EE4-68 Pattern Recognition (2018-2019): Coursework 1

Martin Ferianc, Alexander Montgomerie-Corcoran
{mf2915, am9215}@ic.ac.uk, CID:{00984924, 01052454}

1. Introduction

This coursework investigates a data-set of several varying images of a set of faces. We investigate both discriminative [1] and generative [2] sub-spaces of the data-set, and discuss their effectiveness with respect to classification and reconstruction tasks.

1.1. Data-Set

The total data-set of $N = 520$ centred, grayscale face images of the original dimensions of W and H of 46×56 that were linearised into 1D vectors of dimension D where $D = H \times W$, giving vector $x_i \in R^D; i \in N$. The data-set is class-wise balanced, containing 10 faces per class giving a total of $N_C = 52$ classes. There are significantly more potential features in the data (2576) than there are data points (520).

We split the data into training and test set for testing. We are following a traditional 80%/20% split [3]. The split is done with stratified sampling to ensure equal representation of classes and to avoid bias towards over/under represented classes. Due to sample size, we will use the test set as a validation set for evaluating hyperparameters. This split is crucial to observe whether our model is over-fitting to the training data.

2. Eigenfaces

2.1. Principal Component Analysis

The goal of Principal Component Analysis (PCA) is to represent a set of data, regardless of any labels for it, in a simpler way [4] by reducing its dimensions per sample. For face images in particular, we wish to represent each image as a collection of features rather than individual pixel values. This is done by finding an orthonormal set of vectors which can project the data to a subspace of maximum variance.

2.1.1 Computing PCA

From the procedure outlined in [2], we first find the mean face $\bar{x}$ by computing $\bar{x} = \frac{1}{N} \sum_{n=1}^{N_{train}} x_i$, where N_{train} denotes the number of training images after the split, in our case 416 and x_i denotes image in the training data-set. Qualitatively, the mean face describes what is common between all faces: the general structure with head, eyes and mouth outlined in Figure 10.

The mean face is subtracted from each face, giving $\phi_i = x_i - \bar{x}, \in \mathbb{R}^D$. For training set, we get $A = [\phi_1, \phi_2, ..., \phi_{N_{train}}], \in \mathbb{R}^{D \times N_{train}}$.

Subsequently, we are able to compute the covariance matrix $S_{naive} = \frac{1}{N} AA^T$ for PCA decomposition. The covariance matrix generalises the notion of variance to multiple dimensions.

As $S_{naive} \in R^{D \times D}$, we obtain D eigenvectors. It's worth noting that $rank(S_{naive}) = rank(A) = rank(X - \bar{X}) = N_{train} - 1$, so only $N_{train} - 1$ out of D eigenvectors are of significance and non-zero[1]. After computing the eigenvectors of the covariance matrix, we obtained $N_{train} - 1$ eigenvectors that had non-zero eigenvalues[2]. The issue with this method is that we go through a costly computation to obtain redundant eigenvectors, which leads to long computational time and extensive memory usage due to the high dimensionality.

2.1.2 Efficient PCA

A more efficient method is used to compute PCA of high dimensional data, which reduces computation time, due to reduced covaraince matrix dimensionality. This is achieved using the eigenvector decomposition of $S_{efficient} = \frac{1}{N} A^T A$, where $S_{efficient} \in R^{N_{train} \times N_{train}}$. The procedure to find the eigenvectors z is as follows.

$$\frac{1}{N} A^T A v = \lambda v \iff S_{efficient} v = \lambda v$$

Multiplying both sides with A:

$$\frac{1}{N} A A^T A v = \lambda A v \iff S_{naive} A v = \lambda A v$$

and then letting $z = Av$:

[1] As we have the case that $D >> N_{train}$

[2] Due to the numerical method used, there were actually $D+1-N_{train}$ eigenvalues that were approximately zero.

$$S_{naive} z = \lambda z$$

We note that the rank of $A^T A$ is also the same as in the case of AA^T. Therefore, all we need to do is multiply the eigenvectors z with A to compute the same eigenvectors as in the naive case, v, while the eigenvalues λ are the same for both naive and efficient implementations. Note, as we wish to have an orthonormal basis, we must normalise all the eigenvectors in z after multiplication with A. The total number of meaningful eigenvectors remains $N_{train} - 1$, with the only difference in implimentations is sign of the eigenvectors.

It can also be seen that the efficient method saves computation time: S_{naive} takes 12.64 seconds and $S_{efficient}$ takes 0.11 seconds, with a $115\times$ speed-up in computation time.

There is also saving in memory, due to the reduced covariance dimensions. The observed disadvantage of this method is the added complexity of two more steps by having to multiply the eigenvectors by A and then normalising them, whereas for the naive method we only have to compute the eigenanalysis. Overall, the efficient method proves better, and will be used to obtain the rest of the results.

If we used all the significant N_{train} eigenvectors of S_{naive} we would experience almost no error, except computation error. Nevertheless, we can threshold on the number of M most significant eigenvectors, by looking at the magnitude of their eigenvalues. The reconstruction error can then be quantitatively affected by controlling the number of used eigenvectors and their corresponding eigenvalues, which is proportional to $\sum_{i=1}^{N_{train}} \lambda_i$, where λ_i are the respective eigenvalues. This relationship can be seen in Figure 11.

2.2. Eigenface Evaluation

Having obtained a method of performing PCA on the face images, we can now apply this analysis method for use with both reconstruction and classification problems. We will find how the number of bases for PCA affects the accuracy in both cases.

2.2.1 Reconstruction

Reconstruction is performed by projecting the image from the Eigenface subspace back to the original space.

Figure 1 demonstrates how the number of Eigenfaces affects error[3]. From Figure 2, we can see that the first few eigenvectors with the most significant variance and the highest magnitude contribute the most to the reconstruction of the face. By increasing the number of eigenvectors, we have more *features* present in the reconstructed image.

[3]The error is measured as MSE error.

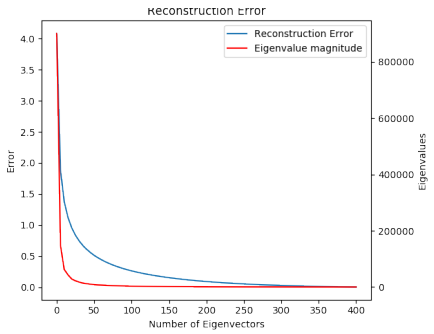

Figure 1: Reconstruction Error and Eigenvalues against Number of Eigenvectors.

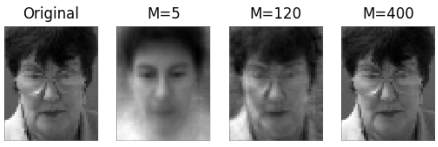

Figure 2: Comparison of reconstructed images for different values of *M*.

Coonversely, having less leads to a more blurred image, similar to that of the mean face.

2.2.2 Classification

PCA can be used to process data before classification to reduce dimensionality with the goal of reducing computation cost as well as making a more simple model. A nearest neighbour (NN) classifier and reconstruction error classifier will be used in this report.

NN Classifier: The NN classifier follows the standard NN algorithm: finds the label of the train image with the smallest L1 norm to the test image. We apply PCA to the test and train images before classification. Figure 4 shows that increasing the number of eigenvectors reduces error, however there is not much improvement after approximately 50 eigenvectors. This is similar to the outcome of reconstruction, and suggests that most of the information of the faces can be summarised in the first 50 dimensions. In fact 88% of covariance is explained in the first 50 eigenvectors, so projection using eigenvectors of little variance will not have much of an effect on classification since all images will have similar points in these dimensions.

Applying PCA does not actually provide much benefit in accuracy, with error actually higher in comparison to baseline NN without PCA, as seen in Table 1. However, there are performance benefits as PCA-NN is $4\times$ faster to compute.

Figure 3 describes the low accuracy of the NN classifier. As we can see, the faces look similar, though are of two different people. PCA is able to obtain these features, however does not hold the more specific information on the classes.

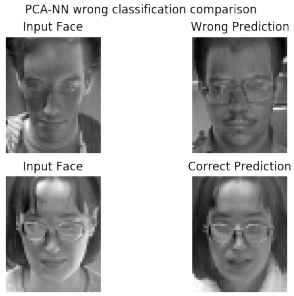

Figure 3: Example of wrong NN classifier.

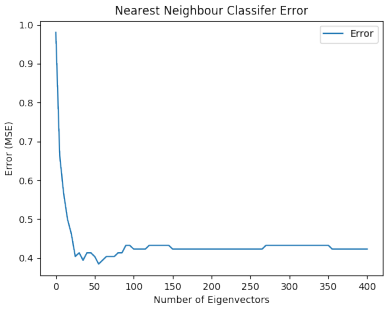

Figure 4: NN Error for different values of *M*.

The features learned by PCA mainly describe light intensity of images, which does not describe facial features entirely. We can say which faces are similar but not say what features are of each class, therefore making it ineffective for classification.

Reconstruction Classifier: If we are to consider the classes of each training image, we will be able to improve classification accuracy. We have designed the classification model as follows: We first obtain the PCA subspace for each class (c_i):

$$W_i = PCA(S_i) \ \textbf{where} \ S_i = \{x_i \ \forall x_i \in c_i\}$$

We then use this subspace to project the test image (x_i) to the subspace and project back for each subspace, choosing the subspace with minimum MSE error.

$$label \ = \ \underset{i}{\operatorname{argmin}} \, ||W_i^T x_{test} W_i - x_{test}||_2^2$$

This classifier holds class information during training, leading to supervised learning. By finding PCA for each class, we are able to extract features common to those classes. Reconstruction using these per class subspaces essentially tells us how prominent these features are in the test image. The issue is that features of a class may be common to other classes aswell. Figure 5 shows that as we increase the number of bases, we are able to better classify the test image as there are more per class features.

Figure 5: Example of Reconstruction Classifier. M refers to number of eigenvectors per class.

Classifier	Error [%]
Baseline NN Classifier	36.5
PCA-NN Classifier	41.3
Reconstruction Classifier	21.2

Table 1: Results of Different Classifiers for section 2. PCA-NN used 50 eigenvectors. The Reconstruction Classifier used 5 eigenvectors per class.

It can also be seen that removing the first eigenvector improves the accuracy for $M = 1$, however it does not apply for more eigenvectors. The first eigenvector is of largest magnitude and is most likely to contain information common to all faces therefore removing it decreases correlation between per class subspaces. The small number of samples however means that the latter eigenvectors of little covariance to the classes so do not help much with classification anyway.

From Table 1, it can be seen that the Reconstruction Classifier performs the best, mainly to the supervised nature in which it learns. The confusion matrices for the best cases can be seen in Figures 12 and 13.

With respect to the memory and compute usage, we have noticed that the resource usage is increasing linearly with respect to the number of eigenvectors, see Figures 14 and 15.

3. PCA & LDA Trade-Off

3.1. Problem Definition

The problem is to describe a model which contains both generative and discriminative features of the input images. To generate a solution, we must understand the data's underlying features. We use PCA for the generative features and Linear Discriminant Analysis (LDA) for the discriminative features. We refer to scatter matrices shown in [1].

The total scatter matrix, S_T, is of rank $N-1$ and tells us about the variation between images and will be used with PCA to find the components which maximise this variance. N is the total number of samples. The between-class scatter matrix, S_B, is of rank $N_C - 1$ and it gives an impres-

sion of the class separation, and allows optimisation of the projected images such that this separation is maximised. The within-class scatter matrix, S_W, describes the variance within each class, which we wish to minimise to achieve clear separation between classes. It is of rank $N - N_C$.

To create a generative subspace, we must maximise the projection of S_T. For a discriminant subspace, we wish to maximise the seperation between classes. This is done by maximising the projection of S_B while minimising the projection for S_W.

To generate a combination of these two subspaces, we present a single objective function defined by a weighted sum of the generative and discriminative subspace. Constant β is a hyper-parameter that defines the ratio of the importance between the subspaces. The objective function is described as:

$$\begin{aligned} &\text{maximise } (1-\beta)W^T S_B W + \beta W^T S_T W \\ &\text{subject to } (1-\beta)W^T S_W W + \beta W^T W = 1 \end{aligned}$$

We constrain β such that $0 \leq \beta \leq 1$. We also constrain the solution such that $W^T W = 1$ to obtain an orthonormal basis as we care only about direction of projection. The first constraint describes the importance of the constraints on the PCA and LDA subspaces. For example, with large β, we allow for large in-class scatter, reducing the ratio between-class seperation and in-class scatter.

A solution can be derived using the Lagrange multiplier with the first constraint. We will choose solutions such that the second constraint is still held. This solution is given by:

$$\begin{aligned} \mathcal{L} &= (1-\beta)W^T S_B W + \beta W^T S_T W \\ &-\lambda(1-(1-\beta)W^T S_W W - \beta W^T W) \end{aligned}$$

This can be solved such that:

$$\nabla_{W,\lambda}\mathcal{L} = (((1-\beta)S_W + \beta I)^{-1}((1-\beta)S_B + \beta S_T) - \lambda I)W = 0$$

W therefore exists in the eigenspace of the previous equation.

This solution provides a trade-off between a discriminative and generative model. By increasing β, we start to see a more generative subspace, as the between-class separation is reduced in magnitude, and the in-class variance has a large constraint. Conversely, decreasing β creates a more discriminative subspace by reducing the effect of total variance. The I term in the inverse of $((1-\beta)S_W + \beta I)$ ensures non-singularity, and therefore makes this problem solvable $\forall\ \beta > 0$. In this aspect, it is similar to a regularised model of LDA [5], where the denominator (S_W) is constrained such that, $S_W = S_W + \alpha I$.

We leave it up to the user to define what they decide is the *optimal* solution. An error function can be used alongside an iterative optimisation method such as Stochastic Gradient Descent (SGD) to find the optimal β for the given problem.

It is worth noting that the solution for LDA and PCA can be found by setting $\beta = 0$ and $\beta = 1$ respectively.

4. PCA-LDA-NN Classifier

We now investigate a classification method which combines PCA, LDA and then NN classifier.

For LDA, the solution is found from $S_B W_{LDA} = \lambda S_W W_{LDA}$, and requires S_W to be non-singular. However, $rank(S_W) = min(D, N - N_C)$, and $S_W \in R^{D \times D}$. Since $D >> N$ for our data, the dimension of the data needs to be reduced to make LDA solvable. By using PCA to project the data to a space with dimensions $< N - N_C$, we are able to create a discriminative subspace. We also benefit from better generalisation, as there are less parameters in the overall model.

For the PCA-LDA aspect to the classifier, we are solving:

$$\begin{aligned} (S_T - \lambda I)W_{PCA} &= 0 \\ ((W_{PCA}^T S_W W_{PCA})^{-1}(W_{PCA}^T S_B W_{PCA}) - \lambda I)W_{LDA} &= 0 \end{aligned}$$

S_T represents the total scatter matrix, W_{PCA}, W_{LDA} the eigenvectors discovered by PCA and LDA respectively. S_B, S_W correspond to the between-class matrix and within-class matrices respectively and λ are the eigenvalues. The overall-optimal projection is defined as $W_{opt}^T = W_{PCA}^T W_{LDA}^T$.

In this section, M_{PCA} refers to the number of of PCA vectors used and M_{LDA} refers to the number of LDA vectors used, both in order of eignenvector magnitude.

4.1. Exploring PCA-LDA

For the PCA-LDA subspace, we have two tune-able parameters: M_{PCA} and M_{LDA}, which we can adjust to find the model that achieves the best accuracy.

Figure 7 describes how the number of both LDA and PCA vectors affects accuracy. It can be seen that having more than 50 LDA vectors decreases accuracy, since it leads to over-fitting to the training data. As the number of PCA vectors increases, the error increases, showing that there is an optimal point balancing under and over-fitting. This can actually be seen more clearly in Figure 8, where we choose the most LDA vectors for the given PCA subspace. There is a point where PCA over-fits to the training data for LDA, and the number of dimensions discourages clear separation.

The best error achieved was **8.7%** with $M_{PCA} = 146$ and $M_{LDA} = 51$ The confusion matrix can be seen in Figure 16. In comparison to section 2.2.2, we observe that the number of PCA eigenvectors is actually $3\times$ higher than in the case where PCA was used alone. We reason, that with 50 eigenvectors the model is not able to discriminate

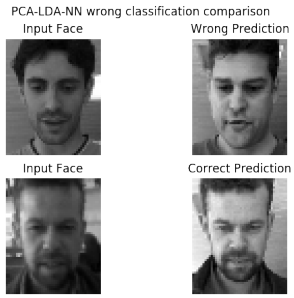

Figure 6: Comparison of wrong and correct classification for PCA-LDA-NN Classifier.

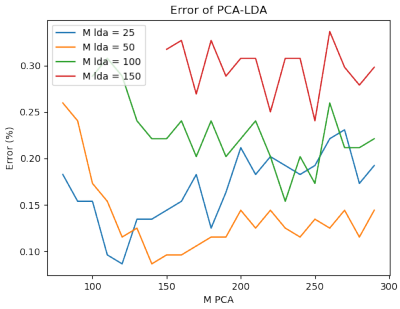

Figure 7: PCA-LDA-NN Classification Error with varying M_{PCA} and M_{LDA}.

Figure 8: PCA-LDA-NN Classification Error with maximum M_{LDA} and varying M_{PCA}.

between different classes and it needs more features to be more discriminative.

Figure 6 shows a comparison of wrong and right classification for PCA-LDA-NN. Similar to PCA-NN, the classifier still finds it hard to determine between similar looking faces. However, it is better at capturing class-based features, and is less dependant on the light intensity of the features unlike PCA-NN. This can be seen for the correct classification, as although they are of different overall light intensity, the classifier can still tell they are both the same face.

4.2. Ensemble Learning

To further improve accuracy of the PCA-LDA-NN Classifier, we perform ensemble learning. We evaluate two Committee Machines formed using two different techniques: Randomising Data Samples (Bagging) and Randomising Model Parameters.

The aim of ensemble learning is to improve the total accuracy of the classifier by using a combination of classifiers. This method of learning relies on there being a low amount of correlation between classifiers, otherwise there will be nothing new discovered from them. Using this technique, we expect to see that $E_{com} \leq E_{av}$ where E_{com} is the error of the committee machine and E_{av} is the average of all of the individual machines.

For each committee machine, we evaluate the machines indidually, and make a classification based on a combination of their outcomes. This leads to signicantly larger cost in computation, as we have to evaluate $N_{machine}$ classifiers.

Fusion Rules: We only evaluate majority voting in this report, as it is the most sensible for discrete classification. As the NN classifier gives us only the predicted label without any confidence, we cannot use this to perform any averaging techniques. Also, as there are discrete classes, averaging or product methods would produce unwanted bias that does not have relevance to discrete classes.

4.2.1 Randomising Data Samples

The first method of ensemble learning evaluated involves randomising the training data. Instead of using the complete training data-set, we take a random subset of it for each classifier to train with.

Bootstrap Aggregation (Bagging) was used to choose samples. The general formula is to uniformly take samples from the training data while replacing each sample in the training data. We look at two ways of doing this: 1. Stratified sampling, with n samples in each class for a total of $n \times N_C$, 2. Complete random sampling, such that there are n samples total across all classes.

We define the *randomness parameter* as the total number of samples for each model. In general, having a large number of samples leads to larger correlation between models as they are more likely to contain similar data points. Therefore, we choose the number of points such that we reduce correlation between models whilst maintaining enough training data per model.

With the completely random sampling method, not every class has equal representation, therefore weighting based on the number of samples per class is needed to compensate for any bias with regards to majority voting. This is done by representing the label output of each machine $n_{C_{label}}$ times where $n_{C_{label}}$ is the number of samples in that class for that label.

Method	Avg. Error [%]	Error [%], Com. Mach.
Random	43.8	26.9
Bagging	30.5	21.2

Table 2: Comparison of Bagging and Random Sampling Methods. They both contain the same number of samples, where $n_{stratified} \times N_C = n_{random} = 260$. There are 10 machines, and $M_{LDA} = 50$, $M_{PCA} = 100$

M_0	M_1	Avg. Error [%]	Error [%], Com. Mach.
0	50	81.4	68.3
0	200	56.2	46.5
50	150	25.7	14.4
150	50	22.0	17.3

Table 3: Comparison of parameter randomisation with different hyper-parameters. There are 10 machines used, each with $M_{PCA} = 200$.

4.2.2 Randomising Model Parameters

Another approach to designing a committee machine is to randomise the hyper-parameters of each machine. With the PCA-LDA machine, we do this by randomly selecting the eigenvectors that form the matrix for PCA projection. We choose the first M_0 ordered eigenvectors of PCA and M_1 of the eigenvectors left uniformly[4]. We fix the size of M_{LDA}.

4.2.3 Evaluation

We have found the best error found through ensemble learning was **5.6%** using parameter randomisation with $M_0 = 140$, $M_1 = 10$, $M_{lda} = 50$ and 17 machines. This is an improvement on our best singular PCA-LDA-NN Classifier, however does not show a significant improvement in accuracy, given the computation that is required to produce the result. The confusion matrix is shown in Figure 17.

Random Samples: Results can be found in Table 2. It can be seen that the bagging method provides better accuracy. This is due to the fact that it has enough of a representation of each class, so it can find a stricter separation between them. In the completely random case, there will be some machines with no members of a class, so they will not be able to classify certain classes at all.

Random Parameters: Table 3 contains the results for varying parameters of M_0 and M_1. It can be seen that by having a significant size for M_0, we are able to achieve better per machine accuracy, as it is ensured there will be vectors with significant contribution to class separation. However, this leads to high correlation between machines, as they share many vectors of significance.

4.2.4 Number of Committee Machines

The number of committee machines has an affect on both accuracy as well as the execution time.

[4]There is no replacement in this method.

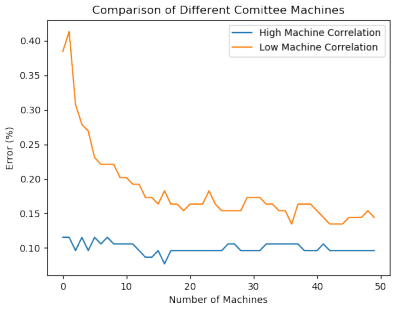

Figure 9: Number of committee machines variation. High Correlation: $M_0 = 140$, $M_1 = 10$. Low Correlation: $M_0 = 50$, $M_1 = 100$.

In general, increasing the number of committee machines increases execution time and memory usage linearly as we are computing the same problem several times.

With regards to accuracy, Figure 9 gives a good overview of how the number of machines affects the committee machine for different hyper-parameters. For a committee machine with low between-machine correlation, there seems to be a benefit for accuracy by increasing machines. This is useful for a case where we do not know the optimal machine hyper-parameters, and have the ability to execute many computations. In the case of low between machine correlation, there is only a little improvement whilst increasing the number of machines, and it is best to find the optimum hyper-parameters for a single machine if we care for computational cost.

5. Future Work

In this report we cover the basics of PCA and LDA, and explore how the hyper-parameters of each affect both classification and reconstruction. However there are many improvements that can be performed to obtain better accuracy for both.

We are assuming that our classes are linearly separable which might not be the case. We could perform kernel based PCA and LDA to project the data into non-linear space, where the data might be better fitted for classification. It has also been suggested that for face images, the first 3 PCA components can be removed for classification as they do not hold any class-specific features [1]. Last but not least, we could explore whitened PCA which makes the features less correlated between each other, and that the features all have the same variance, which can potentially improve reconstruction for previously unseen cases and result in lower error rate in comparison to standard PCA.

It would also be interesting to investigate using supervised learning for classification, such as SVM. We have seen in section 2.2.2 that using class information with classification dramatically improves accuracy.

References

[1] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 711–720, July 1997.

[2] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, pp. 71–86, Jan. 1991.

[3] Y. S. Abu-Mostafa, *Learning from data : a short course*. United States: AMLBook.com, 2012.

[4] J. Shlens, "A tutorial on principal component analysis," in *Systems Neurobiology Laboratory, Salk Institute for Biological Studies*, 2005.

[5] S. Mika, G. Rtsch, J. Weston, B. Schlkopf, and K.-R. Mller, "Fisher discriminant analysis with kernels," 1999.

A. Appendix of Figures

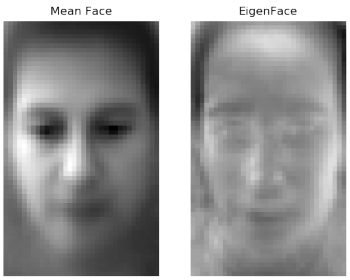

Figure 10: Mean face image and eigenface computed from the training set.

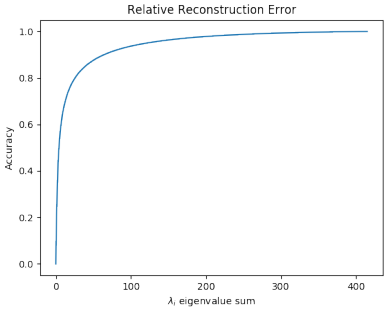

Figure 11: Relative accuracy with respect to M sums of eigenvalues.

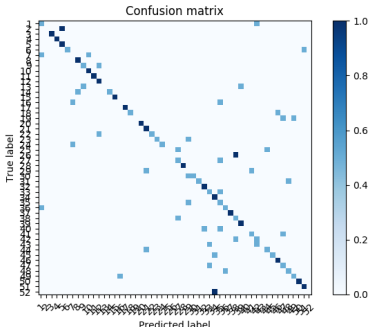

Figure 12: Confusion matrix for the NN-PCA classifier.

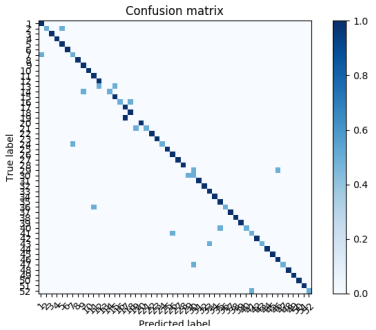

Figure 13: Confusion matrix for the Reconstruction PCA classifier.

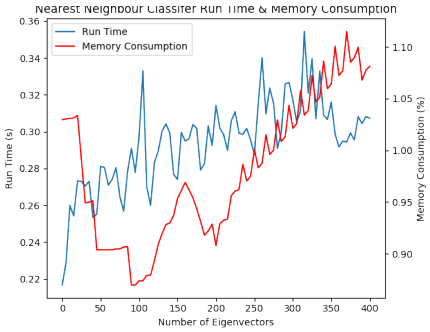

Figure 14: Memory and compute usage for the general PCA classifier.

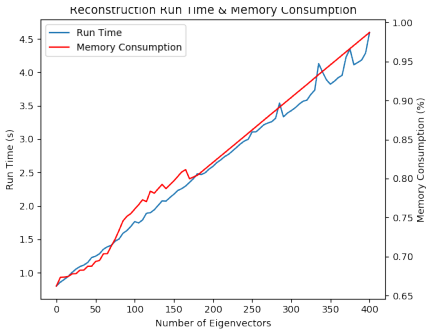

Figure 15: Memory and compute time usage for the Reconstruction PCA classifier.

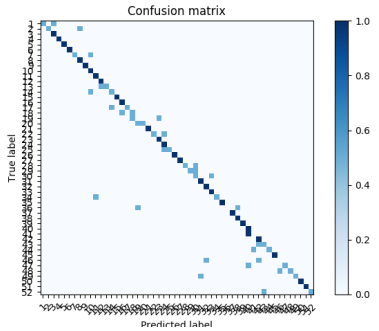

Figure 16: Confusion matrix for PCA-LDA classifier.

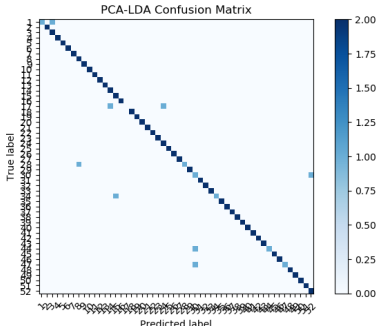

Figure 17: Confusion matrix for ensemble PCA-LDA classifier.