Assignment 09

May 29, 2019

20163228 Yuseon Nam

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

1 Read Files

In [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_label = np.zeros(train_num, dtype=int)
train_a     = np.zeros((train_num, size_row * size_col + 1), dtype=float)

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

    if (line_data[0] == '0') :
        label = 1
    else :
        label = -1
```

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

            train_label[count]   = label
            train_list[count, :] = im_vector

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

In [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_label = np.zeros(test_num, dtype=int)
        test_a     = np.zeros((test_num, size_row * size_col + 1), dtype=float)

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

            if (line_data[0] == '0') :
                label = 1
            else :
                label = -1

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

            test_label[count]   = label
            test_list[count, :] = im_vector

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

In [4]:
```
        real_train_data = np.zeros((train_num, size_row * size_col + 1), dtype=float)
        real_test_data  = np.zeros((test_num, size_row * size_col + 1), dtype=float)

        def divide_train_test() :
            train_90 = train_num * 90
```

```
test_60  = test_num * 60

real_train_data[0:train_90, :] = copy.deepcopy(train_data[0:train_90, :])
real_train_data[train_90: , :] = copy.deepcopy(test_data[0:test_60, :])

real_test_data[0:test_60, :] = copy.deepcopy(train_data[train_90:, :])
real_test_data[test_60: , :] = copy.deepcopy(test_data[test_60:, :])
```

2 Calculate Pseudo Inverse

In [5]:
```
class LeastSquare() :

    def __init__(self, image, label) :
        self.image = image
        self.label = label
        self.n     = len(label)
        self.m     = len(image[0])


    def calculate_ata(self) :
        self.trans_image = self.image.T
        self.ata = np.dot(self.trans_image, self.image)


    def calculate_atb(self) :
        re_label = self.label.reshape(self.n, 1)
        self.atb = np.dot(self.trans_image, re_label)


    def calculate_theta(self) :
        mati_ata = np.linalg.pinv(self.ata)
        mat_atb = np.asmatrix(self.atb)
        self.aia = np.dot(mati_ata, mat_atb)


    def train(self) :
        self.calculate_ata()
        self.calculate_atb()

        self.calculate_theta()

        return self.aia
```

In [6]:
```
ls = LeastSquare(train_a, train_label)
theta = ls.train()
```

In [7]:
```
def calculate_y(matrix_a, theta) :
```

```
    result   = np.dot(matrix_a, theta)
    y_result = np.zeros(len(matrix_a), dtype=int)

    for i in range (len(matrix_a)) :
        if (result[i][0] > 0) :
            y_result[i] = 1
        else :
            y_result[i] = -1

    return y_result
```

```
In [8]: train_pred = calculate_y(train_a, theta)
        test_pred = calculate_y(test_a, theta)

        print(train_pred)
        print(test_pred)
```

```
[-1  1 -1 ... -1 -1 -1]
[-1 -1 -1 ... -1 -1 -1]
```

3 Calculate Accuracy

```
In [9]: train_result = confusion_matrix(train_label, train_pred)
        test_result  = confusion_matrix(test_label, test_pred)
```

```
In [10]: def calculate_accuracy(matrix) :
             accuracy = np.zeros((2, 2), dtype=float)

             accuracy[0][0] = matrix[1][1] / (matrix[1][0] + matrix[1][1])
             accuracy[0][1] = matrix[1][0] / (matrix[1][0] + matrix[1][1])
             accuracy[1][0] = matrix[0][1] / (matrix[0][0] + matrix[0][1])
             accuracy[1][1] = matrix[0][0] / (matrix[0][0] + matrix[0][1])

             show_table(accuracy)
```

```
In [11]: def calculate_accuracy_manually(true, pred) :
             count = np.zeros((2, 2), dtype=float)
             accuracy = np.zeros((2, 2), dtype=float)

             for i in range (len(true)) :
                 if ((true[i] == 1) and (pred[i] == 1)) :      # TP
                     count[0][0] += 1
                 elif ((true[i] == 1) and (pred[i] == -1)) :   # FN
                     count[0][1] += 1
                 elif ((true[i] == -1) and (pred[i] == 1)) :   # FP
```

```
                count[1][0] += 1
            elif ((true[i] == -1) and (pred[i] == -1)) :  # TN
                count[1][1] += 1

        accuracy[0][0] = count[0][0] / (count[0][0] + count[0][1])
        accuracy[0][1] = count[0][1] / (count[0][0] + count[0][1])
        accuracy[1][0] = count[1][0] / (count[1][0] + count[1][1])
        accuracy[1][1] = count[1][1] / (count[1][0] + count[1][1])

        show_table(accuracy)

In [12]: def show_table(accuracy) :
        data = {"True" : [accuracy[0][0], accuracy[1][1]], "False" : [accuracy[0][1], accu
        df = pd.DataFrame(data, columns=["True", "False"], index=["Positive", "Negative"])
        print(df)
```

3.1 Accuracy for Train Data

```
In [13]: calculate_accuracy(train_result)

              True     False
Positive  0.872531  0.127469
Negative  0.996690  0.003310


In [14]: calculate_accuracy_manually(train_label, train_pred)

              True     False
Positive  0.872531  0.127469
Negative  0.996690  0.003310
```

3.2 Accuracy for Test Data

```
In [15]: calculate_accuracy(test_result)

              True     False
Positive  0.883673  0.116327
Negative  0.995233  0.004767


In [16]: calculate_accuracy_manually(test_label, test_pred)

              True     False
Positive  0.883673  0.116327
Negative  0.995233  0.004767
```