Homework 2: Face Recognition and Image Classification Algorithms:

Eigenfaces, Fisherfaces, SVM and SRC Algorithms

Akwasi D. Akwaboah

March 10, 2021

Abstract

Face recognition and image classification algorithms pervade various spheres of our everyday lives in an era where we humans have become highly dependent on technology. From our personal computing devices to more pronounced security, tracking and recognition systems. In this report, I present simulation results for four of such machine learning algorithms – Eigenfaces, Fisherfaces, Support Vector Machine (SVM) and Sparse Representation-based Classifier (SRC). Experiments performed focused on determining the role of the number of training samples and the type of extracted feature on recognition rates for the various algorithms.

1 Introduction

Over the years, there has been an unabated need for computer vision algorithms that facilitate image recognition especially human frontal images for the purposes of identification and verification. These algorithms either utilize the original pixel intensities or some form of handcrafted features such as the Local Binary Patterns (LBP), Histogram of Oriented Gradients (HOG). In this homework report, I perform computational experiments with a focus on the effects of the number of training samples per individual (m) and the error rate (E) for four of such classification and recognition algorithms, viz EigenFaces (formulated around Principal Component Analysis), Fisherfaces (formulated around Linear Discriminant Analysis), Support Vector Classification and Sparse Representation-based classfier. The recognition rate with respect to various kinds of features for the later two are explored as well.

2 Dataset

Similar to Homework 1, this study adopted a lightweight preprocessed version of the YaleB dataset. The dataset comprised of 32×32 size gray scale frontal images for 38 individuals/ labels with about 64 images per individual at varying degrees of illumination and pose. Feature types used here include the raw pixel intensities four others that reduced the dimensionality – Principal Component Analysis (PCA) features, Linear Discriminant Analysis (LDA) features, LBP and HOG features. PCA features were extracted using the python's *sklearn.decomposition.PCA()* module. Principal Components (PCs) that account for 95% of variance in data were used throughout this study. LDA features were extracted using python's *sklearn.discriminant_analysis.LinearDiscriminantAnalysis()* module. Here, the maximum allowable feature size (number of classes(C) - 1 = 37) was used

throughout. Visualizations for the various features using a sample image in shown Figure 1. The HOG descriptors were extracted using the python *skimage.feature.hog()* module with default parameters (9 orientations, pixel block size = 8×8), whereas the LBP features were extracted using the python *skimage.feature.local_binary_pattern()* module with 8 neighbors, radius = 8 and all others being default.

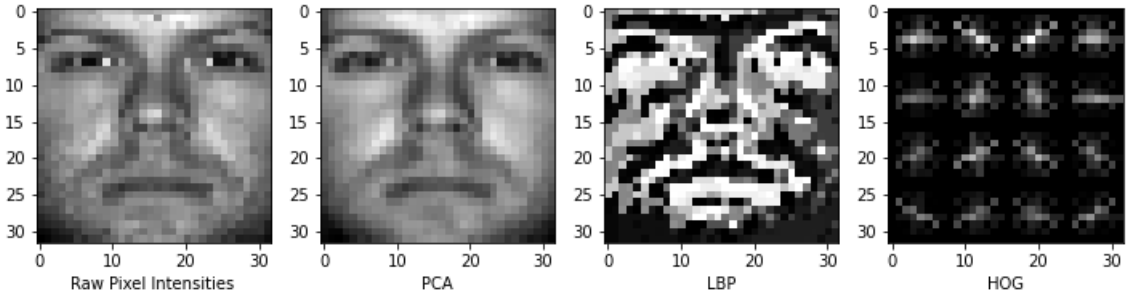

Figure 1: Visualizing features

3 Eigenfaces

This algorithm proposed by [1] leverages the PCA concept to extract the dominant features in sample images. PCs for the training and test data were extracted as described in Dataset section. Simulation experiments for different training and test data sizes defined by m was performed over the range $m = [10, 20, 30, 40, 50]$. In each m iteration, the number of PCs that accounted for 95% variance in the training data was computed for parsimony and has been shown along with classification error rate in Figure 2. A nearest neighbor (NN) classifier with euclidean distance criterion (from the *sklearn.neighbors.KNeighborsClassifier()* module) was used to evaluate the recognition rate on the dimensionality-reduced test data (also at 95% variance). It was observed that the error rate improved with increasing m in line with the large training data requirement of machine learning algorithms.

4 Fisherfaces

First proposed by [2], Fisherfaces employs LDA to maximize the between-class scatter while minimizing the within-class variance and not necessarily projection data onto a hyperplane that maximizes the variance in the projections as PCA blindly does. However, LDA may not sufficiently capture data variance if the original dimensionality of the data is very large relative to the number of classes as is the case here (1024 versus 38). Going by the insight that PCA produces about the number of PCs that account for 95% variance for the various training data size presented in Figure 2, the smallest PC count is 48. This is greater than the largest possible LDA dimensionality of 37; thus, giving an *a priori* indication of possible insufficient recognition. This is evident in the high and anomalous error rate observed in experiments shown in Figure 3.Similarly, a nearest neighbor classifier is used here.

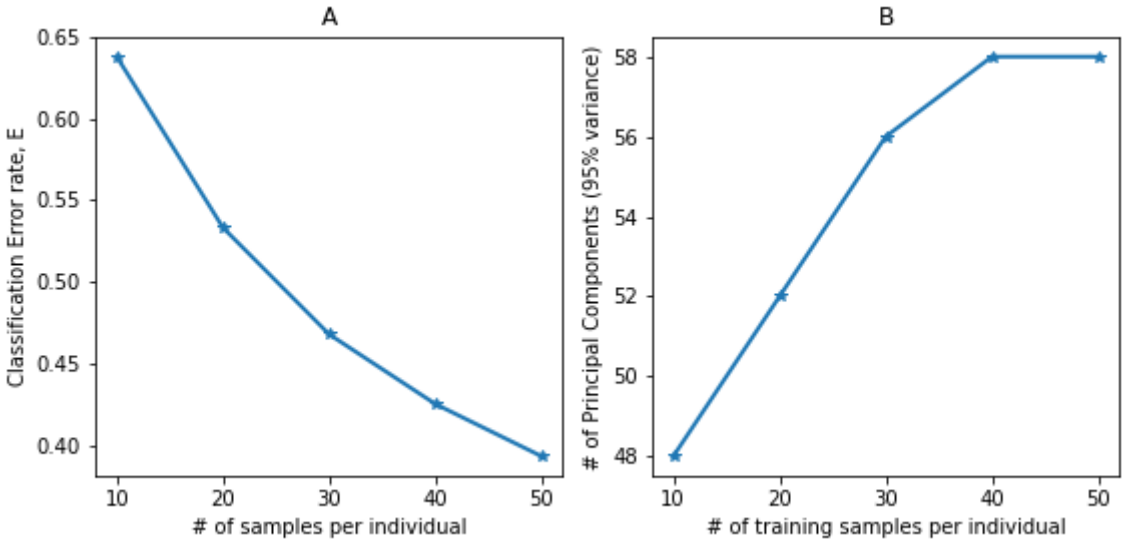

Figure 2: **EigenFaces:** (A) Classification Error rate for the various training data size per individual. (B) Number of PCs that account for 95% variance in the training data

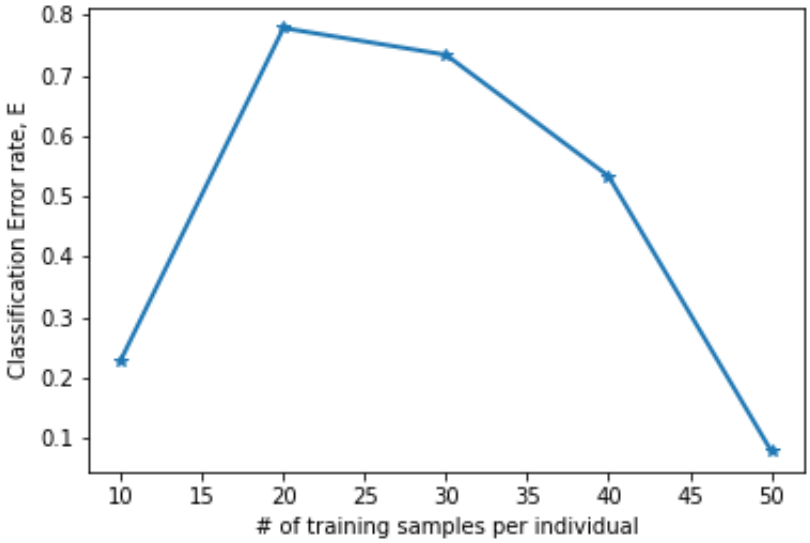

Figure 3: **Fisherfaces:** Classification Error rate for the various training data size per individual.

5 Support Vector Machine (SVM)

A linear SVM classifier [3] imported from python's *sklearn.svm.linearSVC()* module with default parameters (C=1.0) was used. Here the classifier performance on the various feature types was explored. For the SVM classifier on pixel intensities, recognition rate improved as m increased, which is consistent with large training requirement of machine learning algorithms. The SVM, however, performed poorly when PCA and LDA features were used, indicating that both are poor descriptor for this classification approach. It must however, be mentioned that this assertion is not absolute as hyperparameter tuning was not performed. HOG and LBP descriptors, on the other hand, resulted in lower error rates with LBP being optimal. The results for all SVM-feature combinations are presented in Figures 4 and 5

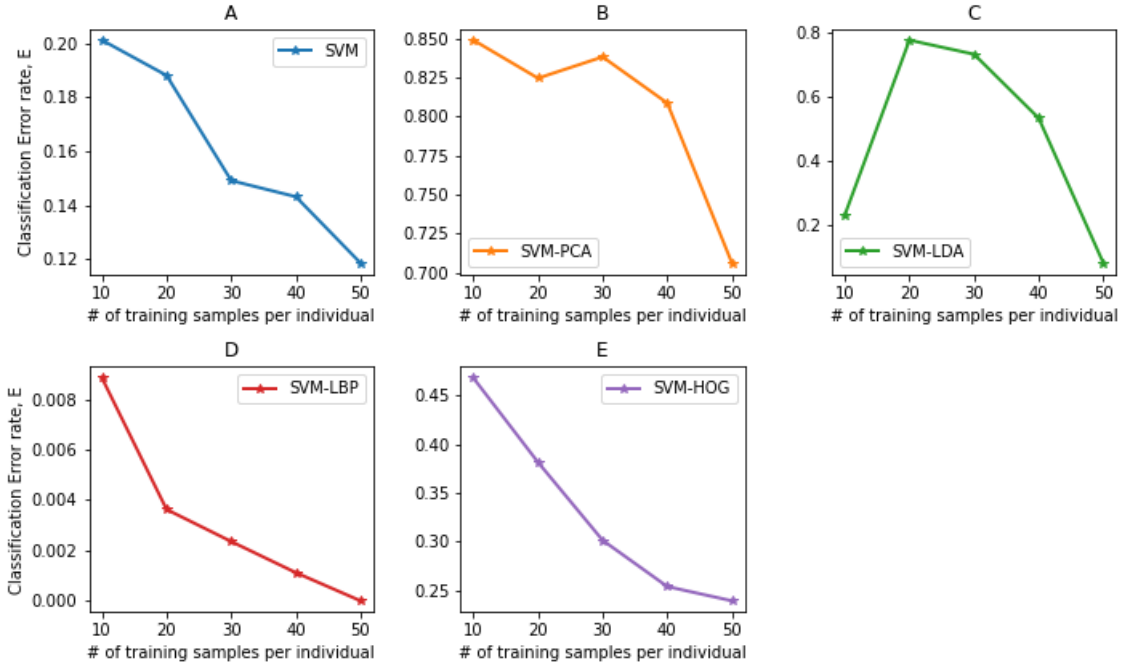

Figure 4: **SVM** Classification Error rate for the various training data size per individual for (A) SVM-Pixel Intensity, (B) SVM-PCA, (C) SVM-LDA, (D) SVM-LBP, (E) SVM-HOG

6 Sparse Representation-based Classifier (SRC)

This algorithm [4] is a dictionary-based approach where a test image is expressed as a linear combination of training data. This method leverages the self-expression property, i.e. sparse coefficients of the dictionary/ training data, with the matching training examples having non-zero coefficients. Test data class prediction, here, is determined by how the computed sparse coefficients reconstruct the test image using a distance metric (ℓ^2-norm in this case). A custom SRC python function was created using the *sklearn.decomposition.sparse_encode()* with an Orthogonal Matching Pursuit

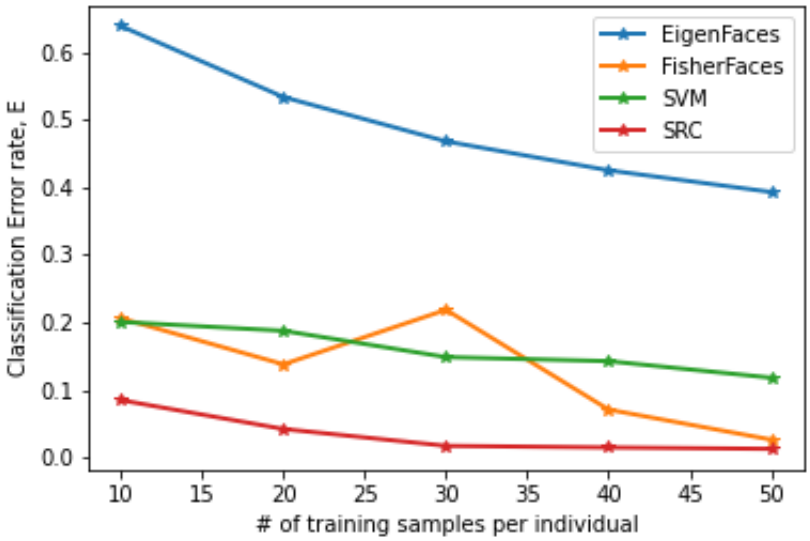

Figure 5: **SVM:** Overlaid Classification Error rate for the various training data size per individual for various SVM classifier-feature combinations.

(OMP) algorithm for sparse coding, i.e. coefficient determination. Similar to the SVM analysis, this classifier was applied on the various feature types. The results for this have been shown in Figures 6 and 7. Throughout all m, the SRC with LBP features produced the lowest error rate and SVM with PCA features recording the highest error rates.

7 All Four Algorithms

The Error rates for all four algorithms with pixel intensity features are presented in Figure 8. Overall, SRC had the best recognition rate (i.e. lowest error rate), whereas EigenFaces performed the worst.

8 Conclusion

The importance of face recognition and image classification algorithms cannot be overemphasized; they provide seamless identification in modern mobile devices and verification systems. In this report, I present four of such algorithms, specifically exploring how their error rate are affected by the size of the training data and feature type. At the end of this exercise, It was observed that the SRC classifier with LBP features had the optimal recognition rates (¡ 10% error rate for all training data sizes used). Code (in python) written for this study has been shared via this **GitHub link** (`https://github.com/Adakwaboah/Face_Recognition`)

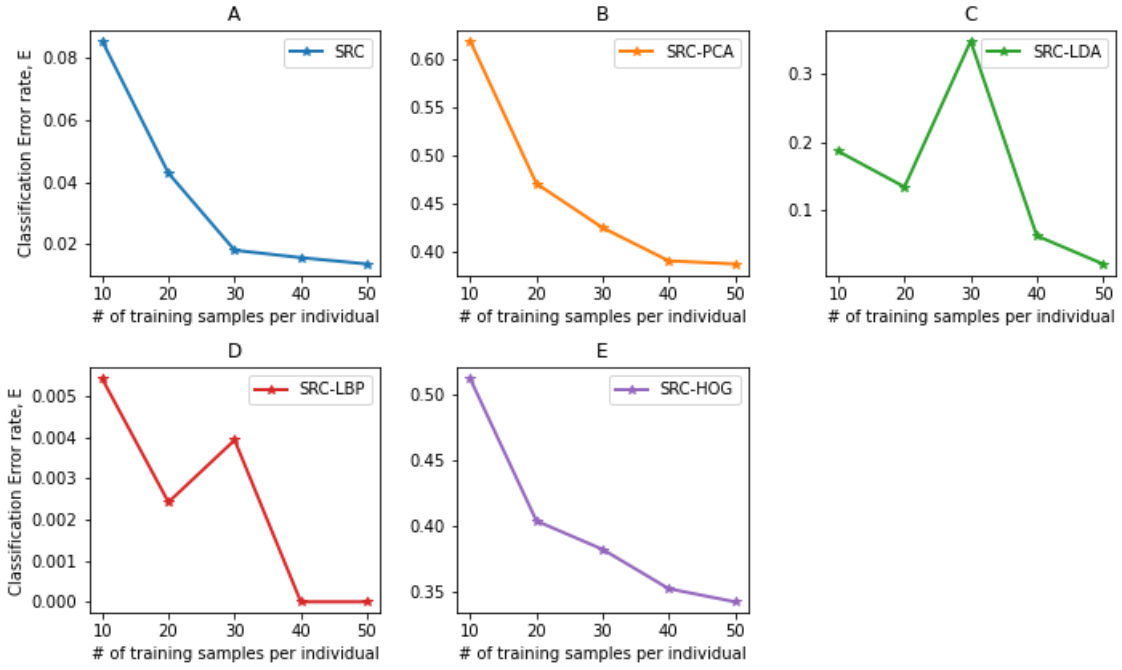

Figure 6: **SRC:** Classification Error rate for the various training data size per individual for (A) SRC-Pixel Intensity, (B) SRC-PCA, (C) SRC-LDA, (D) SRC-LBP, (E) SRC-HOG

References

[1] M. Turk and A. Pentland, "Eigenfaces for recognition," *Journal of cognitive neuroscience*, vol. 3, no. 1, pp. 71–86, 1991.

[2] 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, no. 7, pp. 711–720, 1997.

[3] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[4] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 2, pp. 210–227, 2008.

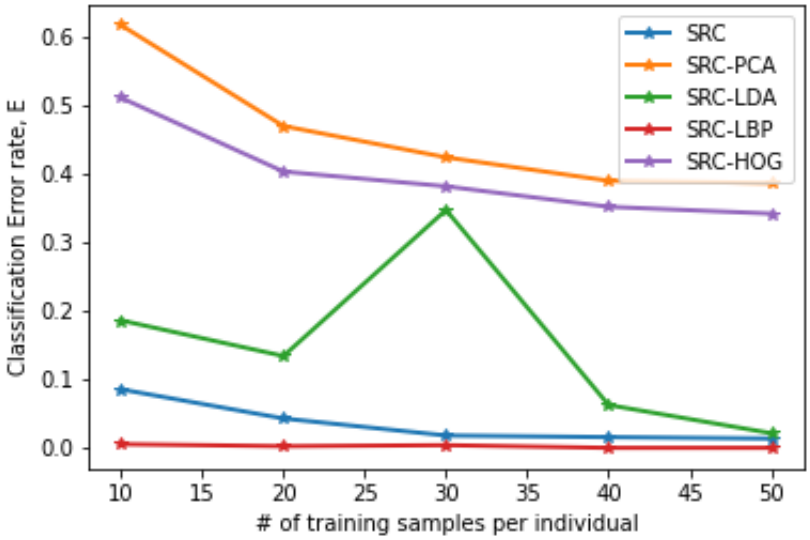

Figure 7: **SRC:** Overlaid Classification Error rate for the various training data size per individual for various SRC classifier-feature combinations.

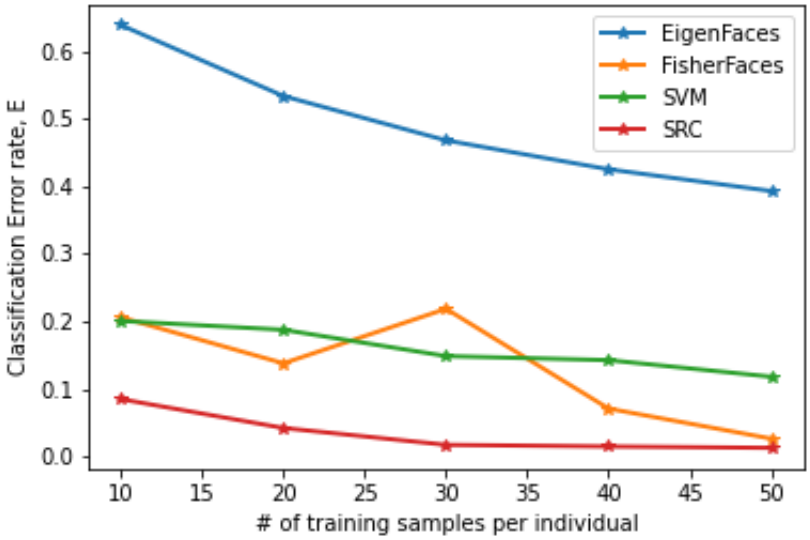

Figure 8: **All Four Algorithms:** Recognition rates over different number of training samples per individuals.