Pattern Recognition Coursework I

Peter Sarvari
CID 987075
ps5714@ic.ac.uk

Orsika Bojtar
CID 01267809
ob116@ic.ac.uk

Imperial College London
S. Kensington, London SW7 2AZ

Abstract

Abstract: In the following, we examine and compare generative and discriminative subspace learning methods. We use PCA to reconstruct images, then compare the nearest neighbor classification accuracy of images in the PCA and LDA subspace. In addition, we use ensemble learning to increase the classification accuracy achieved in the LDA subspace. The results with 520 images of 52 classes confirm the theoretical expectations.

1. Q1

1.1. Eigenfaces

a) The original data set contained 10 face images of 52 different people. We divided the samples belonging to each class randomly into training and testing sets such that the training set contains 70% of every class. In our case this corresponds to 7 training and 3 testing images for each of the 52 classes (364 training and 156 testing samples). For reproducibility of the results, we used rng(7) in Matlab. The eigenfaces, mean image, and eigenvalues of the training set are:

Figure 1. The first 50 eigenfaces

Figure 2. Mean training image

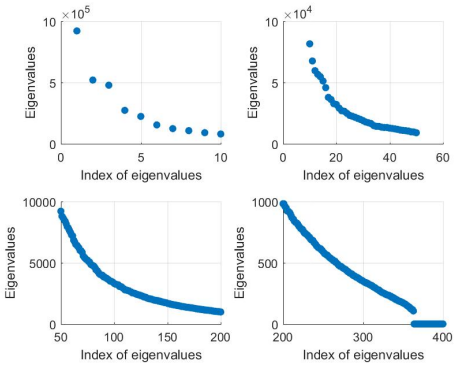

Figure 3. The first 400 eigenvalues

From Figure 3 it can be seen that we have 363 nonzero eigenvalues. This is N-1, where N is the number of training samples. The reason is the following: If we calculate the Hermitian of a matrix of rank N then the result will have rank N. However, since we subtract the mean image from each sample, we lose a rank, so the matrix we take the Hermitian of will have rank N-1 and hence the result will also have rank N-1. The accuracy of the nearest neighbor classifier was calculated for subspaces using 33 to 363 eigenvectors in steps of 10. The best result was achieved using 181 eigenvectors and considering only one neighbor in the nearest neighbor algorithm. The resulting accuracy was 55.13%.

b) The eigenvalues obtained from $S_1 = 1/N(AA^T)$ and $S_2 = 1/N(A^TA)$ were the same aside from a small ($< 10^{-9}$) numerical error. The relationship between the eigenvectors $AA^Tu_i = \lambda_i u_i$ and $A^TAv_i = \lambda_i v_i$ was also as expected from theory. Transforming the first 363 eigenvectors of S_2 (these are the eigenvectors corresponding to nonzero eigenvalues) according to the equation $u_i = Av_i$ and normalizing them resulted in the eigenvectors of S_1. Occasionally a difference in the direction ($\pm u_i$) or a small numerical error appeared in the results.

1.2. Application of eigenfaces

a) We have produced three diagrams corresponding to 30, 100 and 300 eigenvectors used in the reconstruction. The first line is 6 training samples, the second is their reconstructions, the third is 6 test samples and the fourth is their reconstructions. As expected, the reconstruction errors become less as we increase the number of eigenvectors. The process works less well for the testing set images, since they have not been part of the eigenface-space generation.

Figure 4. Reconstruction using 30 eigenfaces

Figure 5. Reconstruction using 100 eigenfaces

Figure 6. Reconstruction using 300 eigenfaces

There are two methods to do PCA. Both use the eigenvectors of the covariance matrix, but

- one uses the mean-normalized (the mean images subtracted from all images) images for projection and during reconstruction the mean image is added back,
- whereas the second projects the original images and during reconstruction the reconstruction of the projection of the mean image is added to the reconstructed images rather than the mean itself.

We have found that in both cases the error is the same and is identical to the sum of the eigenvalues of the unused eigenvectors, so our reconstruction errors are the same as predicted from theory. The following table summarizes the errors in three cases.

No of eigenvectors used	Error
30	$7.91 * 10^5$
100	$2.58 * 10^5$
300	$1.47 * 10^4$

Table 1. Reconstruction error with respect to the number of eigenfaces used

b) As described in 1.1 a) we performed cross-validation to find the best parameters and obtained 55.13% recognitions accuracy with them. The confusion matrix can be seen on Figure 7. Since we only have three test samples for each class, 4 possible recognition accuracies can exist for each class (100, 67, 33 and 0%, denoted by yellow, green, light blue and dark blue, respectively).

Complete failure cases are when none of the test samples from a particular class are classified as belonging to that class. This happens for class 1, 11, 22, 25, 30, 34, 36, 41, 43, 44 and 48 as can be seen on the graph. Two test samples of class 11 were classified as 30, of 36 were classified as 48, of 44 were classified as 30, of 48 were classified as 29 and

two test samples of 50 were classified as 38 (see on Figure 8). The most successful cases are the indices corresponding to the yellow rectangles on the diagonal.

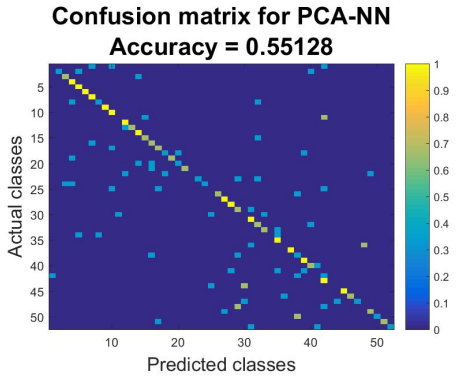

Figure 7. Confusion matrix for PCA-NN classification using the best parameters

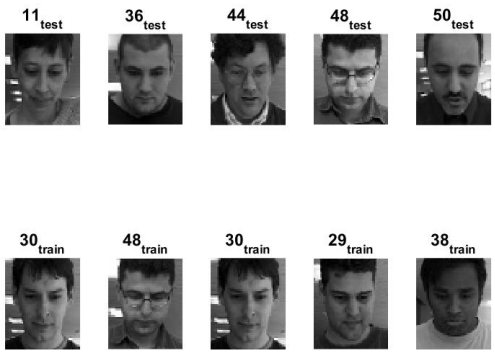

Figure 8. Badly classified results visualized. Each testing sample (upper row) have been classified as its corresponding class below. Similarity is visually noticeable.

The complexity of the most effective implementation of the nearest neighbour classifier is O(n_{test}*n_{train}*d), where n_{test} is the number of test samples, n_{train} is the number of train samples and d is the dimension. In case the quickselect algorithm is used, this is also the complexity of the k nearest neighbour. So time will linearly scale with the dimension, for example, if we use half the number of eigenvectors, the nearest neighbour algorithm is expected to execute twice as quick. Since the nearest neighbour algorithm only stores scalar pairwise distances corresponding to each pair of test-training points, the dimension of the data does not affect the memory required by the algorithm.

2. Q2-2

2.1. PCA-LDA

2.1.1 Recognition accuracies by varying the parameter value, M_{PCA}, M_{LDA}

We also experiment with the number of neighbours used to classify the test sample. The results are summarized in Table 2 for the 3 neighbour case and in the Appendix for other cases. We decided to investigate M_{PCA} in steps of 50. We cannot go above N-C (312), otherwise we encounter problems during the inversion of the within class covariance matrix in LDA. When deciding how many neighbours to use, it is not advisable to go above 7, since 7 is the number of training samples belonging to one class. The best prediction is achieved using $M_{PCA} = 151$, $M_{LDA} = 43$ and 3 neighbours. The accuracy achieved was 79.49%. Note however, that this is not the accuracy we expect to achieve on an unseen dataset, since hyperparameters (M_{PCA}, M_{LDA} and number of neighbours taken into account) were tuned on the test set.

M_{PCA}/M_{LDA}	3	23	43	51
51	0.3013	0.6667	0.6218	0.5833
101	0.2692	0.7756	0.7244	0.7051
151	0.2308	0.7500	**0.7949**	0.7885
201	0.2244	0.6731	0.6987	0.7308
251	0.1538	0.6410	0.6859	0.7308
301	0.1346	0.5705	0.6603	0.7436
312	0.1090	0.5449	0.6667	0.7500

Table 2. Knnclassify uses 3 neighbours

2.1.2 Ranks of scatter matrices

- Between class scatter
  Between-class scatter has a rank of C-1, where C is the number of classes. If we calculate the Hermitian of a matrix of rank C then the result will have rank C. However, since we subtract the mean of the class centres from each class centre, we lose a rank, so the matrix we take the Hermitian of will have rank C-1 and hence the result will also have rank C-1. Since C is 52 in our case, the rank of the between-class scatter is 51.

- Within class scatter
  This is a slightly more complicated case than the previous one. Let us denote N1 the number of samples belonging to class one. When calculating within class scatter for class Y, we subtract the mean of the samples belonging to class Y from each sample and then calculate the Hermitian. For the reasons outlined in sub question a, the result will have rank Ny-1. Since

the final within class scatter matrix is the sum of all such class scatters, the final rank will be N1-1+N2-1+N3-1+NC-1 = N-C (N denotes the total number of samples), which is 364-52 = 312 in our case.

2.1.3 Confusion matrix, example success and failure cases

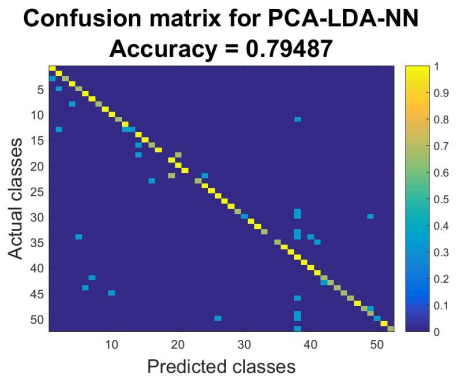

Figure 9. Confusion matrix for PCA-LDA-NN classification using the best parameters

For the best prediction case in 1), we made the confusion matrix (see on Figure 9). similarly as before. Complete failure happens for class 18, 22 and 34 as can be seen on the graph. Two test samples of class 18 were classified to be 20 and two test samples of 22 were classified to be 19. The most successful cases are the indices corresponding to the yellow rectangles on the diagonal.

2.2. PCA-LDA Ensemble

General: The number of neighbours used in the nearest neighbour algorithm was chosen to be 3 and M_{PCA} was chosen to be 151, both because of the result of the previous PCA-LDA case. We also have data for M_{PCA} = 312 (maximum), however the results are worse (see Appendix). M_{LDA} was chosen to be C-1 (51), which is the maximum value. This makes sense in the M_{PCA} = 312 case, because when the number of features after PCA is quite high (>200), the maximum M_{LDA} seems to perform the best in the PCA-LDA case (see Table 2, rows 5-8). In case of $M_{PCA} = 151$, we tried $M_{LDA} = 43$, but we achieved inferior result:

M_{LDA}	43	51
Accuracy	80.13%	82.69%
Avg. accuracy of models	43.10%	43.88%

Table 3. M_{PCA} = 151, no of base models = 100, M0 = 49

2.2.1

Randomization in the feature space was done using the same method described in X. Wang and X. Tang *et al.* [2]. The common eigenvector used in all the models (M0) was chosen to be 49 (corresponds to 88% variance retained). We chose M0 by plotting the eigenvalues and inspecting the differences between consecutive eigenvalues. The last relative big drop in eigenvalues (see Appendix) was found after the 49th eigenvalue. Before that, there was a relative big drop after the 33rd eigenvalue, and we also tested M0 = 100 but the results were inferior. The results are summarized in Table 4.

M0	33	49	100
Accuracy	80.77%	82.69%	80.77%
Avg. accuracy of models	37.99%	43.88%	52.53%

Table 4. M_{PCA} = 151, M_{LDA} = 51, no of base models = 100

Note that the result makes perfect sense: as we increase M0, the randomness decreases and each model has access to more important eigenfaces, so the average accuracy of the models increases. However, the correlation between the models also increases due to reduction in the randomness, so we benefit from the advantages of the committee machine less. From the remaining eigenvectors with nonzero eigenvalues (N-1-M0 = 314), we chose M1 (= M_{PCA}-M0) = 102 eigenvectors randomly for the ensemble models.

2.2.2

Randomization on data samples were obtained via bagging. This was implemented in Matlab in a rather simple way with two lines of code using the rand function picking a value between 0 and 1 from a uniform distribution. We multiply rand by 7 and used the ceil function that rounds the numbers to the next integer. This way, we randomly picked 7 integers, each with the same probability at each draw. The integers were used to as indices for the 7 training samples we had for each class.

2.2.3

Number of base models was varied between 10 and 200 and the average of the model accuracies is compared to the accuracy yielded by majority voting in the Table 5. As the theory predicts, as we increase the number of base models, the accuracy tends to increase.

No of base models	10	30	50
Accuracy	70.51%	81.41%	81.41%
Avg. accuracy of models	43.85%	44.49%	44.05%

No of base models	100	200
Accuracy	82.69%	**83.33**%
Avg. accuracy of models	43.88%	43.12%

Table 5. Effect of changing the number of base models

One might think that the randomness parameter is M1, since it controls how many of the used eigenvectors are chosen randomly in the models. However, it is also important how it compares to the pool of eigenvectors we can choose randomly from (N-1-M0). For example, it is easy to think that increasing M1 will reduce the correlation of the models, however, if M0 = 0, M1 = N-1, then even though the all eigenvectors are chosen randomly, each model will end up containing the same eigenfaces. So the model correlations will be 1. Hence we propose that the randomness parameter here is really the number of ways we can choose the random eigenfaces (M1) from the pool, because the more ways we can do this, the more likely our models will end up being more different. This is mathematically the following expression:

$$\binom{N-1-M0}{M_{PCA}-M0} \tag{1}$$

Note that had we decided not to follow the method in X. Wang and X. Tang *et al*. [2], other randomness parameters could have been defined. For example, rather than saying that the remaining M1 dimensions are randomly selected from the other N-1-M0 eigenfaces, we could have decreased this quantity. This would have resulted in less randomness (and more model correlation), since the remaining M1 dimensions can be chosen from a smaller pool of eigenfaces.

2.2.4

The average accuracies of the ensemble models are included in the Table 5. It can be indeed seen that the error of the committee machine is much smaller than the average error of the individual models, as expected from theory.

2.2.5

In case of nearest neighbour classification there is not a clear notion of posterior probability. Matlab defines it as the number of neighbours belonging to the same class divided by all the neighbours used in the classification. If we decide to adopt this definition, we can use sum and product fusion rules. Otherwise, we can just simply use majority voting and take the mode of the predicted classes from the base models.

2.2.6

The confusion matrix is produced for the case with the best recognition accuracy corresponding to the following parameters: M0 $= 49$, $M_{PCA} = 151$, $M_{LDA} = 51$ and number of base models = 200. As can be seen on the graph, complete failure never occurred for any class. However, class 50 was twice misclassified to class 38.

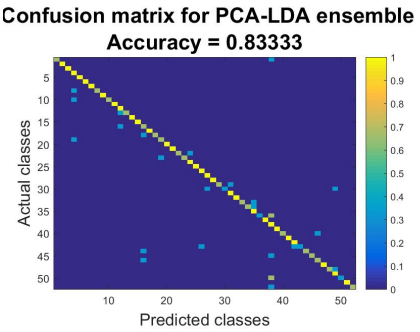

Figure 10. Confusion matrix for combined ensemble models of PCA-LDA-NN classification

References

[1] E. by: Nikunj C. Oza, R. Polikar, J. Kittler, and F. Roli. Multiple classifier systems. *6th International Workshop, MCS*, page 46, 2005.

[2] X. Wang and X. Tang. Random sampling for subspace face recognition. *IJCV*, 70(1):91–104, 2006.

Appendix

PCA-LDA

M_{PCA}/M_{LDA}	3	23	43	51
51	0.2949	0.7115	0.6218	0.5962
101	0.2949	0.7756	0.7500	0.7372
151	0.2372	0.7692	0.7821	0.7756
201	0.2308	0.6603	0.7051	0.7308
251	0.1667	0.6218	0.6923	0.7308
301	0.1282	0.5705	0.6731	0.7308
312	0.1090	0.5449	0.6667	0.7372

Table 6. Knnclassify using only 1 neighbour

M_{PCA}/M_{LDA}	3	23	43	51
51	0.2949	0.7115	0.6218	0.5962
101	0.2949	0.7756	0.7500	0.7372
151	0.2372	0.7692	0.7821	0.7756
201	0.2308	0.6603	0.7051	0.7308
251	0.1667	0.6218	0.6923	0.7308
301	0.1282	0.5705	0.6731	0.7308
312	0.1090	0.5449	0.6667	0.7372

Table 7. Knnclassify using 2 neighbours

M_{PCA}/M_{LDA}	3	23	43	51
51	0.3397	0.6667	0.5897	0.5641
101	0.2756	0.7628	0.7372	0.6859
151	0.2628	0.7436	0.7756	0.7821
201	0.2115	0.6603	0.6987	0.7372
251	0.1538	0.6282	0.6859	0.7308
301	0.1346	0.5641	0.6603	0.7308
312	0.1090	0.5513	0.6667	0.7500

Table 8. Knnclassify using 4 neighbours

M_{PCA}/M_{LDA}	3	23	43	51
51	0.3269	0.6410	0.5705	0.5385
101	0.2885	0.7628	0.7372	0.6795
151	0.2821	0.7115	0.7692	0.7564
201	0.2244	0.6603	0.7051	0.7308
251	0.1603	0.6538	0.6987	0.7244
301	0.1410	0.5641	0.6603	0.7372
312	0.1090	0.5577	0.6795	0.7628

Table 9. Knnclassify using 5 neighbours

2.2.

M_{PCA} = 312 accuracies

No of base models	10	30	50
Accuracy	66.67%	77.56%	79.49%
Avg. accuracy of models	36.99%	36.20%	34.92%

No of base models	100	200
Accuracy	82.05%	80.77%
Avg. accuracy of models	43.21%	33.16%

Table 10. M_{PCA} = 312, M_{LDA} = 51, M0=49

2.2.1.

Eigenvalue plot and choosing M0 via inspecting difference between consecutive eigenvalues

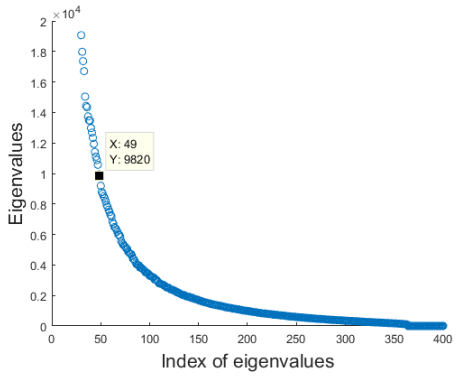

Figure 11. Choosing M0 by looking at the differences between consecutive eigenvalues

Q3

The trade-off between PCA and LDA can be expressed in the function we want to maximize as[1]:

$$W_{opt} = \arg\max_{W} \frac{|W^T[(1-\alpha)S_B + \alpha S_I]W|}{|W^T[(1-\alpha)S_W + \alpha I]W|} \quad (2)$$

If we imagine a 2D case (as we did in the lecture notes), where W has dimension 2*1, we can get rid of the determinant by realizing that both the numerator and the denominator is a scalar. Maximizing the nominator, while keeping the denominator constant (K) yields the following Lagrangian function:

$$W^T[(1-\alpha)S_B+\alpha S_I]W-\lambda[K-W^T[(1-\alpha)S_W+\alpha I]W] \quad (3)$$

Maximizing the Lagrangian function (by taking the derivative and using the necessary condition) yields:

$$2[(1-\alpha)S_B + \alpha S_I - \lambda(1-\alpha)S_W - \lambda\alpha I]W = 0 \quad (4)$$

Rearranging yields:

$$[(1-\alpha)S_B + \alpha S_I]W = \lambda[(1-\alpha)S_W + \alpha I]W \quad (5)$$

We realize that this is the generalized eigenvalue problem. The solution is

$$[(1-\alpha)S_W + \alpha I]^{-1}[(1-\alpha)S_B + \alpha S_I]W = \lambda W \quad (6)$$

So W is an eigenvector of $[(1-\alpha)S_W + \alpha I]^{-1}[(1-\alpha)S_B + \alpha S_I]$ and we choose it such that it corresponds to the higher eigenvalue. Notice then, that for $\alpha = 1$ we have PCA and for $\alpha = 0$ we have LDA.