Comparing PCA and LDA Techniques on the LFW Dataset for Dimensionality Reduction

Abstract

This mini-project aims to compare two dimensionality reduction methods, Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), using labeled face image data. Both PCA and LDA employ eigenvalue analysis to project high-dimensional data into a lower-dimensional space. PCA is an unsupervised method that does not use class labels for dimensionality reduction, whereas LDA is a supervised method that leverages label information.
The comparison was conducted using the Labeled Faces in the Wild (LFW) dataset, with classification performance evaluated via the k-nearest neighbor (k-NN) method. LDA outperformed PCA in terms of recognition accuracy. It was particularly effective in balanced datasets, where each class contained the same number of samples, and showed reduced effectiveness when sample sizes varied across classes. Additionally, in PCA-based dimensionality reduction, the use of the L1 norm as the distance metric in k-NN classification resulted in higher accuracy compared to the L2 norm.

Introduction

The Face Recognition Technology (FERET) dataset is a standard benchmark for face recognition research, containing 14,126 images of 1,199 individuals [1]. A previous study compared PCA and LDA using the FERET dataset and concluded that no single method performs best across all tests; instead, the optimal algorithm depends on the specific task. The authors stated that “no particular projection–metric combination is the best across all standard FERET tests and the choice of appropriate projection–metric combination can only be made for a specific task” [2].
In this mini-project, I compare the dimensionality reduction techniques of PCA and LDA using the LFW dataset provided by the University of Massachusetts Amherst [3]. The LFW dataset is a publicly available benchmark for face recognition, containing more than 13,000 face images collected from the web. Each image is cropped to center the face and annotated with the name of the person shown. The dataset also exhibits considerable variation in “pose, lighting, expression, background, …camera quality, color saturation, and other parameters” [4]. To evaluate performance in a realistic setting, PCA and LDA are applied to face recognition tasks involving such variations in image quality and characteristics. For this analysis, the data used were from classes with a sample size of over 70, consisting of 1,288 grayscale images of 37 × 50 pixels.

Background

This mini-project primarily uses dimensionality reduction techniques and a classification model. Two approaches to dimensionality reduction were employed: principal component analysis (PCA) and linear discriminant analysis (LDA). PCA is a method used to reduce dimensions while preserving maximum variance in high-dimensional datasets. In this project, it is used to extract significant features from image data. On the other hand, LDA helps to enhance the discriminability between different classes by projecting data in a way that maximizes the separation of the classes. For classification, the k-nearest neighbor (k-NN), a supervised learning classifier, was used. k-NN classifies unknown data points based on a 'majority vote' of the k nearest training data points.
In the remainder of this Background section, the specifications of the PCA algorithm implemented from scratch will be discussed, as well as the eigenvalue analysis utilized in both the PCA and LDA dimensionality reduction methods.

The MyPCA Class

The PCA algorithm was implemented from scratch. The MyPCA class contains the following methods and instance variables.

Class methods
__init__(self, num_components, decomposition='eigen')
fit(self, X_nxf, y=None)
transform(self, X_nxf)
Instance variables
self.n_components
self.decomposition
self.components_
self.explained_variance_ratio_
self.explained_variance_ratio_all_
self.mean_

The __init__ method takes the number of features after dimensionality reduction and the decomposition method as arguments. If 'decomposition' is set to 'eigen', eigenvalue analysis is performed, whereas if it is set to 'svd', singular value decomposition is performed. SVD is well suited for high-dimensional data, and the PCA class in scikit-learn performs PCA using SVD.

The fit method includes the following steps:

1. Center the data around its mean.
2. Store the computed mean of the dataset X.
3. Calculate the covariance matrix of the centered data.
4. Calculate eigenvalues and eigenvectors.
5. Sort the eigenvectors in descending order of their corresponding eigenvalues.
6. Store the top ‘n_components’ eigenvectors as the principal components.
7. Store the explained variance ratios for both the selected ‘n_components’ and for all components.

The fit method argument, X_nxf, is a matrix where rows represent samples and columns represent features. The mean and top 'n_components' eigenvectors of X_nxf, stored as instance variables in steps 2 and 6 respectively, are used in the transform() method. The explained variance ratios, stored in an instance variable in step 7, are utilized to determine the number of components. The transform method performs dimensionality reduction on the data using principal components. This method returns a matrix where rows represent samples and columns represent the reduced number of features.

The following changes were added to the MyPCA class to make it compatible with the GridSearchCV pipeline:

1. An additional argument, y=None, is added to the fit() method.
2. The fit() method is altered to return self. This change facilitates method chaining, allowing for a call sequence like mypca.fit().transform().

The Eigenvalue, Eigenvector, and Eigenanalysis

Before moving on to the Methodology section, I will discuss the eigenvalue, eigenvector, and eigenanalysis. PCA and LDA use eigenanalysis to extract features for dimensionality reduction and class separation, respectively. An eigenvector is a vector which, when multiplied by the matrix A, is only multiplied by λ and does not change its direction.

$$\mathbf{A}\overrightarrow{v} = \lambda\overrightarrow{v}$$

The meanings of each symbol are as follows:

- The matrix A is an NxN real matrix
- The vector v is a N×1 real vector
- The vector v is nonzero
- λ is a scalar

In this context, v represents an eigenvector of the matrix A, and λ is a corresponding eigenvalue.

For example, consider a matrix A with entries [[4, 1], [2, 3]]. If we have an eigenvector v represented by the column vector [1, -2], then the corresponding eigenvalue is 2. This means that when the matrix A acts on v, the resulting vector is Av = λv, which is simply v scaled by a factor of 2, without any change in its direction.

$$\mathbf{A}\vec{v} = \begin{pmatrix} 4 & 1 \\ 2 & 3 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \end{pmatrix} = \begin{pmatrix} 2 \\ -4 \end{pmatrix} = 2 \begin{pmatrix} 1 \\ -2 \end{pmatrix} = \lambda\vec{v}$$

Eigenanalysis finds the eigenvectors and eigenvalues for a given matrix A.

$$\mathbf{A}\vec{v} = \lambda\vec{v}$$

This can be rearranged to:

$$\mathbf{A}\vec{v} - \lambda\vec{v} = \vec{0}$$

Which simplifies to the characteristic equation:

$$(\mathbf{A} - \lambda\mathbf{I})\vec{v} = \vec{0}$$

Letting,

$$\mathbf{A} - \lambda\mathbf{I} = \mathbf{S}$$

and we get:

$$\mathbf{S}\vec{v} = \vec{0}$$

If S were not singular, it would have an inverse matrix, which would imply that the only solution to Sv = 0 is v = 0, contradicting the fact that v is a non-zero eigenvector.

$$\mathbf{S}^{-1}\mathbf{S}\vec{v} = \mathbf{S}^{-1}\vec{0}$$

Thus, S is a singular matrix and its determinant is zero.

$$\det(\mathbf{A} - \lambda\mathbf{I}) = 0$$

Suppose we are given a diagonal matrix A with diagonal entries 2 and 3,

$$\mathbf{A} - \lambda\mathbf{I} = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix} - \begin{pmatrix} \lambda & 0 \\ 0 & \lambda \end{pmatrix} = \begin{pmatrix} 2-\lambda & 0 \\ 0 & 3-\lambda \end{pmatrix}$$

We get λ = 2, 3:

$$\begin{vmatrix} 2-\lambda & 0 \\ 0 & 3-\lambda \end{vmatrix} = (2-\lambda)(3-\lambda) = 0$$

When λ=2, we get:

$$\begin{pmatrix} 2-2 & 0 \\ 0 & 3-2 \end{pmatrix} \vec{v} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \vec{v} = \vec{0}$$

Hence, we can choose v = [1, 0] as the eigenvector for λ=2. Similarly, v = [0, 1] for λ=3. This confirms that, from the input matrix A = [[2, 0], [0, 3]], the eigenvalues 2 and 3 can be derived, with the corresponding eigenvectors [1, 0] and [0, 1], respectively.

In summary, eigenanalysis decomposes the input matrix into vectors representing directions (eigenvectors) and scaling factors along those directions (eigenvalues). The larger the eigenvalue, the more its corresponding eigenvector accounts for the characteristics of the original matrix.
The following code demonstrates how to perform eigenvalue analysis using NumPy's numpy.linalg.eig function on a given input matrix A, where eigenvectors are represented as columns of a matrix:

```
import numpy as np

A = np.array([[2,0],[0,3]])
eigenvalues, eigenvectors = np.linalg.eig(A)
# eigenvalues:  [2. 3.]
# eigenvectors:
# [[1. 0.]
#  [0. 1.]]
```

In the case of PCA, the matrix A is the covariance matrix of X. If X is a dataset with three features (a,b,c), the covariance matrix cov(X) is as follows:

$$\mathbf{A} = cov(X) = \begin{bmatrix} \mathrm{Var}(a) & \mathrm{Cov}(a,b) & \mathrm{Cov}(a,c) \\ \mathrm{Cov}(a,b) & \mathrm{Var}(b) & \mathrm{Cov}(b,c) \\ \mathrm{Cov}(a,c) & \mathrm{Cov}(b,c) & \mathrm{Var}(c) \end{bmatrix}$$

This covariance matrix is a real symmetric matrix. A real symmetric matrix has the same elements in the triangular part below the diagonal as in the triangular part above the diagonal. The numpy.linalg.eigh method computes eigenvalues and eigenvectors using only half the elements of the matrix for efficient computation [5]. Thus, the MyPCA class uses the numpy.linalg.eigh method for eigenanalysis.

On the other hand, LDA is a supervised dimensionality reduction technique, which incorporates class information. LDA performs projections in a way that minimizes the variance within each class while maximizing the variance between different classes.

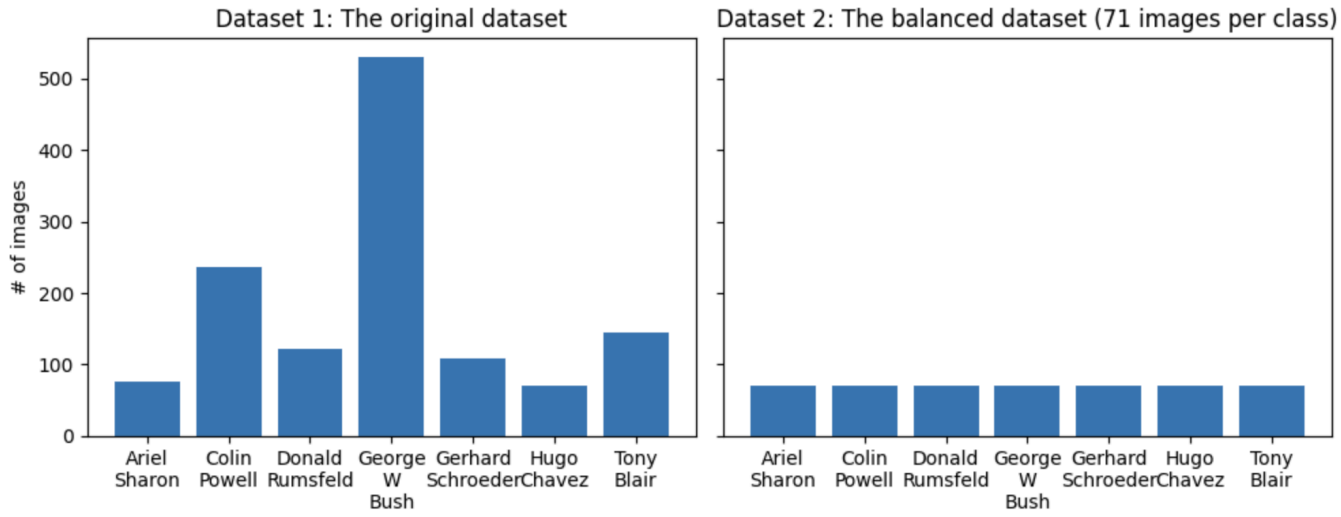
Methodology

In this section, I will focus on five key aspects of the project's approach:

1. Utilizing two dataset variations: original and class-balanced.
2. Dividing data into training, validation, and test sets.
3. Selecting the number of PCA components based on explained variance ratios.
4. Using X_train to fit PCA/LDA to avoid information leakage.
5. Employing N-fold nested cross-validation for more stable evaluation metrics.

1. Utilizing two dataset variations: original and class-balanced

This mini-project uses data from classes, each with at least 70 samples. The left-hand graph below displays the number of samples per class. The labeled faces dataset exhibits a significant imbalance in class sample sizes, with George W. Bush having the most images. Previous research has shown that such class imbalances can negatively affect the performance of LDA [6]. To mitigate this issue, evaluations are conducted using both the original, imbalanced dataset (Dataset 1) and a class-balanced version (Dataset 2). The right-hand graph below illustrates that in Dataset 2, the number of images has been equalized across classes. Employing these two dataset variants facilitates a more in-depth analysis of how class imbalance influences LDA performance.



2. Dividing data into training, validation, and test sets.

Each dataset was split into three parts: training, validation, and test sets. The test set must be large enough to obtain statistically significant results and sufficiently representative of the original data. [7] To meet this requirement, the sklearn.model_selection.StratifiedKFold method was employed for splitting the data. This method ensures that the test data maintains the class ratio present in the original dataset. Given that the commonly used split ratio is 80:20, the n_splits parameter of the StratifiedKFold method was set to 5.

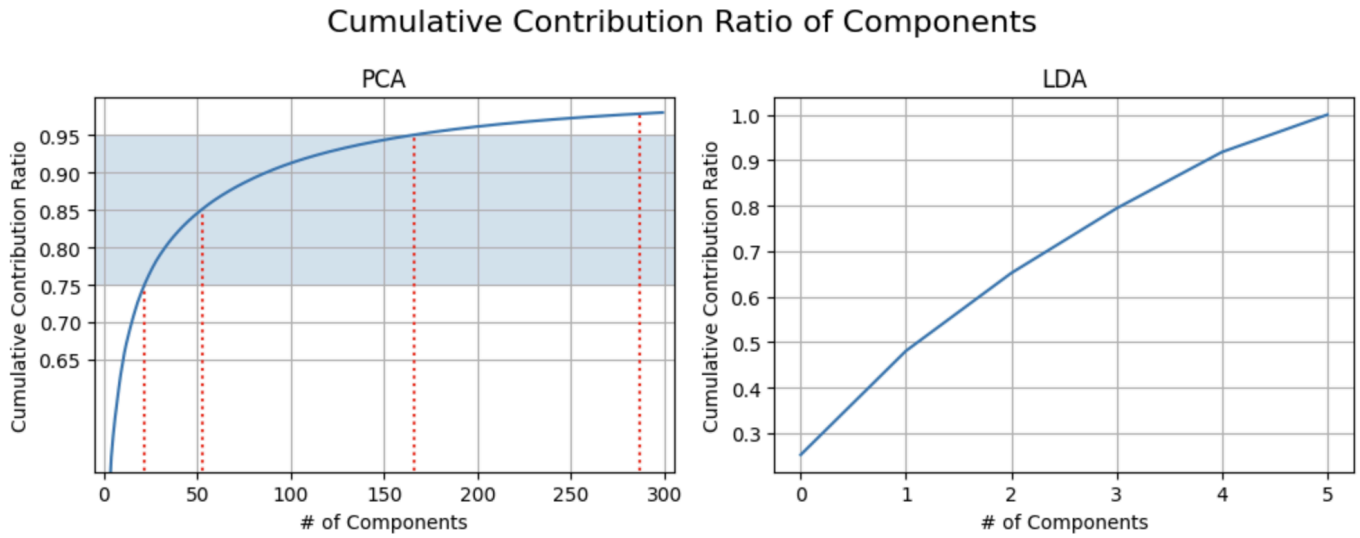
3. Selecting the number of PCA components based on explained variance ratios.

The number of PCA components was determined based on the explained variance ratios. Cumulative contribution ratio thresholds were set at 75%, 85%, and 95%, and the corresponding number of components for each threshold was calculated. Additionally, based on a method used in previous research [2], a number of components that represented 97.85% of the variance was also selected. Consequently, the performance of each set of component numbers {22, 53, 166, 287} was evaluated.

The relationship between cumulative contribution rates and the number of components is summarized in the table below.

Cumulative contribution ratio	> 75%	> 85%	> 95%	> 97.85%
# of components	22	53	166	287

The graphs below represent the relationship between the number of components and the cumulative contribution ratio for PCA and LDA. The graph on the left includes four red dotted lines, which indicate the cumulative contribution ratios at the points where the number of components reaches 22, 53, 166, and 287, respectively.



4. Using X_train to fit PCA/LDA to avoid information leakage.

The use of test data in preprocessing, such as data imputation, is known as train-test contamination. [8] Incorporating test data into the dataset's average makes it known to the model, resulting in improved accuracy. The model performs well in testing, but after deployment, it does not perform as well as it did in testing. To avoid data leakage, it's crucial that the test data remain concealed from the predictive model during the fitting process in PCA and LDA. Therefore, only X_train should be utilized in the fit functions of PCA and LDA.

5. Employing N-fold nested cross-validation for more stable evaluation metrics.

N-fold nested cross-validation was used to obtain more reliable results. The k-NN model was used for classification, and the k values were set to 9, 11, and 13. These odd numbers were chosen for k to avoid tie votes. Additionally, Manhattan distance ('l1') and Euclidean distance ('l2') were selected as the distance measurement metrics.

Results

The results are shown in the table below.

	Method	Dataset	n_components	Accuracy
0	PCA + KNN	1	22	0.545821
1	PCA + KNN	1	53	0.625793
2	PCA + KNN	1	166	0.623458
3	PCA + KNN	1	287	0.611818
4	PCA + KNN	2	22	0.426444
5	PCA + KNN	2	53	0.474626
6	PCA + KNN	2	166	0.508990
7	PCA + KNN	2	287	0.492949
8	LDA + KNN	1	6	0.702648
9	LDA + KNN	2	6	0.752545

The Method column shows the combination of the dimensionality reduction method and the classification model used. The Dataset column indicates the type of dataset used: a value of 1 indicates the original Dataset 1 and 2 indicates class-balanced Dataset 2 respectively. N_components is the number of features after reduction. The Accuracy column is the score of the k-NN model.

As a result, the respective properties and differences between LDA and PCA are as follows:

- The best accuracy was 0.752545, achieved by LDA + k-NN on the balanced dataset.
- LDA consistently outperformed PCA regardless of the dataset or number of components.
- For PCA, the highest accuracy on the imbalanced dataset was obtained using 53 components, which accounted for 85% of the cumulative variance. In contrast, on the balanced dataset, the best result was achieved with 166 components, corresponding to a 95% cumulative contribution rate.
- LDA performed better with a balanced dataset than with a dataset containing more total samples but significant class imbalance.

Another finding is that when dimensionality reduction is performed using PCA, using 'l1' as the distance metric in k-NN yields better accuracy than 'l2'. The following table shows the aggregate best distance metrics determined through N-fold nested cross-validation. As shown in the table, the best parameters for PCA consistently selected 'l1'.

Method	Dataset	knn__metric	count
PCA + KNN	1	l1	20
PCA + KNN	2	l1	20
LDA + KNN	1	l2	5
LDA + KNN	2	l2	3
LDA + KNN	2	l1	2

Evaluation

This mini-project aims to compare the dimensionality reduction techniques PCA and LDA on the LFW dataset and employing k-NN for classification. It became clear that LDA results in higher classification accuracy.
A strength of this project is the precise use of basic ML techniques. The simple data pipeline enabled a more accurate and comprehensive comparison by preparing two datasets and using N-fold nested cross-validation for accuracy measurement. With the defined MyPCA class and N-fold nested cross-validation function, this pipeline is flexible. For example, other dimensionality reduction methods and classification models can be easily tried. Additionally, the fit() and transform() methods in MyPCA allowed GridSearchCV to utilize this class.
A weakness is that the image data was not normalized. By normalizing pixel values to a range between 0 and 1, accuracy might improve. Furthermore, the 'svd' decomposition option implemented in MyPCA was not used, while scikit-learn's LDA utilizes SVD. Using 'svd' might have been more appropriate for comparative analysis.

Conclusions

This mini-project compared PCA and LDA dimensionality reduction methods using the LFW dataset, with k-NN employed for classification. The results showed that LDA achieved higher accuracy. While PCA extracts the main features of the data regardless of class labels, LDA transforms the data based on label information. Therefore, when class labels are available, LDA may lead to better performance.
For PCA, the highest accuracy was achieved when selecting components that accounted for 85% of the cumulative variance. Additionally, in PCA-based dimensionality reduction, using the L1 norm yielded better accuracy than the L2 norm. This finding is consistent with results reported in a previous study [2].
For LDA, using a balanced dataset improved accuracy more effectively than simply having a large number of samples. While accuracy with the original unbalanced dataset was 70%, it increased to 75% with the class-balanced dataset.
This project used appearance-based face recognition, but accuracy may improve by using a feature-based approach, such as detecting eye and nose positions.

Reference

[1] National Institute of Standards and Technology. n.d. Face Recognition Technology (FERET) | NIST. Retrieved January 8, 2024 from https://www.nist.gov/programs-projects/face-recognition-technology-feret.

[2] Kresimir Delac, Mislav Grgic, and Sonja Grgic. 2005. Independent comparative study of PCA, ICA, and LDA on the FERET data set. International Journal of Imaging Systems and Technology 15, 5 (2005), 252–260. https://doi.org/10.1002/ima.20034.

[3] University of Massachusetts Amherst. n.d. Labeled Faces in the Wild : UMass Amherst. Retrieved January 8, 2024, from https://vis-www.cs.umass.edu/lfw/.

[4] Gary B. Huang, Marwan Mattar, Tamara Berg, and Eric Learned-Miller. 2008. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. In Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition, Erik Learned-Miller and Andras Ferencz and Frédéric Jurie, Marseille, France. https://hal.inria.fr/inria-00321923.

[5] NumPy. 2023. numpy.linalg.eigh — NumPy v2.2 Manual. Retrieved January 8, 2024, from https://numpy.org/doc/stable/reference/generated/numpy.linalg.eigh.html.

[6] Jigang Xie and Zhengding Qiu. 2007. The effect of imbalanced data sets on LDA: A theoretical and empirical analysis. Pattern Recognition, 40, 2 (2007), 557–562. https://doi.org/10.1016/j.patcog.2006.01.009.

[7] Google. n.d. Splitting data: Training and test sets. Retrieved January 7, 2024, from https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data.

[8] Kaggle. n.d. Data Leakage | Kaggle. Retrieved January 6, 2024, from https://www.kaggle.com/code/alexisbcook/data-leakage.