Pattern Recognition Coursework 1

Kaiyue Sun
01197813

Husheng Deng
01178574

1 Background

In this coursework, 10 images of each of 52 different people are given as the data set. The images are taken under different conditions and have been preprocessed into aligned images vectors with the dimension $D = 2576$. To evaluate the robustness and performance of the model, the first 8 images from each person are selected to train the various models, and will be referred to as the training data. The other 2 images from each person collectively form the test data. In this report, various implementations of principal component analysis (PCA) and linear discriminant analysis (LDA) and the combinations of them are going to be analysed.

2 Standard and Low-Dimensional PCA

The experiments are first carried out on the $N = 8$ training images of the first person with standard PCA. Standard PCA calculates the covariance matrix $S \in \mathbb{R}^{D\times D}$. Due to the subtraction of a mean image from each input message, the rank of S is expected to be $N-1$. A proof of the result is provided in the appendix.

Hence, for $N = 8$, 2569 out of the 2576 eigenvalues of S should be zero, and their corresponding eigenvectors become insignificant. Due to finite hardware precision and large dimension of S, the eigen-decomposition of S exihibits numerical errors in presenting complex-valued eigenvectors and extremely small but non-zero-valued eigenvalues. As a real-valued and symmetrical matrix, S must have only has real eigenvalues an eigenvectors. Thus, in all experiments, the imaginary parts of all complex values are dropped, and numpy.linalg.matrix_rank, an SVD-based function is used in many cases to determine the number of the non-zero eigenvalues. To improve the efficiency of the code, the various PCA-based models implemented assumes that image vectors are linearly independent, so the rank of image matrix can be simply deduced from the number of image vectors or principal components stored. This assumption has seen no error throughout the experiments in this report.

Low-dimensional PCA is carried out on the same 8 training images. The difference between the low-dimensional and the standard model is small. The element-wise difference between the sorted eigenvectors of S from the two models are calculated. The maximum of the differences is about 3×10^{-4}, about the magnitude of the eighth significant figure of The magnitudes of the eigenvalues. These differences are therefore negligible. Similar evaluations are also carried out on the eigenvectors. The maximum norm of difference vectors for each pair of eigenvectors is calculated to be 2.96×10^{-15}. The plot of the difference face gives no visual indication of significant difference. When calculating the inter-model eigenvector differences, numpy.linalg.eig is observed not to give deterministic directions of eigenvectors. Therefore, after eigenvectors are calculated, they are all re-aligned such that their first dimension is always positive. This ensures the accurate computation of difference vectors but does not affect face reconstruction and prediction in later stages.

The face data is then split into 416 training images and 104 test images. Feeding both PCA models with $N = 416$ training images, 2161 of 2576 eigenvalues for standard PCA and 1 of 416 eigenvalues are detected as zero-valued, which proves emperically that $rank(S) = N - 1$. Some plots that describes the main features of the dataset can be found in appendix B

From the plot of difference of the first ten principal eigenfaces, vague differences can be observed on eigenface 2, 3 and from eigenface 7 onwards. However, the magnitude of the differences between the models is still small. The maximum eigenvalue difference calculated is 1.59×10^{-3}, and the maximum difference vector norm is 4.34×10^{-11}.

Figure 1: Difference faces of the first 10 eigenfaces

In terms of reconstruction error and prediction accuracy, both models have the same performance for the

given testing set. Reconstruction error decreases as M increases, while prediction accuracy increases with M until about M=110, after which the accuracy stays constant or even slightly decreases at M = 310. It is therefore safe to conclude that the low-dimensional PCA does not significantly degrade the accuracy of the principal components obtained by the model.

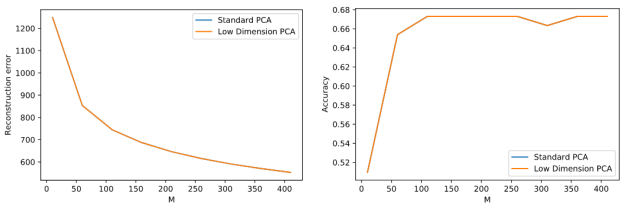

Figure 2: PCA and low-dimension PCA error and accuracy

Due to the alternative covariance formulation in low-dimensional PCA ($N = 416 << D = 2576$), considerable computation time and memory usage reduction is observed compared with standard PCA. From the following figure, the time taken for low-dimensional PCA less than 1s, while standard PCA requires 8 - 9s for the given training set. Standard PCA also takes about 200MB of extra memory compared to low-dimensional PCA. The computation time and memory measurement tools used are timeit and memory_profiler respectively. With jupyter notebook as the experiment platform, the time and memory usage measured cannot be interpreted to be the exact amount required by the algorithms only. They still gives a good indication on the relative time and space efficiency provided by the low-dimensional PCA algorithm due to smaller-sized covariance matrix for eigen-decomposition and storage. Therefore, low-dimension PCA is always preferable to standard PCA when the number of image samples much less than the dimension of the image itself. When the dimensions of the samples become larger than the dimension of the image vectors, standard PCA gives a faster computation.

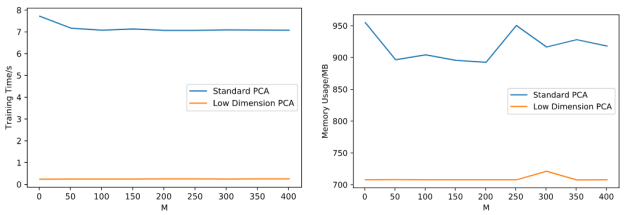

Figure 3: Execution time and memory usage of PCA and low-dimensional PCA

3 Low-Dimensional PCA performance

First of all, reconstruction of training data is performed by using the Low-dimension PCA technique. Each face data point is represented as a coordinate $w_M = [a_1, a_2, \ldots, a_M]^T$ in the M-dimensional eigen-subspace. If $M = N$, the training face can be reconstructed perfectly. If $M < N$, the point is a projection of the original face data point in the N-dimensional space on to the M-dimensional subspace. Therefore, in the N-dimensional space, $w_M = [a_1, a_2, \ldots, a_M, 0_{M+1}, 0_{M+2}, \ldots, 0_N]^T$, the error between w_M and w_N can be expressed as the norm of the difference of the w_N and its projection. $error = ||w_N - w_M||$. Figure 26 shows the change of error against M for all training faces of identity 1 and 2. The change is steepest before $M = 8$, when $M = N = 416$, they all converge to 0.

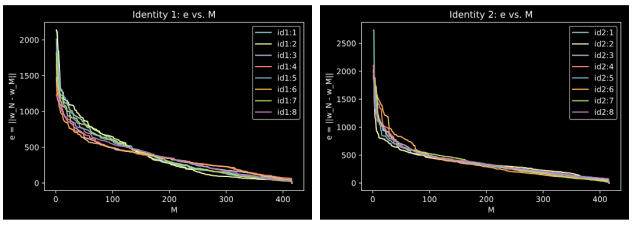

Figure 4: e vs. M

The original training face 0 of identity 1 (0_id1) and the reconstructed face using $M = 50, 150, 416$ are shown in Figure 5. The errors are 827.77, 414.89 and 0 respectively. From the perspective of human vision, the features are captured by and large when $M = 150$ although the plot is a little blurred. All the error curves

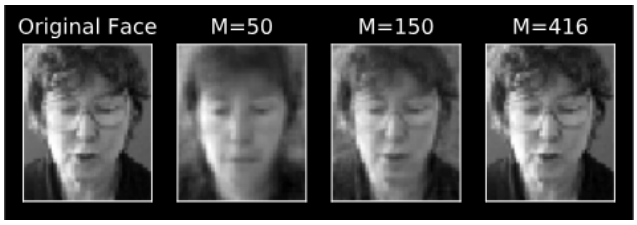

Figure 5: Original and reconstructed face

of single images seem noisy and not smooth, so an average error of all training faces is plotted in Figure 6, which is much smoother and monotonically decreasing. In the space spanned by the eigenvectors of the covariance matrix S which is constructed by the training face data, any of the testing face images does not have an exact coordinates to restore it perfectly. Therefore, to measure the error of the reconstructed testing face, the average coordinate of all the points in the same class is used as a benchmark. $w_{mean} = [\bar{a_1}, \bar{a_2}, \ldots, \bar{a_N}]^T$, $w_M = [a_1, a_2, \ldots, a_M, 0_{M+1}, 0_{M+2}, \ldots, 0_N]^T$, $error = ||w_{mean} - w_M||$. By computing the error in this way, we assume that as a testing point moves closer to a particular training point, it is also closer to the centre of the class of points. In the analysis of face recognition accuracy, it was found that test face 0_id1 was always wrongly predicted no matter what value the integer M took between 1 and 416. After M=26, it was recognised the training face 7_id10. The error of test 0_id1 was plotted against M in Figure 7. After zoomed in, we can

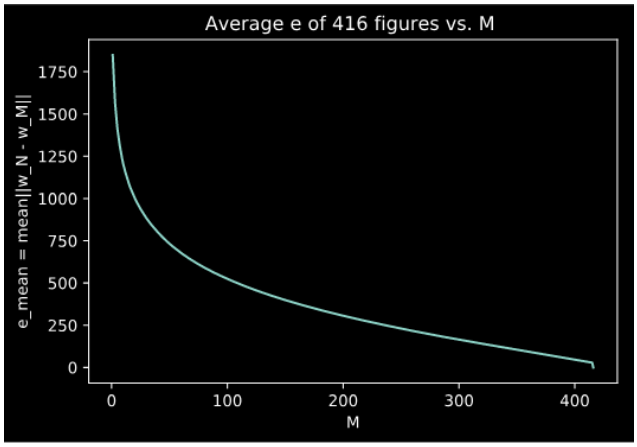

Figure 6: average e vs M

see it starts to rise around $M = 180$ but it is always not close enough to identity 1 when putting together with all of the other training faces. Figure 8 displays the

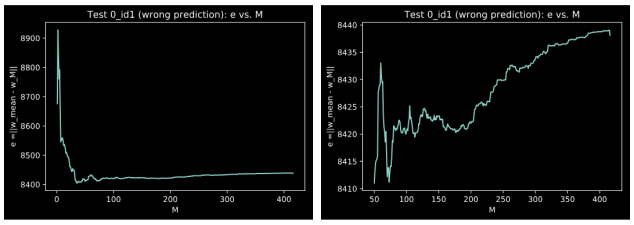

Figure 7: failure case:e vs M

original test face 0_id1, the reconstructed 0_id1 using 180 and 416 eigenspaces and the original training face 7_id10. The reconstructed 0_id1 and 7_id10 do have similar features, for example, the background, angle of view, hair, glasses and most importantly, the brightness. Even so, the reconstructed test face is still closer to it original in human eyes. However, compared with all the training faces of identity 1 (Figure 9) which does not include the original test face, the reconstructed test faces are closer to 7_id10 from the computer vision, because the brightness, angle of view and even some face characteristics such as shape, wrinkle are very different between the reconstructed test face and training faces of the same identity. It is noticeable that the reconstructed image using all the subspaces still has a distance from its original. When using fewer subspaces, the reconstructed image seems smoother, the edges are blurred as the adjacent pixels are similar and correlated. It seems that using higher dimension would add high frequency random noise onto the reconstructed image, which would be made uneven and coarse.

Test face 0_id3 is always closest to training face 6_id3, which means it is always predicted to the correct class. The error plot is shown in Figure 10. It starts to rise around $M = 200$. Again, in Figure 11 the reconstructed image using too high dimension looks worse than lower dimension as suggested by the error plot. But this time, there is a similar enough training image (6_id3) to the testing one from the same class

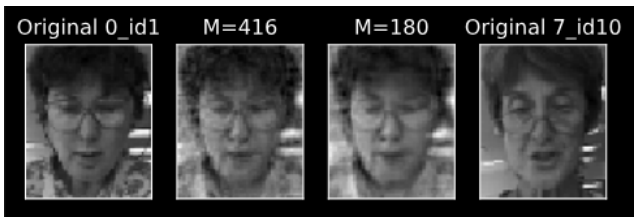

Figure 8: failure case

Figure 9: training faces: id1

and there is not any image from other classes that is coincidentally closer to the testing image. Therefore, the classification succeeded. This implies that successful face recognition depends on both the similarity to the identical class and the distinguishability from all the different classes.

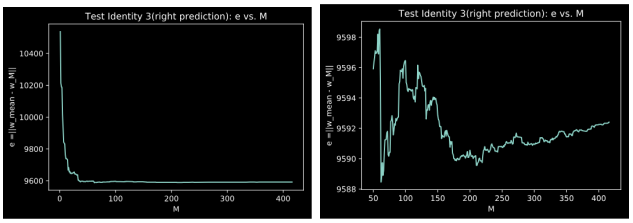

Figure 10: success case:e vs M

It can be seen from some of the error plots again M that the error tends to rise after a certain dimension. By plotting the average error of the 104 test faces in Figure 12, it is extremely clear that the testing error rises at around M=136, which is due to overfitting. However, this error is not directly related to success rate of face recognition since it also depends on the distance from the test images to all the other classes. It can be an overfitting alarm.

Test face is reconstructed by projecting on the M-dimensional eigenspace. The projection is represented as $w = [a_1, a_2, \ldots, a_M]^T$. Face recognition is performed by looking for the minimum error between the test face and the 416 labelled faces in the M-dimensional subspace. Then the test face is assigned the class which the labelled face of minimum error belongs to. When

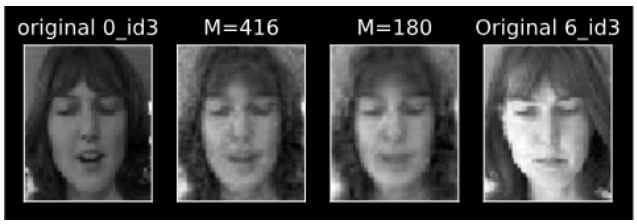

Figure 11: success case

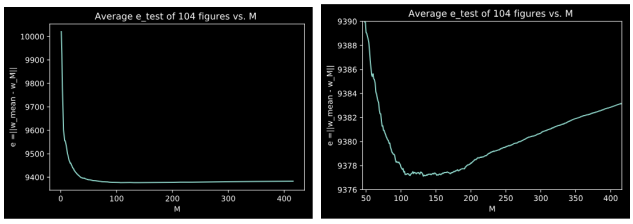

Figure 12: average test e vs M

M=50, the accuracy is 66.35%, 69 faces were correctly recognised among the 104 test faces. The confusion matrix is shown in Figure 13. By varying M, the accuracy is plotted in Figure 14. Overall, the accuracy does not change much after M=50. The number of correctly identified face images varies between 68 and 70 out of 104. After M=94, the accuracy seems overfitted, it stays the same or becomes worse.

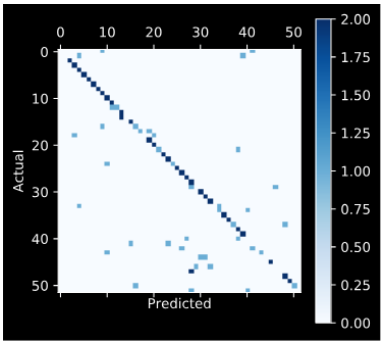

Figure 13: confusion matrix when M=50

4 Incremental PCA

To give the models robustness to merge and split up dynamically, incremental learning can be used to modify the models after initially constructed with a batch PCA algorithm.

The effect of batch size on the efficiency and accuracy of the model is first investigated. In separeted experiments, The training set is further split into multiple batches comprising 1, 2 and 4 images per identity for each experiment. From the figure, when batch size is small, incremental PCA shows significant training overhead compared to other models. This is because the standard covariance matrix is required for each sub-model merged. Small batch sizes cause more calls

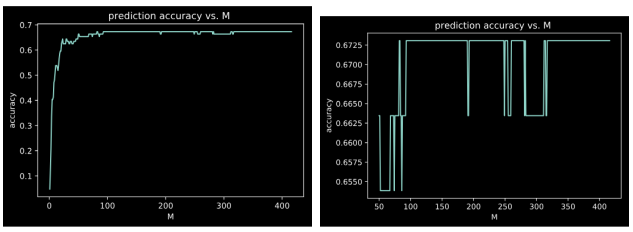

Figure 14: accuracies vs M

to the operation on covariance matrices and increases computation time. However, when M principal components instead the full eigenspaces are stored each case, incremental PCA becomes more efficient than standard PCA because the with calculation of sufficient spanning set Φ, the dimension of $\Phi^T S_3 \Phi$ is much smaller than D, hugely simplifying the eigen-decomposition for optimising sample variance.

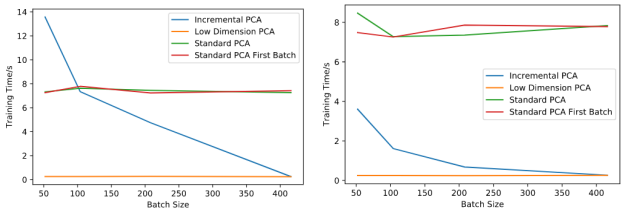

Figure 15: Training time comparison between PCA models with against varying batch size and M

When fixing the batch size to be 104, i.e. 2 images from each identity, incremental PCA exhibits similar reconstruction accuracy to low dimensional PCA. As M increases, subspaces of larger dimension allow more accurate reconstruction. Both model reconstruct much greater than standard PCA with only the first mini-batch, because the dimension of its subspace is limited by N. The recognition accuracy of incremental PCA is also comparable to that of low-dimension PCA, with the models go from underfitting to the best accuracy as M increases to 70. No overfitting is observed as the selected range of M is below $\frac{1}{4}$ of the rank of the image matrix. Further improvement on the efficiency of incremental PCA is also implemented and tested out with the alternative algorithm [1] by simplifying $\Phi^T S_3 \Phi$ with approximations. A reduction of training time by 15% is observed.

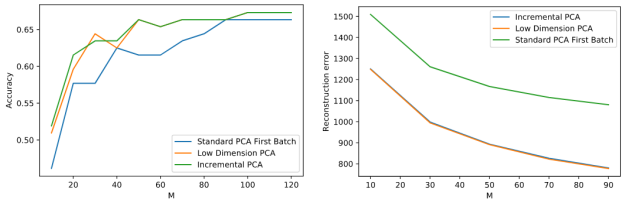

Figure 16: Prediction accuracy and reconstruction error of PCA-based models against M

The optimal training parameter for the incremental PCA include batch size and M. The optimal prediction accuracy is found at M=100 for batch size of 104. It

requires the shortest training time to achieve the prediction accuracy of 0.67, the maximum observed for incremental PCA.

5 PCA-LDA

At the testing stage of PCA-LDA, the input data is projected to a subspace spanned by the first M_{pca} number of eigenvectors generated by Low-dimension PCA ($w_{pca}^T \cdot X$). Then the projected data is fed to PCA-LDA classifier ($y = w_{lda}^T \cdot w_{pca}^T \cdot X$). The differences between the testing face and all the labelled training faces are calculated, the testing face is classified as the identity which the labelled training face of minimum difference belongs to.

As M_{pca} varies from 0 to 364[1] with a step of 10 , and M_{lda} is fixed, the accuracy is plotted against M_{pca} in Figure 17. Each line is plotted with a specific M_{lda}, which is less or equal to 51[2]. When $M_{pca} = 100$, the rank of $(w_{pca}^T \cdot S_W \cdot w_{pca})$ should be 100. When using numpy.linalg.matrix_rank, the returned rank is about 97. This is because the build-in function sets a threshold on eigenvalues to determine the rank. Therefore 2 or 3 nonzero eigenvalues below the threshold are not counted to the rank.
In Figure 17, all of the curves are overfitted after a particular M_{pca} value. With $M_{lda} = 20$, ovefitting occurs earliest when $M_{pca} = 70$, the accuracy reaches its maximum, 88.46%, which means 92 out of 104 test faces are correctly classified. The corresponding confusion matrix is shown in Figure 18. This accuracy is much better than the 66.35% in Low-dimension PCA. The global maximum is 89.42% (93 are correct) when $M_{lda} = 50$, $M_{pca} = 140$. Besides, the accuracy does not show large improvement after $M_{pca} = 50$, which means the first 50 eigenvectors comprise the most important part of the feature space and can capture most of the face characteristics.

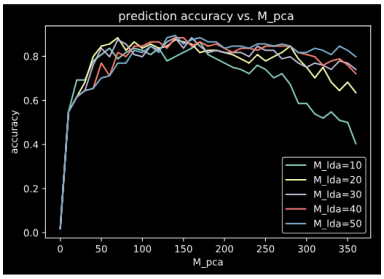

Figure 17: prediction accuracy vs M_{pca}

[1] $364 = N - c = 416 - 52$ is the rank of the within class scatter matrix S_W. M_{pca} should be smaller or equal to this number to reduce the dimension of S_W. Then $(w_{pca}^T \cdot S_W \cdot w_{pca})$ is full rank and has an inverse, therefore, we can find a w_{lda}

[2] $51 = c - 1 = 51 - 2$ is the rank of the between class scatter matrix S_B, but it is a square matrix of dimension $M_{pca} \times M_{pca}$. Therefore, we can reduce its dimension by only taking the generalised eigenvectors of $((w_{pca}^T \cdot S_W \cdot w_{pca})^{-1}((w_{pca}^T \cdot S_B \cdot w_{pca})$ with largest M_{lda} eigenvalues

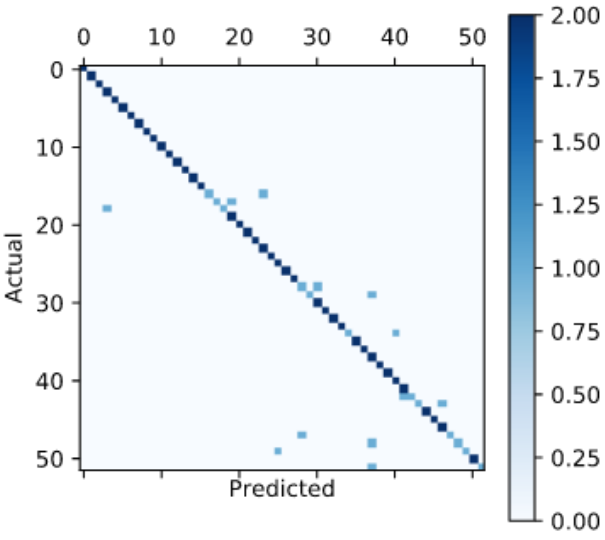

Figure 18: accuracies vs M_{pca}

Test face 1_id18 is wrongly predicted to be training face 20_id7 by PCA-LDA. The two faces are shown in Figure 19. The brightness, face shape, glasses are all similarities between the two faces. While the images in the training set of the testing data shown in Figure 20 are not as close to the testing data as the 20_id7.

Figure 19: failure cases of PCA-LDA

6 PCA-LDA Ensemble

In PCA-LDA Ensemble, the random space is spanned by $M_{pca} = M0 + M1$ dimensions. $M_{pca} = 100$ is fixed and the we adopt different randomisation by varying $M0$ from 0 to 100. Figure 21 displays the accuracy curves each with a unique number of random spaces T. The outputs of the T classifiers are combined using sum rule. Different T does not result in clear discrepancy in accuracy after $M0 = 50$. This result agrees with the conclusion in PCA-LDA. Therefore, after $M0 = 50$, accuracy cannot be bad (within 0.65 - 0.85) no matter how many eigenvectors we pick ($M1$)in the remaining dimensions and how many random subspaces (T) we use. We ran a second experiment using

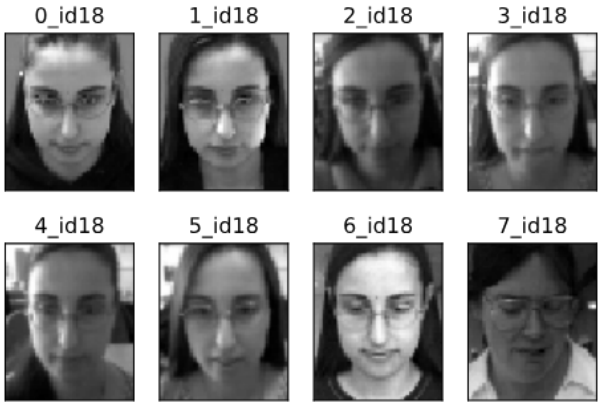

Figure 20: failure cases of PCA-LDA

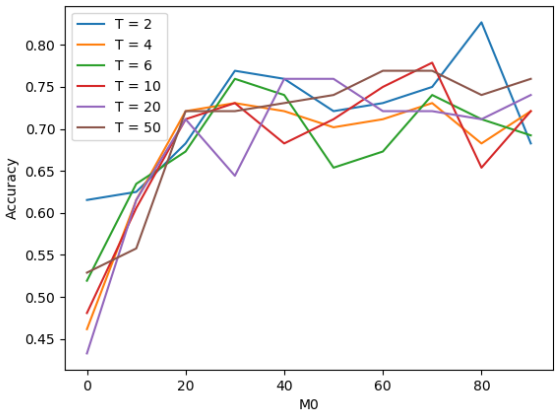

Figure 21: accuracies vs $M0$, sum rule

majority voting rule with the same hyper-parameter values. The accuracies are plotted in Figure 22. The accuracies converge after $M0 = 50$. However, the curves become much smoother than using sum rule and the accuracies also increased to 0.9519 (99 out of 104 are correct) when T=20, $M0$=40. The corresponding confusion matrix is shown in Figure 24 The fluctuating curves in sum rule may implies the uncertainty is high in each random epoch. Therefore majority voting is a more reliable fusion rule and leads to better accuracy.

The calculated committee machine error E_{com} and average model error E_{av} do not conform to the theoretic relationship of $E_{com} = \frac{1}{T}E_{av}$ under the assumption of uncorrelated error. Against different value of M0, their ratio is emperically stable around $E_av \approx 1.05E_{com}$ when $T = 20$ randomised models are used for the committee. Nonetheless, the use committee machine reduces the error of estimation.

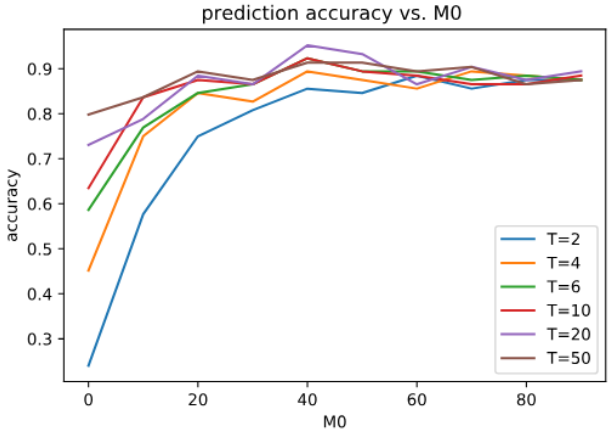

Figure 22: accuracies vs $M0$, majority voting

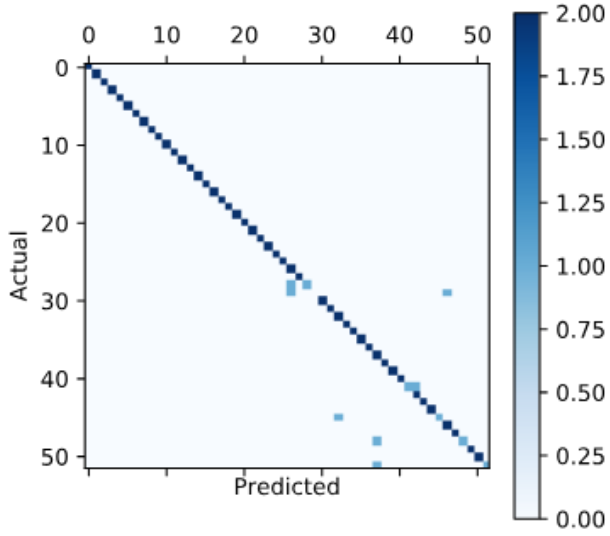

Figure 23: confusion matrix of PCA-LDA Ensemble (majority voting)

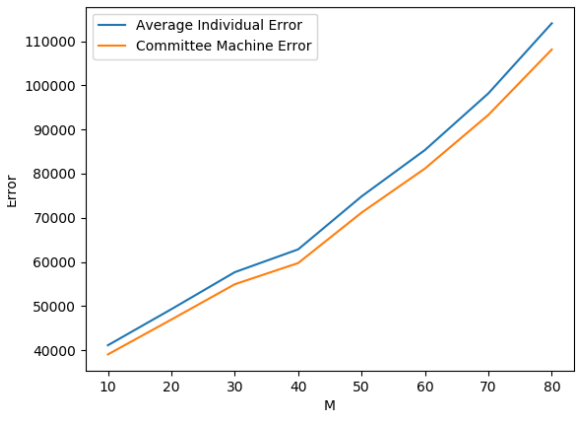

Figure 24: E_{av} and E_{com}

Appendices

A Proof for rank of covariance matrix[2]

The covariance matrix S, ignoring constant coefficients, can be written as:

$$\begin{aligned} S &= \sum_{i=0}^{N} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T \\ &= \sum_{i=0}^{N-1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T + (\boldsymbol{x}_N - \bar{\boldsymbol{x}})(\boldsymbol{x}_N - \bar{\boldsymbol{x}})^T \end{aligned} \tag{1}$$

Note that $\sum_{i=0}^{N} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) = 0$. Hence $\sum_{i=0}^{N-1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}) + (\boldsymbol{x}_N - \bar{\boldsymbol{x}}) = 0$

$$\begin{aligned} &= \sum_{i=0}^{N-1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T + (-\sum_{i=0}^{N-1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}}))(\boldsymbol{x}_N - \bar{\boldsymbol{x}})^T \\ &= \sum_{i=0}^{N-1} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}} - \bar{\boldsymbol{x}}_N)^T \end{aligned} \tag{2}$$

Hence the rank of the covariance matrix is at most N - 1

B Plots of Training Image Features

Figure 25: Mean face

Figure 26: Top 10 eigenfaces

References

[1] Peter Hall, David Marshall, and Ralph Martin. Merging and splitting eigenspace models. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, 22(9):1042–1049, 2000.

[2] amoeba says Reinstate Monica (https://stats.stackexchange.com/users/28666/amoeba-says-reinstate monica). Why is the rank of covariance matrix at most $n - 1$? Cross Validated. URL:https://stats.stackexchange.com/q/180366 (version: 2015-11-05).