Project 4 Face Recognition

Guixiang Zhang

Experiment 1: Cross-validation

MATLAB file: crossvalidation.m

In this experiment, I randomly select 1 sample image from each identity as the test set. The remaining 2500 images are the training set.

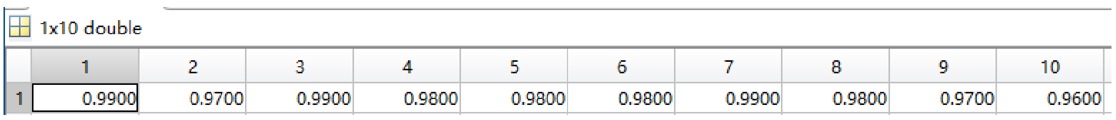
Steps:

1. Vectorize 2500 images into the training set matrix “**training**” whose dimension is 2500-by-19800.
2. Compute the mean of each column **u**, and subtract it to get matrix
$$\boldsymbol{P = training - u}$$
3. Do SVDs of **P** get the matrix **M** which is composed of vectors associate to the first 100 singular values. This is same as using the function “eigs()” to get the eigenvectors.

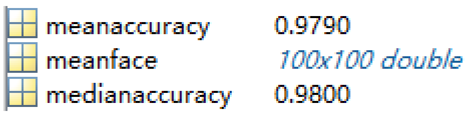
This is the first column of M, the first principle eigenface of this training set.

4. Project training set to the subspace of face and get
$$\boldsymbol{trainingp = training * M}$$
5. Compute mean face **meanface** and covariance matrix **sigmap** for each identity in the training set after projection.
6. Get the test set and project it to the subspace to get matrix **testp** too.
7. Assume the training images of each identity is in normal distribution. Use Mahalanobis distance to find the closest distribution for each test image.
$$\boldsymbol{d = \left(\hat{t} - \hat{\mu}_j\right)^T * \widehat{\Sigma_{x_j}}^{-1} * \left(\hat{t} - \hat{\mu}_j\right)}$$
where **d** is the Mahalanobis distance which represent the similarity of the test image and j^{th} identity. $\boldsymbol{\hat{t}}$ is the test image after projection which is chosen from **testp.** $\boldsymbol{\hat{\mu}_j}$ and $\boldsymbol{\widehat{\Sigma_{x_j}}}$ are the mean face and covariance matrix of j^{th} identity respectively.
8. get the recognition accuracy of this test set save into the vector **accuracy**.
9. Run step 1~8 **k** times to get the mean and median of accuracy, **meanaccuracy** and **medianaccuracy.**

Here is the result when **k**=10.


1x10 double

	1	2	3	4	5	6	7	8	9	10
1	0.9900	0.9700	0.9900	0.9800	0.9800	0.9800	0.9900	0.9800	0.9700	0.9600



meanaccuracy	0.9790
meanface	*100x100 double*
medianaccuracy	0.9800

The mean and median of accuracy are 97.9% and 98.0%.

Remark:

1. The reason we use only the first 100 eigenvectors is that they have more than 95% information of the whole distribution, which make sure the accuracy, and these principle components can make computation less expansive.
2. In step 5, the rank of the covariance matrix **sigmap** is less than 25. If we only use 24 or less vectors to make the projection matrix **M**, the information isn't enough. The accuracy will be low. If we use, like 100 vectors, the dimension of each covariance matrix is 100-by-100, the matrix maybe singular. But we need to inverse it. So, we can use the trick we used when we discussed LDA.
$$\boldsymbol{sigmap} = \boldsymbol{sigmap} + \boldsymbol{\varepsilon I}$$
where $\boldsymbol{\varepsilon}$ is a tiny scalar and $\boldsymbol{I}$ is the identity matrix. This will not influence the matrix much but make it invertible.
3. Project the training matrix to the subspace first and then compute mean and covariance is the same as using the training matrix before projection to compute mean and covariance and then project them to the subspace. But the first way is faster.
4. The accuracy is enough good, so we don't need to consider the illumination, expression and occlusion in this experiment.

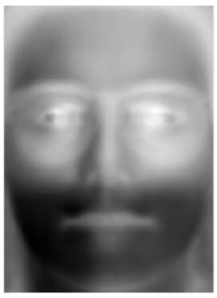
Note: in the submitted code, I set **K**=1. So, there is only 1 accuracy value.

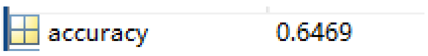
Experiment 2: Duplicates

Mostly like the first experiment, I also use Mahalanobis distance (in step 7) to do recognition. But this time. the training set is the 1300 images of first session. I use 2 different ways to get the result and compare the difference.

1. MATLAB file: duplicates.m

Here are some results.

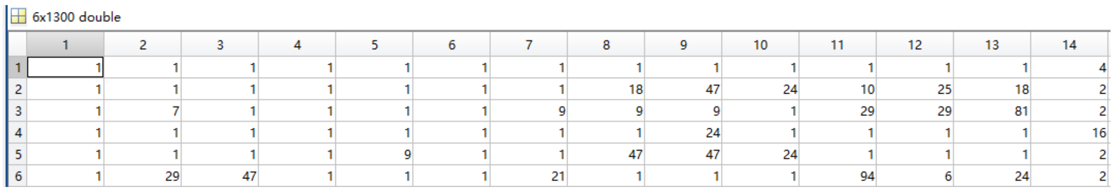

The first principle eigenface of this training set



accuracy	0.6469

global accuracy is 64.69%

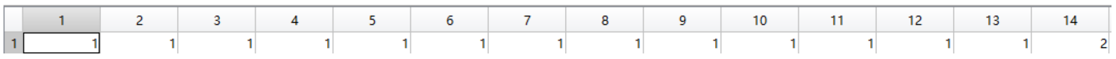
2. MATLAB file duplicatesdivide.m, meansigmaM.m, recogonition.m
   In this algorithm, I divide each image into 6 part. And treat them independently. So, for each test image, I have 6 classifications I just use function "mode" to get the result.



6x1300 double

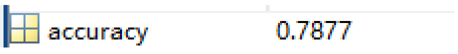
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	1	1	1	1	1	1	1	1	1	1	1	4
2	1	1	1	1	1	1	1	18	47	24	10	25	18	2
3	1	7	1	1	1	1	9	9	9	1	29	29	81	2
4	1	1	1	1	1	1	1	1	24	1	1	1	1	16
5	1	1	1	1	9	1	1	47	47	24	1	1	1	2
6	1	29	47	1	1	1	21	1	1	1	94	6	24	2

The result the divided second session images of first identity



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	1	1	1	1	1	1	1	1	1	1	1	2

The classification result of these 14 test images



accuracy	0.7877

global accuracy is 78.77%

Discussion

1. In fact, there are some images that have 6 different classifications. It should be labeled incorrect, but the function "mode" will choose one of the 6 number which may be correct. So, the real accuracy maybe a little bit less than the result.
2. As for the second experiment. This accuracy is not that high like in the first experiment. And from the result I see, the images with occlusion is more difficult to be recognized. The reason maybe the information of the identity is not enough.
3. To improve the recognition result I divide the image. But I don't remove the occlusion parts to recognition. If I can let the computer do it automatically It may improve the result further.
4. And how to divide the image may also influence the result. Maybe there is a better plan.
5. There are some expression and illumination change in the image, but when I do some additional tests, it seems the divide algorithm can recognize these images correctly with an accuracy about 90%. So, I do not add some expression and illumination invariant metric, such that using optical flow.