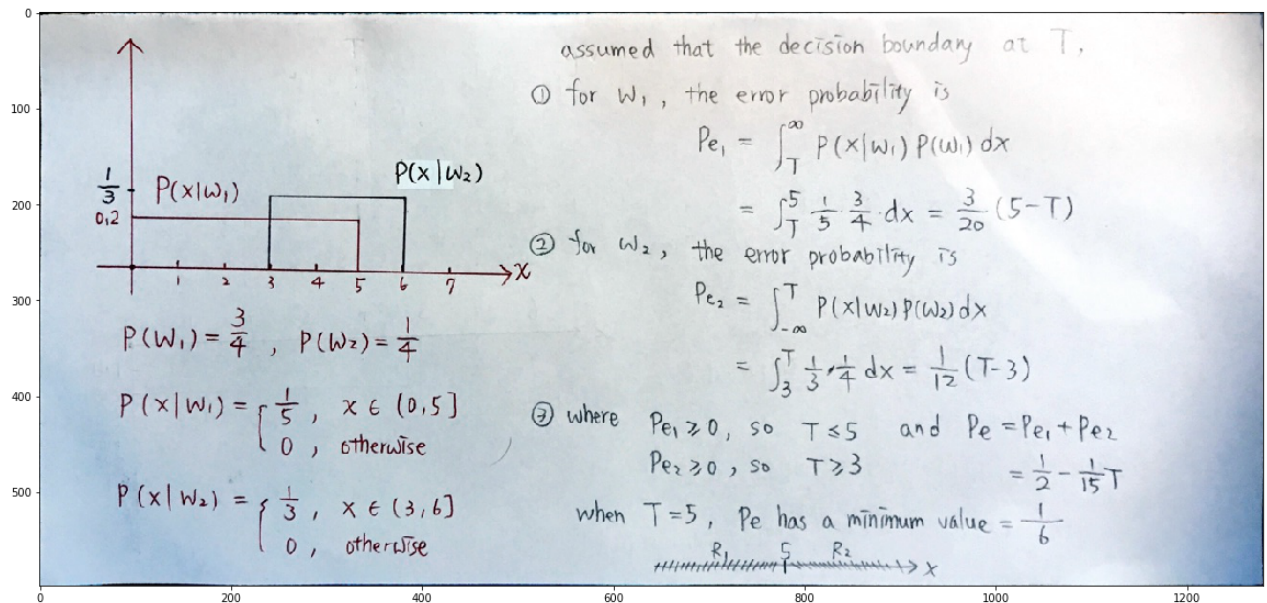


## DLCV HW#1 - d05921027 張鈞閔

### Problem 1

```
In [1]: 1 import cv2
2 import matplotlib.pyplot as plt
3 %matplotlib inline
4
5 prob1 = cv2.imread("prob1.jpg", 1)
6 plt.figure(figsize=(20,20))
7 plt.imshow(prob1)
```

Out[1]: <matplotlib.image.AxesImage at 0x11b621c50>



### Problem 2

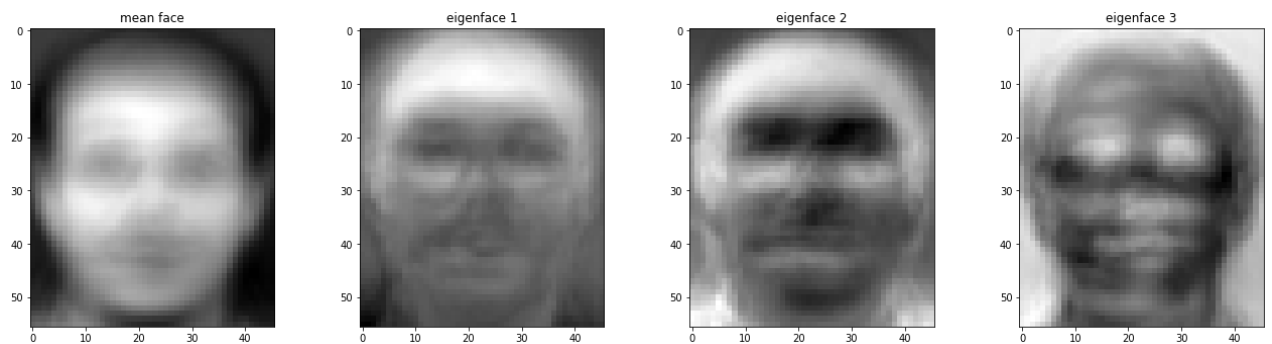
```
In [2]: 1 import os
2 import numpy as np
3 import pandas as pd
4
5 from sklearn.decomposition import PCA
6 from sklearn.model_selection import GridSearchCV
7 from sklearn.metrics import accuracy_score
8 from sklearn.neighbors import KNeighborsClassifier
9
10 data_dir = "data/"
```

```
In [3]: 1 ftrain = []
2 ytrain = []
3 for i in range(40):
4     for j in range(6):
5         ftrain.append(os.path.join(data_dir, str(i+1)+"_"+str(j+1)+".png"))
6         ytrain.append(i)
7 print("number of training images:", len(ftrain))
8
9 original_shape = cv2.imread(ftrain[0], 0).shape
10 print("original shape:", original_shape)
11
12 xtrain = []
13 for fn in ftrain:
14     tmp = cv2.imread(fn, 0)
15     tmp = tmp.reshape(-1)
16     xtrain.append(tmp)
17 xtrain = np.array(xtrain)
18 ytrain = np.array(ytrain)
19 print("training dataset for PCA, its shape =", xtrain.shape)
```

number of training images: 240  
original shape: (56, 46)  
training dataset for PCA, its shape = (240, 2576)

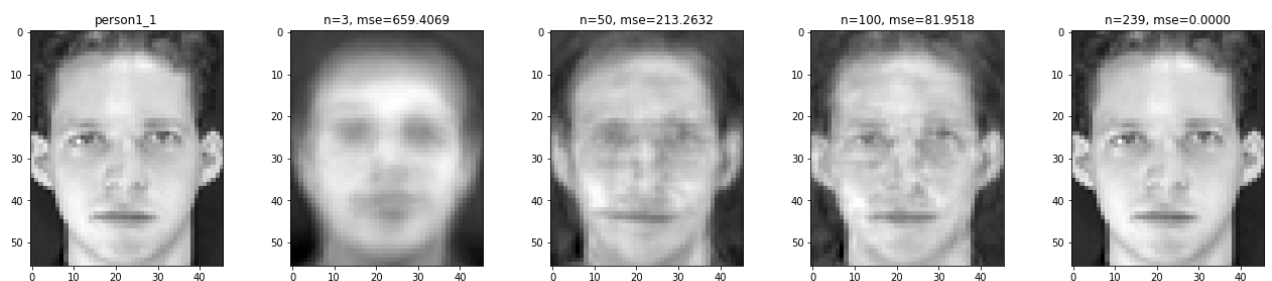
## (a) mean face and the first three eigenfaces

```
In [4]: 1 mean_face = np.mean(xtrain,axis=0)
2 mean_face = np.reshape(mean_face,newshape=original_shape)
3
4 pca = PCA(n_components=min(xtrain.shape)-1)
5 e = pca.fit(xtrain-mean_face.reshape(-1))
6 pc1 = np.reshape(e.components_[0],newshape=original_shape)
7 pc2 = np.reshape(e.components_[1],newshape=original_shape)
8 pc3 = np.reshape(e.components_[2],newshape=original_shape)
9
10 plt.figure(figsize=(20,16))
11 plt.subplot(141)
12 plt.imshow(mean_face, cmap='gray')
13 plt.title("mean face")
14 plt.subplot(142)
15 plt.imshow(pc1,cmap='gray')
16 plt.title("eigenface 1")
17 plt.subplot(143)
18 plt.imshow(pc2,cmap='gray')
19 plt.title("eigenface 2")
20 plt.subplot(144)
21 plt.imshow(pc3,cmap='gray')
22 plt.title("eigenface 3")
23 plt.subplots_adjust(top=0.92, bottom=0.08, left=0.10, right=0.95, hspace=0.25,wspace=0.35)
24 plt.show()
25 plt.close()
```



## (b) reconstruction

```
In [5]: 1 target = cv2.imread("data/1_1.png",0)
2 plt.figure(figsize=(20,16))
3 plt.subplot(1,5,1)
4 plt.title("person1_1")
5 plt.imshow(target, cmap="gray")
6
7 target = np.reshape(target,newshape=(1,-1))
8 e_target = e.transform(target - mean_face.reshape(-1))
9
10 n = [3,50,100,239]
11 for k in range(len(n)):
12     tmp = np.dot(e_target[0:n[k]], e.components_[0:n[k]]) + mean_face.reshape(-1)
13     mse = np.mean((tmp - target)**2)
14     tmp = np.reshape(tmp, newshape=original_shape)
15     plt.subplot(1,5,k+2)
16     plt.title("n=%s, mse=%.4f" % (n[k], mse))
17     plt.imshow(tmp, cmap='gray')
18 plt.subplots_adjust(top=0.92, bottom=0.08, left=0.1, right=0.95, hspace=0.25,wspace=0.35)
19 plt.show()
20 plt.close()
```



## (c) kNN in projected spaces

```
In [6]: 1 ptrain = e.transform(xtrain-mean_face.reshape(-1))
2 ytrain = np.array(ytrain)
3
4 params = {'n_neighbors':[1,3,5]}
5 kNN = KNeighborsClassifier()
6 clf = GridSearchCV(kNN, params,cv=3)
7
8 n = [3, 50, 159]
9 res = dict()
10 for k in n:
11     clf.fit(ptrain[:,k], ytrain)
12     res['n='+str(k)] = np.array(clf.cv_results_['mean_test_score'])
13 res = pd.DataFrame.from_dict(res,orient='index')
14 res.columns = ['k=1','k=3','k=5']
15 print(res)
```

	k=1	k=3	k=5
n=3	0.708333	0.587500	0.487500
n=50	0.929167	0.875000	0.775000
n=159	0.925000	0.870833	0.745833

**Best choice: k=1, n=50**

```
In [7]: 1 k, n = 1, 50
```

```
In [8]: 1 # get testing filenames and labels
2 ftest = []
3 ytest = []
4 for i in range(40):
5     for j in range(6,10):
6         ftest.append(os.path.join(data_dir, str(i+1)+"_"+str(j+1)+".png"))
7         ytest.append(i)
8 print("number of test images:",len(ftest))
9
10 # read testing images
11 xtest = []
12 for fn in ftest:
13     tmp = cv2.imread(fn,0)
14     tmp = tmp.reshape(-1)
15     xtest.append(tmp)
16 xtest = np.array(xtest)
17 ytest = np.array(ytest)
18 print("testing dataset, its shape =",xtest.shape)
19
20 # Project images onto the principal components
21 ptest = e.transform(xtest-mean_face.reshape(-1))
22 print("projected testing dataset, its shape=", ptest.shape)
```

```
number of test images: 160
testing dataset, its shape = (160, 2576)
projected testing dataset, its shape= (160, 239)
```

```
In [9]: 1 # kNN model with optimized hyper-parameter (k,n)
2 bestkNN = KNeighborsClassifier(n_neighbors=k)
3 bestkNN.fit(ptrain[:,n], ytrain)
4 ypred = bestkNN.predict(ptest[:,50])
5 print("overall accuracy:",accuracy_score(y_pred=ypred, y_true=ytest))
```

```
overall accuracy: 0.9625
```