EigenFaces and Mathematics Behind it

Farhan Rashid

March 2024

1 Abstract

We are presenting an model for detection and identification of human faces by identifying the features as a Black or non-black person. Basically this model is able to identify human face images and can figure out if the person is black or non-black. Our initial approach is identifying 2D grayscale images of face which should be image of of face.[3] We used the eigenface which is basically the set of eigenvectors from our training set in order to identify images of faces. As eigenface is a feature selection model we trained our model in such way that it could identify images between black and non-black persons.[4]

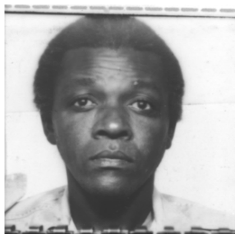
2 Introduction

We are developing a machine learning supervised model with eigenfaces techniques for face recognition. Our main challenge was accurately predicting the black and non-black people from the image and for that reason we trained the model separately with both kind of images.[1]

The process begins by loading pre-processed images which are grayscale with dimension 255×255 to ensure uniformity across the dataset. This images flattened from a 2D matrix to a 1D vector.[1] Th efficiency and accuracy increased when we used principal component analysis in the model that's we determines the larger eigenvectors and important features for identifying faces according to our requirement. Our initial $n_component$ or the PCA was with 150 and later on we trained the model with 100 which provides us more accuracy.[1]

One of the challenge that we expecting was some of the images were from left/right sided which could later cause problem in our training or predicting. However, it doesn't become a big issue as we take in consideration about the big eigenvalues and correspondings. These eigenfaces represents the directions in which the images vary most. By projecting the original centered images onto these eigenfaces we effectively reduces the dimensionality of our data while retaining the most significant features of the faces.[5][6]

In the face recognition part, each image can be represent as linear combination of eigenfaces.These coefficient forms a feature vector for each face in a reduced-dimension space, simplifying the classification task. Eigenfaces thus



captures essential features of faces, making them powerful tool for face recognition.[5][6]

3 Method

The primary objective of our code is to classify images of two catagories. These catagories are black and non-black individuals. In order to achieve this, we employed a separate training approach for each catagory of images this allowed tailored learning from noticeable features throughout the photos.First, grayscale images are each resized to a resolution of 255×255. Standardizing the data in preparation for feature extraction.

Nest, the images are converted from its original 2D matrix to a 1D vector, this will simplify the format for the model. One critical component is the use of PCA. This process not only simplifies the dataset but also enhances the computational efficiency by focusing on the most informative aspects of the facial images. The value which was first used 150, although the accuracy was nearly 78. However for some better accuracy of the model we tried with 100 components which yields a good 82 percent accuracy. So it's clear that the more big eigenfaces for prominent features we select the more accurately it trained the model.

Then we went about trianing the SVM with an RBF kernel. This is tailored to handle class imbalance such that it can create a 'balanced' class weight parameter. To help with this we can separate the dataset with 75 percent of the data as training while the remaining 25 percent would be used as testing. This helped with noticing patterns shown with the photos in the testing set such that the model can match patterns with a distinct category. Finally we get the effectiveness of the model evaluated with precision, recall, F1-score ad accuracy metrics, culminating in a detailed classification report. This allows for us to log all info that goes through the model and ensure that we can have an accuracy that will be applicable to the real world.

In SVM setup with the RBF kernel , the classifier uses this kernel function to transform the data into a higher-dimensional space where it attempts to find the optimal hyperplane that separates the classes with the maximum margin. The use of $class_weight =' balanced'$ helps to adjust weights inversely proportional to class frequencies, aiding in dealing with imbalanced data, which often enhances the classifier's performance on minority classes.

$K(x, x') = e^{-\gamma \|x-x'\|^2}$

The RBF kernel $K(x, x') = \exp(-\gamma\|x - x'\|^2)$ calculates the similarity between two points x and x' based on the exponential decay of their squared Euclidean distance, scaled by γ. Points closer together yield higher similarity.

4 Dataset

The dataset we are going to use for this project is investigated and collected from kaggle which is named as "NIST Mugshots". In this dataset we have 3248 bits gray scale images which of 1573 individual's either front facing or from the side. These are enough images to train the model in order to have some accurate prediction. Here we divided the dataset to 75 percent training data which will be used to train the model. The rest 25 percent is for it's testing which the model will be using in order to find out how much accurately it's working after being trained in a certain way.[4]

5 Background

5.1 PCA

Principal Component Analysis is widely used for face recognition and image processing by reducing the dimensionality of large dataset while preserving the important information. In the Eigenface method for facial recognition, Principal Components Analysis (PCA) is employed to reduce the dimensionality of face image data, enhancing the efficiency and effectiveness of classification algorithms like Support Vector Machines (SVM).[6] in mean factor we will be using PCA because Centering is achieved by subtracting this mean face from each individual face image, which shifts the dataset such that the average face vector is zero. This step is crucial because PCA requires data to be centered around zero to accurately identify the principal components that capture the most significant variations in the data. $\psi = \frac{1}{M}\sum_{i=1}^{M}\gamma;$

In the Eigenfaces method, the covariance matrix is calculated to assess the variance and covariances among the pixel values of the mean-centered face images. This matrix is crucial as it identifies the directions in which the data varies the most. These principal components, derived from the covariance matrix, form the basis of the Eigenfaces, capturing the most significant facial features for recognition tasks.

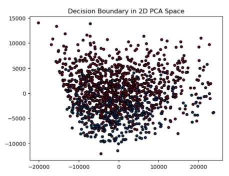


Figure 1: Decision Boundary of 2D PCA Space of our Model

$C = \frac{1}{M} AA^T$ [1]

Initially, PCA standardizes the face images by subtracting the mean face from each image to center the data around zero. It then computes a covariance matrix to capture how different pixels in the images co-vary, identifying patterns of variation across the dataset.[4] By extracting eigenvectors (principal components) and their corresponding eigenvalues from this matrix, PCA determines the directions of maximum variance in the data. These principal components are the key features that capture the most significant facial characteristics necessary for distinguishing between individuals. The SVM classifier then uses these principal components as features to categorically separate and classify different faces, effectively leveraging the reduced dimensions to maintain essential information while improving computational speed and accuracy.[6][5]

5.2 SVM

In the context of Eigenface classification, Support Vector Machines (SVM) serve as a robust method for face recognition by efficiently handling high-dimensional input data. SVM operates by identifying a hyperplane that optimally separates different classes in a feature space, which, in the case of face recognition, would distinguish between different individuals. According to the paper by Theodoros Evgeniou and Massimiliano Pontil, SVM achieves this by transforming the original input space into a higher-dimensional space using a kernel function, thereby allowing for the handling of complex patterns in the data.[5] For face recognition, after applying PCA to reduce dimensionality and capture the essential features of faces (Eigenfaces), the SVM classifier then utilizes these features to classify new images. The SVM approach is particularly effective due to its ability to maximize the margin between different classes, thus enhancing its generalization ability and reducing the risk of overfitting, which is crucial for achieving high accuracy in facial recognition tasks.[5] Linear decision function: $f(x) = \mathbf{w} \cdot \mathbf{x} + b$

W is the weight vector perpendicular to the hyperplane. X is the feature vector and b is the bias term.

Non-Linear decision Function with kernel:

$f(x) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x) + b$

Here the α_i are the lagrange multipliers obtained from solving the SVM optimization problem. These multipliers are non-zero only for the support vectors. y_i are the class labels of the training samples. x_i are the support vectors themselves. b is the bias term and $K(x_i, x)$ is the kernel function evaluated between the new data point of x and each support vector x_i

6 Explanation of EigenFaces

EigenFaces refer to set of eigenvectors when they are used in the computer vision problem of human face recognition. The approach is based on the principal component analysis(PCA), a statistical method used to reduce the dimensionality of data while retaining most of the variance.[3]

The background of this analysis depends on mathematics a lot where we need to use a lot of linear algebra and statistics. For eigenface we need to know what eigenvalue, eigenvector is and how to find them from regular set of matrix.

Eigenvectors and eigenvalues are fundamental in understanding the characteristics of a matrix, offering insights into its behavior. They are crucial in simplifying complex systems, particularly in linear transformation does not alter the direction of vectors. In the realm of image processing, "eigenfaces" are used for facial recognition by capturing the most significant features of faces, allowing for efficient representation and comparison of facial images. This approach dramatically reduces the dimensionality of the data, making computations more manageable and improving recognition accuracy.[3]

7 Explanation of Eigenvalue

An eigenvalue is a scalar that describes the factor by which the eigenvector is scaled during a linear transformation. If you have a square matrix A, an eigenvector v and an eigenvalue λ , the relationship can be expressed as :

$$Av = \lambda v$$

[1][3]

8 Explanation of Eigenvector

An eigenvector is a non-zero vector that only changes by a scalar factor when a linear transformation is applied to it. In the context of the equation

$$Av = \lambda v$$

, the vector v is an eigenvector of the matrix A. It shows the direction that remains unchanged (except for scaling) when the transformation associated with A is applied.[3] In essence, when a linear transformation represented by A is applied to an eigenvector v, the output is a vector that's parallel t v, scaled by a factor of λ, which is the eigenvalue. Eigenvectors and eigenvalues are critical in understanding the properties of linear transformation, and they have practical application in stability analysis, quantum mechanics, facial recognition algorithms, and more.[3]

9 The matrix and calculation

The matrix we have this time or calculation is a 2x2 matrix. I(x,y) =

$$\begin{bmatrix} 1 & 1 \\ -2 & -3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 3 & 2 \end{bmatrix}$$

[1][2] From here we obtain 2 images which we named as $I_1 \quad I_2$ Now we going to represent each of this matrix in vectors. From here we going represent I_i in $\bar{\gamma}$.[1] Here $\bar{\gamma}$ is the converted one dimensional vector of the matrix image we using. $\bar{\gamma_1} = \begin{bmatrix} 1 \\ -2 \\ 1 \\ -3 \end{bmatrix}$

$\bar{\gamma_2} = \begin{bmatrix} 1 \\ 3 \\ -1 \\ 2 \end{bmatrix}$ [1][2] This is the training set of M images of size NxN Here the vector of the training set image is 4×1 [1]

10 Mean Face Vector ψ

This is the mean face vector which we going to denoted as ψ. Its basically the average of the training set image which we going to use later for getting more accurate and effective calculation.[1] $\psi = \frac{1}{M} \sum_{i=1}^{M} \gamma$;[1]

$\psi = \frac{1}{2} \cdot \begin{bmatrix} 1+1 \\ -2+3 \\ 1-1 \\ -3+2 \end{bmatrix}$

$\psi = \begin{bmatrix} 1 \\ \frac{1}{2} \\ 0 \\ \frac{-1}{2} \end{bmatrix}$ [2][1] Here the mean face in the context of eigenfaces represents the average facial features of all the faces in the dataset. It's calculated by averaging each pixel's intensity across all the face images considered. This average image serves as a generic or central face that reflects the common features shared among the faces in the set, without emphasizing any individual's distinctive character. In the eigenfaces method, the mean face is used as a baseline or reference point. All individual face images are compared to this mean face to identify unique variations from average, which helps in emphasizing the distinguishing features necessary for face recognition. Essentially, the mean face embodies the 'typical' facial features of the group under analysis.

11 Difference between each image and average ϕ

This is formula of calculating the difference e between the real image (training image) and the average of the image. $\phi = \gamma - \psi$

$\phi_1 = \gamma_1 - \psi$ $\phi_1 = \begin{bmatrix} 1-1 \\ -2-\frac{1}{2} \\ 1-0 \\ -3+\frac{1}{2} \end{bmatrix}$; $\phi_1 = \begin{bmatrix} 0 \\ \frac{-5}{2} \\ 1 \\ \frac{-5}{2} \end{bmatrix}$ [2][1]

$\phi_2 = \gamma_2 - \psi$ $\phi_2 = \begin{bmatrix} 1-1 \\ 3-\frac{1}{2} \\ -1-0 \\ 2-\frac{1}{2} \end{bmatrix}$;

$\phi_2 = \begin{bmatrix} 0 \\ \frac{5}{2} \\ -1 \\ \frac{5}{2} \end{bmatrix}$ [2][1] These new vectors indicates the distance from the original matrix or image from the average mean face. This will be later useful for comparison. Subtracting the average face from each of the images in the dataset is a preparatory step that enhances the performances of PCA in capturing the most significant facial features. This process not only optimize the facial recognition algorithm for accuracy but also ensures efficient processing by focusing on the most relevant data features.

12 Covariance Matrix C

A covariance matrix expresses the covariance(Joint Variability) between pairs of variables in a dataset. For a dataset of N dimensions, the covariance matrix is an $N \times N$ matrix where each element (i,j) represents the covariance between the i_th and j_th dimensions of the dataset.[1]

In the context of images, if you consider each pixel values as a dimension, then the covariance matrix will tell you about the extent to which corresponding pixel values of different images vary from the mean with respect to each other.[1][4]

$C = \frac{1}{M}AA^T$

Here A is a matrix that is made of with ϕ_1 and ϕ_2 from the previous step. Therefore our matrix A will be,

$A = \begin{bmatrix} 0 & 0 \\ -\frac{5}{2} & \frac{5}{2} \\ 1 & -1 \\ -\frac{5}{2} & \frac{5}{2} \end{bmatrix}$ [2] Again the transpose of the a 4×2 matrix is 2×4 matrix. Therefore A^T will be,

$A^T = \begin{bmatrix} 0 & -\frac{5}{2} & 1 & -\frac{5}{2} \\ 0 & \frac{5}{2} & -1 & \frac{5}{2} \end{bmatrix}$ [2] Now for simplify we calculate A^TA, matrix size $(M \times M)$. Consider eigenvectors v_i of $A^T A$, so we can say $(A^T A)\ v_i = \mu_i v_i$ As it said before v_i is the eigenvector and now mu_i is the eigenvalue.[6][1] now we premultiply A on both sides, where we get $(A^T A)u_i = \lambda_i u_i$. Here $Av_i = u_i$ and $\lambda_i = \mu_i$ It's important to keep in mind that eigenvalue of AA^T and $A^T A$ are equal. It is because even though $A^T A$ and AA^T may look different and different matrix but they stem from same underlying single value of A. That's why their eigenvalue will be equal.[1][6][3]

$AA^T = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{25}{2} & -5 & \frac{25}{2} \\ 0 & -5 & 2 & -5 \\ 0 & \frac{25}{2} & -5 & \frac{25}{2} \end{bmatrix}$ [2] Therefore the covariance matrix will be,

$$C = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 6.25 & -2.5 & 6.25 \\ 0 & -2.5 & 1 & -2.5 \\ 0 & 6.25 & -2.5 & 6.25 \end{bmatrix} \tag{1}$$

[2]

This is the covariance matrix and we calculated this with the help of eigenvector calculator. The covariance matrix C reflects the covariance of these deviation vectors. It captures how pixel values of the images co-vary with each other cross the entire dataset. Mathematically, if A is the matrix whose columns are the vectors ϕ_i, then the covariance matrix is computed by $C = \frac{1}{M}AA^T$, where M is the number of images, and A^T is the transpose of A

13 Calculating Eigenvalues and Eigenvectors

we have to calculate the determinants from the covariance matrix in order to determine the eigenvalue. $A - \lambda I = \begin{bmatrix} -\lambda & 0 & 0 & 0 \\ 0 & 6.25-\lambda & -2.5 & 6.25 \\ 0 & -2.5 & 1-\lambda & -2.5 \\ 0 & 6.25 & -2.5 & 6.25-\lambda \end{bmatrix}$ [2]

$det(A - \lambda I) = 0$ solving the determinants will provide us the value of the eigenvalues. From which we will also calculate the eigenvectors later by putting the values we gain from the determinants. [1] $\lambda = 0$;

$\lambda = \frac{27}{2}$ [2] solving the equations with these eigenvalues we get the eigenvectors. Here for calculation we first put the eigenvalues in the place of λ and then we use the Gaussian elimination to conclude into a solution and get our final vector which we called the eigenvector. [1] After conducting the calculations we came with the required eigenvectors. The vectors : $\lambda = 0$ $V_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

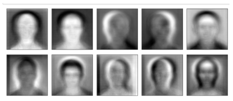


Figure 2: Eigenfaces

$$\lambda = 0 \; V_2 = \begin{bmatrix} 0 \\ \frac{2}{5} \\ 1 \\ 0 \end{bmatrix} \; \lambda = 0 \; V_3 = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix} \; \lambda = \frac{27}{2} \; V_4 = \begin{bmatrix} 0 \\ 1 \\ -\frac{2}{5} \\ 1 \end{bmatrix}$$ [1][2]

14 Normalize eigenvectors

We normalize the vectors to ensure each vectors has unit length, means they all have same scale. This is crucial because it helps to compare the vectors fairly, focusing on their direction not on the magnitude.[1][6] In the context of eigenfaces, normalization ensures that each eigenvectors contributes equally in the facial representation, making the data based on the direction of the data's variablility, not the scale.[1]

The normalization seems like $\mathbf{u} = \frac{\mathbf{v}}{||\mathbf{v}||}$ [1] So after the calculating the magnitude the of the eigenvector we can compute the four normalize eigenvectors as :

$$\mathbf{u_1} = 1 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \; \mathbf{u_2} = \frac{5}{\sqrt{29}} \begin{bmatrix} 0 \\ \frac{2}{5} \\ 1 \\ 0 \end{bmatrix} \; \mathbf{u_3} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix} \; \mathbf{u_4} = \frac{5}{\sqrt{54}} \begin{bmatrix} 0 \\ 1 \\ -\frac{2}{5} \\ 1 \end{bmatrix}$$ [2][3]

So now we can see the normalize vectors $\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}, \mathbf{u_4}$. So this are the eigenfaces of the image we have. Now we are going to reconstruct the image with this eigenfaces, mean faces of the training set and with weight contribution of each eigenfaces.

In recognition, this normalizing ensures that when a new face is projected onto the space of eigenfaces, the similarity measurements and subsequent classification are not biased by the scale of the eigenfaces but rather by the actual content and structure of the face data. This helps accurately identifying and recognizing faces by their essential features.

15 Reconstructing

Here we going to reconstruct the vector I_i. this looks like :

$$\gamma = \psi + [\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3}, \mathbf{u_4}] \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}$$ Here w is the contribution of each eigenfaces and later we a re going to see how to calculate w. Again the main of reconstruction of the matrix is to use the compact representation provided by eigenfaces.

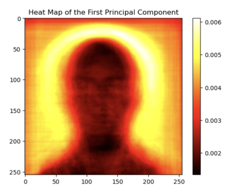


Figure 3: Heat Map of the First Principal Component

so any images we reconstruct it's going to look like this, $\gamma = \psi + \sum_{i_1}^{M} u_i w_i$ we need to remember about $m = k$ we select k most important eienvector for k largest eigenvalues.

Now here comes the weight of the vectors which is, $w_k = u_k^T \phi$; $(\phi = \gamma - \psi)$ With ϕ_1 we get weights of I_1 as: $\omega_1 = \begin{bmatrix} 0 & 0 & 0 & -5.1031 \end{bmatrix}$ [2] $\omega_2 = \begin{bmatrix} 0 & 0 & 0 & 5.1031 \end{bmatrix}$ [2] Now lets reconstruct the I_1. $\gamma_1^{new} = \psi + u_1 w_1 + u_2 w_2 + u_3 w_3 + u_4 w_4$

Similarly for second image we construct I_2 $\gamma_2^{new} = \psi + u_1 w_1 + u_2 w_2 + u_3 w_3 + u_4 w_4$ [1][2][6]

The vale for γ_1^{new} is ; $\gamma_1^{new} = \begin{bmatrix} 1 \\ -2.9722 \\ 1.38889 \\ -3.97222 \end{bmatrix}$

The vale for γ_2^{new} is ; $\gamma_2^{new} = \begin{bmatrix} 1 \\ 2.9722 \\ -1.38889 \\ 3.97222 \end{bmatrix}$ [2][3][1]

Now here we can see there's difference in the new reconstructed image and the actual image we have in the beginning. So now we can calculate the root mean error which is basically the difference between the new reconstructed and real image.[1]

$|\gamma_1^{new} - \gamma_1| = \begin{bmatrix} 0 \\ -0.97222 \\ 0.38889 \\ -6.97222 \end{bmatrix}$

$|\gamma_2^{new} - \gamma_2| = \begin{bmatrix} 0 \\ -0.02778 \\ -0.38889 \\ 1.97222 \end{bmatrix}$ [1][2]

16 Recognition of Test Image

For recognition of test image we going to take $\gamma_n ew$ against the training dataset we have first we going to compute the weight of test image according to the formula. $w_{new} = u_k^T \phi_{new}$ note that $\phi_n ew = \gamma_{new} - \psi$ here γ_{new} is the text

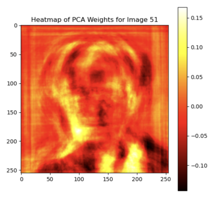


Figure 4: Heatmap of PCA weights for Image 51

image. So now we can say $\omega_{new} = [w_1, w_2, w_3, w_4]$. Here's how a testing image recognized :

16.1 1

First the testing image is normalized by subtracting the mean face just like the training image and then projected onto the eigenface space. [3]

16.2 2

The face recognition typically performed by comparing the weight vector of the testing image with the weight vector of the training image. This can be done through different distance measure like Euclididan distance. Euclidian distance e_d between weight matrix $\omega_n ew$ and each face class ω_k.[3][6]

16.3 3

The testing image is usually recognized as the face whose weight vector in the training set is closest to it according to the chosen distance measure. The final step is to interpret the result of the comparison.This process allows the eigenface system to recognize and verify faces efficiently, even with a relatively low-resolution input, by focusing on the most distinguishing features represented through the principal components.[6][5]

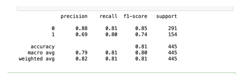
16.4 The mathematical process of reconstruction

Reconstructed Images $= \psi + \sum_{i=1}^{k} w_i u_i$.

Here the ψ is the mean face vector that we have calculated earlier, u_i is the eigenfaces that we just calculated earlier and w_i are the weights calculated for a particular image.

17 Result

Our study incorporated the Eigenfaces technique, which utilizes PCA to efficiently recognize the facial features by transforming high dimensional facial image data to a lower dimensional space.Each image in the training set known



```
              precision    recall  f1-score   support

           0       0.88      0.81      0.85       291
           1       0.69      0.80      0.74       154

    accuracy                           0.81       445
   macro avg       0.79      0.81      0.80       445
weighted avg       0.82      0.81      0.81       445
```

Figure 5: Output of our Model

as γ_i, the avaerage face vector, computed as the mean of these vectors. The difference vectors, $\psi_i = \gamma_i - \phi_i$.

Our model was trained using these eigenfaces, transforming each image into this new eigenface space using the PCS transformation matrix obtained from training data. The resulting lower-dimensional representations(features) were then used to train a SVM classifier.

The classification performance was assessed using a testing dataset, and the results were quantitatively summarized by precision, recall, and F1-score metrics:

Non-Black Offender Class (0): Precision of 0.88 and recall of 0.81 resulted in an F1-score of 0.85. Black Offender Class (1): Precision of 0.69 and recall of 0.80 resulted in an F1-score of 0.74.

Precision measures the accuracy of the positive predictions made by a classifier. It is the ratio of true positive results to all positive results predicted by the model, including both true positives and false positives.

Recall, also known as sensitivity, measures the ability of a classifier to identify all actual positives. It is the ratio of true positives to the total actual positives, which includes both true positives and false negatives.

F1-score This score is the harmonic mean of precision and recall, providing a single metric that balances both the precision and the recall. An F1-score reaches its best value at 1 (perfect precision and recall) and worst at 0.

18 Conclusion

Study demonstrates the mathematical robustness and practical effectiveness of eigenface method in facial recognition tasks. Specially making an mathematical explainable model to identify black and non-black people from 2D grayscale images.By employing PCA we reduced the dimensionality of the facial image data, enabling more efficient processing and analysis. The SVM classifier, trained on these lower-dimensional representations, effectivley differentiated between classes, although with varying levels of precision and recall.

Mathematical foundations laid out this paper - from the calculation of mean face vectors to the derivation of eigenvalues and eigenvectors - highlight the interplay between linear algebra and statistical methods in the realm of image processing and recognition.

19 Reference

1) ”Face Recognition Algorithm using Eigenfaces” Uploaded by ICT UoM (Apr 12, 2021). (Youtube)

2) Wolframe Alpha

3) Matthew A Turk and Alex P. Pentland ”Face Recognition Using Eigenfaces”.

4) NIST Mugshot Dataset https://www.kaggle.com/datasets/kwisatzhaderach/nist-mugshots

5) Theodoros Evgeniou and Massimiliano Pontil Center for Biological and Computational Learning, and Artificial Intelligence Laboratory, ”WORKSHOP ON SUPPORT VECTOR MACHINES: THEORY AND APPLICATIONS”.

6) Lindsay I Smith Department of Computer Science, University of Otago, New Zealand ”A tutorial on Principal Components Analysis”