EE4-68 Pattern Recognition: Coursework 1

Alvaro Robledo Vega
Imperial College London
CID: 01076364

Mariam Sarfraz
Imperial College London
CID: 01069329

1. Eigenfaces

The given dataset consists of 520 faces (46×56 pixels each), representing a total of 52 different people. There are exactly 10 images per individual.

We split the dataset into 70% training and 30% test sets, specifying a fixed *random_state* to ensure that we always operate on the same datasets throughout the rest of this paper, and in a *stratified* way. As a result, our training and test sets have sizes $N_{\text{train}} = 364$ and $N_{\text{test}} = 156$.

1.1. Principal Component Analysis

PCA is a statistical procedure which aims to reduce the dimensionality of data while maximizing the variance. This decreases time and space complexity, and makes it easier to extract patterns from data [1]. Figure 1 shows how our training set looks like when projected onto 2-D space.

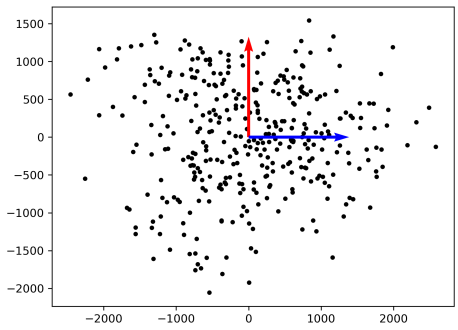


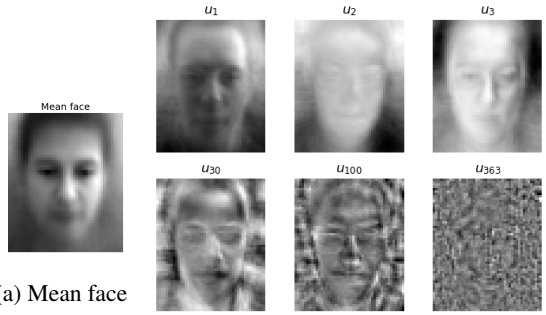
Figure 1: Projection of training set onto 2-D space

To apply PCA to our training dataset, we first compute the average face (shown in Figure 2a), then normalise our data by substracting the average face from each image. Next, we find the eigenvectors and eigenvalues of the covariance matrix $S = (1/N)AA^T$, where N is the number of samples, and A is the normalized training set matrix.

Since matrix S has dimensions 2576×2576, we obtain 2576 eigenvalues, each with their corresponding eigenvector. However, only 363 of them are non-zero. This is because $rank(S) = rank(AA^T) = rank(A) = N_{\text{train}} - 1 = 363$.

Only those 363 eigenvectors with non-zero eigenvalues will be useful when performing face recognition. Each of these eigenvectors is called an *eigenface*. Several of these are depicted in Figure 2b.

In PCA, the eigenvectors (or *principal components*) represent the directions of the largest variance of the data, and the eigenvalues represent the magnitude of this variance in those directions [2]. Thus, when working with eigenfaces, the bigger the size of its associated eigenvalue, the more relevant it will be for our face recognition problem. This is visually represented in Figure 2b, where the top-3 eigenfaces can be seen to contain more information than the others.



(a) Mean face

(b) Top-3 eigenfaces (top) alongside 3 others (bottom)

Figure 2: Mean face together with several eigenfaces

More mathematically, we can express the variance contained in an eigenface as the proportion of its associated eigenvalue to the sum of all eigenvalues. Figure 3 shows the cumulative % of variance contained in the top M eigenfaces. Interestingly, over 85% of the variance is contained just in the first 50 eigenfaces.

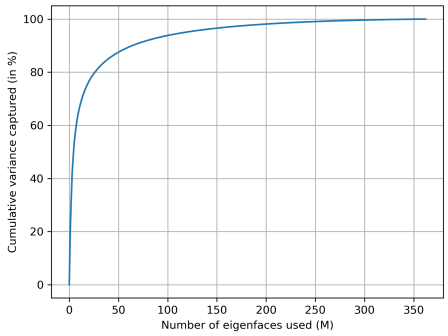


Figure 3: Variance captured (%) for varying M

1.2. Low-dimensional computation

Another approach is to use the *low-dimensional* computation of eigenspace. This involves using $S = (1/N)A^TA$ instead.

S now yields 363 non-zero eigenvalues and eigenvectors. In fact, the eigenvalues are exactly the same as the

363 non-zero eigenvalues from the previous method (refer to Appendix A for proof). Experimentally, the norm of the difference between the vectors of eigenvalues obtained from both methods is 6.19×10^{-10}, small enough to attribute it to numerical and computational noise.

However, the eigenvectors are not the same. This can easily be seen from the fact that they do not even share the same dimensions: these ones are 363×1, whereas the previous ones were 2576×1. However, they are related by $\boldsymbol{u}_i = A\boldsymbol{v}_i$ [3] (refer to Appendix A for proof).

This method has several advantages over the previous one: it is much faster to compute and it requires less memory. From Table 1 we can see that, for our particular case, the low-dimensional method was around 80 times faster. This is because, even though it involves an additional step (multiplying A with $\boldsymbol{v}_i$ to obtain $\boldsymbol{u}_i$), the complexity of computing 364 eigenvalues and eigenvectors is much smaller than that of computing 2576, and overcomes the additional step.

	Original method	Low-dimensional method
Time taken	4.1620 s	0.0524 s

Table 1: Average runtime for each method (in s)

However, all of the above is a consequence of the fact that our number of samples ($N_{train} = 364$) is smaller than the dimensions of each sample ($D = 2576$). If this was not the case, (that is, if we had $N_{train} > D$), then the situation would be reversed, and it would be faster (and more memory efficient) to use our original method of AA^T.

1.3. Application of Eigenfaces

In this section, we explore the applications of PCA using dimensionality reduction and data compression. In PCA, all data points are taken as one big set, independent of their classes. Our main focus is to maximize data variance as this provides optimal results for reconstruction. We will evaluate the quality of image reconstruction using varying number of PCA bases. We will then evaluate two PCA-based face recognition techniques: Nearest Neighbour Algorithm and Alternative Method.

1.3.1 Face Reconstruction

To determine how well a face is reconstructed, we use the *reconstruction error*. For a given dataset of size N, this is calculated as:

$$J = \frac{1}{N}\sum_{n=1}^{N} ||\widetilde{x_n} - x_n||$$

where $\widetilde{x_n}$ (refer to Appendix B for formula) is the reconstructed n-th face and x_n is the original face.

Intuitively, as the number of eigenvectors (M) increases, the more data variance is captured, and the more accurate the reconstruction becomes. Maximizing the variance (i.e. the distance between the projected data point and the origin) is equivalent to minimizing the error (i.e. the distance between the projected data point and the actual data point) [4].

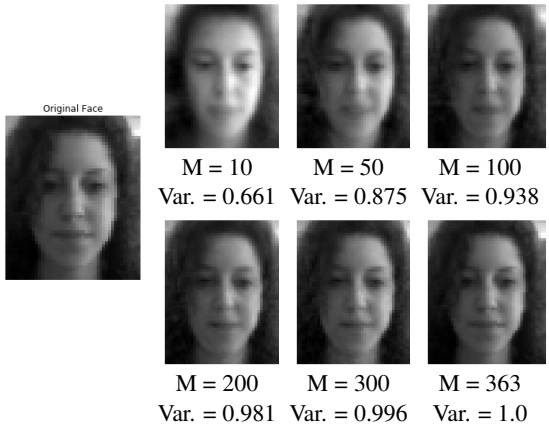


Figure 4: Reconstruction for varying M

In Figure 4, we pick one random test image and reconstruct it using varying number of PCA bases (M). As expected, the quality of the reconstructed image can be seen to improve as we increase M.

Figure 5 shows the *mean reconstruction error* for both our training and test sets for varying PCA bases. This is calculated as a percentage out of the maximum possible error, which is taken to be the error at $M = 0$. As expected, the reconstruction error for both sets decreases with increasing values of M. Particularly, it decreases most steeply for $M < 50$. The plot highlights that the variance and, consequently, the reconstruction error, tends to saturate for $M > 50$. This confirms that, in the domain of face recognition, only the prominent facial details are imperative to capture.

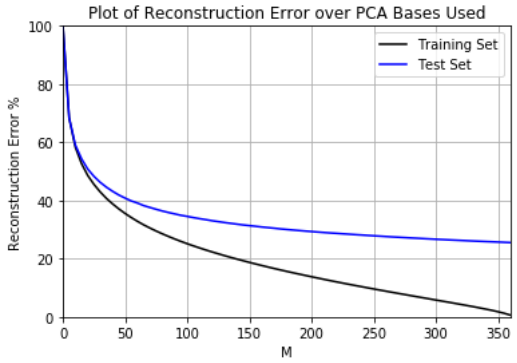


Figure 5: % Reconstruction Error for varying M

The test error is seen to be consistently higher than the training error as our eigenspace is based on the training set. The test set contains the *unseen* data hence, and hence the reconstruction is comparatively less accurate.

1.3.2 Face Recognition: k-NN Classification

In this section, we explore the face recognition technique of k-Nearest Neighbours (NN) classification.

For a fixed M, we calculate the eigenspace. An unknown test image is normalized using the mean face of our training set and then projected onto our eigenspace. The minimum error between the projected test image, w and the projected training images is calculated as:

$$e = min_n||w - w_n|| \quad for\ n = 1, 2, 3...N$$

where w is the projected test image, w_n is the nth projected training image, and N is the number of training images. The class that has the minimum error is assigned as the predicted class of the test image [3].

For $k = 1$, the class of the closest projected point is taken to be the prediction. However, for $k > 1$, a voting method decides the classification by considering the class frequencies of the k nearest projected points. In our case, and probably due to the small size of our dataset, $k = 1$ gave the best results. This was confirmed by running a 5-fold cross validation.

The second parameter we optimize is M. Figure 6 shows the accuracy of the NN-Classifier against varying M. As expected, with increasing M we have increasing accuracy, particularly until $M = 50$, after which, accuracy stabilises at around 65%. We show test-train splits of 7:3 and 8:2. As seen in the graph, there is very slight variation between both curves. However, we choose to use 7:3 as we only have 10 faces per class, and testing on only 2 faces would result in higher uncertainty.

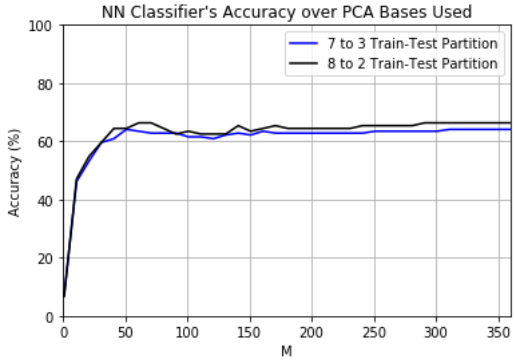


Figure 6: NN Classification Accuracies for varying M

The time taken to run the classifier for varying values of M can be seen in Figure 7. The time complexity for calculating eigenspace is constant as it is independent of M [5]. The complexity for the NN Classification (with $k = 1$) increases with increasing M, as seen in Figure 7: this is because the larger the M, the more time taken for matrix multiplications (projections). The complexity for memory storage is higher than for NN, and it is linear depending on M, $O(M)$, since we are storing the distances.

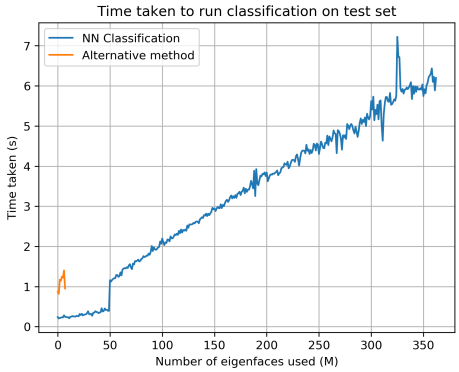


Figure 7: Time taken to run classification for varying M

In Figure 8, examples of correctly and incorrectly classified results are shown for $M = 50$. In the incorrectly classified example, the test image is similar to both the actual and predicted class image. We verify that the predicted class image projections are closer to the test image projection than the actual image projections (refer to Appendix C).

Figure 8: Correctly Classified Example (left) and Incorrectly Classified Example (right) using NN Classification

The confusion matrices for $M = 10$ and $M = 50$ are plotted in Figure 9. These show the correctly and incorrectly classified cases. The diagonal indicates the true class, hence all points on the diagonal are correctly classified and all points elsewhere are incorrectly classified. As expected, we observe that there are more incorrectly classified points for $M = 10$. An interesting result is that classes 2, 7, 41 and 43 are seen to be entirely misclassified for both $M = 10$ and $M = 50$.

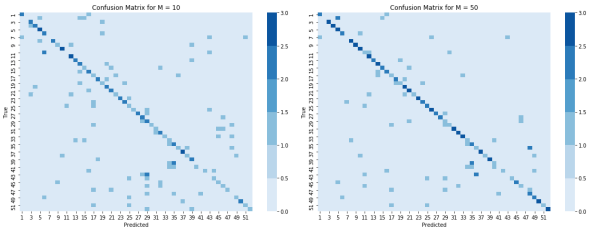


Figure 9: Confusion Matrices for NN Classifiers

1.3.3 Face Recognition: Alternative Method

In this section, another PCA based face recognition technique called Alternative Method [3] is explored. For each class, an eigen-subspace is computed. An unknown test image is then projected on each of these subspaces and its reconstruction error is measured. The class with the minimum

error is chosen as the predicted class. The minimum error is calcuated using:

$$e = arg_i min||x - \widetilde{x_i}|| \quad for\ i = 1, 2, 3...C$$

where $\widetilde{x_i}$ is the reconstructed image by the ith class, x is the test image, and C is the total number of classes. (Refer to Appendix D for an in-depth explanation)

The only parameter to be optimized is M. In Table 2, we can observe that accuracy increases with increasing M. However, for $M > 3$, the accuracy plateaus.

	Accuracy (%)	**Time Taken (s)**
M = 0	53.85	0.79
M = 1	57.69	0.92
M = 2	67.95	1.18
M = 3	71.79	1.19
M = 4	71.79	1.20
M = 5	71.15	1.25
M = 6	70.51	1.38

Table 2: Alternative Method Results for varying M

In Figure 10, we have examples of correctly and incorrectly classified faces (for $M = 3$). Similar to the results seen for NN Classification, the misclassified test image is very similar to the predicted image.

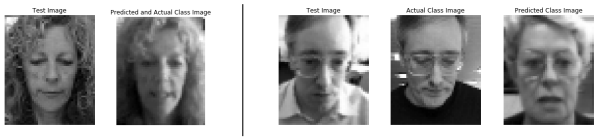


Figure 10: Correctly Classified Example (left) and Incorrectly Classified Example (right) using Alternative Method

It is worth noting that the accuracy obtained from Alternative Method is much higher than that from NN Classification. This is because for NN, we look at the closest projected datapoint, regardless of class. Alternative Method is more resistant to outliers, as the projection of each class's eigenspace is considered separately. This is validated with the confusion matrix (refer to Figure 11).

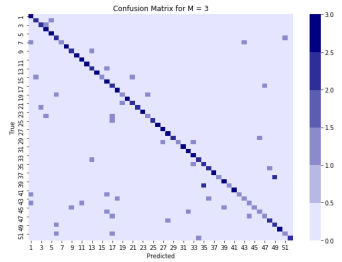


Figure 11: Confusion Matrix for Alternative Method

2. Generative and Discriminative Subspace Learning

For PCA (1) and LDA (2), the optimization problem is solved using the following Lagrange Multiplier Equations:

$$L_1 = w^T S w - \lambda(w^T w - K) \tag{1}$$

$$L_2 = w^T S_B w - \lambda(w^T S_W w - K) \tag{2}$$

To compute the subspace learning that fulfills both aspects of PCA and LDA, we can mathematically formulate the problem by combining both Lagrange Equations [3] in (1) and (2) as the terms to be optimized and constrained remain the same:

$$L = w^T S w - \lambda(w^T w - K) + w^T S_B w - \lambda(w^T S_W w - K) \tag{3}$$

$$\frac{\partial L}{\partial w} = Sw - \lambda w + S_B w - \lambda S_W w = 0 \tag{4}$$

$$(S + S_B)w = \lambda(\mathbf{1} + S_W)w \tag{5}$$

where $\mathbf{1}$ is an all-one D x D matrix. λ and w are the eigenvalues and eigenvectors for $(S + S_B)^{-1}(\mathbf{1} + S_W)$.

3. PCA-LDA Based Face Recognition

3.1. PCA-LDA

In Fisher Linear Discriminant (FLD), all data points are projected onto a lower-dimension space with the aim of optimising data separability of different classes.

In FLD, we aim to maximise the distance between the class means, S_B, and minimize the variations within each class, S_W. This will result in optimal data separability, given by the following function:

$$J(w) = \frac{|w^T S_B w|}{|w^T S_W w|}$$

where S_B is a between-class scatter matrix and S_W a within-class scatter matrix, defined as:

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

$$S_W = \sum_{i=1}^{c} \sum_{x \in c_i} (x - m_i)(x - m_i)^T$$

Using Lagrange multipliers, we reduce the equation, $J(w)$, to the generalized eigenvalue problem:

$$S_B w = \lambda S_W w$$

where w and λ are the respective eigenvectors and eigenvalues of ${S_W}^{-1} S_B$. Please note that from the equations, both S_W and S_B have dimensions $D \times D$.

The Small Sample Size Problem

For S_B, as the global mean face is subtracted from the mean face per class, the sum of all these *normalized* columns will be equal to zero. Hence, we can say that one column can be rewritten as a linear combination of all others. This proves: $rank(S_B) = C - 1$. Similarly for S_W, as the mean face per class is subtracted form all data points, N, we will obtain: $rank(S_W) = N - C$. In order to solve the generalized eigenvalue problem, we need S_W to be invertible. This requires the following condition to be true: $D \leq N - C$. For our given data set, with $D = 2576$, $N = 364$ and $C = 52$, we need to reduce D to at least 312.

The Proposed Solution

The proposed solution is to use PCA-LDA [3]. A technique in which we first we use PCA to reduce dimensions of the feature space to at least 312, with the aim to capture as much variance as possible. We can then perform FLD, to maximize class separability, as S_W is now invertible. The value of M_{lda} is chosen such that $M_{lda} \leq C - 1$. The final, generalized eigenvectors after performing both PCA and FLD are called *fisherfaces*.

Holding the inequalities mentioned above true, we derive the average face recognition accuracies on our test set for all M_{pca} and M_{lda} combinations, as seen in Figure 12. (Refer to Appendix E for details on range used)

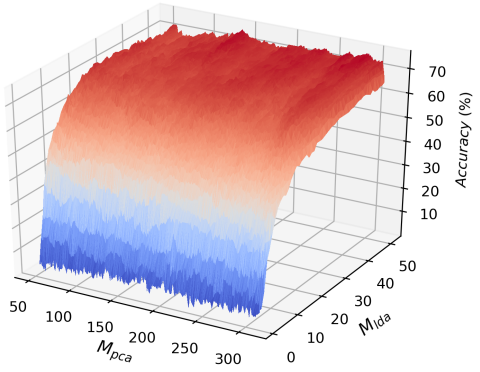


Figure 12: 3-D plot showing the accuracy obtained for varying values of M_{pca} and M_{lda}

In Figure 12, we can observe the accuracy being lowest for $M_{lda} < 10$, regardless of the value of M_{pca}. This makes sense as the the final dimensions are too low to discriminate between classes, regardless of how well the data variance is captured in PCA. As expected, the accuracy shoots up for $10 \leq M_{lda} \leq 50$. M_{pca} serves the sole purpose of reducing the dimensionality while capturing the *right* amount of data variance. We believe the highest value i.e. $M_{pca} = 312$ does not show optimal results because capturing the *maximum* variance results in the use of several low-variance eigenvectors, which might act as noise for our PCA-LDA model. For $M_{pca} = 145$ and $M_{lda} = 48$, we obtain the highest accuracy of 75.64%. In Figure 13, we can see the confusion matrix for these optimal values.

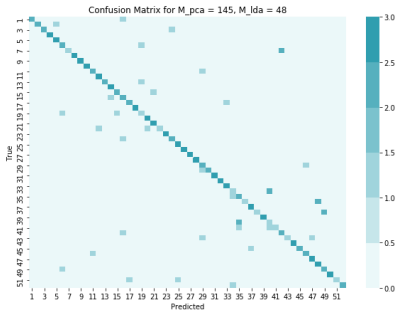


Figure 13: Confusion Matrix for PCA-LDA for $M_{pca} = 145$ and $M_{lda} = 48$

In Figure 14 we have examples of correctly and incorrectly identified faces using PCA-LDA. Similar to the results seen in the last section, the incorrectly classified test image projection is closer to the predicted class projections than the actual class ones. (Refer to Appendix F for 3D visualization)

Figure 14: Correctly Classified Example (left) and Incorrectly Classified Example (right) using NN Classification

3.2. PCA-LDA Ensemble

In Machine Learning, a common approach to improve accuracy is to use *ensembles* of models which are randomly different from each other. If implemented correctly, the randomness introduced by this method can provide greater generalization, usually leading to higher accuracies. Two of the most common ways of creating randomly different models are random training set sampling (bagging) and random model parameter choices (feature randomisation).

3.2.1 Bagging

This technique is essentially based on creating T data subsets by sampling data from the training set *with replacement*. In our case, we generate subsets containing a random set of c_1 images from each of the c classes (which means our bootstrap replicates have a size of $c_1 \times c$), and then we construct a PCA-LDA classifier for each replicate. Finally, we combine these classifiers into an ensemble model using a fusion rule (more on this on section 3.2.3).

The hyperparameters that come into play are M_{pca}, M_{lda}, T and c_1. Due to time and computational constraints, we choose to test the effect of different values of T and c_1, keeping those values of M_{pca} and M_{lda}, which gave us maximum accuracy in the previous section ($M_{pca} = 145$ and $M_{lda} = 48$). From Table 3, we can observe that values below $c_1 = 5$, despite adding more randomness and

	T = 5	T = 20	T = 30
$c_1 = 3$	48.72	55.13	60.9
$c_1 = 4$	61.54	70.51	70.51
$c_1 = 5$	65.38	73.72	74.36
$c_1 = 6$	69.23	73.72	75.64
$c_1 = 7$	66.0	73.72	71.15
$c_1 = 8$	69.87	76.92	75.0
$c_1 = 9$	69.23	74.36	74.36
$c_1 = 10$	71.79	76.28	73.71

Table 3: Accuracy (in %) obtained for different values of T and c_1 using the bagging method

variety, make each model too weak, resulting in bad accuracies. With regards to the parameter T, it shows an increasing monotonic behaviour: the higher the number of models used, the higher the accuracy; however, after a certain value of T the accuracies plateau. This suggests that the number of models used is an important factor only until a certain point, and does not provide significant additional benefits afterwards.

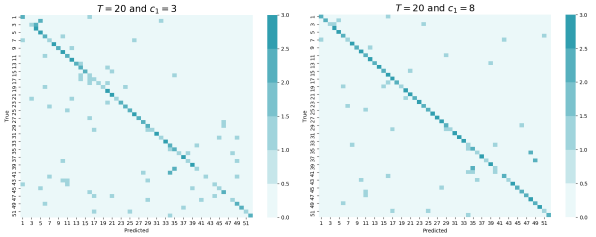


Figure 15: Confusion matrices for $c_1 = 3$ and $c_1 = 8$

From Figure 15, we can see how $c_1 = 3$ makes the models too weak. Since they only take a maximum of 3 unique images per class, the models do not have enough training data to build a robust model (particularly for certain classes), and this out-weights any randomness or decorrelation this technique might be providing the ensemble with.

3.2.2 Feature randomisation

Another approach is to randomise the model parameters. In our case, we construct T models with a small, randomised subset of features: we do this in a similar way as with PCA-LDA, only this time our old M_{pca} is partitioned into $M0 + M1$, where $M0$ denotes the number of top eigenvectors we keep in the model, and $M1$ denotes the number of additional eigenvectors we *randomly* pick from the remaining ones.

Again, we keep $M_{pca} = M0 + M1 = 145$ and $M_{lda} = 48$, and change $M0$ (which, in turn, changes $M1$). Results are shown in Figure 16. Even though this type of feature selection decreases the *strength* of each model, it also decreases the *correlation* between them [3]. We can capitalize on this trade-off by using a committee machine, where, in general, one can still expect that $E_{com} \leq E_{avg}$, where E_{com} is the expected error of the committee machine and E_{avg} is the average error of all models [3]. In fact, in our case, as shown in Figure 16, the accuracy for the committee machine is, in almost all cases, higher than the average accuracy of the individual models.

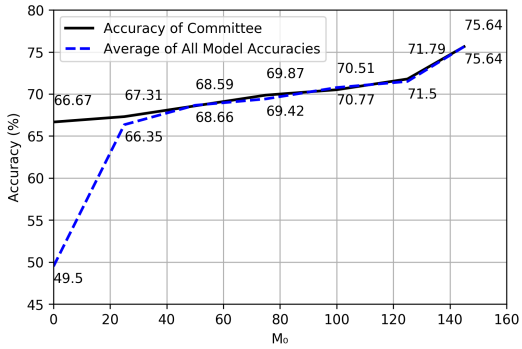


Figure 16: Committee and average accuracies (in %) for different values of M_0 and $M_1 = 145 - M_0$

The randomness parameter ρ controls the amount of randomness as well as the correlation between models. From the scenarios depicted in Figure 16, for $M0 = 0$ we have $\rho = 1$ (the minimum value), which implies that the models have high randomness and low correlation between them. For $M = 145$ we have the maximum value of ρ, which implies low randomness and high model correlation (in fact, all models are exactly the same, hence why the committee and average accuracies are equal).

3.2.3 Fusion rules

In the above scenarios, we have combined all models using *majority voting*, since it is the most appropriate *fusion rule* for our case (1-NN). However, it is worth noting that, if we were to perform k-NN instead, it would be possible to implement other types of fusion rules such as *averaging* or *product* operations [3], or simply using the prediction of the model with highest confidence.

3.2.4 Both approaches together

Combining our feature randomisation method with our stronger bagging method should bring new information to the table, increasing the randomness and decreasing the correlation between models.

We combine the $M_0 = 100$, $M_1 = 45$ feature randomisation model with the bagging model which gave us highest accuracy ($c_1 = 8, T = 20$) using *majority voting*.

The resulting accuracy (73.30%) does indeed improve on that of the feature randomisation model alone. However, it does not improve the best bagging model's accuracy (76.92%). This is probably due to the feature randomisation models being too weak, which raises the question of whether it would be possible to increase this accuracy if a stronger set of parameters for feature randomisation was to be found.

Appendices

A. Appendix

The eigenvalues (λ_i) and eigenvectors ($\boldsymbol{u}_i$) of $S = AA^T$ are given by

$$S\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$$

In the case when S becomes $S = A^T A$, we have that its eigenvalues (λ_i) and eigenvectors ($\boldsymbol{v}_i$) are

$$A^T A\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$$

Multiplying by A on the left we get

$$AA^T A\boldsymbol{v}_i = \lambda_i A\boldsymbol{v}_i$$

Where we note that, letting $\boldsymbol{u}_i = A\boldsymbol{v}_i$, we are back at the very first equation.

Therefore, the eigenvalues for both methods (λ_i) are equal, and the eigenvectors are related by $\boldsymbol{u}_i = A\boldsymbol{v}_i$.

B. Appendix

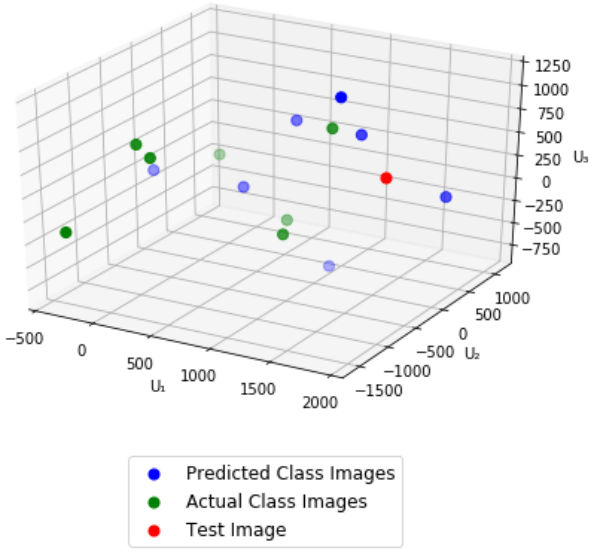
The reconstructed image of the nth face is:

$$\widetilde{x_n} = \bar{x} + \sum_{i=1}^{M} a_{ni}\mathbf{u_i}$$

where $\bar{x}$ is the mean image, $a_{ni\ldots M}$ are the elements of the weight vector (projected data point) of the nth face, $\mathbf{u_{i\ldots M}}$ are the top eigenvectors selected.

C. Appendix

Face Projection in 3 dimensional eigen-subspace using PCA for Incorrectly Classified Test Image.



D. Appendix

As opposed to NN Classification, Alternative Method is heavily dependant on class labels. In our custom function for this method, we construct a reconstruction error matrix of size (N_{test} x C), where each row represents the reconstruction errors for all classes.

$$E = [[e_{11}, e_{12}, e_{13}, ...e_{1C-1}, e_{1C}],$$
$$[e_{21}, e_{22}, e_{23}, ...e_{2C-1}, e_{2C}],$$
$$...$$
$$[e_{N_{test}1}, e_{N_{test}2}, ...e_{N_{test}C}]]$$

The size of the reconstruction error matrix is reduced to (N_{test} x 1) as only the minimum error for each test image (per row) is extracted. The class with this minimum error is chosen as the predicted class.

E. Appendix

While testing for the optimal combination of M_{pca} and M_{lda}, we keep within the following range:

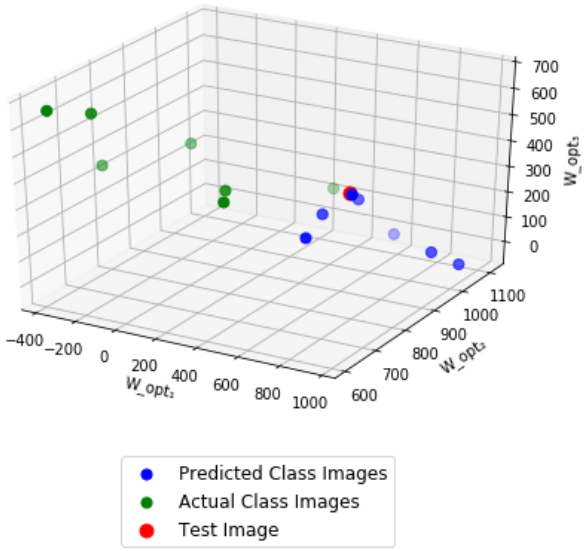
$$51 < M_{pca} \leq 312$$

$$1 \leq M_{lda} \leq 51$$

This ensures that full rank scatter matrices are obtained for S_W and S_B.

F. Appendix

Face Projection in 3 dimensional eigen-subspace using PCA-LDA for Incorrectly Classified Test Image.



References

[1] Gordana Ivosev, Lyle Burton, and Ron Bonner. "Dimensionality Reduction and Visualization in Principal Component Analysis". In: *Analytical Chemistry* 80.13 (2008). PMID: 18537272, pp. 4933–4944. DOI: 10.1021/ac800110w. eprint: https://doi.org/10.1021/ac800110w. URL: https://doi.org/10.1021/ac800110w.

[2] Vincent Spruyt. *A geometric interpretation of the covariance matrix*. Mar. 2015. URL: http://www.visiondummy.com/2014/04/geometric-interpretation-covariance-matrix/.

[3] Tae-Kyun Kim. *EE4-68 Pattern Recognition - Lecture Notes*. 2018.

[4] Dave Blei. *Principal Component Analysis (PCA)*.

[5] Qian Du, Wei Zhu, and James E. Fowler. "Implementation of Low-Complexity Principal Component Analysis for Remotely Sensed Hyperspectral-Image Compression". In: *2007 IEEE Workshop on Signal Processing Systems* (2007). DOI: 10.1109/sips.2007.4387563.