CS485 Machine Learning for Computer Vision

Jaehan Jeong
20233933
fortes@kaist.ac.kr

Minhajur Rahman Chowdhury Mahim
20210753
minhaj@kaist.ac.kr

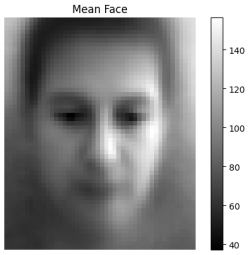

Figure 1. Meanface of face.mat

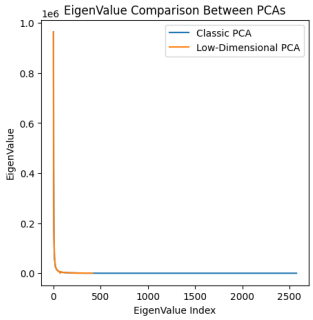

Figure 2. Same eigenvalue observation from different PCAs

1. Experiment with Face Dataset

To ensure unbiased model generalization, the dataset is randomly split into training (416 images) and testing sets (104 images) in a 8:2 ratio while maintaining class distribution. The only preprocessing is normalization, done by subtracting the mean face image from each training image.

2. Eigenface Learning and Its Application

The study uses Principal Component Analysis (PCA) to reduce data dimensions, aiming to retain maximum data variance. This approach is tested by reconstructing the data to its original dimensions and analyzing the results.

2.1. The Classic PCA

Our calculation confirms the number of non-zero eigenvalues to be $415 = N_{train} - 1 \leq D - N_{train}$, complying with the theoretical estimation [1]. Anyway, classic PCA takes time and memory to operate since it works on a very large 2576×2576 matrix as the number of features per image is 2576. Our configuration roughly took 21.71 seconds to perform this PCA.

2.2. The Low-Dimensional PCA

The low-dimensional PCA [1] is a more useful technique especially for the cases when $D >> N$. Compared to the classic one, the matrix size is, therefore, reduced to 416. Figure 2 shows the eigenvalue similarity between these different techniques of PCA. Our configuration took around 0.18 seconds to process the low-dimensional PCA.

2.3. Comparison Between The Techniques

Clearly, the low-dimensional PCA has the advantage in terms of processing time. It is also memory efficient as it works on much lower dimensions. The low-dimensional PCA performs singular value decomposition for projection, which is numerically more stable than direct eigenvalue decomposition of covariance matrix. However, it has less interpretability compared to classic PCA for understanding feature relationship. Moreover, it loses its computational advantage when $D << N$. However, our dataset has the exact opposite case, and takes significantly lesser time. Therefore, we will use the low-dimensional techniques for the next sections.

2.4. Face Image Reconstruction

The reconstructed image of an image x will be $x_{recons} = x_{mean} + Uw_m$ [1], where x_{mean} is the mean face of training set, and U is a projection matrix ($D \times M$) made up of M eigenfaces. Figure 3 shows a sample of our results.

2.5. PCA-based Face Recognition

We performed PCA-based face recognition on test set using NN classification [1] while varying the number of eigenfaces, and show the results in Table: 1. We see that the accuracy stay around in 60% for larger number of eigenfaces. This is because our dataset has varying face pose and lighting. Besides, there are some face images with eyes open

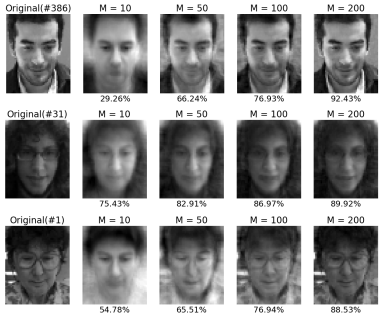

Figure 3. Comparison of reconstruction accuracy using M eigenfaces, with images from the training set (first rows) and the test set (bottom two rows)

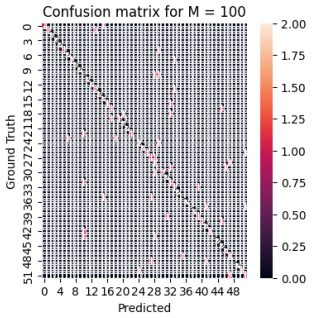

Figure 4. Confusion Matrix for PCA with 100 Eigenfaces

Figure 5. Correct/Wrong Face Recognition by PCA

and closed. PCA is very sensitive to these kind of variations. Considering the speed and accuracy trade-off, we believe choosing top 100 eigenfaces is the best case for face recognition.

Eigenfaces	Time (s)	Accuracy (%)	Memory (KB)
M = 10	5.078	34.62	1489.25
M = 50	12.755	59.62	2294.25
M = 100	19.677	62.50	3300.50
M == 200	38.158	63.46	5313.00

Table 1. PCA-based Face Recognition Comparison

3. Incremental PCA

To illustrate the incremental PCA, we further randomly divide the existing training-test partition into 4 subsets, containing 104 images each. We experiment with different sets of eigenfaces and show the results in Figure 6.

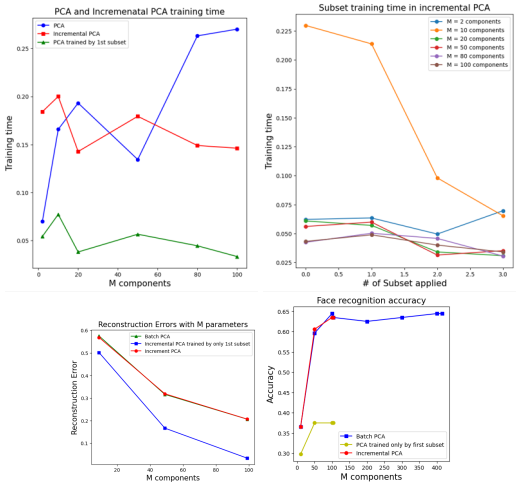

Figure 6. Top-left: PCA and Incremental PCA Training Time by Components, top-right: Subset Training Time incremental PCA for different number of components, bottom-left: Reconstruction Error by Components, bottom-right: Face Recognition Accuracy by Components

3.1. Discussion on Incremental PCA

The accuracy of the incremental method depends on several factors such as batch size, number of principal components etc. Smaller batch size is computationally faster but leads to higher approximation errors. On the other hand, larger batch capture more variance thus, accuracy but requires more time and memory. Therefore, it is more of a trade-off between choices. As aforementioned, batch size and the principal components count are the most important parameter as our results show that within some range, more number of components provide better recognition accuracy.

4. LDA Ensemble for Face Recognition

4.1. PCA-LDA

Here, we first implement PCA-LDA based face recognition and, then we build an ensemble of PCA-LDA models and analyze the results. We get rank(S_B) = 51 and rank(S_W) = 364, complying with the theory, where they denote the rank of between-class matrix and within-class matrix respectively. Figure 7 lets us understand that $20 \geq M_{lda} \geq 50$ leads to decreased recognition accuracy. We find the combination of $M_{pca} = 167$ and $M_{lda} = 47$ giving the best accuracy of 87.5%, which is a significant improvement from 2.5.

4.2. PCA-LDA Ensemble

We ensemble up to 100 PCA-LDA base models and train them by bagging. Figure 10 summarizes our results. We conclude that 50 base models with the randomized param-

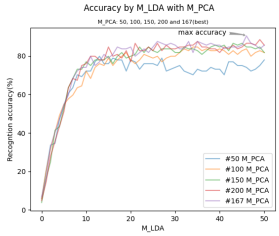

Figure 7. Recognition accuracy with Varying M_{pca}, M_{lda}

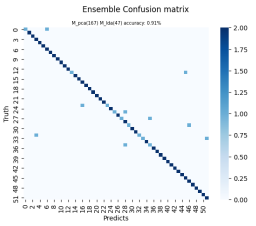

Figure 8. Confusion matrix for PCA-LDA

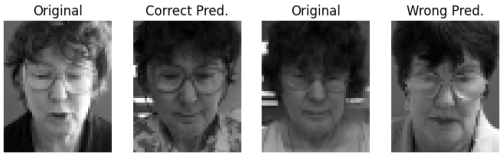

Figure 9. Correct/Wrong Face Recognition by PCA-LDA

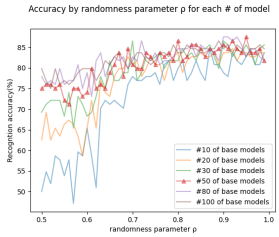

Figure 10. Accuracy by Randomness Parameter For Each Ensemble

eter $\rho = 0.95$ gives the best performance accuracy of 91%. So each model was trained with almost the same training set with little deviation. We then compare the error of the committee machine and the average error of individual base model. Figure 11 shows our results, and we confirm the boundary $E_{com} \leq E_{avg}$ [1] .

We finally perform random sampling on feature space while experimenting with different M_0 and M_1 from M_{pca} and M_{lda}. We get the best accuracy of 93.28% when we chose $M_{pca} = 171(M_0 = 60, M_1 = 111)$, and $M_{lda} = 47$ while setting the randomization parameter, $\rho = 0.95$ with 5 base models. We observe from Figure 12 that the more we randomly choose insignificant eigenvectors (M_1) the more the accuracy goes down. Our understanding is that

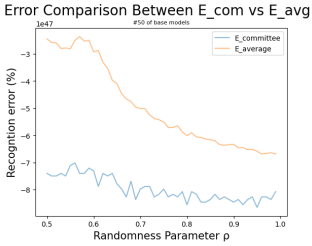

Figure 11. Comparision of E_{com} and E_{avg} for PCA-LDA Ensemble

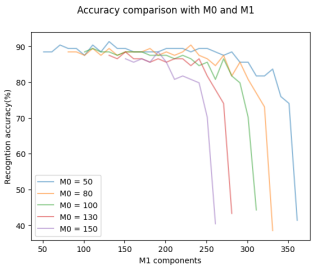

Figure 12. Accuracy Comparison with M_0 and M_1

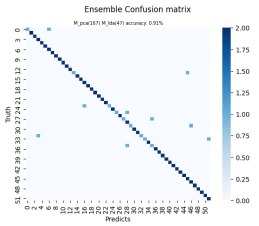

Figure 13. Confusion Matrix for PCA-LDA Ensemble

those unordered sequence of eigenvectors bring noise to the reconstruction and thus, hinders recognition.

In our whole ensemble analysis we chose majority voting as our fusion rule since NN classifier only outputs labels.

5. Generative and Discriminative Subspace Learning

Given a dataset $X = x_1, x_2, ..., x_n$ where $x_i \in \mathbb{R}^{\mathbb{D}}$, we aim to find a projection matrix $W \in \mathbb{R}^{\mathbb{D} \times \mathbb{K}}$ that projects the data onto a lower-dimensional subspace K. Our objective function should combine the PCA and LDA criteria. For PCA, we want to minimize the reconstruction error. The reconstructed data becomes WW^TX, and therefore, the objective function [1] from PCA is:

$$J_{pca} = \min_{\mathrm{W}} \|X - WW^TX\|_F^2$$

On the other hand, for LDA, we aim to maximize the between-class scatter while also minimize the within-scatter. Therefore,

$$J_{lda} = \max_W \frac{\text{tr}(W^T S_B W)}{\text{tr}(W^T S_W W)}$$

where S_b and S_W are between-class and within-class scatter matrices respectively [1].

We introduce a balancing parameter α that will control contribution from both subspace learning. Our objective function becomes:

$$J_{obj} = \min_W (\alpha \|X - WW^T X\|_F^2 - (1-\alpha)\frac{\text{tr}(W^T S_B W)}{\text{tr}(W^T S_W W)})$$

The minus sign next to the LDA criterion in the objective function enables the whole expression of the function under the W minimization objective. Our constraint is that W should be orthonormal, so $W^T W = I$. So, we get the Lagrange:

$$L(W, \Lambda) = \min_W \Big(\alpha \|X - WW^T X\|_F^2 - (1-\alpha)\frac{\text{tr}(W^T S_B W)}{\text{tr}(W^T S_W W)} \Big) - \text{tr}(\Lambda(W^T W - I))$$

where Λ represents a diagonal matrix of Lagrange multipliers$(\lambda_1, \lambda_2, ..., \lambda_K)$. Solving for $\frac{\partial L}{\partial L} = 0$ with a little help from linear algebra gives a generalized eigenvalue problem:

$$(S_W + \alpha(S_B + S_W))W = \Lambda W$$

The diagonal values of the matrix Λ has the eigenvalues of W in descending order while each column of W has the corresponding eigenvectors.

5.1. Discussion

As α reaches 1, the model will behave more like PCA, thus focusing on reconstruction. On the other hand, if it reaches 0, the model will behave more like LDA, focusing on class-wise discrimination. In that case, the model may overfit the training data. However, α offers the flexibility control between reconstruction and discrimination. As a result, we can achieve the combined strength of both PCA and LDA. For the same reason, this model will be computationally costly.

6. Random Forest Classifier

We train the Random Forest using the scikit-learn [2] library while experimenting with the variation of number of trees, maximum depth, minimum samples for splitting a node, and the maximum number of features for split decision. We confirm the significant effect of hyperparameters in Random Forest with the results in Figure: 14.

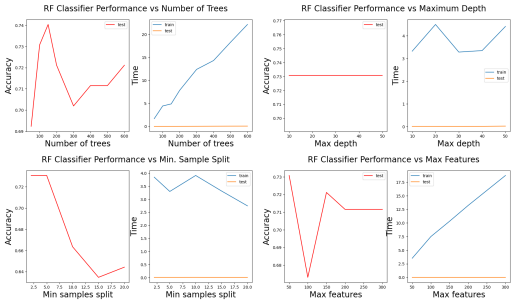

Figure 14. Random Forest Performance With Different Hyperparameters

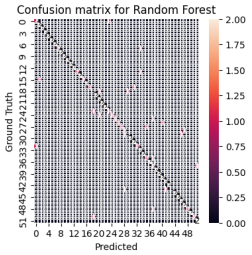

Figure 15. Confusion Matrix for Random Forest

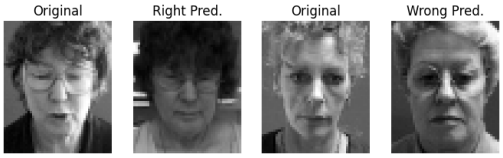

Figure 16. Correct/Wrong Face Recognition by Random Forest

6.1. Discussion

We see that the training time for random forest almost linearly increase with the increase of number of trees and maximum features. On the other hand, the testing time is largely insignificant for any kind of setting. Interestingly, the performance accuracy does not depend on the maximum depth of individual trees while other hyperparameters are kept constant. The reason could be that shallower trees are sufficient to build on our dataset. Again, the minimum sample for splitting underfits the model if it is greater than 5. We also observe sticking around 200 trees give better performance. Overall, we choose 200 trees, minimum sample split = 5, maximum number of features = 100, to get the best accuracy of 79%. The model took 22.17 seconds to train.

Compared to 2.5 and 4, the performance is low even though the time it takes for training is reasonable.

References

[1] Tae-Kyun Kim. KAIST CS485: Machine Learning for Computer Vision Lectures. 1, 3, 4

[2] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 4, 5

Appendix

Parameter	Default Value
bootstrap	True
ccp_alpha	0.0
class_weight	None
criterion	entropy
max_depth	None
max_features	auto
max_leaf_nodes	None
max_samples	None
min_impurity_decrease	0.0
min_samples_leaf	2
min_samples_split	2
min_weight_fraction_leaf	0.0
n_estimators	100
n_jobs	None
oob_score	False
random_state	42
verbose	0
warm_start	False

Table 2. Default Parameters of Random Forest Classifier [2] If Not Changed

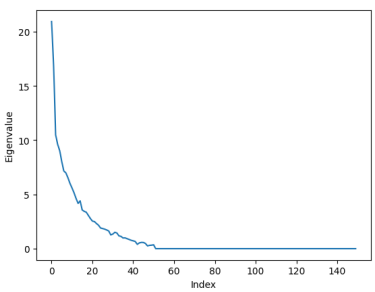

Figure 17. Eigenvalues Achieve from LDA

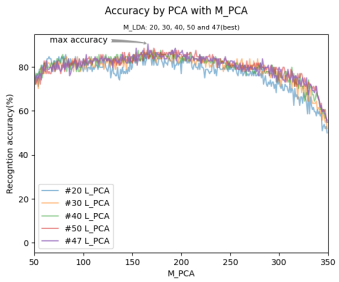

Figure 18. Ensemble Accuracy by M_{pca}

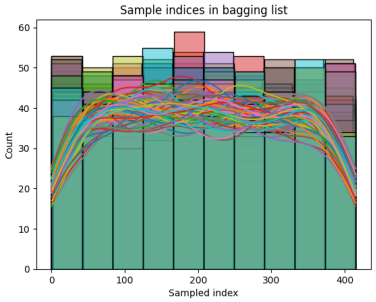

Figure 19. Bagging Count in Ensemble