assignment10

December 6, 2018

1 Assignment09

2 ID: 20155093

3 Name: Sonjeongseol

4 Github Link: https://github.com/wjdtjf1234/assignment01.gitű

5 Importing APIs

```
In [1]: import numpy as np
        import matplotlib.pyplot as plt
        from numpy.linalg import inv
        file_data_train = "mnist_train.csv"
        file_data_test  = "mnist_test.csv"
```

# Declaring a normalizing function for training&testing data

```
In [2]: def normalize(data):
            data_normalized = (data - min(data)) / (max(data) - min(data))
            return(data_normalized)
```

6 Reading data from mnist train&test data sets

```
In [3]: 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
```

7 Arrays for storing labels&image data

```
In [4]: list_image_train    = np.zeros((size_row * size_col, num_train), dtype=float)
        list_label_train    = np.zeros(num_train, dtype=int)
        list_image_test     = np.zeros((size_row * size_col, num_test), dtype=float)
        list_label_test     = np.zeros(num_test, dtype=int)
        list_label_test2    = np.zeros(num_test, dtype=int)
```

8 Data normalizing & assigning labels for each data

```
In [5]: def label_assign(n):
            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    #image sets
                count += 1

            for x in range(count):
                if(list_label_train[x]!=n):
                    list_label_train[x]=-1
                else:
                    list_label_train[x]=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_label_test2[count]    = label
                list_image_test[:, count]   = im_vector
                count += 1

            for x in range(count):
                if(list_label_test[x]!=n):
                    list_label_test[x]=-1
                else:
                    list_label_test[x]=1
```

9 Generating random normally distributed vectors

```
In [6]: r=np.empty((784,4000),dtype=float)
        for i in range(4000):
            for j in range(size_col*size_row):
                r[j,i]=np.random.normal(loc=0.0,scale=1.0)
```

10 Some Executions to get model parameters from training data via pseudo-inverse

```
In [7]: def ps_inv(a):
            y=np.empty((20000,a),dtype=float)
            for k in range(20000):
                for l in range(a):
                    y[k,l]=np.inner(r[:,l],list_image_train[:,k])
            y_t=y.transpose()
            gram_y=np.matmul(y_t,y)
            gram_y_i=inv(gram_y)
            ps_inv_y=np.matmul(gram_y_i,y_t)
            list_label_train_t=list_label_train[0:20000].transpose()
            model_parameter=np.zeros((a,1),dtype=float)
            model_parameter=np.matmul(ps_inv_y,list_label_train_t)
            return model_parameter
```

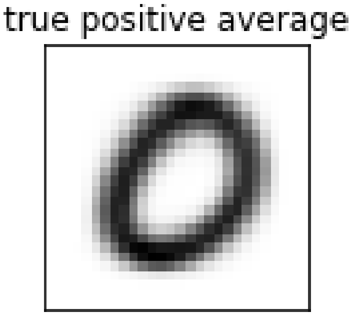
11 Defining a Bi-partitioning function

```
In [8]: def bpf(n,a):
            tp_average=np.zeros((size_row*size_col),dtype=float)
            d_t=np.zeros((1,a),dtype=float)
            table=np.zeros(11,dtype=int)
            label_test=np.empty(num_test,dtype=int)
            mp=ps_inv(a)
            for i in range(num_test):
                for l in range(a):
                    d_t[0,l]=np.inner(r[:,l],list_image_test[:,i])
                d=np.matmul(d_t,mp)
                if(d>=0):
                    if(list_label_test[i]==1):
                        table[n]+=1        #TP
                        for j in range(size_row*size_col):
                            tp_average[j]+=list_image_test[j, i]
                    else:
                        table[list_label_test2[i]-1]+=1     #FP
                elif(d<0):
                    if(list_label_test[i]==1):
                        table[10] += 1         #FN
            tp_average/=table[n]
            plt.subplot(2,1,1)
            plt.title("true positive average")
            plt.imshow(tp_average.reshape((size_row, size_col)), cmap='Greys', interpolation='
            frame   = plt.gca()
            frame.axes.get_xaxis().set_visible(False)
            frame.axes.get_yaxis().set_visible(False)
            plt.show()
            print("%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t%d\t"%(table[0],table[1],table[2],table
            print("%d"%(table[10]))
            return table
```

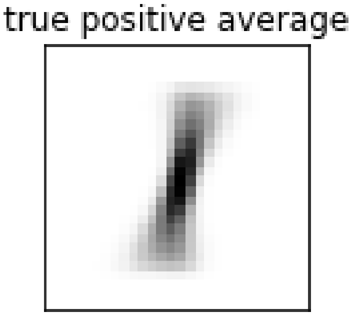
```
In [9]: def pf(a):
            for n in range(10):
                label_assign(n)
                bpf(n,a)
```

12 A result using 100 random vectors

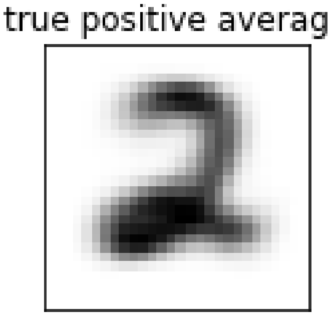
```
In [11]: pf(100)
```



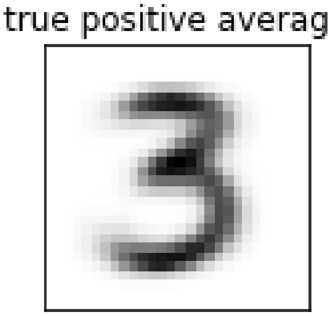
```
886         12         7         1         10         24         6         4         5         0
94
```



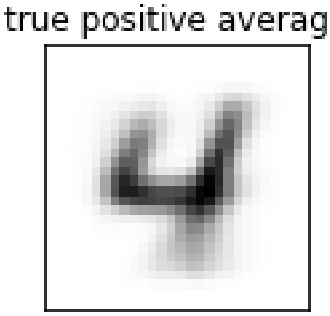
```
0         1054         1         3         8         5         23         11         1         0
108
```



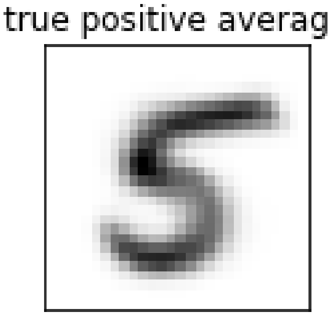
```
2         0         627         4         3         7         8         0         3         0
421
```



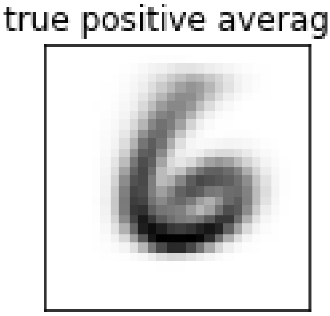
```
1         20         0         610         50         1         7         5         4         0
417
```



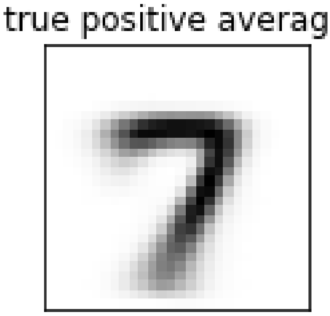
```
1         5         0         0         574        15        14        1         11        0
414
```



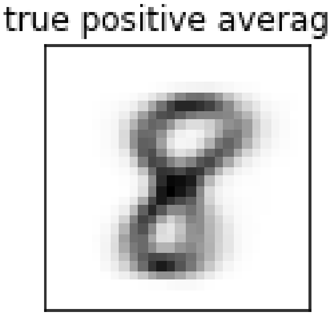
```
1         0         11        1         0          333       1         3         1         0
590
```



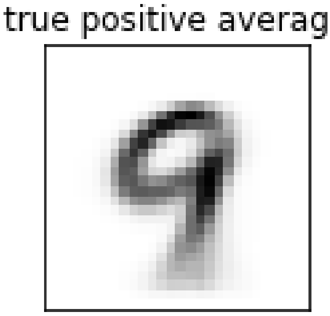
```
4         36         10        5         18        0         783        5         1         0
212
```



```
0          10          11          0          14          1          0          769          8          0
264
```



```
18          21          6          2          39          7          3          0          326          0
668
```



```
0          0          3          24          11          0          41          0          0          360
653
```

13 The confusion matrix M, which of partition function using 100 parameters:

14 886/ 12 / 7 / 1 / 10 / 24 / 6 / 4 / 5 / 0

15 0 /1054/ 1 / 3 / 8 / 5 / 23 / 11 / 1 / 0

16 2 / 0 / 627 / 4 / 3 / 7 / 8 / 0 / 3 / 0

17 1 / 20 / 0 / 610 / 50 / 1 / 7 / 5 / 4 / 0

18 1 / 5 / 0 / 0 / 574 / 15 / 14 / 1 / 11 / 0

19 1 / 0 / 11 / 1 / 0 / 333 / 1 / 3 / 1 / 0

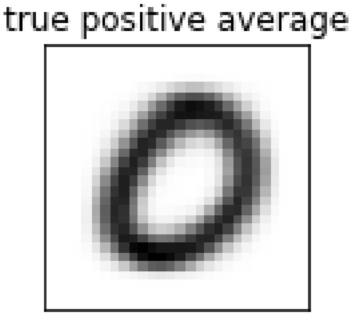
20 4 / 36 / 10 / 5 / 18 / 0 / 783 / 5 / 1 / 0

21 0 / 10 / 11 / 0 / 14 / 1 / 0 / 769 / 8 / 0

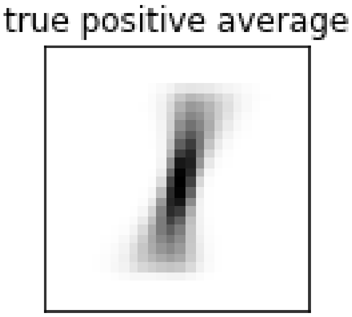
22 18 / 21 / 6 / 2 / 39 / 7 / 3 / 0 / 326 / 0

23 0 / 0 / 3 / 24 / 11 / 0 / 41 / 0 / 0 / 360

```
In [12]: pf(300)
```

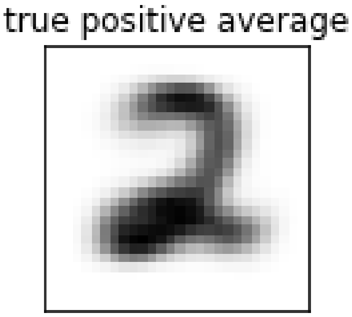


```
918         15         6         1         6         20         10         5         6         0
62
```



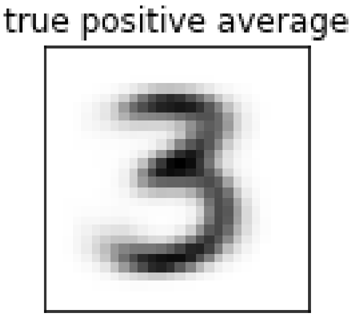
true positive average

```
0          1073          2          7          8          5          17          13          2          0
81
```



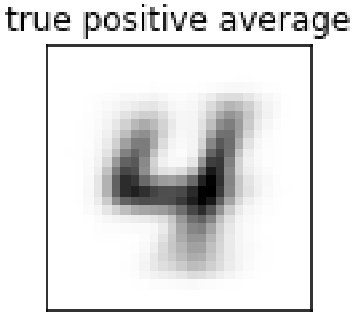
true positive average

```
3          0          686          6          3          8          3          3          1          0
356
```

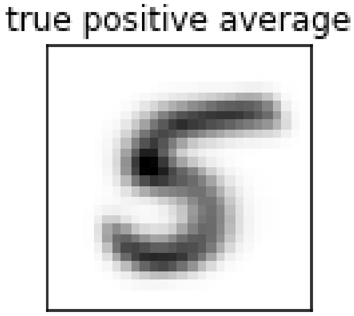


true positive average

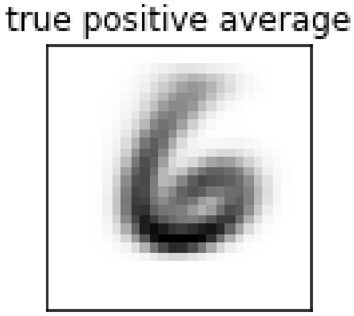
```
1         17         0         672         41         1         12         4         3         0
345
```



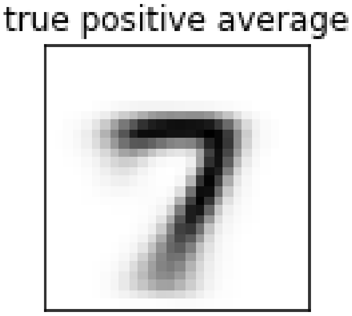
```
1         5          0         0           718        15         11         3         16        0
270
```



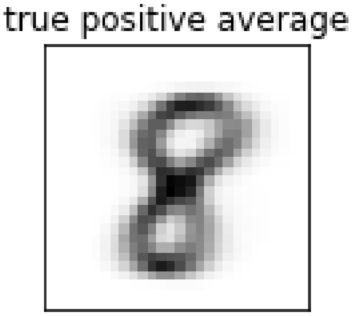
```
1         1          6         0           0          408        0          15        1         0
508
```



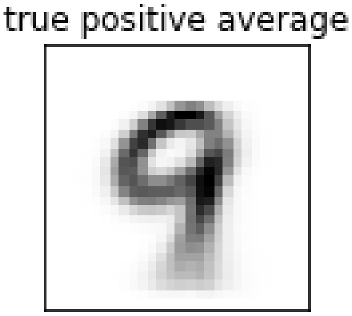
```
4        46       4        6        18       0        814      8        0        0
175
```



```
0        10       13       0        15       0        0        798      14       0
232
```



```
12       22       6        6        46       5        1        0        395      0
600
```



```
0         3         3         25        14        0         51        2         0         533
478
```

24 The confusion matrix M, which of partition function using 300 parameters:

25 918 / 15 / 6 / 1 / 6 / 20 / 10 / 5 / 6 / 0

26 0 / 1073 / 2 / 7 / 8 / 5 / 17 / 13 / 2 / 0

27 3 / 0 / 686 / 6 / 3 / 8 / 3 / 3 / 1 / 0

28 1 / 17 / 0 / 672 / 41 / 1 / 12 / 4 / 3 / 0

29 1 / 5 / 0 / 0 / 718 / 15 / 11 / 3 / 16 / 0

30 1 / 1 / 6 / 0 / 0 / 408 / 0 / 15 / 1 / 0

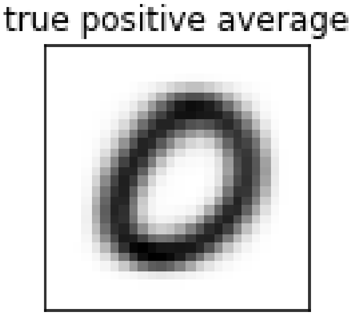
31 4 / 46 / 4 / 6 / 18 / 0 / 814 / 8 / 0 / 0

32 0 / 10 / 13 / 0 / 15 / 0 / 0 / 798 / 14 / 0

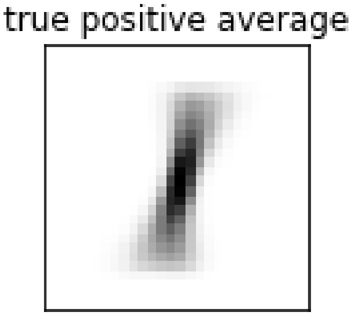
33 12 / 22 / 6 / 6 / 46 / 5 / 1 / 0 / 395 / 0

34 0 / 3 / 3 / 25 / 14 / 0 / 51 / 2 / 0 / 533

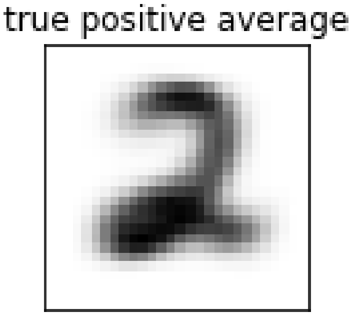
```
In [10]: pf(600)
```



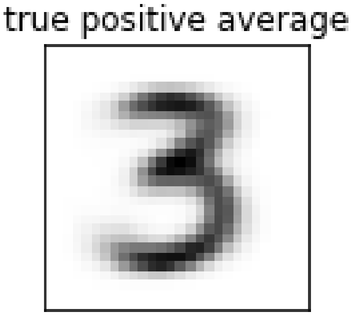
```
922          15          7          0          9          16          8          6          7          0
58
```



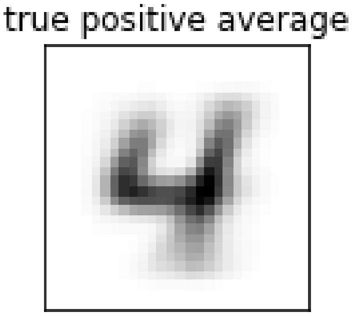
```
0          1074          2          8          7          6          15          13          2          0
84
```



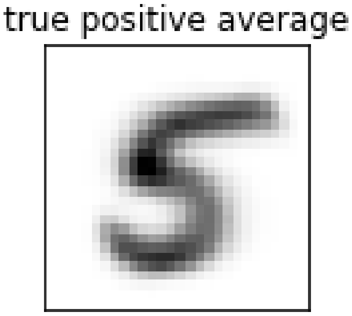
```
2          0          678          8          5          11          1          4          3          0
368
```



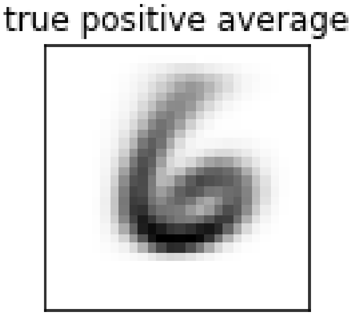
```
2        25        0        672        41        2        9        3        4        0
346
```



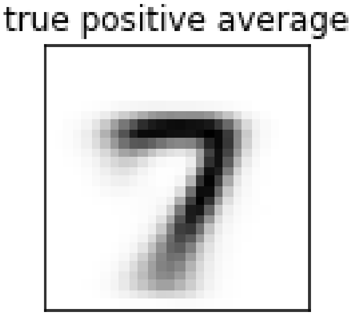
```
1        8        0        0        751        18        15        4        16        0
240
```



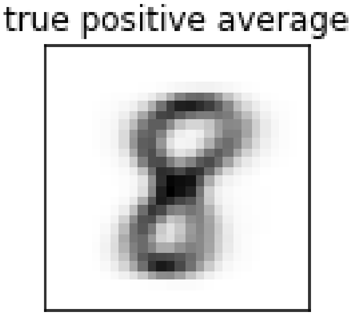
```
0        1        6        4        0        461        1        25        2        0
458
```



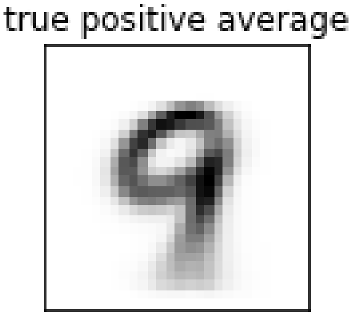
```
4         51         5         7         18         0         812         7         0         0
169
```



```
0         8         14         1         15         2         0         804         26         0
228
```



```
14         25         9         7         49         8         3         0         404         0
594
```



```
0         4         4         42        17        0         60        4         0         569
442
```

35 The confusion matrix M, which of partition function using 600 parameters:

36 922 / 15 / 7 / 0 / 9 / 16 / 8 / 6 / 7 / 0

37 0 / 1074 / 2 / 8 / 7 / 6 / 15 / 13 / 2 / 0

38 2 / 0 / 678 / 8 / 5 / 11 / 1 / 4 / 3 / 0

39 2 / 25 / 0 / 672 / 41 / 2 / 9 / 3 / 4 / 0

40 1 / 8 / 0 / 0 / 751 / 18 / 15 / 4 / 16 / 0

41 0 / 1 / 6 / 4 / 0 / 461 / 1 / 25 / 2 / 0

42 4 / 51 / 5 / 7 / 18 / 0 / 812 / 7 / 0 / 0

43 0 / 8 / 14 / 1 / 15 / 2 / 0 / 804 / 26 / 0

44 14 / 25 / 9 / 7 / 49 / 8 / 3 / 0 / 404 / 0

45 0 / 4 / 4 / 42 / 17 / 0 / 60 / 4 / 0 / 569

46 The best F1 Score can be found when p= 600, and the value is 0.786 .