Yagiz Gençer

EPFL

COM500 - Statistical signal and data processing through applications

Eigenfaces - Assignment #2

GitHub link for the code base: github.com/Luturtun/SSDP_project

---

1 Introduction

In the first submission, I first introduced and then implemented PCA as a dimensionality reduction technique. The dataset used was the Extended Yale Face Database B (cropped) [1], which comprises 2414 gray-scale images across 38 subjects. Each subject was represented by approximately 64 images with a pixel resolution of 192x168. Each image was unflattened into a 1x32256 array and then they were stacked as rows, obtaining a 2414x32256 data matrix X. Then, the number of dimensions of each image was reduced from 32,256 to less than 100. In this reduced space, each coefficient had a corresponding eigenvector and it was possible to approximate each image by linearly combining the eigenvectors of PCA with the image's coefficients in the reduced space. Each eigenvector was size of 32,256, allowing us to reshape them into original image size and display them as a set of "Eigenfaces", giving the project its name. To test if the Eigenfaces method retains the necessary features to allow for face recognition, I trained linear classifiers on the transformed data and discovered that it was possible to detect which reduced data belonged to which person with over 90% accuracy although each image dimension was compressed to less than 1% of its initial size.

In this second assignment, I implement two more advanced dimensionality reduction techniques, namely Fisherfaces and Autoencoders. Since the objective of this second assignment is to explore and implement more advanced methods, I dedicate this report to introduce these methods, explain their implementations and provide some preliminary results. Therefore, for this penultimate submission, I stick to the introduced Extended Yale Face Database B (cropped) [1] dataset to extract preliminary results from these two newly implemented methods. For the third and final submission, my plan is to try out a more involved dataset such as Large-scale CelebFaces Attributes (CelebA) Dataset [2]. This will allow for the comparison of all three implemented dimensionality reduction methods on multiple binary features such as male or female, glasses or no glasses, etc. However, this submission will mainly focus on the implementation and initial comparison on a small, well structured dataset.

2 LDA for Fisherfaces

Linear Discriminant Analysis (LDA) is an algorithm than can be used to reduce the complexity in high-dimensional data such that the class separability of the data in the new low-dimensional feature space is maximized. Clearly, one first needs a measure for this capacity of separation of each new feature space candidate. In 1988, statistician Ronald Fisher suggested a method to maximize a function that highlights the difference

between class means. Fisher's approach aims to increase the separation between the means of different classes (in our case the classes are person IDs) and reduce the spread within each class. This concept introduces two measures: within-class variability and between-class variability. First let us have a look at the within-class variability S_w:

$$S_w = \sum_{j=1}^{C} \sum_{i=1}^{n_j} (x_{ij} - \mu_j)(x_{ij} - \mu_j)^T \tag{1}$$

where C is the number of classes (38 for our case). n_j, μ_j and x_{ij} are the number of members, the mean vector and the data vector of i'th member of class j, respectively. This is a measure of how much data vectors deviate from their own class means in total.

Second, let us give the formula for the between-class variability:

$$S_b = \sum_{j=1}^{C} N_j(\mu_j - \mu)(\mu_j - \mu)^T \tag{2}$$

where C is the number of classes again, N_j and μ_j are the number of members and the mean vector of class j respectively and μ is the mean of all the data vectors in the dataset. This is a measure of how much the class means deviate from the mean of all classes, in other words how separated the classes are.

We now want to find a collection of basis vectors V where S_b is maximized with respect to S_b, where V is a matrix whose columns v_i are the basis vectors defining the subspace. This would allow us for optimum separability (and therefore detection of faces) in the reduced subspace. The objective function is therefore given by:

$$V_{\text{optimum}} = \arg\max_{V} \frac{|V^T S_b V|}{|V^T S_w V|} \tag{3}$$

This problem can be solved by the generalized eigenvalue decomposition:

$$S_b V = S_w V \Lambda \implies S_w^{-1} S_b V = V\Lambda \tag{4}$$

To solve for the Fisherfaces (matrix V), we need to compute the inverse of S_w, i.e., S_w^{-1} where S_w is a dxd matrix where d is the number of dimensions of each data vector (32,256 x 32,256 for our case). The problem is that the rank of this square matrix is at most $N - C$, where N is the total number of data vectors (2414 for our case) and C is the number of classes (38 for our case). This is because S_w can be seen as the sum of S_j's where class j contributes at most $N_j - 1$ to the rank of S_j, where N_j is the number of samples in class j. Since S_j lies in a subspace of at most $N_j - 1$ dimensions, the combined rank of S_w reflects the degrees of freedom across all classes. With N total samples and C class means, the maximum rank of S_w is $N - C$. By following the same logic, The eigenvectors of V associated with positive eigenvalues are the Fisherfaces and there are maximum of $C - 1$ Fisherfaces by the definition of S_b. Then, if the dimensionality of each of our images is more than 2414 - 38 = 2376, then S_w is non-invertable and we cannot solve Equation. 4. Our data matrix X in its initial state has 32,256 features, far more than 2376 and therefore not allowing the application LDA. The idea of Fisherfaces

method often is than to reduce the dimensionality of the data, generally PCA, before applying LDA.

For the visualization of the Fisherfaces, I first reduced the dimensionality of the data matrix X by using PCA as explained in the first submission. Later, I implemented variability measures given in Equations 1 and 2 by hand in Python (please see the GitHub code base link in the project title). Then, I extracted the Fisherfaces by solving the Equation. 4. I then reshaped these Fisherfaces to the original image size and visualized them. These Fisherfaces can be seen in Figure. 1.

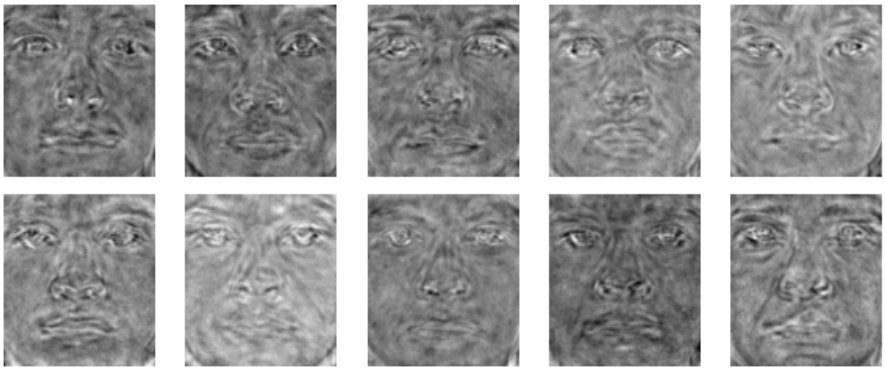

Figure 1: Visualization of the First 8 Fisherfaces

We see that these eigenvectors resemble real faces and they encode the most important feature to allow for the best class separability in the reduced space. Note that for the real application of the LDA I used Scikit-learn's Linear Discriminant Analysis function [3] after applying PCA, which enables a more optimized implementation in terms of time and numeric stability.

3 Convolutional Autoencoders

An autoencoder is an artificial neural network designed to learn efficient representations of unlabeled data through unsupervised learning. It operates by learning two functions: an encoding function that transforms the input data into a different representation, and a decoding function that reconstructs the original input data from this encoded representation. Autoencoders are commonly used for dimensionality reduction by creating an efficient encoding of the data.

A convolutional neural network (CNN) on the other hand is a specialized type of artificial neural network primarily used for processing structured grid (2-D) data, such as images. CNNs are designed to automatically and adaptively learn spatial features from input images. They consist of convolutional layers that apply convolution operations to the input, pooling layers that reduce the spatial dimensions, and fully connected layers that make the final classification or regression predictions.

In this project, I chose to implement a convolutional autoencoder over a standard autoencoder for image dimensionality reduction because they are specifically tailored to handle the spatial structure and features of images. The convolutional layers in CNNs can efficiently capture local patterns, such as edges, textures, and shapes, which are crucial for

understanding the content of images. This is because the input image is retained in 2-D grid form and processed via 2-D convolution filters, a much more efficient way of handling image data. On the other hand, if I were to use a fully connected neural network I would have to flatten the images and then try to extract important features from the flattened version, losing valuable local patterns that make up a face in the process.

The basic idea of a convolutional autoencoder used for dimensionality reduction can be explained as follows: The structure consists of an encoder part followed by a decoder part. In the encoder part, using convolutional layers with strides greater than one (or pooling layers in between) the input image is fed forward for multiple layers and it shrinks in dimension from layer to layer. Let us say that the reduced dimensions of the image after all the encoder convolutional layers is $D_1 x D_2$ and we want to reduce the total dimensions of each image to K. Then, the reduced image is flattened to a vector of size $D_1 x D_2$ and this vector transformed to a vector of size K using a fully connected linear layer. This concludes the encoder part.

In the decoder part, the whole progress is reverted to recover the original image from the reduced vector of size K. Specifically, a reverse fully connected linear layer is used to come back from dimensionality K to $D_1 x D_2$. Then, this vector is unflattened (reshaped into an image). Then, the convolutional layers of the encoder is now replaced by their corresponding convolutional transpose layers in the decoder, which upsample the input image to a higher resolution by applying learned filters. The original input image dimension is reached and at the output of the model we obtain an image with the same dimension as the original image. This concludes the decoder part.

Note that the purpose of this model is to reduce the dimensionality of the data to size K in a way that the original is most recoverable. Therefore, we give an image X as the input during the training and the target (label) of the image is the image X itself. The loss I used in this project to train the model is MSE loss. This means that the model should be able to reduce the dimensionality of an image to K and recover it in the output such that the sum of squared errors of each corresponding pixel between input and recovered image should be minimum. This in turn means that the reduced space of dimension K retains enough information about the original image to allow for its recovery and therefore it should be a valid compressed form of the image. This is the motivation behind the convolutional encoder method.

In this project, I followed the approach proposed in [4] to implement a convolutional autoencoder module using PyTorch. The structure of the module I implemented can be given in Figure 2.

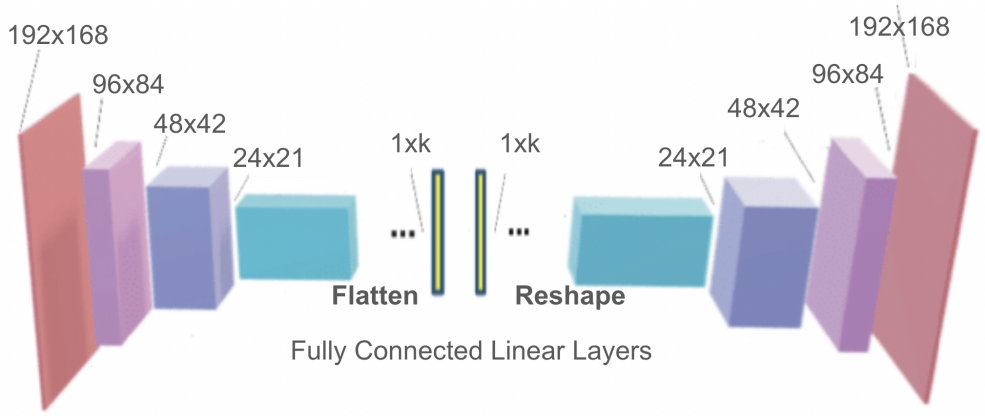

Figure 2: Visualization of my Convolutional Autoencoder Module

In this module, I used 2x2 filters with stride 2 such that each layer reduced (or increased) the input dimension by a factor of 2. The loss used is MSE loss, the optimizer is Adam optimizer, the learning rate is 3×10^{-4} and weight decay of 1×10^{-5} is used to prevent overfitting. 100 epochs are used with 0.75-0.25 train-test split as before. Then, the model is used to reduce the dimension of the test set and the resulting compressed data is tested for classification. The results and initial comparison with the other methods will be provided in the next section.

3.1 Results with Linear Classifier

To test if the methods implemented retain the necessary features to allow for face recognition, we train a linear classifiers on the transformed data. For this, we first split the data as train and test sets by using 75% of data for training and saving 25% for testing. Then, we train a logistic regression model [5] and an SVM (Support Vector Machine) model [6] separately and test the resulting models. For both of the models, maximum number of iterations is 250 and the regularization parameter "C" is set to 0.005 for logistic regression and 0.001 for SVM. We train models with different number of compressed dimension sizes to see the variation of accuracy of each method. Tables 1, 2 and 3 summarize the results.

	8	16	32
Logistic Regression	0.23	0.57	0.81
SVM	0.31	0.69	0.90

Table 1: Eigenfaces: Classification Accuracy for different numbers of dimensions

	8	16	32
Logistic Regression	0.57	0.81	0.96
SVM	0.61	0.82	0.97

Table 2: Fisherfaces: Classification Accuracy for different numbers of dimensions

	8	16	32
Logistic Regression	0.70	0.85	0.95
SVM	0.81	0.94	0.98

Table 3: Convolutional Autoencoder: Classification Accuracy for different numbers of dimensions

We see that although convolutional autoencoders perform better than the other two methods, especially when using lower number of dimensions.

4 Conclusion and Comparison of Methods

In this assignment, I implemented and compared three different dimensionality reduction techniques: PCA (Eigenfaces), Fisherfaces, and Convolutional Autoencoders. Each method has its own set of advantages and disadvantages.

PCA is simple to implement and computationally efficient, making it suitable for many applications. However, it does not take class information into account, which can result in poor class separability in the reduced space. It is also sensitive to scaling and may not perform well if the data has noise or outliers.

Fisherfaces use Linear Discriminant Analysis (LDA) to maximize class separability in the reduced feature space. By incorporating class labels, Fisherfaces improve the discriminative power compared to PCA. However, they are computationally more intensive and require the inversion of the within-class scatter matrix, which can be problematic if the matrix is singular or nearly singular.

Convolutional Autoencoders use convolutional layers to learn efficient representations of images while preserving their spatial structure. They typically perform better than PCA and Fisherfaces, especially with lower-dimensional representations, and can handle non-linear relationships in the data. However, they are more complex to implement, require more computational resources and training time, and need careful tuning of hyperparameters.

Overall, Convolutional Autoencoders provide the highest classification accuracy, especially with lower-dimensional representations. Fisherfaces also perform well, particularly with higher dimensions. PCA, while effective, tends to lag behind in classification performance. PCA is the most computationally efficient, followed by Fisherfaces and then Convolutional Autoencoders. The choice of dimensionality reduction technique depends on the specific requirements of the task, such as accuracy, computational efficiency, and implementation complexity.

References

[1] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. Extended Yale Face Database B Cropped (PNG Version). `https://www.kaggle.com/datasets/tbourton/extyalebcroppedpng`, 2022. Accessed: 2024.

[2] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes

in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[3] sklearn.discriminant_analysis.lineardiscriminantanalysis. `https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html`. Accessed: 2024-05-25.

[4] Dylan Snover, Christopher Johnson, Michael Bianco, and Peter Gerstoft. Deep clustering to identify sources of urban seismic noise in long beach, california. *Seismological Research Letters*, 92, 12 2020.

[5] Scikit-learn Developers. sklearn.linear_model.logisticregression — scikit-learn 1.1.4 documentation. `https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html`, 2023. Accessed: 2024.

[6] Scikit-learn Developers. sklearn.svm.svc — scikit-learn 1.1.4 documentation. `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`, 2023. Accessed: 2024.