Assignment11

June 13, 2019

1 Information

Writer : *Junhyuck Woo*
Std.ID : 20145337
Project : *Build a binary classifier to classify digit* 0 *against all the other digits at MNIST dataset.*

2 import library

```
In [1]: import numpy as np
        import pandas as pd
```

3 Load files

Training data, Testing data

```
In [2]: file_data   = "mnist_train.csv"
        handle_file = open(file_data, "r")
        train_data  = handle_file.readlines()
        handle_file.close()

        file_data   = "mnist_test.csv"
        handle_file = open(file_data, "r")
        test_data   = handle_file.readlines()
        handle_file.close()


        num_train = len(train_data)
        num_test = len(test_data)
        train_image  = np.zeros((28 * 28, num_train), dtype=float)
        train_label = np.zeros(num_train, dtype=int)
        test_image  = np.zeros((28 * 28, num_test), dtype=float)
        test_label = np.zeros(num_test, dtype=int)

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

```
                label = line_data[0]
                train_label[count] = label
                im_vector = np.asfarray(line_data[1:])
                train_image[:,count] = im_vector
                count += 1
        count = 0
        for line in test_data:
            line_data = line.split(',')
            label = line_data[0]
            test_label[count] = label
            im_vector = np.asfarray(line_data[1:])
            test_image[:,count] = im_vector
            count += 1
```

```
In [3]: class classifier:
            def __init__(self, k=64):
                self.k = k
                self.filter = self.generate_filter(k)

            def normalize(self, data):
                data_normalized = (data - np.min(data)) / (np.max(data) - np.min(data))
                return(data_normalized)

            def calculate_average(self, data):
                size_x, size_y = data.shape
                size = size_x*size_y
                average = sum(sum(data))/size
                return(average)

            def calculate_standard(self, average, data):
                variance = self.calculate_average(data*data) - (average*average)
                std = np.sqrt(variance)
                return(std)

            def whitening(self, data):
                avgerage = self.calculate_average(data)
                standard = self.calculate_standard(avgerage, data)
                whiten_data = (data - avgerage) / standard
                return(whiten_data)

            def generate_filter(self, k=64):
                data_filter = np.random.normal(0, 1, (k, 28*28))
                return(np.asmatrix(data_filter))

            def extract_feature(self, data, f):
                feature = f * data
                size_x, size_y = feature.shape
                for i in range(size_x):
```

```
            for j in range(size_y):
                feature[i,j] = max(0, feature[i,j])
        return(feature)

    def reshape_data(self, data):
        num = max(data.shape)
        reshape_data = np.zeros((self.k + 1, num), dtype=float)
        for i in range(num):
            reshape_data[:,i] = np.insert(data[:,i], 0, 1)
        return(reshape_data)

    def reshape_label(self, label, select):
        num = len(label)
        reshape_label = np.zeros(num, dtype=int)
        for i in range(num):
            if(int(label[i])==select):
                reshape_label[i] = 1
            else:
                reshape_label[i] = -1
        return(reshape_label)

    def train(self, train_data, label, num_digit=10):
        count = 0
        normalized_data = self.normalize(train_data)
        whiten_data = self.whitening(normalized_data)
        self.x = np.zeros((num_digit, self.k+1), dtype=float)
        feature = self.extract_feature(whiten_data, self.filter)
        train_image = self.reshape_data(feature)
        A = np.asmatrix(train_image.transpose())
        pinv_A = np.linalg.pinv(A)

        for i in range(num_digit):
            train_label = self.reshape_label(label, i)
            y = np.asmatrix(train_label)
            buf = np.array(pinv_A * y.transpose())
            self.x[i,:]= buf.T
        self.x = np.asmatrix(self.x)
        #return(self.x)

    def predict(self, test_data):
        normalized_data = self.normalize(test_data)
        whiten_data = self.whitening(normalized_data)
        feature = self.extract_feature(whiten_data, self.filter)
        test_image = self.reshape_data(feature)
        A = np.asmatrix(test_image.transpose())
        y = A * self.x.T
        label = []
        for i in range(max(test_data.shape)):
            label.append(np.argmax(y[i,:]))
        label = np.array(label)
        return(label)

    def evaulation(self, prediction, label):
        tp = 0
        error = 0
        result =np.zeros((11,11), dtype=int)
        for i in range(len(prediction)):
            result[prediction[i]][label[i]] +=1
            if(prediction[i] == label[i]):
                tp += 1
            else:
                error += 1

        for i in range(10):
            result[10][i] = sum(result.T[:][i])
            result[i][10] = sum(result[:][i])
        result[10][10] = sum(result[:,10])

        # Plot
        print("True Possitive: ", tp/result[10][10])
        print("Error Rate: ", error/result[10][10])
        chart = pd.DataFrame(result.T)
        return(chart)
```

```
In [4]: binary_classifier = classifier(64)
        binary_classifier.train(train_image, train_label)
```

```
In [5]: y = binary_classifier.predict(train_image)
        binary_classifier.evaulation(y, train_label)
```

```
True Possitive:  0.77035
Error Rate:  0.22965
```

```
Out[5]:       0     1     2     3     4     5     6     7     8     9     10
        0   5355    22    91    70    14    99   144    65    38    25   5923
        1      2  6415    89    50    35    24    27    20    69    11   6742
        2    131   335  4279   169   163    32   417   204   160    68   5958
        3    133   219   209  4674    48   285    87   131   171   174   6131
        4     43   204    68    68  4470    79   173   141    89   507   5842
        5    373   337    70   515   163  3218   216   115   224   190   5421
        6    141   312   201    60   113    88  4917    15    51    20   5918
        7     71   222    98    52   190    26    15  5332    19   240   6265
        8    133   338   206   425   130   221   145   139  3879   235   5851
        9     84   153    62   160   806   112    73   706   111  3682   5949
        10  6466  8557  5373  6243  6132  4184  6214  6868  4811  5152  60000
```

```
In [6]: y = binary_classifier.predict(test_image)
        binary_classifier.evaulation(y, test_label)
```

```
True Possitive:  0.7766
Error Rate:  0.2234
```

```
Out[6]:        0     1    2     3     4    5     6     7    8    9     10
        0    881     2   12    17     2   18    28    15    4    1    980
        1      0  1086   11    13     2    2     3     0   14    4   1135
        2     30    83  718    27    30    7    64    34   32    7   1032
        3     29    30   24   804     5   37    11    22   25   23   1010
        4      3    30    7     6   758   17    35    23   17   86    982
        5     56    40   19    79    26  559    27    24   34   28    892
        6     39    35   33     6    26   15   787     3   11    3    958
        7      5    49   25     6    26    3     4   853    3   54   1028
        8     37    32   24    76    16   30    29    31  668   31    974
        9     14    18    6    22   150   18    17    91   21  652   1009
        10  1094  1405  879  1056  1041  706  1005  1096  829  889  10000
```

```
In [7]: binary_classifier = classifier(256)
        binary_classifier.train(train_image, train_label)
```

```
In [8]: y = binary_classifier.predict(train_image)
        binary_classifier.evaulation(y, train_label)
```

```
True Possitive:  0.88595
Error Rate:  0.11405
```

```
Out[8]:        0     1     2     3     4     5     6     7     8     9     10
        0   5645     2    15    23    16    49   101    11    50    11   5923
        1      1  6600    28    21    15    20    12    15    24     6   6742
        2     67   169  5087   124    95    31    99   118   138    30   5958
        3     49    92   182  5222    26   150    56   101   158    95   6131
        4     12    70    31     4  5205    21    85    19    21   374   5842
        5    115    74    45   287   106  4355   155    60   136    88   5421
        6     56    58    42     7    47   115  5562     6    22     3   5918
        7     36   132    66    24   128    18    11  5583    17   250   6265
        8     49   247    86   198    70   204    66    44  4750   137   5851
        9     41    47    23    95   259    42    17   220    57  5148   5949
        10  6071  7491  5605  6005  5967  5005  6164  6177  5373  6142  60000
```

```
In [9]: y = binary_classifier.predict(test_image)
        binary_classifier.evaulation(y, test_label)
```

```
True Possitive:  0.8955
Error Rate:  0.1045
```

```
Out[9]:       0     1    2     3    4    5     6     7    8     9     10
        0   945     1    6     1    0    9    10     2    5     1    980
        1     0  1120    2     3    0    2     4     0    4     0   1135
        2    16    24  879    25   12    5    19    16   35     1   1032
        3     4     9   27   892    2   21     7    18   22     8   1010
        4     0    10    6     0  876    2    22     4    3    59    982
        5    12    10    6    47   12  718    27    24   26    10    892
        6    14    10    4     0   14   15   899     1    1     0    958
        7     3    27   21     4    6    0     2   912    4    49   1028
        8     9    25   11    23   15   34    16     6  823    12    974
        9    13    11    3    14   42    2     3    21    9   891   1009
        10 1016  1247  965  1009  979  808  1009  1004  932  1031  10000
```

```
In [10]: binary_classifier = classifier(1024)
         binary_classifier.train(train_image, train_label)
```

```
In [11]: y = binary_classifier.predict(train_image)
         binary_classifier.evaulation(y, train_label)
```

```
True Possitive:  0.9479666666666666
Error Rate:  0.052033333333333334
```

```
Out[11]:      0     1     2     3     4     5     6     7     8     9     10
         0  5801     2    14     5     7     9    31     5    44     5   5923
         1     0  6636    36    13    16     5     2     9    16     9   6742
         2    31    34  5588    50    48     6    35    62    84    20   5958
         3     8    21    83  5698     8    95    16    49    99    54   6131
         4     6    31    15     3  5551     5    33     7    20   171   5842
         5    33    19    17    96    30  5024    86    13    63    40   5421
         6    32    18     9     2    15    53  5754     0    35     0   5918
         7    15    58    47    12    57     7     4  5938    14   113   6265
         8    15    66    53    78    30    94    44    15  5389    67   5851
         9    25    13    12    73   118    39     4   118    48  5499   5949
         10 5966  6898  5874  6030  5880  5337  6009  6216  5812  5978  60000
```

```
In [12]: y = binary_classifier.predict(test_image)
         binary_classifier.evaulation(y, test_label)
```

```
True Possitive:  0.9443
Error Rate:  0.0557
```

```
Out[12]:      0     1    2     3    4    5    6     7    8     9     10
         0  961     0    2     0    0    3    7     3    4     0    980
         1    0  1122    2     3    1    0    3     0    4     0   1135
         2    9     1  958    13    5    1    9    11   22     3   1032
         3    0     0   10   951    1   12    1    11   17     7   1010
         4    1     5    4     0  921    1    8     2    4    36    982
```

```
5     3     2    0    24    6  826   10     5   10     6    892
6     7     2    1     0    7    7  927     1    6     0    958
7     1    16   15     2   11    0    2   952    4    25   1028
8     5     2    6    22    3   11    9     8  901     7    974
9     7     7    1     7   23   14    2    16    8   924   1009
10  994  1157  999  1022  978  875  978  1009  980  1008  10000
```

```
In [ ]:
```