MLCV Coursework 1 Report

Shinyoung Yu
20200405
School of Computing
sahaa0918@kaist.ac.kr

Jiyun Park
20210263
School of Computing
jiyun02@kaist.ac.kr

1. Eigenfaces

Please follow the steps outlined below when submitting your manuscript to the IEEE Computer Society Press. This style guide now has several important modifications (for example, you are no longer warned against the use of sticky tape to attach your artwork to the paper), so all authors should read this new version.

1.1. Eigenfaces

All manuscripts must be in English.

1.2. Application of Eigenfaces

Please refer to the author guidelines on the web page for a discussion of the policy on dual submissions.

2. Incremental PCA

2.1. Important parameter in implementation: d_3

To implement incremental PCA, we utilized the algorithm from the "Online Learning" slides presented in class. Here, a key parameter is d_2 and d_3. When new data arrives in incremental PCA, computing the eigenspace model for this subset requires $O(\min(D, N')^3)$ time, where N' is the number of data points in the subset. Additionally, merging this new eigenspace model with the existing data takes $O((d_1 + d_2 + 1)^3)$ time, where d_1 is equal to the previously computed eigenspace model's d_3 value. Therefore, to enhance time efficiency in incremental PCA, it is essential to keep d_3 small, although this results in a time-accuracy tradeoff by dropping less-significant eigenvector information, which is represented by our experiment result Fig. 1

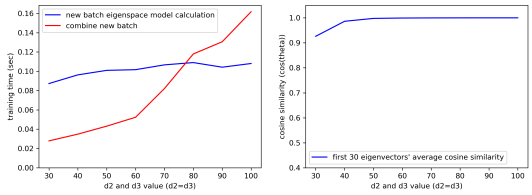
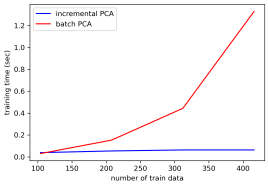


Figure 1. Incremental PCA's time-accuracy tradeoff according to the value of d_3

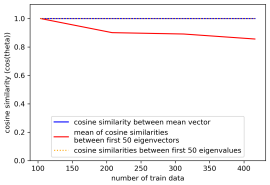
2.2. Comparison with other PCA

We compared the results of each incremental PCA stage (i.e.adding training data in four batches) with the results of batch PCA in the following four aspects. In summary, incremental PCA is a good approximation of batch PCA, and even requires less training time.

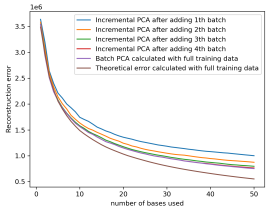
- Training time: For batch PCA, we measured training time by re-training the model each time new data was added. The results are shown in Fig. 2a. Using one subset, the training time is approximately the same for both methods. However, as we add more data, the value of N increases for batch PCA, while the values of N' and d_3 remain constant for incremental PCA, maintaining constant time and improving time-wise efficiency.
- Accuracy of incremental PCA: In incremental PCA, time-accuracy tradeoff occurs since less-significant eigenvectors are dropped during $d_1 + d_2$ merging, retaining only the top d_3 eigenvectors. We calculated the cosine similarity of eigenvectors, eigenvalues, and mean vectors between incremental PCA at each stage and batch PCA calculated by corresponding data, as shown in Fig. 2b. As more training data is added, the number of discarded less-significant eigenvectors increases, resulting in decreased similarity between eigenvectors; the similarity after adding the last subset is 0.856. For mean vectors and eigenvalues, cosine similarities are 1 and close to 1 respectively, indicating that the incremental PCA is a good approximation.
- Reconstruction error: Referring to Fig. 2c, the reconstruction error for incremental PCA using all training data (i.e., after adding the 4th batch) is almost identical to that of batch PCA with the same amount of data. This also demonstrates that incremental PCA gives similar result to batch PCA.
- Face recognition accuracy: In the optimal settings of batch PCA found in Question1 (i.e. K=1 and bases=90) the accuracy ranks as follows: full-data batch PCA > full-



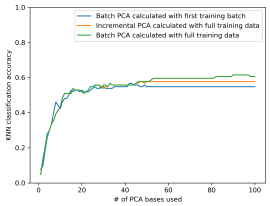
(a) Training time



(b) Cosine similarity of results



(c) Reconstruction error



(d) NN-classification face recognition accuracy ($n\_nearest = 1$)

Figure 2. Comparison between Incremental and Batch PCA methods

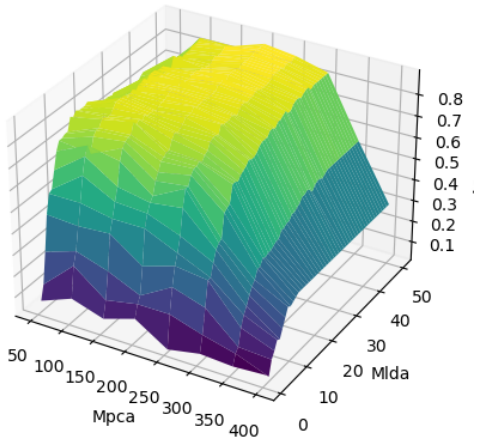


Figure 3. Classification accuracy varying Mpca and Mlda

data incremental PCA > PCA using only the first training set. This indicates that classification accuracy improves progressively with the increase in training data. Additionally, using incremental PCA in place of batch PCA results in an accuracy drop of approximately 6.67%, also indicating a time-accuracy tradeoff.

3. LDA Ensemble for Face Recognition

PCA can effectively reduce the dimension of input data preserving important features. And LDA can maximize the variance of between-class while minimize between-class. Thus, using PCA-LDA is expected to increase computing efficiency and classification accuracy. In this section, we try to figure out the effect of PCA-LDA via some experiments.

3.1. Recognition accuracy of PCA-LDA

To implement PCA-LDA, we have to set M_{pca} and M_{lda} to determine projection dimensions of each of PCA and LDA. For best performance of the PCA-LDA model, we measure the accuracy of classification varying M_{pca} from 1 to 415 and M_{lda} from 1 to $min(M_{pca} - 1, 51)$. This is because, the maximum possible projection dimension for PCA is $(the\ total\ number\ of\ data) - 1$ since the total number of principal components can not be larger than overall data, and for LDA is $(the\ total\ number\ of\ classes) - 1$ since the number of direction for maximizing the distance of between-class anc minimizing within-class cannot be larger than the total number of classes. As we can see in Fig. 3, the accuracy was highest when $M_{pca} = 150$ and $M_{lda} = 50$ and decreases further away from this point. To be specific, the larger M_{lda}, the better performance. This is because, large M_{lda} helps to get more discriminative data. And for M_{pca}, the performance is highest when $M_{pca} = 100 - 200$. This is because, values smaller than this may ignore too much important information while values larger than this risk overfitting.

In addition, for LDA, the rank of within-class scatter matrix(S_w) is $min(364, M_{pca})$ where $N - n_{class} = 416 - 52 = 364$ and the rank of between-class scatter matrix(S_b) is $n_{class} - 1 = 51$. For the former, it is because, since $\sum_{x \in D_i}(x - m_i) = 0$, thus each class is linearly dependent, so $S_w = \sum_{i=1}^{c} \sum_{x \in D_i}(x - m_i)(x - m_i)^T$, can have at most $N - n_{class}$ linearly independent row vector. If we reduce the dimension using PCA, the rank of S_w can not exceed M_{pca} since vectors are placed in PCA projection space. Next for the latter, since $S_b = \sum_{i=1}^{c}(m_i - m)(m_i - m)^T$, it only affected by class mean. Because the relationship of class mean does not change after PCA projection since it is linear transformation, and for the same linearly dependent relation as S_w, the maximum possible value for S_b is $N - n_{class}$

Based on this observation, we decided to fix $M_{pca} = 150$ and $M_{lda} = 50$ for further experiments.

3.2. Result of PCA-LDA

Fig. 4 is the confusion matrix of PCA-LDA classification result. As most of prediction result is on the diagonal entry, it indicates that most of prediction is successful. We take a closer look at success and failure cases. Fig. 5 shows successfully predicted cases. Despite the different angles of the faces, the model infer the class accurately. Fig. 6 is failure cases. It seems that the prediction failed because of the similar glasses and face expression.

3.3. Time and Memory

Comparison btw pca/pca-lda (accuracy), lda/pca-lda(time), pca/lda/pca-lda(memory)

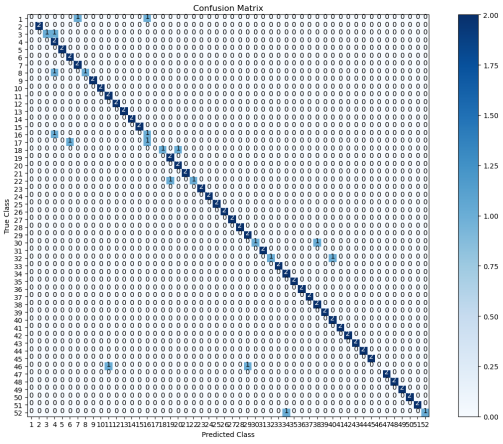


Figure 4. Classification accuracy varying Mpca and Mlda

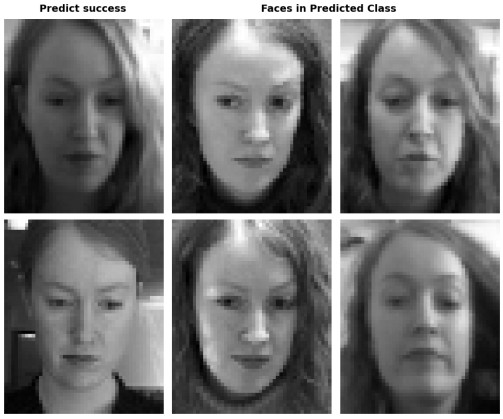


Figure 5. Classification accuracy varying Mpca and Mlda

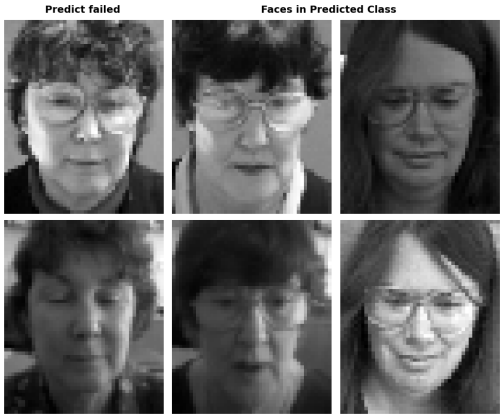


Figure 6. Classification accuracy varying Mpca and Mlda

3.4. PCA-LDA Ensemble

For PCA-LDA Ensemble model, we combined two different types of models. The first type is randomization in feature space, which select vectors for pca projection randomly by a certain percentage. The second type is randomization in data sampling, which randomly subsampling the train data by a certain percentage. Both models were used in equal numbers. For combining prediction results of each models, we used 'majority voting' among various fusion rules. This is because, since our task is predicting class for classification and each classes don't have special meaning in numeric value, majority voting looks the most reasonable compared to other methods like averaging and finding maximum.

3.5. Randomization

randomization in feature space (m0) randomization in data samples (subset_rate) randomization in model number (model_num) randomness parameter

3.6. Result of PCA-LDA Ensemble

error (committee machine, individual models) accuracy, confusion matrix

4. Generative and Discriminative Subspace Learning

Please follow the steps outlined below when submitting your manuscript to the IEEE Computer Society Press. This style guide now has several important modifications (for example, you are no longer warned against the use of sticky tape to attach your artwork to the paper), so all authors should read this new version.

4.1. lalalala

All manuscripts must be in English.

4.2. lalalala

Please refer to the author guidelines on the web page for a discussion of the policy on dual submissions.

5. RF classifier

The random forest model has several key parameters, and testing them all simultaneously would require excessive effort and time. Therefore, we configured the optimal parameters in the sequence described below. Also, to ensure result stability, each test was executed 10 times, with the average result displayed in a graph, and the confusion matrix representing the best result among the 10 runs is recorded in Appendix.

1. Number of Trees & Tree Depth: We tested using an axis-aligned weak learner with the split number fixed at 10.
2. Split Number (Randomness Parameter): Using the optimal number of trees and tree depth derived in step 1, with an axis-aligned weak learner.

3. Type of Weak Learner: With the optimal values for other parameters determined in steps 1 and 2, we tested different weak learners.

5.1. Number of trees & The depth of trees

We varied the number of trees and their depths, obtaining the results shown in Fig. 7a. In the graph, the best accuracy 0.625 occurred at $N(numberoftrees) = 250$ and $D(depth) = 10$, explained as follows:

- Number of Trees: A single tree in random forests tends to overfit to data; therefore, we can generalize the model using ensembles of trees. As shown in Fig. 7a, the accuracy converges near $N = 250$, which is selected as optimal parameter.
- Tree Depth: When depth is D, maximum of $2^D - 1$ nodes are generated. As shown in Fig. 7a, for a given N, accuracy initially increases with tree depth but later decreases due to overfitting. Notably, the optimal depth D increases with N, indicating that a larger N result in less overfitting. Also, we chose $D = 8$ as the later experiment's standard value, accuracy of 0.610, which is sufficiently high.

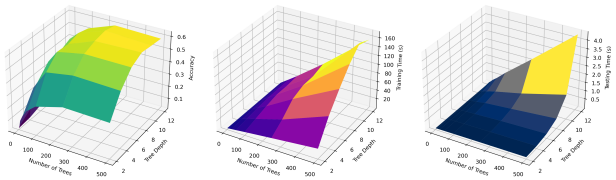
The theoretical training/testing time when adjusting the number of trees and tree depth is as follows, and our testing time results align with theoretical predictions Fig. 7b, Fig. 7c. However, due to the small size of the training data, some splits stop prematurely, not reaching the maximum tree depth, resulting in less training time than theory.

- Number of Trees N: $O(N)$ time
- Tree Depth D: $O(2^D)$ time

Moreover, our code do not utilize parallelization of each tree's growth. Based on theoretical insights from course materials, training each tree in parallel could reduce the time for the tree numbers to $O(1)$, not $O(N)$.

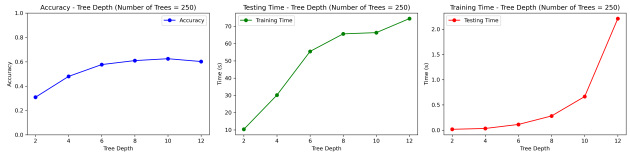
5.2. Randomness parameter

Due to the high dimension image data, we randomly select dimensions and threshold values to create the split function. For each split function, we attempt ρ random splits and choose the one with the highest information gain. There is a trade-off in the magnitude of ρ value: If the ρ value is too low, fewer splits are attempted, which reduces similarity between trees but increases the risk of the less-optimal split. Conversely, as ρ increases, more various split functions are tested, leading to greater similarity among trees, which decreases the advantage of ensemble multiple trees. As shown in Fig. 8a, accuracy initially increases with higher ρ values until $\rho = 10$ and then decreases, as expected. In Fig. 8b, training time is same as expected, $O(\rho)$. However, since ρ does not affect the resulting tree structure in training, it has constant testing time.



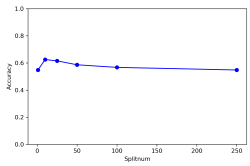
(a) Test accuracy, Training time, Testing time according to the number of tree and the depth of tree

(b) Test accuracy, Training time, Testing time according to the number of tree (D = 8)

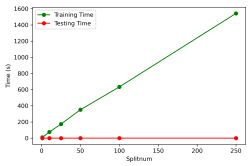


(c) Test accuracy, Training time, Testing time according to the depth of tree (N = 250)

Figure 7. Test accuracy, training/testing time according to the number and depth of trees



(a) Test accuracy according to the randomness parameter



(b) Training time, Testing time according to the randomness parameter

Figure 8. Test accuracy, training/testing time according to the randomness parameter

5.3. Impact of the different types of weak-learners

The impact of the weak learner are shown in Fig. 9. As expected, the axis-aligned method provides sufficiently good testing accuracy along with the shorter training and testing times. However, even with all other parameters being the same, the accuracy when using the two-pixel test is approximately 8.32% higher. Therefore, the two-points test's information gain per split is greater than that of the axis-aligned method. Although other weak learners, such as linear and non-linear methods, were also implemented and tested, they took so long to measure training and testing times that it was expected impractical for real-time training/testing, and thus they were not included in the report.

Figure 9. Axis-aligned vs Two-pixel test

5.4. Confusion matrix, Example success/failures, Information gain histogram

The main results' confusion matrix, example success/failures, example node's histogram representing information gain along splitting are written in Appendix: A.1, A.2

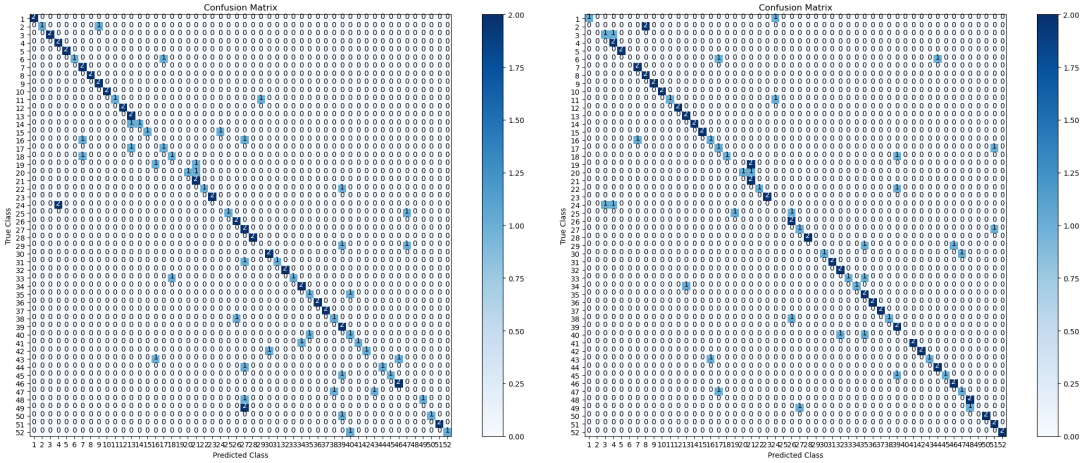
5.5. Comparision with PCA and PCA-LDA method

Since parallel programming was not used our random forest, both the training and testing times are significantly large compared to the PCA and PCA-LDA methods. In addition, for PCA and PCA-LDA, the differences are further amplified due to Python NumPy's parallelization for matrix operations and scikit-learn's optimized KNN classification, which is well-tuned for handling multiple data points efficiently. About the accuracy, because raw pixel values were used directly as inputs, the optimal accuracy of Q1 PCA is almost identical to random forest's result. Overall, the accuracy is much lower compared to the PCA-LDA method, indicating that in situations where the size of the training dataset is small ($N << D$), applying the random forest with no PCA is relatively less suitable. Additionally, similar to the PCA-LDA ensemble, as the number of base models (tree) increases, there is a tendency for accuracy to improve, indicating that the 'ensemble models' is beneficial.

A. Appendix

A.1. Q5: Test result's confusion matrix and success/failure cases

The optimal parameters determined for the random forest through experiments are $N = 250$, $D = 8$, and splitnum $= 10$. With this configuration, each real-time executable weak learner—axis-aligned and two-pixel tests—each was trained 10 times. The confusion matrix results below represent the best test accuracy among the 10 repetitive train results.



(a) Axis-aligned weak learner, test accuracy=0.644 (b) Two-pixel test weak learner, test accuracy=0.692

Figure 10. Confusion matrix of optimal cases ($N = 250$, $D = 8$, and splitnum $= 10$)

Additionally, the example success and failure cases based on the confusion matrix results above are shown below. Compared to the success cases, the failure cases show a higher similarity between the failed image and the predicted class, meaning that our model is reasonable.

A.2. Q5: Visualization of random forest's information gain process

Figure 11. Axis-aligned weak learner: Example success case

Figure 12. Axis-aligned weak learner: Example failure case

Figure 13. Two-pixel test weak learner: Example success case

Figure 14. Two-pixel test weak learner: Example failure case