Lecture 13: Face Recognition and LDA

JR Cabansag, Yuxing Chen, Jonathan Griffin, Dunchadhn Lyons, George Preudhomme
Department of Computer Science
Stanford University
Stanford, CA 94305
{cabansag, yxchen28, jgriffi2, dunlyons, gpreud}@cs.stanford.edu

1 Introduction to Facial Recognition

1.1 Neuroscience Background

In the 1960's and 1970's, neuroscientists discovered that, depending on the angle of observation, certain neurons in the brain fire when looking at a face. More recently, they have to come to believe that an area of the brain known as the **Fusiform Face Area (FFA)** is primarily responsible for reacting to faces. These advances in the biological understanding of facial recognition have been mirrored by similar advances in computer vision, as new techniques have attempted to come closer to the standard of human facial recognition.

1.2 Applications

Computer facial recognition has a wide range of applications:

- Digital Photography: Identifying specific faces in an image allows programs to respond uniquely to different individuals, such as centering the image focus on a particular individual or improving aesthetics through various image operations (blur, saturation, etc).
- Surveillance: By recognizing the faces of specific individuals, we can use surveillance cameras to detect when they enter a location.
- Album Organization: If we can recognize a specific person, we can group images in which they appear.
- Person tracking: If we can recognize a specific person, we can track their location through frames of video (useful for surveillance or for robots in the home).
- Emotions and Expressions: By detecting emotions or expressions in faces, we can build smart devices that interact with us based on mood.
- Security and Warfare: If we can recognize a specific person, we can identify enemies of interest in drone images and video.
- Teleconferencing: If we can recognize specific people, then teleconferencing applications could automatically provide information to users about who they are communicating with.

1.3 A Key Distinction: Detection vs. Recognition

While face **Detection** entails determining whether an image contains a face and where in the image the face exists, face **Recognition** entails determining whose face an image contains.

1.4 Space of Faces

If we consider an $m \times n$ image of a face, that image can be represented by a point in high dimensional space ($\mathbb{R}^{mn}$). But relatively few high-dimensional vectors consist of valid face images (images can contain much more than just faces), and thus the region that an arbitrary face image could fall into is a relatively small subspace. The task is to effectively model this subspace of face images.

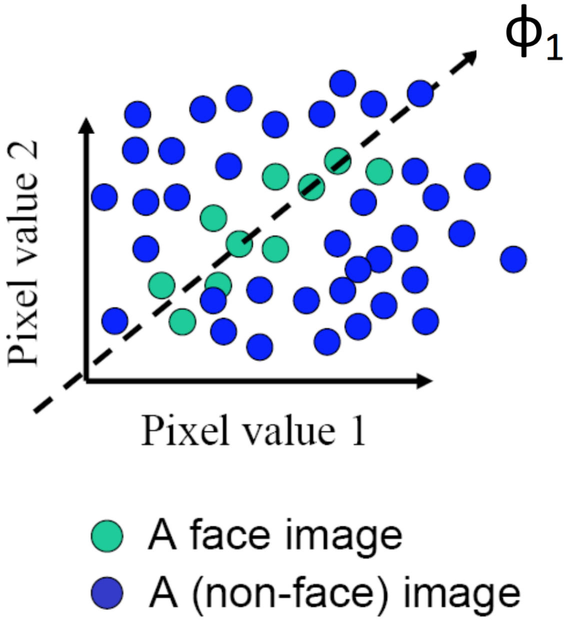

Figure 1: The region occupied by images of faces is a small subspace of the total space of images. Source: Lecture 13, slide 14

In order to model this subspace or "face-space" we compute the k-dimensional subspace such that the projection of the data points onto the subspace has the largest variance among all k-dimensional subspaces. This low-dimensional subspace captures the key appearance characteristics of faces.

2 The Eigenfaces Algorithm

2.1 Key Ideas and Assumptions

- Assume that most face images lie on a low-dimensional subspace determined by the first k directions of maximum variance.

- Use Principle Components Analysis (PCA) to determine the vectors or “eigenfaces” that span that subspace.

- Represent all face images in the dataset as linear combinations of eigenfaces, where eigenfaces are sort of defined as the principal components of SVD decomposition.

2.1.1 What are eigenfaces?

"Eigenfaces" are the visual representations of the eigenvectors in the directions of maximum variance. They often resemble generic-looking faces.

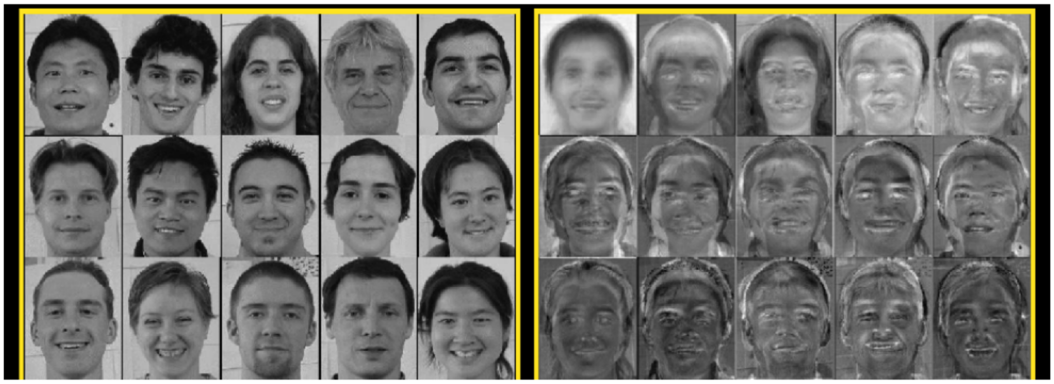

Figure 2: Faces and Eigenfaces. Source: Lecture 13, slide 29

2.2 Training Algorithm

Algorithm 1 Eigenfaces Training Algorithm [2]

```
Align training images $x_1, ..., x_n$
Compute the average face:
    $$\mu = \frac{1}{N} \sum x_i$$
Compute the difference image (the centered data matrix):
    $$X_c = X - \mu 1^T = X - \frac{1}{N} X 1 1^T = X(1 - \frac{1}{N} 1 1^T)$$
Compute the covariance matrix:
    $$\Sigma = \frac{1}{N} X_c {X_c}^T$$
Compute the eigenvectors of the covariance matrix $\Sigma$ using PCA (Principle Components Analysis)
Compute each training image $x_i$'s projections as
    $$x_i \rightarrow ({x_i}^c \cdot \phi_1, {x_i}^c \cdot \phi_2, ..., {x_i}^c \cdot \phi_k) \equiv (a_1, a_2, ..., a_k)$$
    where $\phi_i$ is the $i$'th-highest ranked eigenvector
The reconstructed face $x_i \approx \mu + a_1 \cdot \phi_1 + ... + a_k \cdot \phi_k$
```

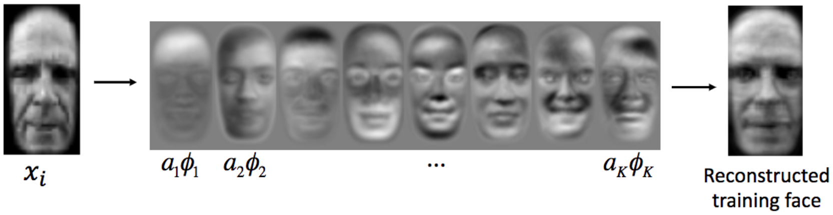

Figure 3: The reconstructed face after projection. Source: Lecture 13, slide 25

2.2.1 Why can we do this?

Empirically, the eigenvalues (variance along eigenvectors) drop rapidly with the number of principle components, which is why we can reduce dimensionality without much loss of information.

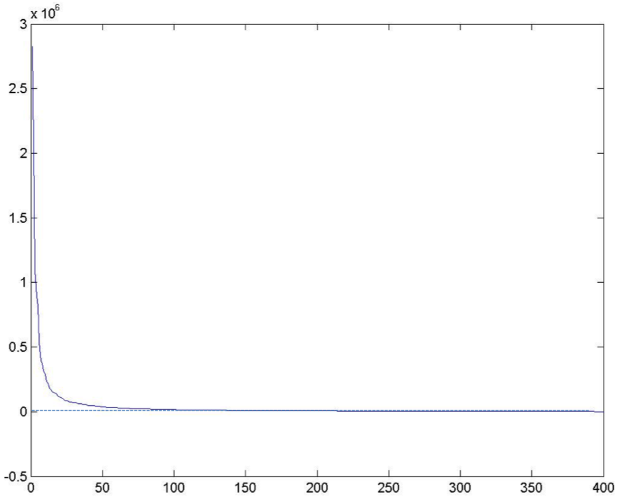

Figure 4: Eigenvalues sorted in descending order of magnitude. Source: Lecture 13, slide 26

2.2.2 Reconstruction and Error

We only select the top k eigenfaces, which reduces the dimensionality. Fewer eigenfaces result in more information loss, and hence less discrimination between faces.

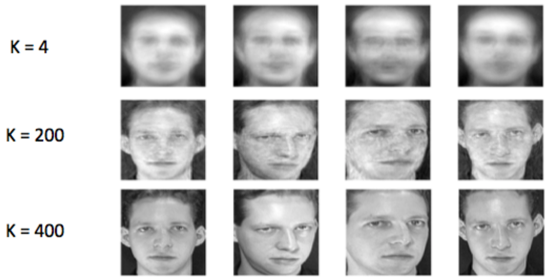

Figure 5: Reconstructed faces with varying number of eigenfaces. Source: Lecture 13, slide 27

2.3 Testing Algorithm

Algorithm 2 Eigenfaces Testing Algorithm [2]

```
Take query image $t$
Project onto eigenvectors:
    $$t \rightarrow ((t-\mu)\cdot\phi_1, (t-\mu)\cdot\phi_2, ..., (t-\mu)\cdot\phi_k) \equiv (w_1, w_2, ..., w_k)$$
Compare projection $w$ with all $N$ training projections. Use euclidean distance and nearest-neighbors algorithm to output a label
```

2.4 Advantages

- This method is completely knowledge-free – it does not know anything about faces, expressions, etc.
- It is a non-iterative (fast), globally-optimal solution.

2.5 Disadvantages

- This technique requires carefully controlled data.

  1. All faces must be centered in the frame. Otherwise the results may be noisy.
  2. The images must be the same size.
  3. There is some sensitivity to the face angle.

- Method is completely knowledge free.

  1. It makes no effort to preserve class distinctions.
  2. PCA doesn't take into account who it is trying to represent in this lower dimensional space (it doesn't take into account the labels associated with the faces). Therefore, it might map different faces near the same subspace, making it difficult for classifiers to distinguish between them.

- PCA projection is optimal for reconstruction from a low dimensional basis but may not be optimal for discrimination (the algorithm does not attempt to preserve class distinctions).

2.6 Beyond Facial Recognition: Expressions and Emotions

This technique also generalizes beyond simple facial recognition and can be used to detect expressions and emotions. The subspaces would therefore represent happiness, disgust, or other potential expressions, and the algorithm would remain unchanged.

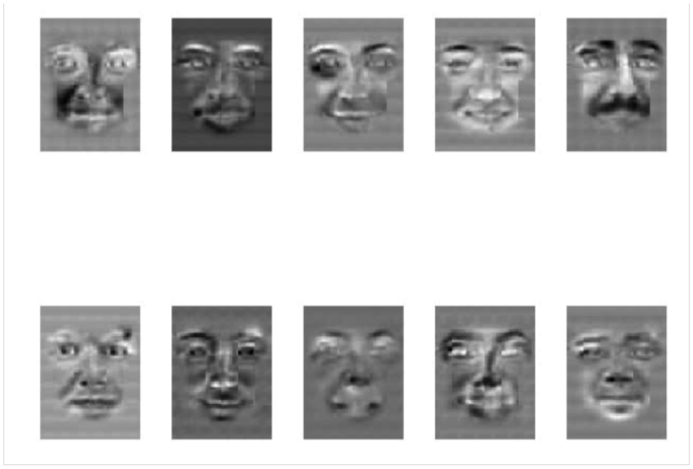

Figure 6: Eigenfaces expressing happiness. Source: Lecture 13, slide 33

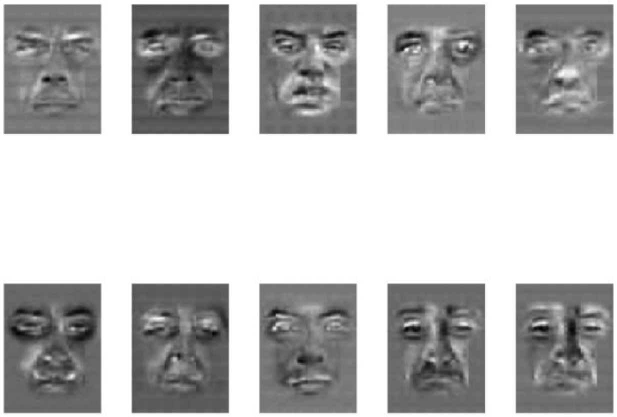

Figure 7: Eigenfaces expressing disgust. Source: Lecture 13, slide 34

3 Linear Discriminant Analysis

3.1 PCA vs. LDA

PCA and LDA are similar in that both reduce the dimensions of a sample. However, PCA projections don't consider the labels of the classes. An alternative approach is to move away from PCA toward an algorithm that is optimal for classification (as opposed to reconstruction). Linear Discriminant Analysis (LDA) finds a projection that keeps different classes far away from each other.

- PCA maintains maximum variance.
- LDA allows for class discrimination by finding a projection that maximizes scatter between classes and minimizes scatter within classes.

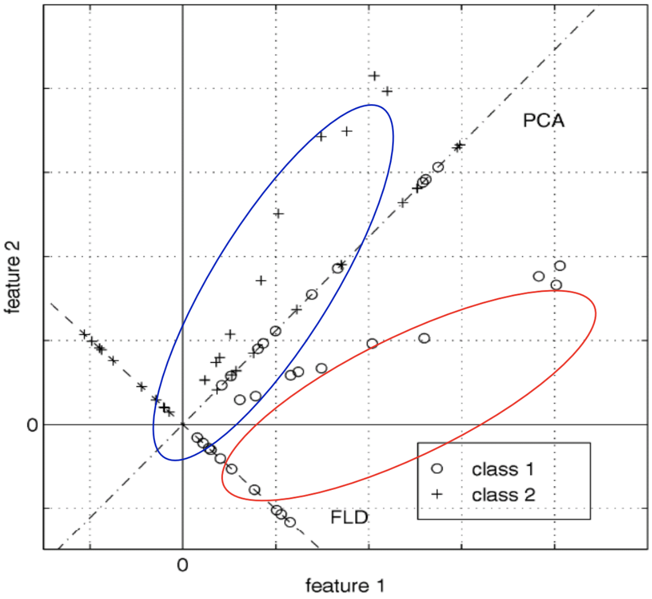

Figure 8: PCA vs. LDA. Source: Lecture 13, slide 41

The difference between PCA and LDA projections is demonstrated in the figure above. PCA preserves the maximum variance and maps the points of the classes along the line with the positive slope, which makes it difficult to distinguish a points' class. Meanwhile, LDA maps the points onto the line with the negative slope, which results in points being located close to other points in their class and far from points in the opposite class.

3.2 General Idea

LDA operates using two values: between class scatter and within class scatter. Between class scatter is concerned with the distance between different class clusters, whereas within class scatter refers to the distance between points of a class. LDA maximizes between class scatter and minimizes within class scatter.

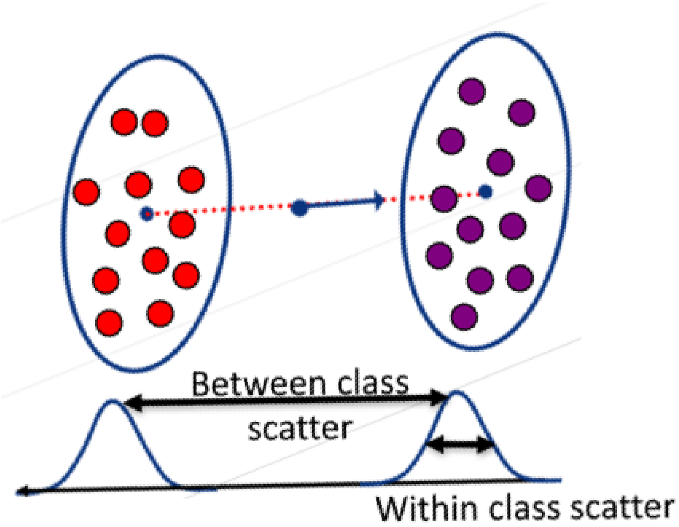

Figure 9: Between Class Scatter vs. Within Class Scatter. Source: Lecture 13, slide 43

3.3 Mathematical Formulation of LDA with 2 Variables

We want to find a projection w that maps points with classes 0 and 1 in the space $x \in R^n$ to a new space $z \in R^m$, such that $z = w^T x$. Ideally, $m < n$, and our projection should maximize the function:

$$J(w) = \frac{S_B \text{ when projected onto } w}{S_W \text{ when projected onto } w}$$

In this equation, S_B represents between class scatter and S_W represents within class scatter. Let us then define a variable μ_i that represents the mean of a class' points:

$$\mu_i = E_{X|Y}[X|Y = i]$$

Let us also define a variable Σ_i that represents the covariance matrix of a class:

$$\Sigma_i = E_{X|Y}[(X - \mu_i)(X - \mu_i)^T|Y = i]$$

Using these values, we can redefine our variables S_B and S_W to be:

$$S_B = (\mu_1 - \mu_0)^2 = (\mu_1 - \mu_0)(\mu_1 - \mu_0)^T$$
$$S_W = (\Sigma_1 + \Sigma_0)$$

Plugging these new values of S_B and S_W back into $J(w)$, we get:

$$J(w) = \frac{(\mu_1 - \mu_0)^2 \text{ when projected onto } w}{(\Sigma_1 + \Sigma_0) \text{ when projected onto } w} = \frac{w^T(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w}{w^T(\Sigma_1 + \Sigma_0)w}$$

We can maximize $J(w)$ by maximizing the numerator, $w^T(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w$, and keeping its denominator, $w^T(\Sigma_1 + \Sigma_0)w$ constant. In other words:

$$\max w^T(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w \text{ subject to } w^T(\Sigma_1 + \Sigma_0)w = K$$

Using Lagrange multipliers, we can define the Lagrangian as:

$$L = w^T S_B w - \lambda(w^T S_W w - K) = w^T(S_B - \lambda S_W)w + K$$

We must then maximize L with respect to λ and w. We can do so by taking its gradient with respect to w and finding where the critical points are:

$$\nabla_w L = 2(S_B - \lambda S_W)w = 0$$

Using this equation, we get that the critical points are located at:

$$S_B w = \lambda S_W w$$

This is a generalized eigenvector problem. In the case where $S_W^{-1} = (\Sigma_1 + \Sigma_0)^{-1}$ exists, we obtain:

$$S_W^{-1} S_B w = \lambda w$$

We can then plug in our definition of S_B to get:

$$S_W^{-1}(\mu_1 - \mu_0)(\mu_1 - \mu_0)^T w = \lambda w$$

Notice that $(\mu_1 - \mu_0)^T w$ is a scalar, and thus we can represent it by a term α such that:

$$S_W^{-1}(\mu_1 - \mu_0) = \frac{\lambda}{\alpha} w$$

The magnitude of w does not matter, so we can represent our projection w as:

$$w* = S_W^{-1}(\mu_1 - \mu_0) = (\Sigma_1 - \Sigma_0)^{-1}(\mu_1 - \mu_0)$$

3.4 LDA with N Variables and C Classes

3.4.1 Preliminaries

Variables:

- N sample images: $\{x_1, \cdots, x_N\}$
- C classes: $\{Y_1, Y_2, \cdots, Y_C\}$. Each of the N sample images is associated with one class in $\{Y_1, Y_2, \cdots, Y_C\}$.
- Average of each class: the mean for class i is $\mu_i = \frac{1}{N_i} \sum\limits_{x_k \in Y_i} x_k$
- Average of all data: $\mu = \frac{1}{N} \sum\limits_{k=1}^{N} x_k$

Scatter Matrices:

- Scatter of class i: $S_i = \sum_{x_k \in Y_i} (x_k - \mu_i)(x_k - \mu_i)^T$
- Within class scatter: $S_w = \sum_{i=1}^{c} S_i$
- Between class scatter: $S_b = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T$

3.4.2 Mathematical Formulation

We want to learn a projection W such that it converts all the points from $x \in \mathbb{R}^m$ to a new space $z \in \mathbb{R}^n$, where in general m and n are unknown:

$$z = w^T x, x \in \mathbb{R}^m, z \in \mathbb{R}^n$$

After the projection, the between class scatter is $\widetilde{S}_B = W^T S_B W$, where W and S_B are calculated from our original dataset. The within class scatter is $\widetilde{S}_W = W^T S_W W$. So the objective becomes:

$$W_{opt} = \underset{W}{\operatorname{argmax}} \frac{\left|\widetilde{S}_B\right|}{\left|\widetilde{S}_W\right|} = \underset{W}{\operatorname{argmax}} \frac{|W^T S_B W|}{|W^T S_W W|}$$

Again, after applying Lagrange multipliers we obtain a generalized eigenvector problem, where we have:

$$S_B w_i = \lambda_i S_W w_i, i = 1, \ldots, m$$

Note that $\operatorname{Rank}(S_B)$ and $\operatorname{Rank}(S_W)$ are limited by the number of classes (C) and the number of sample images (N):

$$\operatorname{Rank}(S_B) \leq C - 1$$

$$\operatorname{Rank}(S_W) \leq N - C$$

And therefore the rank of W_{opt} is limited as well.

3.5 Results: Eigenface vs. Fisherface

Belhumeur, Hespanha, Kriegman performed an experiment comparing the recognition error rates of PCA to LDA ("Eigenface vs Fisherface") using a dataset of 160 images of 10 distinct people. The images contained significant variation in lighting, facial expressions, and eye-wear. Error rates were determined using the "leaving-one-out" strategy, where a single image was classified by removing that image from the dataset and training on the other 159 images, at which point classification was done on the left-out image with a nearest-neighbors classifier. This process was repeated over all 160 images in order to determine an error rate [1].

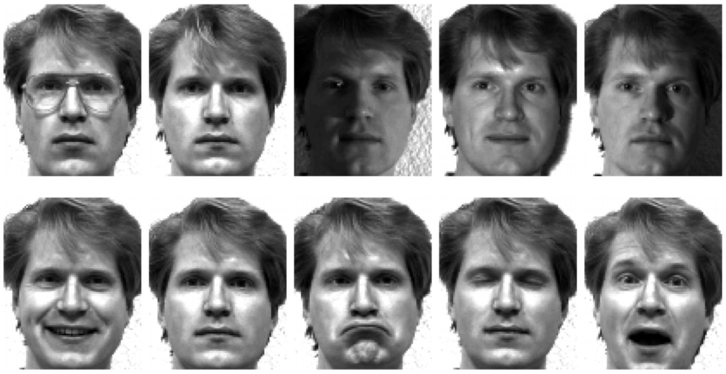

Figure 10: Variation in Facial Expression, Eyewear, and Lighting. Source: [1]

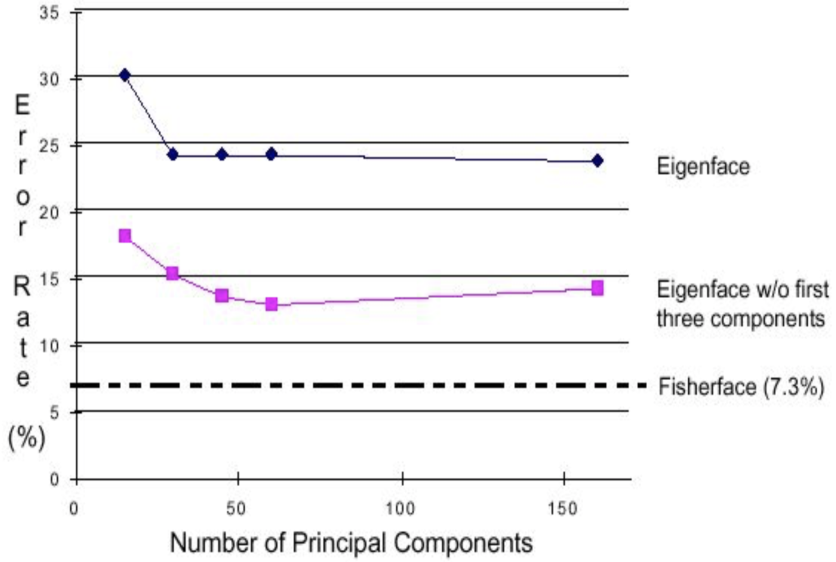

Figure 11: Eigenface vs. Fisherface. Source: Lecture 13, slide 61

Error rates for the two algorithms (and a variation of standard Eigenface) are plotted in the figure above. Eigenface's error rate actually improves by removing the first three prinicple components. Fisherface, as shown in the figure above, gives the lowest error rate.

References

[1] J. Hespanha P. Belhumeur and D. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on pattern analysis and machine intelligence*, 19(7):711–720, 1997.

[2] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.