754    2   21    1]
 [  16   70   43   21   53    7    1 5900   10  144]
 [  17   67   40  108   26  103   45   15 5349   81]
 [  25   19   18   83  147   32    2  142   47 5434]]
```

[13]:

```
# Calculate true positive rate & error rate
train_tp  = 0
train_err = 0
for i in range (10) :
    train_tp += train_result[i][i]

train_tp /= train_num
train_err = 1 - train_tp
```

[14]:

```
print("Train Data")
print("True Positive Rate : ", train_tp)
print("Error Rate         : ", train_err)
```

```
Train Data
True Positive Rate :  0.9442166666666667
Error Rate         :  0.05578333333333296
```

## 5.2 Test Data

[15]:

```
test_result  = confusion_matrix(test_original_label , test_pred)
print(test_result)
```

```
[[ 962    0    2    1    0    1    9    2    3    0]
 [   0 1123    2    2    0    2    4    0    2    0]
 [   7    2  962   15    5    1    6   10   20    4]
 [   1    0   11  952    0   18    1    9   12    6]
 [   1    7    4    1  923    1   13    3    6   23]
 [   5    1    0   19    4  831   14    4   10    4]
 [  11    3    2    0    3   10  926    1    2    0]
 [   0   19   13    4    9    1    1  949    1   31]
 [   5    4    6   11    7   15   12    7  900    7]
 [   6    7    2   15   27    5    3   16    7  921]]
```

[16]:

```
# Calculate true positive rate & error rate
test_tp  = 0
test_err = 0
for i in range (10) :
    test_tp += test_result[i][i]

test_tp /= test_num
test_err = 1 - test_tp
```

[17]:

```
print("Test Data")
print("True Positive Rate : ", test_tp)
print("Error Rate         : ", test_err)
```

```
Test Data
True Positive Rate :  0.9449
Error Rate         :  0.05510000000000004
```