Assignment08

November 22, 2018

1 Syed Farhan Alam Zaidi

2 2018210031

3 Assignment 08

Github Link: https://github.com/farhan-93/assignment08.git ## Binary- Least Square Classification on MNIST data Classifies digit 0 among digit 1-9

Import required libraries for the work.

```
In [151]: import numpy as np
          import matplotlib.pyplot as plt
          import sklearn.metrics as metrics
```

Below function will load training and testing data from CSV files available

```
In [152]: def data_load():
              file_data_train = "mnist_train.csv"
              file_data_test  = "mnist_test.csv"

              h_data_train    = open(file_data_train, "r")
              h_data_test     = open(file_data_test, "r")

              data_train      = h_data_train.readlines()
              data_test       = h_data_test.readlines()

              h_data_train.close()
              h_data_test.close()

              size_row    = 28    # height of the image
              size_col    = 28    # width of the image

              num_train   = len(data_train)   # number of training images
              num_test    = len(data_test)    # number of testing images

              #
              # normalize the values of the input data to be [0, 1]
```

```
#
def normalize(data):

    data_normalized = (data - min(data)) / (max(data) - min(data))

    return(data_normalized)

#
# example of distance function between two vectors x and y
#
def distance(x, y):

    d = (x - y) ** 2
    s = np.sum(d)
    # r = np.sqrt(s)

    return(s)

#
# make a matrix each column of which represents an images in a vector form
#
list_image_train    = np.empty((size_row * size_col, num_train), dtype=float)
list_label_train    = np.empty(num_train, dtype=int)

list_image_test     = np.empty((size_row * size_col, num_test), dtype=float)
list_label_test     = np.empty(num_test, dtype=int)

count = 0

for line in data_train:

    line_data   = line.split(',')
    label       = line_data[0]
    im_vector   = np.asfarray(line_data[1:])
    im_vector   = normalize(im_vector)

    list_label_train[count]     = label
    list_image_train[:, count]  = im_vector

    count += 1

count = 0

for line in data_test:

    line_data   = line.split(',')
    label       = line_data[0]
    im_vector   = np.asfarray(line_data[1:])
```

```
    im_vector    = normalize(im_vector)

    list_label_test[count]      = label
    list_image_test[:, count]   = im_vector

    count += 1

#
# plot first 150 images out of 10,000 with their labels
#
f1 = plt.figure(1)

for i in range(150):

    label       = list_label_train[i]
    im_vector   = list_image_train[:, i]
    im_matrix   = im_vector.reshape((size_row, size_col))

    plt.subplot(10, 15, i+1)
    plt.title(label)
    plt.imshow(im_matrix, cmap='Greys', interpolation='None')

    frame   = plt.gca()
    frame.axes.get_xaxis().set_visible(False)
    frame.axes.get_yaxis().set_visible(False)

#plt.show()

#
# plot the average image of all the images for each digit
#
f2 = plt.figure(2)

im_average  = np.zeros((size_row * size_col, 10), dtype=float)
im_count    = np.zeros(10, dtype=int)

for i in range(num_train):

    im_average[:, list_label_train[i]] += list_image_train[:, i]
    im_count[list_label_train[i]] += 1

for i in range(10):

    im_average[:, i] /= im_count[i]

    plt.subplot(2, 5, i+1)
    plt.title(i)
    plt.imshow(im_average[:,i].reshape((size_row, size_col)), cmap='Greys', inter
```

```
            frame   = plt.gca()
            frame.axes.get_xaxis().set_visible(False)
            frame.axes.get_yaxis().set_visible(False)

        plt.show()
        return list_image_train.T, list_label_train, list_image_test.T, list_label_test
```

Below function perform least square fitting and found the Theta Values (Model paramenters) for both class -1 and class 1. Class -1 is set for Zero-digit and 1 is for Non-zero digit.

```
In [153]: def train(X_train, y_train, reg=1):

        # x_train is 60000 x 786
        # y_train is 60000 x 10
        # left is 10 x 786
        # right is 786 x 786
        ##### Converting the labels into vector
        y_train = unit_vec(y_train)
        right = np.zeros((X_train.shape[1], y_train.shape[1]))

        left = np.zeros((X_train.shape[1], X_train.shape[1]))
        for i in range(X_train.shape[0]):
            if i % 10000 == 0:
                print(i,'/',59999)

            right += np.outer(X_train[i], np.transpose(y_train[i]))
            left += np.outer(X_train[i], np.transpose(X_train[i]))
        left = left + reg*np.identity(X_train.shape[1])
        left = np.linalg.inv(left)
        print ('training complete')
        #### return the model parameters
        return np.dot(left, right)
```

Bellow function converts the the labels in categorical labels. There are two classes Class -1 (Zero) and Class 1(non-zero)

```
In [154]: def unit_vec(labels_train):
        '''Convert categorical labels 0 or non-zero to standard basis vectors in R^{2} '
        result = np.zeros((labels_train.shape[0], 2))
        for i in range(labels_train.shape[0]):
            result[i][labels_train[i]] = 1
        #### return the vector of labels.s
        return result
```

Below code predict the class of X data.

```
In [155]: def predict(model, X):
        ''' From model and data points, output prediction vectors '''
```

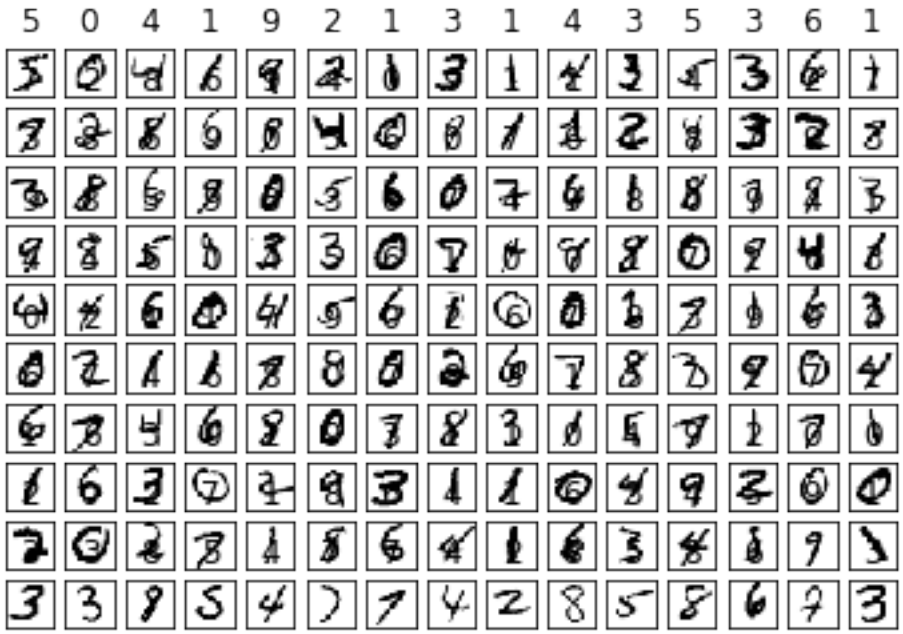
```
    results = np.zeros(X.shape[0])
    results1 = np.zeros(X.shape[0])
    for i in range(X.shape[0]):
        a = np.argmax(np.dot(np.transpose(model), X[i]))
        if a ==0:
            ##### if the index is 1 than the value -1. (For digit classified as zero
            results[i]=-1
        else:
            ##### if the index is not 1 than the value 1 (For digit classified as no
            results[i]=1
    #### Result is the predict labels for the data
    return results
```

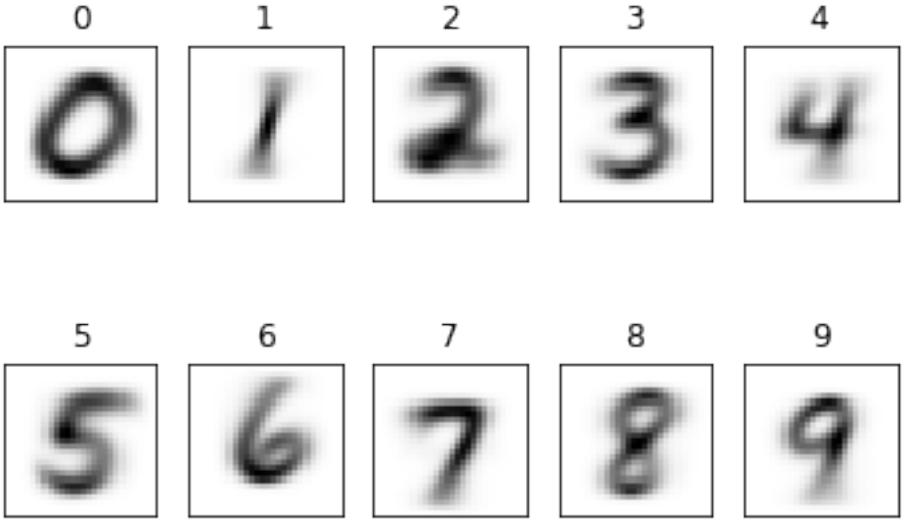
Below function show the training and testing accuracy, and confusion matrix for both raining and testing.

```
In [156]: def eval_metrics(labels_train2,red_labels_train,labels_test1,pred_labels_test):
    print("Train accuracy: {0}".format(metrics.accuracy_score(labels_train2, pred_lab
    print("Test accuracy: {0}".format(metrics.accuracy_score(labels_test1, pred_label

    print("================================================================================
    print("Confusion Matrix for Training Data")
    cm=metrics.confusion_matrix(labels_train2, pred_labels_train)
    print(cm)

    print("================================================================================
    print("Classification Report for Training Data")
    print(metrics.classification_report(labels_train2, pred_labels_train))

    print("================================================================================
    print("Confusion Matrix for Testing Data")
    print(metrics.confusion_matrix(labels_test1, pred_labels_test))

    print("================================================================================
    print("Classification Report for Testing Data")
    print(metrics.classification_report(labels_test1, pred_labels_test))

    print("================================================================================
    print("The weights for Class 1 and Class 2")
    for i in range(2):

        #label       = list_label_train[i]
        im_vector   = model[:, i]
        im_matrix   = im_vector.reshape((size_row, size_col))

        plt.subplot(2, 1, i+1)
        plt.title('Weights for class'+ str(i))
        plt.imshow(im_matrix, cmap='Greys', interpolation='None')
```

```
        frame   = plt.gca()
        frame.axes.get_xaxis().set_visible(False)
        frame.axes.get_yaxis().set_visible(False)
```

```
In [157]: if __name__ == "__main__":
            X_train, labels_trian, X_test, labels_test=data_load()
            #convert the labels of dataset
            ##For digit Zero we used 0 label and for Non Zero We used 1 label.
            labels_train1=labels_trian

            for i in range(len(labels_train1)):
                if labels_train1[i] <=0:
                    labels_train1[i]=0
                else:
                    labels_train1[i]=1

            ## Call Function that perform least square fitting on Mnist data and return the
            ## each class (zero and non-zero)
            model = train(X_train, labels_train1)

            ## Convert training labels in -1 and 1. -1 is for Zero and 1 is for non Zero
            labels_train2=list_label_train
            for i in range(len(labels_train2)):
                    if labels_train2[i] <=0:
                        labels_train2[i]=-1
                    else:
                        labels_train2[i]=1


            #Converting test labels in -1 and 1. (-1 for Zero and 1 for non zero)
            labels_test1=labels_test
            for j in range(len(labels_test1)):

                    if labels_test[j] <=0:
                        labels_test1[j]=-1
                    else:
                        labels_test1[j]=1
```



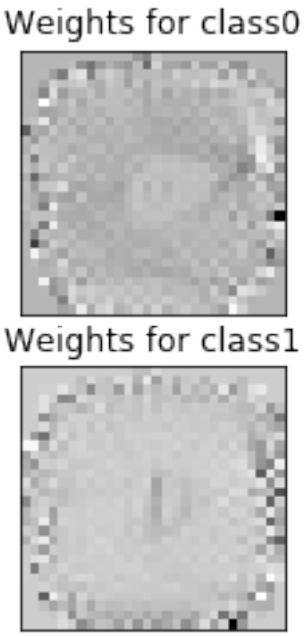



```
0 / 59999
10000 / 59999
20000 / 59999
30000 / 59999
40000 / 59999
50000 / 59999
```

```
training complete
```

In [158]:
```
#Prediction on train data
pred_labels_train = predict(model, X_train)
#Prediction on test data
pred_labels_test = predict(model, X_test)
#### Show the evaluation of data
eval_metrics(labels_train2,pred_labels_train,labels_test1,pred_labels_test)
```

```
Train accuracy: 0.9850666666666666
Test accuracy: 0.9877
================================================================================================
Confusion Matrix for Training Data
[[ 5346   577]
 [  319 53758]]
================================================================================================
Classification Report for Training Data
             precision    recall  f1-score   support

         -1       0.94      0.90      0.92      5923
          1       0.99      0.99      0.99     54077

avg / total       0.98      0.99      0.98     60000

================================================================================================
Confusion Matrix for Testing Data
[[ 917   63]
 [  60 8960]]
================================================================================================
Classification Report for Testing Data
             precision    recall  f1-score   support

         -1       0.94      0.94      0.94       980
          1       0.99      0.99      0.99      9020

avg / total       0.99      0.99      0.99     10000

================================================================================================
The weights for Class 1 and Class 2
```


Weights for class0
Weights for class1