Principal Component Analysis and Singular Value Decomposition

Volker Tresp, Clemens Otte

Summer 2014

Motivation

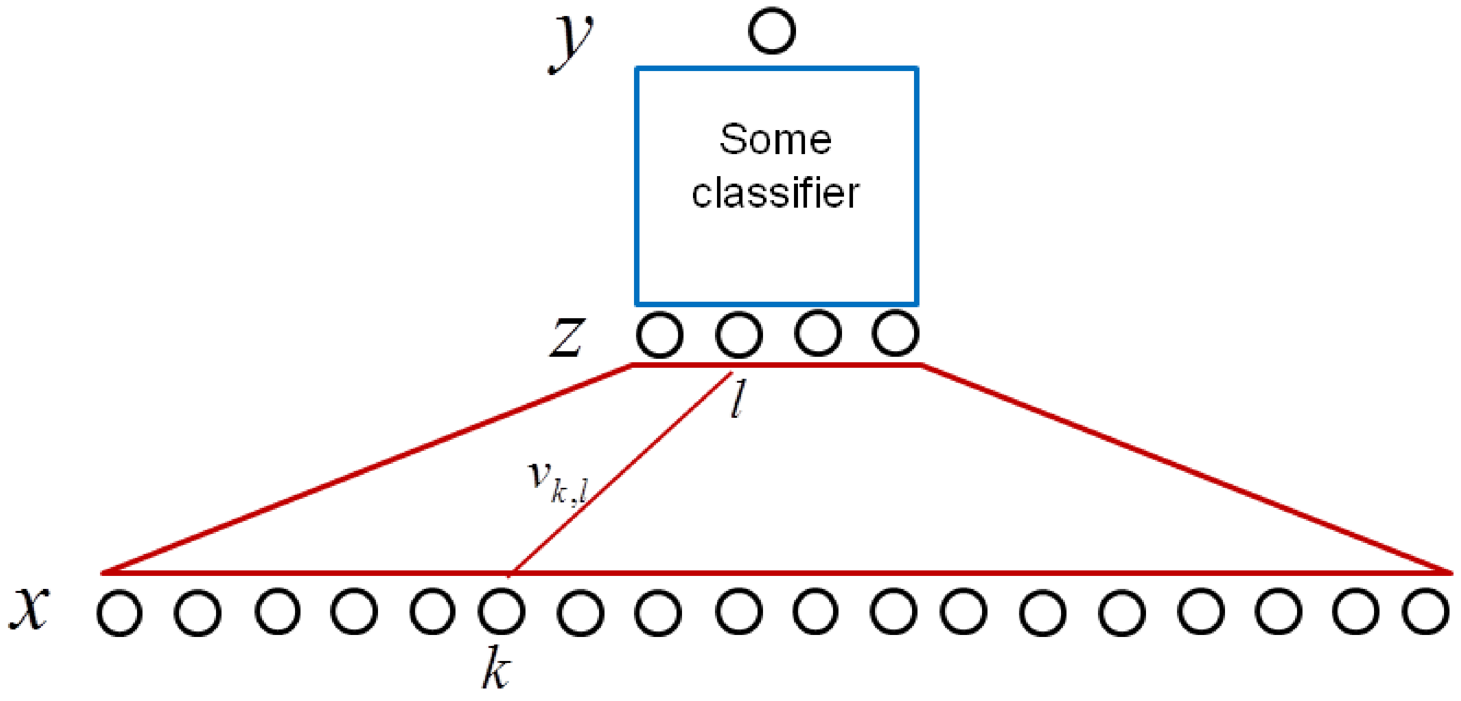
- So far we always argued for a high-dimensional feature space
- Still, in some cases it makes sense to first reduce the dimensionality before applying a learning algorithm
- Example: The data are pixels in an image

Dimensionality Reduction

- We want to compress the M-dimensional $\mathbf{x}$ to an $r-$dimensional $\mathbf{z}$ using a **linear** transformation
- We want $\mathbf{x}$ to be reconstructed from $\mathbf{z}$ as well as possible in the mean squared error sense for all data points $\mathbf{x}_i$

$$\sum_i (\mathbf{x}_i - V\mathbf{z}_i)^T(\mathbf{x}_i - V\mathbf{z}_i)$$

where V is an $M \times r$ matrix.



$$z_l = v_{1,l}x_1 + v_{2,l}x_2 + \ldots + v_{M,l}x_M$$

First Component

- Let's first look at $r = 1$ and we want to find the vector $\mathbf{v}$
- Without loss of generality, we assume that $\|\mathbf{v}\| = 1$
- The reconstruction error for a particular $\mathbf{x}_i$ is given by

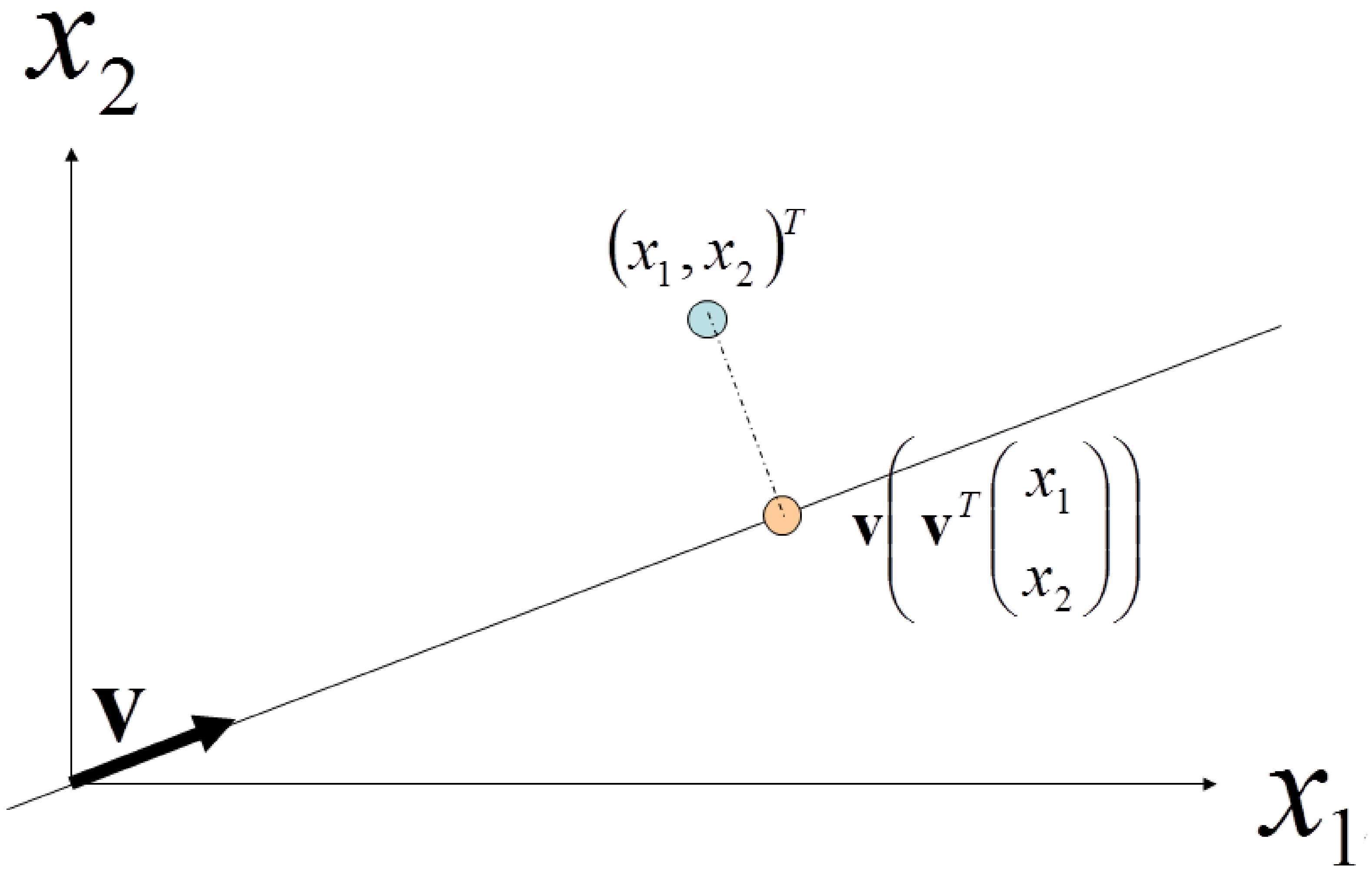
$$(\mathbf{x}_i - \hat{\mathbf{x}}_i)^T(\mathbf{x}_i - \hat{\mathbf{x}}_i) = (\mathbf{x}_i - \mathbf{v}z_i)^T(\mathbf{x}_i - \mathbf{v}z_i).$$

  The optimal z_i is then (see figure)

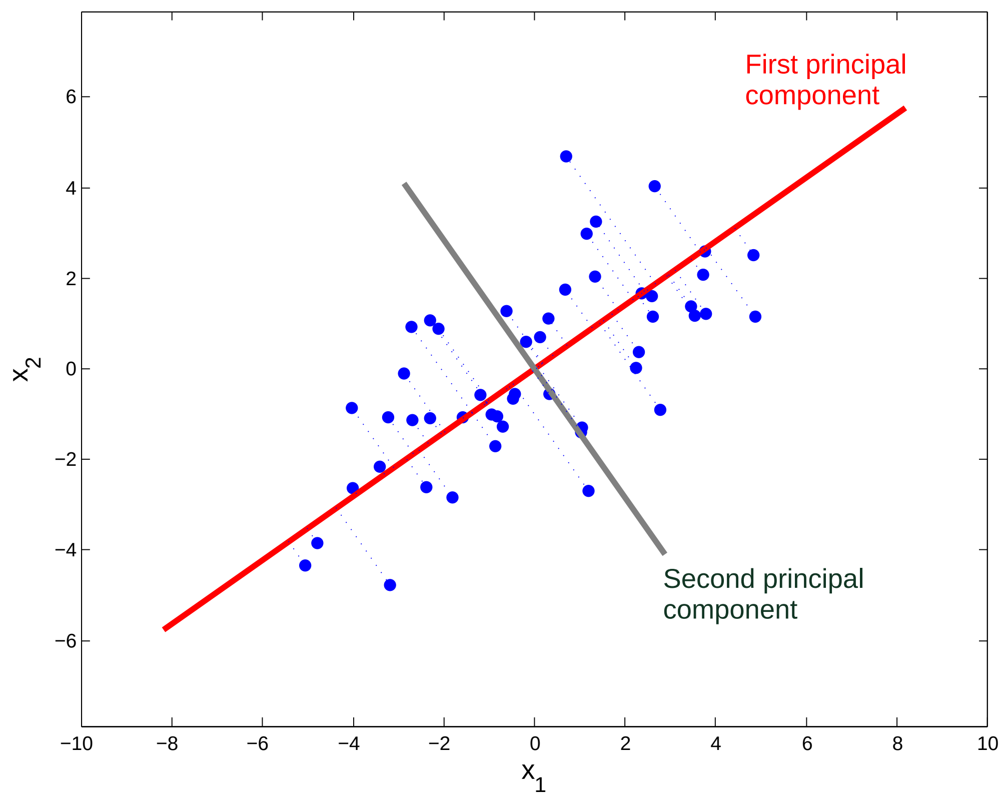
$$z_i = \mathbf{v}^T\mathbf{x}_i \qquad \text{(note: } z_i \text{ is scalar)}$$

  Thus we get

$$\hat{\mathbf{x}}_i = \mathbf{v}\mathbf{v}^T\mathbf{x}_i$$


x_2
$(x_1, x_2)^T$
$\mathbf{v}\left(\mathbf{v}^T\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right)$
$\mathbf{v}$
x_1

Example: 50 data points in 2d



- First principal vector is the direction that maximizes the variance of the projected data
- It is also the direction that minimizes the reconstruction error
- Here we centered the data (i.e. subtracted the sample mean) before applying PCA. More on that later

Computing the First Principal Vector

- So what is $\mathbf{v}$? We are looking for a $\mathbf{v}$ that minimizes the reconstruction error over all data points. We use the Lagrange parameter λ to guarantee length 1

$$L = \sum_{i=1}^{N} (\mathbf{v}\mathbf{v}^T\mathbf{x}_i - \mathbf{x}_i)^T(\mathbf{v}\mathbf{v}^T\mathbf{x}_i - \mathbf{x}_i) + \lambda(\mathbf{v}^T\mathbf{v} - 1)$$

$$= \sum_{i=1}^{N} \mathbf{x}_i^T\mathbf{v}\mathbf{v}^T\mathbf{v}\mathbf{v}^T\mathbf{x}_i + \mathbf{x}_i^T\mathbf{x}_i - \mathbf{x}_i^T\mathbf{v}\mathbf{v}^T\mathbf{x}_i - \mathbf{x}_i^T\mathbf{v}\mathbf{v}^T\mathbf{x}_i + \lambda(\mathbf{v}^T\mathbf{v} - 1)$$

$$= \sum_{i=1}^{N} \mathbf{x}_i^T\mathbf{x}_i - \mathbf{x}_i^T\mathbf{v}\mathbf{v}^T\mathbf{x}_i + \lambda(\mathbf{v}^T\mathbf{v} - 1)$$

Computing the First Principal Vector (cont'd)

- We take the derivative with respect to $\mathbf{v}$ and obtain

$$\frac{\partial}{\partial \mathbf{v}} \mathbf{x}_i^T \mathbf{v}\mathbf{v}^T \mathbf{x}_i$$

$$= \frac{\partial}{\partial \mathbf{v}} (\mathbf{v}^T \mathbf{x}_i)^T (\mathbf{v}^T \mathbf{x}_i) = 2 \left(\frac{\partial}{\partial \mathbf{v}} \mathbf{v}^T \mathbf{x}_i \right) (\mathbf{v}^T \mathbf{x}_i)$$

$$= 2\mathbf{x}_i (\mathbf{v}^T \mathbf{x}_i) = 2\mathbf{x}_i (\mathbf{x}_i^T \mathbf{v}) = 2(\mathbf{x}_i \mathbf{x}_i^T)\mathbf{v}$$

and

$$\frac{\partial}{\partial \mathbf{v}} \lambda \mathbf{v}^T \mathbf{v} = 2\lambda \mathbf{v}$$

- We set the derivative to zero and get

$$\sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T \mathbf{v} = \lambda \mathbf{v}$$

or in matrix form

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

where $\Sigma = X^T X$

- Recall that the Lagrangian is maximized with respect to λ
- Thus the first **principal vector** $\mathbf{v}$ is the first eigenvector of Σ (with the largest eigenvalue)
- $z_i = \mathbf{v}^T\mathbf{x}_i$ is called the first **principal component** of $\mathbf{x}_i$

Computing all Principal Vectors / Rank-r approx.

- The second principal vector is given by the second eigenvector of Σ and so on
- Note: the principal vectors are mutually orthogonal
- For a rank-r approximation we get

$$\mathbf{z}_i = V_r^T \mathbf{x}_i$$

and the optimal reconstruction is

$$\hat{\mathbf{x}}_i = V_r \mathbf{z}_i$$

PCA Applications

Classification and Regression

- First perform a PCA of X and then use as input to the model $\mathbf{z}_i$ instead of $\mathbf{x}_i$, where

$$\mathbf{z}_i = V_r^T \mathbf{x}_i$$

- Example: Linear regression on $\mathbf{z}_i$ is called PCR (Principal Component Regression)

Similarity and Novelty

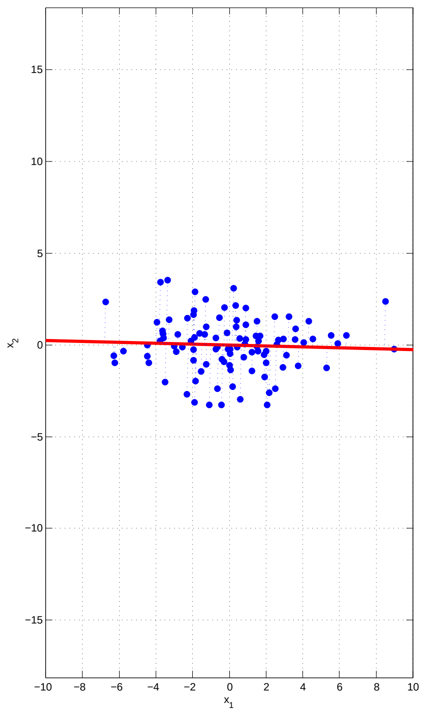
- A distance measure (Euclidian distance) based on the principal components is often more meaningful than a distance measure calculated in the original space
- Novelty detection / outlier detection: We calculate the reconstruction of a new vector $\mathbf{x}$ and calculate

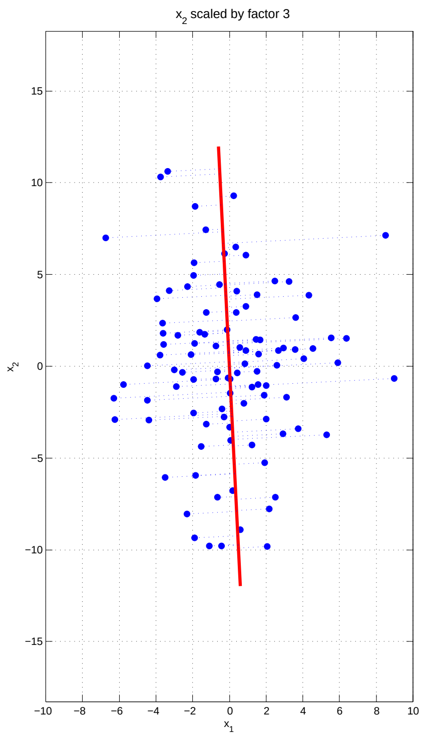
$$\|\mathbf{x} - V_r V_r^T \mathbf{x}\| = \|V_{-r}^T \mathbf{x}\|$$

  If this distance is large, then the new input is unusual, i.e. might be an outlier
- Here V_{-r} contains the $M - r$ eigenvectors $\mathbf{v}_{r+1}, ..., \mathbf{v}_M$ of Σ.

Practical Hint 1

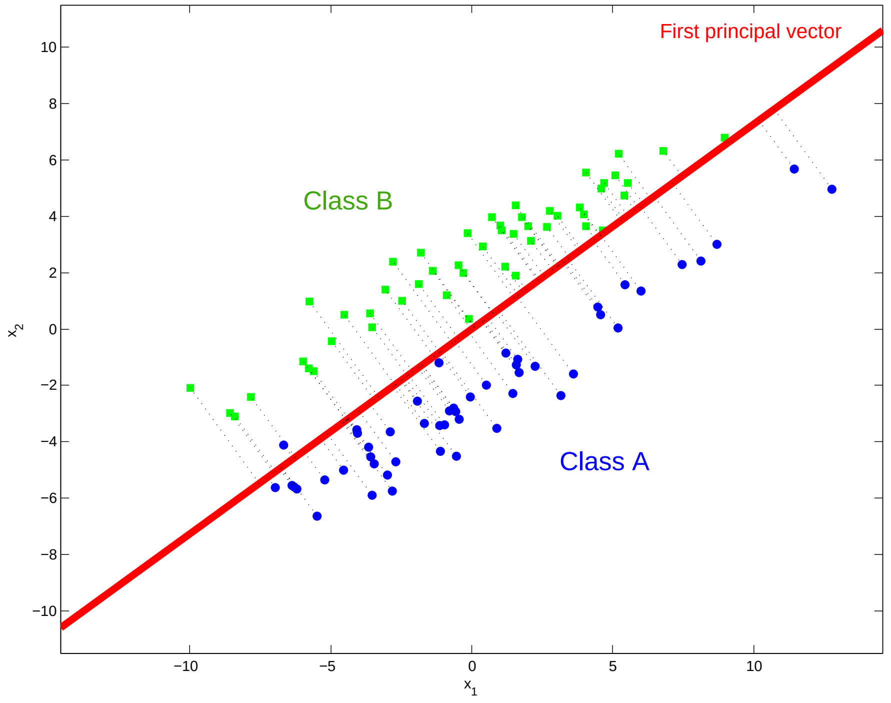
- Scaling of inputs affects the resulting principal components
- If the given scaling is arbitrary (e.g. different physical units: mbar, PSI, K), consider **standardizing** data before performing PCA (e.g. $x = x/\text{std}(x)$)
- Example shows first principal vector. Right side: x_2 scaled by factor 3





Practical Hint 2

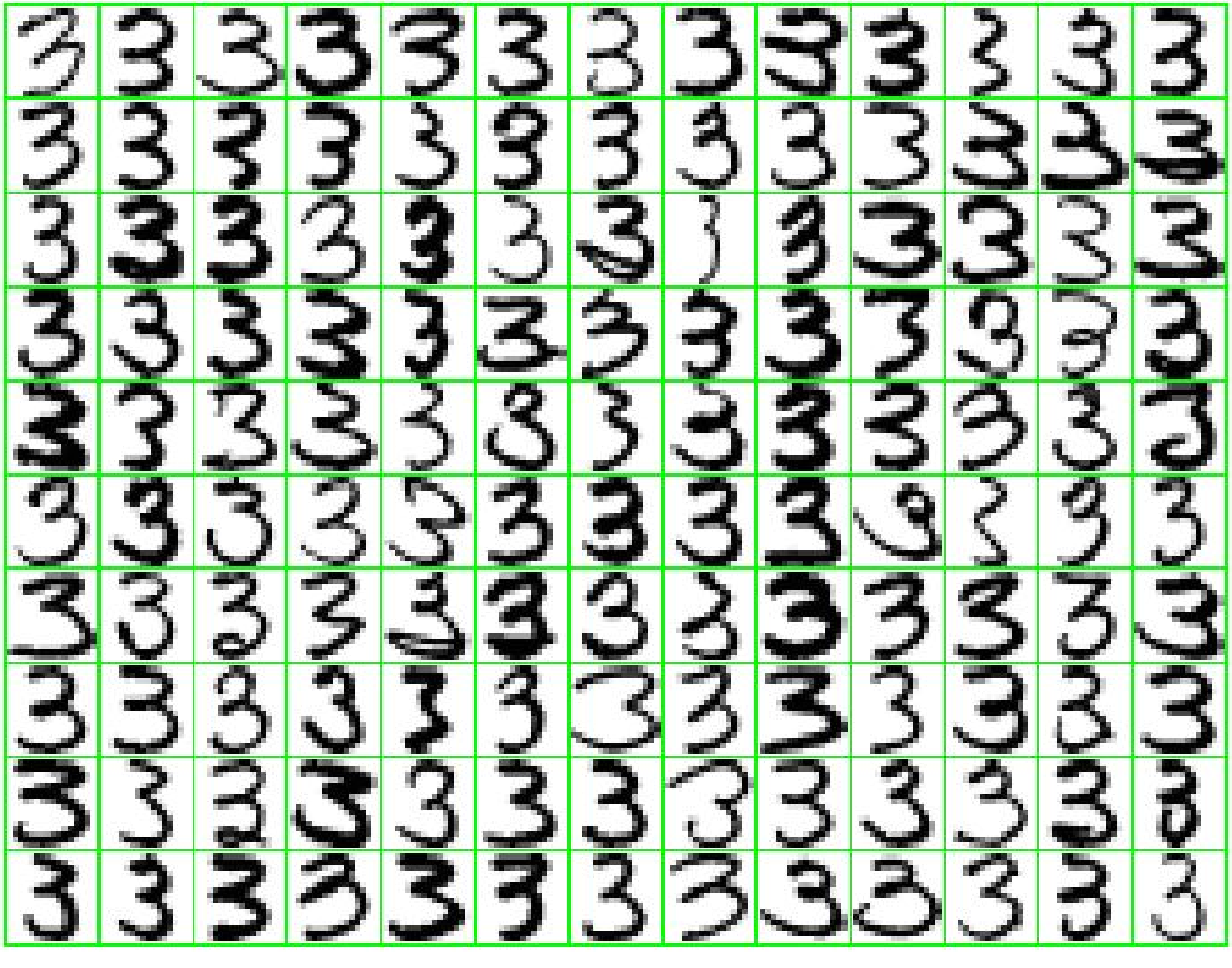
- PCA does not consider class information (i.e. it is an **unsupervised** method)
- Example: Projecting the shown data onto the first principal vector removes the class separability completely!
- There are methods that include class information, e.g. Partial Least Squares, or combinations of PCA with supervised methods, e.g. "Supervised Principal Components" [Hastie et al., ESLII, p.674]



PCA Example: Handwritten Digits

Data Set

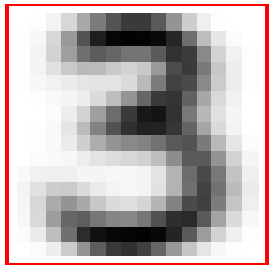
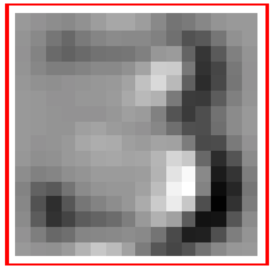
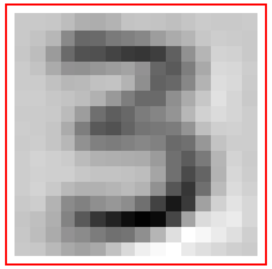
- 130 handwritten digits “ 3 ” (in total: 658): significant difference in style
- The images have 16×16 grey valued pixels. Each input vector $\mathbf{x}$ consists of the 256 grey values of the pixels. Applying a linear classifier to the original pixels gives bad results



Visualisation

- We see the first two principal vectors $\mathbf{v}_1$, $\mathbf{v}_2$
- $\mathbf{v}_1$ prolongs the lower portion of the “3”
- $\mathbf{v}_2$ modulates thickness

$\hat{\mathbf{x}}_i =$

 $+\ z_{i,1} \cdot$  $+\ z_{i,2} \cdot$  .

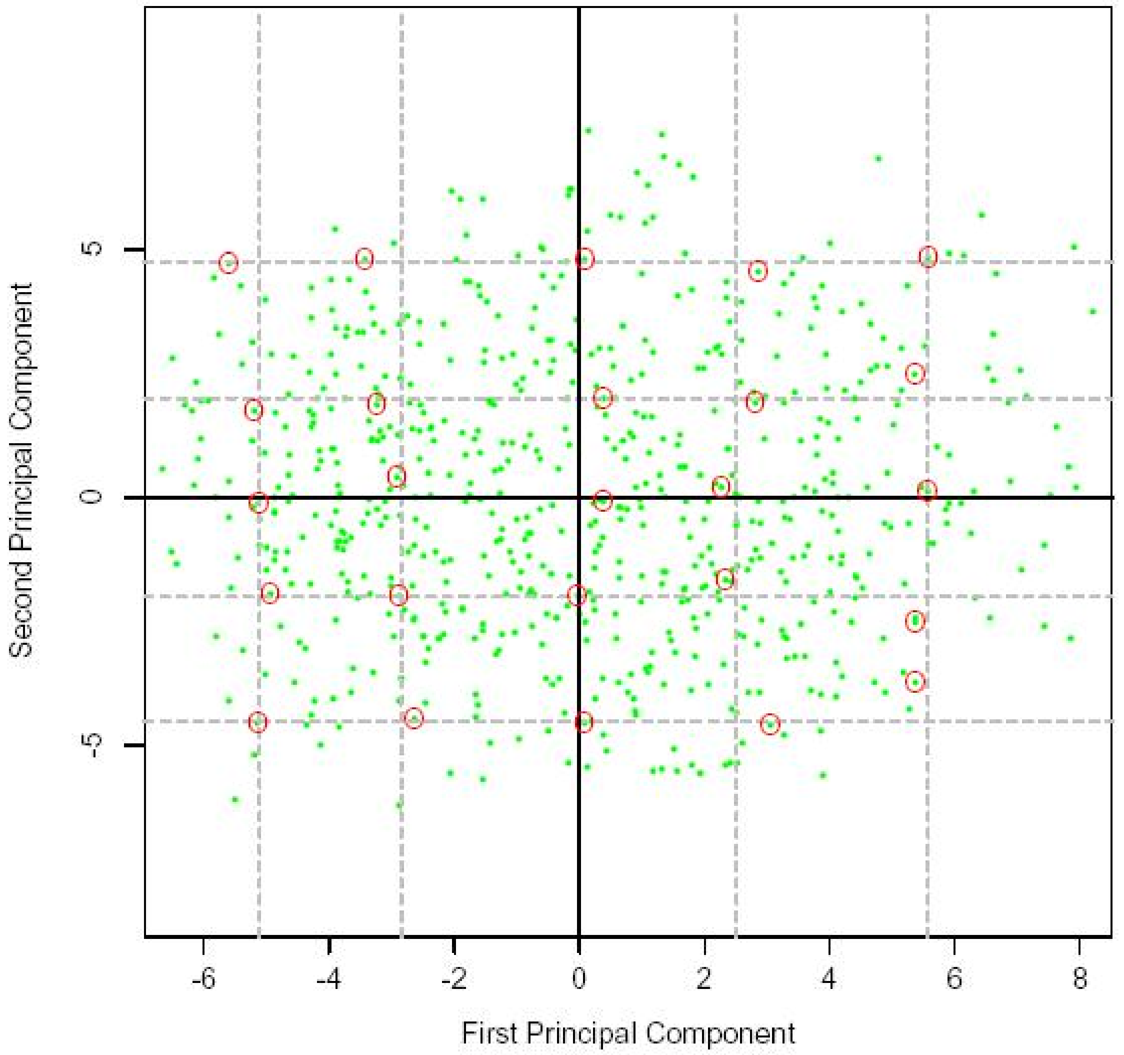
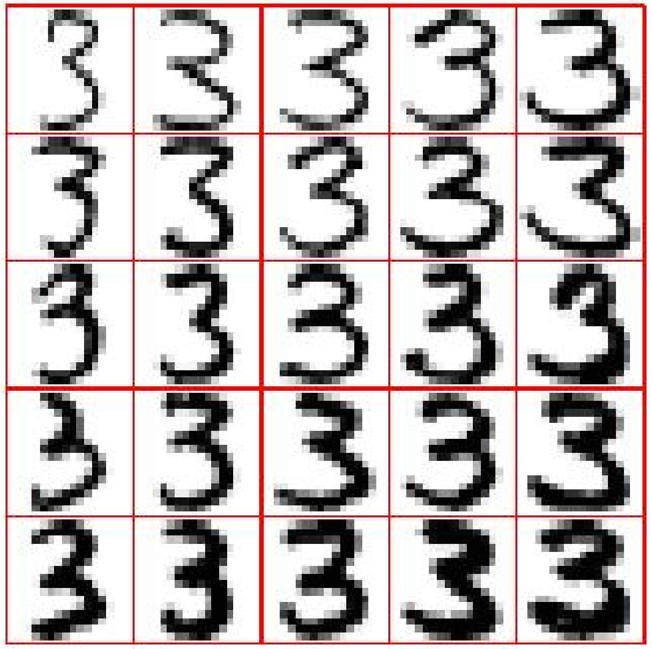
$\mathbf{m}$ $\qquad\qquad$ $\mathbf{v}_1$ $\qquad\qquad$ $\mathbf{v}_2$

Visualisation: Reconstruction

- For different values of the principal components z_1 and z_2 the reconstructed image is shown

$$\hat{\mathbf{x}} = \mathbf{m} + z_1\mathbf{v}_1 + z_2\mathbf{v}_2$$

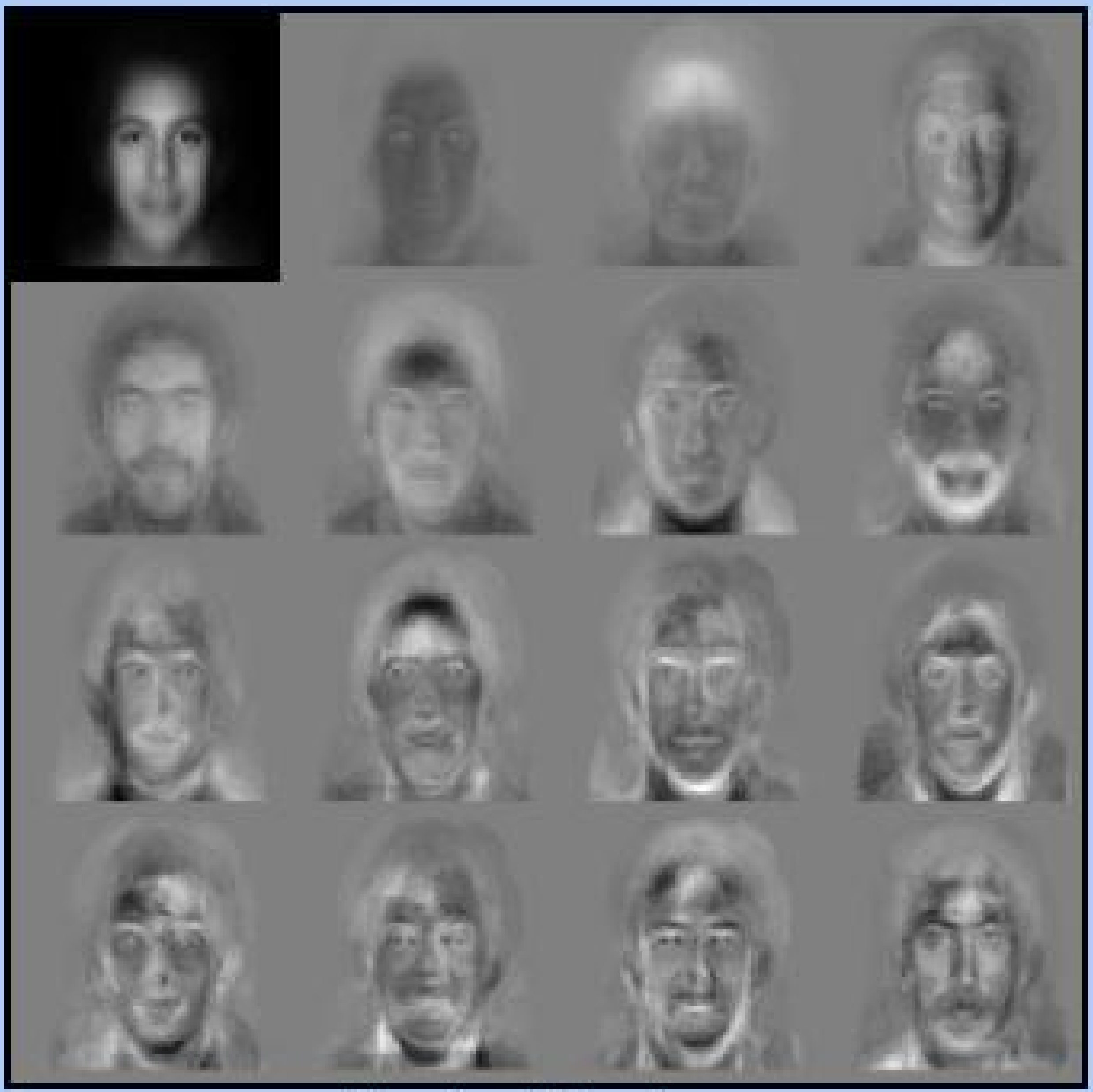
- $\mathbf{m}$ is a mean vector that was subtracted before the PCA was performed and is now added again. $\mathbf{m}$ represents 256 mean pixel values averaged over all samples


Second Principal Component
First Principal Component
5
0
-5
-6
-4
-2
0
2
4
6
8


Eigenfaces

Data Set

- PCA for face recognition
- http://vismod.media.mit.edu/vismod/demos/facerec/basic.html
- 7562 images from 3000 persons
- $\mathbf{x}_i$ contains the pixel values of the i-th image. Obviously it does not make sense to build a classifier directly on the $256 \times 256 = 65536$ pixel values
- Eigenfaces were calculated based on 128 images (eigenfaces might sound cooler than principal vectors!) (training set)
- For recognition on test images, the first $r = 20$ principal components are used
- Almost each person had at least 2 images; many persons had images with varying facial expression, different hair style, different beards, ...

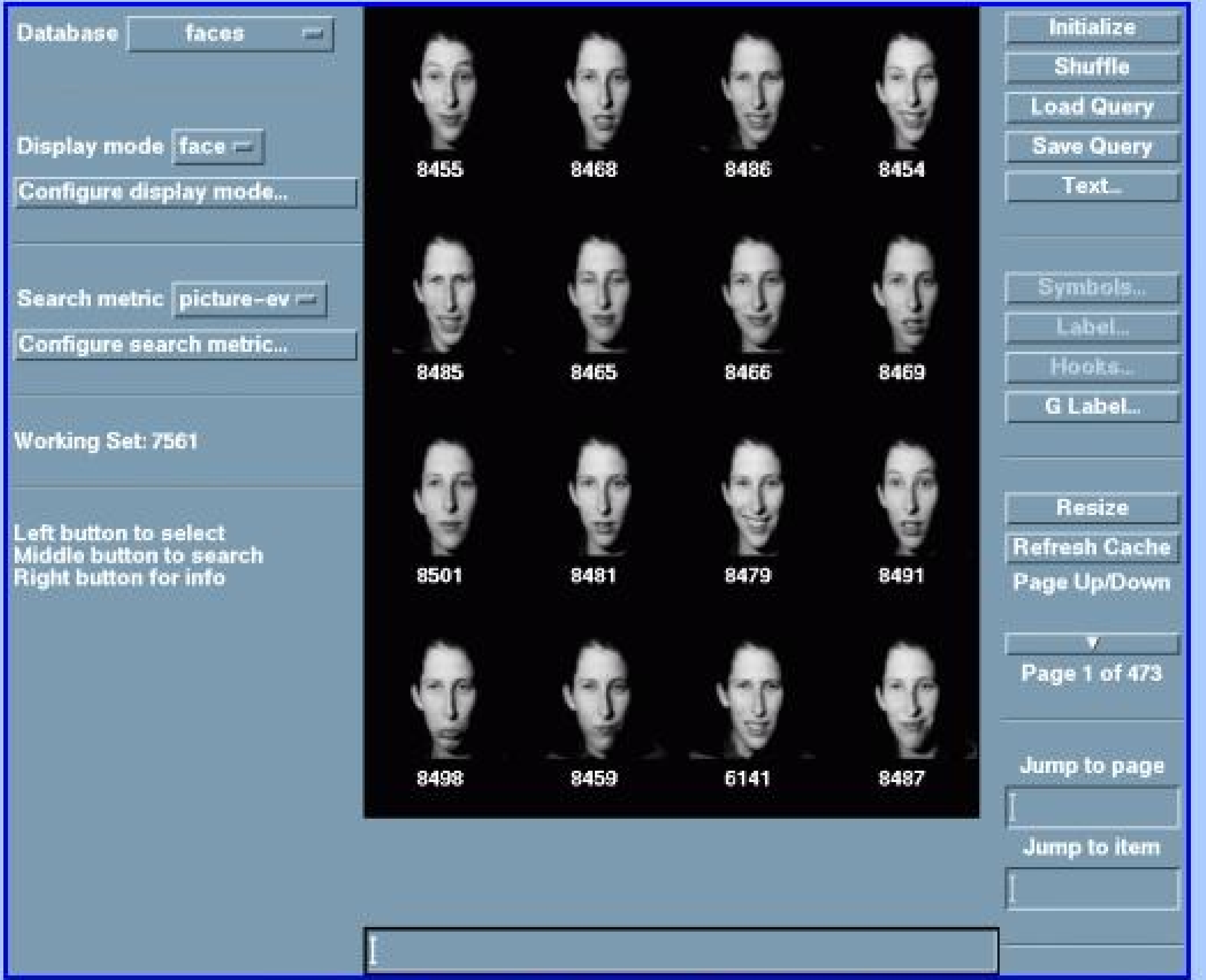


Standard Eigenfaces

Similarity Search based on Principal Components

- The upper left image is the test image. Based on the Euclidian distance in PCA-space the other 15 images were classified as nearest neighbors. All 15 images came from the correct person, although the data base contained more than 7562 images!
- Thus, distance is evaluated following

$$||\mathbf{z} - \mathbf{z}_i||$$


Database: faces

Display mode: face

Search metric: picture-ev

Working Set: 7561

Left button to select
Middle button to search
Right button for info

Page 1 of 473

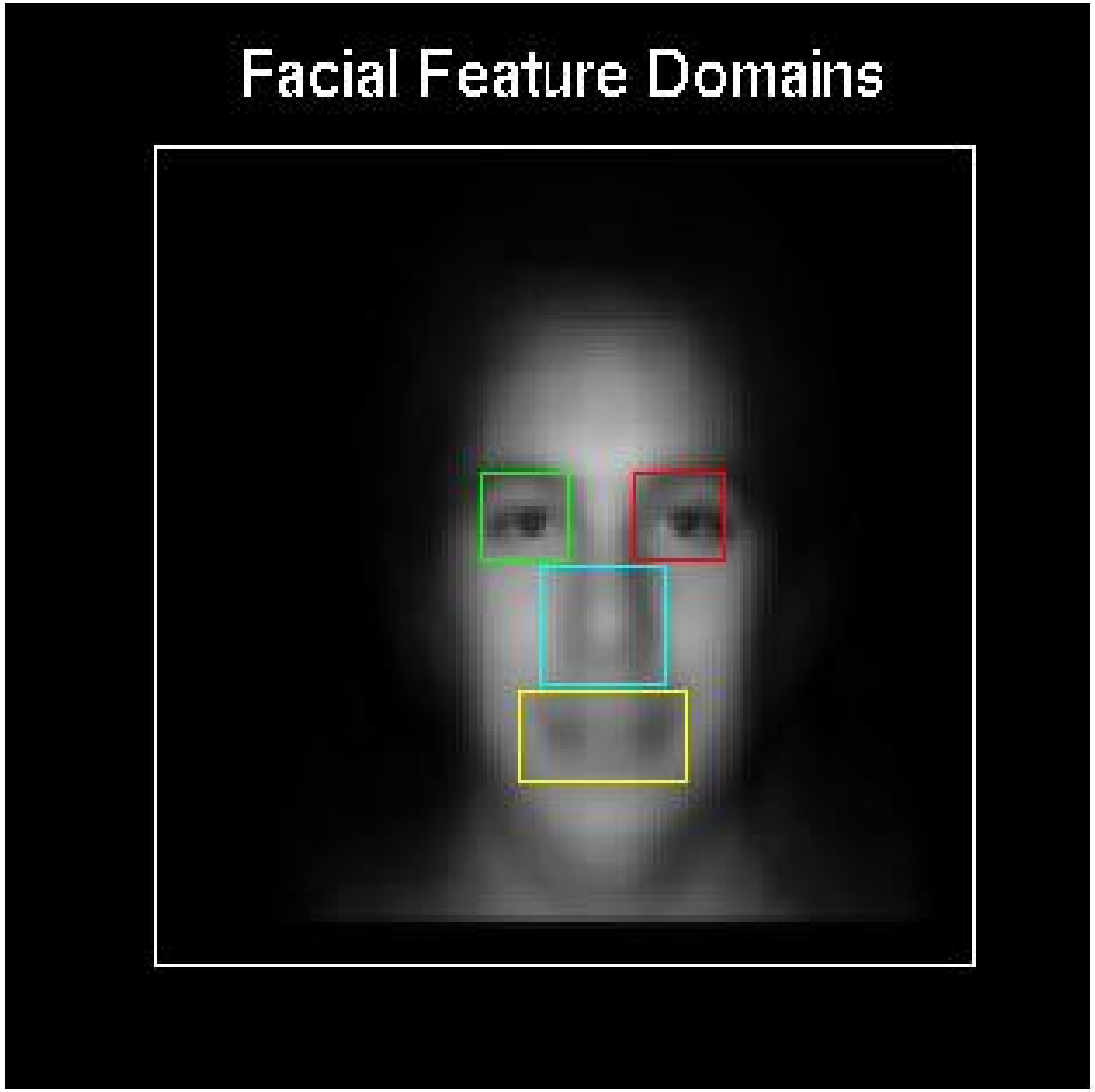
MIT Media Lab Database Photobook

Recognition Rate

- 200 pictures were selected randomly from the test set. In 96% of all cases the nearest neighbor was the correct person

Modular Eigenspaces

- The method can also be applied to facial features as eigeneyes, eigennoses, eigenmouths.
- Analysis of human eye movements also showed that humans concentrate on these local features as well


Facial Feature Domains

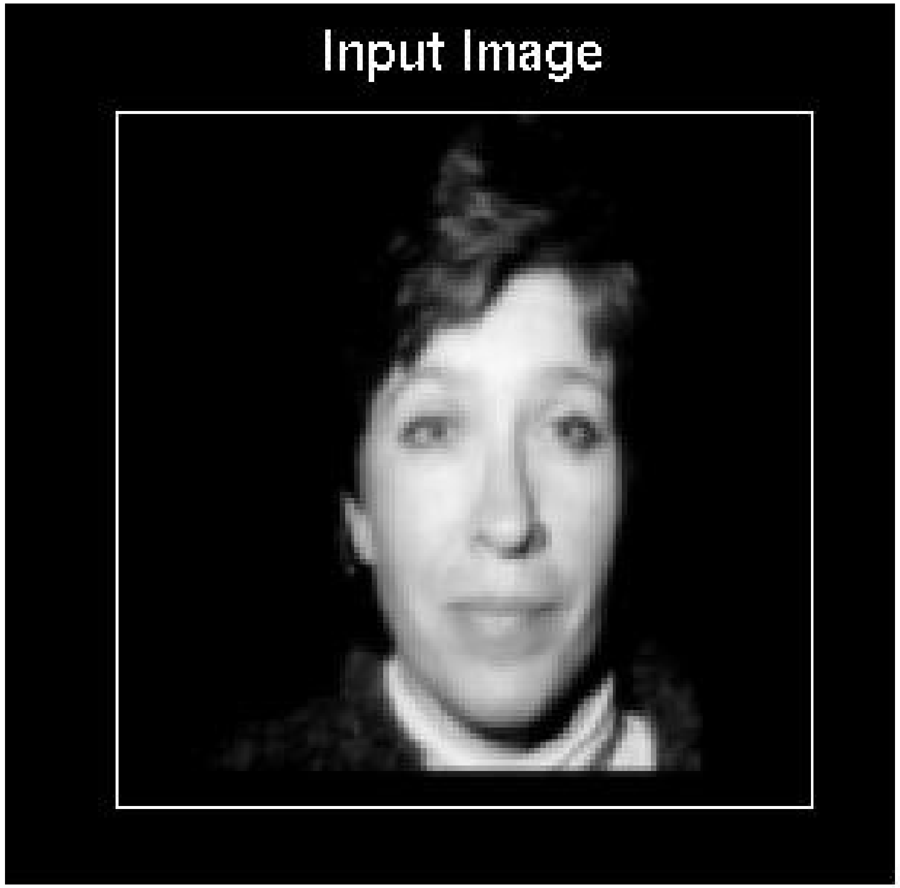
Automatically Finding the Facial Features

- The modular methods require an automatic way of finding the facial features (eyes, nose, mouth)
- One defines rectangular windows that are indexed by the central pixel in the window
- One computes the anomaly of the image window for all locations, where the detector was trained on a feature class (e.g., left eye) by using a rank 10 PCA. When the anomaly is minimum the feature (eye) is detected.

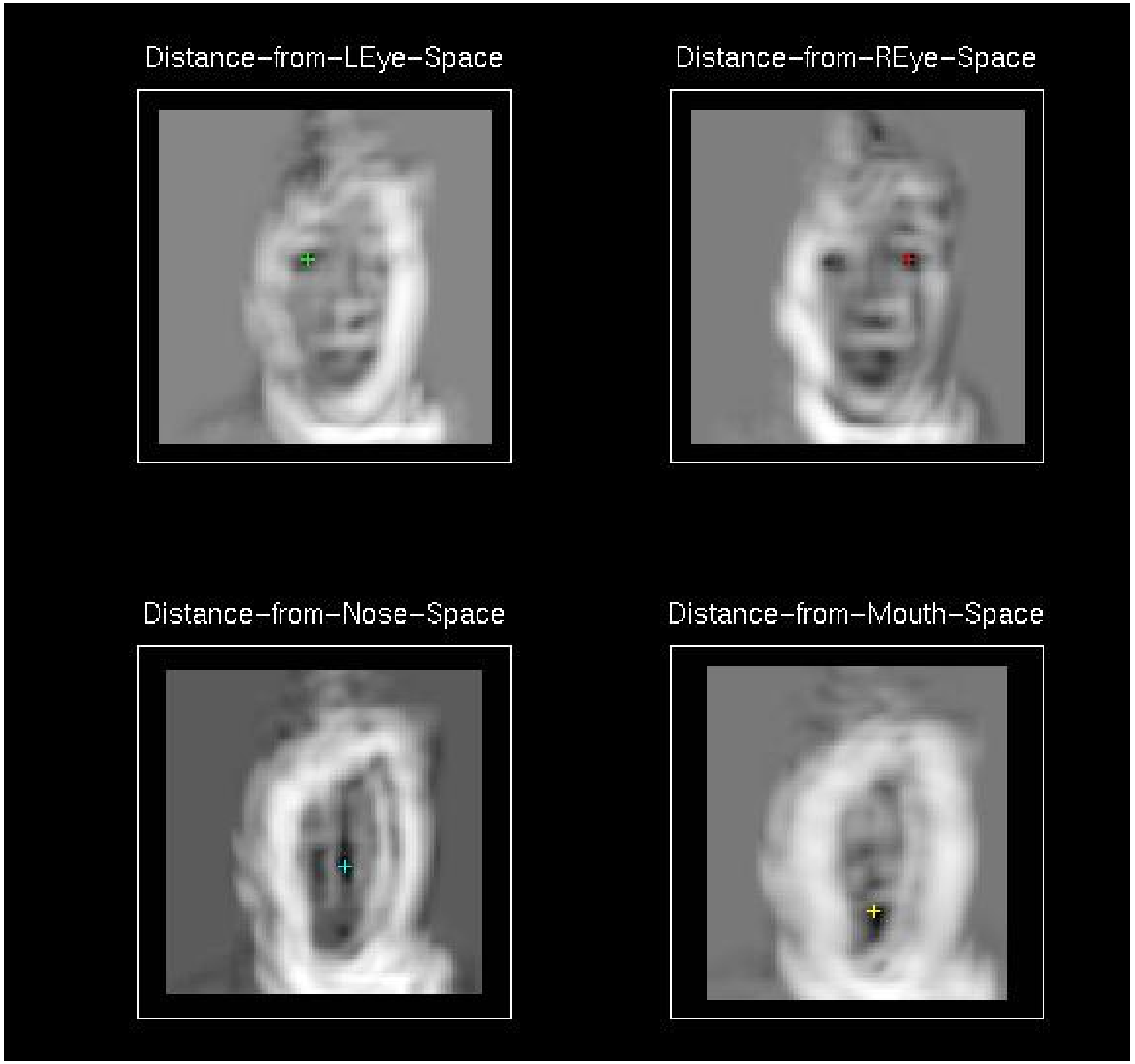
$$\mathrm{AN}_{\text{left eye}}(\mathbf{x}_{pos_k}) = \|\mathbf{x}_{pos_k} - V_r V_r^T \mathbf{x}_{pos_k}\|^2$$

- In the following images, brightness is anomaly

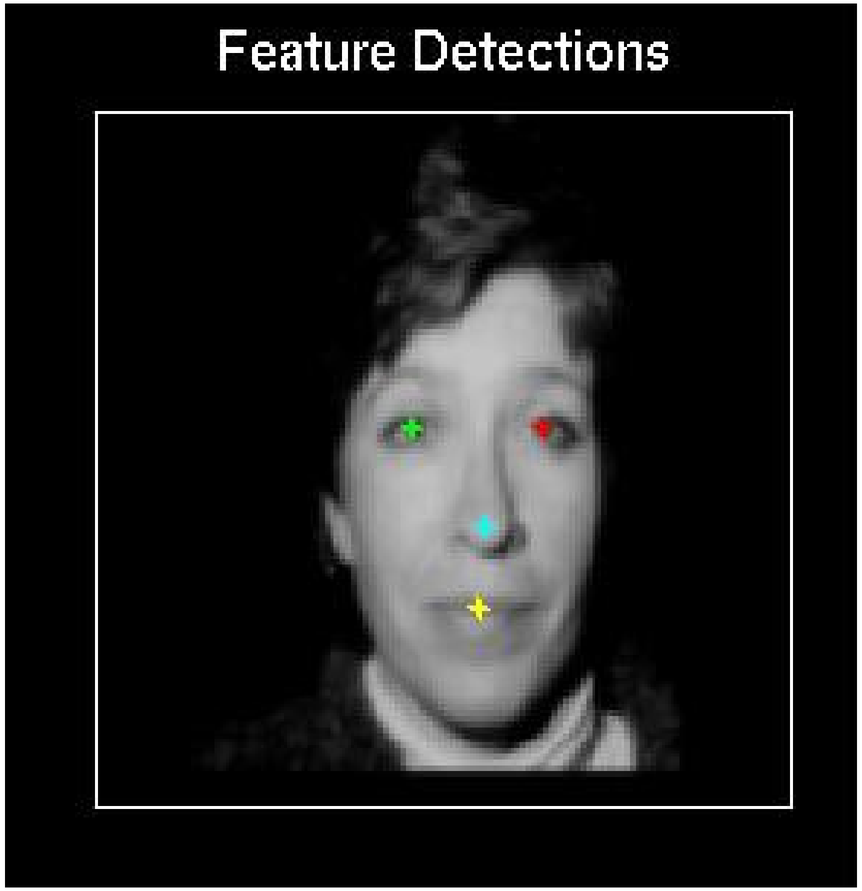
Input Image



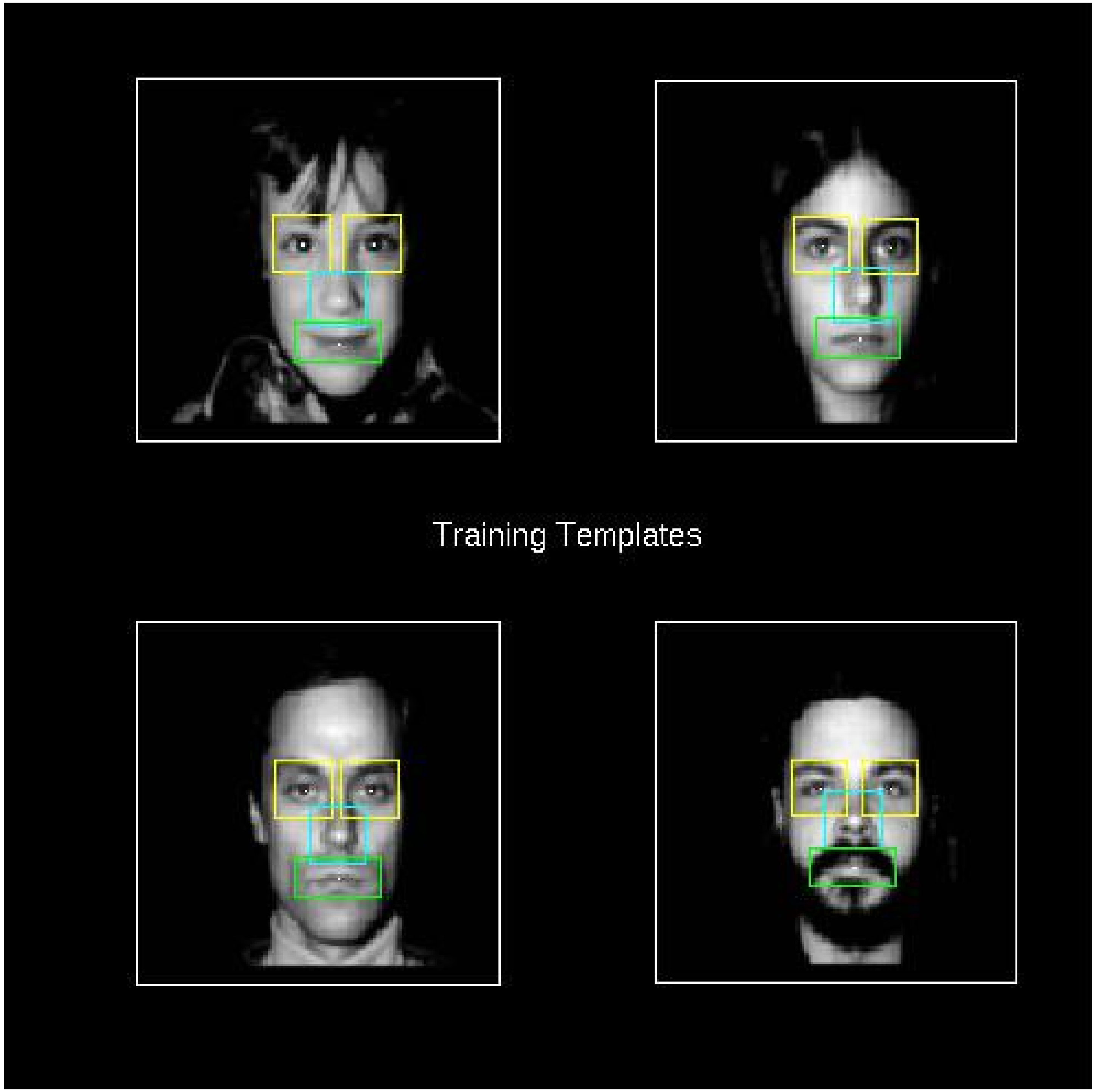
Distances


Distance–from–LEye–Space
Distance–from–REye–Space
Distance–from–Nose–Space
Distance–from–Mouth–Space

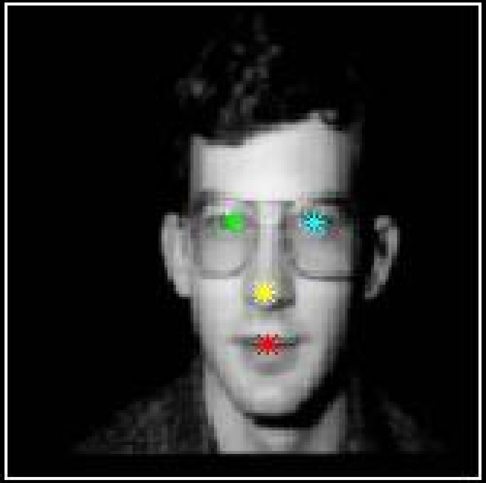
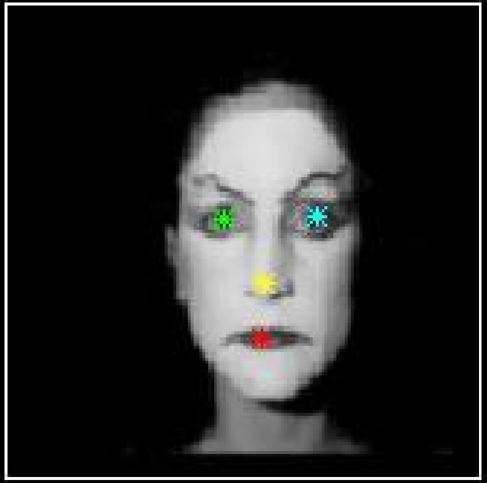
Detection



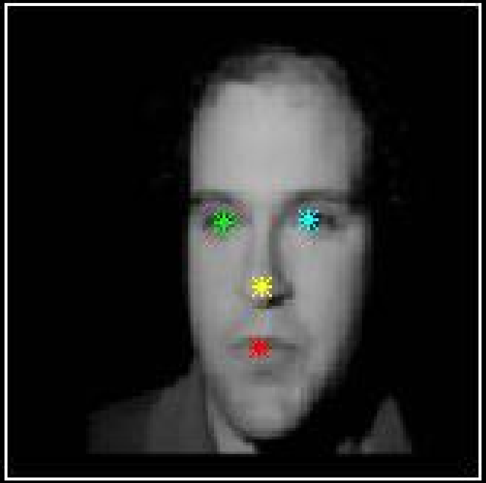
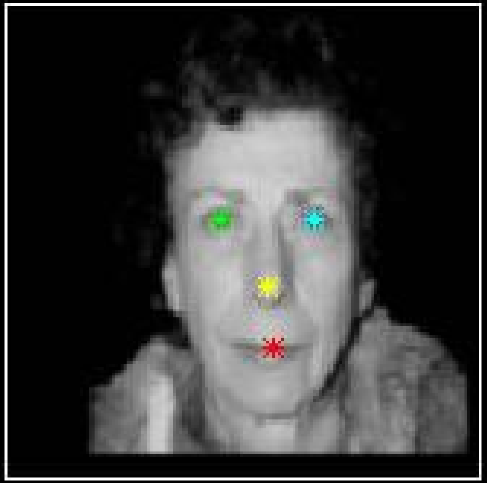
Training Templates


Training Templates

Typical Detections

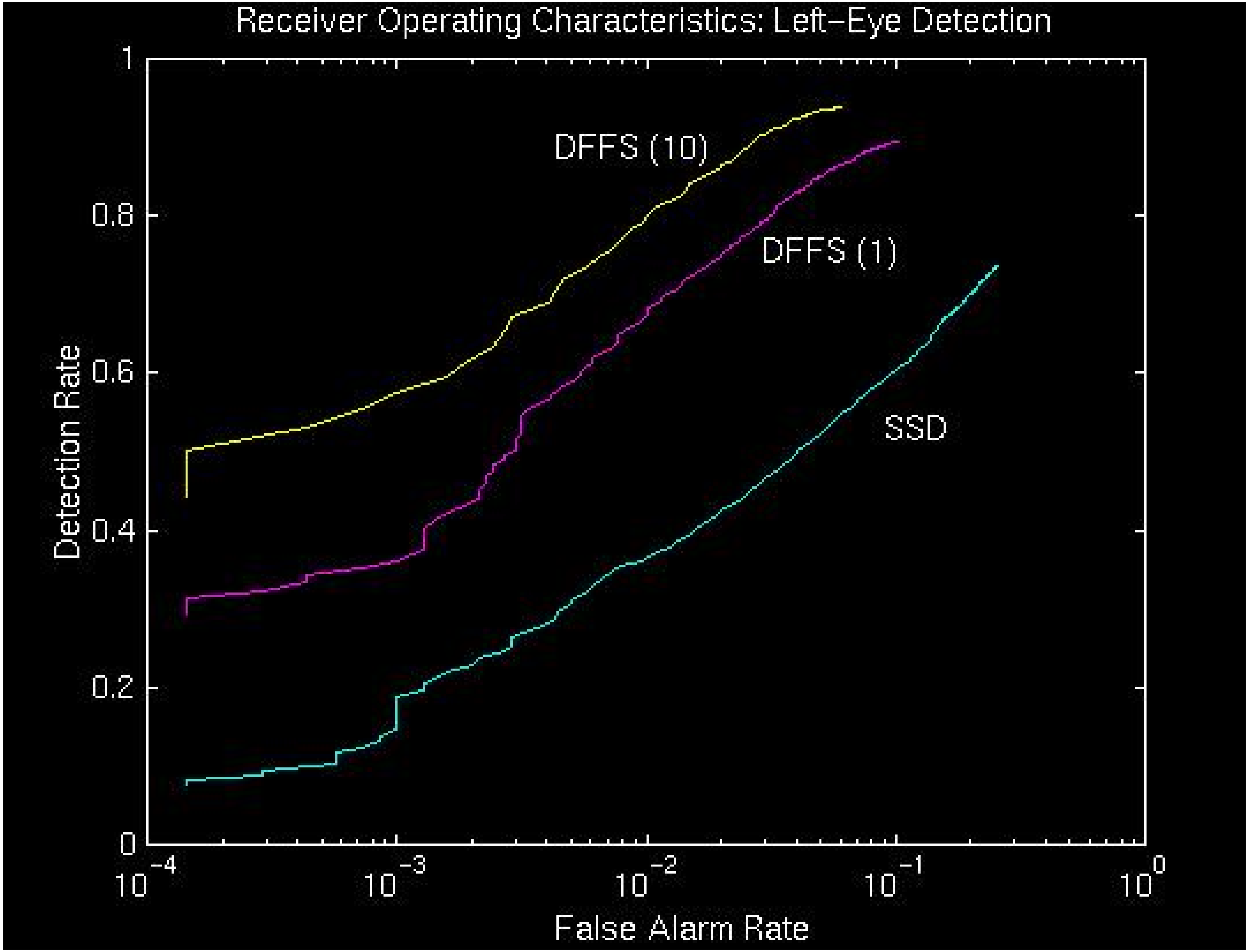



Typical Detections

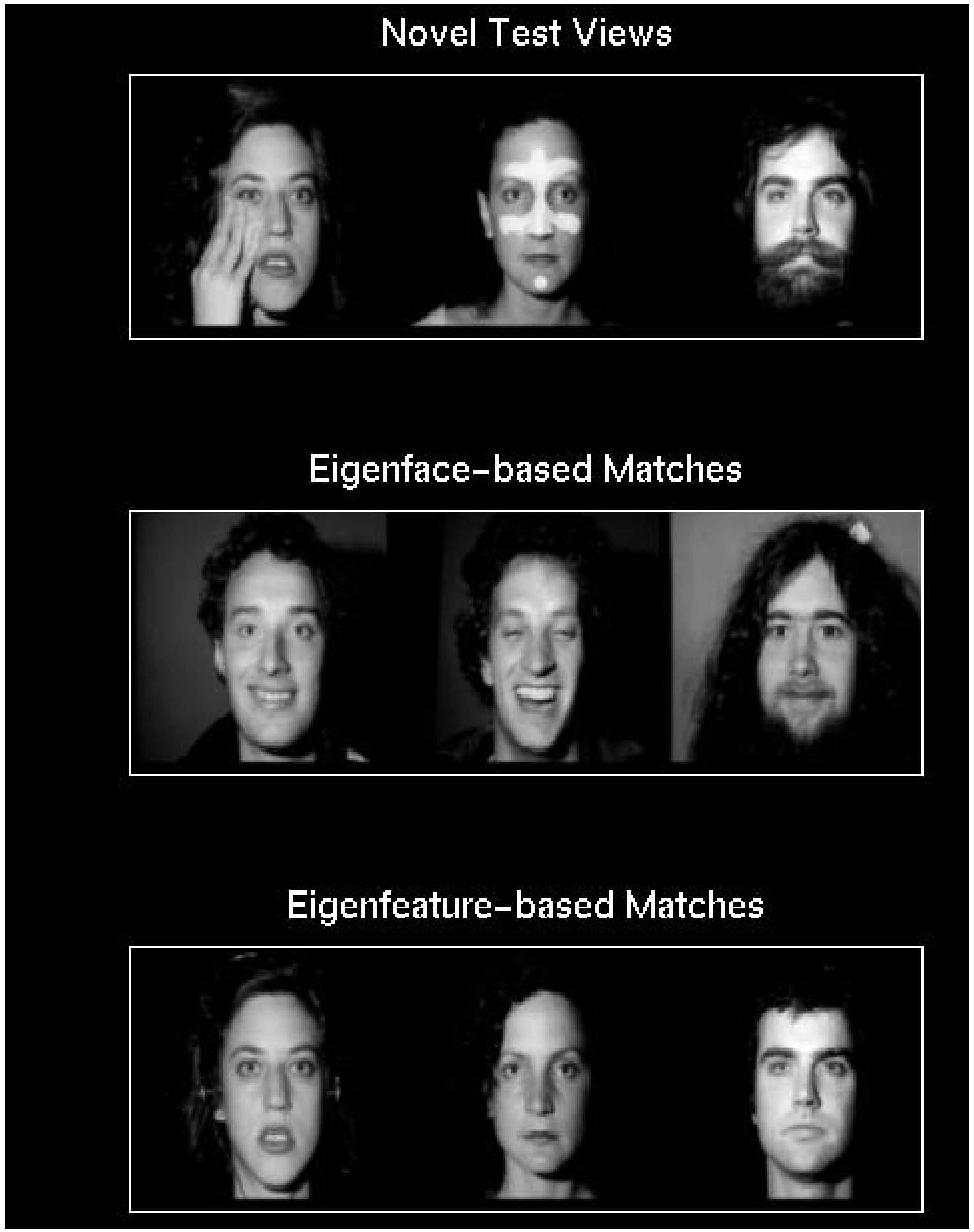
Detection Rate

- The next plot shows the performance of the left-eye-detector based on $\mathrm{AN}_{\text{left eye}}(\mathbf{z}_{pos_k})$ (labeled as DFFS) with rank one and with rank 10. Also shown are the results for simple template matching (distance to the mean left eye image (SSD)).
- **Definition of Detection:** The global optimum is below a threshold value α and is within 5 pixels of the correct location
- **Definition of False Alarm:** The global optimum is below a threshold value α and is outside of 5 pixels of the correct location
- In the curves, α is varied. DFFS(10) reaches a correct detection of 94% at a false alarm rate of 6%. this means that in 94% of all cases, where a left eye has been detected in the image, it was detected at the right location and in 6% of all cases, where a left eye has been detected in the image, it was detected at the wrong location


Receiver Operating Characteristics: Left-Eye Detection
DFFS (10)
DFFS (1)
SSD
Detection Rate
False Alarm Rate
0
0.2
0.4
0.6
0.8
1
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}

Robustness

- A potential advantage of the eigenfeature layer is the ability to overcome the shortcomings of the standard eigenface method. A pure eigenface recognition system can be fooled by gross variations in the input image (hats, beards, etc.).
- The first row of the figure above shows additional testing views of 3 individuals in the above dataset of 45. These test images are indicative of the type of variations which can lead to false matches: a hand near the face, a painted face, and a beard.
- The second row in the figure above shows the nearest matches found based on a standard eigenface classification. Neither of the 3 matches correspond to the correct individual.
- On the other hand, the third row shows the nearest matches based on the eyes and nose features, and results in correct identification in each case. This simple example illustrates the advantage of a modular representation in disambiguating false eigenface matches.


Novel Test Views
Eigenface-based Matches
Eigenfeature-based Matches

PCA with Centered Data

- Often the mean is subtracted first

$$\tilde{x}_{i,j} = x_{i,j} - m_j$$

  where

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

- $\tilde{X}$ now contains the centered data as (N, M) matrix
- Note: $\tilde{X}^T \tilde{X} / N$ is the empirical covariance matrix
- Centering is particularly recommended when data are approximately Gaussian distributed

PCA with Centered Data (cont'd)

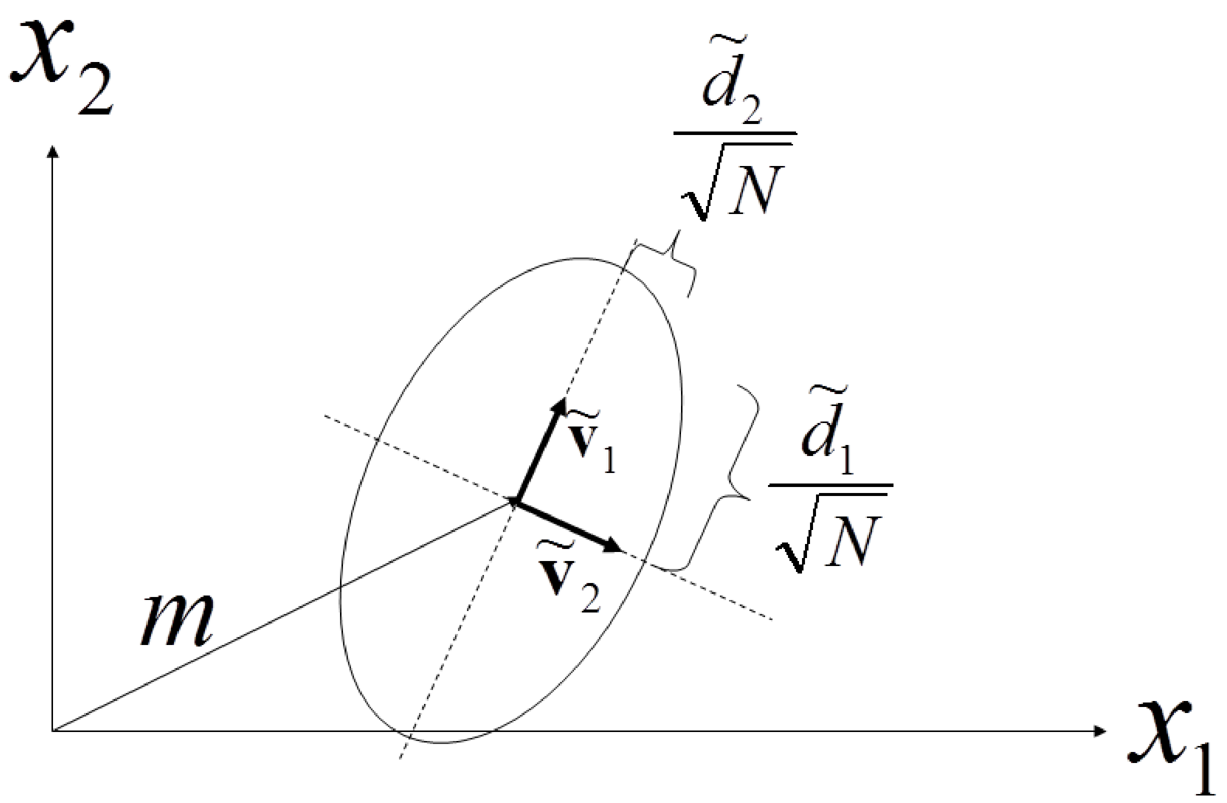
- Let $\tilde{v}_l, \; l = 1, \dots, r$, be the first r principal vectors.

  Then

$$\hat{\mathbf{x}}_i = \mathbf{m} + \sum_{l=1}^{r} \tilde{\mathbf{v}}_l \tilde{z}_{i,l}$$

  with $m = (m_1, \dots, m_M)^T$

$$\tilde{z}_{i,l} = \tilde{\mathbf{v}}_l^T \tilde{\mathbf{x}}_i$$



x_2
$\frac{\tilde{d}_2}{\sqrt{N}}$
$\tilde{\mathbf{v}}_1$
$\frac{\tilde{d}_1}{\sqrt{N}}$
$\tilde{\mathbf{v}}_2$
m
x_1

($\tilde{d}_l/\sqrt{N}$ is the standard deviation of the projected samples along the l-th principal direction. More on that later)

PCA and Singular Value Decomposition

Singular Value Decomposition (SVD)

- Any $N \times M$ matrix X can be factored as

$$X = UDV^T$$

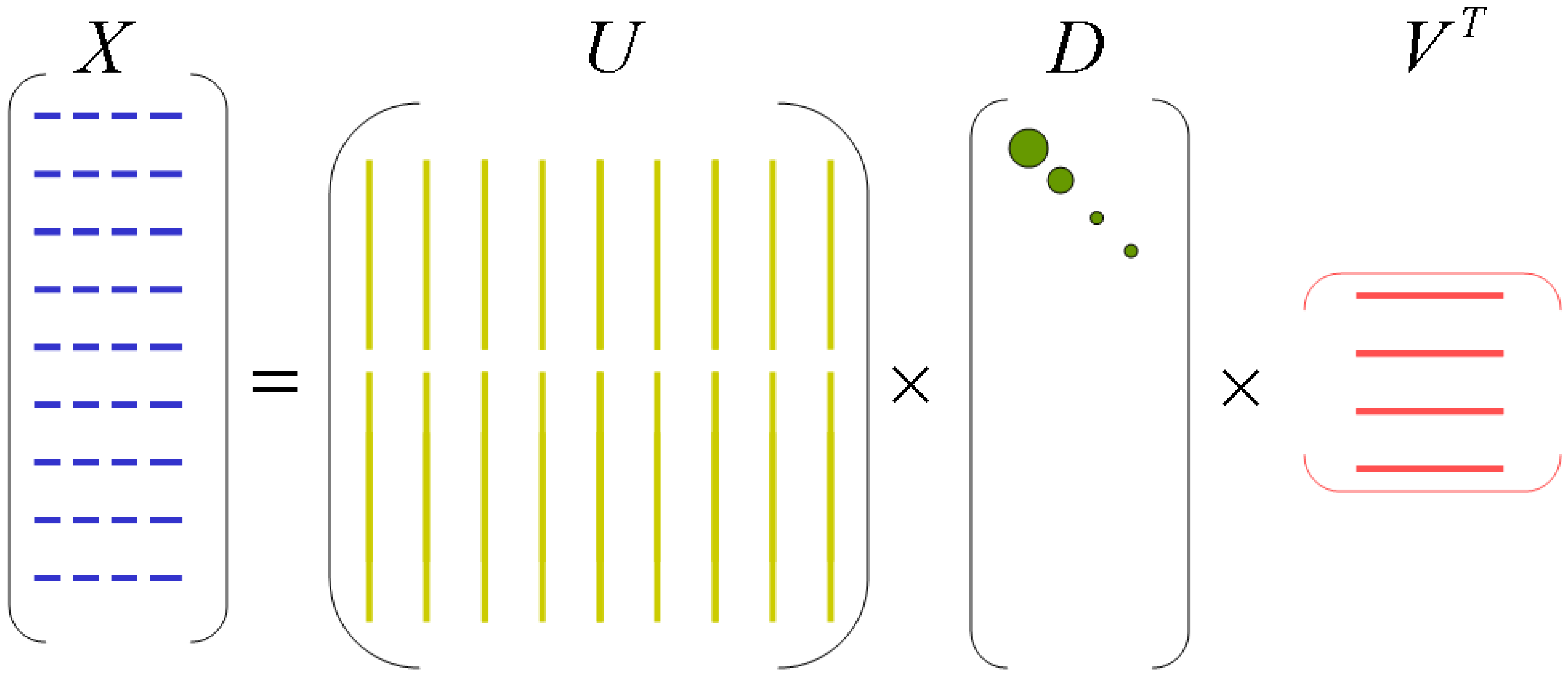
where U and V are both **orthogonal** matrices (i.e. their rows and columns are pairwise orthonormal). U is an $N \times N$ matrix and V is an $M \times M$ matrix.

- D is an $N \times M$ **diagonal matrix** with diagonal entries (singular values) $d_i \geq 0, i = 1, \ldots, \tilde{r}$, with $\tilde{r} = \min(M, N)$

- The $\mathbf{u}_j$ (columns of U) are the left singular vectors

- The $\mathbf{v}_j$ (columns of V) are the right singular vectors

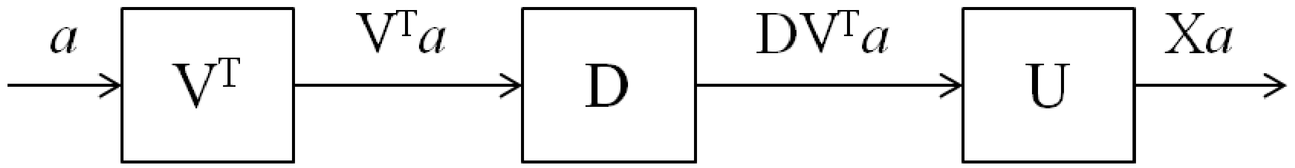
- The d_j are the singular values


X
=
U
×
D
×
V^T

Interpretation of SVD

$$X = UDV^T$$

Linear mapping Xa for all vectors $a \in \mathbb{R}^M$ can be decomposed


$$\begin{tikzcd} {} \arrow[r, "a"] & \boxed{V^T} \arrow[r, "V^T a"] & \boxed{D} \arrow[r, "DV^T a"] & \boxed{U} \arrow[r, "Xa"] & {} \end{tikzcd}$$

- rotate (by V^T)
- scale axes by d_i ($d_i = 0$ for $i > \tilde{r}$)
- rotate (by U)

Covariance Matrix and Kernel Matrix

- With **centered** X we get for the $(M \times M)$ empirical covariance matrix

$$\Sigma = \frac{1}{N} X^T X = \frac{1}{N} V D^T U^T U D V^T = \frac{1}{N} V D^T D V^T = \frac{1}{N} V D_V V^T$$

- And for the $(N \times N)$ empirical kernel matrix

$$K = \frac{1}{M} X X^T = \frac{1}{M} U D V^T V D^T U^T = \frac{1}{M} U D D^T U^T = \frac{1}{M} U D_U U^T$$

- With

$$\Sigma V = \frac{1}{N} V D_V \quad K U = \frac{1}{M} U D_U$$

one sees that the **columns of V are the eigenvectors of Σ** and the columns of U are the eigenvectors of K. The **eigenvalues** are the diagonal entries of D_V (same as in D_U).

- Apparent by now: The columns of V are the principal vectors!
- Thus, the j-th principal component is given by $\mathbf{z}_j = X\mathbf{v}_j$ where $\mathbf{v}_j$ denotes the j-th column of V. Its variance is

$$\mathsf{Var}(\mathbf{z}_j) = \mathsf{Var}(X\mathbf{v}_j) = d_j^2/N$$

- We have seen that PCA can be performed by SVD on the centered X. As there are good, numerically stable algorithms for calculating SVD, many implementations of PCA (e.g. `pca()` in Matlab) internally use SVD.

More Expressions (used on next slide)

- The SVD is

$$X = UDV^T$$

from which we get

$$X = UU^TX$$

$$X = XVV^T$$

Reduced Rank

- In the SVD, the d_i are ordered: $d_1 \geq d_2 \geq d_3 ... \geq d_{\tilde{r}}$. In many cases one can neglect $d_i, i > r$ and one obtains a rank-r approximation. Let D_r be a diagonal matrix with the corresponding entries. Then we get the approximation

$$\hat{X} = U_r D_r V_r^T$$

$$\hat{X} = U_r U_r^T X$$

$$\hat{X} = X V_r V_r^T$$

where U_r contains the first r columns of U. Correspondingly, V_r.

Best Approximation

- The approximation above is the best rank-r approximation with respect to the squared error (Frobenius Norm). The approximation error is

$$\sum_{i=1}^{N} \sum_{j=1}^{M} (x_{i,j} - \widehat{x}_{i,j})^2 = \sum_{j=r+1}^{\tilde{r}} d_j^2$$

LSA: Similarities Between Documents

Feature Vectors for Documents

- Given a collection of N documents and M keywords
- X is the *term-frequency* (tf) matrix; $x_{i,j}$ indicates how often word j occurred in document i.
- Some classifiers use this representation as inputs
- On the other hand, two documents might discuss similar topics (are "semantically similar") without using the same key words
- By doing a PCA we can find document representations $\mathbf{z}_i = V^T \mathbf{x}_{i,*}^T$ that provide us with a good similarity measure between documents
- New: We can also find an improved representation of words via $\mathbf{t}_j = U^T \mathbf{x}_{*,j}$!
- This is known as *Latent Semantic Analysis* (LSA)

Simple Example

- In total 9 sentences (documents):
  - 5 documents on human-computer interaction (c1 - c5)
  - 4 documents on mathematical graph theory (m1 - m4)
- The 12 key words are in italic letters

From: Landauer, T. K., Foltz, P. W., Laham, D. (1998). Introduction to Latent Semantic Analysis. Discourse Processes, 25, 259-284.

Example of text data: Titles of Some Technical Memos

c1:	*Human* machine *interface* for ABC *computer* applications
c2:	A *survey* of *user* opinion of *computer system response time*
c3:	The *EPS user interface* management *system*
c4:	*System* and *human system* engineering testing of *EPS*
c5:	Relation of *user* perceived *response time* to error measurement
m1:	The generation of random, binary, ordered *trees*
m2:	The intersection *graph* of paths in *trees*
m3:	*Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4:	*Graph minors*: A *survey*

tf-Matrix and Word Correlations

- The tf-Matrix X
- Based on the original data, the Pearson correlation between *human* and *user* is negative, although one would assume a large semantic correlation

X^T

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

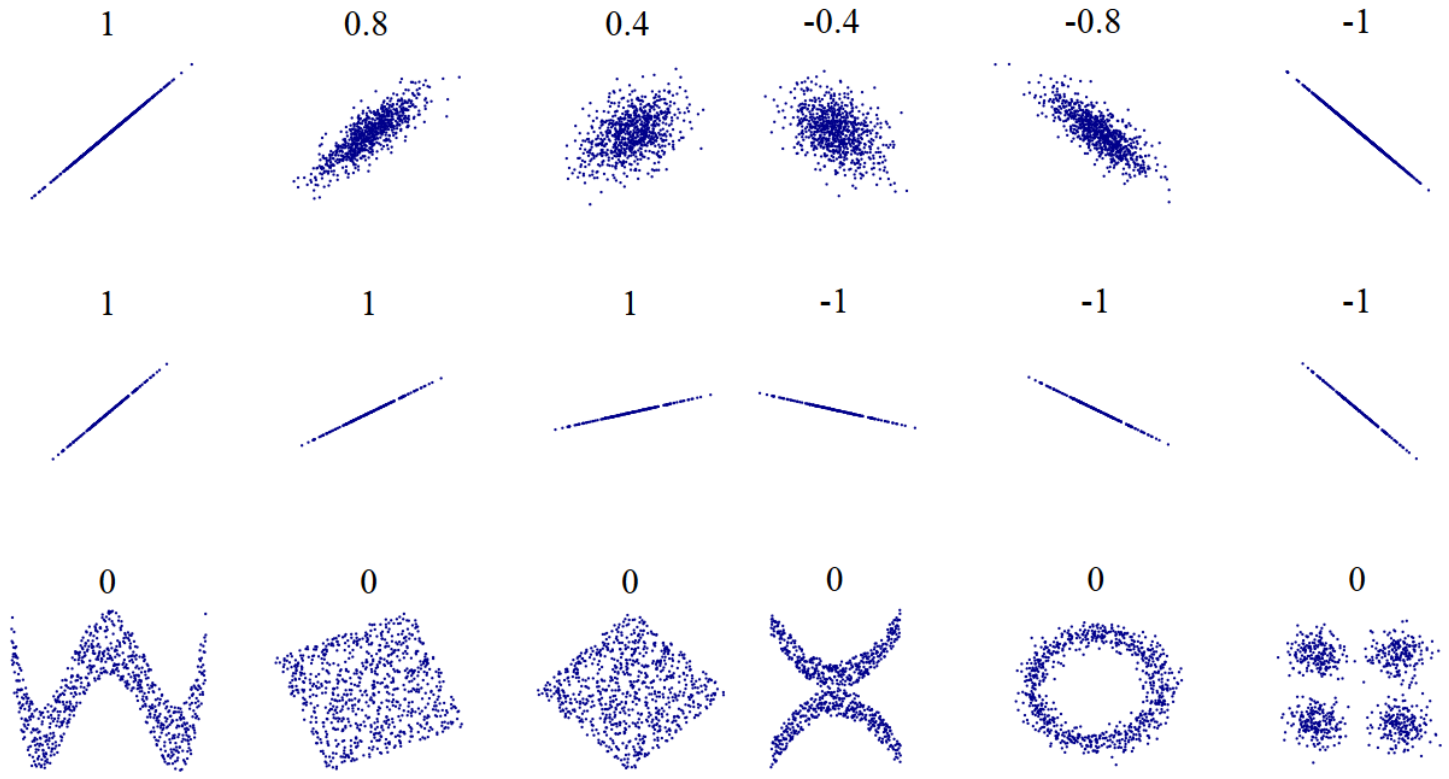
$\underline{r}$ (human.user) = -.38 Pearson correlation between the words *human* and *user*

$\underline{r}$ (human.minors) = -.29 Pearson correlation between the words *human* and *minor*

Site note: Pearson Correlation

- Measures linear correlation (dependence) between two random variables $X, Y \in \mathbb{R}$

$$r_{XY} = \frac{\text{cov}(X, Y)}{\text{std}(X)\text{std}(Y)} \in [-1, 1]$$



Singular Value Decomposition

- Decomposition $X = UDV^T$

$$X = UDV^T$$

V =

0.22	-0.11	0.29	-0.41	-0.11	-0.34	0.52	-0.06	-0.41
0.20	-0.07	0.14	-0.55	0.28	0.50	-0.07	-0.01	-0.11
0.24	0.04	-0.16	-0.59	-0.11	-0.25	-0.30	0.06	0.49
0.40	0.06	-0.34	0.10	0.33	0.38	0.00	0.00	0.01
0.64	-0.17	0.36	0.33	-0.16	-0.21	-0.17	0.03	0.27
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.27	0.11	-0.43	0.07	0.08	-0.17	0.28	-0.02	-0.05
0.30	-0.14	0.33	0.19	0.11	0.27	0.03	-0.02	-0.17
0.21	0.27	-0.18	-0.03	-0.54	0.08	-0.47	-0.04	-0.58
0.01	0.49	0.23	0.03	0.59	-0.39	-0.29	0.25	-0.23
0.04	0.62	0.22	0.00	-0.07	0.11	0.16	-0.68	0.23
0.03	0.45	0.14	-0.01	-0.30	0.28	0.34	0.68	0.18

D =

3.34								
	2.54							
		2.35						
			1.64					
				1.50				
					1.31			
						0.85		
							0.56	
								0.36

U =

0.20	0.61	0.46	0.54	0.28	0.00	0.01	0.02	0.08
-0.06	0.17	-0.13	-0.23	0.11	0.19	0.44	0.62	0.53
0.11	-0.50	0.21	0.57	-0.51	0.10	0.19	0.25	0.08
-0.95	-0.03	0.04	0.27	0.15	0.02	0.02	0.01	-0.03
0.05	-0.21	0.38	-0.21	0.33	0.39	0.35	0.15	-0.60
-0.08	-0.26	0.72	-0.37	0.03	-0.30	-0.21	0.00	0.36
0.18	-0.43	-0.24	0.26	0.67	-0.34	-0.15	0.25	0.04
-0.01	0.05	0.01	-0.02	-0.06	0.45	-0.76	0.45	-0.07
-0.06	0.24	0.02	-0.08	-0.26	-0.62	0.02	0.52	-0.45

Approximation with $r = 2$ and Word Correlations

- Reconstruction $\hat{X}$ with $r = 2$
- Shown is $\hat{X}^T$
- Based on $\hat{X}$ the correlation between *human* and *user* is almost one! The similarity between *human* and *minors* is strongly negative (as it should be)
- In document $m4$: *Graph minors: a survey* the word *survey* which is in the original document gets a smaller value than the term *trees*, which was not in the document originally

$\hat{X}^T$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

r (human.user) = .94 — Pearson correlation between the words *human* and *user*

r (human.minors) = -.83 — Pearson correlation between the words *human* and *minor*

Document Correlations in the Original and the Reconstructed Data

- Top: document correlation in the original data X: The average correlation between documents in the c-class is almost zero
- Bottom: in $\widehat{X}$ there is a strong correlation between documents in the same class and strong negative correlation across document classes

Correlations between titles in raw data:

	c1	c2	c3	c4	c5	m1	m2	m3
c2	-0.19							
c3	0.00	0.00						
c4	0.00	0.00	0.47					
c5	-0.33	0.58	0.00	-0.31				
m1	-0.17	-0.30	-0.21	-0.16	-0.17			
m2	-0.26	-0.45	-0.32	-0.24	-0.26	0.67		
m3	-0.33	-0.58	-0.41	-0.31	-0.33	0.52	0.77	
m4	-0.33	-0.19	-0.41	-0.31	-0.33	-0.17	0.26	0.56

0.02		
-0.30	0.44	Average Pearson correlation in the three document blocks in the raw data

Correlations in two dimensional space:

c2	0.91							
c3	1.00	0.91						
c4	1.00	0.88	1.00					
c5	0.85	0.99	0.85	0.81				
m1	-0.85	-0.56	-0.85	-0.88	-0.45			
m2	-0.85	-0.56	-0.85	-0.88	-0.44	1.00		
m3	-0.85	-0.56	-0.85	-0.88	-0.44	1.00	1.00	
m4	-0.81	-0.50	-0.81	-0.84	-0.37	1.00	1.00	1.00

0.92		
-0.72	1.00	Average Pearson correlation in the three document blocks in the reconstructed data

Applications of LSA

- LSA-similarity often corresponds to the human perception of document or word similarity
- There are commercial applications in the evaluation of term papers
- There are indications that search engine providers like Google and Yahoo, use LSA for the ranking of pages and to filter out spam (spam is unusual, novel)

Recommendation Engines

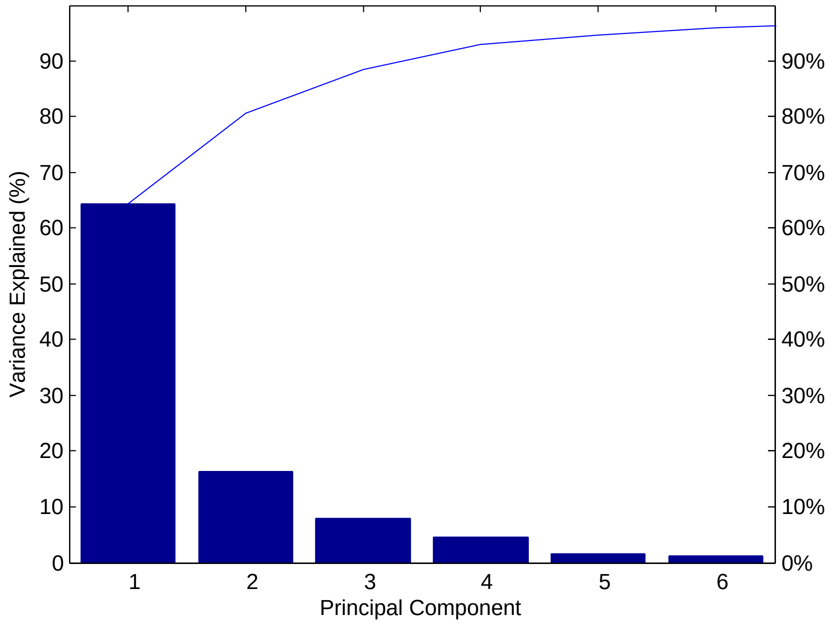
- The requirement that the approximation is optimal for any r lead to the PCA/SVD as unique solution
- If we only require a decomposition of the form $X = AB^T$ where A and B have r columns, then the solution is not unique and the columns of A and B are not necessarily orthonormal
- Such a decomposition is the basis for recommender systems where matrix entries correspond to rating of a **user (row)** for a **movie (column)**
- The components of A and B are regularized and the Frobenius norm is only calculated with respect to known ratings (which means that missing ratings are ignored in the optimization)

Further Example: PCA on Turbine Data

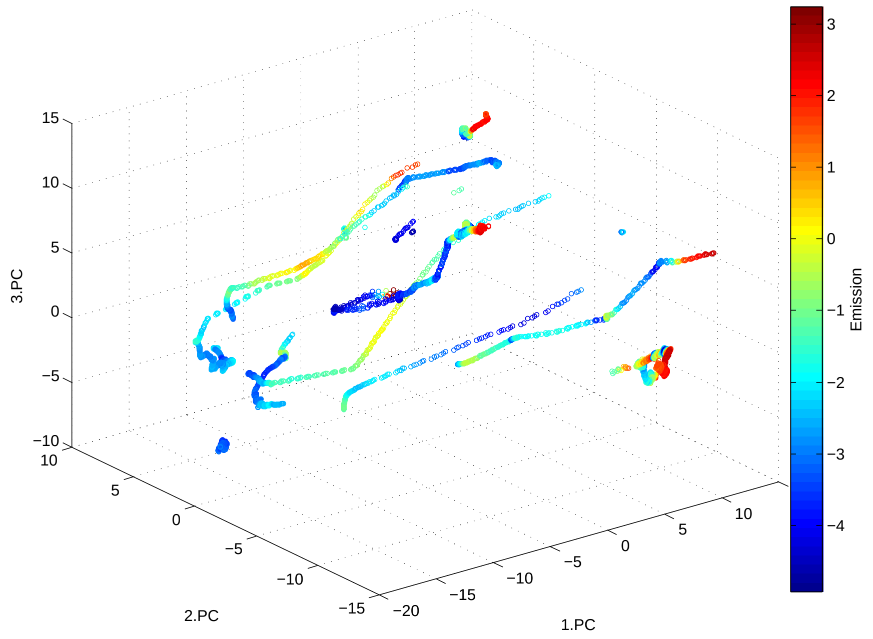
- 9000 data points, each with 279 inputs (sensor measurements), 1 output (emission)
- Visualizing first 3 principal components gives a good impression of the data distribution (here: trajectories in state space). The first 3 PCs explain nearly 90 percent of the variance of the data
- However, the principal components are not easy to interpret (linear combination of the original inputs)
- As Matlab code:

```
[coef,scores,variances] = pca(X);
percent_explained = 100*variances/sum(variances);
figure; pareto(percent_explained)
xlabel('Principal Component')
ylabel('Variance Explained (%)')

figure; scatter3(scores(:,1),scores(:,2),scores(:,3),10,Y);
xlabel('1.PC'); ylabel('2.PC'); zlabel('3.PC');
set(get(colorbar,'ylabel'),'String', 'Emission');
```



Variance Explained (%)
Principal Component
0
10
20
30
40
50
60
70
80
90
1
2
3
4
5
6
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%



3.PC
2.PC
1.PC
Emission
15
10
5
0
−5
−10
10
5
0
−5
−10
−15
−20
−15
−10
−5
0
5
10
3
2
1
0
−1
−2
−3
−4