EE4-68: Pattern Recognition

Jorge Sanmiguel - CID: 01064565   Ricardo Vera - CID: 01061547
Electrical and Electronic Engineering Department
Imperial College London

1. Eigenfaces

1.1. Data

520 images of size 56 by 46 pixels (as a vector of $n_{px} = 2576$ values) containing faces (52 identities, with 10 images for each) are provided and are to be split into training and test sets. Randomly partitioning the amount of images in a 70:30 ratio (train:test) provides a good trade-off between having enough samples to train the algorithm (i.e. $n_{train} = 364$ vectors), and being able to obtain a representative performance metric of the algorithm (test error with $n_{test} = 156$ vectors). The random seed was set to a fixed number to get reproducible results.

To perform face recognition, the model must have seen each class during training. The data was hence stratified to guarantee the presence of all classes in both sets, each with the same number of occurrences.

The average face for the entire/training set are shown in Figure 1. Both images are indistinguishable with the naked eye, as the training set is a statistically faithful representation of the entire dataset.

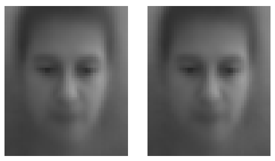


Figure 1: Average face for the entire set (left), and for the training set (right).

1.2. High Dimensional PCA

Principal component analysis (PCA) is the computation of the eigenvalues and eigenvectors of the covariance matrix, $S = AA^T$ ($n_{px} \times n_{px}$ (2576×2576)), for the training data A ($n_{px} \times n_{train}$ (2576×364), each column being a face image.). The set of eigenvectors $u = n_{px} \times n_{train}$ (organised in columns) is a basis of the training space (real valued), from which all samples can be fully reconstructed, and the eigenvalues represent the weight of each eigenvector. The maximum eigenvalue corresponds to the eigenvector pointing to the direction of maximum data variance.

In Figure 2 it can be seen a large drop in the eigenvalue magnitude ($< 10^{-10}$) starting from the 364th eigenvalue (including). This shows there are 363 non-zero eigenvalues, corresponding to the rank of the covariance matrix S (i.e. the number of faces in the training set - 1, due to average face subtraction. Without face subtraction, the first eigenvector would correspond to the mean face.).

Choosing the number of eigenvectors to be used for face reconstruction is a trade off between low reconstruction error and low complexity of the algorithm. From Figure 3 it can be seen that eigenvectors with smaller eigenvalues contribute high frequency information to a reconstructed face, whereas those with larger eigenvalues contribute lower frequency information. From a qualitatively point of view, the first most important eigenvectors contain most of the information of the image, and the rest add finer details to the image. It is sensible to assume, that not having all of the high frequency information of an image in face recognition will affect little its performance.

As an initial guess, using around 150 eigenvectors should yield good face reconstruction, and should allow accurate face recognition. Higher eigenvectors, like $M = 300$ (see Figure 3), look similar to white noise. These eigenvectors might be too representative of the training data, and may lead to overfitting of test data if used for face recognition.

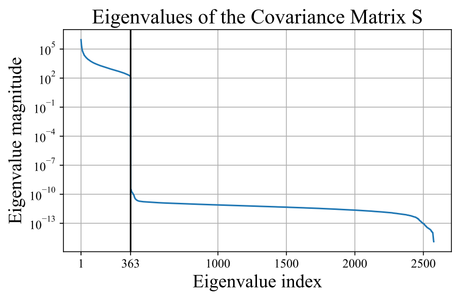


Figure 2: Eigenvalue magnitude log plot.

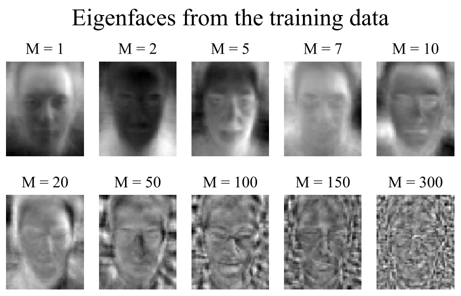


Figure 3: Eigenfaces obtained from the training data. These will be used for face reconstruction and classification

1.3. Low Dimensional PCA

The low complexity PCA method means calculating the covariance matrix as $S = A^T A$, where A is the training samples matrix, hence reducing the dimension of the covariance matrix to $n_{train} \times n_{train} (n = 364)$. All eigenvalues from the covariance matrix are non-zero except for the one introduced by the sample mean subtraction. This is a more efficient procedure as the eigenvalue decomposition is much quicker for the smaller matrix, since the number of samples is much smaller than the feature space of the samples.

The eigenvalues obtained are the same as those for high dimensional PCA, only with $n_{px} - n_{train}$ fewer zero eigenvalues. The eigenvectors are real valued, but not the same; there are only n_{train} of them and have length n_{train} too. Let $v = n_{train} \times n_{train}$ denote the matrix of eigenvectors (organised as columns). In order to map them to those obtained from high dimensional PCA, $u = Av$ has to be performed, where $u = n_{px} \times n_{train}$ are the eigenvectors from the high dimensional PCA. Just like with the eigenvalues, $n_{px} - n_{train}$ fewer eigenvectors will be obtained, but since those have corresponding zero eigenvalues, it is irrelevant.

1.4. High vs. Low Dimensional PCA

Low dimensional PCA is a more computational efficient procedure when performing eigenvalue decomposition due to the lower dimensions of the covariance matrix ($n_{px} \times n_{px}$ vs. $n_{train} \times n_{train}$), which in our case means it having 98% less values. Low dimensional PCA however requires the additional step $u = Av$ to map the eigenvectors obtained to those required for face reconstruction/recognition, hence needing to perform an extra step to obtain the eigenfaces.

Low dimensional PCA is clearly more efficient than high dimensional PCA in terms of efficiency with an average time of 0.13 vs 5.04 seconds for operation, hence it will be used in the rest of the exercises. These results are further discussed in Section 2.2.3 (Table 2).

It is also worth noting, that the low dimensional implementation here discussed will only be more efficient than high dimensional PCA whilst the number of samples in the training set is smaller than the feature space. Otherwise, the efficiencies would reverse.

2. Application of Eigenfaces

2.1. Face reconstruction

PCA can be used to construct an eigenspace of a set of faces and build a model able to reconstruct faces from the model's eigenvectors. Combining this with a classification algorithm such as k-Nearest Neighbours (kNN) provides a means for a face recognition model by comparing the projection of input faces on the learnt facespace, with the learnt projected faces (from the train dataset). When a new face is passed to the algorithm, kNN will find the training image that most closely matches the input one (ideally a different image of the same person). The amount of eigenvectors used for face recognition will determine the accuracy of the reconstruction and so the accuracy of face recognition.

Images from the training dataset can be reconstructed perfectly when using all the eigenvectors of the subspace, which is expected since the set of eigenvectors spans the set of training data, and hence each training face can be expressed as a linear combination of the eigenvectors (for an example of this, see Appendix 6.1). Using 150 eigenvectors, the reconstructed image is clearly distinguishable, with much of the original image details present. The remaining eigenvectors then add finer details to generate a perfectly reconstructed image. This behaviour matches the pattern observed in the eigenfaces of the face subspace: the first eigenvectors contain most of the information, and the latter ones contain the high-frequency details.

Test images cannot be perfectly reconstructed once projected into the face subspace, see Figure 4. Nevertheless, the reconstructed image clearly conserves the identity of the person. Eigenvectors larger than 150 contribute little to the clarity of the reconstructed image. Moreover using all eigenvectors can result in a perceived increase of noise in some regions of the reconstructed image. This can be seen in Figure 4, for M = 364, which has more noise present in the background of the image than for M = 150.

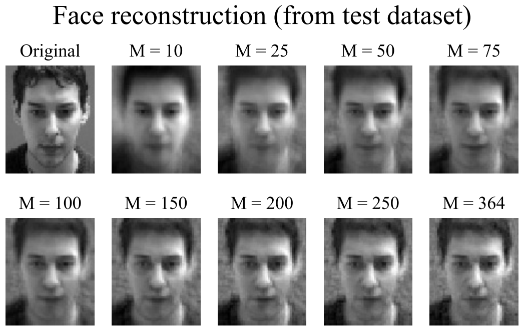


Figure 4: Face reconstruction of a test image with varying number of eigenvectors (M) used.

Since little detail is added to the reconstructed images when using more than 150 eigenvectors, both for train and test images, it was decided this threshold should be further studied.

A metric that helps determine a sensible number of eigenvectors to be used is the *Proportion of variance explained*, Equation 1 [1]. It is a measure of the variance that is used from the total variance by using the first M eigenvectors.

$$\frac{\sum_{i=1}^{M} \lambda_i}{\sum_{j=1}^{n_{train}} \lambda_j} \tag{1}$$

For $M = 150$, this comes out to be 96.6%. Since determining a threshold for M is arbitrary, it was decided to use M = 150 as it captures most of the model variance, and is a nice round number.

The reconstruction error can also be measured, rather than simply relying on perceived quality loss. Figure 5 shows the mean squared error (MSE) when reconstructing train and test images for different numbers of PCA bases used. The trend is the same for the train and test sets: at first there is a steep decrease of the MSE as the number of bases increases. The error curve then gradually flattens out and becomes relatively constant. As expected, the MSE for train samples reaches 0, and the MSE for test samples doesn't. For further comments on Figure 5 see Appendix 6.2.

Another metric that can be used to determine the optimal number of eigenvectors for face reconstruction/recognition is to use a model estimator, such as the Akaike information criterion (AIC)[2], Equation 2, where N is the maximum order value. As shown in Figure 5, the optimum number of eigenvectors to use as estimated by this model estimator is 87.

$$AIC(n) = \ln E_{MSE} + \frac{2n}{N} \tag{2}$$

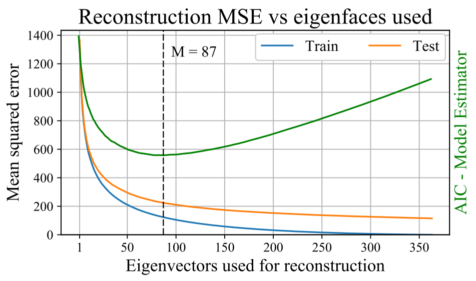


Figure 5: Average reconstruction mean squared error for train and test data, against the number of eigenfaces used in the reconstruction. AIC complexity model curve superimposed.

2.2. Face Recognition

2.2.1 Nearest Neighbours

The first classification algorithm explored is Nearest Neighbours classification. For a detailed description of the algorithm see Appendix 6.3

As seen in Table 1, the best performance obtained by this model is 60.9% when using 150 eigenvectors and $k = 1$. This is much higher than the $\approx 2\%$ chance for random classification, but there is still a large room for improvement. As k increases, both the train and test accuracies decrease. Using 87 eigenvectors decreases slightly the test error for $k = 1$, and slightly improves for the rest values of k.

	λ_{87}		λ_{150}	
k-neighbours	Train	Test	Train	Test
1	100.0%	58.33%	100.0%	60.90%
3	72.53%	41.03%	71.15%	40.38%
5	64.29%	43.59%	64.46%	42.31%
7	56.32%	39.10%	54.95%	35.90%

Table 1: Train and test accuracies for NN classification. 87 & 150 PCA bases are used, and 'k' is number of neighbours used.

The error increasing as k increases suggests that the the training dataset faces are not closely clustered in the faces subspace by classes. Checking only the nearest neighbour yields the best accuracy, as there might be many cases in which the test image being fed into the algorithm is really similar to one of the same class in the training set, and hence the classifier correctly labels it. All whilst the other non similar train images of the same class might be far away in the face subspace.

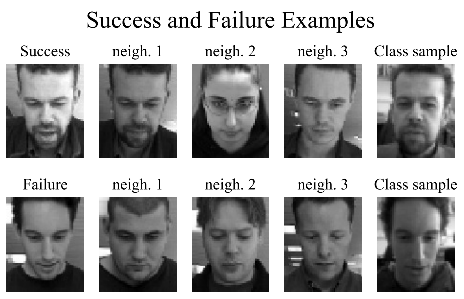


Figure 6: Success (top) and failure (bottom) examples when classifying faces from the test data set for k = 1. The three closest neighbours are shown, together with a randomly picked image of the target class in the training set.

This phenomenon can be seen in Figure 6, which shows an example of successful and unsuccessful classification of two test images. In the first case, the closest neighbour to the test image is extremely similar to the test image, whereas the two next closest neighbours are not. In the failure case, none of the neighbours are similar to the test image, but the backgrounds of the images are (ceiling lighting pattern).

2.2.2 Using Reconstruction Errors

An alternative method to using nearest neighbours for face recognition is using reconstruction errors. PCA is performed independently for the 52 classes, generating 52 different subspaces, each specific to the class. Every test image is then projected into each subspace, and the label of the subspace that generates the smallest reconstruction error is chosen. Ideally, the test data should map best to the class subspace it belongs, as it should be the only space to have the (most) correct eigenvectors needed for reconstruction.

This alternative method obtains a remarkable result of 75% test accuracy (100% train accuracy) when using 3 eigenvectors per class, a total of 156 different eigenvectors. This can be seen in Figure 8 (further discussed in section 2.2.3). The accuracies obtained by varying the number of eigenvectors used converges rapidly to this value, and remains constant when increased. Note that the exact same train and test sets as with kNN have been used.

By using the reconstruction error to classify each test image, a measure of how close the test image is to all of those forming the subspace is obtained. This measure yields better accuracies since it averages out misclassifications that result from isolated face patterns. This contrasts with the kNN method, which just finds the class of the training image that is most similar to the test image.

Figure 7 shows some examples of correct matches and failures of the algorithm. It can be seen, that it can match faces to the correct class that have different illumination conditions, facial expressions and perspective. On the other hand it still fails to correctly label test images that have matching backgrounds with other training images (first failure example), or even cases in which the face is partially occluded by an object (such as hair, second failure example).

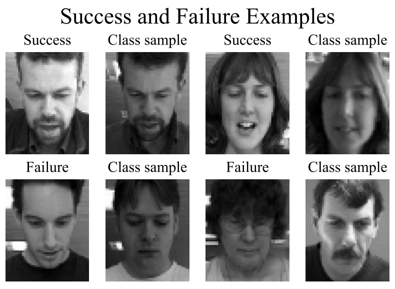


Figure 7: Success (top) and failure (bottom) examples when classifying faces from the test data set, next to samples images from the predicted class.

2.2.3 Results and Comparison

The Reconstruction Error method yields higher classification accuracies overall ($\sim$ 75%) when compared with Nearest Neighbours classification ($\sim 60\%$), as seen in Figure 8. As expected, the training accuracy is 100% no matter the number of PCA bases used, except for a slight decrease in accuracy when only using 52 PCA bases (1 per class) in the reconstruction method.

NN test classification's accuracy stabilises at around 90 PCA bases, which is spot on to the model selection estimate given by AIC. On the other hand, the reconstruction methods needs a larger number of eigenvectors to perform optimally, that is 153 eigenvectors, or 3 per class.

The reconstruction error method is however more computationally expensive, due to the eigenvalue decomposition of 52 classes and the expensive error comparison calculations.

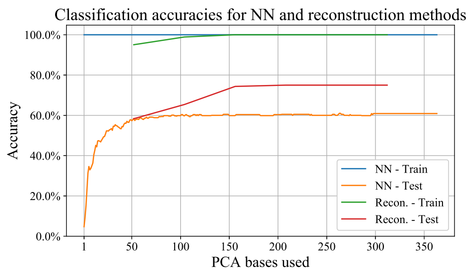
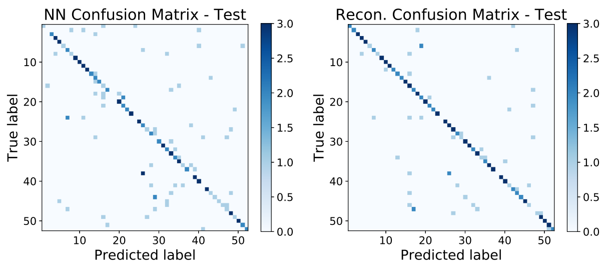


Figure 8: Train and test error for kNN (k = 1) and reconstruction error. Note that for the latter, the number of 'PCA bases used' corresponds to the total number of unique eigenvectors used (i.e. if 3 are used per class, a total of 153 unique vectors are used). Results averaged over 5 runs.



(a) kNN Confusion Mat. (b) Rec. Confusion Mat.

Figure 9: Confusion Matrices for kNN (M=150) and Reconstruction Error (M=3) models for test samples. Training confusion matrices display all correct matches.

As can be seen from Figure 10, both methods show similar struggles in classification, just with better performance by the reconstruction error method. Both seem to struggle to correctly classify labels 24 & 38, and both struggle to give confident predictions for labels 12-20 or 41-48, where most variance is shown. Clearly the reconstruction error method performs better, showing much less variance overall, and more specifically in the complicated regions mentioned above, where it does give accurate results for some of the labels.

	Best	Worst	Average
High Dimension PCA	4.92	5.12	5.04
Low Dimension PCA	0.10	0.15	0.13
Reconstruction Error	2.39	2.69	2.50

Table 2: Average performance times in seconds over 6 iterations. Times consider all the steps from first calculation to label prediction.

Table 2 shows the average time taken for the high/low dimensional PCA and reconstruction error implementation, when tested on the same computer. Low dimensional PCA takes 39 times less on average to execute than high dimensional PCA, even when it has to compute the additional mapping $u = Av$. Even though the Reconstruction Error method is quicker than high dimensional PCA, it is still far slower than low dimensional PCA. Hence a trade-off has to be considered between performance time (Table 2) and accuracy of the model (Figure 8).

3. Generative and Discriminative Subspace Learning

The cost function to maximise of the combination of PCA and LDA can be seen as the addition of the independent cost functions:

$$J_{\text{PCA}} = \frac{\mathbf{w}^T\mathbf{S}\mathbf{w}}{\mathbf{w}^T\mathbf{w}} \quad (3) \qquad J_{\text{LDA}} = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}} \quad (4)$$

Therefore, $J(w) = J_{PCA} + J_{LDA}$, which can be written as $J(\mathbf{w}) = \mathbf{w}^T\mathbf{S_B}\mathbf{w} + \mathbf{w}^T\mathbf{S}\mathbf{w}$, subject to $\mathbf{w}^T\mathbf{w} = 1$ and $\mathbf{w}^T\mathbf{S}_W\mathbf{w} = k$. Then:

$$\begin{aligned}
J(\mathbf{w}, \lambda, \beta) &= \mathbf{w}^T\mathbf{S}_B\mathbf{w} + \mathbf{w}^T\mathbf{S}\mathbf{w} + \lambda(k - \mathbf{w}^T\mathbf{S}_W\mathbf{w}) + \beta(1 - \mathbf{w}^T\mathbf{w}) \\
J(\mathbf{w}, \lambda, \beta) &= \mathbf{w}^T(\mathbf{S_B} + \mathbf{S} + \lambda\mathbf{S_W} - \beta)\mathbf{w} + \lambda k + \beta \\
J(\mathbf{w})' &= 2(\mathbf{S_B} + \mathbf{S} - \lambda\mathbf{S_W} - \beta)\mathbf{w} = 0 \\
J(\lambda)' &= k - \mathbf{w^T}\mathbf{S_W}\mathbf{w} = 0 \rightarrow \mathbf{w^T}\mathbf{S_W}\mathbf{w} = k \\
J(\beta)' &= 1 - \mathbf{w^T}\mathbf{w} = 0 \rightarrow \mathbf{w^T}\mathbf{w} = 1 \\
\mathbf{w}^T J(\mathbf{w})' &= 2\mathbf{w}^T(\mathbf{S_B} + \mathbf{S})\mathbf{w} - 2\lambda k - 2\beta = 0 \\
\mathbf{w}^T(\mathbf{S_B} + \mathbf{S})\mathbf{w} &= \lambda k + \beta
\end{aligned} \quad (5)$$

The final line can be expressed as $(\mathbf{S_B} + \mathbf{S})\mathbf{w} = (\lambda k + \beta)\mathbf{w}$ from $\mathbf{w^T}\mathbf{w} = 1$. Hence, the solution is the eigenvectors of $(\mathbf{S_B} + \mathbf{S})$.

4. LDA Ensemble for Face Recognition

4.1. PCA-LDA

Whilst PCA finds the direction of maximum variance to obtain the main components, LDA aims at finding the direction that best separates the data in classes. That is, LDA aims at maximising Equation 6, where S_W is the within-class scatter and S_B is the between-class scatter.

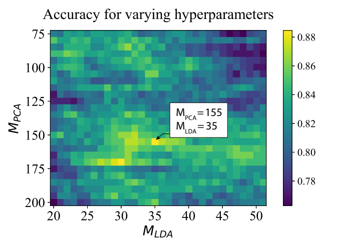
$$J(\mathbf{w}) = \frac{\mathbf{w}^T\mathbf{S}_B\mathbf{w}}{\mathbf{w}^T\mathbf{S}_W\mathbf{w}} \quad (6)$$

Hence, by maximising Equation 6 w.r.t $\mathbf{w}$, the eigenvalue decomposition of $(\mathbf{w}_{pca}^T\mathbf{S}_W\mathbf{w}_{pca})^{-1}(\mathbf{w}_{pca}^T\mathbf{S}_B\mathbf{w}_{pca})$ is effectively obtained, and thus the LDA optimum weights are achieved.

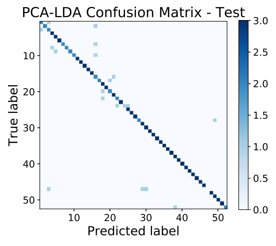
S_B is the sum of S_i for each class i for which $\text{rank}(S_i) \leq 1$ for one class, thus $\text{rank}(S_B) \leq \sum_i S_i$, which is at most $c - 1$ (where c is the number of classes, 52). The rank of S_W is at most $N - c$, and so is singular as its dimensions are far greater than the number of samples (N). In our case, the rank of the matrices matched the predicted ranks from theory, that is $\text{rank}(S_W) = 52 - 1 = 51$ and $\text{rank}(S_W) = N - c = 364 - 52 = 312$.

To find the optimum hyperparamaters for the model, a range of them values were tried, as seen in Figure 10a. Cross validation was considered at this point, but due to the reduced size of the dataset, the idea was discarded. The optimum hyperparameter values for M_{PCA} and M_{LDA} were found to be 155 & 35 respectively. Different values of k where tried for NN, but $k = 1$ consistently outperformed all other values. These optimum hyperparameters give a recognition accuracy of 88.5%. The original PCA dimension selection of

150 eigenvectors was very close to the actual optimum of $M_{PCA} = 155$.



(a) PCA-LDA accuracy for varying parameters of M_{PCA} and M_{LDA} ($k = 1$)



(b) PCA-LDA confusion matrix for $M_{PCA} = 155$, $M_{LDA} = 35$ and $k = 1$

Figure 10

As can be seen from Figure 10b, the PCA-LDA model has much better performance than the previous PCA implementations. It is worth noting, that similarly to the previous models, PCA-LDA is unable to properly identify label 47, and similarly struggles to give confident predictions for labels 16-20, which indicates that those labels are particularly difficult to classify. It is however clear that the PCA-LDA model performs at a high standard as can be seen from the little deviations in predictions in Figure 10b.

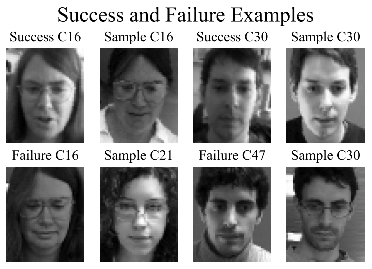


Figure 11: Success (top) and failure (bottom) examples when classifying faces from the test data set.

Figure 11 shows sample success and failures of the PCA-LDA model. It can be seen how PCA-LDA can classify some samples of a class but not others (first success/failure case in Figure 10b. In any case, all predicted classes are really similar to the test classes, showing that PCA-LDA only struggles for people that look alike.

Overall, implementing LDA and minimising the distance between samples of the same class allows kNN to classify more accurately test data that the previous methods explored, and outperforms PCA by more than 28%.

4.2. PCA-LDA Ensemble

Number of Models and Fusion Rules

Creating an ensemble of models with *bagging* or with *feature randomisation* from the same dataset allows overfitting to be avoided to a greater extent, yielding a more robust model. This is done by using multiple weak models to outperform a unique, stronger one. One of the first things to consider is the number of models to create. It is expected that the accuracy of the ensembles will stabilise after a small finite number of sets, since the dataset is relatively small, and the randomness introduced will converge to a statistical mean. A range of sets from 1 to ~50 will be considered, and the optimal number will be picked after measuring the ensemble accuracies.

After all the different models are generated, the test data is evaluated in each model and their output is then merged via a fusion function. An example would be using majority voting, which will make the most popular label upon all models the ensemble output, whilst functions such as max or min will make the ensemble output that of the model which is most/least confident on its classification.

Using *max* as a fusion function makes little sense, as each model will determine with probability 1 a label for k = 1, and hence all models are equally confident on their output. Increasing k results in some larger variance on the classification confidence by each model, but still does not give enough resolution. Multiplying the probabilities of each model together and then picking the label with then largest probability was also considered. However, for the same reason as before, the results were pretty disappointing. Each model being completely confident on most of the labels having 0 probability results in that even if one model erroneously classifies a test sample, the ensemble output will report that label having 0 probability. In the following results majority voting will be used, as it consistently yield the best results.

Bagging

To improve stability of face recognition, this methods implements *bootstrap aggregating*, also known as *bagging*. The bagging procedure generates N new datasets of size n', randomly sampled from the original dataset (with replacement), hence introducing decorrelation between the new sets. Applying PCA-LDA to the different N datasets generates N different models for the ensemble. If correctly done, the output of the ensemble will result on more accurate classifications with less overfitting. It is important not to decorrelate the datesets too much, as otherwise the ensemble will start performing poorly. Another disadvantage of bagging is its higher computational complexity required to split the dataset N times, and then performing PCA-LDA N times too.

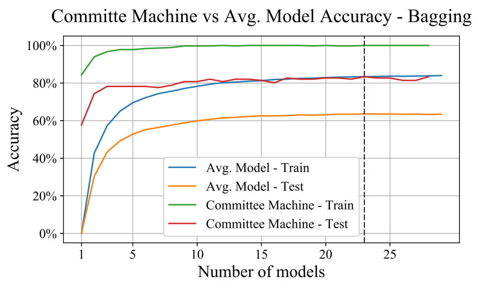


Figure 12: Bagging ensemble accuracy for varying values of setsize

The best performance of bagging is 83.3% for 23 sets (vs 63.6% average model accuracy), as seen in Figure 12. The committee accuracy is consistently higher than the average accuracy of the discrete models by about 20%. As predicted earlier on, the accuracy converges to 83.3% for higher number of sets. In this implementation, the number of samples per training set was 364 (i.e., $n' = n$). This is because if $n' = n$, then for large n, the set D_i is expected to have $(1 - 1/e)$ (63.2%) of the unique examples of D, the rest being duplicates [3].

The best accuracy using bagging (83.3%) is less than that of PCA-LDA on its own (88.5%). Since cross-validation was not performed because of the small dataset, it is likely that the result given by PCA-LDA is over optimistic, and still overfits the underlying statistical features of the data. The ensemble using bagging is then likely to overcome this, and hence its accuracy might be closer to reality.

Feature Parameter Randomisation

Feature *parameter randomisation* offers another alternative for generating different models for the ensemble, by randomising the model parameters from the same base model. In this case, the PCA eigenvectors are randomised to generate T different models. The first M_0 eigenvectors are fixed, and correspond to the first M_0 eigenvectors from the base model. Adding on to this M_1 further eigenvectors which are selected randomly from the remaining eigenvectors of the base model ($M_0 + M_1 = M_{PCA}$). This is done T different times to generate T different models.

The randomness parameter ρ represents the randomness between the different generated models. In this case $\rho = M_0$, since for M_{PCA} fixed in $M_{PCA} = M_0 + M_1$, the amount of vectors randomly selected ($M_1 = M_{PCA} - M0$) determines model correlation. For $\rho = |T| = 363$, $M_1 = 0$ giving total correlation between models and no randomisation. If $\rho = 1$, $M_1 = 362$ and so the models will vary greatly as 362 vectors are being chosen at random for each model, resulting in low model correlation.

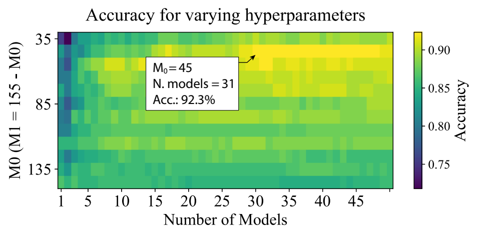


Figure 13: Feature randomisation ensemble accuracy for varying values of ρ

From Figure 13 it can be seen that the optimum value of ρ (i.e. M_0) is 45 with an ensemble consisting of 31 models, which gives a committee accuracy of 92.3%. For the same parameters, the average model accuracy is 80.1%, more than 12% lower than the committee accuracy. This effectively shows the benefit of using randomisation of models, each with 110 randomly selected eigenvectors. See Table 3 for average/committee accuracy comparison between bagging and feature randomisation.

	Model Avg.	Com. M.V	Com. Mult.
Bagging	64.6%	84.3%	16.67%
Feature Randomisation	80.1%	92.3%	37.18%

Table 3: Bagging and feature randomisation accuracies

Joint Ensemble

The final implementation is the combination of both the bagging ensemble and the feature randomisation ensemble with their respective optimal parameters previously derived ($M_{PCA} = 155$, $M_{LDA} = 35$, $N_{bag.} = 23$, $N_{feat.} = 31$, $\rho = 45$, $k = 1$), to generate a joint ensemble.

Since the number of hyperparameters for the joint ensemble is large (> 5), it is impractical to run a bruteforce algorithm for finding the optimal parameters. Hence, based on the results of the previous implementations, educated guesses were made and the joint ensemble accuracies compared. See Table 4

Modified Parameters	Model Avg.	Com. M.V	Com. Mult.
None	74.5%	88.46%	16.00%
$k = 3$	72.61%	88.46%	21.15%
$M_{LDA} = 51$	75.19%	87.12%	16.03%
$\text{Sets}_{F.Rand} = 40$ & $\rho = 55$	75.76%	90.38%	16.67%

Table 4: Accuracies for varying parameters of the base joint ensemble model. Average model, committee majority voting and multiply fusion rule methods shown.

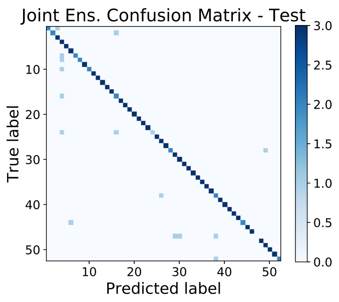


Figure 14: Ensemble confusion matrix for test data

Figure 14 shows how the joint ensemble performs similar to previous models, failing in similar areas just with a better performace.

Because of the committee machine behaviour of the ensemble model, the error achieved far outperforms the average error of the individual models. Due to the correlation of the different models $E_{com} \neq \frac{E_{av}}{T}$, however the correlation isn't significant enough and so $E_{com} \leq E_{av}$.

The joint ensemble gives higher error than the feature randomisation ensemble due to the presence of the Bagging models. These perform somehow worse than the feature randomisation and so the overall accuracy is reduced. This however may give a more robust model against overfitting, due to the greater generalisation introduced by the Bagging models.

From Table 4 it can be seen that the optimum parameters from previous methods give a very high 88.46% accuracy, however by varying ρ and the number of feature randomisation models within the ensemble a maximum accuracy of 90.38% is obtained which is slightly better than the individual PCA-LDA model by 1.98%.

5. Conclusion

This report has described the different procedures to develop a face recognition algorithm. The different steps have been assessed along the way in terms of accuracy and time complexity. PCA performs relatively mediocre in terms of recognition and is outperformed by the reconstruction error method, though it is far better in terms of complexity. Low dimensional PCA is considerably accurate in face recognition when combined with LDA, obtaining very high accuracies for a single model. Finally, by creating an ensemble of models via bagging and feature randomisation, the highest accuracy is obtained by means of fusion functions, mainly majority voting. The drawback on an ensemble versus a single model is the higher time and memory complexity needed, as multiple models have to be made, trained and their outputs combined.

References

[1] Ladislav Rampasek Wenjie Luo. "Principal Component Analysis (PCA)".

[2] H. Akaike. "A new look at the statistical model identification". eng. In: *Automatic Control, IEEE Transactions on* 19.6 (1974), pp. 716–723. ISSN: 0018-9286.

[3] Javed A. Aslam, Raluca A. Popa, and Ronald L. Rivest. "On Estimating the Size and Confidence of a Statistical Audit". In: *Proceedings of the USENIX Workshop on Accurate Electronic Voting Technology*. EVT'07. Boston, MA: USENIX Association, 2007, pp. 8–8. URL: `http://dl.acm.org/citation.cfm?id=1323111.1323119`.

6. Appendix

6.1. Face reconstruction - training images

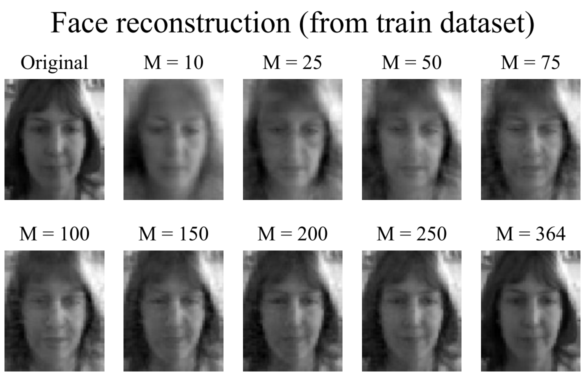


Figure 15: Face reconstruction of a training set image.

6.2. Further comments on face reconstruction

The reconstruction MSE decreases 3.96% when increasing by one the number of eigenvectors used from 10. It decreases by 1.13% and 0.310% when doing so from 50 and 150 eigenvectors respectively. This result supports the argument that using 150 eigenvectors is a good compromise between the algorithm complexity and the reconstruction error.

6.3. In detail description of Nearest Neighbours algorithm

The training data is used to learn the weights associated to each training face when projected on to the face subspace, together with its class label. When a test image is fed into the algorithm as to classify its label, it is projected on to the previously calculated face subspace, and the weights obtained are used by the Nearest Neighbours algorithm to return a predicted label. This is done by finding the k closest labels learnt from the training set, and then choosing the prediction with majority voting.