Assignment10

December 6, 2018

1 Syed Farhan Alam Zaidi

2 2018210031

3 Assignment 10

Github Link: https://github.com/farhan-93/assignment10.git ## Multi Label- Least Square Classification on MNIST data by random vectors

Import required libraries for the work.

```
In [1]: 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 [2]: 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', interpo
```

```
            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 converts the labels into their unit vectors

```
In [3]: def unit_vec(labels_train):
        '''Convert categorical labels 0 -9  to standard basis vectors in R^{10} '''
        result = np.zeros((labels_train.shape[0], 10))
        for i in range(labels_train.shape[0]):
            result[i][labels_train[i]] = 1
        #### return the vector of labels.s
        print(result)
        return result
```

Function performs least square fitting with random generated vectors. And returns the model parameters. The dimension of model parameters are $(10, 784)$

```
In [4]: def leastSquarefit(r,x,y):

        y=unit_vec(y)
        A=np.dot(r[0],x.T)

        for i in range(1,len(r)):
            f=np.dot(r[i],x.T)
            A = np.c_[A,f]
        AG=np.linalg.inv(A.T.dot(A))
        print(AG.shape)
        AY=A.T.dot(y)
        print(AY.shape)
        theta=np.zeros((10, 784))

        theta=AG.dot(AY)
        theta = theta.T

        return theta
```

Below function performms the multi label classification by trained model parameters and returns the predicted labels

```
In [5]: def predict(model, X):
        ''' From model and data points, output prediction vectors '''
        #print(X.shape)
        results = np.zeros(X.shape[0])
        #results1 = np.zeros(X.shape[0])
```

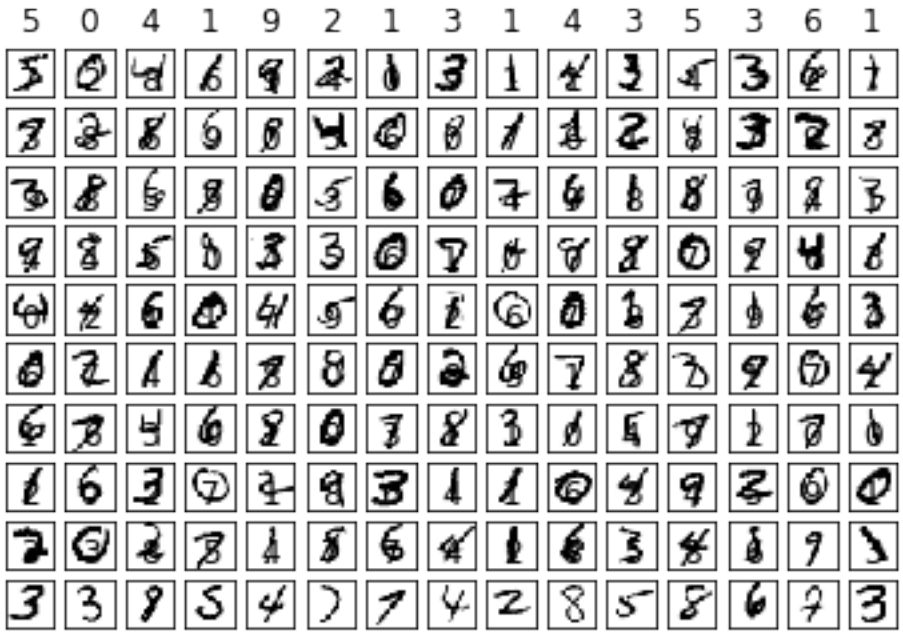
```
            for i in range(X.shape[0]):
                results[i] = np.argmax((np.dot(model, X[i])))

            return results
```

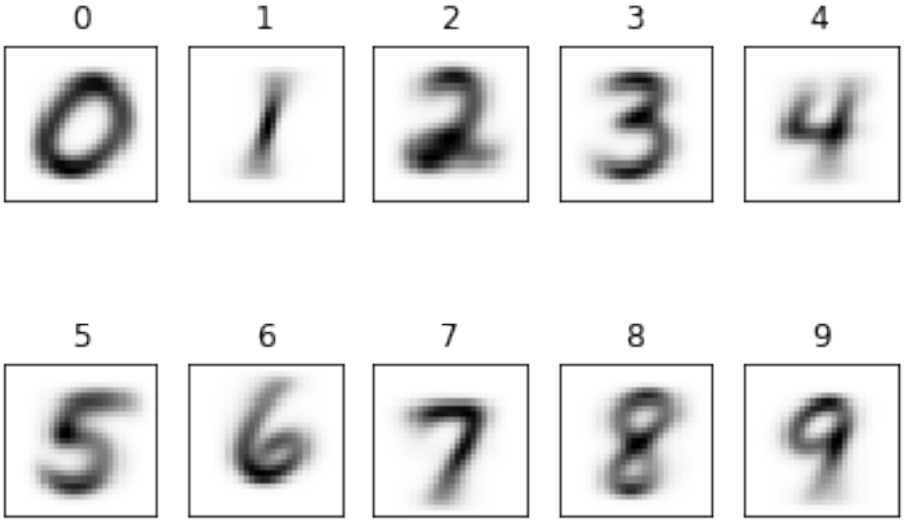
Below function predict the training and testing data with different numbers of model parameters by setting them to zero.

```
In [15]: def diff_p():
             j=4
             while j<=9:
                 tt=np.copy(t)

                 indices = np.random.choice(np.arange(784), 784-(2**j), replace=False)
                 tt[:,indices]=0

                 print("Training Set Evaluation With paramenters : ", 2**j)
                 pred_train= predict(tt, X_train)
                 print("Confusion Matrix for train Data")
                 print(metrics.confusion_matrix(labels_trian, pred_train))
                 print(metrics.accuracy_score(labels_trian, pred_train))
                 print(metrics.f1_score(labels_trian, pred_train,average='macro'))
                 pred_test= predict(tt, X_test)
                 print("Confusion Matrix for Test Data")
                 print(metrics.confusion_matrix(labels_test, pred_test))
                 print(metrics.accuracy_score(labels_test, pred_test))
                 print(metrics.f1_score(labels_test, pred_test,average='macro'))
                 j+=1
```

This is the main function and execution starts here

```
In [8]: if __name__ == "__main__":
            X_train, labels_trian, X_test, labels_test=data_load()
            #print("Hello")
            mean = 0
            std = 1

            r=np.random.normal(0,1,(784,784))
```





In [10]:
```
t=leastSquarefit(r,X_train,labels_trian)

pred_train= predict(t, X_train)
print("================================================================================
print("Confusion Matrix for train Data")
print(metrics.confusion_matrix(labels_trian, pred_train))
```

```
            print(metrics.accuracy_score(labels_trian, pred_train))
            print(metrics.f1_score(labels_trian, pred_train,average='macro'))
            pred_test= predict(t, X_test)
            print("==========================================================================
            print("Confusion Matrix for Testing Data")
            print(metrics.confusion_matrix(labels_test, pred_test))
            print(metrics.accuracy_score(labels_test, pred_test))
            print(metrics.f1_score(labels_test, pred_test,average='macro'))
```

```
[[0. 0. 0. ... 0. 0. 0.]
 [1. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 1. 0.]]
(784, 784)
(784, 10)
==========================================================================================
Confusion Matrix for train Data
[[   0   21    0    1    0    0 3834  411    0 1656]
 [   0   13    0    0    0    0 3523 1714    0 1492]
 [   0    1    0    0    0    0 3218  558    0 2181]
 [   0   57    0   26    1    0 2950 1262    0 1835]
 [   0   14    0    0    0    1 4236  346    0 1245]
 [   1   40    0    5    0    1 3889  367    0 1118]
 [   0    5    0    0    0    0 5001   67    0  845]
 [   0   25    1   21    0    1 3412 1165    0 1640]
 [   0    8    0    1    0    0 3437  500    0 1905]
 [   0   14    0    0    0    0 3388  793    1 1753]]
0.13265
0.058161352716203554
==========================================================================================
Confusion Matrix for Testing Data
[[  0   3   0   0   0   0 617  74   0 286]
 [  0   4   0   0   0   0 610 255   0 266]
 [  0   0   0   0   0   0 527  96   0 409]
 [  0   6   0   6   0   0 419 211   0 368]
 [  0   3   0   0   0   0 688  56   0 235]
 [  0   5   0   1   0   0 631  71   0 184]
 [  0   1   0   0   0   0 806  17   0 134]
 [  0   5   0   4   0   0 538 201   0 280]
 [  0   0   0   0   0   0 525  81   0 368]
 [  0   1   0   0   0   0 557 137   0 314]]
0.1331
0.05965723495188115
```

```
C:\Users\Farhan\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135: UndefinedMe
  'precision', 'predicted', average, warn_for)
```

```
In [16]:     diff_p()
```

```
Training Set Evaluation With paramenters :  16
Confusion Matrix for train Data
[[  18 1288    0    1    0    0 4594    3    0   19]
 [1586 1732    0   11    0    0 3391    4    0   18]
 [ 427 1878    0    0    1    0 3584   22    0   46]
 [  48  717    0    0    1    0 5218   42    0  105]
 [ 321 2126    0    8   10    0 3311    7    0   59]
 [  48 1125    0    8    1    0 4102   15    0  122]
 [ 112  368    0    1    1    0 5423    2    0   11]
 [ 554 2375    0    2    0    0 3165  134    0   35]
 [  31  681    0    3    1    0 5056    5    0   74]
 [ 355 2275    0    0   18    0 3232    3    0   66]]
0.12305
0.04621090844104506
Confusion Matrix for Test Data
```

```
C:\Users\Farhan\Anaconda3\lib\site-packages\sklearn\metrics\classification.py:1135: UndefinedMe
  'precision', 'predicted', average, warn_for)
```

```
[[  2 204   0   0   0   0 773   1   0   0]
 [288 277   0   0   0   0 566   0   0   4]
 [ 69 352   0   0   0   0 603   1   0   7]
 [  8 142   0   0   0   0 829   7   0  24]
 [ 55 366   0   3   2   0 539   6   0  11]
 [  4 204   0   1   0   0 663   1   0  19]
 [ 19  76   0   1   0   0 862   0   0   0]
 [103 413   0   0   0   0 490  19   0   3]
 [  5 119   0   1   0   0 838   1   0  10]
 [ 50 418   0   0   4   0 528   0   0   9]]
0.1171
0.0433589331725502
Training Set Evaluation With paramenters :  32
Confusion Matrix for train Data
[[   2   21    0  189    1    0  243 4111    0 1356]
 [  59  424    0   13    9    0 1236 4706    0  295]
 [   0  174    0  256    1    0 1013 3146    0 1368]
 [   5  553    0  560    7    4  787 3281    0  934]
 [   9  166    0  270    1    0 1899 1294    0 2203]
 [  12  245    0  450    6    1 1458 1860    0 1389]
```

```
 [   7   50    0  349    7    0  433 3540    0 1532]
 [  27 1052    0  175    8    1 1276 3094    0  632]
 [   0  174    0  182    4    0  676 3162    0 1653]
 [   1  519    0  721    7    1  277 3661    0  762]]
0.08795
0.0509213887588301
Confusion Matrix for Test Data
[[  0   3   0  34   0   0  39 699   0 205]
 [  7  75   0   2   2   0 199 810   0  40]
 [  1  24   0  44   0   0 178 550   0 235]
 [  2  84   0  98   0   0 148 534   0 144]
 [  1  25   0  47   1   0 308 230   0 370]
 [  1  36   0  71   3   1 233 309   0 238]
 [  0   2   0  44   0   0  95 543   0 274]
 [  6 177   0  26   1   0 192 530   0  96]
 [  0  40   0  44   0   1 106 506   0 277]
 [  1  88   0 158   1   0  44 577   0 140]]
0.094
0.0553573159733917
Training Set Evaluation With paramenters :  64
Confusion Matrix for train Data
[[   0 4001    1   14    0    0  307 1531    6   63]
 [   0 3328    5    8    0   10  657 2545    0  189]
 [   2 3311   10   17    1    0 1232 1230    4  151]
 [   1 2413    2    2    0    1 1728 1547    0  437]
 [   1  728    4   33    0    0 3265 1088    2  721]
 [   2 2076    5   10    1    2  935 2111    1  278]
 [  14 1146    2  105    3    0 1876 2544    5  223]
 [   0 1652   26  250    0    0 2283 1424    0  630]
 [   1 2465    3    3    0    5 1331 1598    0  445]
 [   1 1724   20   14    0    1 1809 1392    1  987]]
0.12715
0.07237715649703688
Confusion Matrix for Test Data
[[  1 660   0   3   0   0  51 251   0  14]
 [  0 553   1   0   0   2 113 435   0  31]
 [  0 593   1   4   0   1 213 195   0  25]
 [  0 362   0   3   0   0 274 298   0  73]
 [  0 108   0   5   0   0 589 155   1 124]
 [  2 326   0   1   0   0 164 349   0  50]
 [  4 203   1  24   0   0 362 324   2  38]
 [  0 273   1  48   0   0 400 219   1  86]
 [  0 347   0   1   0   1 265 287   0  73]
 [  0 249   2   1   0   0 330 258   0 169]]
0.1308
0.07491261939777667
Training Set Evaluation With paramenters :  128
Confusion Matrix for train Data
```

```
[[   0 4880    1    6    0    0  924  108    0    4]
 [   2 2366    0   24    0   27 4042  180    0  101]
 [   1 2022    0    3    0    2 3854   67    0    9]
 [   8 4400    2   43    2    1 1403  252    0   20]
 [   0 1791    0    3    0    0 4027   19    0    2]
 [   4 3469    2   32    0    4 1678  199    0   33]
 [   0  117    0    4    0    1 5764   16    0   16]
 [   0 3090    0   19    1    1 3028  120    0    6]
 [   0 2628    0    2    1   32 3031  125    0   32]
 [   0 2230    0   10    0    0 3619   80    0   10]]
0.13845
0.05001214792000401
Confusion Matrix for Test Data
[[  0 770   0   2   0   0 181  26   0   1]
 [  0 366   0   1   0   5 713  34   0  16]
 [  0 350   0   1   0   0 665  16   0   0]
 [  0 720   0  21   0   0 236  30   0   3]
 [  0 304   0   0   0   0 672   6   0   0]
 [  1 571   0   2   0   0 274  35   0   9]
 [  0  26   0   1   0   0 929   2   0   0]
 [  0 516   0   2   0   0 495  15   0   0]
 [  0 405   0   0   0   2 549  14   0   4]
 [  0 367   0   0   0   0 622  18   0   2]]
0.1333
0.04962965133290776
Training Set Evaluation With paramenters :  256
Confusion Matrix for train Data
[[   0  325   22   25    6    0 3661 1261    1  622]
 [   0   44    3    2    1   10 4322 1312    0 1048]
 [   1   78    0    1    0    1 3095 1594    1 1187]
 [   6 1049    9   17    6    0 1535 2805    2  702]
 [   0   68    2   28    4    0 4609  460    0  671]
 [   6  832   10   20    4    3 2901 1204    0  441]
 [   3   26    2   12    4    1 4919  218    0  733]
 [   8  275   15  120    1    0 4794  576    7  469]
 [   0  117    5    5    5    2 2512 2006    1 1198]
 [   3  299    2   24    8    5 3227 1284    2 1095]]
0.11098333333333334
0.04699919189266619
Confusion Matrix for Test Data
[[  1  42   2   4   1   0 591 221   1 117]
 [  0   5   1   0   0   3 774 192   0 160]
 [  0   8   0   0   1   0 506 298   1 218]
 [  0 198   3   6   0   0 219 466   1 117]
 [  0   9   0   2   0   0 779  95   0  97]
 [  2 126   0   3   2   2 480 204   0  73]
 [  0   5   0   0   0   0 793  37   0 123]
 [  0  44   2  25   1   0 776  95   0  85]
```

```
 [  0  18   0   1   1   1 378 352   0 223]
 [  0  38   0   1   2   0 517 231   0 220]]
0.1122
0.04974772374268875
Training Set Evaluation With paramenters :  512
Confusion Matrix for train Data
[[   0   50    0    0    1    6 4364  525    0  977]
 [   0  222    0   11    6    1 4286 1354    0  862]
 [   0   60    0    3    4    1 4143  593    0 1154]
 [   0  520    0   74    7   35 4026  921    1  547]
 [   0  102    0    0    1    7 5310  169    0  253]
 [   0  195    0   19    1    6 4554  267    0  379]
 [   1   77    0    8    2    4 4730  273    1  822]
 [   0  290    0   16    0    6 3808 1253    0  892]
 [   0   72    0    7    2    4 4163  709    0  894]
 [   0  143    0    1    0   26 4624  531    0  624]]
0.11516666666666667
0.05563326369604917
Confusion Matrix for Test Data
[[  0   5   0   0   0   0 702 108   0 165]
 [  0  39   0   1   1   2 765 207   0 120]
 [  0   9   0   1   1   0 709 112   0 200]
 [  0  76   0  15   1   3 632 165   0 118]
 [  0  24   0   0   0   1 901  20   0  36]
 [  0  40   0   4   1   2 737  41   0  67]
 [  0   7   0   0   0   0 781  34   0 136]
 [  2  56   0   5   1   2 620 201   0 141]
 [  0  15   0   0   1   1 667 103   0 187]
 [  0  20   0   0   0   1 785  91   0 112]]
0.115
0.05655488128762287
```

According to accuracy results, model with 128 parameters shows best accuracy. and show best F1 Score with 64 parameters.