Object Recognition

Chapter 4: Global Subspace Features

Prof. Dr. Johannes Maucher

HdM CSM

Version 1.1
23.06. 2020

Document History

Version Nr.	Date	Changes
1.0	26.02.2013	Initial Version
1.1	23.06.2020	Adaptations for SS 20

Global Features considered so far

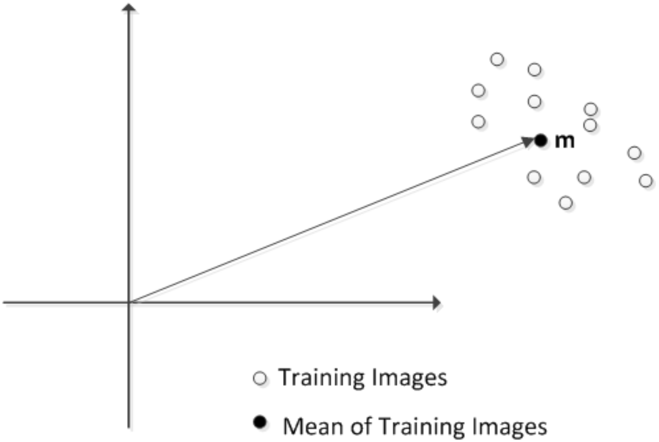
- Pixel Intensities: L (=number of channels) values per pixel for each pixel in the image or subwindow.
  - Requires cropped and aligned objects
  - not robust w.r.t. translations, rotations, scale, illumination, occlusion
  - Extremely high dimensional feature space $L \cdot r \cdot c$, where r is the number of lines and c is the number of pixels per line.
- Histogram based descriptors:
  - Color histogram and multidimensional receptive field histograms
  - Robust w.r.t. translation, rotation, partial occlusion
  - Quite long descriptors

Idea of Subspace Representation of Global Features

- Depending on the camera resolution the space of pixel intensities is extremely high dimensional.
- If all images depict similar objects (e.g. cropped faces), then the representations of these images in the high-dimensional space occupy only a small subspace.
- The images in this small subspace can be described, by a mean image plus a weighted sum of vectors. These vectors must
  - be linear independent (every dependent vector would be redundant)
  - capable to describe the variations in the set of relevant pictures
- Perform object recognition (matching) in the transformed low-dimensional space, which is spanned by the set of linear independent vectors.

Idea of dimensionality reduction

- Each image constitutes a point in an $(L \cdot r \cdot c)$-dimensional space
- Simple model of a 2-dimensional space:



Example: Space spanned by two linear independent vectors

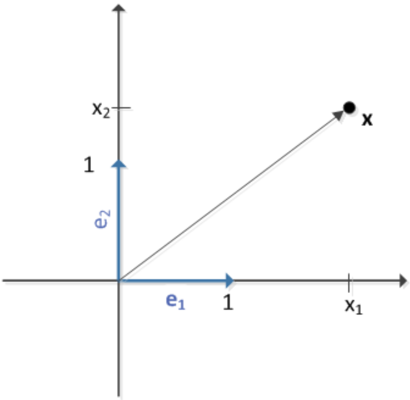
- Set of 2 linear independent vectors (here unity vectors)

$$\mathbf{e_1}$$
$$\mathbf{e_2}$$

- Any point $\mathbf{x}$ in the 2-dimensional space can be described as a linear combination of $\mathbf{e_1}$ and $\mathbf{e_2}$:

$$\mathbf{x} = x_1\mathbf{e_1} + x_2\mathbf{e_2}$$

- Scalars x_1 and x_2 are the coordinates of vector $\mathbf{x}$ w.r.t. the coordinate system spanned by $\mathbf{e_1}$ and $\mathbf{e_2}$.



Idea of Principal Component Analysis

Problem:

- Given a set of similar training images (e.g. cropped and aligned faces), how to find the set of linear independent vectors, that span a subspace, in which all images can be represented with a minimal information-loss?

Solution:

- The image points are assumed to be distributed according to a multidimensional Gaussian distribution
- The variations of such a distribution are described by the covariance matrix
- The Eigenvectors of the covariance matrix constitute a set of orthogonal vectors.
- The relevance of an Eigenvector is determined by it's associated Eigenvalues.
- A matrix which contains the most relevant Eigenvectors as columns defines the PCA transformation.
- The most relevant Eigenvectors are also called Principal Components.

Eigenfaces

- In the case of face recognition, the Eigenvectors are called Eigenfaces
- The set of Eigenfaces describes the variations within the given set of faces, i.e. Eigenfaces constitute discriminative Features.
- Each face can be described as a linear combination of Eigenfaces plus a mean face.
- The subspace spanned by the Eigenfaces can have a much lower dimensionality than the original space.
- The original space has $r \cdot c$ dimensions, since greyscale images are applied.
- The Eigenface approach to face recognition has been introduced by Turk and Pentland in [Turk and Pentland, 1991].

Eigenface Training Process [Turk and Pentland, 1991]

Training Phase

1. Acquire initial set of face images. Greyscale images, cropped, aligned and of similar illumination
2. Apply PCA to calculate set of Eigenfaces. Keep only the M most relevant Eigenfaces, the ones with the highest Eigenvalues. These M images span the *face space*[a].
3. Project each of the training face images into the M-dimensional *face space*

[a] As new faces are experienced the eigenfaces can be recalculated and updated

Eigenface Recognition Process [Turk and Pentland, 1991]

Recognition Phase

1. Project the query image into the M-dimensional *face space* and calculate the weight (coefficient) w.r.t. each Eigenface.
2. Determine if the image is a face at all, by checking if the image is sufficiently close to the *face space*.
3. If it is a face, apply a nearest-neighbor strategy in the *face space* in order to find the closest face in the training set [a]. If the face is not sufficiently close to the known faces label it as *unknown*.
4. (Optional) Update the eigenfaces and the distribution of the images in the *face space*.
5. (Optional) If the same unknown face is seen several times, incorporate into the known faces.

[a] Usually the training faces are tagged with the name of the persons. Thus the person is recognized

Principal Component Analysis: Concept

- Principal component analysis (PCA) is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components[1].
- The columns of the orthogonal transformation matrix are the Eigenvectors of the covariance matrix of the given data. Here the Eigenvectors are the principal components.
- With respect to the prinicpal components the covariance of the data is 0, i.e. the covariance matrix is a diagonal matrix containing the variances along the principal components.
- Along some principal components (dimensions) the variance is very small
- These dimensions can be deleted with a marginal loss of information (Dimensionality Reduction)

[1] http://en.wikipedia.org/wiki/Principal_component_analysis

PCA by Example: Step 1 Collect Data

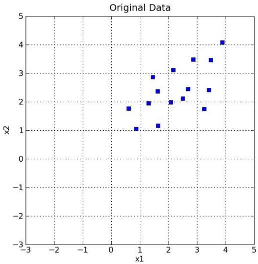
- Given: Set of N observations, each described by d features.

$$X = \begin{pmatrix} x_{1,1} & x_{1,2}; & \cdots & x_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2}; & \cdots & x_{N,d} \end{pmatrix}$$

- Example (Note that the features x_1 and x_2 are correlated:

X =

x1	x2
0.859	1.042
0.599	1.771
1.302	1.953
1.615	2.370
2.161	3.125
2.865	3.490
3.411	2.422
3.255	1.745
2.500	2.109
2.682	2.448
3.490	3.464
3.880	4.089
2.083	1.979
1.641	1.172
1.458	2.865



PCA by Example: Step 2 Subtract Mean

- For each column (i.e. each feature) in X:
  - Calculate mean of column

$$\overline{x_j} = \frac{1}{N} \sum_{i=1}^{N} x_{i,j}$$

  - Subtract mean $\overline{x_j}$ from all values $x_{i,j}$ in column j of X
- In the new representation X' each column has a mean of 0.
- In the example the mean values of X are

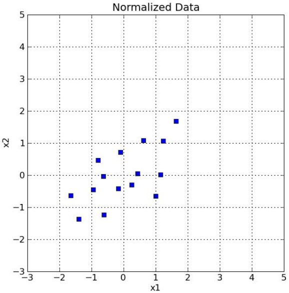
$$\begin{aligned}\overline{x_1} &= 2.253 \\ \overline{x_2} &= 2.402\end{aligned}$$

PCA by Example: Step 2 Subtract Mean

Normed[2] data of the example:

X' =

x1	x2
-1.394	-1.361
-1.654	-0.632
-0.951	-0.450
-0.638	-0.033
-0.092	0.722
0.612	1.087
1.158	0.019
1.002	-0.658
0.247	-0.294
0.429	0.045
1.237	1.061
1.627	1.686
-0.170	-0.424
-0.612	-1.231
-0.795	0.462



[2] Here normed means mean value free

PCA by Example: Step 3 Calculate Covariance Matrix

- Variance σ_j^2 of feature x_j in X:

$$\sigma_j^2 = c_{jj} = \frac{1}{N-1} \sum_{i=1}^{N} (x_{i,j} - \overline{x_j}) \cdot (x_{i,j} - \overline{x_j})$$

- Covariance $\sigma_{j,k}$ between features x_j and x_k:

$$\sigma_{j,k} = \sigma_{k,j} = c_{jk} = c_{kj} = \frac{1}{N-1} \sum_{i=1}^{N} (x_{i,j} - \overline{x_j}) \cdot (x_{i,k} - \overline{x_k})$$

- Covariance matrix C of X:

$$C = \begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1d} \\ c_{21} & c_{22} & \cdots & c_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ c_{d1} & c_{d2} & \cdots & c_{dd} \end{pmatrix}$$

- In the example:

$$C = \begin{pmatrix} 1.013 & 0.558 \\ 0.558 & 0.754 \end{pmatrix}$$

PCA by Example: Eigenvectors and Eigenvalues (1)

- Let V denote a d-dimensional linear space, spanned by the basis vectors

$$(\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_d})$$

- By applying a linear transformation (rotation) V can be transformed into a d-dimensional space V'.
- Such a transformation is defined by a $d \times d$-matrix A:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,d} \\ \vdots & \vdots & \ddots & \vdots \\ a_{d,1} & a_{d,2} & \ldots & a_{d,d} \end{pmatrix}$$

- An arbitrary point $\mathbf{p} = (p_1, p_2, \ldots p_d) \in V$ is transformed to

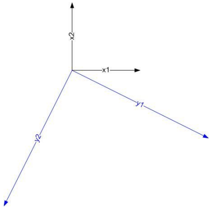
$$\mathbf{b}^T = A \cdot \mathbf{p}^T$$

in V'.

PCA by Example: Eigenvectors and Eigenvalues (2)

- Example: Linear transformation, defined by matrix

$$A = \begin{pmatrix} 2 & -1 \\ -1 & -2 \end{pmatrix}$$



- The transformation is orthogonal, since all columns in A are pairwise orthogonal (scalar product of 0).
- An orthogonal transform keeps the orthogonality of the basis vectors. I.e. if the basis vectors $\mathbf{x_i}$ of V are orthogonal to each other, than also their transformations $\mathbf{y_i}^T = A \cdot \mathbf{x_i}^T$ are orthogonal to each other.
- This transform is not orthonormal because the magnitude of the columns of A is not 1. I.e. the length of the new basis vectors is different to the length of the old basis vectors.

PCA by Example: Eigenvectors and Eigenvalues (3)

- Eigenvectors of a $(d \times d)$-matrix A are those vectors, which have the same direction in the old and the new rotated coordinate system. I.e.

$$A \cdot \mathbf{u}_i^T = \lambda_i \cdot \mathbf{u}_i^T \tag{1}$$

  holds for each Eigenvector $\mathbf{u}_i$ of A.
- The scalar λ_i in equation (1) is called the Eigenvalue of Eigenvector $\mathbf{u}_i$.
- If the $(d \times d)$-matrix A has full rank and is symmetric, than there exist d Eigenvectors $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_d}$ of A. These Eigenvectors are orthogonal to each other.
- The Eigenvalues λ_i define whether the corresponding Eigenvector $\mathbf{u_i}$ is compressed $(\lambda_i < 1)$ or stretched $(\lambda_i > 1)$ in the new space V'.
- In order to avoid ambuigities all Eigenvectors are usually normed to a length of 1.

PCA by Example: Step 4 Eigenvectors and Eigenvalues of Covariance Matrix

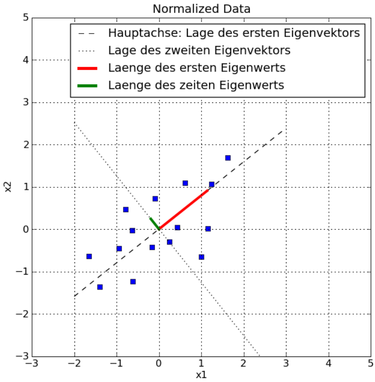
- The covariance matrix of the example is:

$$C = \begin{pmatrix} 1.013 & 0.558 \\ 0.558 & 0.754 \end{pmatrix}$$

- Normed Eigenvectors and Eigenvalues of C are:

$$\begin{aligned} \mathbf{u}_1 &= \begin{pmatrix} 0.783 \\ 0.622 \end{pmatrix}, \ \lambda_1 = 1.456 \\ \mathbf{u}_2 &= \begin{pmatrix} -0.622 \\ 0.783 \end{pmatrix}, \ \lambda_2 = 0.310 \end{aligned}$$

PCA by Example: Visualization of the Eigenvectors and Eigenvalues



PCA by Example: Step 5 Arrange Eigenvectors and Eigenvalues

- The Eigenvectors and Eigenvalues are ordered according to decreasing Eigenvalue. After this rearrangement
  - The first Eigenvector $\mathbf{u}_1$ is the one which corresponds to the largest Eigenvalue. This largest Eigenvalue is then denoted by λ_1,
  - the second Eigenvector $\mathbf{u}_2$ is the one which corresponds to the second largest Eigenvalue. This second largest Eigenvalue is then denoted by λ_2
  - ...
- The first Eigenvector $\mathbf{u}_1$ points into the direction of the largest variance in the data.
- The second Eigenvector $\mathbf{u}_2$ is orthogonal to the first Eigenvector and points into the direction of the second largest variance.
- The *i.th* Eigenvector $\mathbf{u}_i$ is orthogonal to all previous Eigenvectors and points into the direction of the *i.th* largest variance.

PCA by Example: Step 6 PCA Transformation Matrix

- Arrange the ordered Eigenvectors as columns in a matrix:

$$U = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_d] \tag{2}$$

- Transform the data X', using transformation matrix U.
- W.r.t. the new coordinates, defined by the Eigenvectors, the variance is uncorrelated, i.e. the covariance matrix C' of the transformed data is a diagonal matrix. The values on the diagonal, i.e. the variances, are the Eigenvalues

$$C' = \begin{pmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \lambda_d \end{pmatrix} \tag{3}$$

- In the Example the transormation matrix U with the Eigenvectors as columns and the covariance matrix C' in the new space are:

$$U = \begin{pmatrix} 0.783 & -0.622 \\ 0.622 & 0.783 \end{pmatrix} \quad C' = \begin{pmatrix} 1.456 & 0 \\ 0 & 0.310 \end{pmatrix} \tag{4}$$

PCA by Example: Step 7 Dimensionality Reduction

- If U is applied as transformation matrix, then the new space has the same dimensionality as the original space.
- Data w.r.t. the new dimensions is decorrelated and typically there are some dimensions with relatively small variances.
- Dimensions along which the variance is small contain only marginal information.
- For dimensionality reduction the w dimensions with the smallest variances (Eigenvalues) can be removed. The corresponding information loss is minimal, if the variances of the removed dimensions are small.
- This dimensionality reduction can be implemented by removing the w rightmost columns in U and applying the reduced matrix U_M with $M = d - w$ columns for transformation:

$$U_M = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_{d-w}] \tag{5}$$

- In the example the transformation matrix U_M obtained by removing the last column (i.e. $w = 1$) of U is:

$$U_1 = \begin{pmatrix} 0.783 \\ 0.622 \end{pmatrix} \tag{6}$$

PCA by Example: Step 8 Transform Data into new space

- The normed data matrix X' as defined in slide 2 has N rows and d columns.
- This data is PCA mapped into the new M-dimensional space by

$$Y^T = U_M^T \cdot X'^T \tag{7}$$

The N rows in Y are the data samples represented in the new space of lower dimensionality

- Backtransformation:

$$X''^T = (U_M) \cdot Y^T \tag{8}$$

- In the case of dimensionality reduction ($M < d$), matrix X'' is not the same as X'. However, PCA guarantees a minimal MSE between X' and X'' for given M.
- Reconstruction Mean Square Error (MSE):

$$MSE = \frac{1}{N} \sum_{i=1}^{N} d(\mathbf{x}'_i, \mathbf{x}''_i) \tag{9}$$

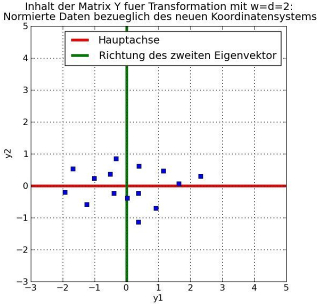
where $\mathbf{x}'_i$ and $\mathbf{x}''_i$ are the $i.th$ rows in X' and X'' respectively.

PCA by Example: Data in the transformed space $M = 2$

- Transformed data in the case of no dimensionality reduction, i.e. $M = d = 2$

Y =

y1	y2
-1.938	-0.198
-1.688	0.535
-1.025	0.240
-0.520	0.371
0.377	0.623
1.155	0.470
0.918	-0.705
0.375	-1.138
0.010	-0.384
0.364	-0.231
1.628	0.061
2.322	0.308
-0.397	-0.226
-1.245	-0.583
-0.335	0.857



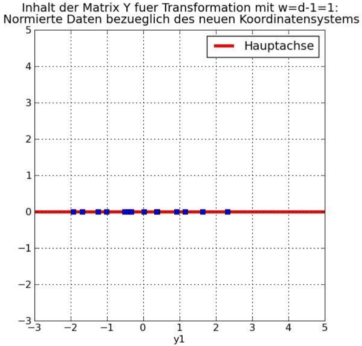
- Transformed Features are uncorrelated
- Variance along feature on horizontal axis much larger than variance along vertical axis.

PCA by Example: Data in the transformed space $M = 1$

- Transformed data in the case of dimensionality reduction with $M = d - 1 = 1$

Y =

y1
-1.938
-1.688
-1.025
-0.520
0.377
1.155
0.918
0.375
0.010
0.364
1.628
2.322
-0.397
-1.245
-0.335

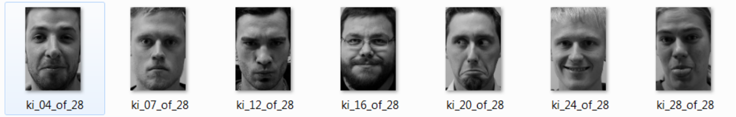


Eigenface: Training

1. Collect N greyscale images of the persons, that should be recognized. Preferable > 1 image per person. All images must be of same size $(r \times c)$
2. Serialize each image sucht that it is represented as a 1-dimensional vector of length $d = r \cdot c$. The N serialzed images are denoted by $\Gamma_1, \Gamma_2, \ldots, \Gamma_N$
3. Calculate the mean face
$$\bar{\Gamma} = \frac{1}{N} \sum_{i=1}^{M} \Gamma_i \tag{10}$$
4. Subtract mean face from all images, the mean-value free images are then:
$$\Phi_i = \Gamma_i - \bar{\Gamma} \tag{11}$$

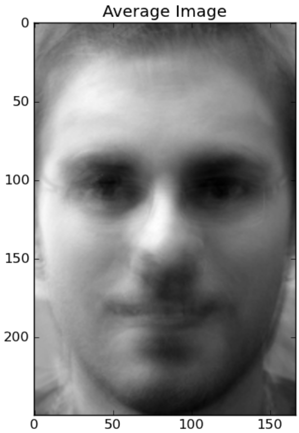
Eigenface: Example

21 training images (above) and 7 test images (below)



Eigenface: Example

Mean image $\bar{\Gamma}$ over all training images



Eigenface: Training

5. Arrange the mean-value free images as rows of the matrix

$$X = \begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \vdots \\ \Phi_N \end{pmatrix} \tag{12}$$

6. Since the rows in X already have a mean value of 0, the covariance of X can be calculated as

$$C = X^T \cdot X \tag{13}$$

   The next step would be the calculation of the Eigenvectors and Eigenvalues of C, but ...

Remarks on Eigenvector and Eigenvalue Calculation

- Note that covariance C is of size $(d \times d)$, where d is the number of pixels in an image.
- For standard resolution images it is impossible to calculate the d eigenvectors and eigenvalues from this matrix.
- As described in [Turk and Pentland, 1991] the number of relevant Eigenvectors (Eigenvectors with non-marginal Eigenvalues) is below N, which is the number of images.
- The N most relevant Eigenvectors can be calculated from the $(N \times N)$-Matrix

$$R = X \cdot X^T \tag{14}$$

as described in the next slide

Eigenface: Training

7. Calculate the N Eigenvectors $\mathbf{v_1}, \ldots, \mathbf{v_N}$ and Eigenvalues $\mu_1, \ldots, \mu_N$ of matrix R. By definition for these Eigenvectors and Eigenvalues

$$X \cdot X^T \cdot \mathbf{v_i} = \mu_i \mathbf{v_i} \tag{15}$$

holds for all $i \in [1, N]$. Left-multiplying both sides of (15) by X^T yields:

$$X^T \cdot X \cdot X^T \cdot \mathbf{v_i} = \mu_i X^T \cdot \mathbf{v_i} \tag{16}$$

Thus for $i \in [1, N]$ the vectors

$$\mathbf{u}_i = X^T \mathbf{v}_i \tag{17}$$

are the Eigenvectors of $C = X^T \cdot X$.

Eigenface: Training

8. Order the Eigenvectors according to decreasing values of the corresponding Eigenvalues and write the ordered Eigenvectors as columns into matrix U:

$$U = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_d] \tag{18}$$

9. Put only the M first columns of U into the PCA transformation matrix

$$U_M = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_M] \tag{19}$$

   The set of M relevant Eigenvectors $\mathbf{u}_1, \cdots, \mathbf{u}_M$ are called Eigenfaces.

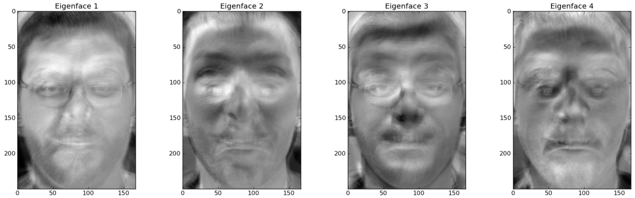
In [Turk and Pentland, 1991] a set of $M = 7$ Eigenfaces has been enough to successfully recognize 16 different faces in images of size (256×256).

Eigenface: Recognition

- The Eigenfaces $\mathbf{u}_1, \cdots, \mathbf{u}_M$ span a linear vector space, called Eigenspace.
- For recognition all training images and the query image are transformed into the Eigenspace.
- In the Eigenspace a nearest neighbour strategy is applied to find the face, which is closest to the query-image.

Eigenface: Example

The 4 Eigenfaces used in this experiment:



Eigenface: Recognition

1. Calculate for each normalized training image Φ_i (see equation (11)) its representation in Eigenspace

$$\mathbf{w}_i = [\omega_{1,i}, \omega_{2,i}, \ldots, \omega_{M,i}], \quad \text{where} \quad \omega_{k,i} = \mathbf{u}_k^T \Phi_i^T \tag{20}$$

2. The normalized version Φ of the query-image Γ is also projected into Eigenspace:

$$\mathbf{w} = [\omega_1, \omega_2, \ldots, \omega_M], \quad \text{where} \quad \omega_k = \mathbf{u}_k^T \Phi^T \tag{21}$$

3. Determine the training image Φ_j which is closest to the query-image:

$$j = argmin_i \left(d(\mathbf{w}, \mathbf{w}_i) \right) \tag{22}$$

where $d(\mathbf{w}, \mathbf{w}_i\}$ is the euclidean distance between $\mathbf{w}$ and $\mathbf{w}_i$.

Eigenface: Recognition

4. If there exist more than one image per person in the training data, then the distance between **w** and the mean-image-per-person

$$\bar{\mathbf{w}}_j = \frac{1}{|W_j|} \sum_{i \in W_j} \mathbf{w}_i \tag{23}$$

is used in equation (22), where W_j is the set of all image indices, that belong to person j.

5. If

$$d_{min} = \min_i \left(d(\mathbf{w}, \mathbf{w}_i) \right) \quad \text{or} \quad d_{min} = \min_j \left(d(\mathbf{w}, \bar{\mathbf{w}}_j) \right) \tag{24}$$

is larger than a predefined threshold T, it is assumed that the query-image is of an unknown person.

Eigenface: Recognition

6. Due to the projection into a low-dimensional space it can happen that a non-face image is mapped closely to one of the training images in Eigenspace. In order to check if the found image Φ_f is a face image its distance

$$d_{min} = \min_i \left(d(\Phi, \Phi_f) \right) \tag{25}$$

to the query-image in the original space is calculated and compared to a threshold S.

Eigenface: Example

Query-image (left) and found best match (right)

Drawbacks

- For a given number M of dimensions in the Eigenspace, PCA finds the best projection w.r.t. Reconstruction MSE (equation (9)), but PCA is unsupervised, i.e. class labels are ignored. See following slides for the corresponding effects.
- Not robust, if
  - objects in the image are not aligned
  - background varies
  - illumination varies
- The approach assumes that data is gaussian distributed

$\Rightarrow$ However, in the case of cropped images of equal illumination, the approach performs well. It can also be applied to other objects of the same category.

Linear Discriminant Analysis (LDA)

- LDA can be considered as an extension of PCA
- Like PCA, LDA transforms data into a low-dimensional subspace. Both methods constitute dimensionality reduction.
- LDA incorporates class labels and is therefore a supervised method.
- LDA is also called Fishers Linear Discriminant Analysis (FLDA)
- The Fisherfaces approach was introduced und evaluated in [Belhumeur et al., 1997].

PCA and LDA: High-Level Comparison

PCA

- Finds best subspace w.r.t. minimizing the Reconstruction MSE of the training data.

LDA

- Finds a discriminant subspace such that class separability is maximized.
- Strategy: Find subspace in which
  - scatter between images of same class is minimized
  - scatter between images of different classes is maximized

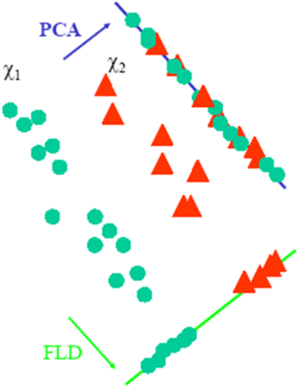


Figure: Source: www1.cs.columbia.edu/~belhumeur/courses/biometrics/2010/eigenfisherfaces.ppt

Varying illumination in faces of same class

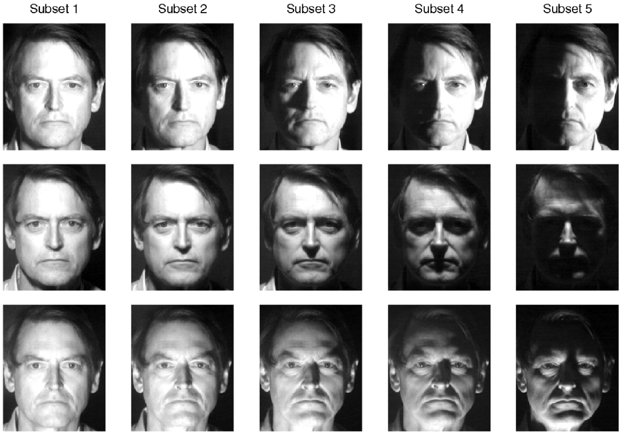


Figure: Faces of same class but different illumination may have larger distance to each other, than faces of different classes (persons) but identical illumination. *Image Source: [Belhumeur et al., 1997]*

Eigenface and Fisherface Comparison [Belhumeur et al., 1997]

- 330 images of 5 people, subdivided into 5 subsets according to varying lighting conditions (see figure in previous slide)
- Higher lighting variations for higher subset index.
- Leave-one-out testing: Choose one test image and apply all others for training. In each iteration a new test image is chosen.

Extrapolating from Subset 1				
Method	Reduced Space	Error Rate (%)		
		Subset 1	Subset 2	Subset 3
Eigenface	4	0.0	31.1	47.7
	10	0.0	4.4	41.5
Eigenface w/o 1st 3	4	0.0	13.3	41.5
	10	0.0	4.4	27.7
Correlation	29	0.0	0.0	33.9
Linear Subspace	15	0.0	4.4	9.2
Fisherface	4	0.0	0.0	4.6

General measure for class separability

- Training Set

$$\mathcal{X} = \{\mathbf{x}^t, r^t\}_{t=1}^N$$

where $r^t = 1$ if $\mathbf{x}^t \in C_1$ (green dots) and $r^t = 0$ if $\mathbf{x}^t \in C_2$ (orange triangles)

- Mean of C_1 in original space:

$$\boldsymbol{\mu}_1 = \frac{\sum_t \mathbf{x}^t r^t}{\sum_t r^t}$$

- Mean of C_2 in original space:

$$\boldsymbol{\mu}_2 = \frac{\sum_t \mathbf{x}^t (1 - r^t)}{\sum_t (1 - r^t)}$$

- Mean of all classes:

$$\boldsymbol{\mu} = \frac{\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2}{2}$$

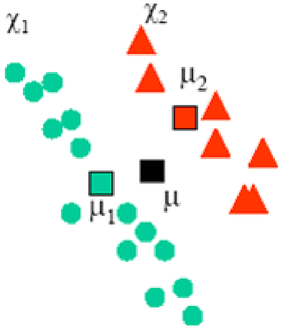


Figure: Source: www1.cs.columbia.edu/~belhumeur/courses/biometrics/2010/eigenfisherfaces.ppt

General measure for class separability

- Intraclass scatter S_1 and covariance matrix Σ_1 of class C_1:

$$S_1 = \sum_t (\mathbf{x}^t - \boldsymbol{\mu}_1)(\mathbf{x}^t - \boldsymbol{\mu}_1)^T r^t \quad , \quad \Sigma_1 = \frac{S_1}{\sum_t r_t} \tag{26}$$

- Intraclass scatter S_2 and covariance matrix Σ_2 of class C_2:

$$S_2 = \sum_t (\mathbf{x}^t - \boldsymbol{\mu}_2)(\mathbf{x}^t - \boldsymbol{\mu}_2)^T (1 - r^t) \quad , \quad \Sigma_2 = \frac{S_2}{\sum_t (1 - r_t)} \tag{27}$$

- For a good class separability
  - the distance between the means μ_1 and μ_2 should be large
  - the intraclass scatter within each class should be small
- Thus

$$\frac{|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2|^2}{|S_1 + S_2|} \tag{28}$$

should be large.

Goal of LDA

Goal of LDA: Find a projection from a high dimensional data space into a low dimensional target space such that the class separability according to the criteria in equation (28) is maximized in target space.

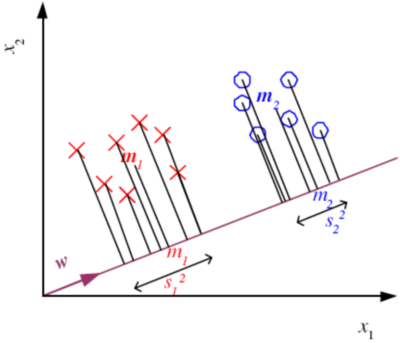


Figure: 1-dimensional target space is defined by **w**. Then the goal is to find the best **w**. Image source [Alpaydin, 2010]

Special Case: Projection of 2-Class data from d-dimensional space into 1-dimensional space

- Mapping of d-dimensional vector $\mathbf{x}$ into 1-dimensional target space:

$$z = \mathbf{w}^T \mathbf{x} \tag{29}$$

- Means in the 1-dimensional target space:

$$m_1 = \frac{\sum_t \mathbf{w}^T \mathbf{x}^t r^t}{\sum_t r^t} = \mathbf{w}^T \boldsymbol{\mu}_1 \tag{30}$$

$$m_2 = \frac{\sum_t \mathbf{w}^T \mathbf{x}^t (1 - r^t)}{\sum_t (1 - r^t)} = \mathbf{w}^T \boldsymbol{\mu}_2 \tag{31}$$

- Scatter in the 1-dimensional target space:

$$s_1^2 = \sum_t (\mathbf{w}^T \mathbf{x}^t - m_1)^2 r^t \tag{32}$$

$$s_2^2 = \sum_t (\mathbf{w}^T \mathbf{x}^t - m_2)^2 (1 - r^t) \tag{33}$$

Special Case: Projection of 2-Class data from *d*-dimensional space into 1-dimensional space

- Fishers linear discriminant is the $\mathbf{w}$ that maximizes

$$J(\mathbf{w}) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2} \tag{34}$$

- Nominator in equation (34):

$$\begin{aligned}(m_1 - m_2)^2 &= (\mathbf{w}^T \boldsymbol{\mu}_1 - \mathbf{w}^T \boldsymbol{\mu}_2)^2 \\ &= \mathbf{w}^T (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \mathbf{w} \\ &= \mathbf{w}^T S_B \mathbf{w} \end{aligned} \tag{35}$$

  where

$$S_B = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \tag{36}$$

  is the Interclass scatter matrix.

Special Case: Projection of 2-Class data from d-dimensional space into 1-dimensional space

Denominator in equation (34):

- For class C_1:

$$\begin{aligned} s_1^2 &= \sum_t (\mathbf{w}^T \mathbf{x}^t - m_1)^2 r^t \\ &= \sum_t \mathbf{w}^T (\mathbf{x}^t - \mu_1)(\mathbf{x}^t - \mu_1)^T \mathbf{w} r^t \\ &= \mathbf{w}^T S_1 \mathbf{w} \end{aligned} \tag{37}$$

  where S_1 is the Intraclass Scatter of class C_1, as defined in equation(26).

- Similarly for class C_2:

$$s_2^2 = \mathbf{w}^T S_2 \mathbf{w} \tag{38}$$

  where S_2 is the Intraclass Scatter of class C_2, as defined in equation(27).

- Entire Denominator:

$$s_1^2 + s_2^2 = \mathbf{w}^T S_W \mathbf{w} \quad \text{where} \quad S_w = S_1 + S_2 \tag{39}$$

Special Case: Projection of 2-Class data from d-dimensional space into 1-dimensional space

- Equation (34) can then be formulated as

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}} = \frac{|\mathbf{w}^T(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)|^2}{\mathbf{w}^T S_W \mathbf{w}} \tag{40}$$

- Calculate first derivation of equation (40) w.r.t. $\mathbf{w}$ and setting it $= 0$ yields the following optimal weights:

$$\mathbf{w} = c \cdot S_w^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{41}$$

The constant c can be chosen arbitrarily, usually $c = 1$.

General Case: $K > 2$ classes, $m > 1$ dimensions in target space

- Training Set

$$\mathcal{X} = \{\mathbf{x}^t, r_i^t\}_{t=1}^N$$

  where $r_i^t = 1$ if $\mathbf{x}^t \in C_i$, else $r_i^t = 0$.

- Mapping of d-dimensional vector $\mathbf{x}$ into m-dimensional target space:

$$\mathbf{z} = W^T \mathbf{x} \tag{42}$$

  where $\mathbf{z}$ is a m-dimensional vector and W is of size $(d \times m)$.

- Intraclass scatter S_i of class C_i:

$$S_i = \sum_t (\mathbf{x}^t - \boldsymbol{\mu}_i)(\mathbf{x}^t - \boldsymbol{\mu}_i)^T r_i^t \tag{43}$$

- Total intraclass scatter:

$$S_w = \sum_{i=1}^K S_i \tag{44}$$

General Case: $K > 2$ classes, $m > 1$ dimensions in target space

- Mean over all classes:

$$\boldsymbol{\mu} = \frac{1}{K} \sum_{i=1}^{K} \boldsymbol{\mu}_i \tag{45}$$

- Interclass Scatter

$$S_B = \sum_{i=1}^{K} N_i (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T \quad \text{where} \quad N_i = \sum_{t=1}^{N} r_i^T \tag{46}$$

- Interclass scatter matrix after projection:

$$W^T S_B W \tag{47}$$

- Matrix of Intraclass scatters after projection:

$$W^T S_W W \tag{48}$$

Both of these scatter matrices are of size $(m \times m)$.

General Case: $K > 2$ classes, $m > 1$ dimensions in target space

- Fishers linear discriminant is the matrix W, that maximizes

$$J(W) = \frac{|W^T S_B W|}{|W^T S_W W|} \tag{49}$$

- Nominator and denominator of this equation are determinants of $(m \times m)$-matrices.
- The determinant of a square-matrix is the product of its Eigenvalues[3]
- An Eigenvalue describes the variance along the corresponding Eigenvector.
- Since the variance (scatter) of the nominator shall be large, and the variance of denominator shall be small, the columns of Fishers Linear Discriminant W are the m Eigenvectors of the largest Eigenvalues of matrix $S_W^{-1} S_B$.

[3] See e.g. http://de.wikipedia.org/wiki/Determinante

Problem when LDA is applied to face recognition

- Problem: Since in face recognition the number of training images N is usually much smaller than the number of features (pixels), Matrix S_W has no full rank. Hence it is singular and can not be inverted.
- Solution: Fisherfaces (proposed in [Belhumeur et al., 1997]):
  - First apply PCA to project the image set into a $N-K$-dimensional space so that the resulting Intraclass Scatter S_W is nonsingular.
  - Apply then the standard FLDA
- The optimal overall projection is then defined by

$$W_{opt}^T = W_{fld}^T U_{pca}^T \tag{50}$$

where U_{pca} is the PCA transformation matrix as defined in equation (19) with $M = N - K$ and W_{fld} whose columns are the Eigenvectors of the m largest Eigenvalues of the matrix

$$\left(U_{pca}^T S_W U_{pca}\right)^{-1} \left(U_{pca}^T S_B U_{pca}\right)$$

References I

Alpaydin, E. (2010).
Introduction to Machine Learning.
MIT Press, 2 edition.

Belhumeur, P. N., Hespanha, J. P., and Kriegman, D. J. (1997).
Eigenfaces vs. fisherfaces: Recognition using class specific linear projection.
IEEE Trans. Pattern Anal. Mach. Intell., 19(7):711–720.

Turk, M. A. and Pentland, A. P. (1991).
Eigenfaces for Recognition.
Journal of Cognitive Neuroscience, 3(1):71–86.