20145271 Geum Kang Hyeon

Zero Classifying Using MNIST Dateset

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

The prediction function $f_w(x)$ is defined by the linear combination of data $(1, x)$ and the model parameter w: $f_w(x) = w_0 * 1 + w_1 * x_1 + w_2 * x_2 + \ldots + w_m * x_m$ where $w = (w_0, w_1, \ldots, w_m)$

The prediction function f_w(x) should have the following values: $f_w(x)$ = +1 if label(x) is 0, $f_w(x)$ = -1 if label(x) is not 0

The optimal model parameter w is obtained by minimizing the following objective function: $\sum_i (f_w(x^i - y^i)^2)$

1. Declare required variables

References to Assignment 03 code

In [8]:

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

train_file_data = "mnist_train.csv"
test_file_data = "mnist_test.csv"

handle_file = open(train_file_data, "r")
data = handle_file.readlines()
handle_file.close()

handle_file = open(test_file_data, "r")
data_Test = handle_file.readlines()
handle_file.close()

size_row = 28
size_col = 28

num_image = len(data)
num_image_Test = len(data_Test)

count = 0
count_Test = 0

list_image  = np.empty((num_image, size_row * size_col), dtype=float)
list_label  = np.empty(num_image, dtype=int)

list_image_Test  = np.empty((num_image_Test, size_row * size_col), dtype=float)
list_label_Test  = np.empty(num_image_Test, dtype=int)
```

2. Read Data

In [9]:

```
# Read Train Data
for line in data:
    line_data   = line.split(',')
    label       = line_data[0]
    im_vector   = np.asfarray(line_data[1:])

    list_label[count] = label
    list_image[count] = im_vector
    count += 1

# Read Test Data
for line in data_Test:
    line_data   = line.split(',')
    label       = line_data[0]
    im_vector   = np.asfarray(line_data[1:])

    list_label_Test[count_Test] = label
    list_image_Test[count_Test] = im_vector
    count_Test += 1
```

3. Align Data A anb b in formula $Ax = b$

In [10]:

```
A = np.zeros((count, size_row*size_col), dtype=float)
b = np.zeros(count, dtype=float)
b_Test = np.zeros(count, dtype=float)

A = list_image
for i in range(count):
    if list_label[i] == 0:
        b[i] = 1
    else:
        b[i] = -1

for i in range(count_Test):
    if list_label_Test[i] == 0:
        b_Test[i] = 1
    else:
        b_Test[i] = -1
```

4. Calculate x

$$x = (A^T A)^{-1} A^T b$$

$$x = A^+ b$$

In [11]:

```
transpose_A = np.transpose(A)

# Calculate Pseudo Inverse
step1 = np.matmul(transpose_A, A)
step2 = np.linalg.pinv(step1)
step3 = np.matmul(step2, transpose_A)

# x = (Pseudo Inverse of A) * b
result = np.matmul(step3, b)
```

5. Compute TP, FP, TN, FN using Train Dataset

TP, TN are Answer, FP, FN are Wrong Answer

Put the actual data in the expression Ax = b and compare it with the answer in the train dataset

In [12]:

```
TP, TN, FP, FN = (0, 0, 0, 0)

for i in range(count):
    value = np.matmul(list_image[i], result)
    if value >= 0 and b[i] == 1:
        TP += 1
    elif value < 0 and b[i] == -1:
        TN += 1
    elif value >= 0 and b[i] == -1:
        FP += 1
    elif value < 0 and b[i] == 1:
        FN += 1
```

6. Compute TP, FP, TN, FN using Test Dataset

x uses x obtained through the train dataset

Model from train dataset can be verified

In [13]:

```
TP_Test, TN_Test, FP_Test, FN_Test = (0, 0, 0, 0)

for i in range(count_Test):
    value = np.matmul(list_image_Test[i], result)
    if value >= 0 and b_Test[i] == 1:
        TP_Test += 1
    elif value < 0 and b_Test[i] == -1:
        TN_Test += 1
    elif value >= 0 and b_Test[i] == -1:
        FP_Test += 1
    elif value < 0 and b_Test[i] == 1:
        FN_Test += 1
```

7. Result

In [14]:

```
print('[Train Data Set]')
print('Total Data Count : ' + str(count) + '₩n')
print('TRUE POSITIVE : ' + str(TP) + ' (' + str("%0.1f" % (TP / count * 100)) + '%)')
print('TRUE NEGATIVE : ' + str(TN) + ' (' + str("%0.1f" % (TN / count * 100)) + '%)')
print('FALSE POSITIVE : ' + str(FP) + ' (' + str("%0.1f" % (FP / count * 100)) + '%)')
print('FALSE NEGATIVE : ' + str(FN) + ' (' + str("%0.1f" % (FN / count * 100)) + '%)')

print('₩n₩n[Test Data Set]')
print('Total Data Count : ' + str(count_Test) + '₩n')
print('TRUE POSITIVE : ' + str(TP_Test) + ' (' + str("%0.1f" % (TP_Test / count_Test * 100)) +
'%)')
print('TRUE NEGATIVE : ' + str(TN_Test) + ' (' + str("%0.1f" % (TN_Test / count_Test * 100)) +
'%)')
print('FALSE POSITIVE : ' + str(FP_Test) + ' (' + str("%0.1f" % (FP_Test / count_Test * 100)) +
'%)')
print('FALSE NEGATIVE : ' + str(FN_Test) + ' (' + str("%0.1f" % (FN_Test / count_Test * 100)) +
'%)')
```

```
[Train Data Set]
Total Data Count : 60000

TRUE POSITIVE : 5347 (8.9%)
TRUE NEGATIVE : 53759 (89.6%)
FALSE POSITIVE : 318 (0.5%)
FALSE NEGATIVE : 576 (1.0%)


[Test Data Set]
Total Data Count : 10000

TRUE POSITIVE : 917 (9.2%)
TRUE NEGATIVE : 8959 (89.6%)
FALSE POSITIVE : 61 (0.6%)
FALSE NEGATIVE : 63 (0.6%)
```

Train Data	True	False
Positive	5347 (8.9%)	*318 (0.5%)*
Negative	53759 (89.6%)	*576 (1.0%)*

Test Data	True	False
Positive	917 (9,2%)	*61 (0.6%)*
Negative	8959 (89.6%)	*63 (0.6%)*