Pattern Recognition - Coursework 1

Koral Hassan, Mohika Gupta
Imperial College London
Kensington, London SW7 2AZ
kbh15@ic.ac.uk, mg5215@ic.ac.uk

Abstract

Several approaches to the problem of facial recognition are investigated. Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are used for dimensionality reduction, and methods such as Nearest Neighbour (NN) are used for classification. Our research on the methods to maximize recognition accuracy are described in detail.

1. The Dataset

We have been provided with a dataset of $N = 520$ images that correspond to $L = 52$ faces. There are 10 images of each face respectively. All images are centred and of the same size, so we do not need to normalize (e.g. using eye locations) for scale, orientation, translation etc. The images have height $H = 56$ and width $W = 46$. Figure 1 shows one of the images in the dataset as an example. Each image $I_n \in \mathbb{R}^{H \times W}$ is represented as a vector $x_n \in \mathbb{R}^{D \times 1}$ where $D = 2576$.

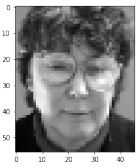

Figure 1. Examplar image of a face from dataset.

1.1. Train Test Split

The dataset is too small for cross-validation. It is also too small to be worth setting aside a validation set. In this case, we will only split the data into a training set of size N_{train} and a testing set of size N_{test}. We will set aside 30% of our data for testing, which is a reasonable partition and an industry standard. From here on wards, we will refer to these subsets as $N_{train} = (1 - 0.3) \times 520 = 364$ and $N_{test} = 0.3 \times 520 = 156$, containing the images used for training and testing respectively.

We split the data in a stratified fashion to obtain sets that are more representative, i.e the training set comprised of 70% of the total number of images in each class.

2. Eigenfaces

Each image in our dataset is represented as a point in a D-dimension feature space, where D is very large. In this case, D = 2576 and each feature contains information for one pixel of the image. However, applying facial recognition algorithms on such a high dimensional feature space is computationally inefficient as many of these features measure related properties and are therefore redundant. For more efficient facial recognition, we will project the data onto a new subspace of dimension M where $M << D$, in order to maximize the projected data variance. This is achieved by extracting the most important features and using these in our classification algorithms. This will require compromising on reconstruction accuracy as we will lose the least significant details of the images.

2.1. Procedure:

PCA defines the subspace spanned by the eigenvectors u_i of the covariance matrix S corresponding to the M largest eigenvalues to be the subspace with the largest data variance.

First we subtract the mean face $\bar{x}$ from each image in the training dataset to normalize our training data:

$$\phi_n = x_n - \bar{x}$$

where

$$\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n$$

The difference between one of the faces and the normalised image is visualized in Figure 2 as an example.

We can now compile our matrix A where

$$A = [\phi_1, \phi_2, ..., \phi_{N_{train}}] \in \mathbb{R}^{D \times N_{train}}$$

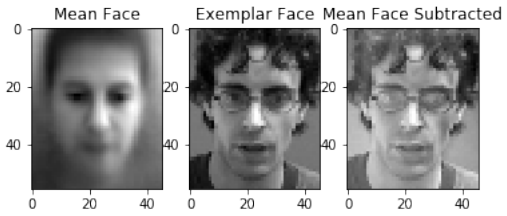

Figure 2. Difference between x_1 and , ϕ_1.

We know that

$$rank_A \leq min(D, N) = N_{train} = 364$$

since $A \in R^{D \times N_{train}}$. We expect

$$rank_A \leq N_{train} - 1 = 363$$

due to the mean face being subtracted. Due to the identity

$$rank_A = rank_{AA_T} = rank_S$$

we expect S to have at most 363 non-zero eigenvalues out of 2576.

2.2. Naive PCA

In this application of PCA, the eigenvectors of the covariance matrix $S = \frac{1}{N}AA^T$ are calculated directly. Since S is a positive-definite symmetric matrix, we expect its eigenvalues to be real and positive. Furthermore, we expect S to only have $rank_S = 363$ non-zero eigenvalues. We instead found that some of the eigenvalues were complex and negative. This is due to limitations in the software, as the complex and negative values were in the order of 10^-13 and below they can be safely approximated to zero. We can reassure ourselves of this by calculating that $rank_S = 363$ as expected.

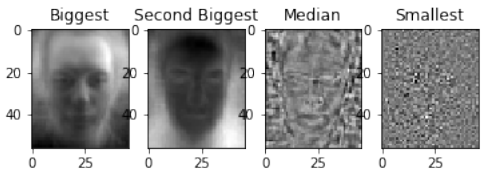

Figure 3. Eigenfaces.

Figure 3 illustrates some of the eigenvectors as eigenfaces.

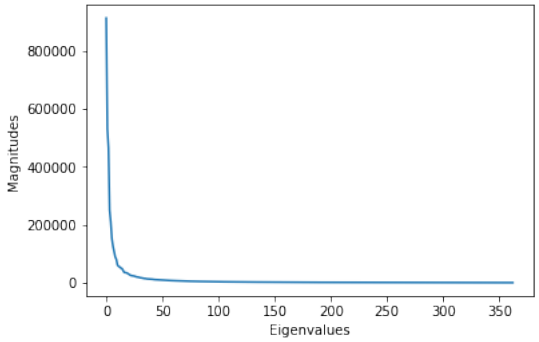

Figure 4. Eigenvalues in order of decreasing magnitude.

Figure 4 shows the exponential decrease in the magnitude of the eigenvalues. In fact, there are only 50 out of 363 non-zero eigenvalues that are at least 1% the magnitude of biggest one. Note how the eigenfaces corresponding to eigenvalues of lower magnitude contain a significantly lower amount of face information, and when the eigenvalues are zero (as shown in the smallest image), the eigenfaces contain no information relevant to a face at all, and therefore can be discarded. This leaves a maximum of 363 eigenfaces to be used for recognition.

2.3. Reconstruction

The M eigenvectors corresponding to the M largest eigenvalues will be our principal components and the basis of the subspace we will project our images onto in order to maximize variance.

Table 1 demonstrates how many principal components are needed for a given reconstruction accuracy, which is also visualized with the decreasing levels of detail in Figure 6.

Reconstruction Accuracy	Number of Principal Components, M
80%	26
85%	39
90%	63
95%	116
99%	245

Table 1. Reconstruction accuracy thresholds.

It is observable in Figure 5 that after a point, a large increase in the number of principal components used does not lead to a large increase in reconstruction accuracy. This is expected as the eigenvectors corresponding to eigenvalues of lower magnitude only contain low level information about the faces, such as fine details and shadows, as shown in Figure 6.

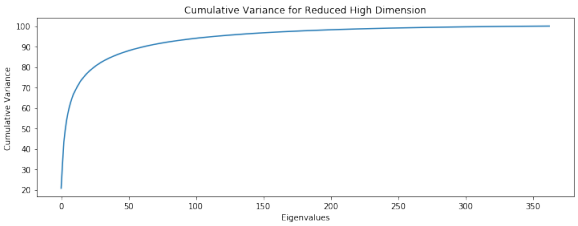

Figure 5. Cumulative variance of M eigenvalues.

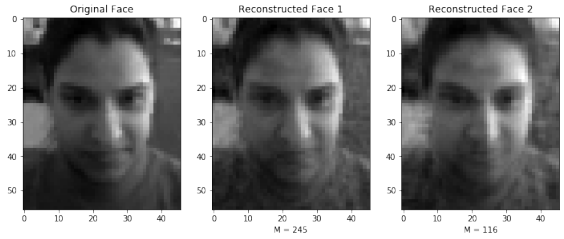

Figure 6. Reconstructed faces with $M = 245$ and $M = 116$

2.4. Efficient PCA

Now let us use the low-dimensional computation of eigenspace instead. In this implementation we calculate the eigenvectors u_i of $\frac{1}{N}A^TA$, which is a $N_{train} \times N_{train}$ as opposed to the $D \times D$ covariance matrix. The eigenvectors v_i of $\frac{1}{N}A^TA \in \mathbb{R}^{N \times N}$ are related to the N biggest eigenvectors u_i of S such that $Av_i = u_i$. This is advantageous because calculating the eigenvectors of $S \in \mathbb{R}^{D \times D}$ is computationally expensive, both in terms of memory and computation time. The computation times of the naive and efficient PCA are compared in Table 2.

Computation Method	Performance $(seconds)$
Direct	61.97
Low Dimensional	0.88

Table 2. Comparison of direct and low dimensional computations.

Although the direct calculation of the eigenvalues and eigenvectors of S is more straightforward than the low dimension computation, clearly it is significantly more efficient (70.4 times faster) than the naive implementation.

Note that despite the calculation methods of the eigenvalues of S being different, the resulting non-zero eigenvalues for both implementations are almost identical with negligible difference (in the order of 10^{-10} and lower.

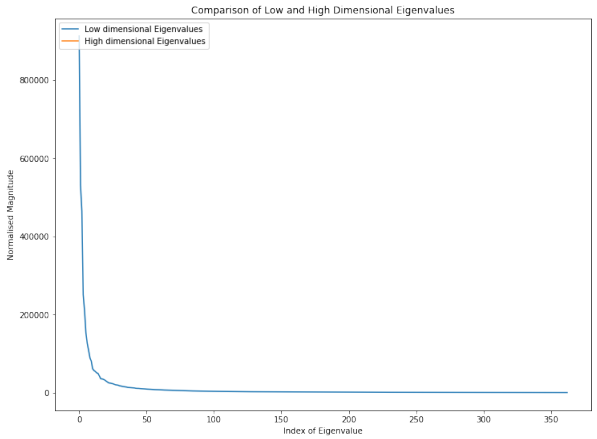

Figure 7. Comparison of Eigenvalues computed using Naive and Efficient PCA

2.5. Classification Methods

The use of two different recognition algorithms, Nearest Neighbour (NN) and an alternative method which minimized reconstruction error were explored. NN classifies images by comparing the projection of the test image onto the principal space to the projections of all the training data and selecting the class of the image with the minimum error:

$$e = min_n \parallel \omega_n - \omega \parallel, n = 1, ..., N_{train}$$

In this implementation, the principal space is calculated over all training data.

The alternative method used projects the test image onto the principal subspace computed per class. The image is then reconstructed based on the principal components for each class and the class of the image which minimizes the reconstruction error is assigned. The reconstruction error is defined as : $arg_i min \parallel x - \tilde{x} \parallel$. Comparisons of the classification accuracy and computation time for both methods are shown in Table **??** and Table 4.

No. of Principal Components, M	39	63	116	200	245	363
NN	54.499	58.97	58.33	58.97	59.6	60.26
Alternative Method	73.07	73.07	73.07	73.07	73.07	73.04

Table 3. Accuracy(%) for NN and Alternative Method.

No. of Principal Components, M	39	63	116	200	245	363
NN	0.039	0.041	0.079	0.0869	0.110	0.177
Alternative Method	1.83	2.170	2.530	2.634	2.83	3.16

Table 4. Computation Time($seconds$) for NN and Alternative Method.

This can also be visualized in the confusion matrices (Figure 8) for both methods for the case $M = 200$. Note how the confusion matrix for NN is more scattered than the Alternative method, i.e. NN makes more incorrect predictions.

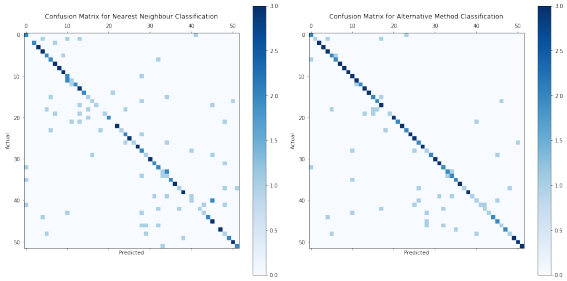

Figure 8. Confusion Matrices for Alternative and NN Classification

An example of a successfully recognized test image for both classification methods is shown in Figure 9.

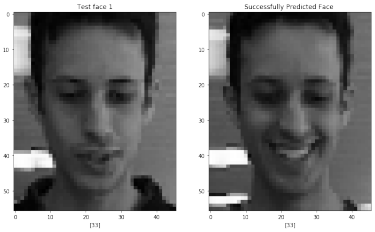

Figure 9. Successful Classification of an image in class 33

Figure 10 gives an example of a face that was incorrectly classified by NN but correctly classified by the Alternative Method.

Figure 10. Incorrect Classification by NN

This proves the superiority of the Alternative Method over NN classification. Since NN classification does not take into account different classes when computing the principal space, so it is easy to incorrectly assign people with similar features to the same class. In comparison, the Alternative Method calculates the principal space over each class, allowing the differentiation between similar images from different classes as shown in Figure 10.

3. PCA-LDA

The variance among faces in a dataset may come from distortions such as shadowing, facial expression, pose etc. In some cases, these variations may be larger than the variances between standard faces. For this reason, we investigate the use of Linear Discriminant Analysis (LDA) to improve our face recognition accuracy.

3.1. LDA

LDA aims to find the projection W_{opt} that optimally separates the data of different classes by minimizing the Within- Class Scatter matrices S_W and maximizing the Between Class Scatter matrices S_B where:

$$S_w = \sum_{i=1}^{C} \sum_{x \in C_i} (x - m_i)(x - m_i)^T$$

and

$$S_B = \sum_{i=1}^{C} (m_i - m)(m_i - m)^T$$

Where m_i = the mean image of class $C_i$and m is the average face over the whole dataset. In comparison, the optimal projection generated from PCA maximizes both S_B and S_W.

3.2. Procedure

LDA generates the optimal projection space for classification by taking the eigenvectors corresponding to the M_{lda} largest eigenvalues of the following generalized eigenvector problem:

$$S_W^{-1} S_B w = \lambda w$$

However, direct implementation of LDA is not possible in our dataset due to $N_{train} << D$. Theoretically, $rank_{S_B} = C - 1$ and $rank_{S_W} = N_{train} - C$. However, since $S_w, S_B \in \mathbb{R}^{D \times D}$, these matrices are not full rank, hence solutions of the generalized eigenvector problem cannot be computed. To enable this computation we must project our training dataset from a D-dimensional space to an M_{pca} dimensional space using PCA where $M_{pca} \leq N_{train} - C$. This makes S_W full-rank, allowing for the use of LDA to further reduce the dimension to $M_{lda} \leq C - 1$ by taking the eigenvectors corresponding to the M_{lda} largest eigenvectors w_i of:

$$(W_{pca}^T S_W W_{pca})^{-1} (W_{pca}^T S_B W_{pca}) w = \lambda w$$

Finally, the optimal projection space (fisherface) is defined as:

$$W_{opt}^T = W_{pca}^T W_{lda}^T$$

We then use NN classification to implement face recognition. We expect the accuracy from the Fisherface implementation to be higher than PCA alone as Fisherfaces accounts for the classes for the images and will therefore be able to recognize faces with higher accuracy.

3.3. Results

Upon implementing PCA-LDA we found $rank_{S_W} = 312$ and $rank_{S_B} = 51$ as expected. Our results from the implementation of PCA-LDA for varied values of M_{pca} and M_{lda} are shown in Table 5.

M_{LDA} vs M_{PCA}	311	200	1006	10
51	52.6%	78.2%	53.2%	54.4%
25	50.8%	68.5%	52.7%	54.0
10	49.4%	57.4%	51.1%	53.9

Table 5. PCA-LDA accuracy for different hyper-parameter values.

Notice how the accuracy of PCA-LDA decreases for very high and very low values of M_{pca}, this is as expected. For high values of M_{pca}, there has been little removal or redundant features, making the model susceptible to error due to outliers, while low values of M_{pca} remove too much information from the feature space, making it difficult to distinguish classes of faces from one another.

4. PCA Ensemble

Ensemble learning aims to provide a solution to over-fitting. Over-fitting a data set in when a learned model is overly complex and fits too closely to the training dataset. When applied to a test dataset, this model then behaves poorly as it has been developed to perfectly fit only the dataset it has been trained on, we say the model has poor generalization.

A proposed solution to over-fitting is ensemble learning. Ensemble learning takes multiple overly complex models trained on the same data (with perturbations) and combines them to give output a combined model. Since the perturbations between models is random, individual model predictions are de-correlated, resulting in a combined model with improved generalization.

4.1. Bagging

We were able to generate randomness between models by randomly sampling the training set (i.e. bagging). We applied bagging with replacement on a number of subsets T_j with each subset containing $nt = (\mid T_j \mid = \rho)$ samples. In order to keep the sampled subsets consistent with the original use of PCA we opted to sample an equal amount of each class for each subset.

After generating our subsets T_j we applied PCA and NN classification to each bag. We then applied majority voting to the results to generate our final class prediction. Majority voting takes the predicted classes generated from each model and assigns the class with the highest 'number of votes' to each test image.

Theory dictates that the expected error of the final class assignment (i.e. the error of the committee machine) is lower than that of the average error across individual models. This because, as explained previously, each individual model is over fitting the data in each subset and therefore has poor generalization and accuracy when given test images.

4.2. Results

The results generated from varying the number of subsets (or base models) T_j and the number of samples in each bag ρ are shown in Figure 11. It is clear that after an initial jump in the majority voting accuracy, we start to see increasingly diminishing returns from adding more bags.

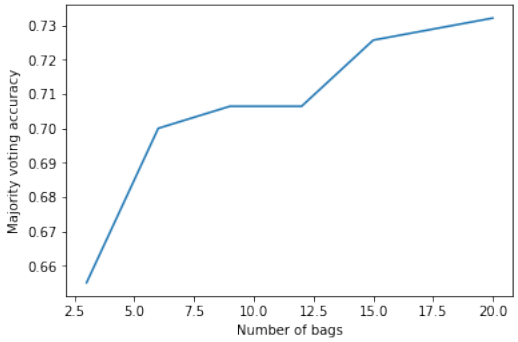

Figure 11. Committee Machine Accuracy for varied T

Number of images per bag vs Number of bags	3	6	9	12	15	20
156	57.2%	58.4%	58.5%	59.0%	59.4%	59.1%
260	64.7%	66.6%	65.8%	66.4%	67.0%	66.2%
364	65.9%	70.0%	69.2%	69.6%	69.4%	71.0%

Table 6. Average accuracies of individual models.

Number of images per bag vs Number of bags	3	6	9	12	15	20
156	52.7%	58.5%	52.7%	59.7%	59.7%	56.5%
260	65.5%	67.4%	64.8%	69.8%	71.9%	68.1%
364	65.6%	70.0%	70.7%	70.6%	72.6%	73.2%

Table 7. Committee machine accuracies.