Coursework on the Computationally Efficient PCA, Incremental PCA, PCA-LDA Ensembles for Face Recognition

Spyridon Couvaras (CID: 01119198), Renke Wang (CID: 01762400)
Department of Electrical and Electronical Engineering, Imperial College London

I. Computationally Efficient Eigenfaces

A. Eigenspace Computation

In the first experiment, the Eigenface approach is considered, where Eigenfaces capture the variations among a set of different images. Principal Component Analysis (PCA) employs Eigenfaces to provide a compact feature of representation of a face image. The Eigenfaces, are considered as vectors that span a space known as Eigenspace. PCA is a method to compute the principal Eigenspace. When performing batch PCA, given a set of image-vectors $\mathbf{x} \in R^D$, a computationally efficient approach is to project the data onto a lower-dimensional Eigenspace R^M (M $\ll$ D), on which the data variance is maximized.

Initially, 'face.mat' file is loaded where matrix X consists of 520 image-vectors $\mathbf{x}$, where $\mathbf{x} \in R^{2576}$. For each identity, 10 images are assigned, so overall 52 different people are in the original dataset. The training data consists of the first 8 images from each identity and the testing data consists of the 2 images remaining from each identity. Using the training data obtained, an average face vector $x_{average} = \frac{1}{N}\sum_{n=1}^{N} x_n$ can be computed, where N is the number of training images. The resulting image is shown in Fig. 1.

Fig. 1. Average face of all 52 identities.

After computing the average face, the covariance matrix S is computed. Eigen decomposition is one connection between a linear transformation and the covariance matrix. The value of the covariance matrix is computed using:

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(x_n - x_{average})(x_n - x_{average})^T \qquad (1)$$

where x_n is the n^{th} training image. In our case, covariance matrix $\mathbf{S} \in R^{2576\times2576}$, which is relatively large in terms of dimension.

B. Low Dimensional Technique for Eigenspace Computation

An alternative way to get eigenvectors with corresponding non-zero eigenvalues is low-dimension technique. In this computationally efficient approach, alternative covariance is computed as:

$$\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}(x_n - x_{average})^T(x_n - x_{average}) \qquad (2)$$

The eigenvectors and eigenvalues were computed for both approaches using function 'eig' in MATLAB. For the low-dimensional approach, there are 416 eigenvalues and eigenvectors. Only 1 eigenvalue can be considered as 0 as its value is 4.13×10^{-12}. For the high-dimensional approach, there are 2576 eigenvalues and eigenvectors. From the set of eigenvalues, only the first 415 can be considered non-zero. For both approaches, the non-zero eigenvalues and eigenvectors are the same, which means that both approaches have identical results. The timing needed for the low dimensional approach was measured around 0.27 s, while for the high dimensional one 3 s. The drawback of the large dimensional technique is the additional computational cost due to the size of the covariance matrix described in (1). This makes the high dimensional technique less efficient compared to the low dimensional technique for real-time applications, as the computational cost plays a major role.

Fig. 2. Face reconstruction using increasing number of bases M.

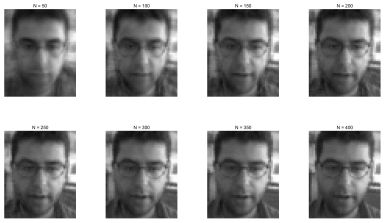

Fig. 3. Face reconstruction using increasing number of bases M.

C. Face Reconstruction

Using the low dimensional approach, image reconstruction was implemented for different images. Initially, the eigenvalues (with the corresponding eigenvectors) were sorted in

descending order. The reconstruction was implemented for a varying number of eigenvectors (PCA bases learned). As shown in Fig. 2 and Fig. 3, when having an increase in the number of the PCA bases, the reconstructed image looks more similar to the original one.

D. Face Recognition with NN Classifier

As a next step, face recognition is performed using the Nearest Neighbor (NN) algorithm. As described, the eigenspace is spanned by the eigenvectors used. Firstly, all training images are projected onto the eigenspace. For the set of 104 testing images, each one is projected on the same eigenspace that the training images were projected on. We denote ω as the projection of a testing image onto the eigenspace and ω_n as the projection of a training image, then the distance between a test image and training image on the eigenspace e can be defined as:

$$e = ||\omega - \omega_n|| \tag{3}$$

For each testing image, the training image that gives the smallest distance as shown in (3), is considered to be the of the same class as the testing image.

By performing the NN classification for the set of testing images, some face recognitions were successful (see Fig. 4) and others were not (see Fig. 5).

Fig. 4. Successful image recognition.

Fig. 5. Unsuccessful image recognition.

The success rate of the face recognition for all testing images, for varying number of bases learned, is shown in Fig. 6. The number of bases varies from 10 to 400. As it is seen, when the number of bases used is less than 100, then the success rate is increasing with fluctuations until reaching the maximum value of 0.6731 (67% accuracy). After $M = 100$, the plot is fluctuating, with the success rate ranging from 0.6731 to 0.6635.

Finally, the confusion matrix is plotted as shown in Fig. 7. As it is seen, across the diagonal of the matrix the colour is blue, which indicates that across the diagonal there were many successful face recognition. That was expected as the algorithm reaches almost 70% accuracy. For a smaller number of chosen principle components, confusion matrix are given in Appendix.

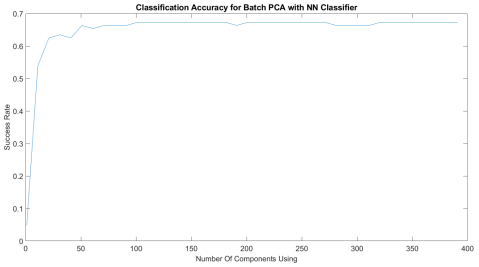

Fig. 6. Testing image recognition accuracy for varying number of bases M.

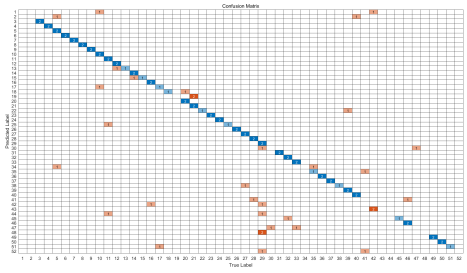

Fig. 7. Confusion matrix

II. Incremental PCA

In this section, incremental PCA is being used to compute the eigenspace model in a way that is more realistic to real applications. In batch PCA performed before, all 416 training data were used to get the eigenspace model. Incremental PCA updates an existing eigenspace by adding new training data over time. Batch PCA uses covariance matrix (2) to compute the eigenvectors P and the eigenvalues Λ.

$$S = P\Lambda P^T \tag{4}$$

In incremental PCA different set of data are being combined. Having two sets of observations, we consider their eigenspace models:

- $\Omega = (\overline{x}, U, \Lambda, N, C)$
- $\Psi = (\overline{y}, V, \Delta, M, D)$

$\overline{x}, \overline{y}$ are the training data mean vectors of each set. U, V are the eigenvectors of each set and Λ, Δ are the eigenvalues. N,M correspond to the training vectors of each set and C,D are the covariance matrices of each set.

The combined covariance matrix of two sets is:

$$S_3 = \frac{N}{N+M}C + \frac{M}{N+M}D + \frac{N*M}{(N+M)^2}(\overline{x} - \overline{y})(\overline{x} - \overline{y})^T \tag{5}$$

Where,

$$S_3 = W\amalg W^T \tag{6}$$

The eigenvectors W we seek can be represented as

$$W = \Phi R \tag{7}$$

Φ, is an orthonormal basis set that spans both eigenspaces (Ω, Ψ) and $\overline{x} - \overline{y}$. R is a rotation matrix needed to rotate the

orthonormal basis set to get the required eigenvectors. The sufficient spanning set Φ is written as:

$$\Phi = [U, v] \tag{8}$$

Where, $v = h([H, a])$. H contains the residues of eigenvectors V with respect to the eigenspace Ω and a is residue of $\overline{x} - \overline{y}$ with respect to Ω. h is an ortho-normalization function which makes vectors orthogonal to each other and normalises them.

Having the combined covariance matrix S_3 :

$$S_3 = W\Pi W^T = \Phi R \Pi R^T \Phi^T \Rightarrow$$

$$\Phi^T S_3 \Phi = R\Pi R^T \tag{9}$$

Equation (9), is the new eigenproblem considered. By getting the eigenvectors of the equation above we can get the rotation matrix R needed to compute the eigenvectors W as shown (7) [1].

For our set up initially, the 416 training data were separated into four different subsets. So each subset contains 104 vectorized images.

For the comparison between IPCA and PCA, we are considering three different cases:

1) Batch PCA performed on all 4 data sets
2) Incremental PCA performed on all 4 data sets taking
3) Incremental PCA performed only on the first data set

A. Training Time Comparison

Training time for the above three cases is computed and reported in Table. 1. IPCA is expected to have a lower computation time than Batch PCA since it only performs an update on existing eigenspace. However, we are getting around 0.5 s for IPCA and 0.3 s for Batch PCA. This is because the data sets we are using are of comparable dimension, and taken into account extra steps for eigenspace merging [1], we might see a longer training time for IPCA, for example, in the orthogonalization step, it has complexity of $O(N^2M)$, which is far higher than the complexity of eigen-decomposition.

TABLE I
TRAINING TIME COMPARISON

Batch PCA	0.27129 s
Incremental PCA with 4 datasets	0.49873 s
Incremental PCA with 1 dataset	0.16123 s

B. Reconstruction Accuracy Comparison: SSIM

After the training times between the three methods were compared, the reconstruction accuracy of each method was calculated. The results are displayed in Fig. 8. For computing the reconstruction accuracy, the Structural Similarity Index (SSIM) was used [3]. For better comparison, we normalized the x-axis to the ratio of components chosen. So now the x-axis is between 0 and 1.

As it is shown in the Figure above, batch PCA was the most successful in reconstructing the original image for a varying number of bases learned. The second most successful technique was incremental PCA and the least successful was PCA trained only with one of the subsets. The decrease of accuracy in IPCA algorithm compared to Batch PCA results from the fact that approximations are used to merge different eigenspaces to decrease computation difficulty.

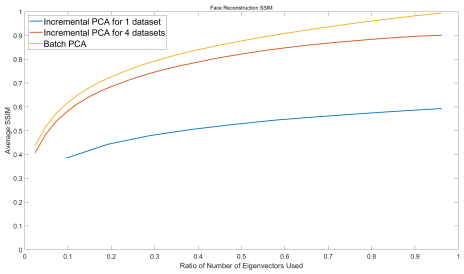

Fig. 8. Face reconstruction accuracy comparison

C. Recognition Accuracy Comparison

After computing the face reconstruction accuracy of each case, the face recognition accuracy of each case was also considered. The results are shown in Fig. 9.

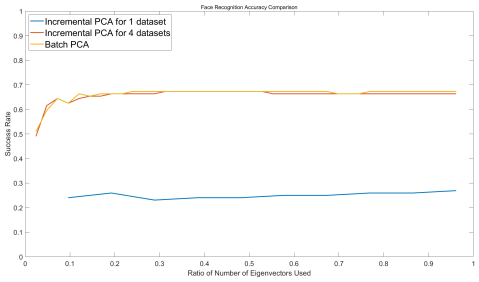

Fig. 9. Face recognition accuracies comparison

The batch PCA approach showed to have the best face recognition accuracy compared to the other approaches. Incremental PCA shows to have very similar accuracy to the batch PCA, while PCA trained only by the first subset has the lowest accuracy. That was expected as the eigenspace created by the PCA trained only with one of the subsets, does not cover many discriminating features of the images.

III. PCA-LDA Ensemble for Face Recognition

A. PCA-LDA Based Face Recognition

In this section Linear Discriminant Analysis (LDA) is a learning method introduced for finding the best direction for discriminating our data. As described before, PCA finds the subspace for which the variance between data is maximised. On the other hand, LDA finds the subspace that maximises the separation between multiple classes. That means that PCA is considered an unsupervised algorithm, whereas LDA is considered a supervised algorithm. LDA introduces the concepts of within-class scatter matrix S_W and the between-class scatter matrix S_B. Where S_W and S_B are defined as:

$$S_W = \sum_{n=1}^{c} \sum_{x \in c_i} N_i (x - m_i)(x - m_i)^T \tag{10}$$

$$S_B = \sum_{n=1}^{c} N_i (m_i - m)(m_i - m)^T \tag{11}$$

In PCA the optimal eigenspace for the data to be projected on is defined as:

$$W_{opt} = arg\{\max_w |w^T S_T w|\} \tag{12}$$

So, the optimal eigenspace W_{opt} maximises the determinant of the total scatter matrix of projected samples.

For better discriminate different classes of data, intuitively we would like between-class distance to be larger and within-class distance to be smaller. Thus, the optimal subspace for LDA is defined as:

$$W_{opt} = arg\{\max_w \frac{|w^T S_B w|}{|w^T S_W w|}\} \tag{13}$$

The optimisation problem defined in (13) is known the criterion for Fisher Linear Discriminant (FLD). The solution of the optimisation problem is produced using Lagrangian multipliers. The solution given is:

$$S_B w_i = \lambda_i S_W w_i, i = 1, ..., M \tag{14}$$

In equation (14) the rank of matrix S_B, where $S_B \in R^{D \times D}$ is at most c-1. In many cases, S_W is a singular matrix as its rank is at most N-c. This is known as the small sample size problem and it increases the computational difficulty of implementing LDA [5]. This happens because often there are limited training observations N compared to the feature space D (N¡D). To overcome the singularity problem Fisherfaces approach is used. For the Fisherface approach, PCA is initially used to reduce the dimension of feature space to an intermediate dimension which should not be more than the rank of S_w [4]. When this happens, the resulting within-class matrix becomes full rank and then FLD is applied. This means that the new optimisation problem changes from (13) to :

$$W_{lda} = arg\{max_w \frac{|W^T W_{pca}^T S_B W W_{pca}|}{|W^T W_{pca}^T S_W W W_{pca}|}\} \tag{15}$$

Where W_{pca} is defined in (12). And the final projection matrix is:

$$W_{opt} = W_{pca} W_{lda} \tag{16}$$

Using training data, we first compute the scatter matrix S_B and S_W and check their rank:

$$rank(S_B) = 51 = c - 1 \tag{17}$$

$$rank(S_W) = 364 = N - c \tag{18}$$

This is exactly what we are expecting. Since there are 52 classes in total, thus the between-class matrix should be of rank 51. Since $S_T = S_B + S_W$, which is not full rank, which is of rank $N-1$, the rank of S_W should be $N-1-(c-1) = N-c$.

Then we implemented PCA-LDA algorithm. By varying hyper-parameters, we get different recognition accuracy which is shown in Fig. 10. As we can see, recognition accuracy is increasing with the increase of M_{LDA}, when M_{LDA} reaches its maximum value ($M_{LDA} = 51$) the recognition accuracy around 0.9. However, it is also noticeable that the increase of M_{PCA} doesn't affect the performance that much. This happens since PCA is no longer the criterion of discriminating our data, but is just used to solve the small sample size problem mentioned before. Compared with batch-PCA or incremental PCA, we see that by forcing the criterion in Equation (13) to be higher, we gain much better performance. Another observation we can make is that for a fixed M_{LDA}, increasing M_{PCA}, we might see a decrease in accuracy, and this is probably due to over-fitting of training data. The change of recognition accuracy with respect to M_{PCA} and M_{LDA} are given in Appendix.

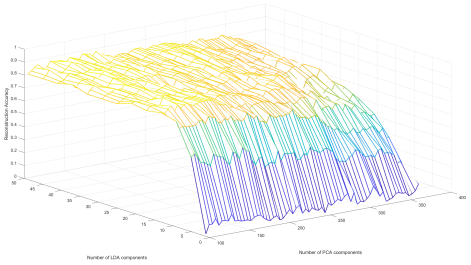

Fig. 10. Face reconstruction accuracy by varying hyper-parameter values

The resulting confusion matrix after NN classification is shown in Fig. 11. On the graph, we see that the diagonal value are mostly dark blue, which indicates we have very high accuracy. Additionally, we see much fewer red blocks outside the diagonal of the confusion matrix, compared to the confusion matrix shown in Fig. 7.

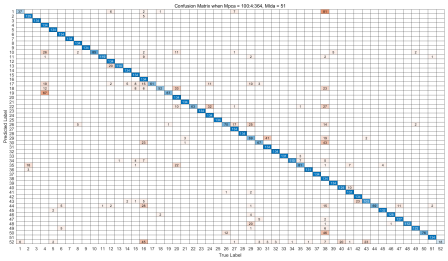

Fig. 11. Confusion Matrix

In Fig. 12 and Fig.13, we give examples of success case and fail case.

B. PCA-LDA Ensemble

The basic idea for ensemble machine is that by ensembling multiple learning models, we obtain better predictive performance than that could be obtained from any of the constituent

Fig. 12. Successful image recognition

Fig. 13. Unsuccessful image recognition

learning models. The ensembles can be shown to be able to over-fit the training data more than a single training model would.

One key point in the ensemble model is that to get better overall performance, we need the constituent models to be randomly different from one another. The randomisation, in fact, results in improved generalization in the sense that, overall mitigates the over-fitting problem in each individual model. The conflict here is that we want every individual model to be optimal, but if all constituents are the same, there would be no need for an ensemble machine.

Randomness can come from either training or feature space. Here we are adopting feature space randomness by forming the feature space in a relatively random and flexible way.

For our experiment, M_0 denotes the predetermined fixed number of eigenvectors in W_{PCA}. M_1 denotes the number of randomly chosen eigenvectors. And we set: $M_0 = 100$ and $M_1 = 50$. What we do it to pick the eigenvectors corresponding to the M_0 eigenvectors with the largest magnitude. Then we randomly choose M_1 eigenvectors from the remaining set. Combining those two sets, we get W_{PCA} for an individual model.

After that, we use the same PCA-LDA algorithm for finding final feature space.

We set the number of individual machine to $T = 13$ for our experiment.

For combining the classification results from different individual machine, various fusion rules can be applied. Here we consider mainly sum rule and majority vote.

Majority vote fusion rule is relatively intuitive. Each model votes a predication to a query image. And classification is done by assigning the class that has the highest number of votes.

Sum rule, on the other hand, is not as intuitive as majority vote. It needs the posterior probability of: given a data point x_k, the probability that it belongs to class c_i. This probability is denoted by:

$$P(c_i|x_k) \tag{19}$$

For a given data point, each individual machine will give posterior probability corresponding to every class: $P_t(c_i|x_k),\ t \in [1, T]$. The sum rule yields that:

$$P(c_i|x_k) = \sum_{t=1}^{T} P_t(c_i|x_k) \tag{20}$$

Then the data point is assigned to the class for which the posterior probability is the highest, which means that the data point belongs most likely to that class.

For computing of posterior probability, the idea is to use the mean value to represent a class, then to project all the classes and data point on the pre-computed feature space. On the feature space, the data point is assigned to the class that is most aligned:[2]

$$P(c_i|x_k) = \frac{1 + \frac{(w_k^x)^T (w_k^i)}{\|w_k^x\| \|w_k^i\|}}{2} \tag{21}$$

where

$$w_k^i = W_k^T m_i \tag{22}$$

$$w_k^x = W_k^T x \tag{23}$$

We then implement the algorithm. And get the comparison between Ensemble machine with majority vote, ensemble machine with sum rule, an average of the individual model.

Fig. 14 shows the recognition accuracy by these three methods with M_{LDA} varying from 1 to 51 (as we see before, the varying of M_{LDA} has a more significant effect on recognition accuracy.

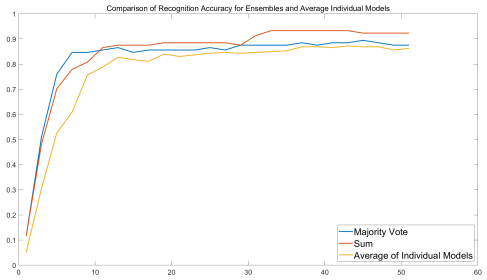

Fig. 14. Comparison of recognition accuracy

As can be seen in Fig. 14, the two ensemble models outperform individual model by 2-5 per cent.

In terms of fusion rule, sum rule is giving better results in general. By taking the average posterior probability of each class, more information is preserved using sum rule than that of majority voting, which might force the individual machine to give a not so reliable prediction.

We further compare the error of the ensemble machine and individual machine. Error values are estimated empirically from the non-diagonal values from the confusion matrix. It can also be viewed as one minus accuracy.

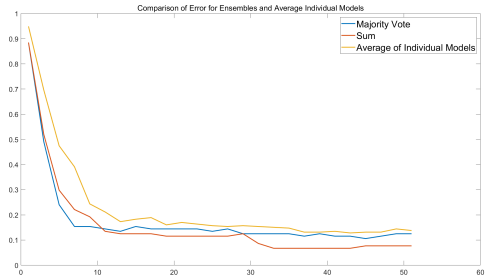

Fig. 15. Comparison of recognition error

To further investigate on the trade-off between randomness and optimality of the model, we performed ensemble machine with sum rule and majority rule with $M1$ set to 5, 10, 50, 100, 200 respectively.

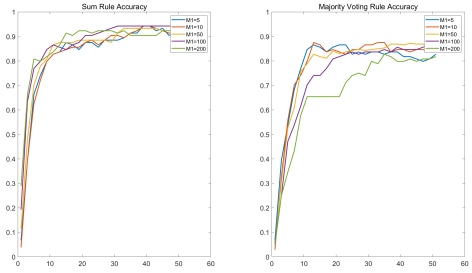

Fig. 16. Majority Voting vs Sum with Varying M1

The first conclusion we can make is that sum rule is a better fusion method. By using majority voting, over-fitting phenomena are still not fully resolved. Furthermore, as shown in Fig. 16, sum rule accuracy graphs do not fluctuate that much compared to majority voting.

We then plot the confusion matrix for each method shown from Fig. 17 to Fig. 19:

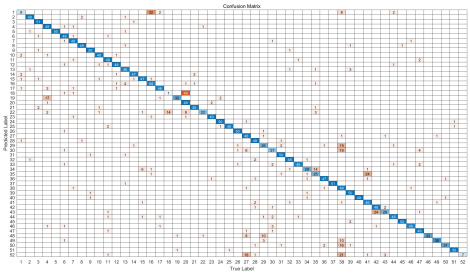

Fig. 17. Confusion matrix for ensemble machine with majority vote

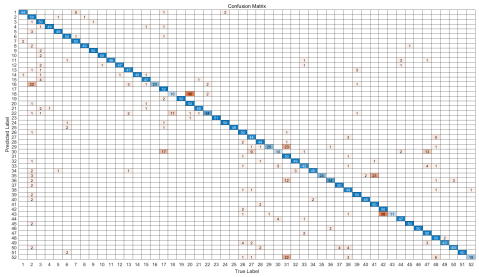

Fig. 18. Confusion matrix for ensemble machine with sum rule

We can conclude that the ensemble machine, by correctly compromising between randomness and optimization of each individual model, can outperform the individual model in terms of recognition accuracy. Also, different fusion rules for model combination will affect the overall accuracy.

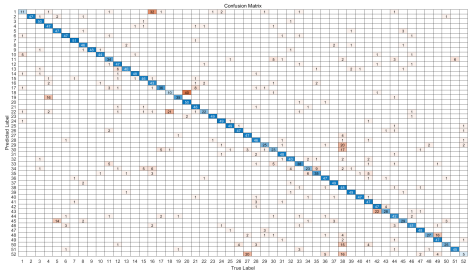

Fig. 19. Confusion matrix for average individual model

REFERENCES

1 Peter Hall, David Marshall and Ralph Martin, *Merging and Splitting Eigenspace Models*

2 XiaoGang Wang and XiaoOu Tang,*Random Sampling for Subspace Face Recognition*

3 Zhou Wang, Alan C.Bovik, Hamid R.sheikh, Eero P Simoncelli, *Image Quality Assessment: From Error Visibility to Structural Similarity*

4 Rui Huang, Qingshan Liu, Hanqing Lu, Songde Ma, *Solving the Small Sample Size Problem of LDA*

5 Li-Den Chen, Hong-Yuan Mark Liao, Ming-Tat Ko, Ja-Chen Lin, Gwo-Jong Yu, *A new LDA-based face recognition system which can solve the small sample size problem*

APPENDIX

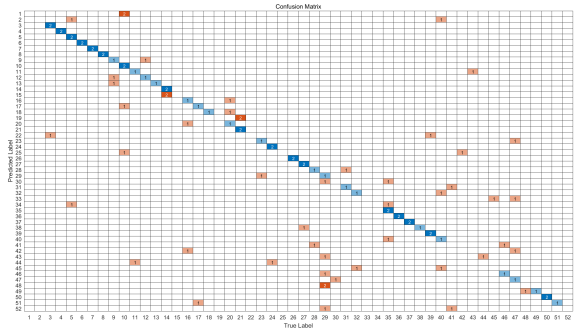

Fig. 20. Confusion Matrix for PCA with 10 Principle Components Selected

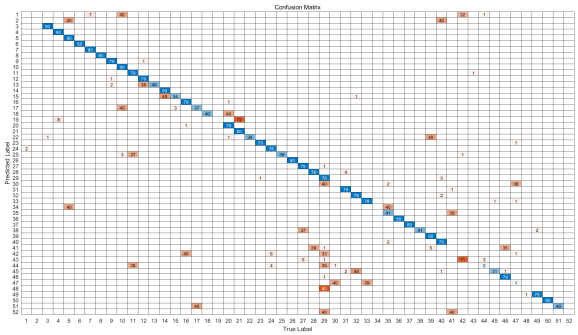

Fig. 21. Overall Confusion Matrix for PCA with Varying Principle Components Chosen

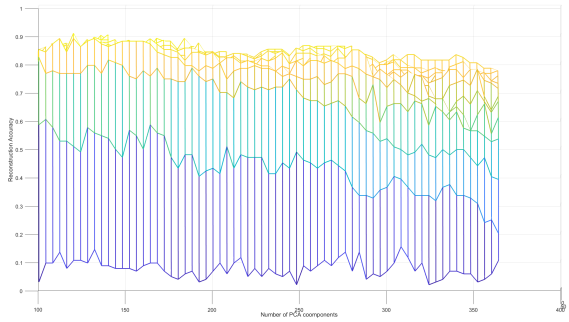

Fig. 22. Success Rate wrt M_{PCA} in PCA-DLA

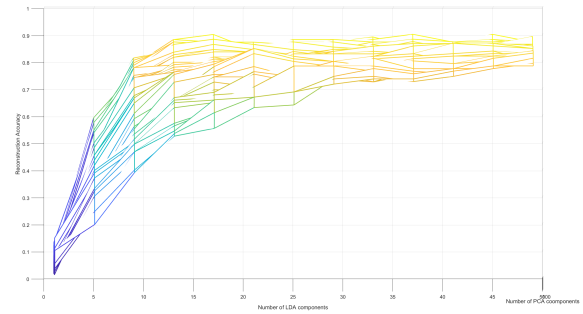

Fig. 23. Success Rate wrt M_{LDA} in PCA-DLA