Assignment 11

과제 정의

Build a binary classifier based on k random features for each digit against all the other digits at MNIST dataset.

Let x = (x_1, x_2, ... , x_m) be a vector representing an image in the dataset.

The prediction function f_d(x; w) is defined by the linear combination of input vector x and the model parameter w for each digit d :

f_d(x; w) = w_0 *1 + w_1* g_1 + w_2 *g_2 + ... + w_k* g_k

where w = (w_0, w_1, ... , w_k) and the basis function g_k is defined by the inner product of random vector r_k and input vector x.

You may want to try to use g_k = max(inner production(r_k, x), 0) to see if it improves the performance.

The prediction function f_d(x; w) should have the following values:

f_d(x; w) = +1 if label(x) = d f_d(x; w) = -1 if label(x) is not d

The optimal model parameter w is obtained by minimizing the following objective function for each digit d : \sum_i (f_d(x^(i); w) - y^(i))^2

and the label of input x is given by:

argmax_d f_d(x; w)

1. Compute an optimal model parameter using the training dataset for each classifier f_d(x, w)
2. Compute (1) true positive rate, (2) error rate using (1) training dataset and (2) testing dataset.

모듈 정의

In [1]:

```
import numpy as np
import collections
import matplotlib.pyplot as plt
```

1. Compute an optimal model parameter using the training dataset for each classifier f_d(x, w)

28 * 28크기의 랜덤 벡터 1000개를 -100 이상 100이하의 값을 가지고 생성한다

In [55]:

```
r_v = []
for i in range(0,1000):
    r_v.append(np.random.uniform(-100,100,784))
```

In [56]:

```
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


def normalize(data):
    data_normalized = (data - min(data)) / (max(data) - min(data))
    return(data_normalized)


# list_label  = np.empty(num_image, dtype=int)
list_label = []
int_data = []
int_data_y = {0:[],1:[],2:[],3:[],4:[],5:[],6:[],7:[],8:[],9:[]}
X = dict()
for line in data:
    im_vector2 = []
    line_data   = line.split(',')
    label       = line_data[0]
    im_vector   = np.asfarray(line_data[1:])
    for i in range(0,1000):
        im_vector2.append(np.dot(im_vector,r_v[i]))
    im_vector = np.asfarray(im_vector2)
    im_vector   = normalize(im_vector)
    im_vector = np.insert(im_vector, 0, 1)
    int_data.append(im_vector)
    list_label.append(int(label))
    for x in range(0,10):
        if x == int(label):
            int_data_y[x].append(1.0)
        else:
            int_data_y[x].append(-1.0);
    count += 1

xn = np.array(int_data,dtype=float)
for i in range(0,10):
    X[i] = np.dot(np.linalg.pinv(xn) , np.array(int_data_y[i],dtype=float))
print(X)
```

```
{0: array([-0.72254829,  3.19454759, -4.0624533 , ..., -3.54886959,
       -4.44109884, -2.52343013]), 1: array([-0.18705929, -4.1844391
9,  1.64693456, ..., -1.3095842 ,
        1.51053281, -1.75180674]), 2: array([-0.95525303, -1.3084089
6,  0.83358349, ...,  1.3453345 ,
        0.22275157,  3.47499215]), 3: array([-1.07666679,  7.2544938
4, -0.68155142, ...,  1.27501709,
       -1.18583415,  2.35142979]), 4: array([-0.52243036, -4.4455741
7,  2.69156528, ...,  1.21823688,
        2.24918714, -2.37223084]), 5: array([-0.44605947,  3.3592525
7, -6.0436259 , ..., -1.46100522,
       -2.05726897,  2.78326134]), 6: array([-0.88490181, -2.3188609
5,  0.32071445, ...,  1.04551108,
        0.92000189, -1.14728974]), 7: array([-0.56690924,  0.7205275
4, -0.28483894, ...,  0.50464734,
        0.47622473, -0.3148575 ]), 8: array([-1.79370427,  1.4029289
,  0.09256054, ...,  2.07420214,
       -1.17398195,  1.55996428]), 9: array([-0.84446744, -3.6744671
8,  5.48711125, ..., -1.14349003,
        3.47948576, -2.06003261])}
```

2. Compute true positive rate, error rate using training dataset

- 예측한 label (argmax)

In [57]:

```
count = 0
tp = 0
error = 0
for x in range(len(xn)):
    f1 = plt.figure(1)
    argmax = []
    for i in range(0,10):
        argmax.append(np.dot(xn[x],X[i]))
    label       = argmax.index(max(argmax))
    if list_label[x] == label:
        tp = tp + 1
    else:
        error = error + 1
```

In [58]:

```
print("true positive rate :" + str(tp/num_image))
print("error rate : " + str(error/num_image))
```

```
true positive rate :0.8632666666666666
error rate : 0.13673333333333335
```

28 * 28크기의 랜덤 벡터 1000개를 0 이상 100이하의 값을 가지고 생성한다.

In [59]:

```
r_v = []
for i in range(0,1000):
    r_v.append(np.random.uniform(0,100,784))
```

In [60]:

```
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


def normalize(data):
    data_normalized = (data - min(data)) / (max(data) - min(data))
    return(data_normalized)


# list_label  = np.empty(num_image, dtype=int)
list_label = []
int_data = []
int_data_y = {0:[],1:[],2:[],3:[],4:[],5:[],6:[],7:[],8:[],9:[]}
X = dict()
for line in data:
    im_vector2 = []
    line_data   = line.split(',')
    label       = line_data[0]
    im_vector   = np.asfarray(line_data[1:])
    for i in range(0,1000):
        im_vector2.append(np.dot(im_vector,r_v[i]))
    im_vector = np.asfarray(im_vector2)
    im_vector   = normalize(im_vector)
    im_vector = np.insert(im_vector, 0, 1)
    int_data.append(im_vector)
    list_label.append(int(label))
    for x in range(0,10):
        if x == int(label):
            int_data_y[x].append(1.0)
        else:
            int_data_y[x].append(-1.0);
    count += 1

xn = np.array(int_data,dtype=float)
for i in range(0,10):
    X[i] = np.dot(np.linalg.pinv(xn) , np.array(int_data_y[i],dtype=float))
print(X)
```

```
{0: array([-0.4668312 , -1.69602874, -1.98363809, ...,  2.07035167,
        3.01145283,  5.99737307]), 1: array([-0.14704801, -3.3949314
1, -5.98400725, ..., -3.47042636,
       -1.20146487,  1.52225734]), 2: array([-0.45729751,  2.4524691
7, -3.39744655, ..., -0.02372528,
       -0.64321627, -2.65970507]), 3: array([-0.94767904,  3.0055666
4, 10.52287724, ...,  1.05004842,
       -3.45554081, -3.96145203]), 4: array([-0.31525493,  1.6491439
4, -4.85749609, ..., -1.81795399,
       -0.61375117,  0.68506044]), 5: array([-0.63391791,  4.7088706
6, -2.17934349, ..., -0.95957429,
       -1.15669778, -0.53074787]), 6: array([-1.25512835,  1.5101800
7,  5.05874818, ...,  1.23629766,
        0.16299412,  1.19577497]), 7: array([-0.63156056, -0.3553204
,  5.71788694, ..., -0.96553524,
        2.51483294, -0.02089986]), 8: array([-2.23806658,  1.1990105
5,  3.36951911, ...,  1.83342694,
        1.01419043, -3.15203179]), 9: array([-0.90721593, -9.0789604
6, -6.26709999, ...,  1.04709047,
        0.36720058,  0.92437079])}
```

In [61]:

```
count = 0
tp = 0
error = 0
for x in range(len(xn)):
    f1 = plt.figure(1)
    argmax = []
    for i in range(0,10):
        argmax.append(np.dot(xn[x],X[i]))
    label        = argmax.index(max(argmax))
    if list_label[x] == label:
        tp = tp + 1
    else:
        error = error + 1
```

In [62]:

```
print("true positive rate :" + str(tp/num_image))
print("error rate : " + str(error/num_image))
```

```
true positive rate :0.86505
error rate : 0.13495
```

2. Compute true positive rate, error rate using testing dataset

- 예측한 label (argmax)

In [63]:

```
file_data   = "mnist_test.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


def normalize(data):
    data_normalized = (data - min(data)) / (max(data) - min(data))
    return(data_normalized)


# list_label  = np.empty(num_image, dtype=int)
list_label = []
int_data = []
for line in data:
    im_vector2 = []
    line_data   = line.split(',')
    label       = line_data[0]
    im_vector   = np.asfarray(line_data[1:])
    for i in range(0,1000):
        im_vector2.append(np.inner(im_vector,r_v[i]))
    im_vector = np.asfarray(im_vector2)
    im_vector   = normalize(im_vector)
    im_vector = np.insert(im_vector, 0, 1)
    int_data.append(im_vector)
    list_label.append(int(label))

xn = np.array(int_data,dtype=float)
```

In [64]:

```
count = 0
tp = 0
error = 0
for x in range(len(xn)):
    f1 = plt.figure(1)
    argmax = []
    for i in range(0,10):
        argmax.append(np.dot(xn[x],X[i]))
    label       = argmax.index(max(argmax))
    if list_label[x] == label:
        tp = tp + 1
    else:
        error = error + 1
```

In [65]:

```
print("true positive rate :" + str(tp/num_image))
print("error rate : " + str(error/num_image))
```

```
true positive rate :0.8658
error rate : 0.1342
```

g_k = max(inner production(r_k, x), 0) 의 성능향상도 확인할 수 있었다.