COMPUTATIONAL VISION: Face Recognition

Master in Artificial Intelligence

Department of Mathematics and Computer Science

2019-2020

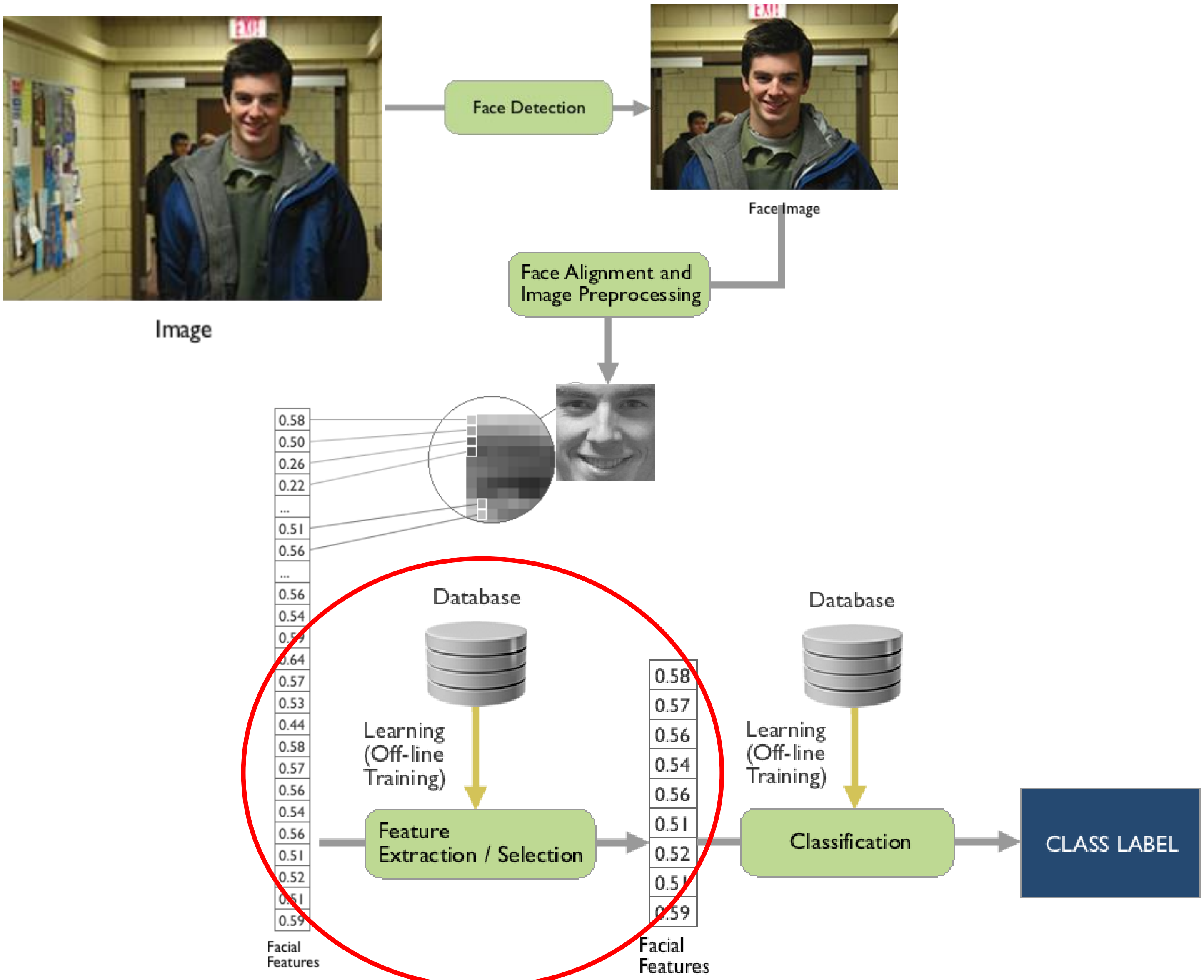
UNIVERSITAT DE BARCELONA

Outline

Face Recognition

1. Principal Component Analysis
2. EigenFaces
3. Linear Discriminant Approach

Automatic Face Recognition



Feature Selection and Extraction

- Feature selection: Choosing $k<d$ important features, ignoring the remaining $d-k$
- Example:

  Given the data $\mathbf{x}=(x_1, x_2,..., x_d)$, where x_i corresponds to a gene information. Choose the k most relevant to represent the data.

- Feature extraction: Transform the original high-dimensional data to lower dimensional data intended to be informative and non-redundant.
- Feature extraction is related to dimensionality reduction.

Dimensionality Reduction

- Dimensionality Reduction is an approach to deal with high dimensional data.

- Project high dimensional data onto a lower dimensional sub-space using linear or non-linear transformations.

$$\mathbf{x}=(x_1, x_2,\ldots, x_d) \xrightarrow{\hspace{4cm}} \mathbf{z}=(z_1, z_2,\ldots, z_k)$$

Reduce dimensionality

$k \ll d$

Advantages of Dimensionality Reduction

1. Reduces time complexity: Less computation
2. Reduces space complexity: Less parameters
3. Simpler models are more robust on small datasets
4. More interpretable; simpler explanation
5. Data visualization (structure, groups, outliers, etc.) if plotted in 2 or 3 dimensions
6. Smart ways of reducing dimensions can “improve” data representation

Linear Approaches

- Linear transformations are simple to compute and tractable

$$\mathrm{b} = P\mathrm{x}$$

k x 1 k x d d x 1 (k<<d)

projection matrix

- Classical linear approaches:
  - Principal Component Analysis (PCA)
  - Linear Discriminant Analysis (LDA) or Fisher Discriminant Analysis (FDA).

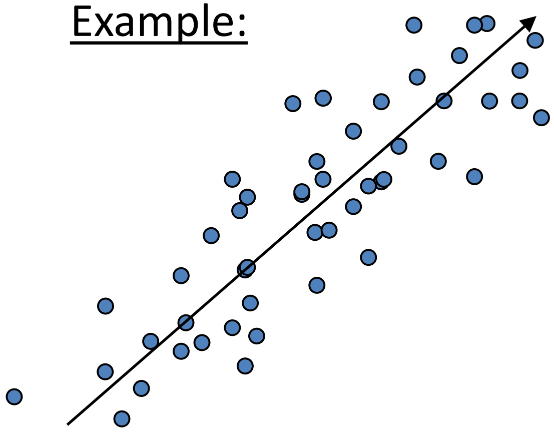
Linear Approaches

- Each dimensionality reduction technique finds an appropriate **transformation by satisfying certain criteria** (e.g., information loss, data discrimination, etc.)

- The goal of *Principal Component Analysis* (PCA) is to reduce the dimensionality of the data while **retaining as much as possible of the variation present in the dataset.**

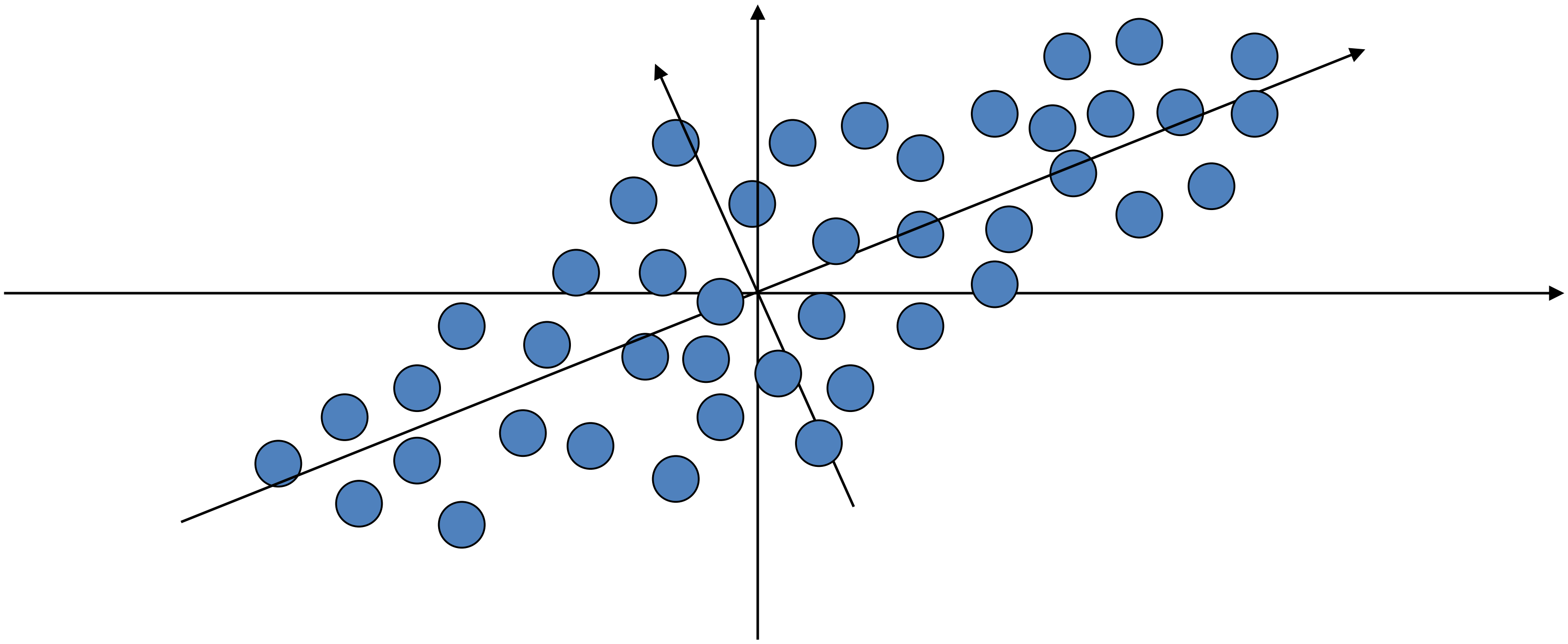
PCA

Example:



- Imagine that we have many examples of the same pattern.
- Data appear clouded, unclear and even redundant.

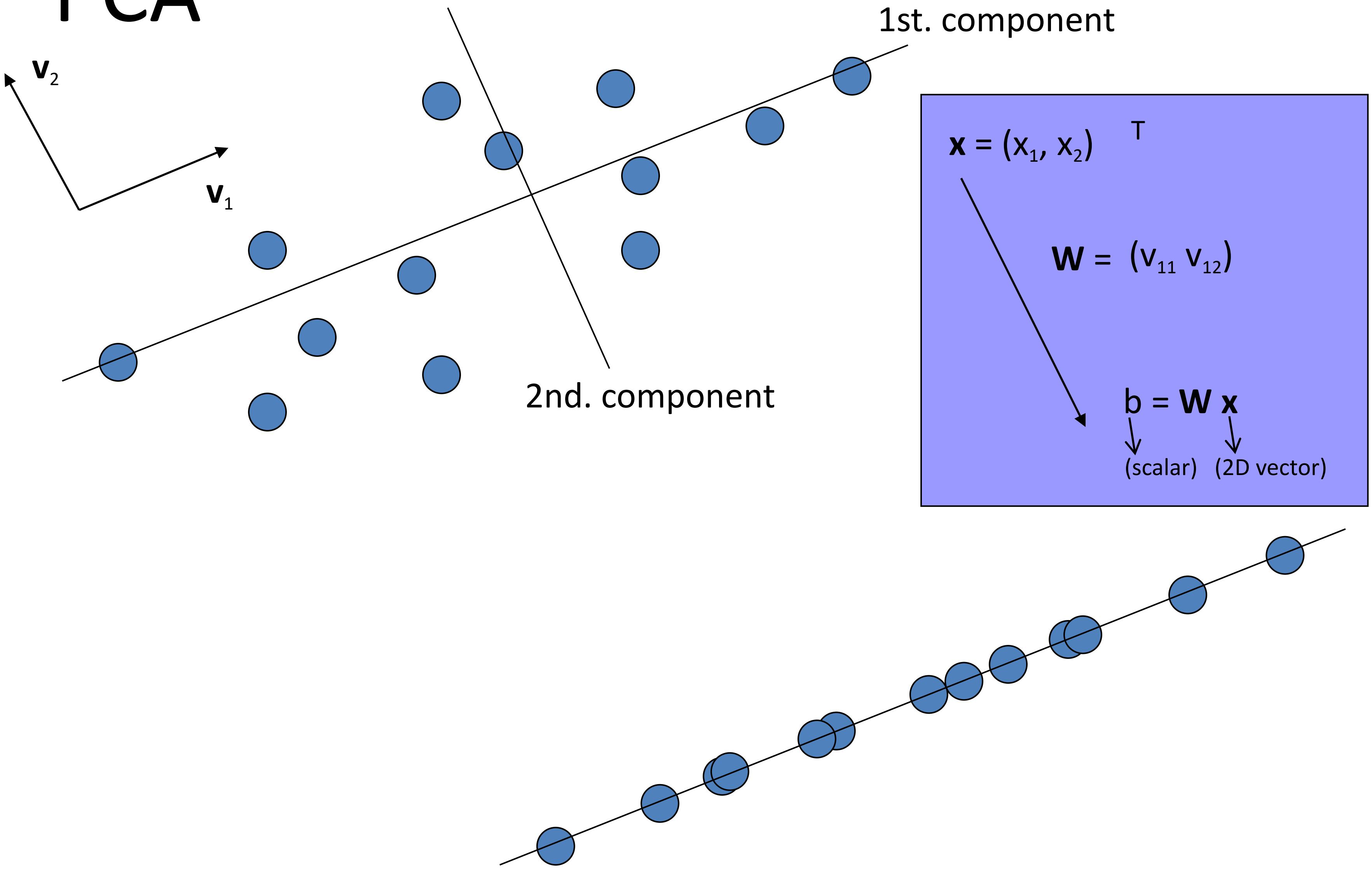
- Goal of PCA: find new representation (basis) to filter the noise and reveal hidden dynamics.



Maximal variance directions

PCA

PCA



$\mathbf{v}_2$
$\mathbf{v}_1$
1st. component
2nd. component
$\mathbf{x} = (x_1, x_2)^T$
$\mathbf{W} = (v_{11}\ v_{12})$
$b = \mathbf{W}\ \mathbf{x}$
(scalar)
(2D vector)

PCA

- Principal Component Analysis (PCA)
  - Mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of **uncorrelated** variables called *principal components*.
  - The first principal component accounts for **as much of the variability in the data as possible**, and each succeeding component accounts for as much of the remaining variability as possible.

What PCA does?

- Find new axis
- Decide which are significant
- Form a new coordinate system defined by the significant axis
- Lower dimensions for new representation
- Map data to the new space

  → Compressed Data

PCA algorithm

1. Get data: $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, *d*-dimensional column vectors.
2. Compute the mean: $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i$
3. Substract the mean from data $\Phi_i = \mathbf{x}_i - \boldsymbol{\mu}$ and build *d x n* matrix $A = [\Phi_1\ \Phi_2\ \ \ldots\ \ \Phi_n]$
4. Calculate the covariance matrix $C = AA^T$ (*d x d* matrix)
5. Calculate the eigenvectors $\mathbf{u}_i$ and eigenvalues λ_i of the covariance matrix *C*.
   Order eigenvectors $U = [\mathbf{u}_1\ \mathbf{u}_2\ \ \ldots\ \ \mathbf{u}_d]$ with descending order $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$
6. Choose *k* components and form a matrix of eigenvectors $W = [\mathbf{u}_1\ \mathbf{u}_2\ \ \ldots\ \ \mathbf{u}_k]$
7. Derive the new data set: $\mathbf{b} = \mathrm{W}^T(\mathbf{x} - \boldsymbol{\mu})$

Expressing points using eigenvectors

- Since the covariance matrix C is symmetric, $\{\mathbf{u}_1, \mathbf{u}_2, \quad ..., \quad \mathbf{u}_d\}$ form a basis, that we represent in a *d x d* matrix:

$$U = [\mathbf{u}_1 \; \mathbf{u}_2 \quad ... \quad \mathbf{u}_d]$$

- Any vector **x** or actually (**x**-μ), can be written as a linear combination of the eigenvectors:

$$\boxed{\mathbf{x} - \boldsymbol{\mu} = b_1\mathbf{u}_1 + b_2\mathbf{u}_2 + ... + b_d\mathbf{u}_d = \sum_{i=1}^{d} b_i\mathbf{u}_i}$$

- The coefficients b_i can be computed as follows:

$$\mathbf{b} = U^T(\mathbf{x} - \boldsymbol{\mu})$$

New data representation

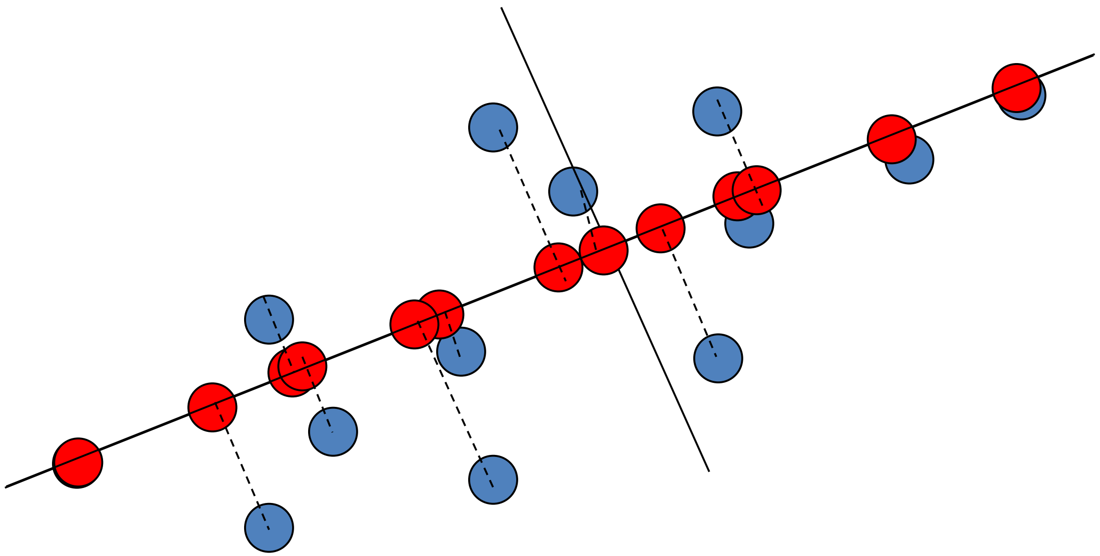
- For dimensionality reduction we keep only *k* vectors, corresponding to the *k* largest eigenvectors.

- Applying the *d x k* matrix $W = [\mathbf{u}_1\ \mathbf{u}_2\ \ \dots\ \ \mathbf{u}_k]$ we project data into the *k*-dimensional space.

$$\mathbf{b} = W^T(\mathbf{x} - \boldsymbol{\mu})$$

New data representation

$$\boxed{\mathbf{b} = [b_1\ \ b_2\ \ \dots\ \ b_k]^T}$$



Reconstruction

- So, then the point x is now $\hat{x}$:

$$\hat{\mathbf{x}} - \boldsymbol{\mu} = \sum_{i=1}^{k} b_i \mathbf{u}_i \quad \text{or} \quad \boxed{\hat{\mathbf{x}} = \boldsymbol{\mu} + \sum_{i=1}^{k} b_i \mathbf{u}_i} \text{ where } k \ll d$$

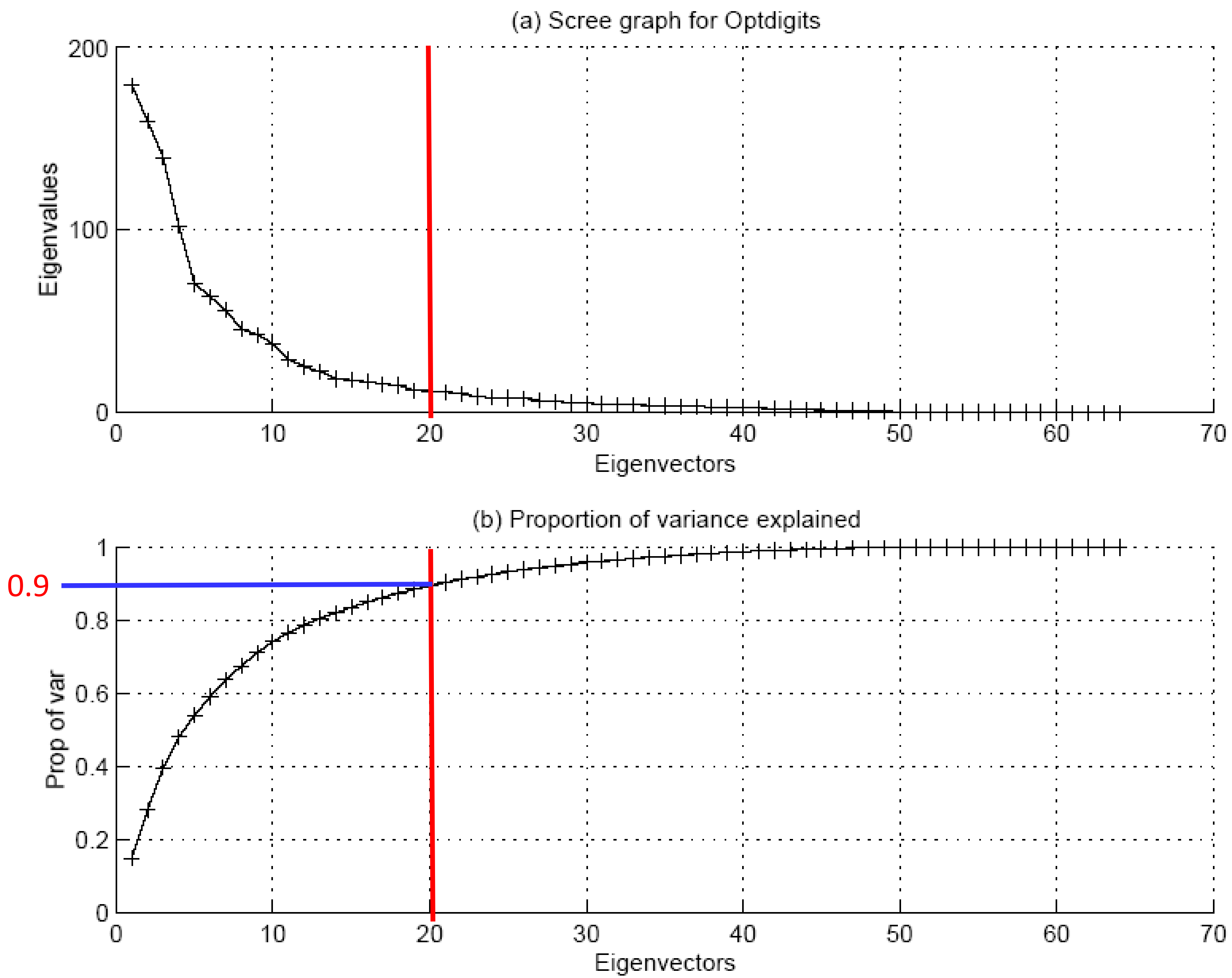
How to choose k?

- Proportion of Variance (PoV) explained (or accumulated percent explained):

$$PoV = \frac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_k + \cdots + \lambda_d}$$

when λ_i are sorted in descending order

- Typically, stop at PoV>0.9



(a) Scree graph for Optdigits
Eigenvalues
200
100
0
0
10
20
30
40
50
60
70
Eigenvectors
(b) Proportion of variance explained
0.9
Prop of var
1
0.8
0.6
0.4
0.2
0
0
10
20
30
40
50
60
70
Eigenvectors

What is the error due to dimensionality reduction?

- We know that:

$$\hat{\mathbf{x}} - \boldsymbol{\mu} = \sum_{i=1}^{k} b_i \mathbf{u}_i$$

- It can be shown that the low-dimensional basis based on principal components **minimizes the reconstruction error**:

$$\mathrm{e} = \left\| \mathbf{x} - \hat{\mathbf{x}} \right\|$$

- The error is equal to: $\mathrm{e} = \frac{1}{2} \sum_{i=k+1}^{d} \lambda_i$

Standarization

- Normalization (or Standarization)
  - The principal components are dependent on the units used to measure the original variables
  - We should always normalize the data prior to use PCA
  - A common normalization method is to transform all the data to have zero mean and unit standard deviation:

$$\frac{(\mathbf{x}_i - \boldsymbol{\mu})}{\sigma}$$

Mean of $\mathbf{x}_i$

Standard deviation of $\mathbf{x}_i$

EIGENFACES FOR FACE RECOGNITION

PCA

- Eigenfaces for Face Recognition
  - M. Turk, A. Pentland, "Eigenfaces for Recognition", *Journal of Cognitive Neuroscience*, vol. 3, no. 1, pp. 71-86, 1991.

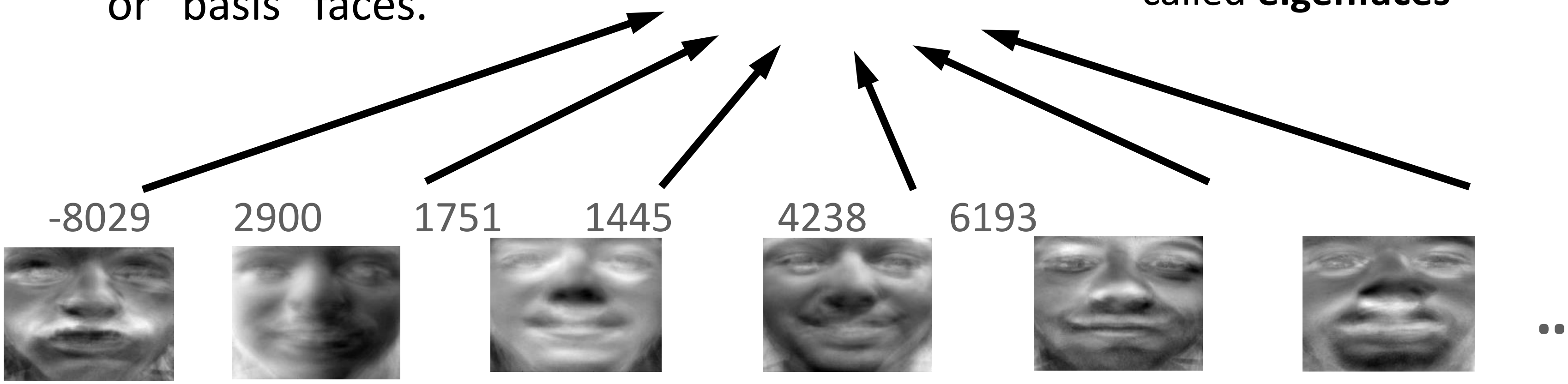
- Face Representation
  - Problems arise when performing recognition in a high-dimensional space.
  - Significant improvements can be achieved by first mapping the data into a *lower dimensionality* space.
  - How to find this lower-dimensional space?

Eigenfaces: the idea

- Think of a face as being a weighted combination of some “component” or “basis” faces.

$$\mathbf{x} = \sum_{i=1}^{k} b_i \mathbf{u}_i$$

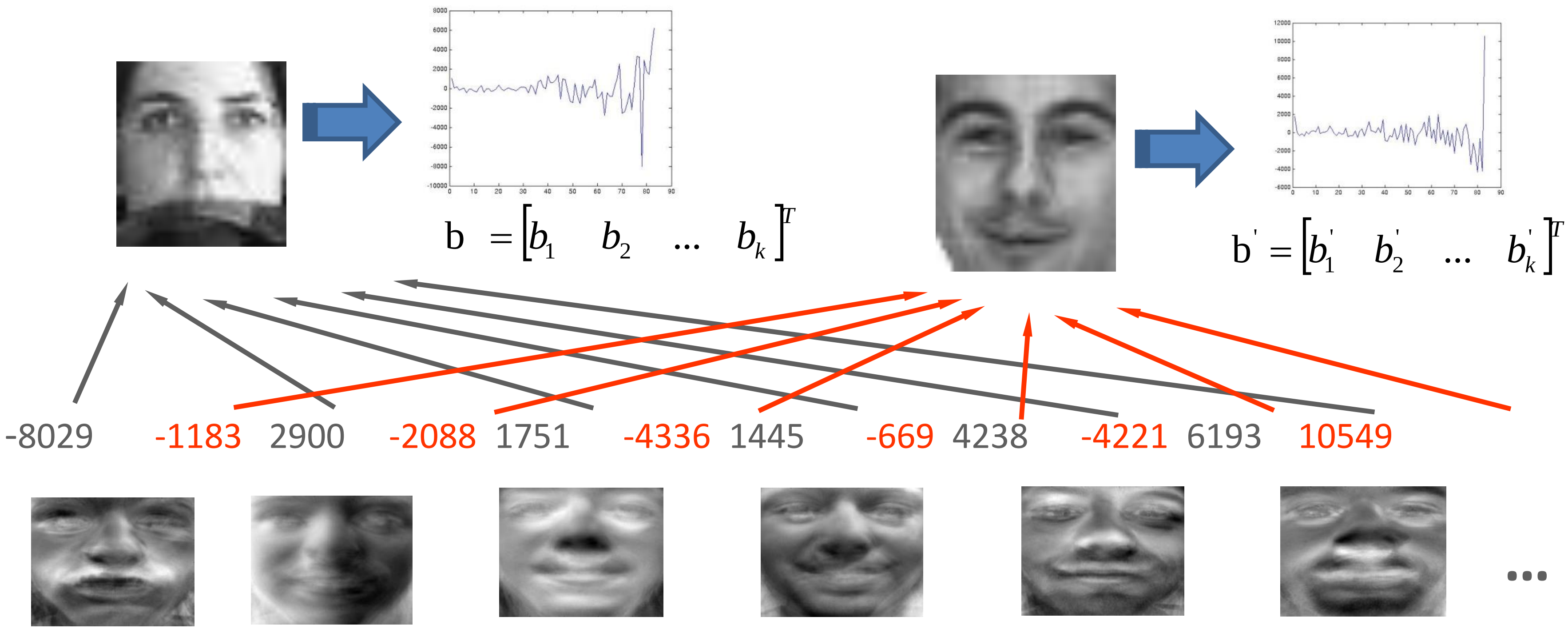
These basis faces are called **eigenfaces**



(Figure by Jeremy Wyatt)

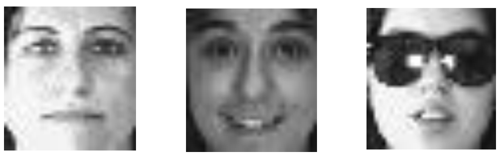
Eigenfaces

- These eigenfaces can be differently weighted to represent any face **x**
- So, we can use the vectors of weights to **represent** faces into a low-dimensional space: $\mathbf{b} = W^T \mathbf{x}$



Learning Eigenfaces

- We take a set of real training faces



…

- Then we use PCA to find (learn) a set of basis faces which best represent the differences between them.
- The statistical criterion for measuring this notion is: “best representation of the differences between the training faces”
- We can then store each face as a set of weights for those basis faces.

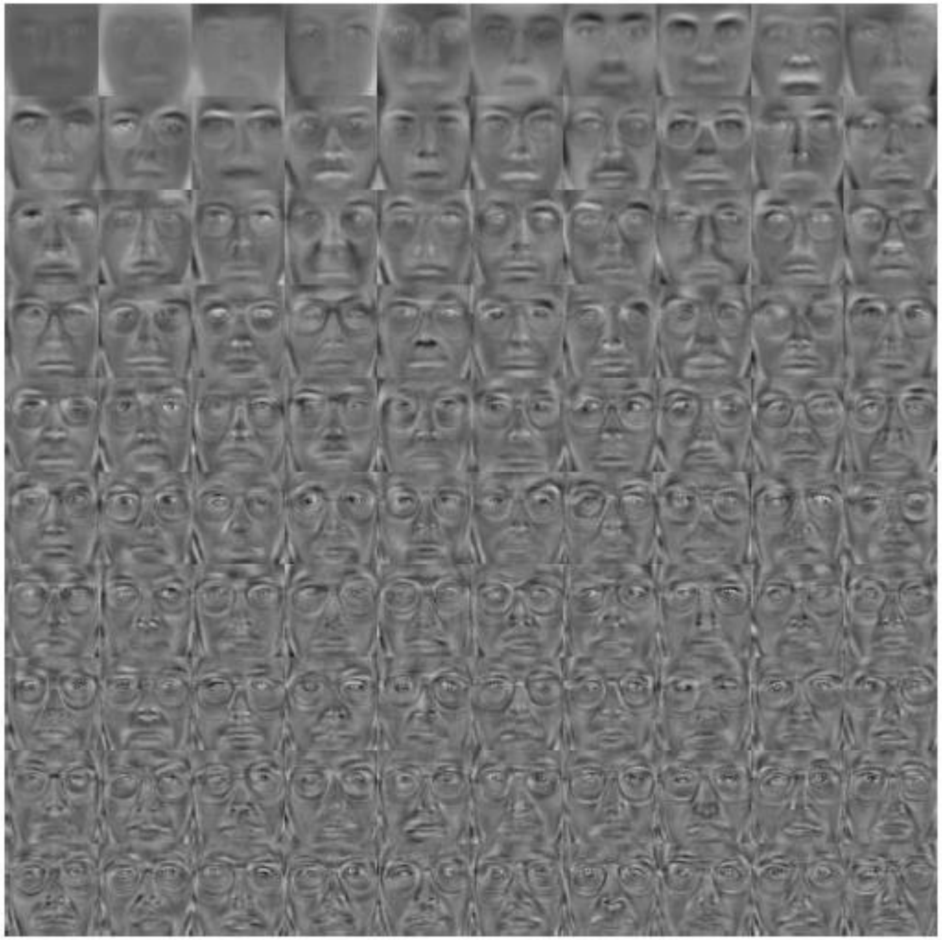
Eigenfaces

Graphical example of a mean Face

Graphical illustration of the eigenvectors of the covariance matrix

$$\{\mathbf{u}_1, \mathbf{u}_2, \quad \dots, \quad \mathbf{u}_d\}$$

These images are the ***Eigenfaces***



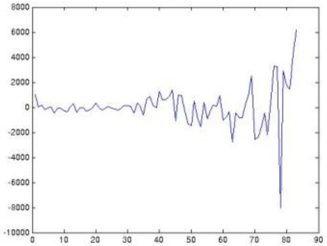
Computation of Eigenfaces

- We transform an image of a face into a vector of weights (projection from **image space** to **face or feature space**)
- There are *d* eigenfaces $\mathbf{u}_i$ in the face space →we choose the best *k* ones.
- We calculate the corresponding weight for every eigenface b_i:

$$\mathbf{b} = \mathrm{W}^T(\mathbf{x} - \boldsymbol{\mu})$$

The j^{th} face in image space is a vector $\mathbf{x}^j$

The corresponding weight vector in face space: $\boldsymbol{b}^j = \left[b_1^j \; b_2^j \; \ldots b_k^j\right]^T$



Computation of Eigenfaces (Trick)

- If the dimension of the data is very large (*d* > *n*):
- The covariance matrix $\mathrm{C} = \mathrm{AA}^{\mathrm{T}}$ is very large → **Not practical!**

- **<u>Alternative:</u>**

1. Consider the matrix $\mathrm{A}^T\mathrm{A}$ (*n* x *n*)
2. Compute the eigenvectors of $\mathrm{A}^T\mathrm{A}$: $\mathrm{A}^T\mathrm{A}\,\mathrm{v}_\mathrm{i} = \lambda_\mathrm{i}\mathrm{v}_\mathrm{i}$

$$\mathrm{A}^T\mathrm{A}\,\mathrm{v}_\mathrm{i} = \lambda_\mathrm{i}\mathrm{v}_\mathrm{i} \Longrightarrow \mathrm{AA}^T\mathrm{A}\,\mathrm{v}_\mathrm{i} = \lambda_\mathrm{i}\mathrm{Av}_\mathrm{i} \Longrightarrow C\mathrm{A}\,\mathrm{v}_\mathrm{i} = \lambda_\mathrm{i}\mathrm{Av}_\mathrm{i}$$

or $\mathrm{Cu}_\mathrm{i} = \lambda_\mathrm{i}\mathrm{u}_\mathrm{i}$ where $\mathrm{u}_\mathrm{i} = \mathrm{Av}_\mathrm{i}$

Thus, AA^T and $\mathrm{A}^T\mathrm{A}$ have the same eigenvalues and their eigenvectors are related as follows: $\mathrm{u}_\mathrm{i} = \mathrm{Av}_\mathrm{i}$

Computation of Eigenfaces (Trick)

Remarks:

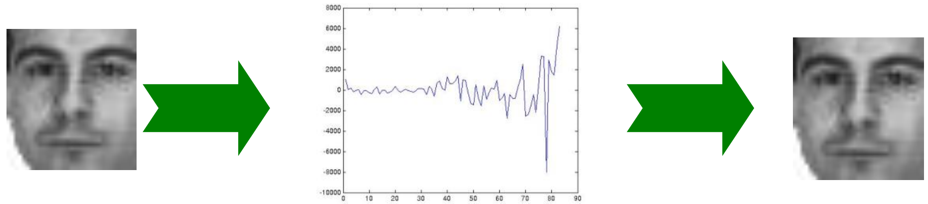
1. AA^T can have up to *d* eigenvalues and eigenvectors
2. $A^T A$ can have up to *n* eigenvalues and eigenvectors
3. The *n* eigenvalues of $A^T A$ (along with their corresponding eigenvectors) are the *n* largest eigenvalues of AA^T (along with their corresponding eigenvectors).

APPLICATIONS: USING EIGENFACES FOR FACE RECOGNITION

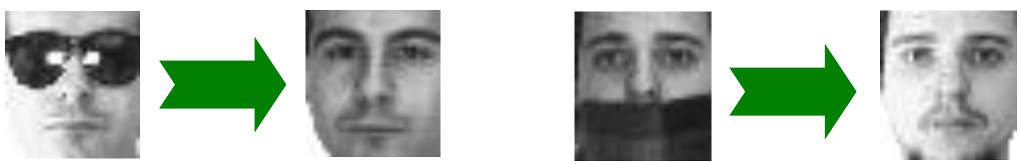
Reconstruction, Recognition & Detection

We can use the eigenfaces in the following ways:

1. We can store and then reconstruct a face from a set of weights



2. We can recognise a new picture of a familiar face



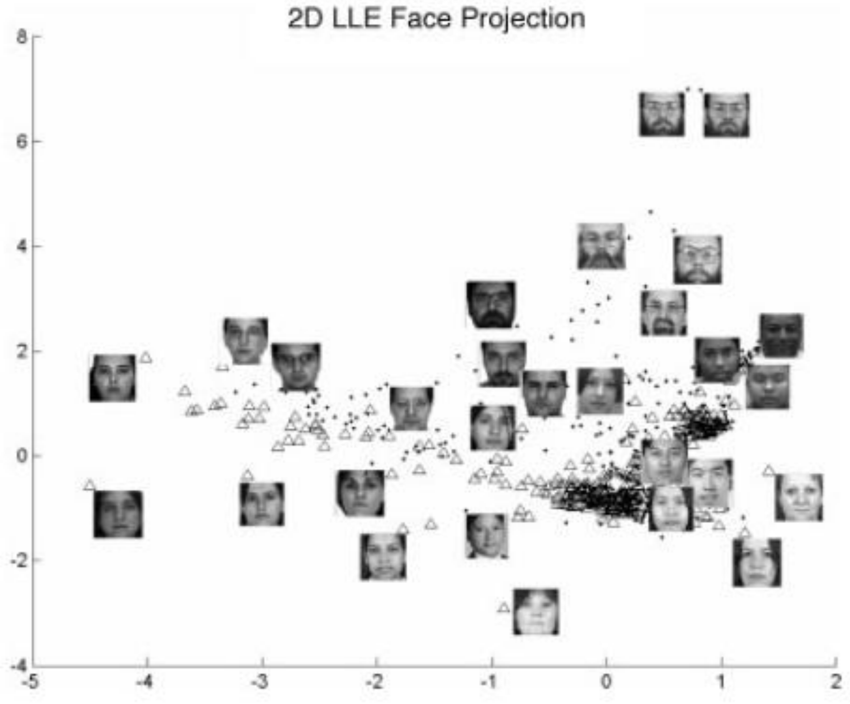
How?

3. We can detect if an image is a face or not

How?

Recognition

- Recognition performed by means of similarity
  - Each face is compared with all the known faces in our system, and the result is the most similar face

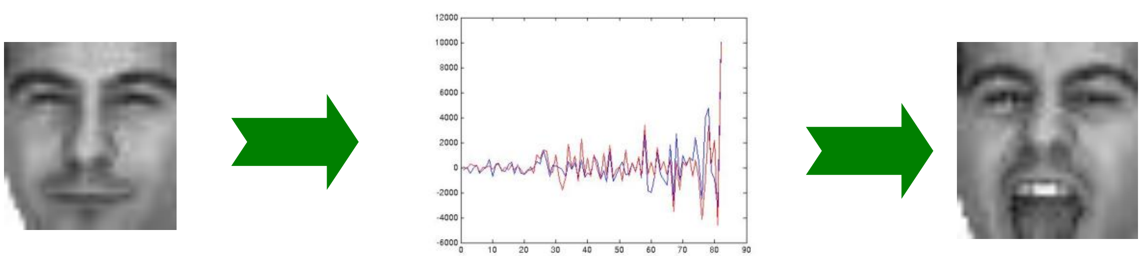


Recognition (more details)

- Given an unknown face image **x*** (after subtracting the mean).
- Project it to face or feature space: $\mathbf{b}^*$
- Compute the Euclidean distance $\boldsymbol{D}$ between the face $\mathbf{b}^*$ and all the training faces $\mathbf{b}^j$ in the feature space

$$D(\mathbf{b}^*, \mathbf{b}^j) = \sqrt{\sum_{i=1}^{k} (b_i^* - b_i^j)^2}, \forall j = 1, \ldots, n,\ j \neq *$$

- Two options:
  - The closest face in feature space is the chosen match
  - K-Nearest Neighbour: predicts test sample‘ class memberships based on the ***k* closest training samples** in the feature space.



Recognition: Remarks

- Find $e_r = \min_j D(\mathbf{b}^*, \mathbf{b}^j)$
  - If $e_r \leq T_r$, then $\mathbf{b}^*$ is recognized as face j from the training set
  - If $e_r > T_r$, then $\mathbf{b}^*$ is not recognized as any face.

- The distance e_r is called Distance **within** face or feature space.

- We can use the Euclidean distance to compute e_r, however, it has been reported that the Mahalanobis distance performs better:

$$D(\mathbf{b}^*, \mathbf{b}^j) = \sqrt{\sum_{i=1}^{k} \frac{1}{\lambda_i} (b_i^* - b_i^j)^2}, \forall j = 1, \ldots, n$$

(variations along all axes are treated as equally significant)

Detection

- Given an unknown image $\mathbf{x}$ (after subtracting the mean)
- Project it to face space: **b**
- Project back to image space:

$$\hat{\mathbf{x}} = \sum_{i=1}^{k} b_i \mathbf{u}_i$$

- Compute the Euclidean distance between the face image $\mathbf{x}$ and $\hat{\mathbf{x}}$

$$e_d = \sqrt{\sum_{i=1}^{d} (\mathrm{x}_i - \hat{\mathrm{x}}_i)^2}$$

- If $e_d < T_d$, then **x** is a face, otherwise **x** is not a face.

Remarks:

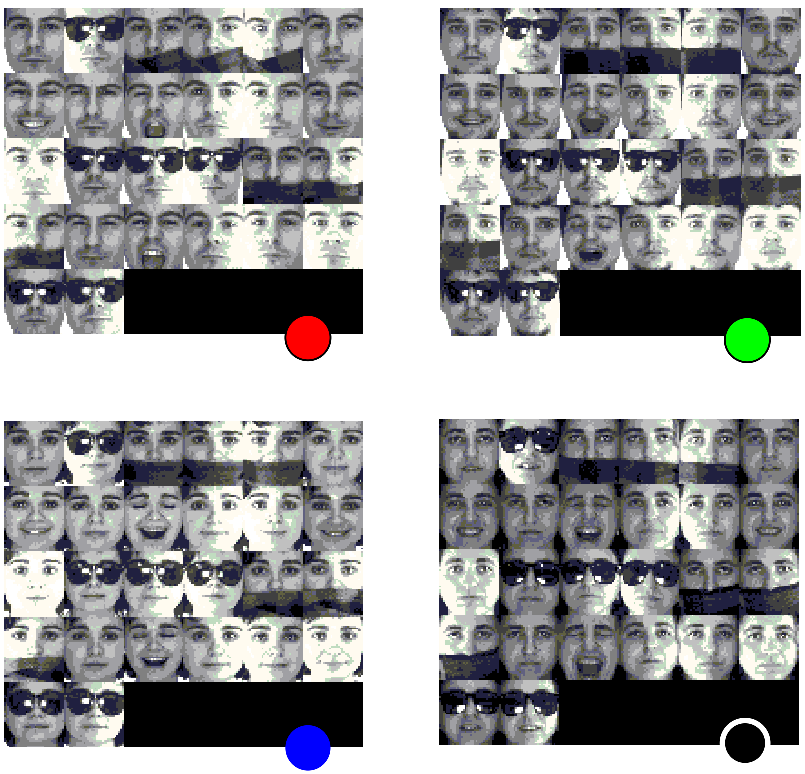
- e_d is called Distance **from** face or feature space.
- We are assuring that face images can be explained using the computed eigenface basis.

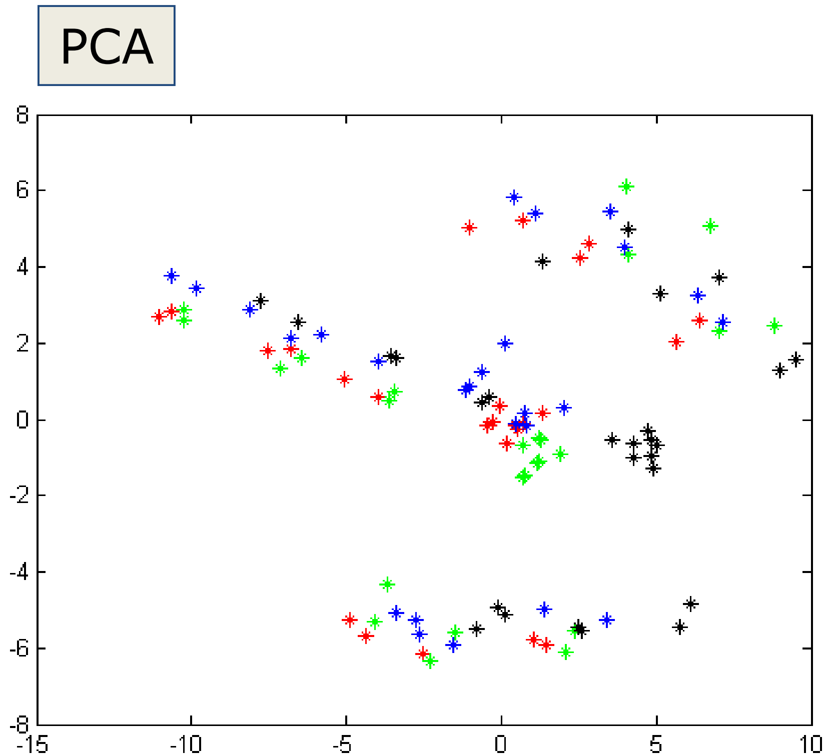
Sumary: PCA for Eigenfaces extraction

- **Non-supervised** feature extraction technique (data labels are not taken into account)
- PCA finds a linear transformation of the data into a lower dimensional space, such that
  - The reconstruction error according to the Euclidean Distance is minimum.
  - Data do not lose much information.
  - Specially appropriated to reduce the noise of the data
- Normally called: Compression-base feature-extraction scheme.

PCA Limitations

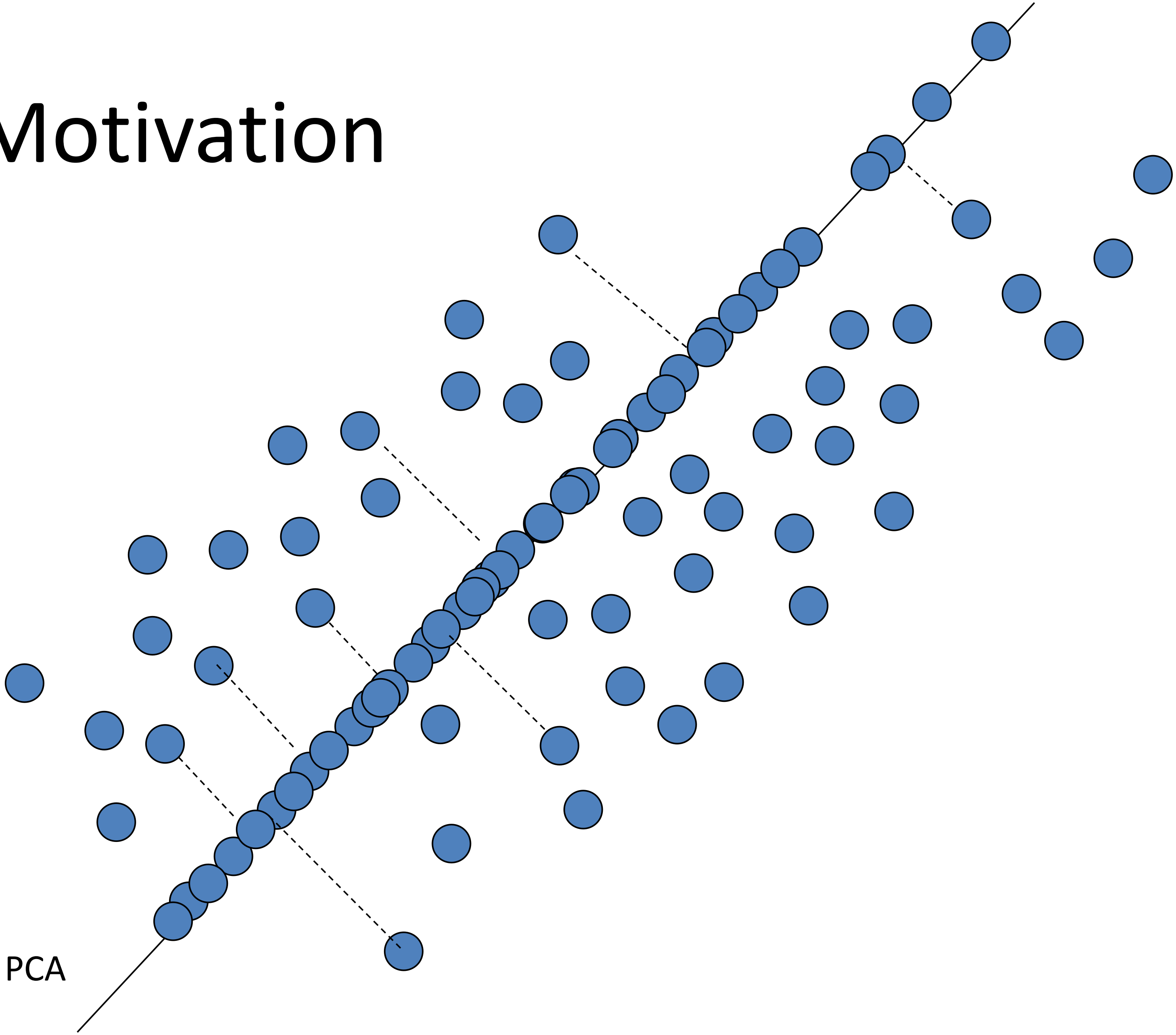
In this case, data could be too mixed to successfully perform **face recognition**.





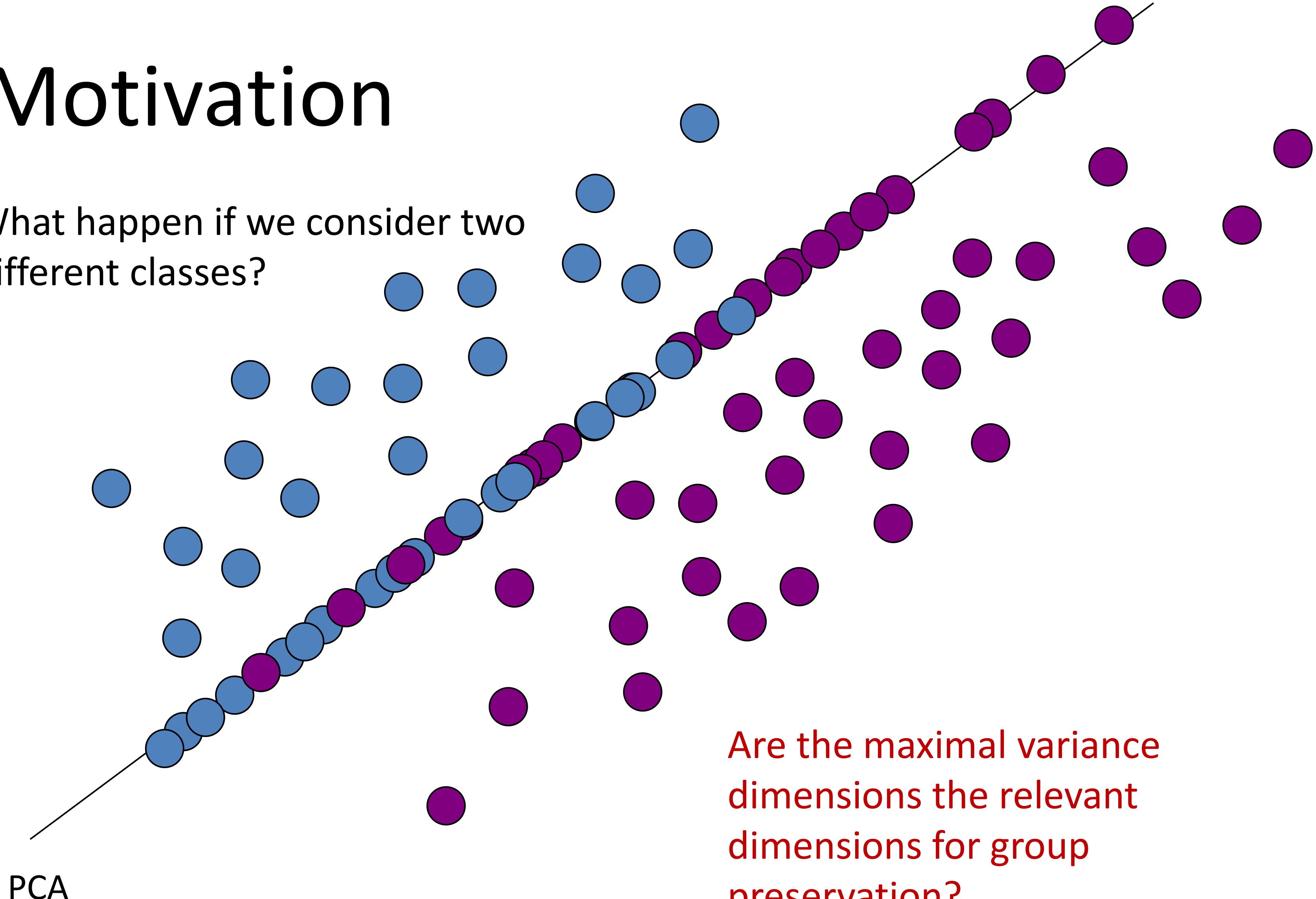
LINEAR DISCRIMINANT ANALYSIS (LDA) & FISHERFACES

Motivation



Motivation

What happen if we consider two different classes?



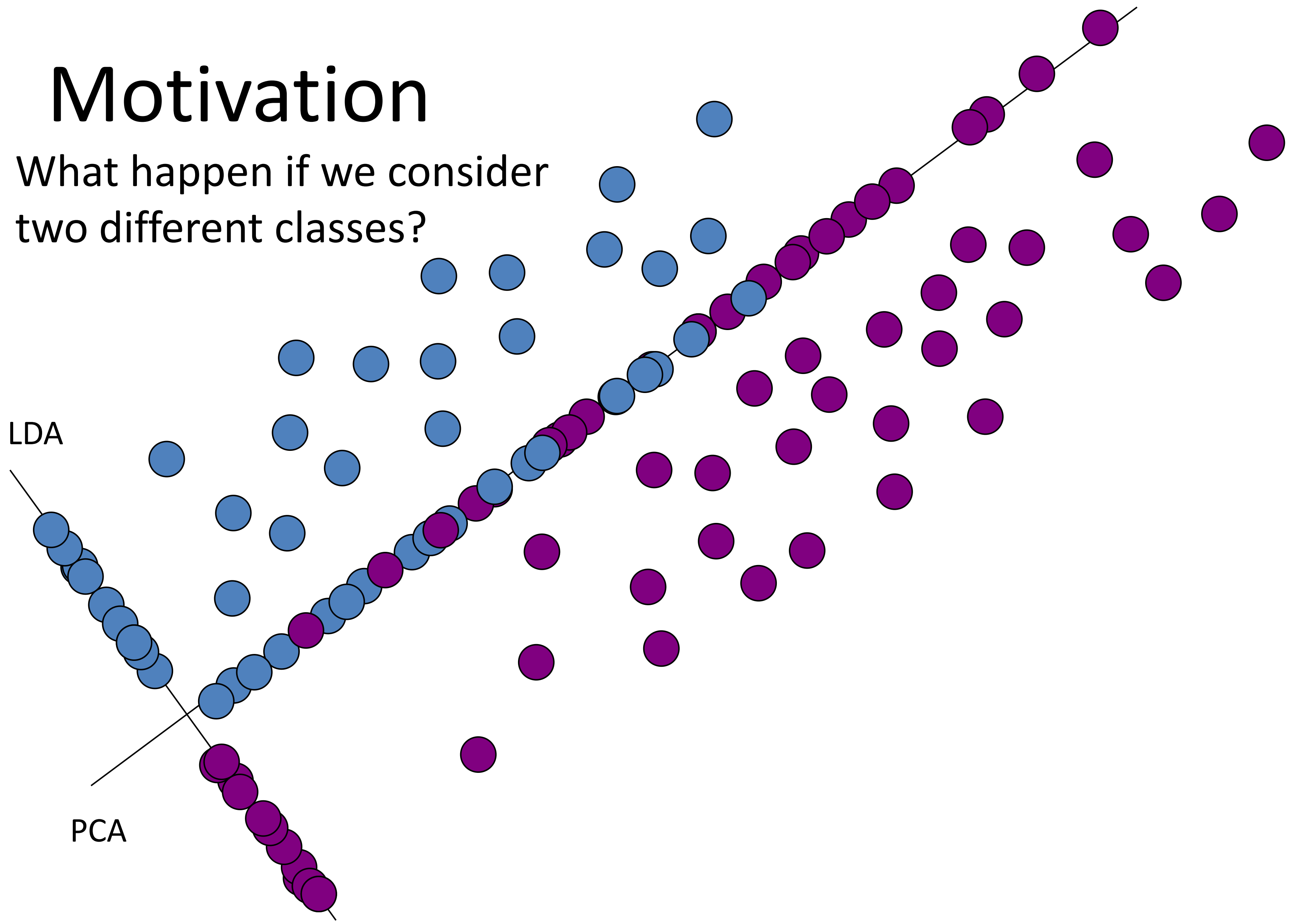
Are the maximal variance dimensions the relevant dimensions for group preservation?

Motivation

Why do we not start with criteria based on **discrimination** from the beginning to make the whole process more consistent?

Motivation

What happen if we consider two different classes?



Linear Discriminant Analysis (LDA)

- LDA is a linear **supervised** feature extraction technique
- Performs dimensionality reduction
- It finds a linear transformation of the data that maximize the class separability of the points.
- More concretely, the method seeks for a new data representation where **points of the same class are as close as possible, while points of different classes are as far as possible**.

⬇

Good for classification purposes!

Linear Discriminant Analysis (LDA)

- Given C classes with M_i the number of samples within class i, i=1,..,C
- Let M be the total number of samples: $M = \sum_{i=1}^{C} M_i$
- Let $\boldsymbol{\mu}_i$ be the mean vector of class i (intra class mean)
- Let $\boldsymbol{\mu}$ be the mean of the entire data set: $\boldsymbol{\mu} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{\mu}_i$

- **Whithin-class scatter matrix:**

$$S_w = \sum_{i=1}^{C} \sum_{j=1}^{M_i} (x_j - \boldsymbol{\mu}_i)(x_j - \boldsymbol{\mu}_i)^T$$

S_W ~ measures the dispersion of the elements that belong to the same class.

- **Between-class scatter matrix:**

$$S_b = \sum_{i=1}^{C} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^T$$

S_B ~ measures the dispersion of the elements that belong to different classes.

LDA

- Find projection (from *d*-dim. space to *f*-dim. space):

$$(x_1, x_2, \ldots, x_d) \rightarrow (y_1, y_2, \ldots, y_f),\ f << d;$$

$$\mathrm{y} = \mathrm{U}^T \mathrm{x}$$

projection matrix

- LDA computes a transformation that **maximizes the between-class scatter while minimizing the within-class scatter.**

- One way to do this:

$$\max \frac{|S_b|}{|S_w|}$$

S_b, S_w: scatter matrices of the projected data y.

Let us see how we solve this problem

LDA

- The LDA solution is given by the eigenvectors of the *generalized eigenvector problem*:

$$S_b \mathrm{u}_\mathrm{k} = \lambda_k S_w \mathrm{u}_\mathrm{k} \longrightarrow \boxed{S_w^{-1} S_b \mathrm{u}_\mathrm{k} = \lambda_k \mathrm{u}_\mathrm{k}}$$

The linear transformation is given by a projection matrix *U* whose columns are the eigenvectors of the above problem.

$$U = \left(\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_\mathrm{d}\right), \quad \mathbf{b} = U^T(\mathbf{x} - \boldsymbol{\mu})$$

Important to note:

- Since S_b has at most rank C-1, the max number of eigenvectors with non-zero eigenvalues is C-1 (i.e., max dimensionality of subspace is C-1)

For proves see: Fisher R.A. *The statistical utilization of multiple measurements*, Annals of Eugenics, 8, 376–386, 1938).

LDA

- Does S_w^{-1} always exist?

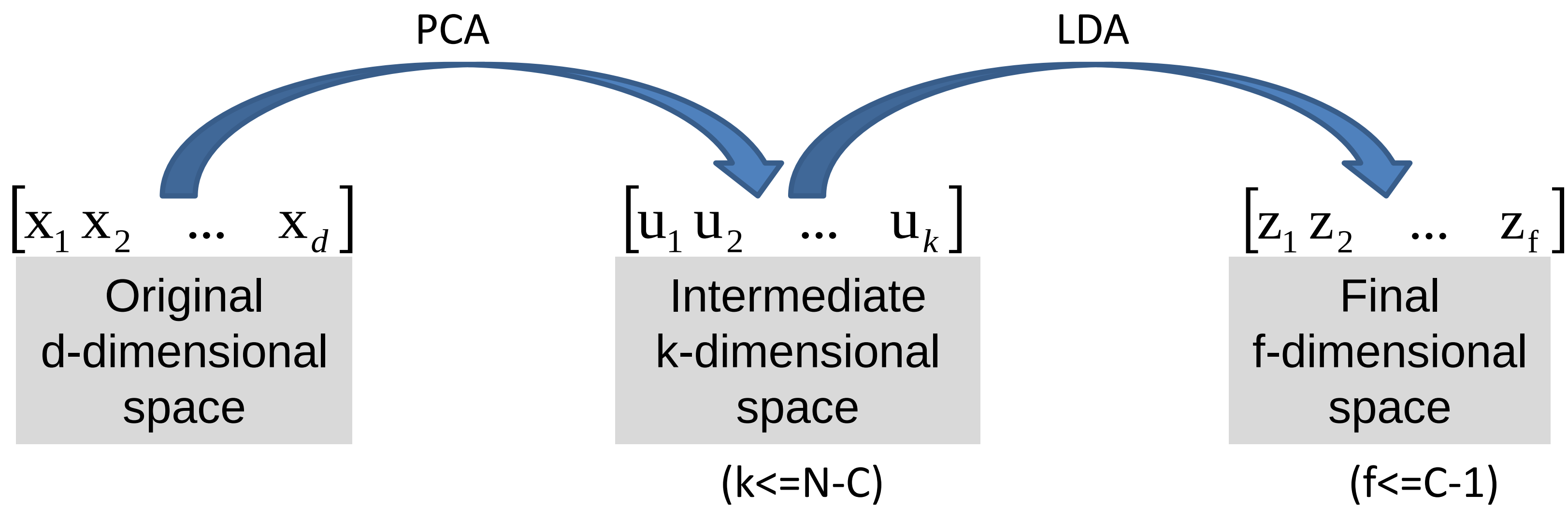
  – If S_w is non-singular:
  we can solve the conventional eigenvalue problem by writing:

$$S_w^{-1} S_b \mathbf{u}_\mathrm{k} = \lambda_k \mathbf{u}_\mathrm{k}$$

  – In practice, S_w is often singular, since the data are image intensity vectors with large dimensionality while the size of the data set is much smaller (n << d)

LDA

- To alleviate this problem, we can use PCA first:



1) PCA is first applied to the data set to reduce its dimensionality.
2) LDA is then applied to find the most discriminative directions.

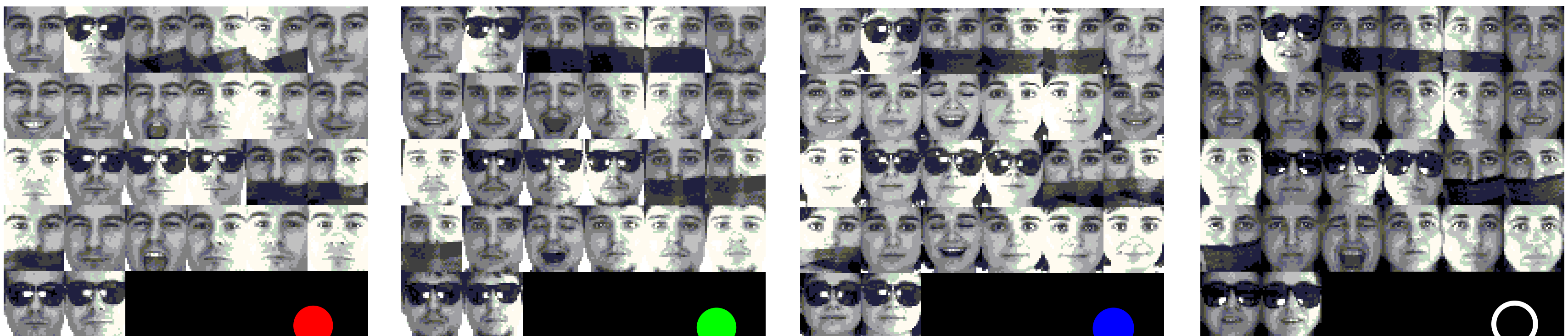
CASE OF STUDY: FISHERFACES FOR FACE RECOGNITION

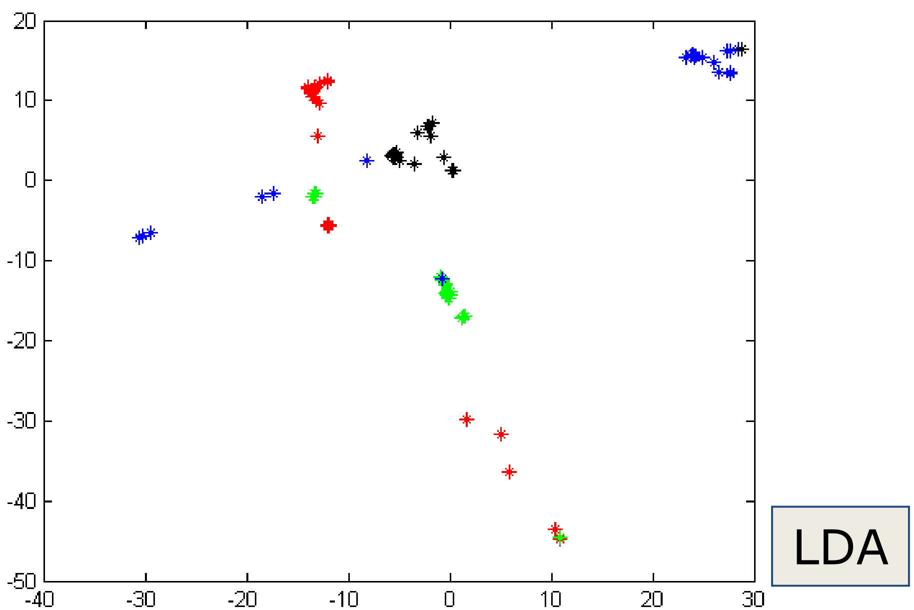
Fisherfaces

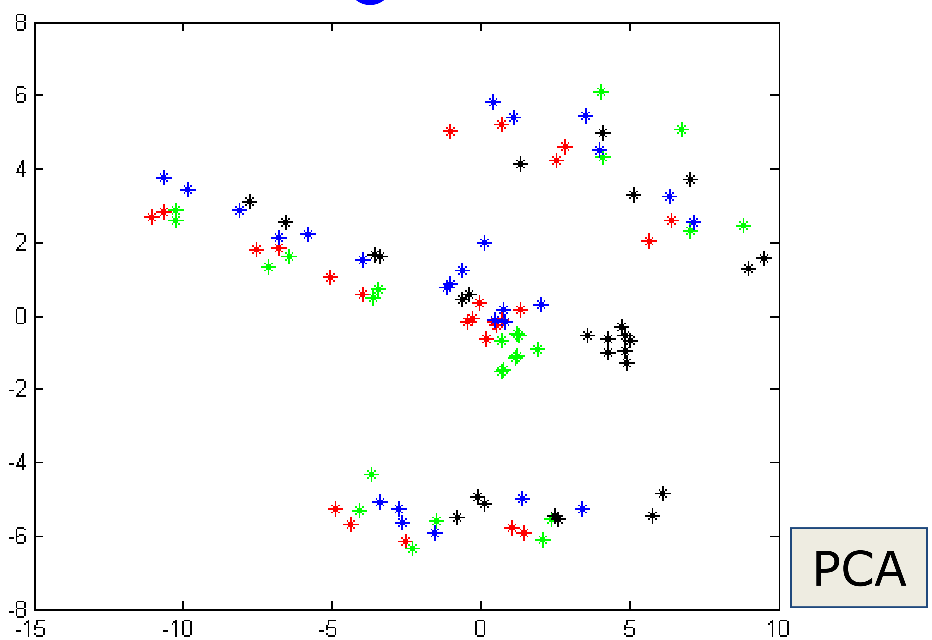
- Fisherfaces is the name of the LDA or Fisher solution for dimensionality reduction.

PCA versus LDA: Subject Recognition

- Clustering effect







PCA versus LDA

- Is LDA always better than PCA for face recognition?
  - There has been a tendency in the computer vision community to prefer LDA over PCA.
  - This is mainly because LDA deals directly with discrimination between classes, while PCA does not pay attention to the underlying class structure.

- <u>**Reference:**</u>
  - Martinez, A. Kak, "PCA versus LDA", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228-233, 2001.

PCA versus LDA

- Main results of this study of Martinez et al.:

    1. When the training set is **small**, PCA can outperform LDA.
    2. When the number of samples is **large** and **representative** for each class, LDA outperforms PCA.

Martinez, A. Kak, "PCA versus LDA", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 228-233, 2001.

PCA versus LDA

- Is LDA always better than PCA for face recognition?

AR Face Database

Fig. 3. Images of one subject in the AR face database. The images (a)-(m) were taken during one session and the images (n)-(z) at a different session.

Experiment preparation

Recognition problem. From AR database:

- Consider 50 different individuals (25 males and 25 females)→ **c**=50 classes.
- Images are transformed into 85x60 pixels arrays.
- To simulate small training data set use:
  - Training set made by: 2 images per subject

    → Total number of samples in the training set **n** = 2*50 = 100.
  - Test set made by: 5 images per subject

    → Total number of samples in the test set = 5*50 = 250.
- From the 7*50 images available, there are 21 ways of building these training and test set

  → 21 runs are done.
- Dimensionality reduction methods:

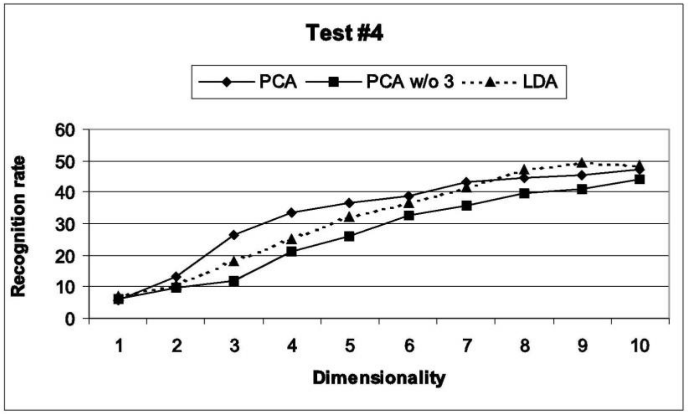
  1) PCA

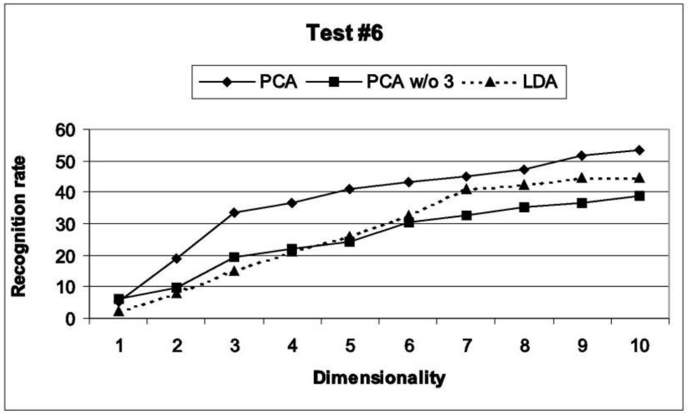
  2) PCA w/o 3 (without the first three eigenvectors)

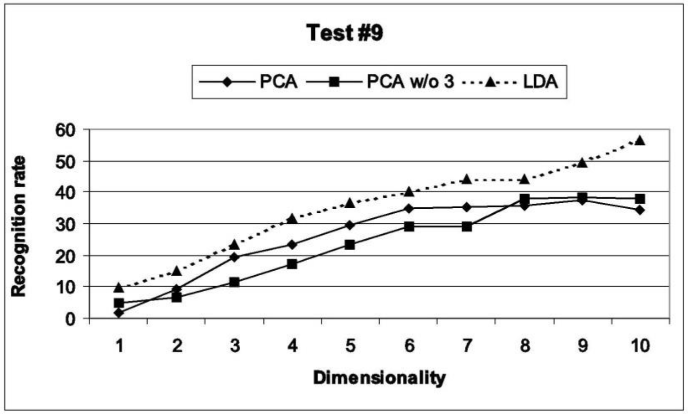
  3) LDA
- Classification: Nearest-neighbor algorithm using standard L_2-norm for the Euclidean distance.

PCA versus LDA

LDA is not always better when training set is **small** (high-dimension space)



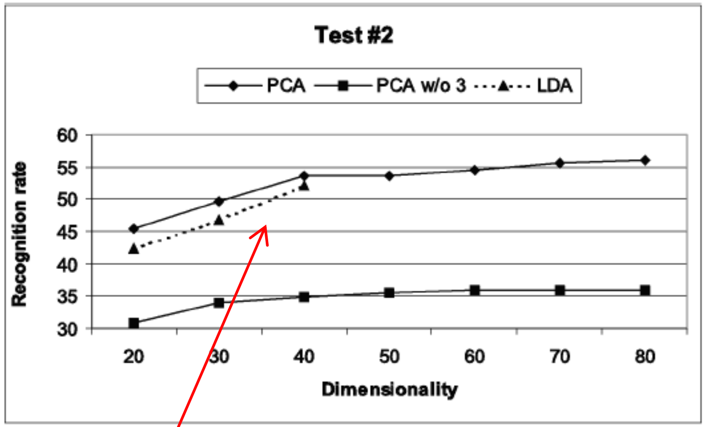


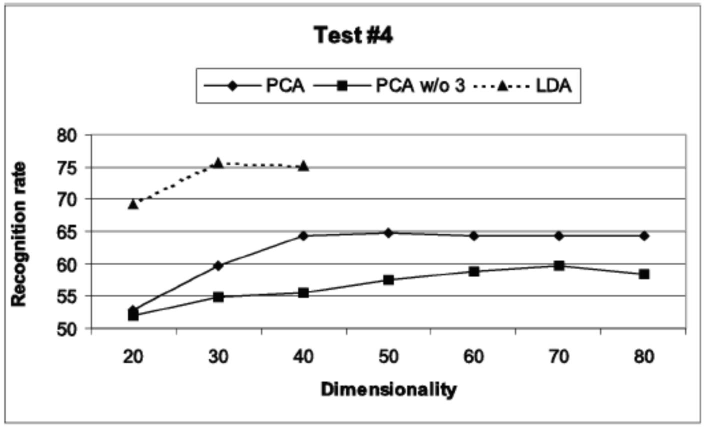


Dimensionality = f parameter.
For every f, all possible values of k parameter are tried (from 15 to 50) and the best is chosen.

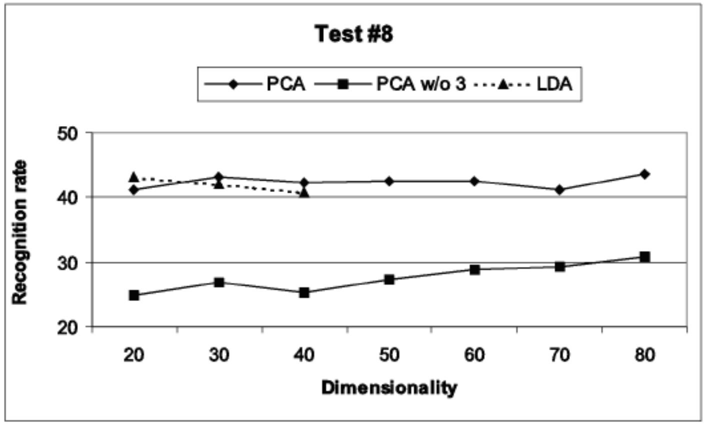
PCA versus LDA

LDA is not always better when training set is **small** (high-dimension space)





The LDA curves do not go beyond f=40! Why?



More tests.

PCA versus LDA

Why the LDA curves do not go beyond f =40?

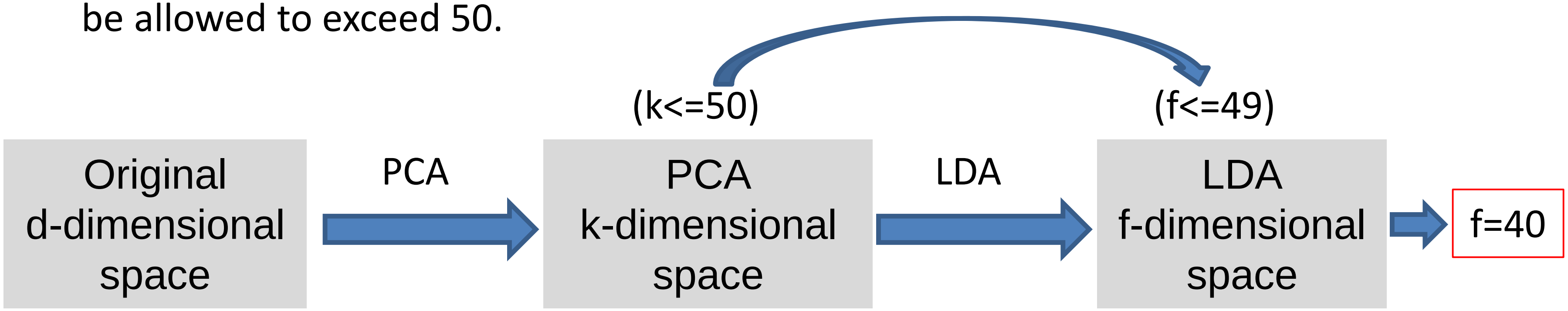
This is dictated by the following two considerations:

- The dimensionality of LDA is upper-bounded by c – 1.

  → Since c=50, this gives an upper bound of 49 for the dimensionality of the LDA space.

- The dimensionality of the underlying PCA space (from which the LDA space is carved out) cannot be allowed to exceed n – c.

  →Since n=100 and c=50, the dimensionality of the underlying PCA space cannot be allowed to exceed 50.

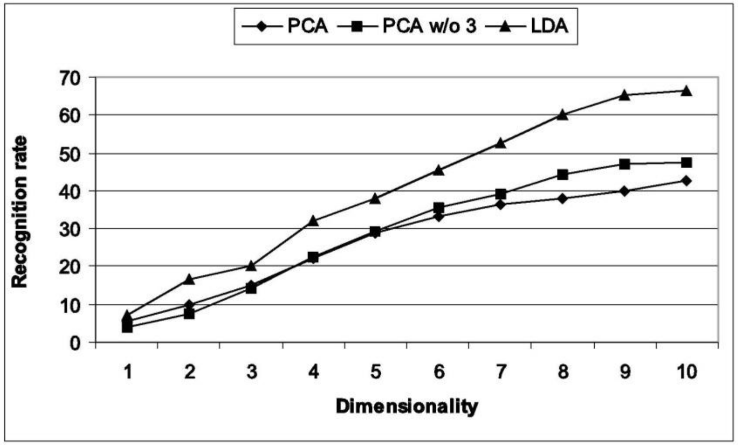


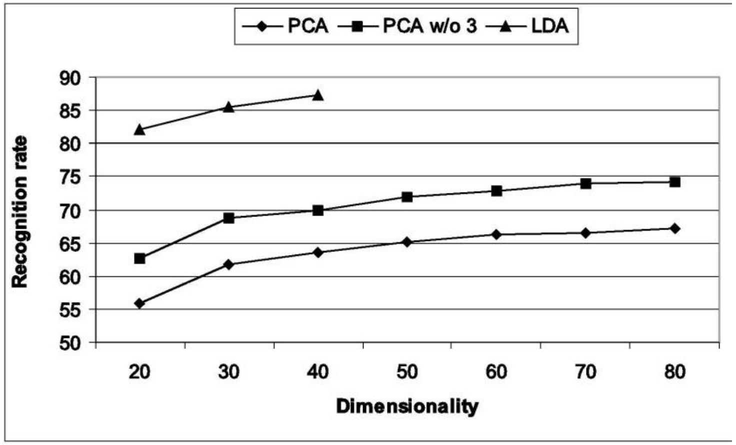
Since it makes no sense to extract a 49 dimensional LDA subspace out of a 50 dimensional PCA space, the dimensionality of the LDA space was arbitrarily hard-limited to 40.

PCA versus LDA

LDA outperforms PCA when training set is **large**

- **Training** set made by: 13 images per subject
  → Total number of samples in the training set: n = 13*50 = 650.
- **Test** set made by: 13 images per subject
  → Total number of samples in the test set: 13*50 = 650.





For each value of f, we tried all values of k from a low of 50 to the maximum allowed value of 600.

References

For dimensionality reduction using PCA:

- R.O. Duda, P.E. Hart and D.G. Stork, "Pattern Classification". Wiley-Interscience Publication (2000). Chapter 3 – Section 3.8.
- "A tutorial on principal component analysis Derivation, Discussion and Singular Value Decomposition“, Jon Shlens.
- “A tutorial on Principal Component Analysis”, Lindsay I Smith.
- “Face Recognition”, Jeremy Wyatt.

References

For face recognition using dimensionality reduction:

- M. Turk, A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, 3(1), pp. 71-86, 1991.
- D. Swets, J. Weng, "Using Discriminant Eigenfeatures for Image Retrieval", IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(8), pp. 831-836, 1996.
- A. Martinez, A. Kak, "PCA versus LDA", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 2, pp. 228-233, 2001.
- K. Ohba and K. Ikeuchi, "Detectability, Uniqueness, and Reliability of Eigen Windows for Stable Verification of Partially Occluded Objects", *IEEE Transactions on Pattern Analysis and Machine Intelligenve, vol. 19, no. 9, pp. 1043-1048,* 1997.
- H. Murase and S. Nayar, "Visual Learning and Recognition of 3D Objects from Appearance", *Interantional Journal of Computer Vision, vol 14, pp. 5-24, 1995.*

References

More interesting readings:

- S. Gong et al., *Dynamic Vision: From Images to Face Recognition, Imperial College* Press, 2001.
- K. Kastleman, *Digital Image Processing, Prentice Hall.*
- F. Ham and I. Kostanic. *Principles of Neurocomputing for Science and Engineering,* Prentice Hall.
- A. Jain, R. Duin, and J. Mao, "Statistical Pattern Recognition: A Review", *IEEE Transactions on Pattern Analysis and Machine Intelligenve, vol. 22, no. 1, pp.* 4-37, 2000.

References

- L. Smith, A Tutorial on Principal Components Analysis, www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf, 2002.
- Jolliffe, I. T. (1986). *Principal Component Analysis*, Springer-Verlag. pp. 487. R. Kramer, Chemometric Techniques for Quantitative Analysis, (1998) Marcel–Dekker.
- Shaw PJA, Multivariate statistics for the Environmental Sciences, (2003) Hodder-Arnold.
- Patra sk et al., J Photochemistry & Photobiology A:Chemistry, (1999) 122:23–31.
- Martinez, A. Kak, "PCA versus LDA", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 2, pp. 228-233, 2001.
- Fisher R.A. *The statistical utilization of multiple measurements*, Annals of Eugenics, 8, 376–386, 1938).
- Belhumeur P. N., Hespanha J. P., and Kriegman D. J., *Eigenfaces vs. Fisherfaces: RecognitionUsing Class Specific Linear Projection.* IEEE Transactions on Pattern analyisi and machine intelligence, 19, 711-720. 1997.

COMPUTATIONAL VISION: Face Recognition

Master in Artificial Intelligence

Department of Mathematics and Computer Science

2019-2020

UNIVERSITAT DE BARCELONA