Face Recognition

Face Recognition Scenarios

- **Face verification** – Am I who say I am?

  – a one-to-one match that compares a query face image against a template face image whose identity is being claimed

- **Face identification** – Who am I?

  – a one-to-many matching process that compares a query image against all template images in a face database to determine the identity of the query face

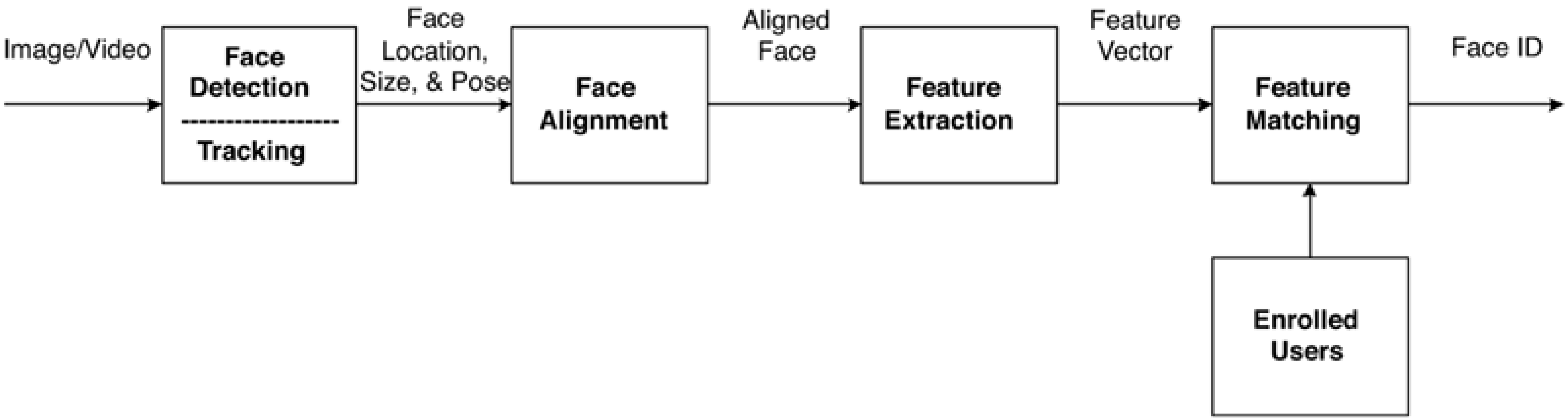
  – a similarity score is found for each comparison

- **Watch list** – Are you looking for me?

  – is an open-universe test, the test individual may or may not be in the system database

  – perform face identification first and rank similarity scores, *if the highest similarity score is higher than a preset threshold, an alarm is raised*

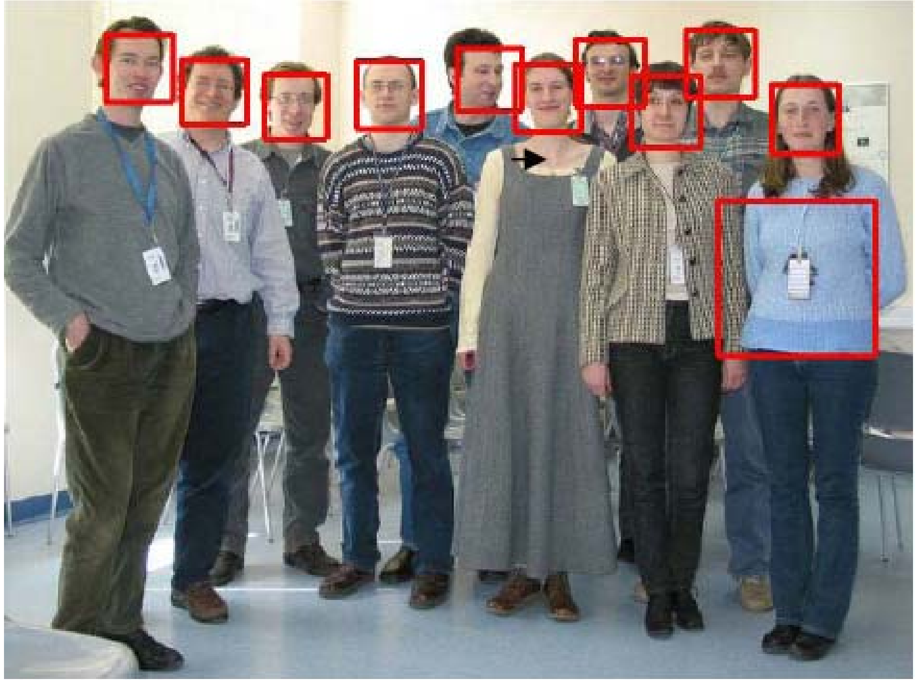
Face Recognition System Block Diagram

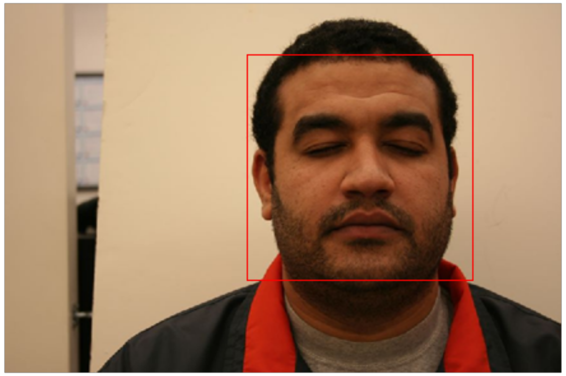


Face detection:

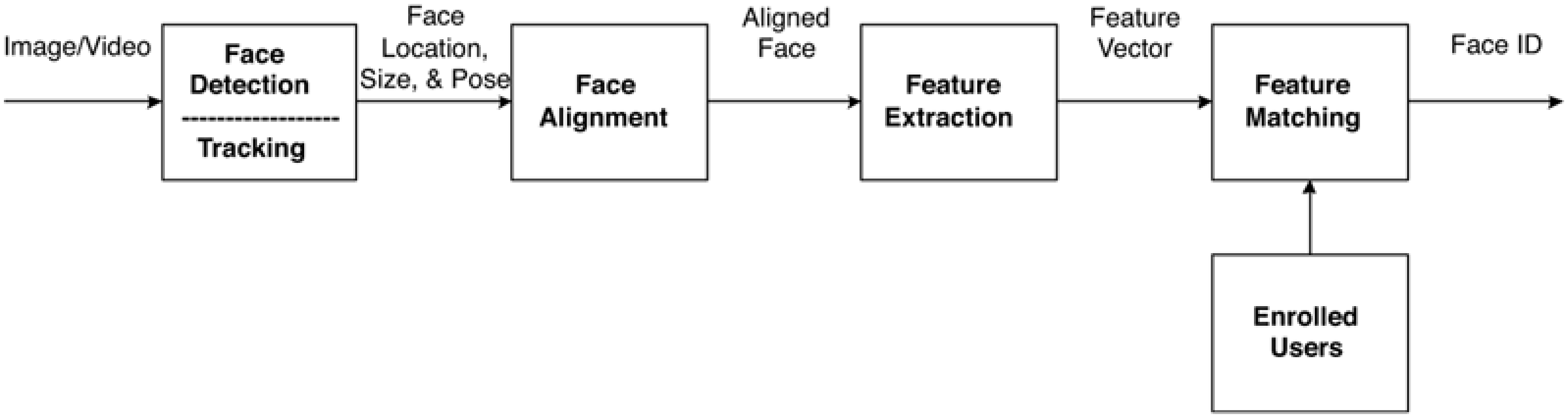
Given a single image or video, an ideal face detector must be able to identify and locate all the present faces regardless of their *position*, *scale*, *orientation*, *age*, and *expression*.

Face Detection: Intel OpenCV Implementation



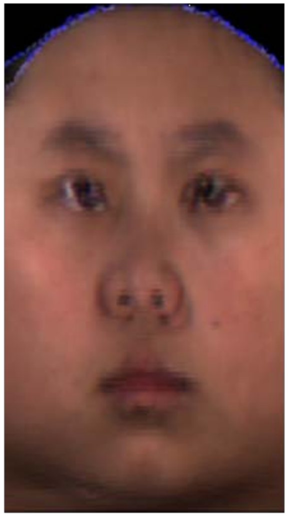


Face Alignment

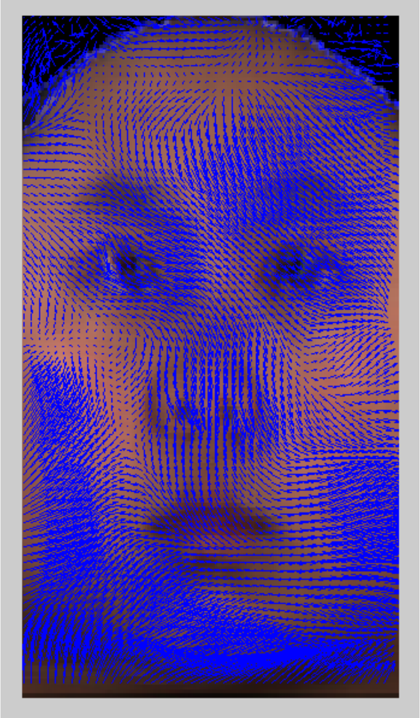


In *face alignment*, facial components, such as eyes, nose, and mouth, and facial outline are located, and thereby the input face image is normalized in geometry and photometry.

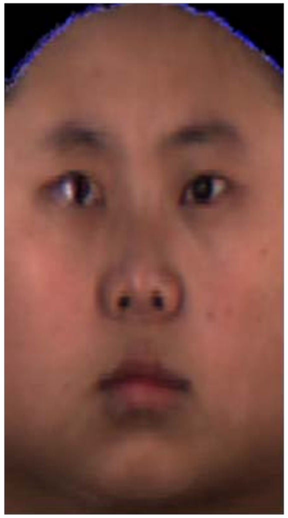
Face Alignment using a Modified Optical Flow: Results



Two images averaged w/o correspondence

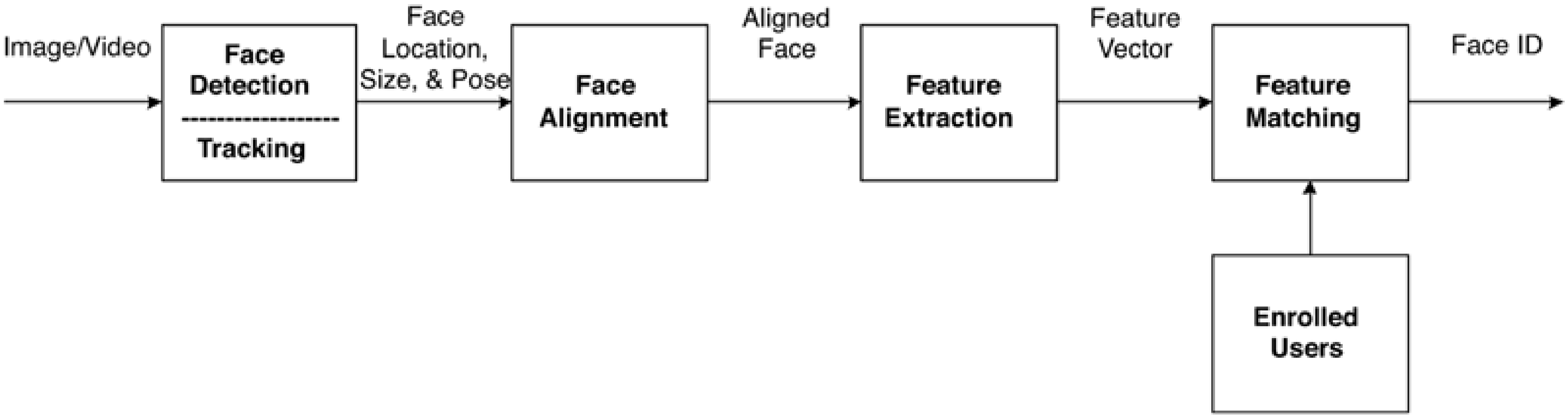


Optical flow



Two images averaged w/ correspondence

Feature Extraction/Matching



Feature Extraction/Matching: Linear Subspace Approaches

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- Independent Component Analysis (ICA)

Face Recognition: Challenges

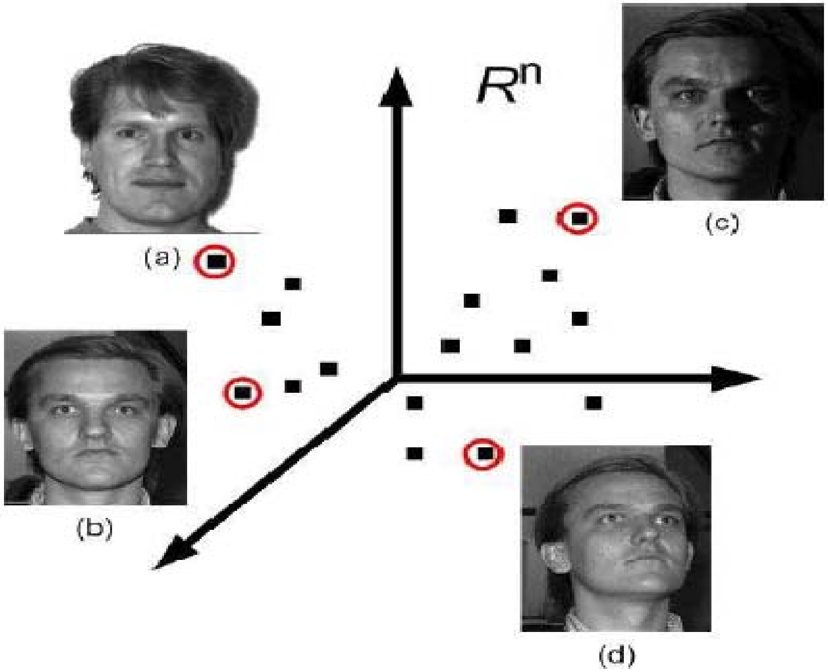


Figure: *Inter-subject* versus *intra-subject* variations. (a) and (b) are images from different subjects, but their appearance variations represented in the input space can be smaller than images from the same subject b, c and d.

Face Recognition in Subspaces (PCA and LDA)

- Vector representation of images – a 2D image data can be represented as vectors

$$I = \begin{bmatrix} 2 & 3 & 4 \\ 4 & 5 & 6 \\ 8 & 2 & 1 \end{bmatrix} \quad I_V = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 4 \\ 5 \\ 6 \\ 8 \\ 2 \\ 1 \end{bmatrix} \quad I_V = \begin{bmatrix} 2 \\ 4 \\ 8 \\ 3 \\ 5 \\ 2 \\ 4 \\ 6 \\ 1 \end{bmatrix}$$

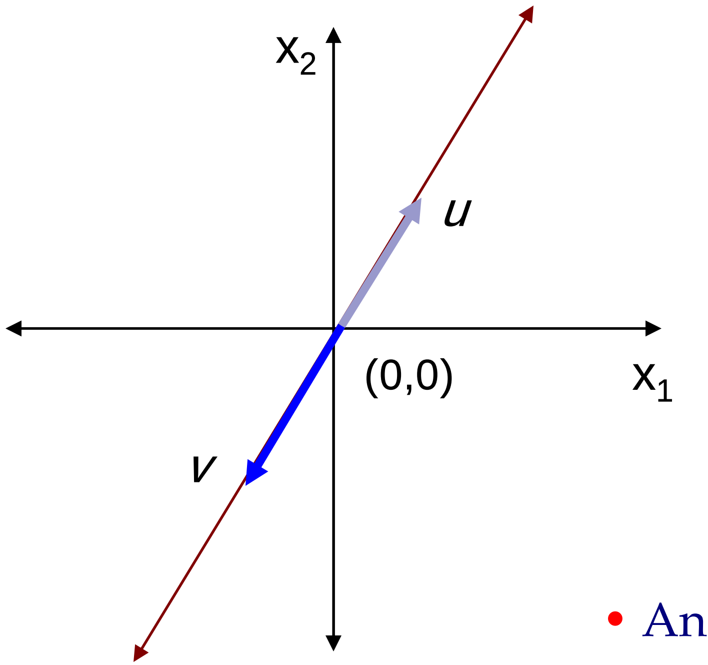
- Consider a (60x50) image – we have a (3000x1) vector

Face Recognition in Subspaces: Image Space vs. Face Space

- the appearance of faces is highly constrained; any frontal view of a face is highly symmetrical, has eyes on the sides, nose in the middle, etc.
- a vast proportion of points in the image space do not represent faces, like the second image above
- therefore, the natural constraints dictate that face images are confined to a subspace, which is referred to as the face space (a subspace of the image space)

Face Recognition in Subspaces: Subspace Illustration

- Consider a straight line in R^2, passing through the origin



- A subset H of a vector space V is a subspace if it is closed under the two operations:

- $u, v \in H \Rightarrow u + v \in H$
- $u \in H, c \in R, \Rightarrow cu \in H$

- Analogy: line – face space, R^2 – image space

Face Recognition: Linear Subspace Analysis

- Three classical linear appearance-based models: PCA, LDA and ICA
- Each model has its own representation (basis vectors) of a high-dimensional face vector space based on different statistical viewpoints.
- All the three representations can be considered as a linear transformation from the original image vector to a projection feature vector

$$Y = W^T X$$

where Y – (d x 1), X – (n x 1) and W – (n x d), $d << n$

Face Recognition: Principal Component Analysis (PCA)

- The objective is to perform dimensionality reduction while preserving as much of the randomness (variance) in the high-dimensional space as possible

$$Y = W^T X$$

- The PCA basis vectors are defined as the eigenvectors of the scatter matrix (S_T)

$$S_T = \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T$$

where x_i – *i* th sample, and μ - sample mean

- The transformation matrix W_{PCA} is composed of the eigenvectors corresponding to the *d* largest eigenvalues

$$\boxed{S_T W = \lambda W}$$

Principal Component Analysis (PCA): Numerical Example

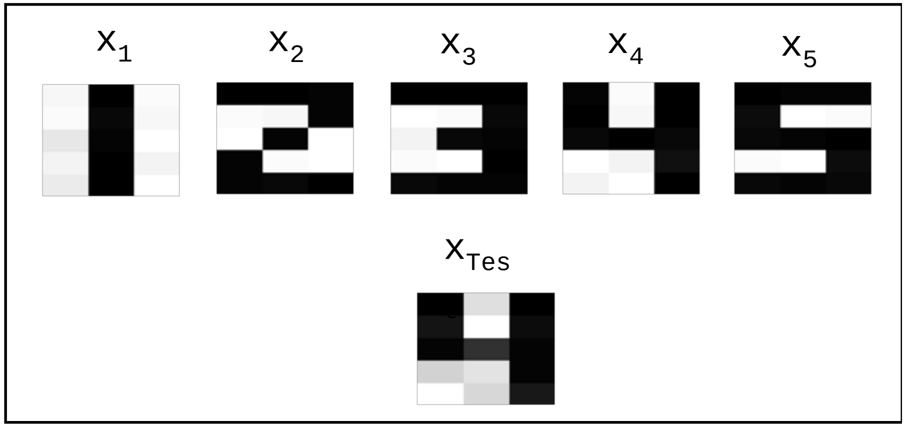
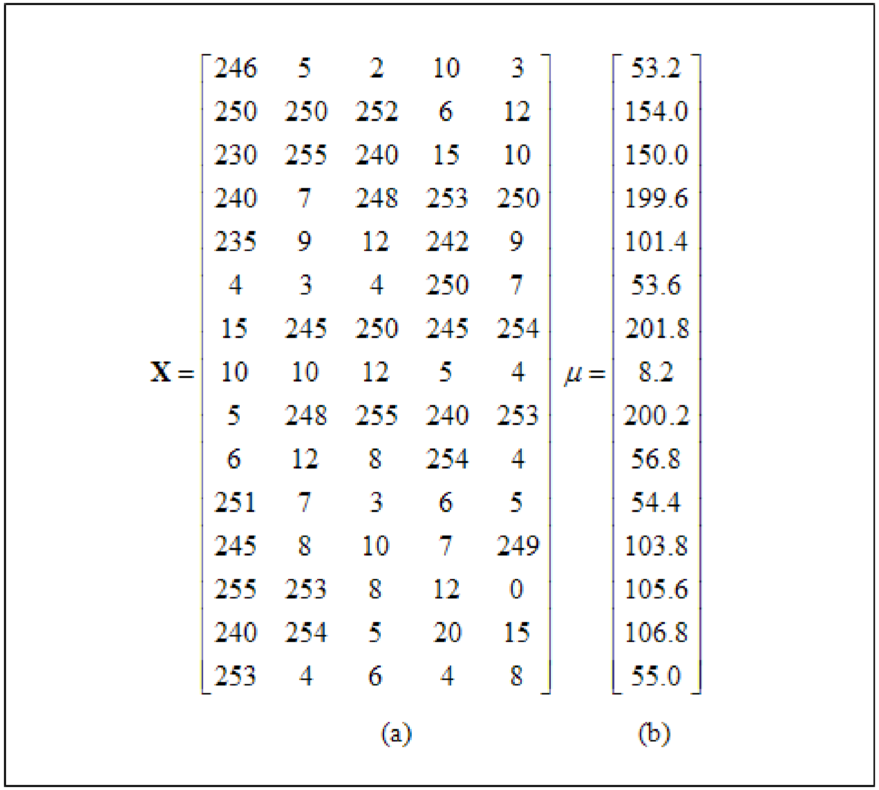


Figure: Train/Test Images for the PCA Numerical Example



$$\mathbf{X} = \begin{bmatrix} 246 & 5 & 2 & 10 & 3 \\ 250 & 250 & 252 & 6 & 12 \\ 230 & 255 & 240 & 15 & 10 \\ 240 & 7 & 248 & 253 & 250 \\ 235 & 9 & 12 & 242 & 9 \\ 4 & 3 & 4 & 250 & 7 \\ 15 & 245 & 250 & 245 & 254 \\ 10 & 10 & 12 & 5 & 4 \\ 5 & 248 & 255 & 240 & 253 \\ 6 & 12 & 8 & 254 & 4 \\ 251 & 7 & 3 & 6 & 5 \\ 245 & 8 & 10 & 7 & 249 \\ 255 & 253 & 8 & 12 & 0 \\ 240 & 254 & 5 & 20 & 15 \\ 253 & 4 & 6 & 4 & 8 \end{bmatrix} \qquad \mu = \begin{bmatrix} 53.2 \\ 154.0 \\ 150.0 \\ 199.6 \\ 101.4 \\ 53.6 \\ 201.8 \\ 8.2 \\ 200.2 \\ 56.8 \\ 54.4 \\ 103.8 \\ 105.6 \\ 106.8 \\ 55.0 \end{bmatrix}$$

(a) (b)

Figure: (a) The data matrix, consisting of pixel values in columns, for the 5 training images, (b) the pixel values of the mean image.

Principal Component Analysis (PCA): Numerical Example

Create Data Matrix:

The (*5x3*) images are mapped into a (15x1) column vector by concatenating each column of the image. The column vectors are combined to form a data matrix of size (*15x5*), since there are 15 pixels in the image and 5 training images. For the example above, the original data matrix (**X**) and the sample mean (**μ**) is shown below.

$$\mathbf{X} = \begin{bmatrix} 246 & 5 & 2 & 10 & 3 \\ 250 & 250 & 252 & 6 & 12 \\ 230 & 255 & 240 & 15 & 10 \\ 240 & 7 & 248 & 253 & 250 \\ 235 & 9 & 12 & 242 & 9 \\ 4 & 3 & 4 & 250 & 7 \\ 15 & 245 & 250 & 245 & 254 \\ 10 & 10 & 12 & 5 & 4 \\ 5 & 248 & 255 & 240 & 253 \\ 6 & 12 & 8 & 254 & 4 \\ 251 & 7 & 3 & 6 & 5 \\ 245 & 8 & 10 & 7 & 249 \\ 255 & 253 & 8 & 12 & 0 \\ 240 & 254 & 5 & 20 & 15 \\ 253 & 4 & 6 & 4 & 8 \end{bmatrix} \qquad \mu = \begin{bmatrix} 53.2 \\ 154.0 \\ 150.0 \\ 199.6 \\ 101.4 \\ 53.6 \\ 201.8 \\ 8.2 \\ 200.2 \\ 56.8 \\ 54.4 \\ 103.8 \\ 105.6 \\ 106.8 \\ 55.0 \end{bmatrix}$$

(a) (b)

Principal Component Analysis (PCA): Numerical Example

Center Data: The training images must be centered by subtracting the mean image from each of the training images. This is an essential preprocessing step for the principal component algorithm. Figure *i* shows the mean-centered data in columns.

$$S_T = \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T$$

$$\mathbf{X} = \begin{bmatrix} 246 & 5 & 2 & 10 & 3 \\ 250 & 250 & 252 & 6 & 12 \\ 230 & 255 & 240 & 15 & 10 \\ 240 & 7 & 248 & 253 & 250 \\ 235 & 9 & 12 & 242 & 9 \\ 4 & 3 & 4 & 250 & 7 \\ 15 & 245 & 250 & 245 & 254 \\ 10 & 10 & 12 & 5 & 4 \\ 5 & 248 & 255 & 240 & 253 \\ 6 & 12 & 8 & 254 & 4 \\ 251 & 7 & 3 & 6 & 5 \\ 245 & 8 & 10 & 7 & 249 \\ 255 & 253 & 8 & 12 & 0 \\ 240 & 254 & 5 & 20 & 15 \\ 253 & 4 & 6 & 4 & 8 \end{bmatrix} \quad \mu = \begin{bmatrix} 53.2 \\ 154.0 \\ 150.0 \\ 199.6 \\ 101.4 \\ 53.6 \\ 201.8 \\ 8.2 \\ 200.2 \\ 56.8 \\ 54.4 \\ 103.8 \\ 105.6 \\ 106.8 \\ 55.0 \end{bmatrix}$$

(a) (b)

$$\mathbf{X'} = \begin{bmatrix} 192.8 & -48.2 & -51.2 & -43.2 & -50.2 \\ 96.0 & 96.0 & 98.0 & -148.0 & -142.0 \\ 80.0 & 105.0 & 90.0 & -135.0 & -140.0 \\ 40.4 & -192.6 & 48.4 & 53.4 & 50.4 \\ 133.6 & -92.4 & -89.4 & 140.6 & -92.4 \\ -49.6 & -50.6 & -49.6 & 196.4 & -46.6 \\ -186.8 & 43.2 & 48.2 & 43.2 & 52.2 \\ 1.8 & 1.8 & 3.8 & -3.2 & -4.2 \\ -195.2 & 47.8 & 54.8 & 39.8 & 52.8 \\ -50.0 & -44.8 & -48.8 & 197.2 & -52.8 \\ 196.6 & -47.4 & -51.4 & -48.4 & -49.4 \\ 141.2 & -95.8 & -93.8 & -96.8 & 145.2 \\ 149.4 & 147.4 & -97.6 & -93.6 & -105.6 \\ 133.2 & 147.2 & -101.8 & -86.8 & -91.8 \\ 198.0 & -51.0 & -49.0 & -51.0 & -47.0 \end{bmatrix}$$

Principal Component Analysis (PCA): Numerical Example

Create Covariance Matrix:

The centered data matrix (**X'**) is multiplied by its transpose to create a covariance matrix (C = X'X'^{T}) where the eigenvalue analysis is applied. Since the size of the data matrix is (*15x5*), the covariance matrix size is (*15x15*). When using the singular value decomposition (SVD) to compute the eigenvalues and eigenvectors, it is not necessary to compute the covariance matrix, as will be discussed later. Due to the size of the covariance matrix, it will not be shown here.

$$S_T = \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T$$

Principal Component Analysis (PCA): Numerical Example

Solve Eigenvalue Problem: The eigenvalues and eigenvectors of the covariance matrix can be solved from the following equation,

$$\mathbf{C}v = \lambda v$$

where **C** is the covariance matrix, *v* is the eigenvector and λ is the corresponding eigenvalue. The *Matlab* function *eig.m* can be used to solve for the eigenvalues and eigenvectors.

Principal Component Analysis (PCA): Numerical Example

Order Eigenvectors:

The eigenvectors are then ordered according to their corresponding eigenvalues from highest to lowest. For the example above, only the first four eigenvectors corresponding to the highest eigenvalues (the nonzero eigenvalues) are kept. The eigenvectors corresponding to eigenvalues $\lambda_1 = 397360$, $\lambda_2 = 217350$, $\lambda_3 = 112950$, and $\lambda_4 = 54730$ are shown in Figure *i*.

$$v_1 = \begin{bmatrix} 0.3044 \\ 0.2950 \\ 0.2685 \\ -0.0696 \\ 0.1168 \\ -0.1807 \\ -0.2981 \\ 0.0061 \\ -0.3073 \\ -0.1786 \\ 0.3128 \\ 0.1817 \\ 0.3706 \\ 0.3393 \\ 0.3138 \end{bmatrix} \quad v_2 = \begin{bmatrix} 0.2075 \\ -0.3314 \\ -0.3376 \\ 0.3532 \\ 0.4352 \\ 0.2656 \\ -0.1924 \\ -0.0085 \\ -0.2144 \\ 0.2545 \\ 0.2043 \\ 0.2645 \\ -0.1522 \\ -0.1511 \\ 0.2070 \end{bmatrix} \quad v_3 = \begin{bmatrix} 0.0176 \\ -0.0741 \\ -0.1086 \\ 0.2481 \\ -0.3729 \\ -0.4200 \\ -0.0090 \\ -0.0028 \\ -0.0041 \\ -0.4402 \\ 0.0278 \\ 0.5398 \\ -0.2486 \\ -0.2358 \\ 0.0427 \end{bmatrix} \quad v_4 = \begin{bmatrix} -0.0732 \\ -0.4490 \\ -0.3963 \\ -0.4374 \\ -0.0830 \\ 0.0056 \\ 0.0758 \\ -0.0179 \\ 0.0659 \\ 0.0025 \\ -0.0709 \\ 0.3608 \\ 0.3548 \\ 0.3995 \\ -0.0816 \end{bmatrix}$$

Principal Component Analysis (PCA): Numerical Example

Project Training/Test Images:

The chosen eigenvectors form the basis vectors of the subspace defined by the reduced sample images. Each of the training and test images are projected into this new space to reduce the dimensions of the image feature vectors. To project the images to the subspace, the dot product between the image and each of the eigenvectors is computed. The resulting dimension of the reduced feature vector is equal to the chosen number of eigenvectors. In matrix notation, the eigenvectors are stacked into a matrix **V** = [$\mathbf{v}_1$ $\mathbf{v}_2$ … $\mathbf{v}_n$], where $\mathbf{v}_n$ represents the column-wise eigenvector. To get the reduced feature vector, each image in column-vector form is multiplied to the transpose of **V**. The five centered training images projected into the new space are shown in Figure *i*.

$$x_1 = \begin{bmatrix} 504.86 \\ 180.90 \\ 15.06 \\ -18.08 \end{bmatrix} \quad x_2 = \begin{bmatrix} 90.37 \\ -319.19 \\ -118.87 \\ 100.77 \end{bmatrix} \quad x_3 = \begin{bmatrix} -109.04 \\ -157.16 \\ 63.17 \\ -184.36 \end{bmatrix} \quad x_4 = \begin{bmatrix} -289.06 \\ 232.57 \\ -196.21 \\ 0.41 \end{bmatrix} \quad x_5 = \begin{bmatrix} -197.12 \\ 62.87 \\ 236.84 \\ 101.25 \end{bmatrix}$$

$$Y = W^T X$$

Principal Component Analysis (PCA): Numerical Example

Identify Test Image:

The test image is projected into the new space by first centering the data using the sample mean of the training images and then multiplying it to the transpose of the eigenvector matrix **V** mentioned above. The class of the training image that has the closest distance to the test image is the desired result. The training and test images can be compared by several distance measures, such as the *L2* norm (the Euclidean distance).

The method used here to classify the test images is commonly known as the *nearest-neighbor classifier*, which has the advantage of having no need of preliminary training. The original test image *xTest*, the mean image μ, the centered test data, and the reduced feature vector are shown in Figure 10. Using the *L2* norm as the distance measure for the nearest-neighbor classifier, the test image is closest to the fourth image, which is the desired result.

Principal Component Analysis (PCA): Numerical Example

$$x_{Test} = \begin{bmatrix} 1 \\ 20 \\ 6 \\ 210 \\ 255 \\ 222 \\ 255 \\ 50 \\ 225 \\ 215 \\ 0 \\ 15 \\ 5 \\ 6 \\ 24 \end{bmatrix} \quad \mu = \begin{bmatrix} 53.2 \\ 154.0 \\ 150.0 \\ 199.6 \\ 101.4 \\ 53.6 \\ 201.8 \\ 8.2 \\ 200.2 \\ 56.8 \\ 54.4 \\ 103.8 \\ 105.6 \\ 106.8 \\ 55.0 \end{bmatrix} \quad \overline{x}_{Test} = \begin{bmatrix} -52.2 \\ -134.0 \\ -144.0 \\ 10.4 \\ 153.6 \\ 168.4 \\ 53.2 \\ 41.8 \\ 24.8 \\ 158.2 \\ -54.4 \\ -88.8 \\ -100.6 \\ -100.8 \\ -31.0 \end{bmatrix} \quad x_{TestRed} = \begin{bmatrix} -273.13 \\ 211.33 \\ -173.12 \\ 8.40 \end{bmatrix}$$

(a) (b) (c) (d)

$$x_1 = \begin{bmatrix} 504.86 \\ 180.90 \\ 15.06 \\ -18.08 \end{bmatrix} \quad x_2 = \begin{bmatrix} 90.37 \\ -319.19 \\ -118.87 \\ 100.77 \end{bmatrix} \quad x_3 = \begin{bmatrix} -109.04 \\ -157.16 \\ 63.17 \\ -184.36 \end{bmatrix} \quad x_4 = \begin{bmatrix} -289.06 \\ 232.57 \\ -196.21 \\ 0.41 \end{bmatrix} \quad x_5 = \begin{bmatrix} -197.12 \\ 62.87 \\ 236.84 \\ 101.25 \end{bmatrix}$$

PCA for High-Dimensional Data

When the number of training image samples (m, e.g. $m = 400$) is much less than the number of pixels (n, e.g. $n = 3000$) in an image, the dimension of the original feature vector ($m << n$), there will only be $m – 1$, instead of n meaningful eigenvectors.

The remaining eigenvectors have an associated eigenvalues of zero. In fact, in the numerical example above, the eigenvalues of the eigenvectors, except the four chosen ones, have eigenvalues of zero.

PCA for High-Dimensional Data

In practical problems such as *face recognition*, training images of size (*60*x*50*) form a covariance matrix of size (*3000*x*3000*) that is computationally demanding in solving its eigenvalues and eigenvectors. The *n*-dimensional eigenvectors can be solved by first computing the eigenvectors of the (*m* x *m*, e.g. *400*x*400*) covariance matrix, a less computationally intensive task [7]. Consider the eigenvectors v_i of the matrix $\mathbf{X}^T\mathbf{X}$,

$$\left(\mathbf{X}^T\mathbf{X}\right)v_i = \lambda_i v_i$$

Premultiplying both sides by **X**, this results into

$$\left(\mathbf{X}\mathbf{X}^T\right)\mathbf{X}v_i = \lambda_i \mathbf{X}v_i$$

where $\mathbf{X}v_i$ are the eigenvectors of the original covariance matrix $\mathbf{C} = \mathbf{X}\mathbf{X}^T$. Therefore, it is possible to solve the eigenvectors of the original covariance ($\mathbf{C} = \mathbf{X}\mathbf{X}^T$) matrix by first getting the eigenvectors of the much smaller covariance matrix ($\mathbf{C} = \mathbf{X}^T\mathbf{X}$) and then multiplying it by the data matrix **X**, resulting to the actual eigenvectors, $v_i^* = \mathbf{X}v_i$.

PCA for High-Dimensional Data: Numerical Example

$$v_1 = \begin{bmatrix} -0.8009 \\ -0.1434 \\ 0.1730 \\ 0.4586 \\ 0.3127 \end{bmatrix} \quad v_2 = \begin{bmatrix} 0.3880 \\ -0.6847 \\ -0.3371 \\ 0.4989 \\ 0.1349 \end{bmatrix} \quad v_3 = \begin{bmatrix} 0.0448 \\ -0.3537 \\ 0.1880 \\ -0.5838 \\ 0.7048 \end{bmatrix} \quad v_4 = \begin{bmatrix} 0.0773 \\ -0.4308 \\ 0.7881 \\ -0.0018 \\ -0.4328 \end{bmatrix}$$

$$\lambda_1 = 397360 \quad \lambda_2 = 217350 \quad \lambda_3 = 112950 \quad \lambda_4 = 54730$$

$$v_1 = \begin{bmatrix} -191.8696 \\ -185.9698 \\ -169.2429 \\ 43.8772 \\ -73.6407 \\ 113.8876 \\ 187.8875 \\ -3.8231 \\ 193.7258 \\ 112.5835 \\ -197.1967 \\ -114.5619 \\ -233.6161 \\ -213.9056 \\ -197.8283 \end{bmatrix} \quad v_2 = \begin{bmatrix} 96.7505 \\ -154.4944 \\ -157.4129 \\ 164.6605 \\ 202.9169 \\ 123.8084 \\ -89.7181 \\ -3.9777 \\ -99.9670 \\ 118.6650 \\ 95.2586 \\ 123.2928 \\ -70.9800 \\ -70.4599 \\ 96.4842 \end{bmatrix} \quad v_3 = \begin{bmatrix} 5.9096 \\ -24.9000 \\ -36.4848 \\ 83.3793 \\ -125.3388 \\ -141.1545 \\ -3.0265 \\ -0.9334 \\ -1.3816 \\ -147.9460 \\ 9.3591 \\ 181.4277 \\ -83.5646 \\ -79.2540 \\ 14.3557 \end{bmatrix} \quad v_4 = \begin{bmatrix} 17.1252 \\ 105.0272 \\ 92.7194 \\ 102.3269 \\ 19.4202 \\ -1.3082 \\ -17.7373 \\ 4.1821 \\ -15.4206 \\ -0.5869 \\ 16.5798 \\ -84.4111 \\ -82.9910 \\ -93.4526 \\ 19.0962 \end{bmatrix}$$

$$v_1 = \begin{bmatrix} -0.3044 \\ -0.2950 \\ -0.2685 \\ 0.0696 \\ -0.1168 \\ 0.1807 \\ 0.2981 \\ -0.0061 \\ 0.3073 \\ 0.1786 \\ -0.3128 \\ -0.1817 \\ -0.3706 \\ -0.3393 \\ -0.3138 \end{bmatrix} \quad v_2 = \begin{bmatrix} 0.2075 \\ -0.3314 \\ -0.3376 \\ 0.3532 \\ 0.4352 \\ 0.2656 \\ -0.1924 \\ -0.0085 \\ -0.2144 \\ 0.2545 \\ 0.2043 \\ 0.2645 \\ -0.1522 \\ -0.1511 \\ 0.2070 \end{bmatrix} \quad v_3 = \begin{bmatrix} 0.0176 \\ -0.0741 \\ -0.1086 \\ 0.2481 \\ -0.3729 \\ -0.4200 \\ -0.0090 \\ -0.0028 \\ -0.0041 \\ -0.4402 \\ 0.0278 \\ 0.5398 \\ -0.2486 \\ -0.2358 \\ 0.0427 \end{bmatrix} \quad v_4 = \begin{bmatrix} 0.0732 \\ 0.4490 \\ 0.3963 \\ 0.4374 \\ 0.0830 \\ -0.0056 \\ -0.0758 \\ 0.0179 \\ -0.0659 \\ -0.0025 \\ 0.0709 \\ -0.3608 \\ -0.3548 \\ -0.3995 \\ 0.0816 \end{bmatrix}$$

Face Recognition: Linear Discriminant Analysis (LDA)

- The objective of LDA is to perform dimensionality reduction while preserving as much of the class discriminatory information as possible

$$Y = W^T X$$

- The method selects W_{LDA} in such a way that the ratio of the between-class scatter (S_B) and the within-class scatter (S_W) is maximized

$$S_W = \sum_{i=1}^{c} \sum_{x_k \in X_i} (x_k - \mu_i)(x_k - \mu_i)^T \qquad S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

N_i – *no. of samples in class* X_i

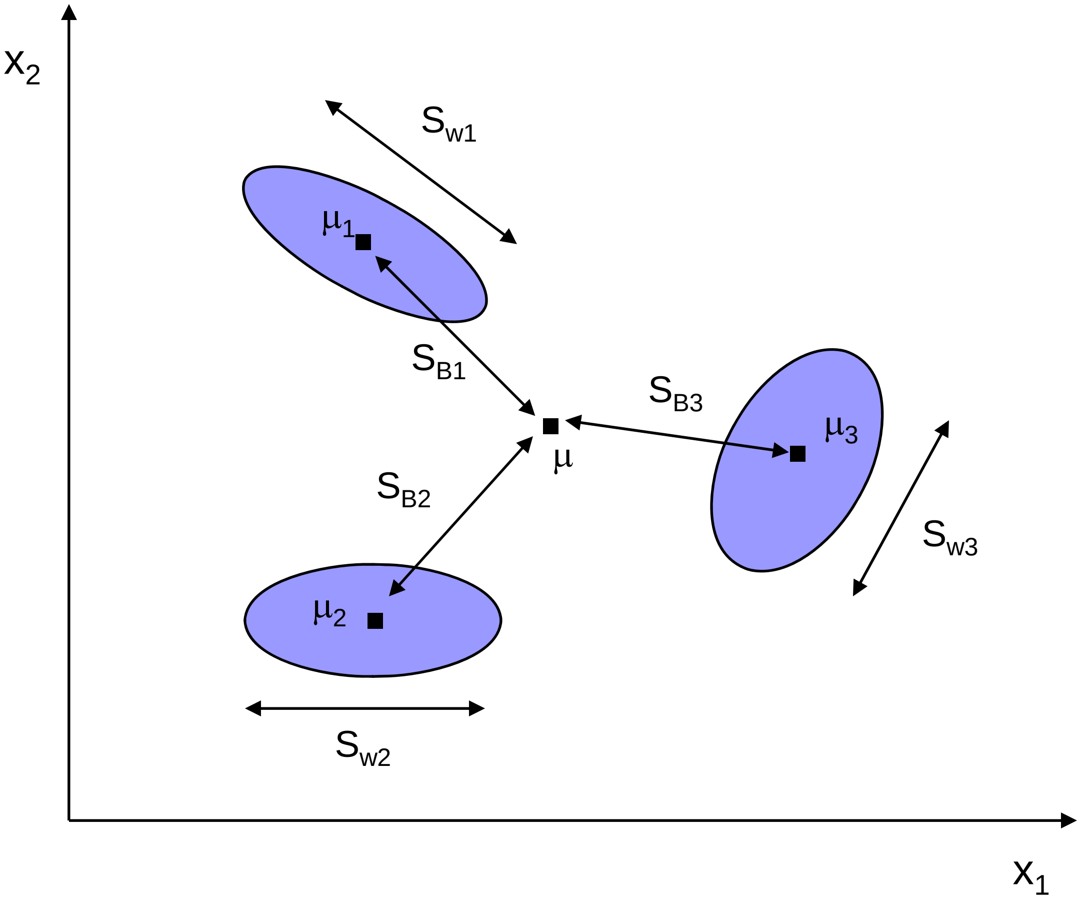
μ – *overall mean*

μ_i – *mean image of class* X_i

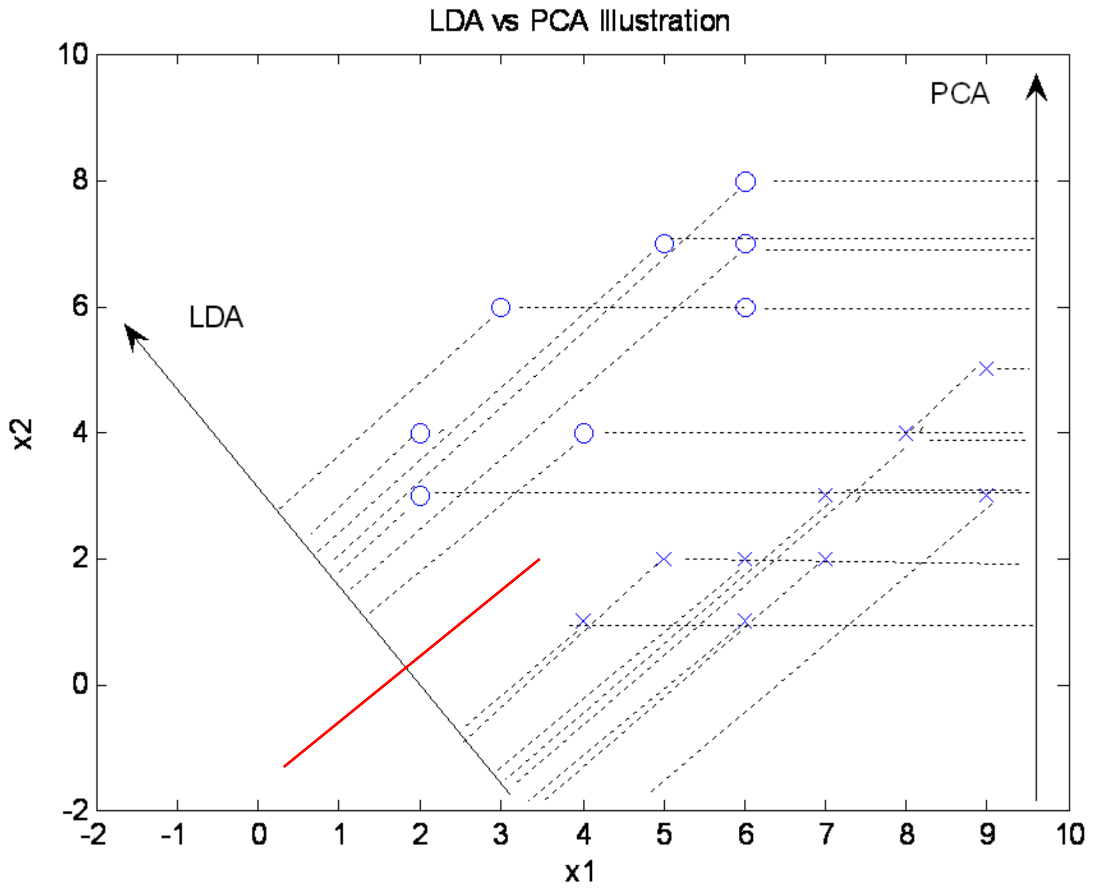
$$\boxed{S_B W = \lambda S_W W}$$

Face Recognition: Linear Discriminant Analysis (LDA)

- The method selects W_{LDA} in such a way that the ratio of the between-class scatter (S_B) and the within-class scatter (S_W) is maximized

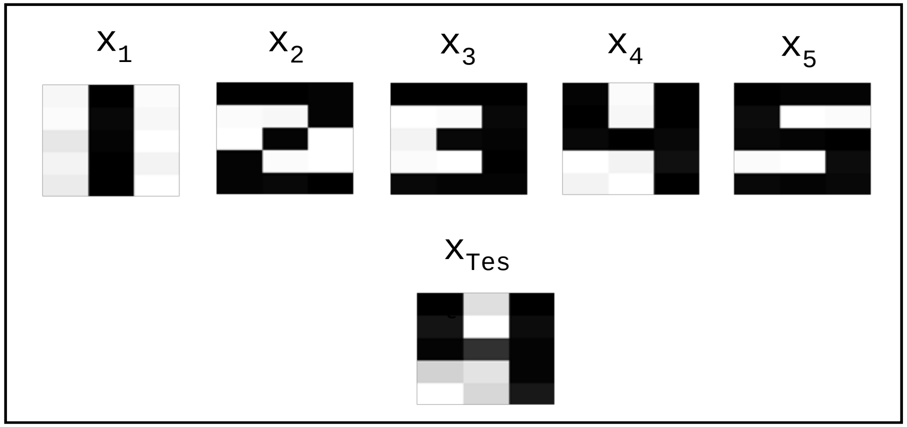


Face Recognition: Linear Discriminant Analysis (LDA)



- Numerical Example (2D case)

Linear Discriminant Analysis (LDA): Numerical Example



$$\mathbf{X}_1 = \begin{bmatrix} 246 & 255 & 252 & 245 \\ 250 & 242 & 251 & 236 \\ 230 & 233 & 228 & 216 \\ 240 & 240 & 246 & 234 \\ 235 & 247 & 239 & 220 \\ 4 & 12 & 13 & 1 \\ 15 & 14 & 29 & 20 \\ 10 & 0 & 11 & 0 \\ 5 & 15 & 16 & 0 \\ 6 & 4 & 0 & 9 \\ 251 & 254 & 255 & 254 \\ 245 & 254 & 238 & 230 \\ 255 & 255 & 248 & 240 \\ 240 & 247 & 251 & 231 \\ 253 & 243 & 255 & 255 \end{bmatrix} \quad \mathbf{X}_2 = \begin{bmatrix} 5 & 2 & 0 & 0 \\ 250 & 255 & 235 & 246 \\ 255 & 255 & 255 & 255 \\ 7 & 4 & 0 & 14 \\ 9 & 21 & 3 & 15 \\ 3 & 0 & 8 & 0 \\ 245 & 241 & 239 & 244 \\ 10 & 19 & 9 & 8 \\ 248 & 233 & 235 & 244 \\ 12 & 1 & 27 & 2 \\ 7 & 0 & 9 & 12 \\ 8 & 0 & 6 & 14 \\ 253 & 255 & 253 & 255 \\ 254 & 247 & 249 & 253 \\ 4 & 0 & 2 & 6 \end{bmatrix} \quad \mathbf{X}_3 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 252 & 255 & 254 & 251 \\ 240 & 238 & 248 & 246 \\ 248 & 255 & 249 & 255 \\ 12 & 11 & 16 & 5 \\ 4 & 2 & 0 & 0 \\ 250 & 255 & 246 & 255 \\ 12 & 13 & 20 & 4 \\ 255 & 246 & 255 & 255 \\ 8 & 13 & 7 & 20 \\ 3 & 13 & 5 & 0 \\ 10 & 0 & 19 & 2 \\ 8 & 13 & 0 & 0 \\ 5 & 1 & 8 & 0 \\ 6 & 16 & 0 & 10 \end{bmatrix}$$

$$\mathbf{X}_4 = \begin{bmatrix} 10 & 10 & 7 & 1 \\ 6 & 12 & 0 & 16 \\ 15 & 13 & 26 & 5 \\ 253 & 247 & 238 & 243 \\ 242 & 233 & 250 & 255 \\ 250 & 241 & 255 & 248 \\ 245 & 250 & 255 & 240 \\ 5 & 0 & 14 & 0 \\ 240 & 241 & 238 & 236 \\ 254 & 244 & 254 & 251 \\ 6 & 12 & 0 & 9 \\ 7 & 3 & 11 & 0 \\ 12 & 23 & 7 & 0 \\ 20 & 31 & 34 & 19 \\ 4 & 7 & 11 & 15 \end{bmatrix} \quad \mathbf{X}_5 = \begin{bmatrix} 3 & 3 & 0 & 16 \\ 12 & 24 & 19 & 5 \\ 10 & 20 & 3 & 0 \\ 250 & 254 & 248 & 255 \\ 9 & 19 & 22 & 1 \\ 7 & 12 & 12 & 11 \\ 254 & 249 & 245 & 255 \\ 4 & 0 & 14 & 9 \\ 253 & 248 & 255 & 255 \\ 4 & 5 & 0 & 0 \\ 5 & 12 & 0 & 0 \\ 249 & 243 & 252 & 255 \\ 0 & 10 & 4 & 0 \\ 15 & 17 & 11 & 27 \\ 8 & 4 & 10 & 15 \end{bmatrix}$$

Linear Discriminant Analysis (LDA): Problems

$$S_B W = \lambda S_W W$$

Problem: Within-class scatter matrix (S_W) is always singular. Rank (S_W) is at most N – c, and the no. of images is much smaller than the number of pixels in each image (N – total no. of samples, c – no. of classes)

Linear Discriminant Analysis (LDA): Problems

Solution to the singularity problem (Belhumeur, 1997) : Fisherface method

Fisherface method: Project the image set to a lower-dimensional space to avoid nonsingularity in the within-class scatter (S_W)

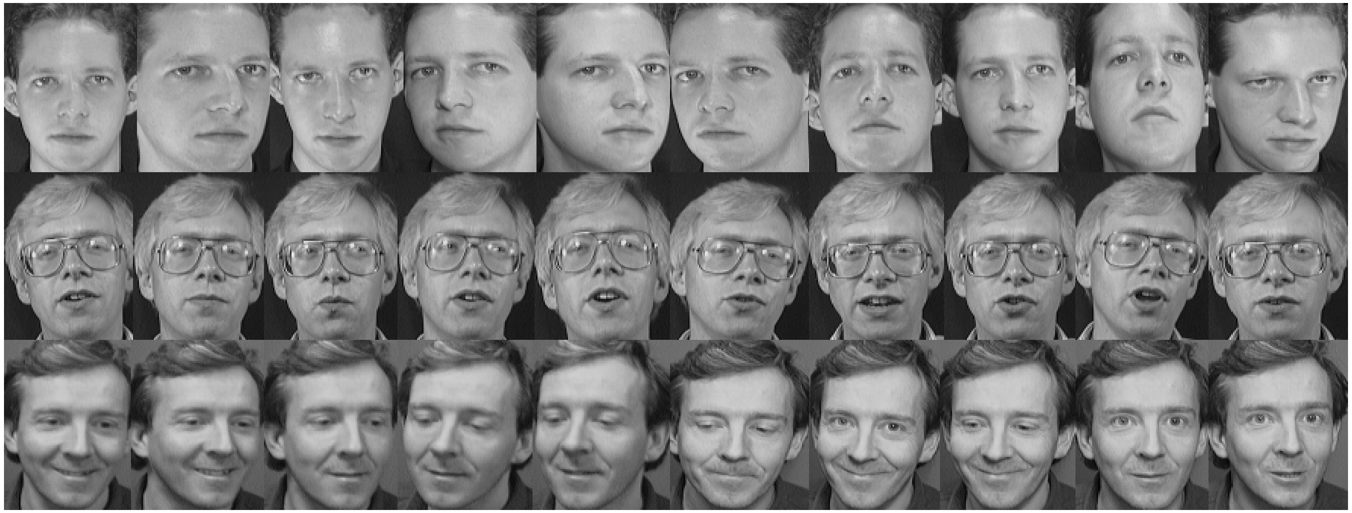
Use PCA to achieve dimension reduction to *N* – *c*, and apply the Fisher Linear Discriminant (FLD)

$$W_{opt}{}^{T} = W_{fld}{}^{T} W_{pca}{}^{T}$$

$$W_{pca} = \arg\max_{W} \left| W^{T} S_{T} W \right| \qquad W_{fld} = \arg\max_{W} \frac{\left| W^{T} W_{pca}{}^{T} S_{B} W_{pca} W \right|}{\left| W^{T} W_{pca}{}^{T} S_{W} W_{pca} W \right|}$$

Face Recognition: Databases

- ORL Database (400 images: 40 people, 10 images each)



Face Recognition: Databases

- FERET Database – contains more extensive training and test images for evaluating face recognition algorithms

- Gallery – set of known individuals
- Probe – image of unknown face presented to the algorithm
- Duplicate – a type of probe but the corresponding gallery image was taken on a different day

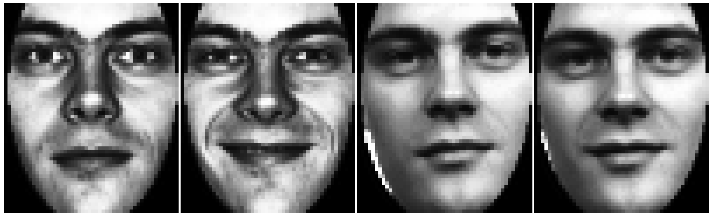
- Original (Grayscale) FERET: fa - gallery, fb – probe, dup1 – probe (within 1 minute to 1031 days), dup2 - probe

fa

fb

dup1

dup2



- Normalized Version

Face Recognition: Distance measures

- Use 1-nearest neighbor (NN) classifier: similarity measures to evaluate the efficiency of different representation and recognition methods

$$d_{L1}(\mathbf{x},\mathbf{y}) = \sum_i |x_i - y_i|$$

L1 Norm

$$d_{L2}(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^T(\mathbf{x}-\mathbf{y})$$

L2 Norm

$$d_{\cos}(\mathbf{x},\mathbf{y}) = -\frac{\mathbf{x}\bullet\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|}$$

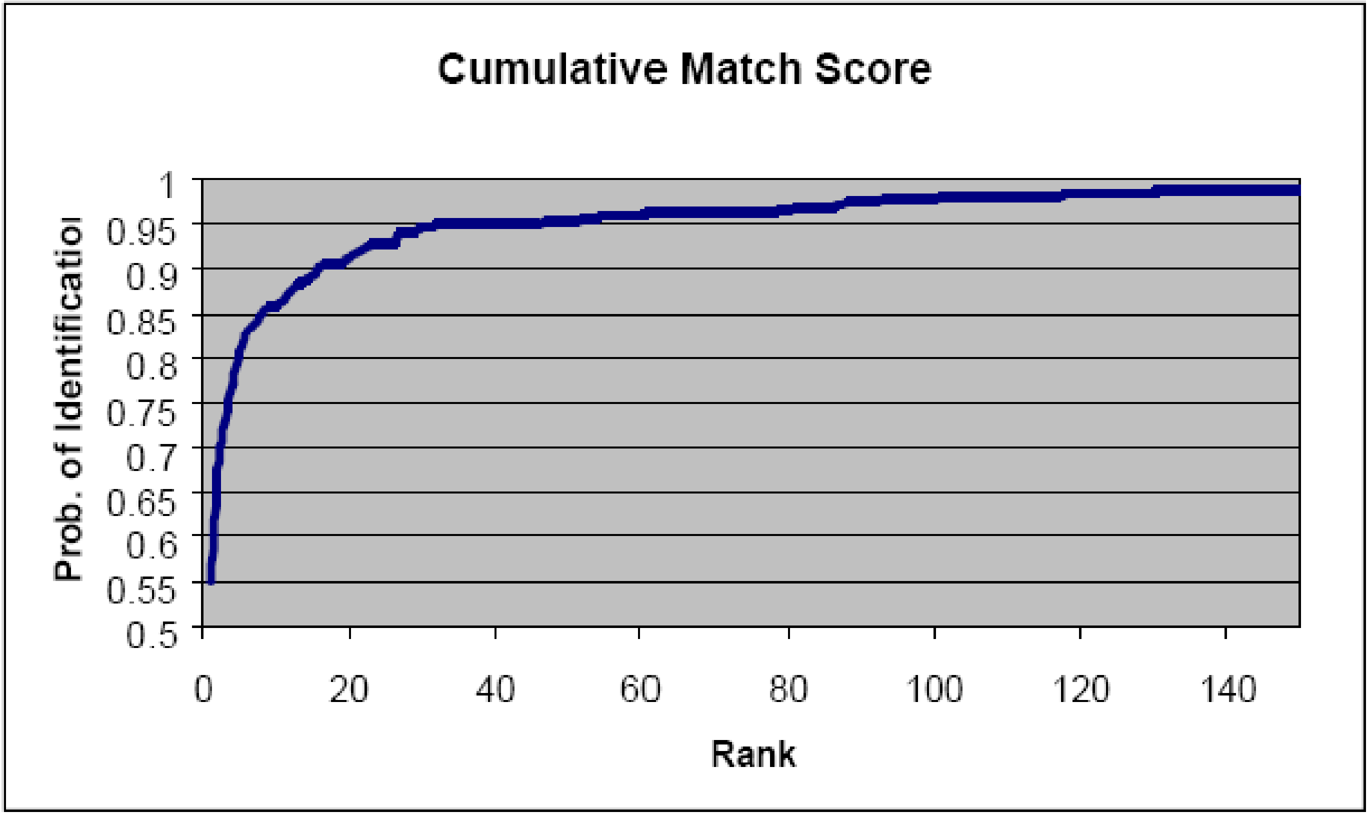
Cosine Distance

$$d_{Md}(\mathbf{x},\mathbf{y}) = (\mathbf{x}-\mathbf{y})^T\Sigma^{-1}(\mathbf{x}-\mathbf{y})$$

Mahalanobis Distance

Face Recognition: Cumulative Match Score Curves (CMC)

- Cumulative Match Score (CMS) – rank *n* vs. percentage of correct identification



Face Recognition: Cumulative Match Score Curves (CMC)

- Cumulative Match Score (CMS) – rank *n* vs. percentage of correct identification

Probe Image

Gallery Images

Classifier Scores (Nearest Neighbor)

1.75

2.55

1.25

1.35

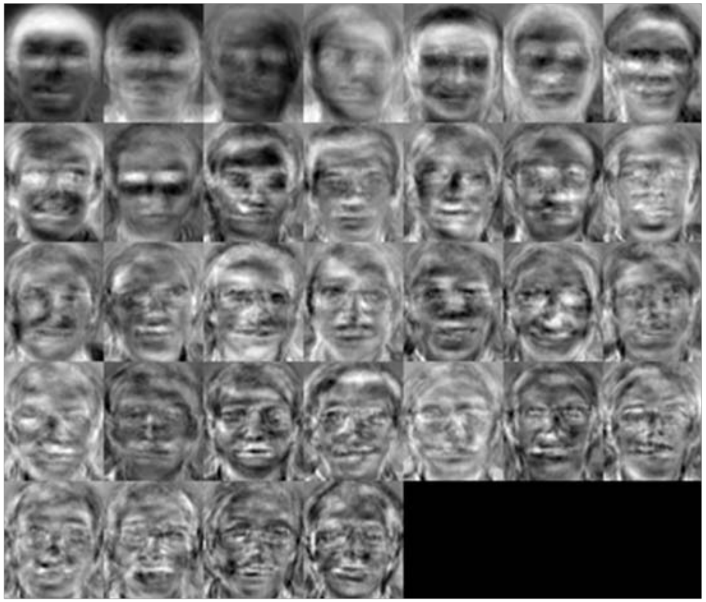
Sorted Scores

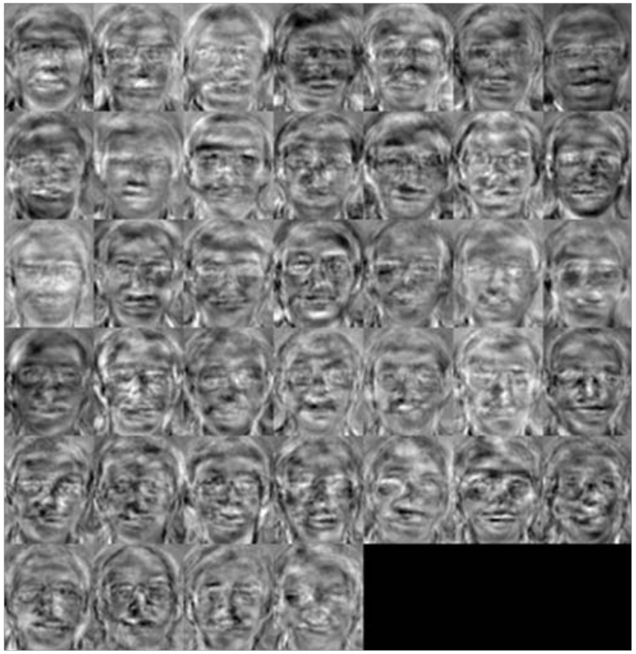
1.25	Rank-*1* (+0)
1.35	Rank-*2* (+1)
1.75	Rank-*3* (+1)
2.55	Rank-*4* (+1)

- You continue counting for all probe images

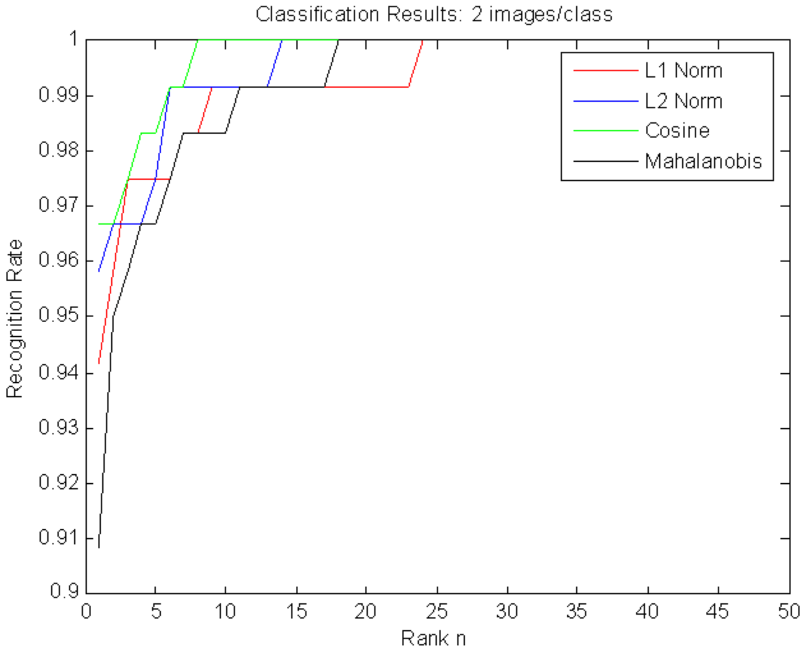
Face Recognition: Results and Discussions (ORL)

- Eigenfaces/Fisherfaces

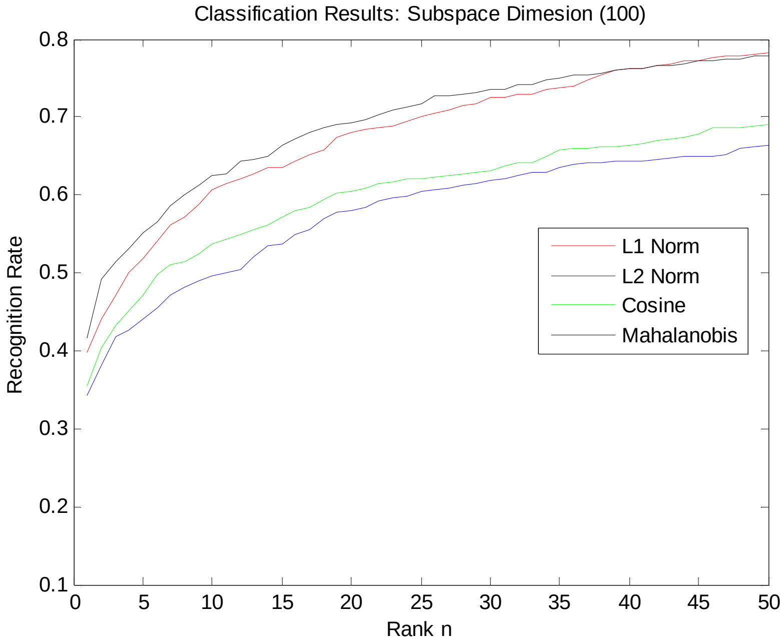

Eigenfaces (2 images/class)


Fisherfaces

Face Recognition: Results and Discussions (ORL and FERET)



ORL



FERET

References

X. Lu, “Image Analysis for Face Recognition,” Personal Notes, May 2003, http://www.face-rec.org/interesting-papers/General/ImAna4FacRcg_lu.pdf

P. Belhumuer, J. Hespanha, and D. Kriegman, “Eigenfaces vs. fisherfaces: Recognition using class specific linear projection,” IEEE Trans. Pattern Analysis and Machine Intelligence, 19(7): 711-720, 1997

M. Turk and A. Pentland, “Eigenfaces for Recognition,” J. Cog. Neuroscience, Jan. 1991, pp. 71-86