CHAPTER 6:

DIMENSIONALITY REDUCTION

Stella Grasshof

Overview: Dimensionality Reduction

1) Intro
2) Subset Selection
3) Principal Component Analysis (PCA)
   - Feature Embedding
   - SVD and Factorization
   - Factor Analysis (FA)
4) Canonical Correlation Analysis (CCA)
5) Linear Discriminant Analysis (LDA)
6) Multidimensional Scaling (MDS)
7) Isomap
8) LLE (Locally Linear Embedding)
9) Laplacian Eigenmaps

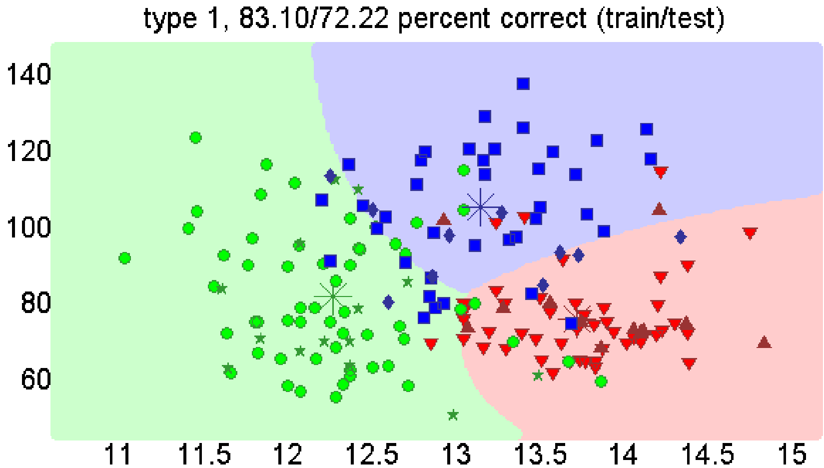
Feature Extraction

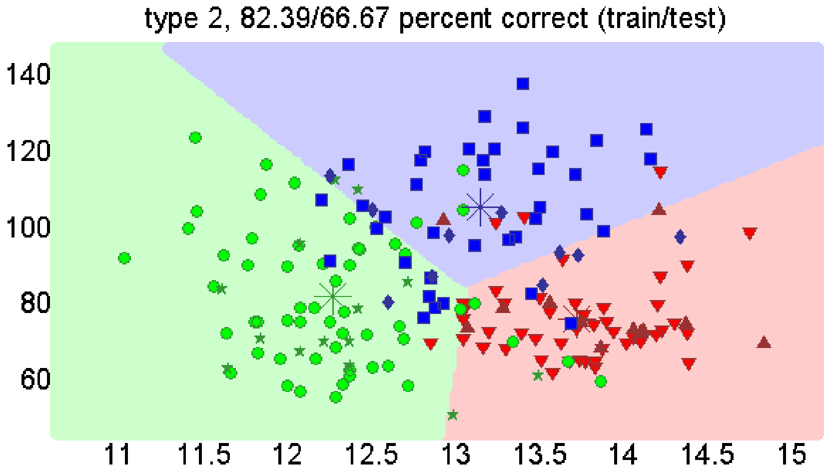
Why Reduce Dimensionality?

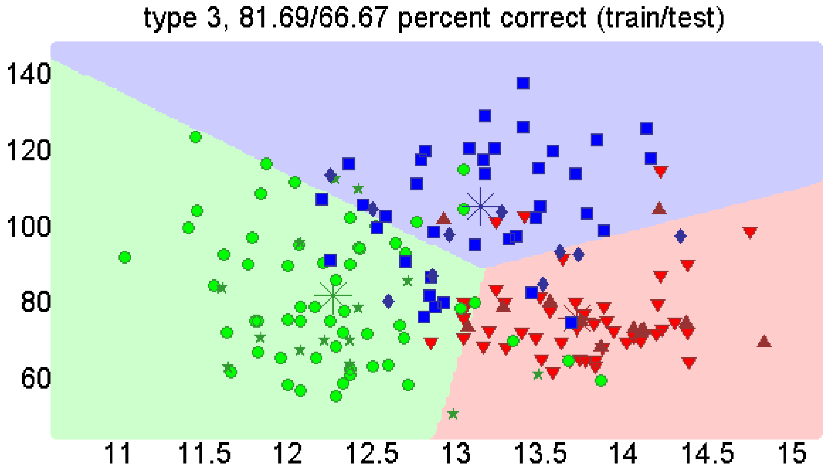
- Reduces time complexity: Less computation
- Reduces space complexity: Fewer parameters
- Saves the cost of observing the feature
- Simpler models are more robust on small datasets
- More interpretable, simpler explanation
- Data visualization (structure, groups, outliers, etc) if plotted in 2 or 3 dimensions

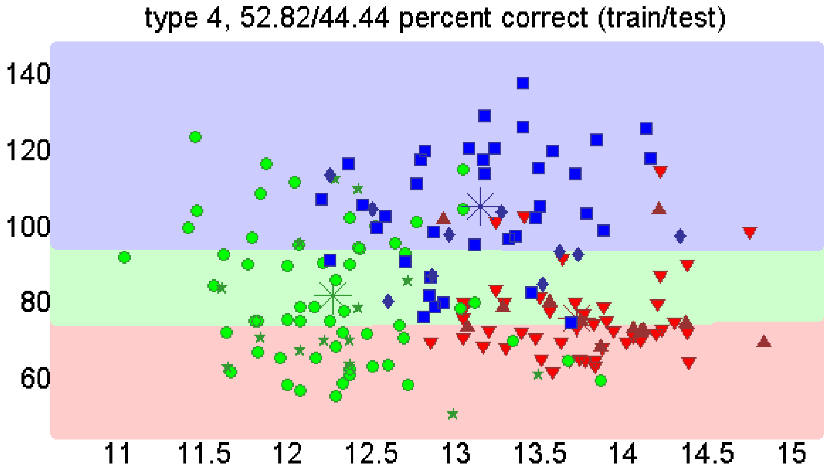
Recap: Parametric Classification

Result **without** preprocessing by z-normalization



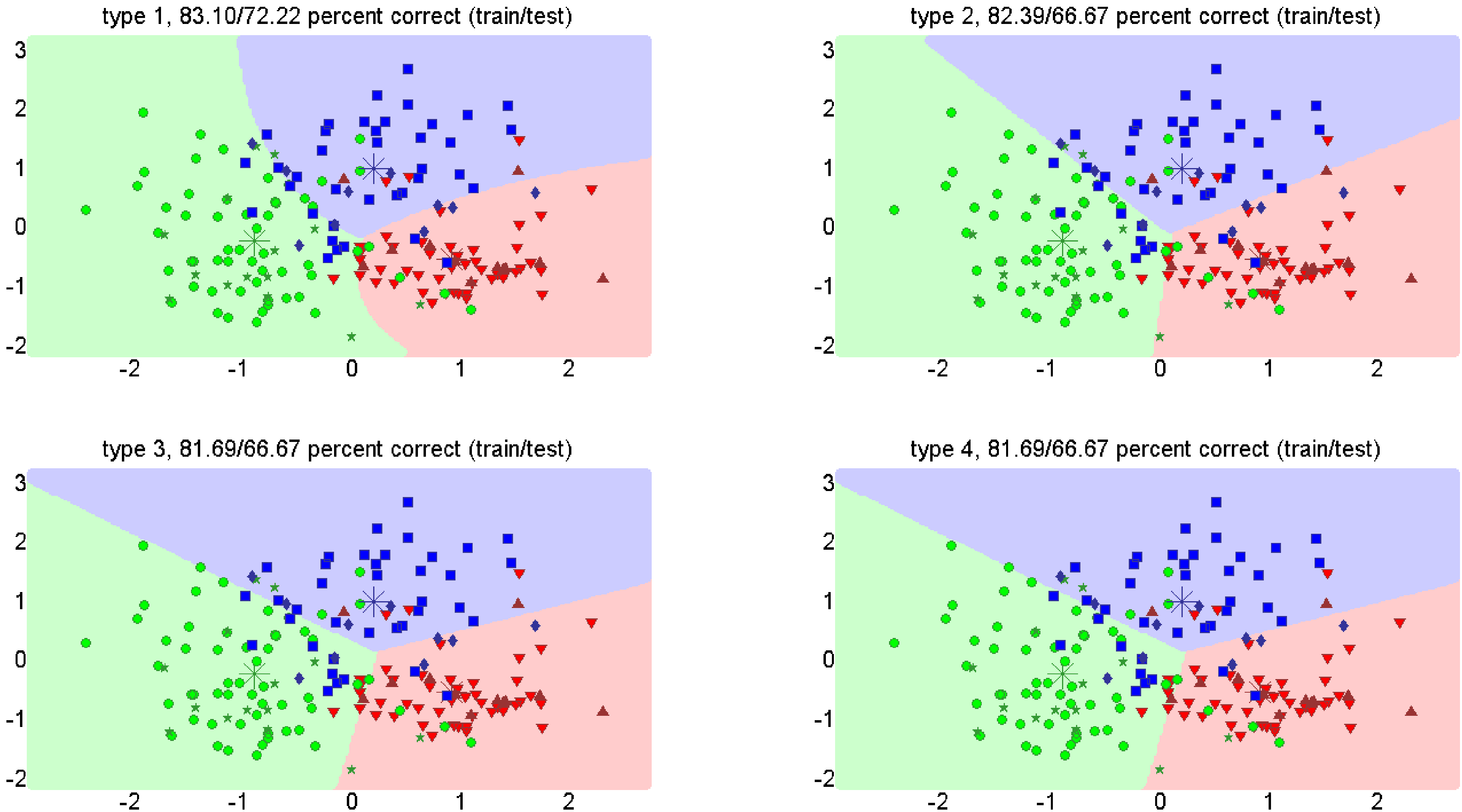






Recap: Parametric Classification

Result **with** preprocessing by z-normalization



Feature Selection vs. Extraction

Feature selection:

- Choosing $K<D$ important features
- ignoring the remaining $D-K$

$\Rightarrow$ Subset selection algorithms

Feature extraction:

- Project the original x_d , $d=1,...,D$ to $K<D$ new dimensions z_k , $k=1,...,K$

Subset Selection

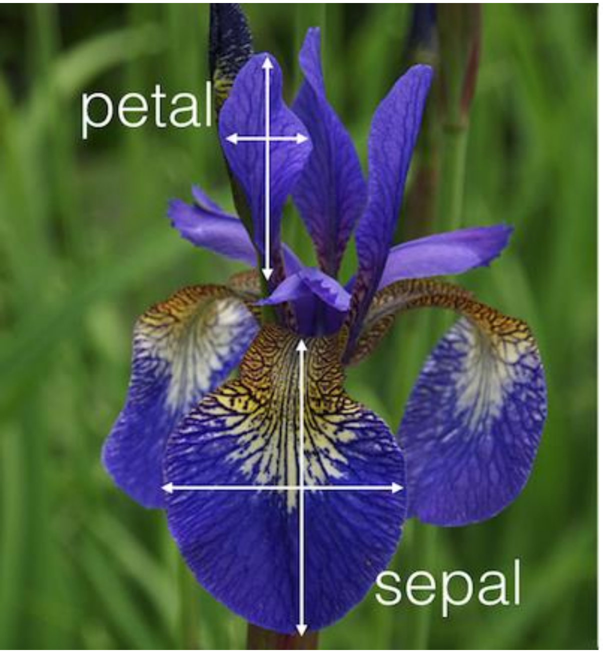
- **Forward search**: Add the “best“ feature at each step
  - Initialize set of features F as empty set Ø
  - At each iteration:
    - Find best new feature: $d = \text{argmin}_i \, Err(F \cup x_i)$
    - Add x_d to F **if** $Err(F \cup x_d) < Err(F)$

Problems:
costly, greedy, no guarantee of “best“ subset

- **Backward search**:
  Start with all features and remove one at a time

- **Floating search**:
  not one-by-one, instead:
  add K, remove M

Iris Dataset



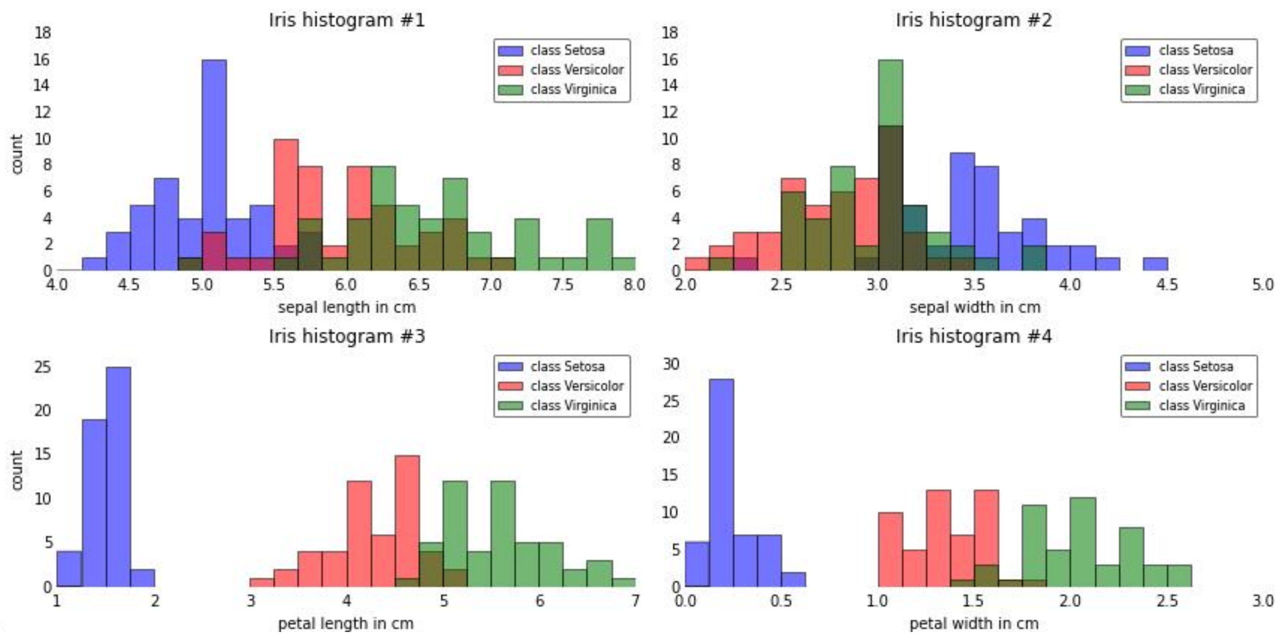


Image credit: **Sebastian Raschka**
Source:
https://github.com/ChildMindInstitute/pattern-classification-tutorials/blob/master/machine_learning/supervised_intro/introduction_to_supervised_machine_learning.md

Iris Data: Select 1 of 4

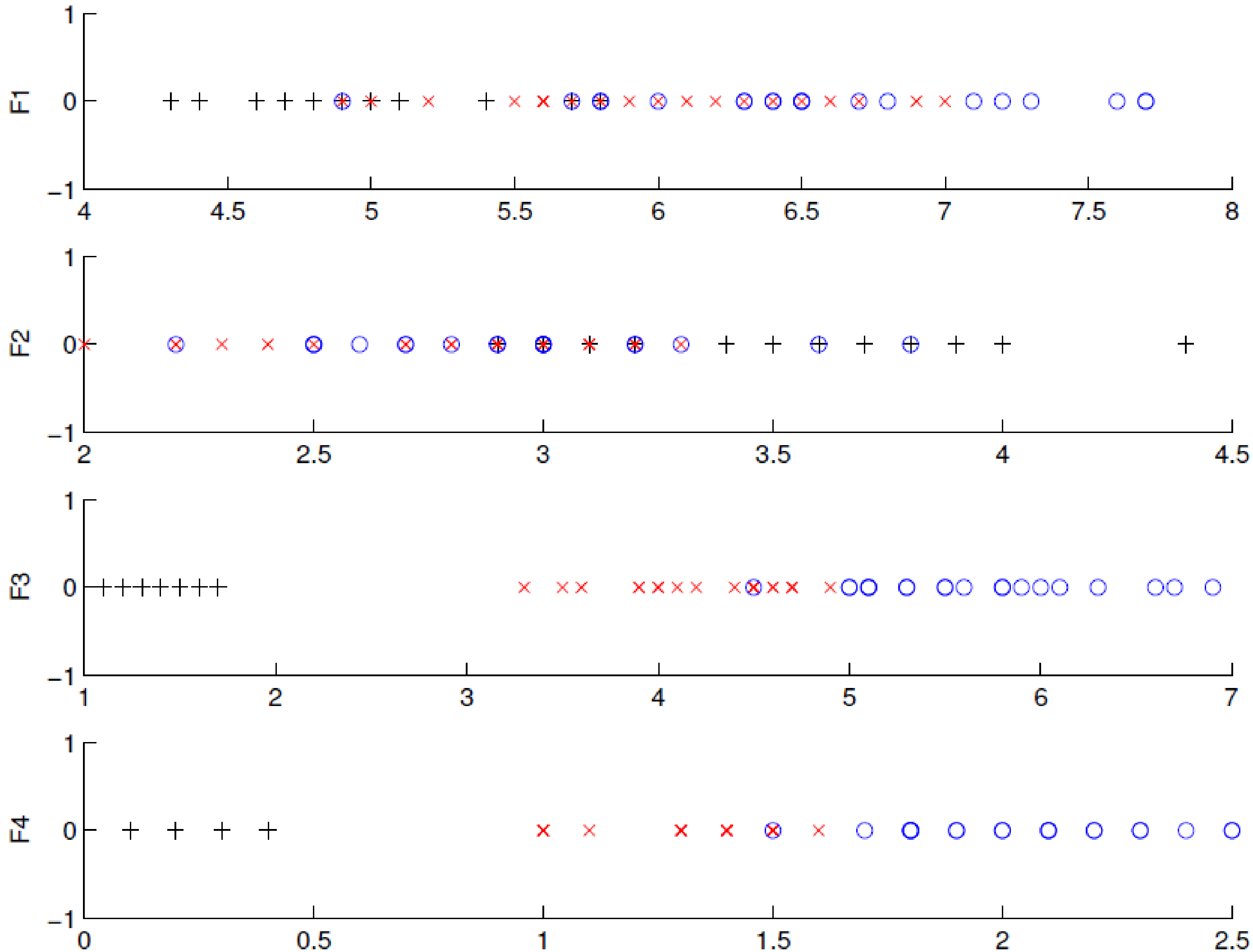
Selection Criteria: max accuracy of nearest mean classifier

76%

57%

92%

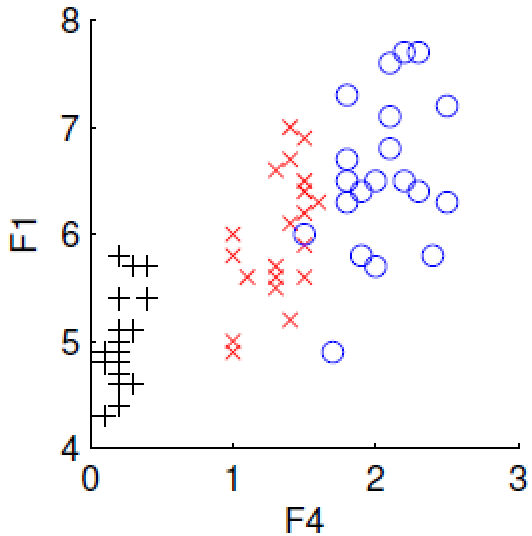
94%



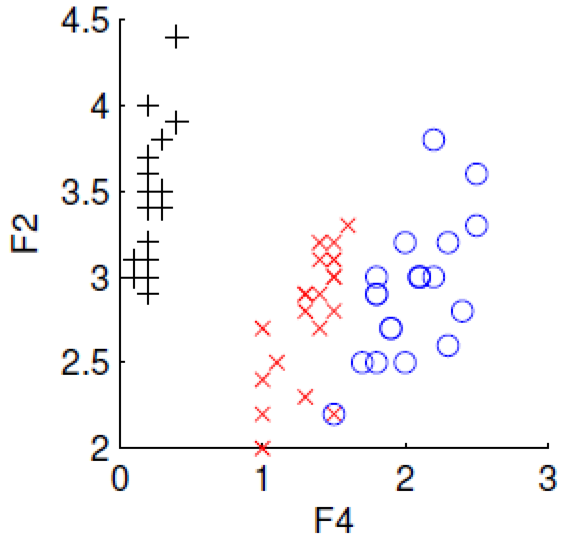
<= Chosen

Iris Data: Select 2 of 4

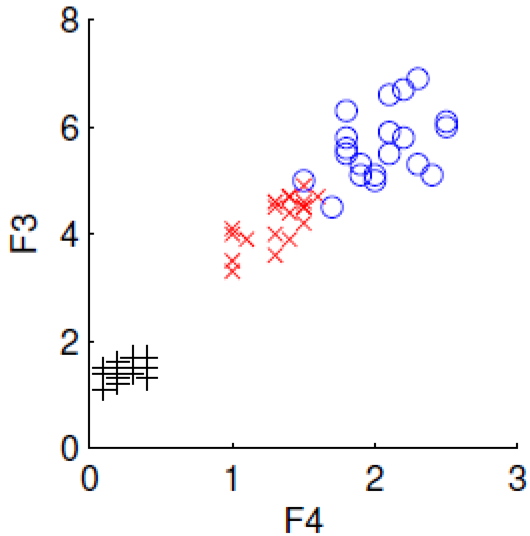
Selection Criteria: max accuracy of nearest mean classifier



87%



92%



96%

=> Subset (F3,F4)

No third feature will be added, because accuracy does not increase

Subset Selection

When is it sensible?

- independent features
- Requires some prior knowledge, hence **supervised**

When is it not sensible?

- e.g. if features are single pixels of an image, because pixels of one image are correlated

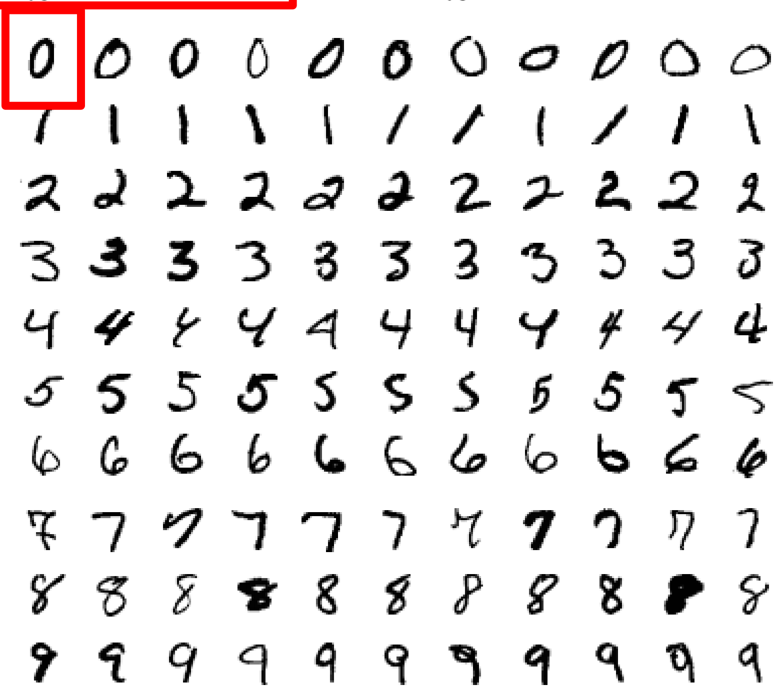
=> Now: get new features by Feature Extraction

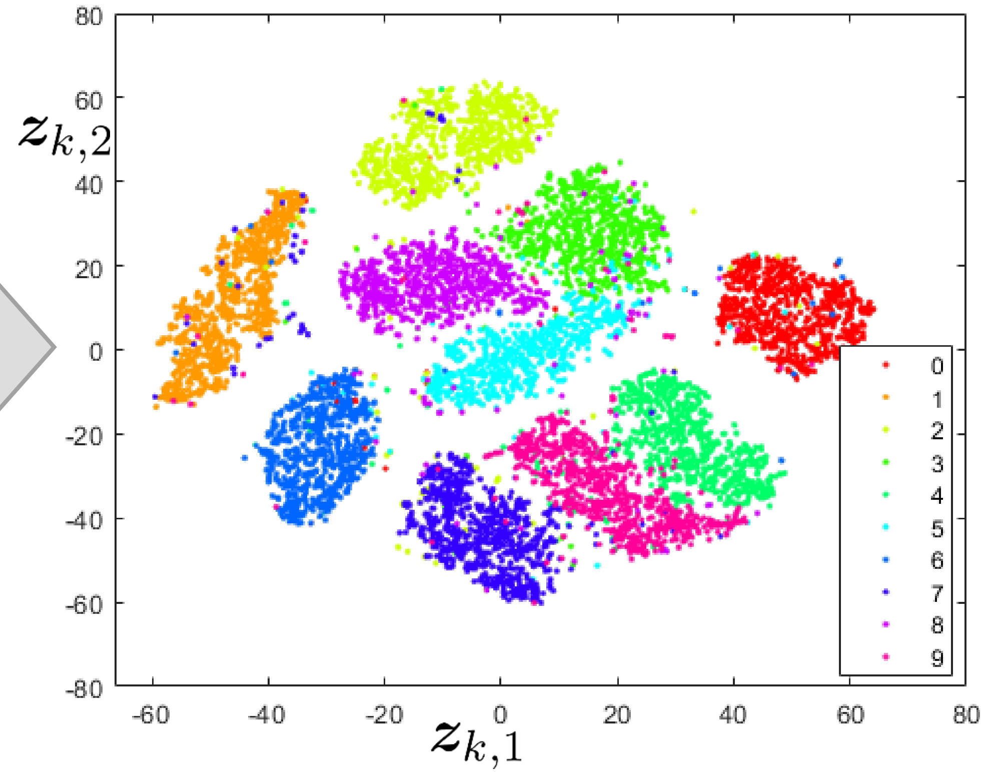
Feature Extraction

Consider high-dimensional data is given, we want:

- Compact representation of the data
- Extract **most relevant** information

$\boxed{\boldsymbol{x}_k \in \mathbb{R}^{64\cdot 64}} \mapsto \boldsymbol{z}_k \in \mathbb{R}^2$





Sources:
[1] MNIST, wikpedia.org, Josef Steppan [CC BY-SA 4.0 (https://creativecommons.org/licenses/by-sa/4.0)]
[2] MathWorks, https://se.mathworks.com/help/stats/visualize-high-dimensional-data-using-t-sne.html

Short Revisit: Multivariate Data

- Random variable $\boldsymbol{x} \in \mathbb{R}^D$
- Expectation value $\mathrm{E}[\boldsymbol{x}] = \boldsymbol{\mu} = [\mu_1, \ldots, \mu_D]^{\mathrm{T}} \in \mathbb{R}^D$
- Covariance matrix

$$\boldsymbol{\Sigma} \equiv \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \ldots & \sigma_{1D} \\ \sigma_{21} & \sigma_2^2 & \ldots & \sigma_{2D} \\ & \vdots & & \\ \sigma_{D1} & \sigma_{D2} & \ldots & \sigma_D^2 \end{pmatrix} \in \mathbb{R}^{D \times D}$$

$$\boldsymbol{\Sigma} \equiv \mathrm{Cov}(\boldsymbol{x}) = \mathrm{E}[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T]$$

- Data matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ with N samples

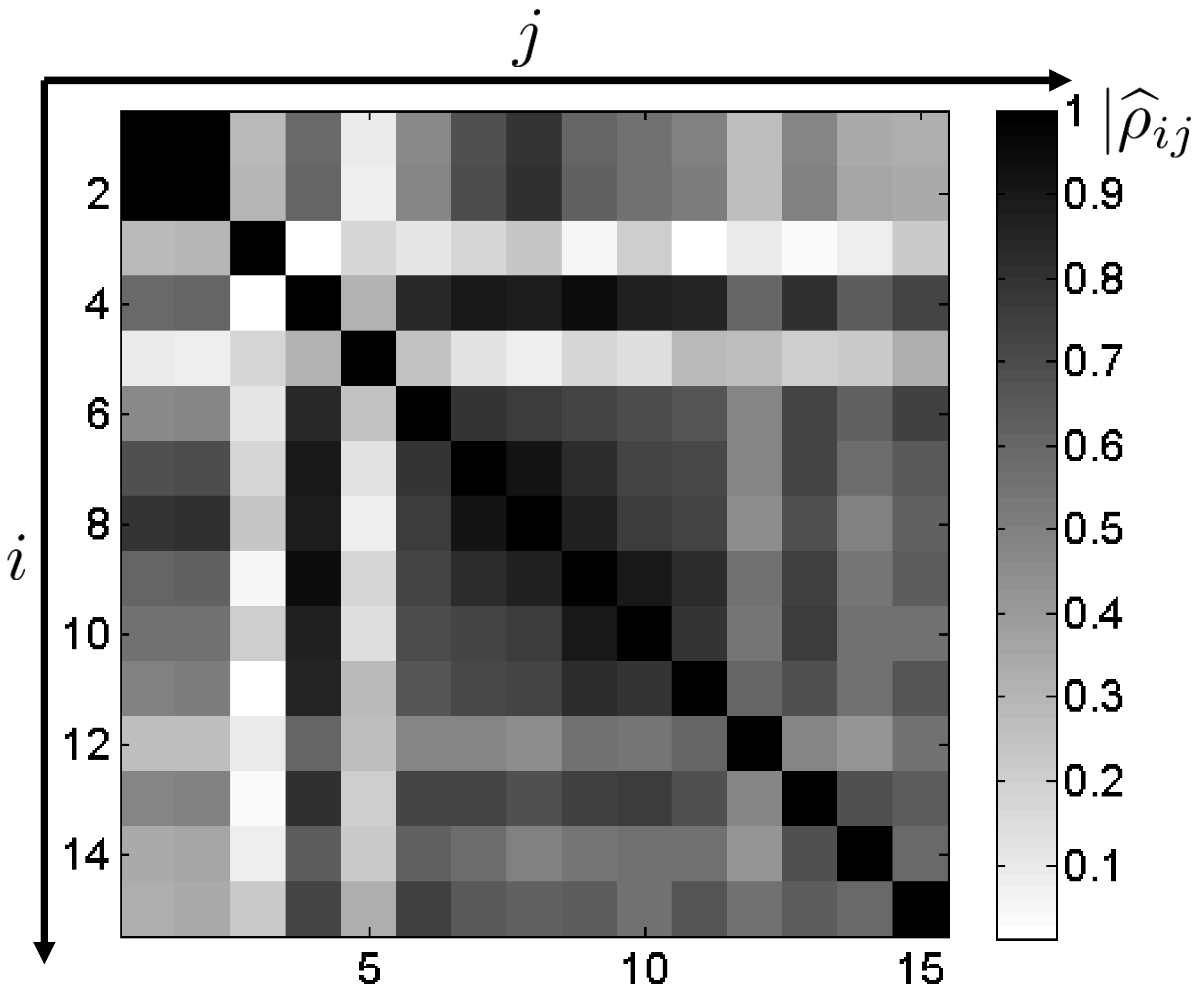
$$\mathbf{X} = \begin{pmatrix} X_1^1 & X_2^1 & \ldots & X_D^1 \\ X_1^2 & X_2^2 & \ldots & X_D^2 \\ & \vdots & & \\ X_1^N & X_2^N & \ldots & X_D^N \end{pmatrix}$$

$$\boldsymbol{M} = \begin{pmatrix} \widehat{\boldsymbol{\mu}}^{\mathrm{T}} \\ \vdots \\ \widehat{\boldsymbol{\mu}}^{\mathrm{T}} \end{pmatrix} \in \mathbb{R}^{N \times D}$$

$$\widehat{\boldsymbol{\Sigma}} = \frac{1}{N}(\boldsymbol{X} - \boldsymbol{M})^{\mathrm{T}}(\boldsymbol{X} - \boldsymbol{M})$$

Short Revisit: Correlation

absolute values of scaled covariance matrix: $|\widehat{\mathbf{\Sigma}}|$, $\widehat{\mathbf{\Sigma}} = \widehat{\mathbf{\Sigma}}^{\mathrm{T}}$



Correlation matrix consists of scaled entries σ_{ij} of $\mathbf{\Sigma}$, i.e.

$$\mathrm{Corr}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \rho_{ij}$$

$$\mathrm{Corr}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sigma_{ij}}{\sigma_i \sigma_j}$$

pairwise correlation between features

$\boldsymbol{x}_i, \ \boldsymbol{x}_j,$

$i, j = 1, \ldots, D$

=> the darker the pixel at (i,j), the larger $|\rho_{ij}|$

A high/low correlation is neither good nor bad.

Principal Components Analysis (PCA)

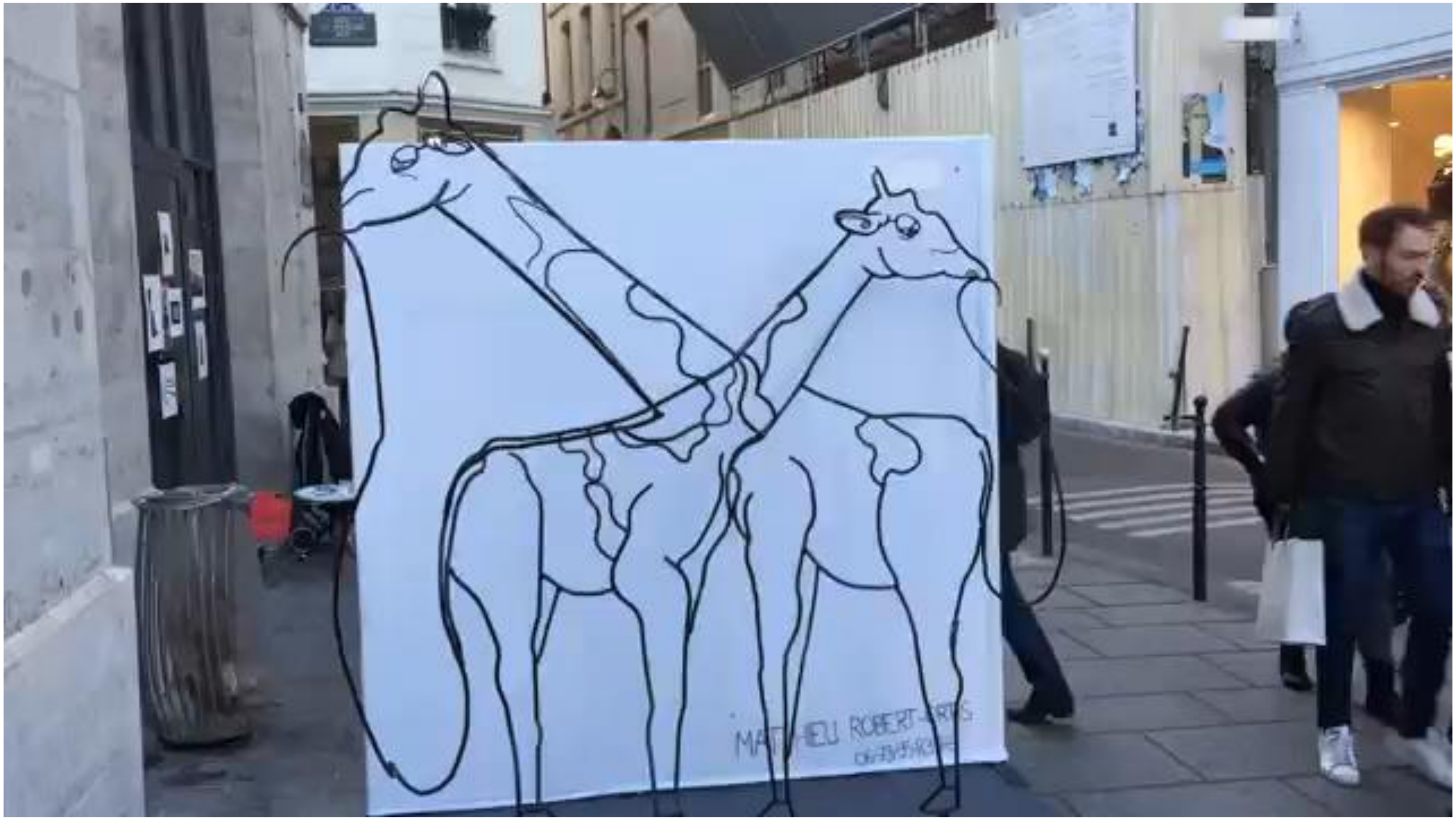
- Find a low-dimensional space such that: when **x** is projected there, “information loss“ is minimized
- Find **direction of maximum variance**
- new directions must be **uncorrelated**, i.e. covariance matrix is diagonal

Idea:

- rotate original data

Principal Components Analysis (PCA)

Why would rotating the data tell me more?

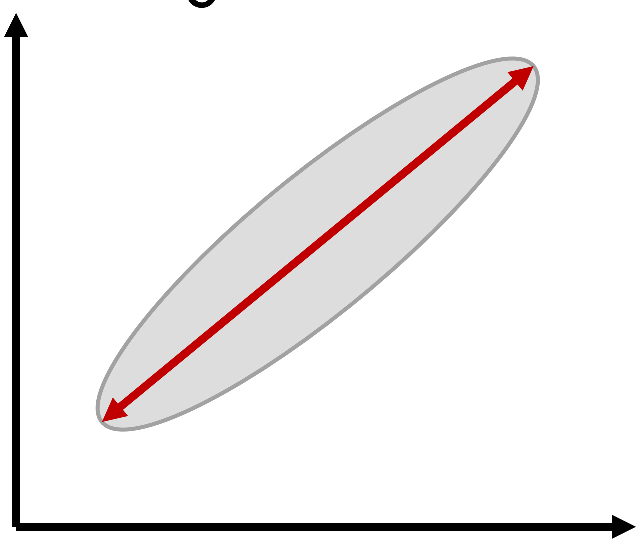


https://www.youtube.com/watch?v=PiYMoI0VjWo

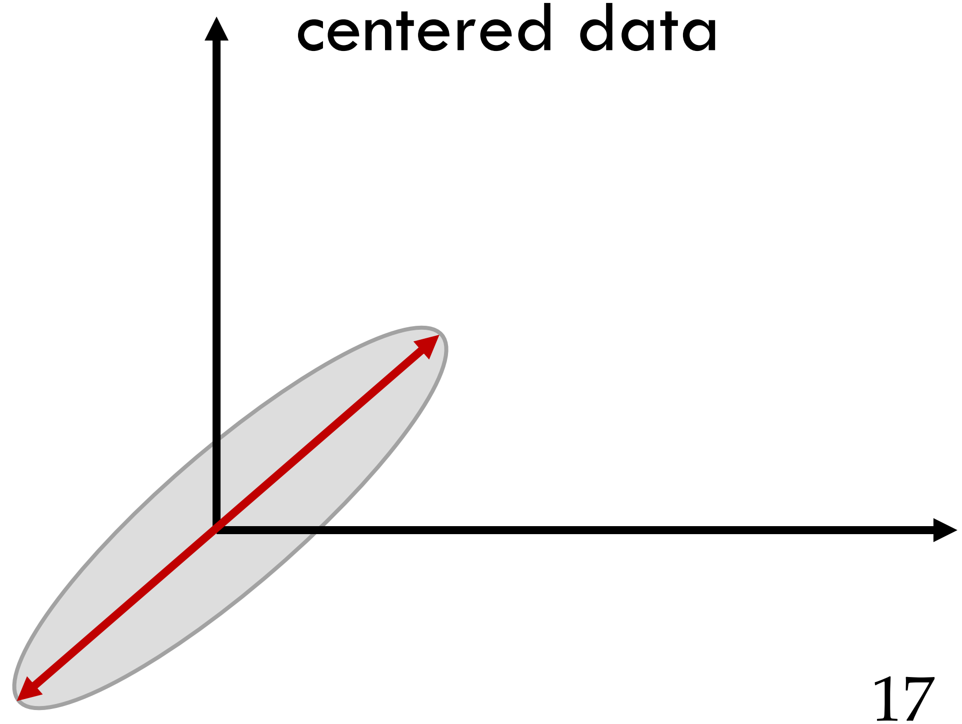
Principal Components Analysis (PCA)

- Before rotating the data, we must center the data
- Where is the direction of maximum variance?

original data



mean-free
centered data



Principal Components Analysis (PCA)

$$\mathbf{X} \in \mathbb{R}^{N \times D}$$

$$\mathrm{E}[\boldsymbol{x}] = \boldsymbol{\mu} = [\mu_1, \ldots, \mu_D]^{\mathrm{T}} \in \mathbb{R}^D$$

1. Subtract mean $\boldsymbol{X} - \boldsymbol{M}, \quad \boldsymbol{M} = (\widehat{\boldsymbol{\mu}}, \ldots, \widehat{\boldsymbol{\mu}})^{\mathrm{T}} \in \mathbb{R}^{N \times D}$
2. Compute covariance matrix $\widehat{\boldsymbol{\Sigma}} = \dfrac{1}{N}(\boldsymbol{X} - \boldsymbol{M})^{\mathrm{T}}(\boldsymbol{X} - \boldsymbol{M})$
3. Compute eigenvectors of covariance matrix

$$\widehat{\boldsymbol{\Sigma}}\boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k,\ k = 1, ..., D,\ \lambda_i \geq \lambda_j,\ i > j$$

$$\boldsymbol{w}_i^{\mathrm{T}}\boldsymbol{w}_j = \begin{cases} 1 & , i = j \\ 0 & , i \neq j \end{cases}$$

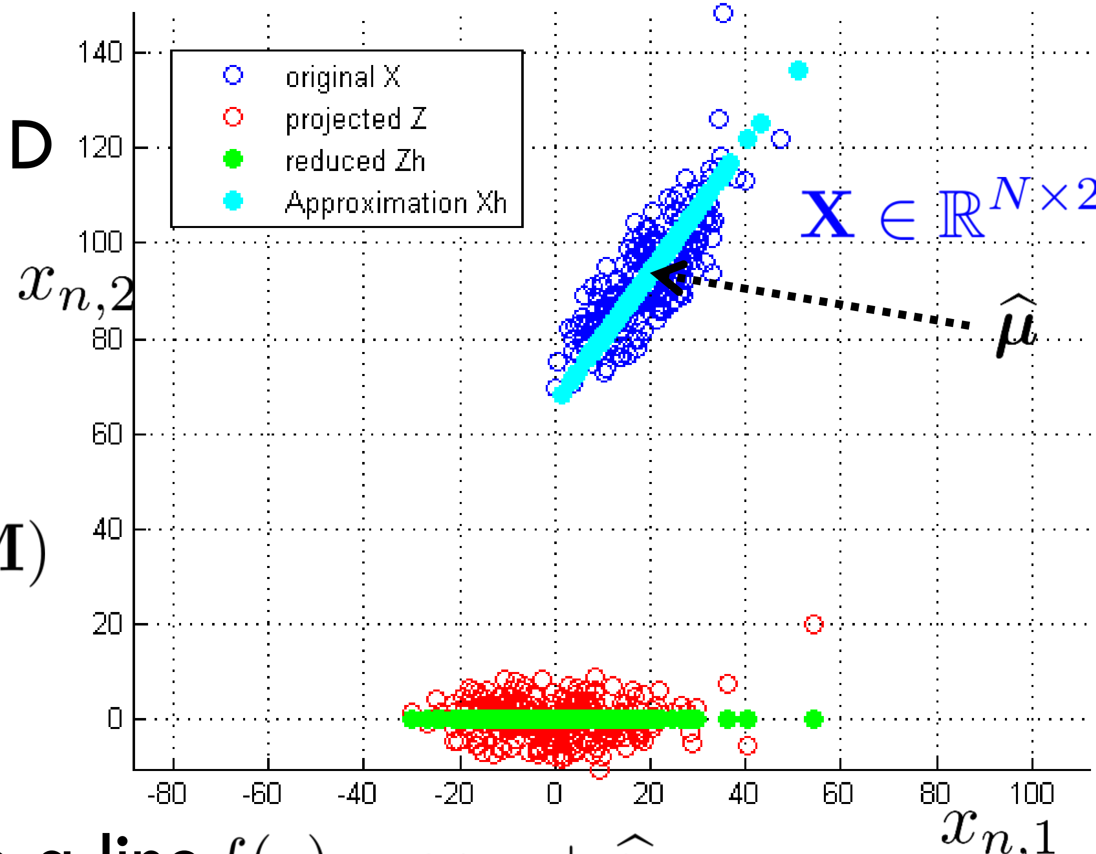
$$[\boldsymbol{w}_1, \ldots, \boldsymbol{w}_D] = \boldsymbol{W}^{\mathrm{T}}$$

$$[\boldsymbol{w}_1, \ldots, \boldsymbol{w}_K] = \boldsymbol{W}_K^{\mathrm{T}}$$

$\boldsymbol{w}_k$ are **principal components**

4. New variables $\boldsymbol{z}_n = \boldsymbol{W}^{\mathrm{T}}(\boldsymbol{x}_n - \widehat{\boldsymbol{\mu}})$
5. Reconstrution $\widehat{\boldsymbol{x}}_n = \boldsymbol{W}_K \boldsymbol{z}_n + \widehat{\boldsymbol{\mu}}$

Principal Components Analysis (PCA)



1. Consider 2D data shall be mapped to 1D

   $$\boldsymbol{x}_n \in \mathbb{R}^2 \;\mapsto\; \boldsymbol{z}_n \in \mathbb{R}^1$$

2. Covariance of mean centerd data

   $$\widehat{\Sigma} = \frac{1}{N}(\mathbf{X}-\mathbf{M})^T(\mathbf{X}-\mathbf{M})$$

3. Eigenvectors

   $$\widehat{\boldsymbol{\Sigma}}\boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k,\;\; k = 1,2$$

4. Choose k=1 to define a line $f(z) = \boldsymbol{w}_1 z + \widehat{\boldsymbol{\mu}}$ leads to
   - Reduction $z_n = \boldsymbol{w}_1^{\mathrm{T}}(\boldsymbol{x}_n - \widehat{\boldsymbol{\mu}}) \in \mathbb{R}^1$
   - Reconstruction $\widehat{\boldsymbol{x}}_n = \boldsymbol{w}_1 z_n + \widehat{\boldsymbol{\mu}} \in \mathbb{R}^2$

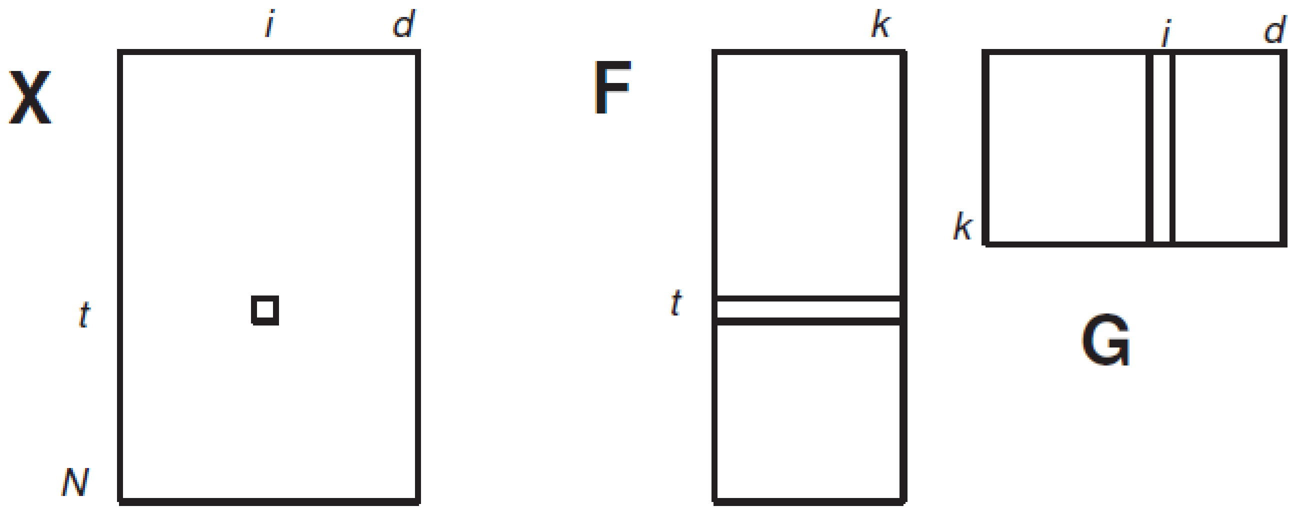
Matrix Factorization

X = ***F*** ***G***

[N x D] = *[NxK] [KxD]*

$$\mathbf{X}_{ti} = \mathbf{F}_t^T \mathbf{G}_i = \sum_{j=1}^{k} \mathbf{F}_{tj} \mathbf{G}_{ji}$$

*=> K can change, without **X** changing size*

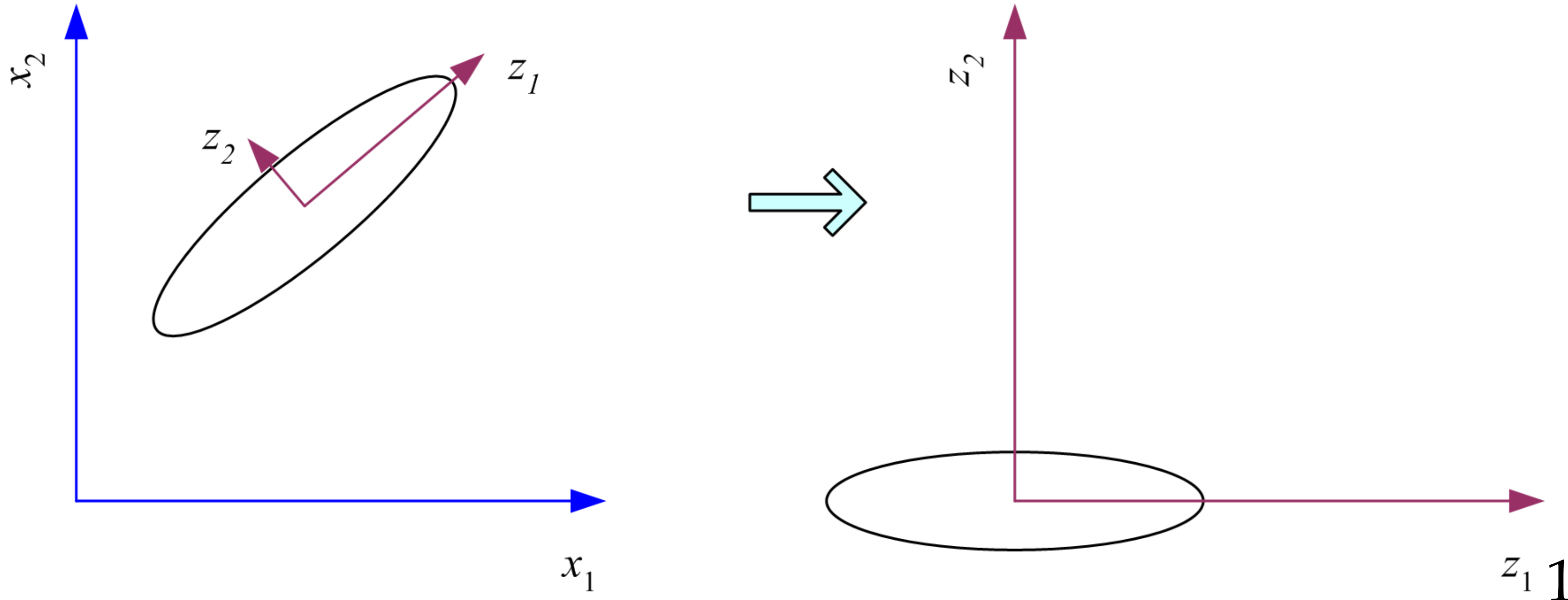


Principal Components Analysis (PCA)

$$\boldsymbol{z} = \mathbf{W}^T(\boldsymbol{x} - \boldsymbol{m})$$

where the columns of **W** are the eigenvectors of ∑ and ***m*** is sample mean

Centers the data at the origin and rotates the axes



Principal Components Analysis (PCA)

Eigenfaces (Turk 1991), 40 images, each 256x256

Source: Jean-Luc Nagel

https://www.researchgate.net/figure/1-Example-of-eigenfaces-Example-obtained-from-the-X2MVTS-database-cf-Subsection_fig3_33682412

Principal Components Analysis (PCA)

Dimension problem? $\boldsymbol{X} \leftarrow \boldsymbol{X} - \boldsymbol{M}$

- *N=40 images 256x256* $\Rightarrow$ *D=65,536* $\Rightarrow N \ll D$
- ***Problem:*** $\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} \in \mathbb{R}^{D \times D}$
  - size is [65,536 x 65,536]
  - but at most rank 40, because $\min(D, N) \overset{\text{here}}{=} N = 40$

- ***Trick:***

[65,536 x 65,536] vs. [40 x 40]

$$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X} \in \mathbb{R}^{D \times D} \rightsquigarrow \boldsymbol{X}\boldsymbol{X}^{\mathrm{T}} \in \mathbb{R}^{N \times N}$$

$$\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{w}_i = \lambda_i \boldsymbol{w}_i$$

$$\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}} \underbrace{\boldsymbol{X}\boldsymbol{w}_i}_{\boldsymbol{v}_i} = \lambda_i \underbrace{\boldsymbol{X}\boldsymbol{w}_i}_{\boldsymbol{v}_i}$$

$$\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i$$

Attention: Scaling!

Principal Components Analysis (PCA)

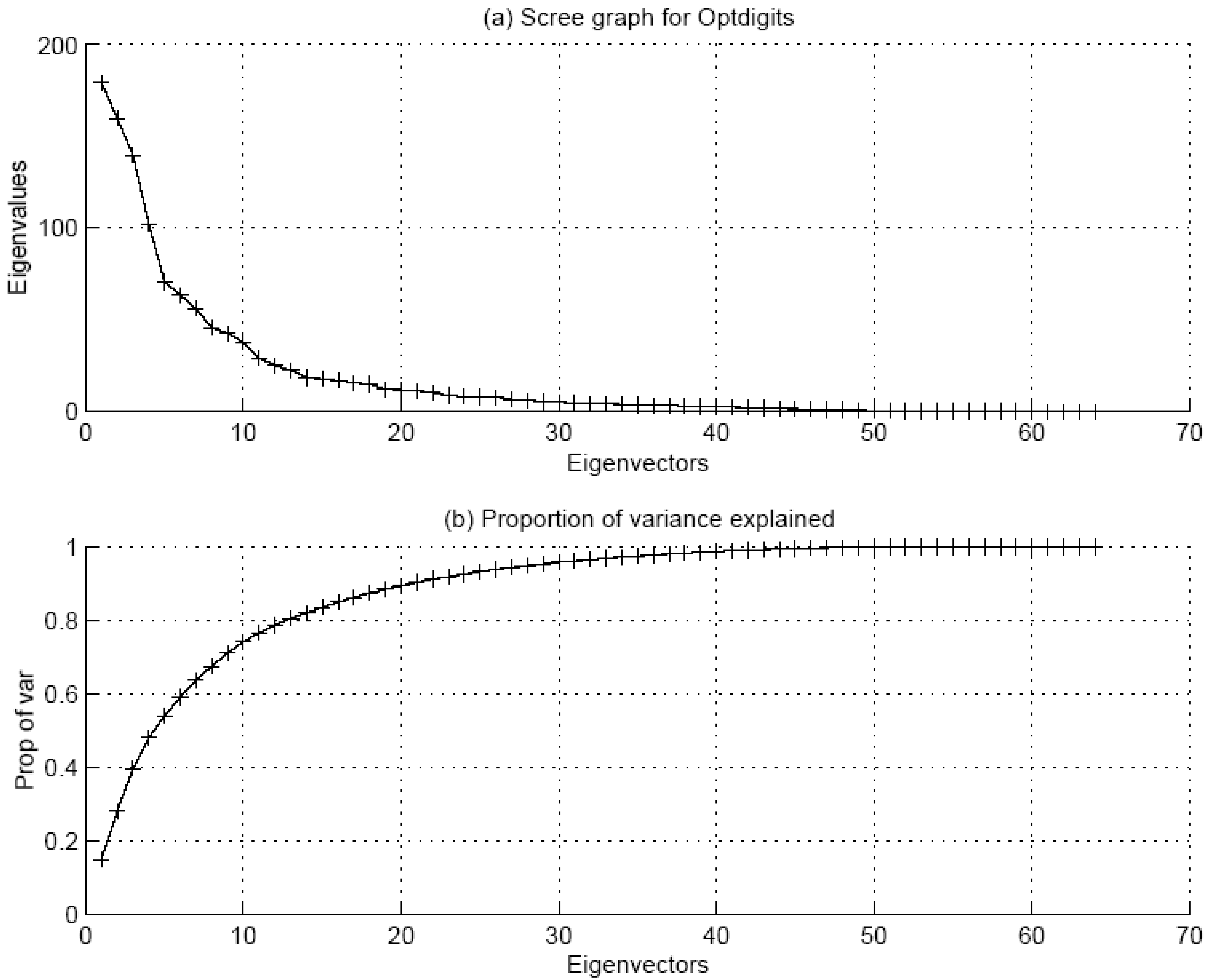
- Assume **X** is mean centered
- When $\boldsymbol{X}$ is the NxD data matrix,
  - $\boldsymbol{X}^T\boldsymbol{X}$ is the DxD matrix (covariance of features, if mean-centered)
  - $\boldsymbol{X}\boldsymbol{X}^T$ is the NxN matrix (pairwise similarities of instances)
- PCA: eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$ are D-dim, can be used for projection
- ***Feature embedding:*** eigenvectors of $\boldsymbol{X}\boldsymbol{X}^T$ are N-dim, give directly the coordinates after projection
- If only pairwise similarities (or distances) between instances: we can use ***feature embedding*** without needing to represent instances as vectors.

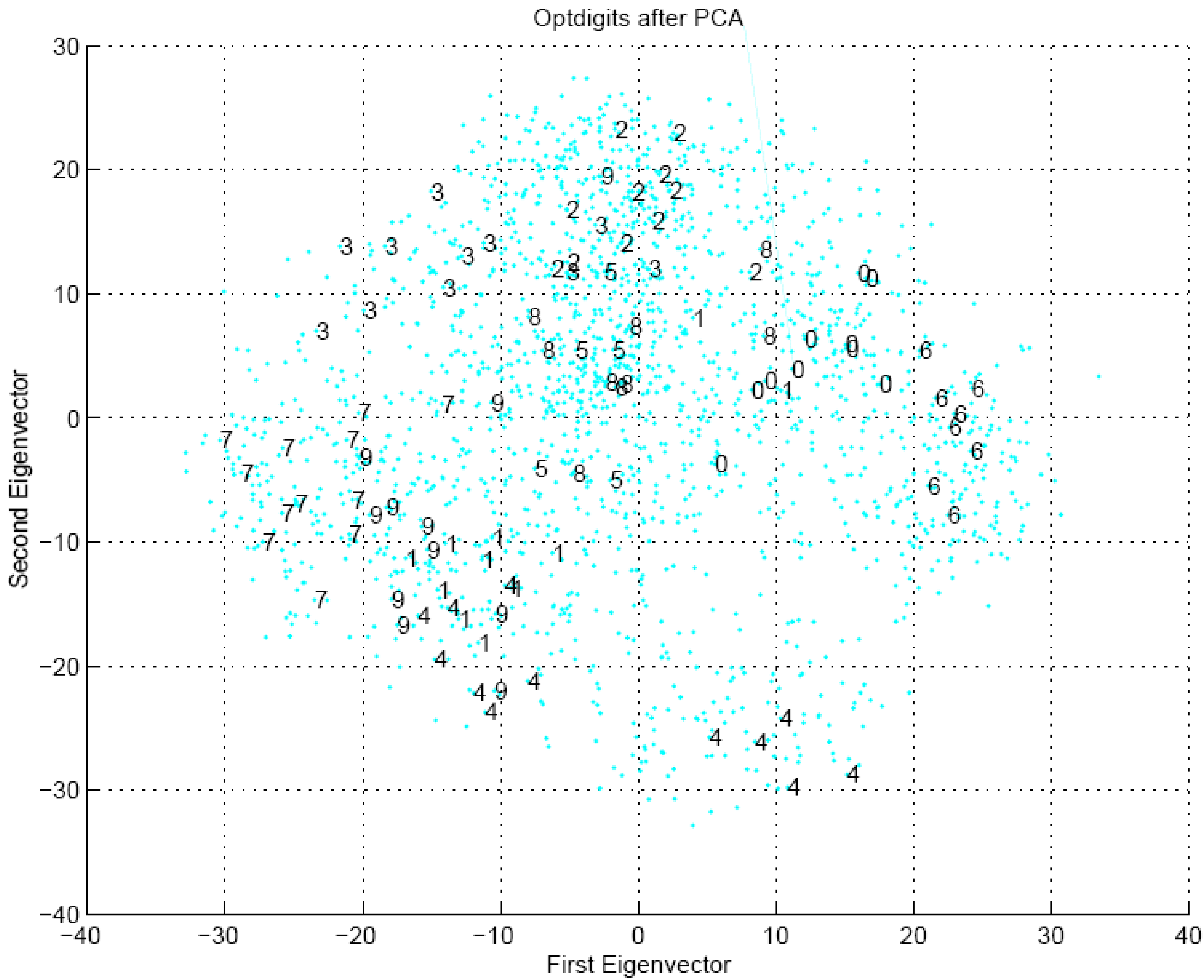
How to choose K ?

- Proportion of Variance (PoV) explained

$$\frac{\lambda_1 + \ldots + \lambda_K}{\lambda_1 + \ldots + \lambda_D} = \frac{\sum\limits_{i=1}^{K} \lambda_i}{\sum\limits_{i=1}^{D} \lambda_i}, \quad K \leq D$$

  when λ_i are sorted in descending order
- Typically, stop at PoV>0.9
- Scree graph plots of PoV vs. *k*: stop at “elbow”


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


Optdigits after PCA
Second Eigenvector
First Eigenvector

Principal Components Analysis (PCA)

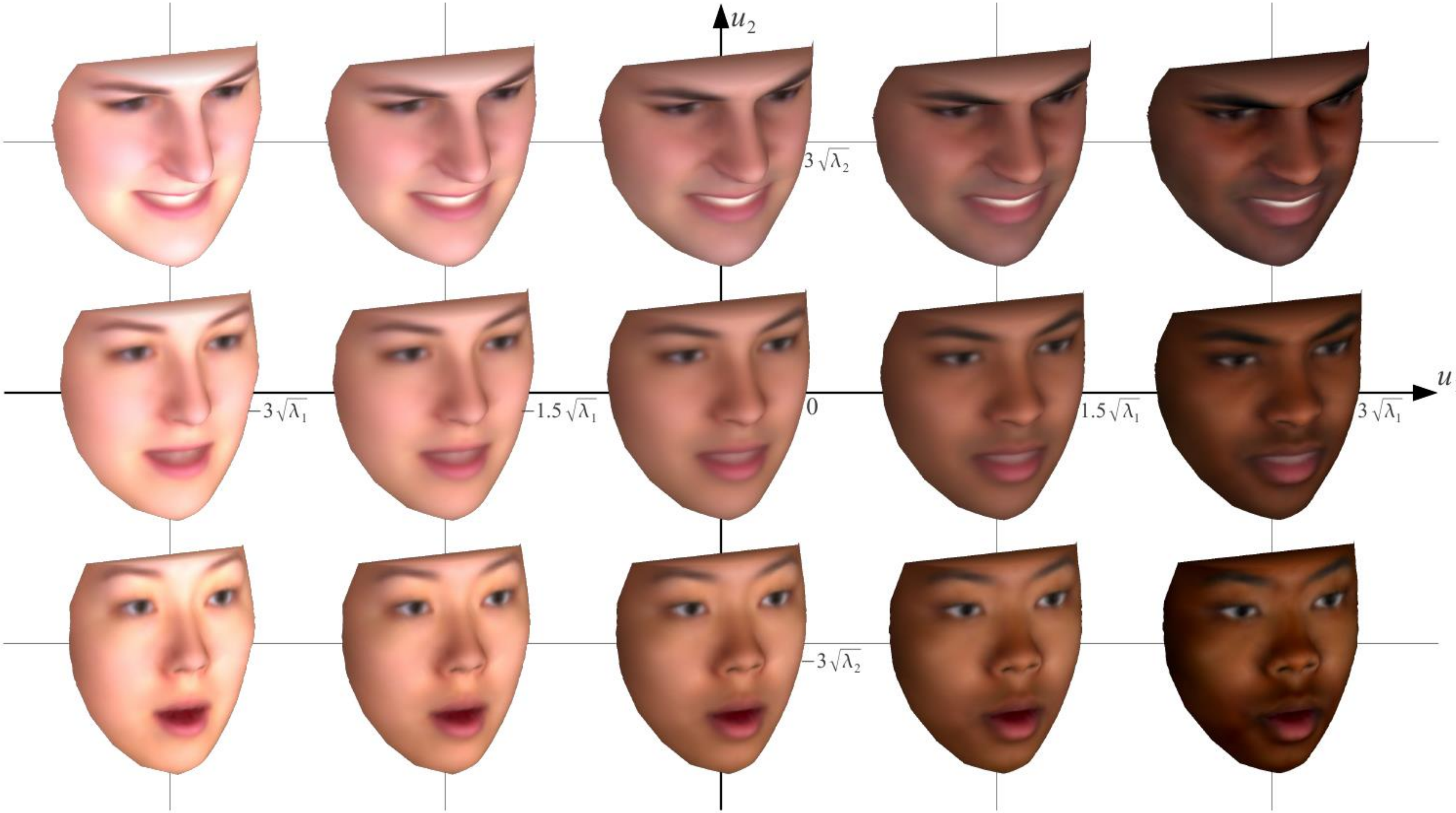
Properties

- Unsupervised, automatic
- Linear combination of input variables
- Preprocessing is crucial (mean-free)
- Extensions:
  - kernel PCA: enables nonlinear data
  - Incremental PCA: PCA on batches

Applications

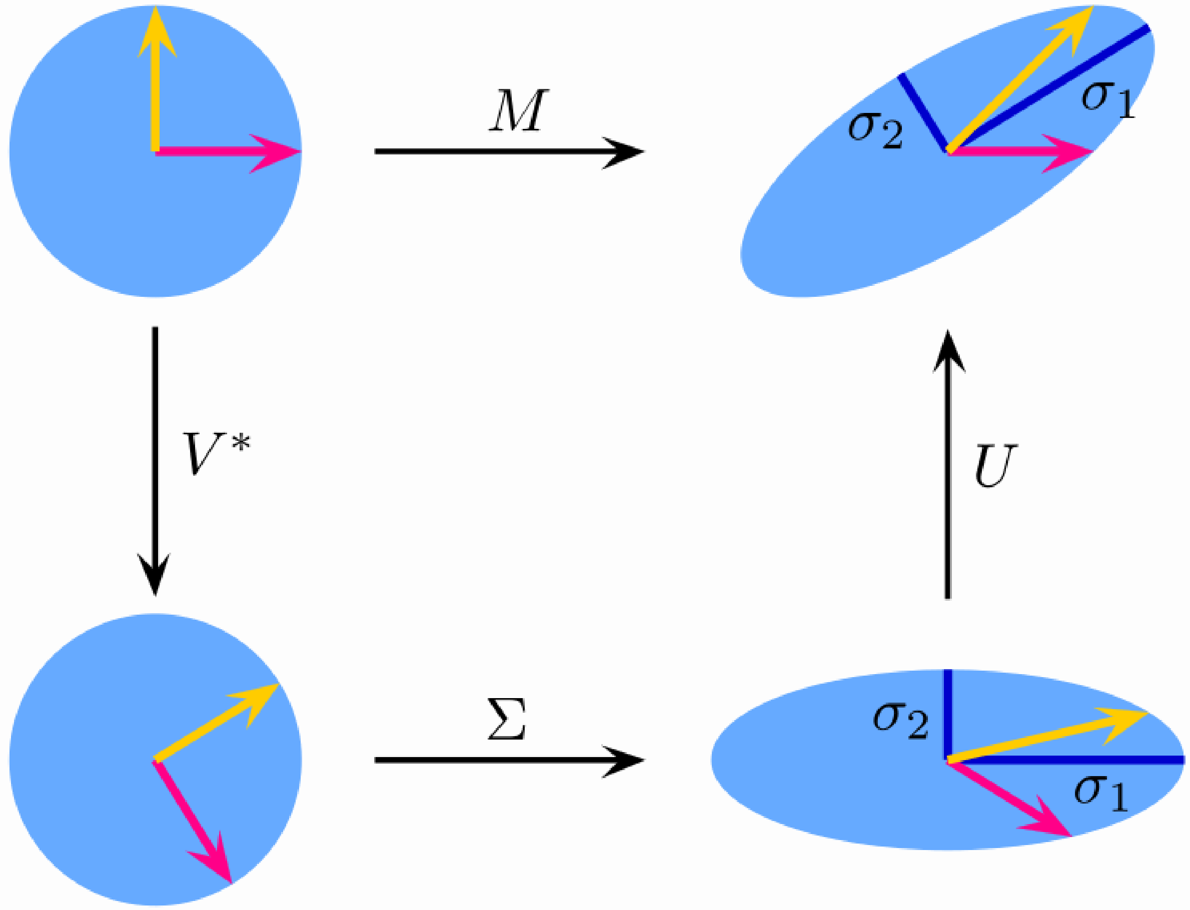
- Dimension Reduction
- Reconstruction
- …

Principal Components Analysis (PCA)



"Dense point-to-point correspondences between 3D faces using parametric remeshing for constructing 3D Morphable Models", Kaiser, et al., 2011

Singular Value Decomposition (SVD)



$$M = U \cdot \Sigma \cdot V^*$$

https://en.wikipedia.org/wiki/Singular_value_decomposition

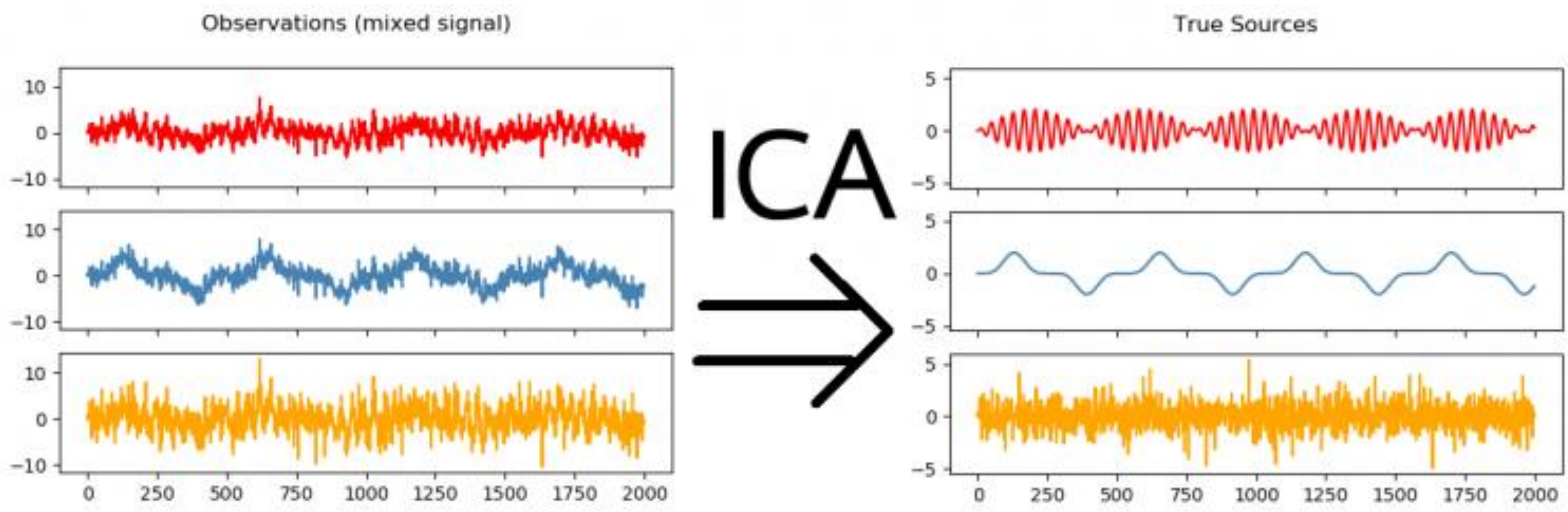
Singular Value Decomposition (SVD)

- $\boldsymbol{X}$ is NxD
- Singular value decomposition: $\boldsymbol{X}=\boldsymbol{VAW}^T$
  - $\boldsymbol{V}$ is NxN contains the eigenvectors of $\boldsymbol{XX}^T$
  - $\boldsymbol{W}$ is DxD contains the eigenvectors of $\boldsymbol{X}^T\boldsymbol{X}$
  - $\mathbf{A}$ is NxD contains singular values on its first K diagonal
- $\boldsymbol{X} = \boldsymbol{v}_1 a_1 \boldsymbol{w}_1^T + \ldots + \boldsymbol{v}_K a_K \boldsymbol{w}_K^T$ where K is the rank of $\boldsymbol{X}$
- ***Attention:*** *sign ambiguity!*
  $\boldsymbol{v}_k a_k \boldsymbol{w}_k^T = (-\boldsymbol{v}_k) a_k (-\boldsymbol{w}_k)^T$

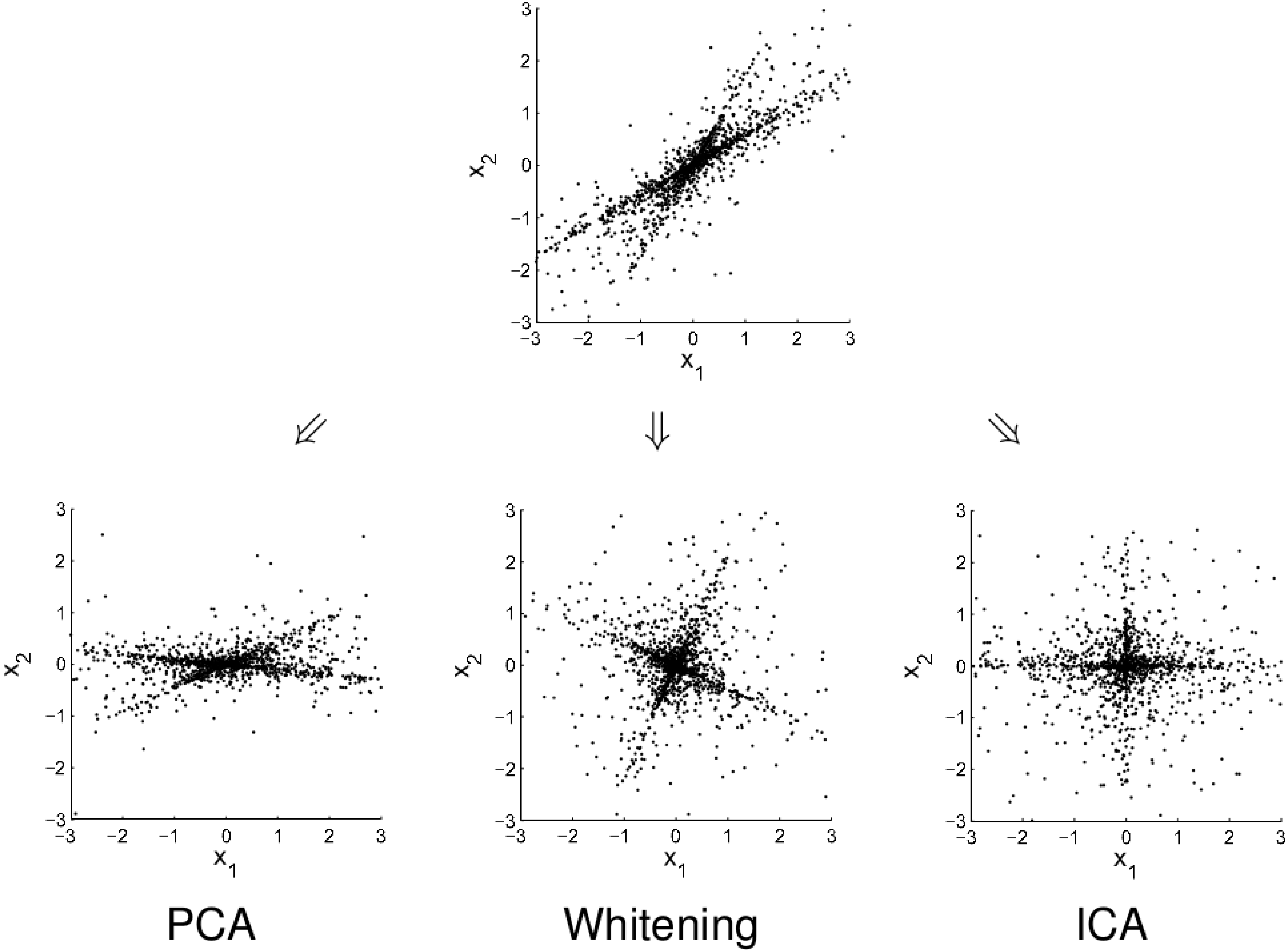
Independent Component Analysis (ICA)

- PCA gives uncorrelated components (features)
- ICA gives independent components
  - **Given:** mixture of signals $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}$
    Goal: find the original source signals
    assume they are independent
  - e.g. Cocktail party problem, source separation
  - problem: no unique solution
  - Preprocessing usually: PCA and Whitening

Independent Component Analysis (ICA)



https://team.inria.fr/parietal/research/statistical-and-machine-learning-methods-for-large-scale-data/faster-independent-component-analysis-for-real-data/



x1
x2
x1
x2
PCA
x1
x2
Whitening
x1
x2
ICA

Factor Analysis (FA)

- Find a small number of **factors *z*,** which when combined generate **x** :

$$x_d - \mu_d = v_{d1} z_1 + v_{d2} z_2 \ldots + v_{dK} z_K + \epsilon_d, \; d = 1, \ldots, D$$

- v_{ij} are the **factor loadings**
- z_i, $j = 1, \ldots, K$ are the **latent factors** with

$$E[z_i] = 0, \; \mathrm{Var}(z_i) = 1, \; \mathrm{Cov}(z_i, z_j) = 0, \; i \neq j$$

$$\mathrm{Cov}(\boldsymbol{z}) = \boldsymbol{I}$$

- ϵ_i are the **noise sources**

$$E[\epsilon_i] = \Psi_i,$$

$$\mathrm{Cov}(\epsilon_i, \epsilon_j) = 0,$$

$$\mathrm{Cov}(\epsilon_i, z_j) = 0, \; i \neq j$$

$$\mathrm{Cov}(\boldsymbol{\epsilon}) = \boldsymbol{\Psi}$$

$$\boldsymbol{\Psi} = \mathrm{diag}(\psi_1, \ldots, \psi_D) = \begin{pmatrix} \psi_1 & 0 & \ldots & 0 \\ 0 & \ddots & 0 & 0 \\ \vdots & & \ddots & 0 \\ 0 & \ldots & 0 & \psi_D \end{pmatrix}$$

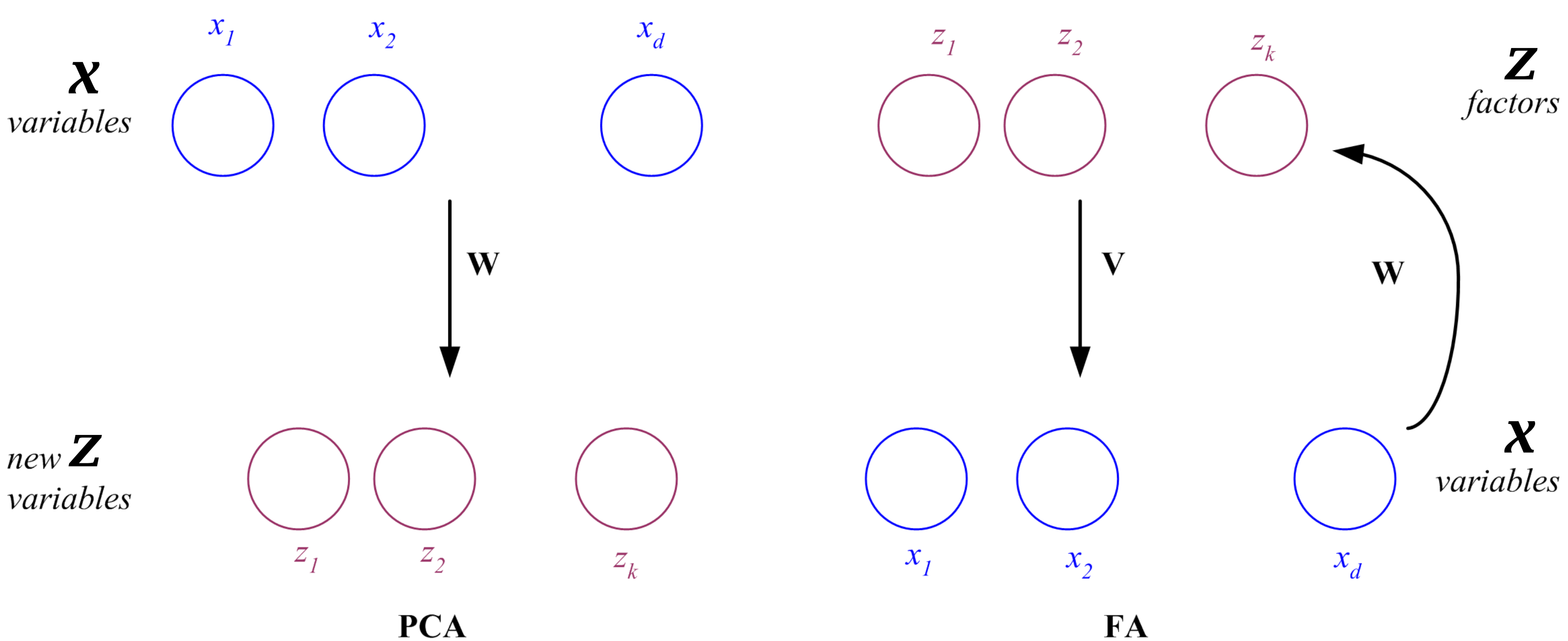
PCA vs. FA

- **PCA** is a linear combination of variables

  PCA From $\mathbf{x}$ to $\mathbf{z}$ $\mathbf{z} = \mathbf{W}^T(\mathbf{x} - \boldsymbol{\mu})$

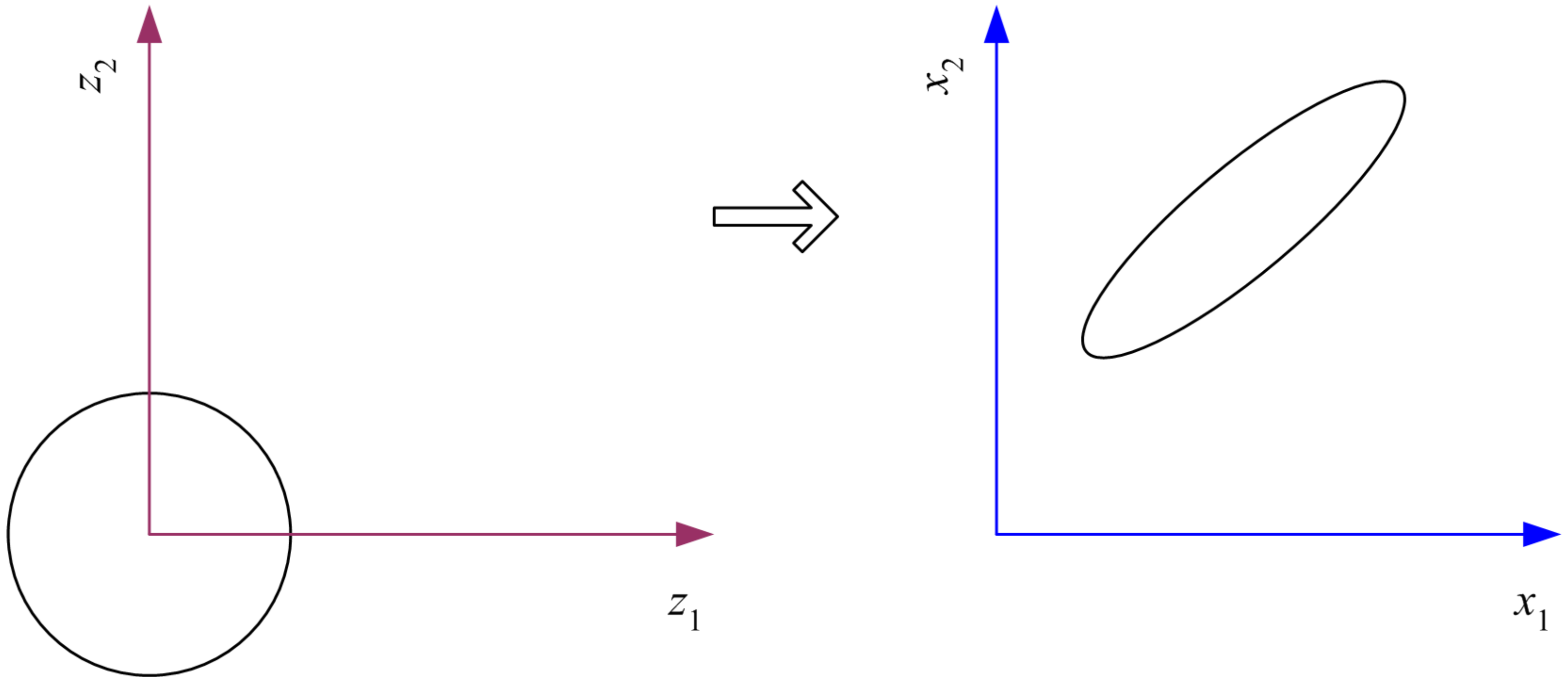
- **Factor Analysis** is a measurement model of a latent variable.

  FA From $\mathbf{z}$ to $\mathbf{x}$ $\mathbf{x} - \boldsymbol{\mu} = \mathbf{V}\mathbf{z} + \boldsymbol{\varepsilon}$



Factor Analysis (FA)

In FA, factors z_i are stretched, rotated and translated to generate **x**



Canonical Correlation Analysis (CCA)

Consider data given for T frames:

- Video (image sequence) per person, e.g. dimension D=Tx100x100x3 (width x height x RGB)
- Audio per person, dimension E=Tx1

Problem:

- Length T is the same, but **dimension per frame differ**

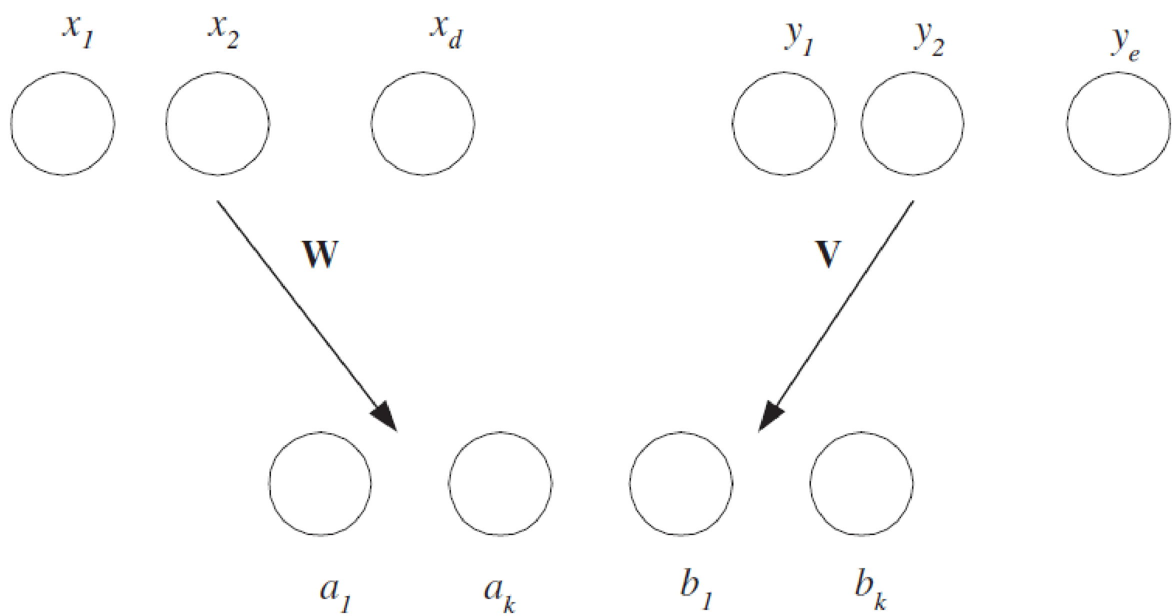
Goal:

- Reduce the data to a **joint dimension**

Canonical Correlation Analysis (CCA)

x and ***y*** may be two different views or modalities

CCA does a joint mapping



Canonical Correlation Analysis (CCA)

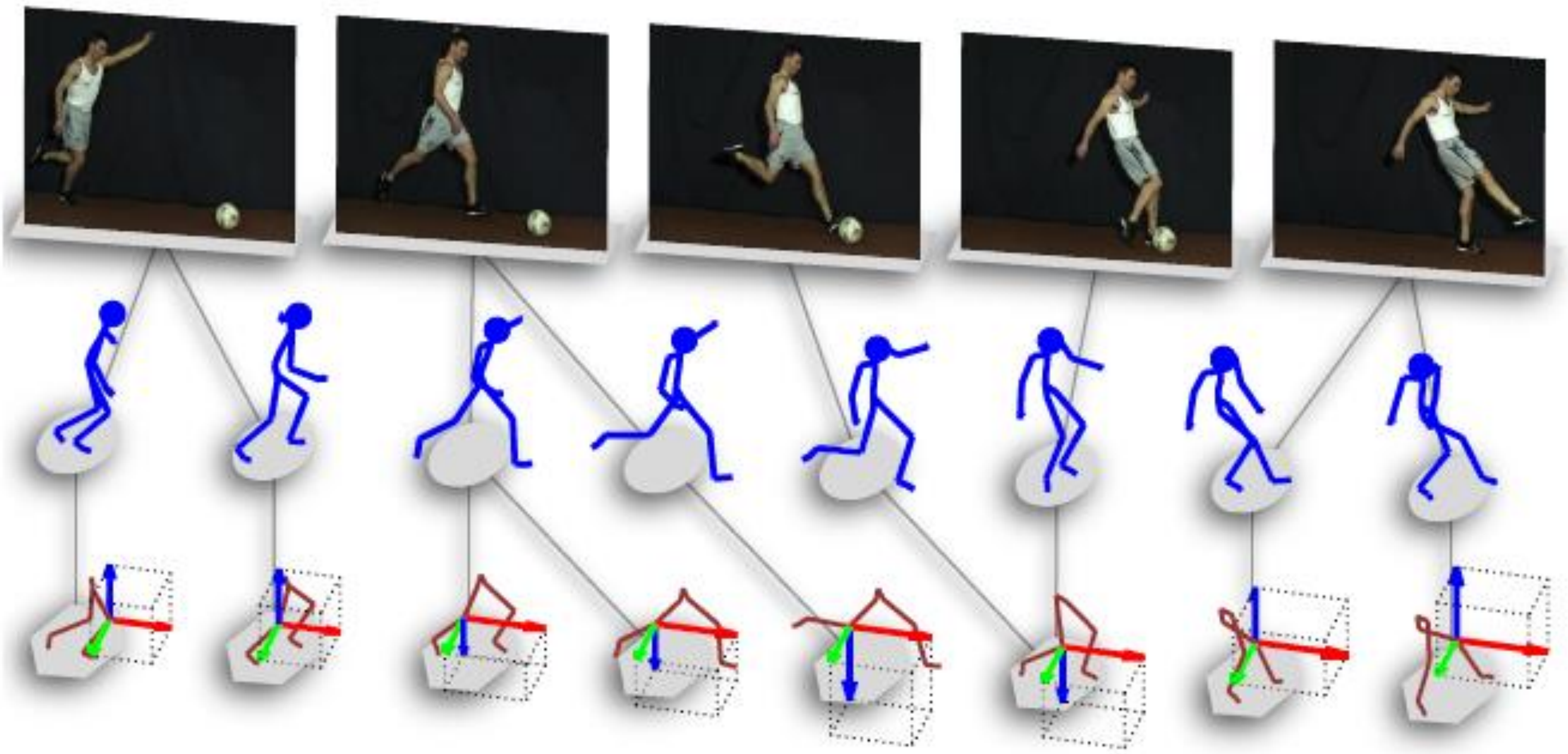


image from:
„Generalized Canonical Time Warping“, Zhou et al., 2013

Canonical Correlation Analysis (CCA)

- $\boldsymbol{X}=\{\boldsymbol{x}_k, \boldsymbol{y}_k\}$: two sets of variables $\boldsymbol{x}$ and $\boldsymbol{y}$
- We want to find two projections $\boldsymbol{w}$ and $\boldsymbol{v}$
  - when $\boldsymbol{x}$ is projected along $\boldsymbol{w}$
  - and $\boldsymbol{y}$ is projected along $\boldsymbol{v}$
- the correlation is maximized:

$$\begin{aligned}\rho &= \mathrm{Corr}(\boldsymbol{w}^T\boldsymbol{x}, \boldsymbol{v}^T\boldsymbol{y}) = \frac{\mathrm{Cov}(\boldsymbol{w}^T\boldsymbol{x}, \boldsymbol{v}^T\boldsymbol{y})}{\sqrt{\mathrm{Var}(\boldsymbol{w}^T\boldsymbol{x})}\sqrt{\mathrm{Var}(\boldsymbol{v}^T\boldsymbol{y})}} \\ &= \frac{\boldsymbol{w}^T\mathrm{Cov}(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{v}}{\sqrt{\boldsymbol{w}^T\mathrm{Var}(\boldsymbol{x})\boldsymbol{w}}\sqrt{\boldsymbol{v}^T\mathrm{Var}(\boldsymbol{y})\boldsymbol{v}}} = \frac{\boldsymbol{w}^T\mathbf{S}_{xy}\boldsymbol{v}}{\sqrt{\boldsymbol{w}^T\mathbf{S}_{xx}\boldsymbol{w}}\sqrt{\boldsymbol{v}^T\mathbf{S}_{yy}\boldsymbol{v}}}\end{aligned}$$

https://en.wikipedia.org/wiki/Canonical_correlation

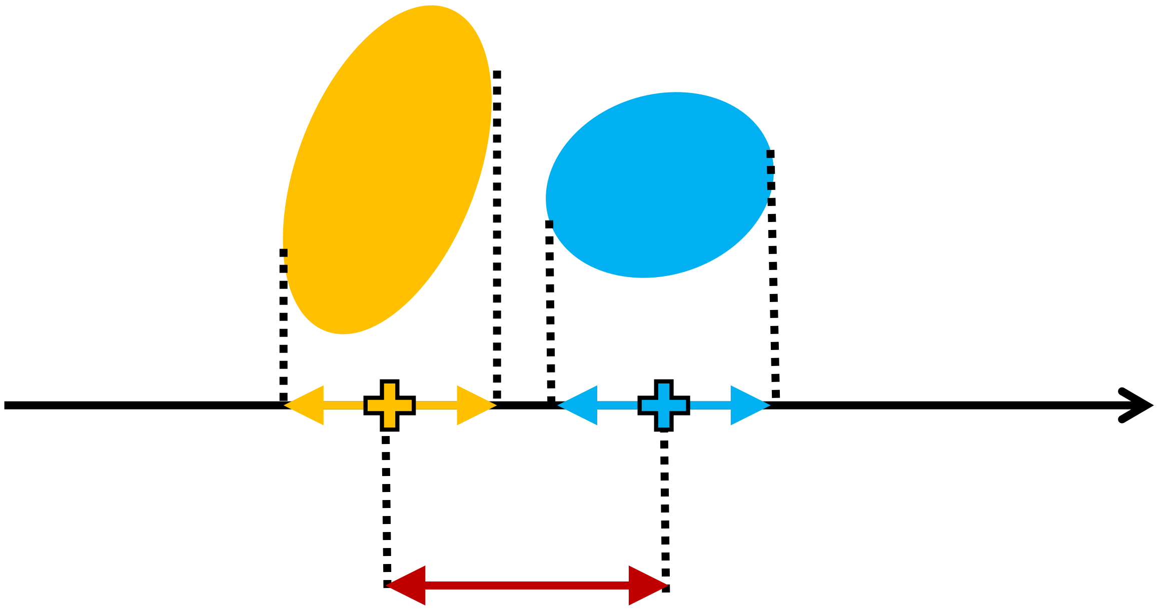
Summary: unsupervised

- All previously presented methods are **unsupervised**: no class information is required or used
  - PCA, FA, ICA: linear
  - CCA
- Now: Linear Discriminant Analysis (LDA)
  - Uses class labels = **supervised**

Linear Discriminant Analysis (LDA)

Find a low-dimensional space, such that classes are well-separated by:

- **Maximize** distance between the means of projected classes 
- **Minimize variance** for each projected class



Linear Discriminant Analysis (LDA)

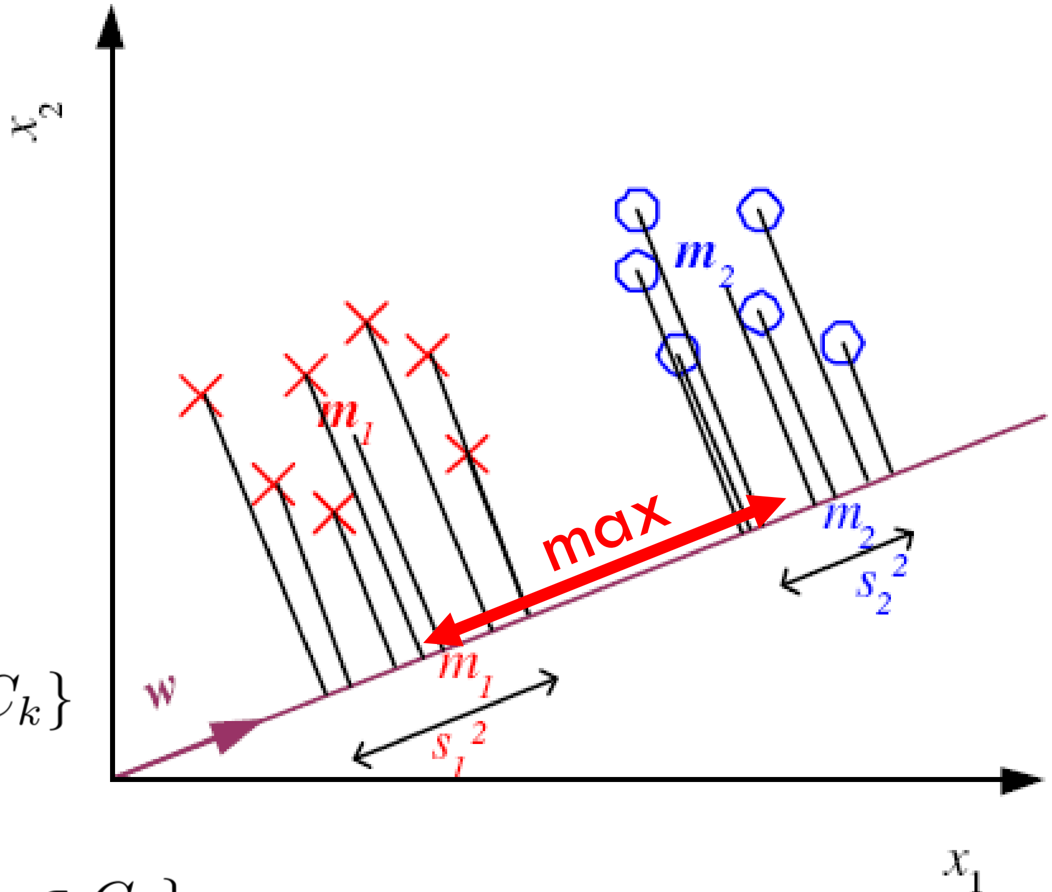
Find ***w*** that maximizes

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

$$N_k = \sum_{n=1}^{N} \mathbb{1}\{\boldsymbol{x}_n \in C_k\}$$

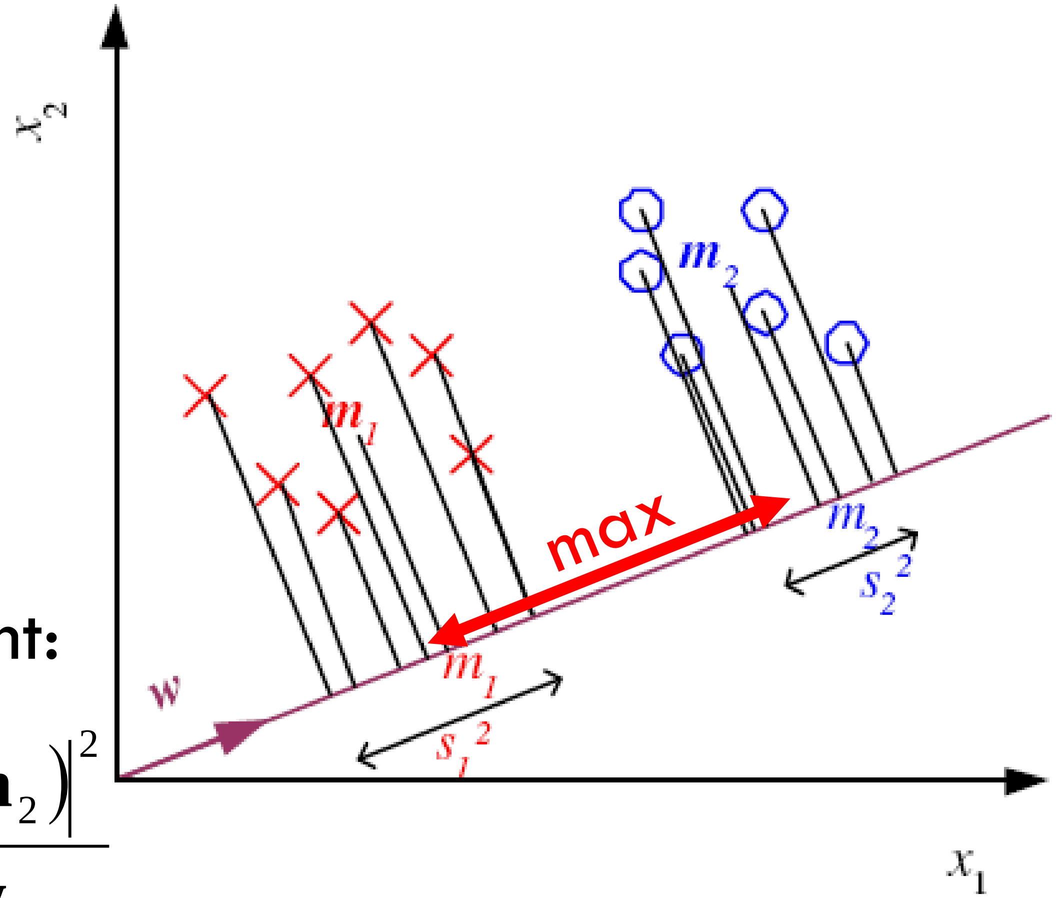
$$m_k = \frac{1}{N_k} \sum_{n=1}^{N} \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n \mathbb{1}\{\boldsymbol{x}_n \in C_k\}$$

$$s_k^2 = \sum_{n=1}^{N} (\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_n - m_k)^2 \mathbb{1}\{\boldsymbol{x}_n \in C_k\}$$



Linear Discriminant Analysis (LDA)

- Maximize
  between-class scatter
- Minimize
  within-class scatter



Fisher's Linear Discriminant:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_B \mathbf{w}}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}} = \frac{\left|\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)\right|^2}{\mathbf{w}^T \mathbf{S}_W \mathbf{w}}$$

$$\mathbf{w} = c \cdot \mathbf{S}_W^{-1}(\mathbf{m}_1 - \mathbf{m}_2)$$

LDA with K>2 Classes

- Within-class scatter:

$$\mathbf{S}_W = \sum_{i=1}^{K} \mathbf{S}_i \qquad \mathbf{S}_i = \sum_t r_i^t \left(\mathbf{x}^t - \mathbf{m}_i\right)\left(\mathbf{x}^t - \mathbf{m}_i\right)^T$$

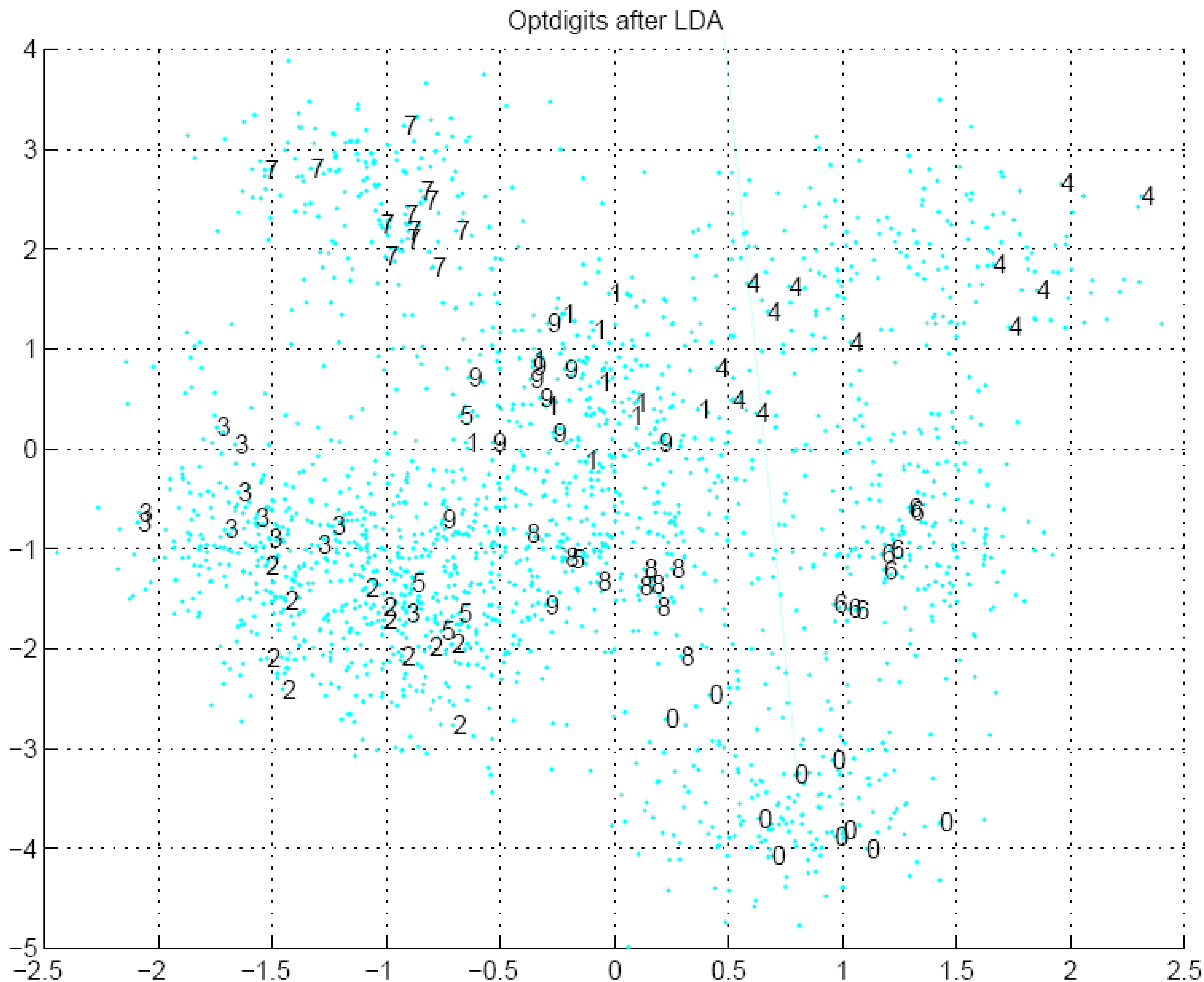
- Between-class scatter:

$$\mathbf{S}_B = \sum_{i=1}^{K} N_i \left(\mathbf{m}_i - \mathbf{m}\right)\left(\mathbf{m}_i - \mathbf{m}\right)^T \qquad \mathbf{m} = \frac{1}{K}\sum_{i=1}^{K} \mathbf{m}_i$$

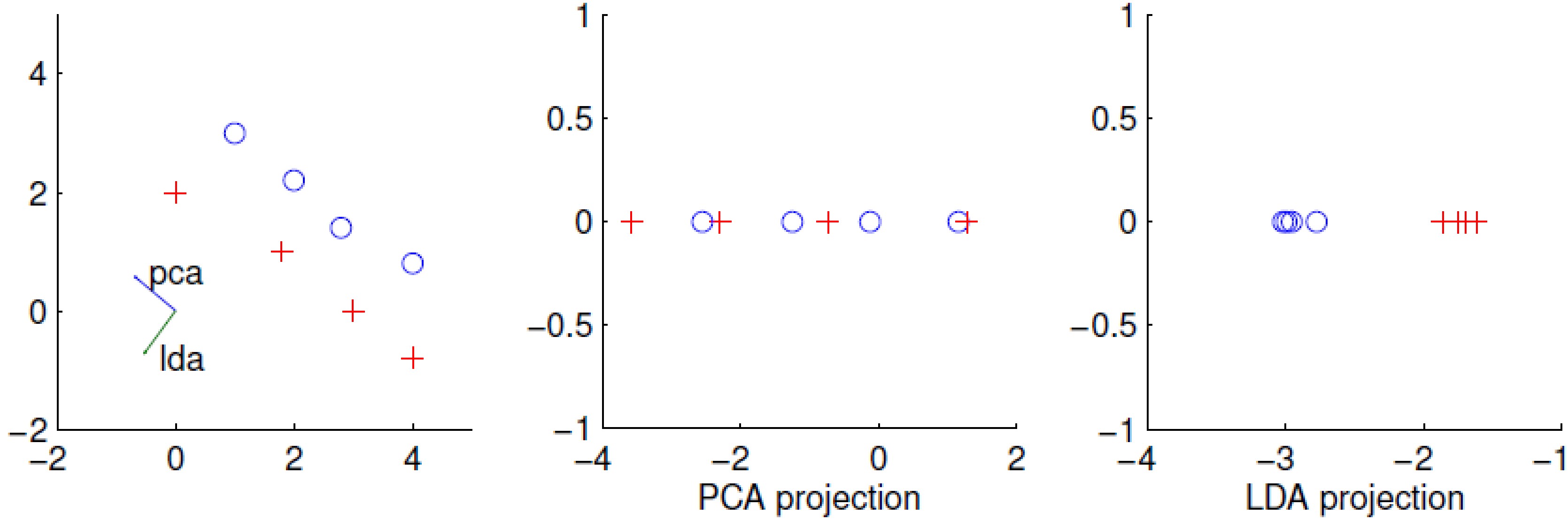
- Find **W** that max

$$J(\mathbf{W}) = \frac{\left|\mathbf{W}^T \mathbf{S}_B \mathbf{W}\right|}{\left|\mathbf{W}^T \mathbf{S}_W \mathbf{W}\right|}$$

The largest eigenvectors of $\mathbf{S}_W^{-1}\mathbf{S}_B$; maximum rank of K-1


Optdigits after LDA
4
3
2
1
0
−1
−2
−3
−4
−5
−2.5
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
2.5

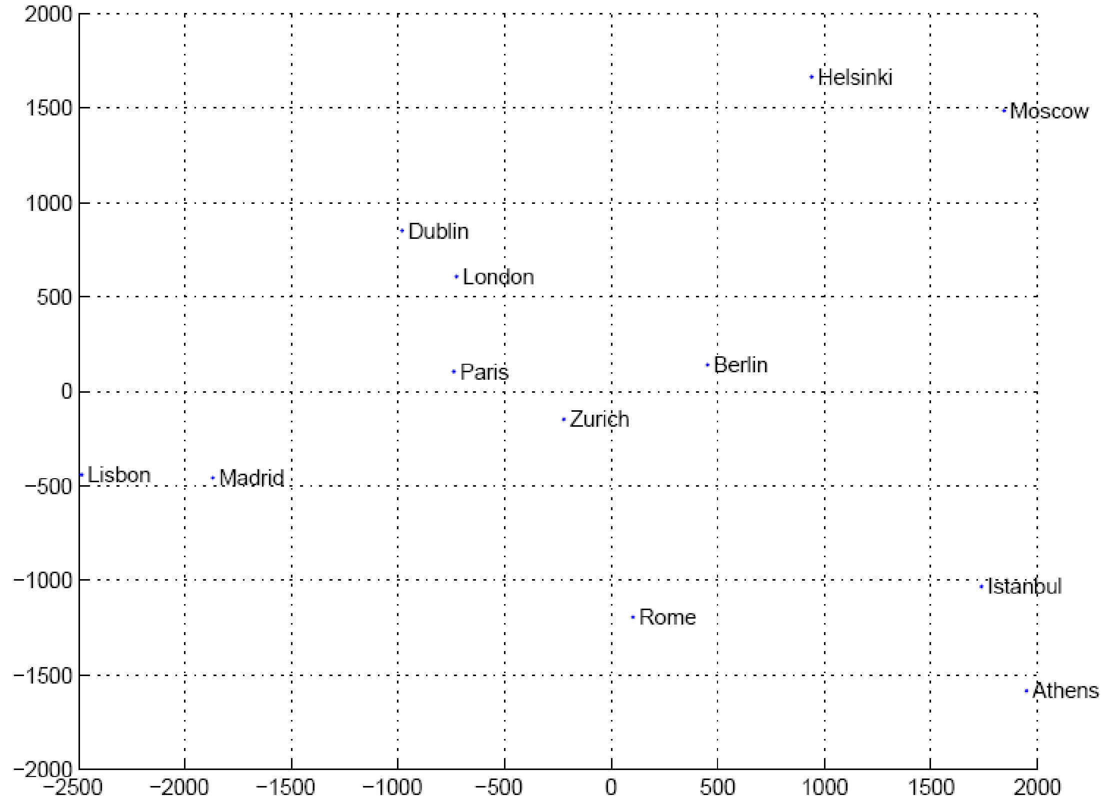
PCA vs LDA



Multidimensional Scaling (MDS)

Given: distances between 3D points on a sphere, e.g. cities

Goal: projection to 2D where output preserves distances



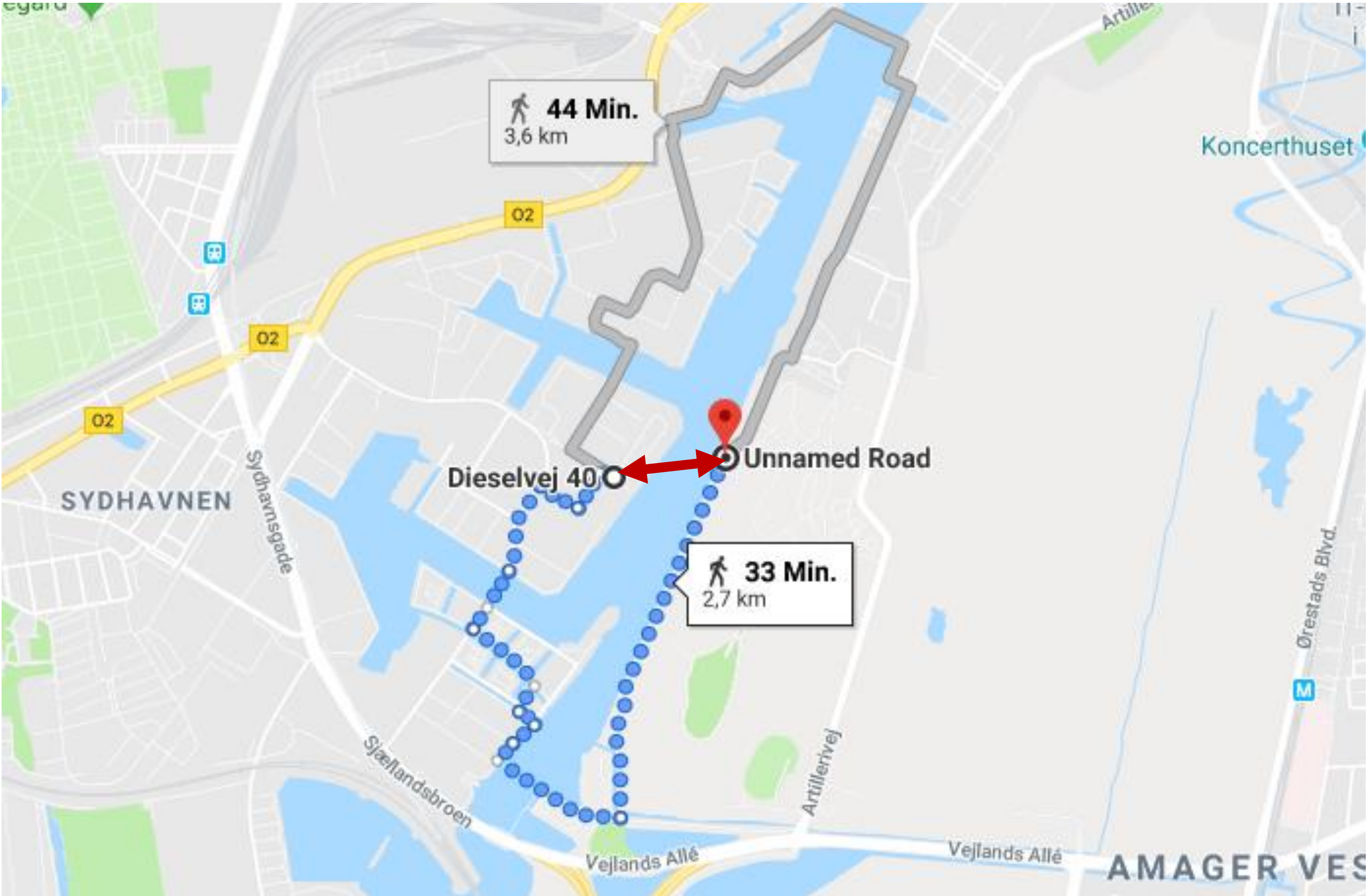
Map from CIA – The World Factbook: http://www.cia.gov/

Multidimensional Scaling (MDS)

- Given pairwise distances between N points $\mathbf{x}_i \in \mathbb{R}^D$
  d_{ij} : dist($\mathbf{x}_i$, $\mathbf{x}_j$) $i,j = 1,\ldots,N$
- Find $\mathbf{z}_i \in \mathbb{R}^M,\ M < D$ of lower dimension
  δ_{ij} : dist($\mathbf{z}_i$, $\mathbf{z}_j$) $i,j = 1,\ldots,N$
- such that distances are preserved: $d_{ij} \approx \delta_{ij}$
  - Find $\mathbf{z}_i$ $\min \frac{\|\delta_{ij} - d_{ij}\|^2}{\delta_{ij}}$

    => Solve by eigenvalue problem on $\mathbf{B}=\mathbf{X}\mathbf{X}^{\mathbf{T}}$
  - Find regression function $\mathbf{g}$ with parameters ϑ:
    $\mathbf{z} = \mathbf{g}(\mathbf{x} \mid \vartheta) = \mathbf{W}^T\mathbf{x}$
- Comparable to PCA

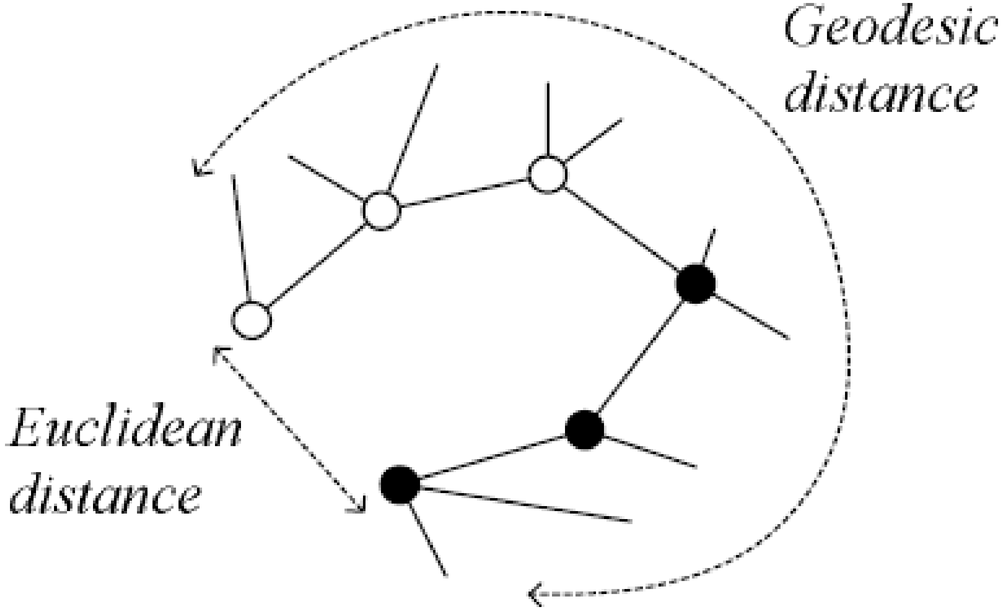
Geodesic vs. Euclidean distance



Isomap

Given: $\boldsymbol{x}_k \in \mathbb{R}^D,\ k = 1, \ldots, N$

Idea: approximate geodesic distance by local Euclidean distances



Isomap

1) For each point get nearest neighbors by either:
(a) "K nearest" or
(b) "inside radius R"

$$\boldsymbol{x}_k \in \mathbb{R}^D, \ k = 1, \ldots, N$$

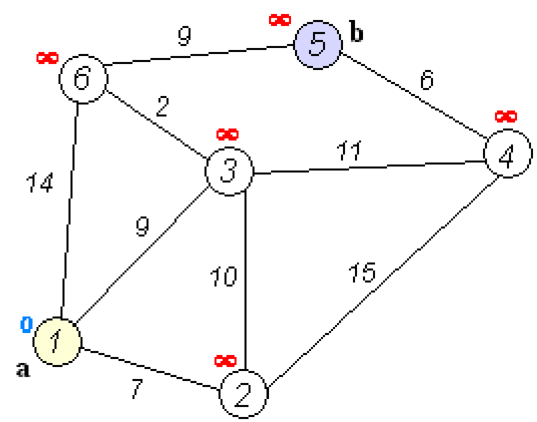
$$w_{ij} = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2$$

2) Build neighbor graph:
$\boldsymbol{x}_i, \boldsymbol{x}_j$ are connected if (a) or (b)

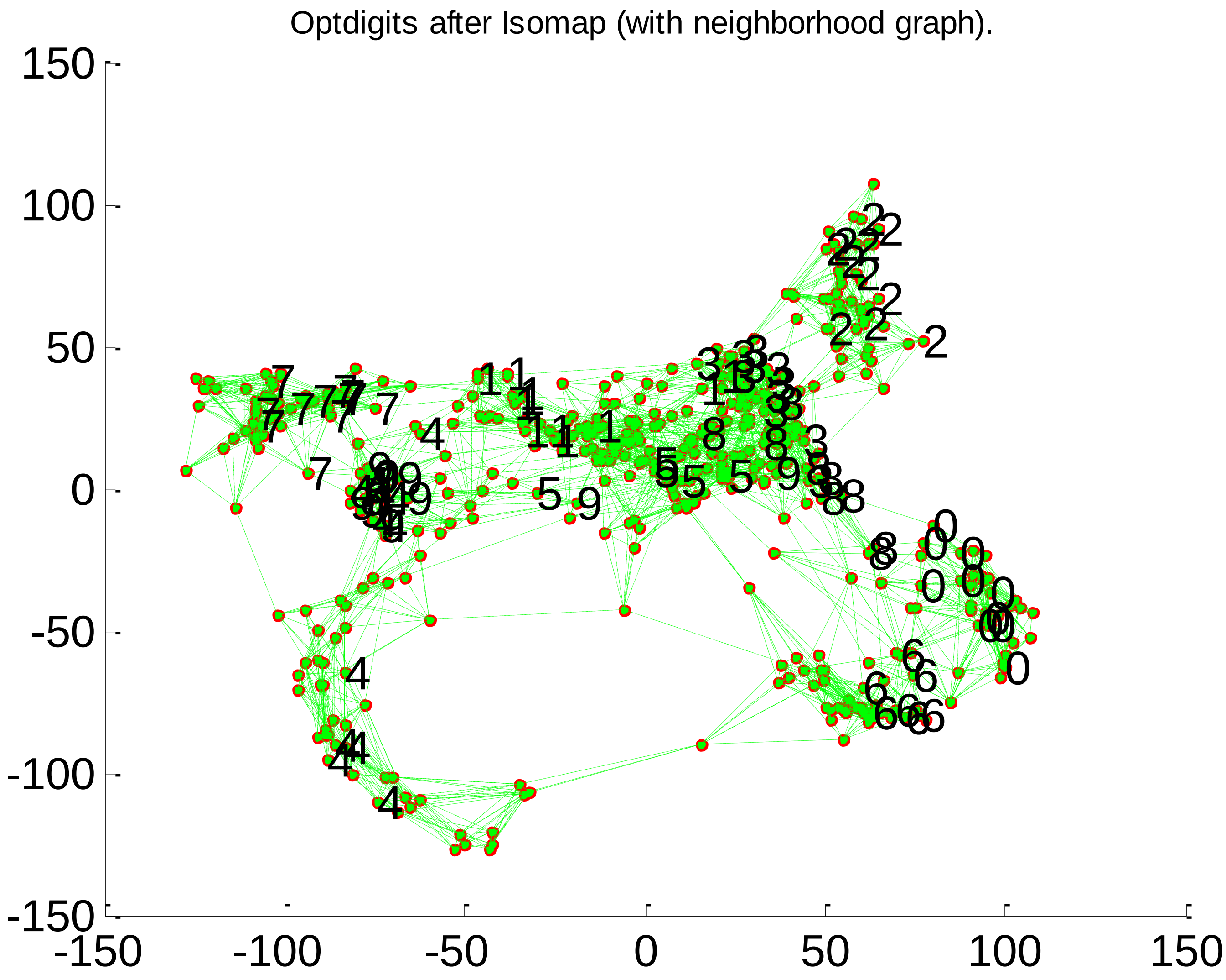
3) Compute shortest path between pairs of points $\boldsymbol{x}_i, \boldsymbol{x}_j$
e.g. by Dijkstra, distances as d_{ij}

4) Apply MDS (Multidimensional Scaling) on the distances

$$d_{ij} \in \boldsymbol{D} \in \mathbb{R}^{N \times N}$$



https://commons.wikimedia.org/wiki/File:Dijkstra_Animation.gif
Ibmua



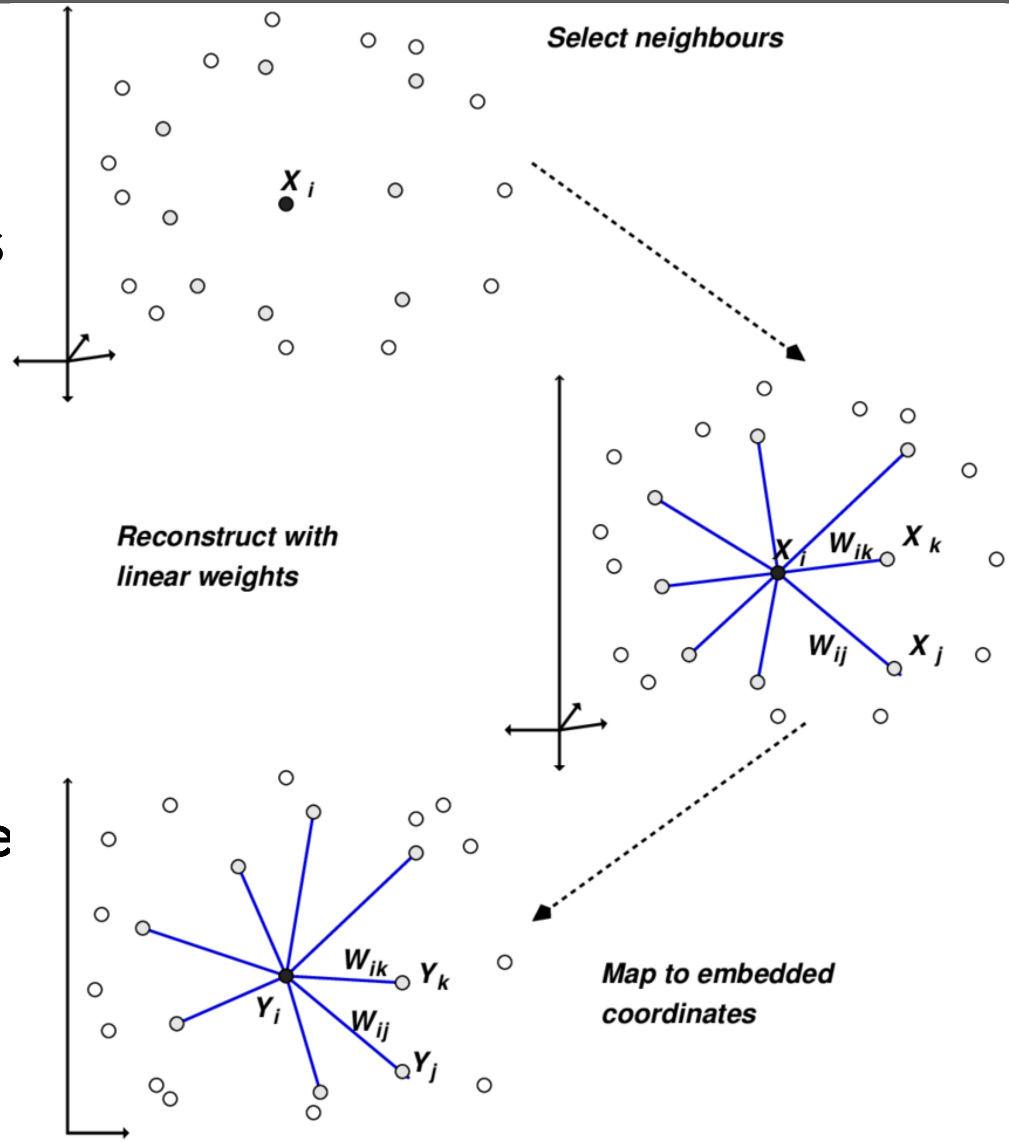
Matlab source from http://web.mit.edu/cocosci/isomap/isomap.html

Locally Linear Embedding (LLE)

- Similar to Isomap, but faster, because it uses sparse matrix computations

Idea:

- represent each point by a weighted sum of its neighbors, i.e. search **W**
- Estimate lower dimensional **representations Z** using the neighbor weights W



Source: https://www.researchgate.net/figure/Overview-of-the-steps-involved-in-locally-linear-embedding_fig9_230595014

Locally Linear Embedding (LLE)

Goal: $\boldsymbol{x}_i \in \mathbb{R}^D \rightsquigarrow \boldsymbol{z}_i \in \mathbb{R}^E,\ E < D$

1) For each point $\boldsymbol{x}_i \in \mathbb{R}^D$
   get K nearest neighbors $\boldsymbol{x}_{i,k} \in \mathbb{R}^D,\ k = 1, \ldots, K$

2) **Estimate weights** to reconstruct $\boldsymbol{x}_i$ by its neighbors

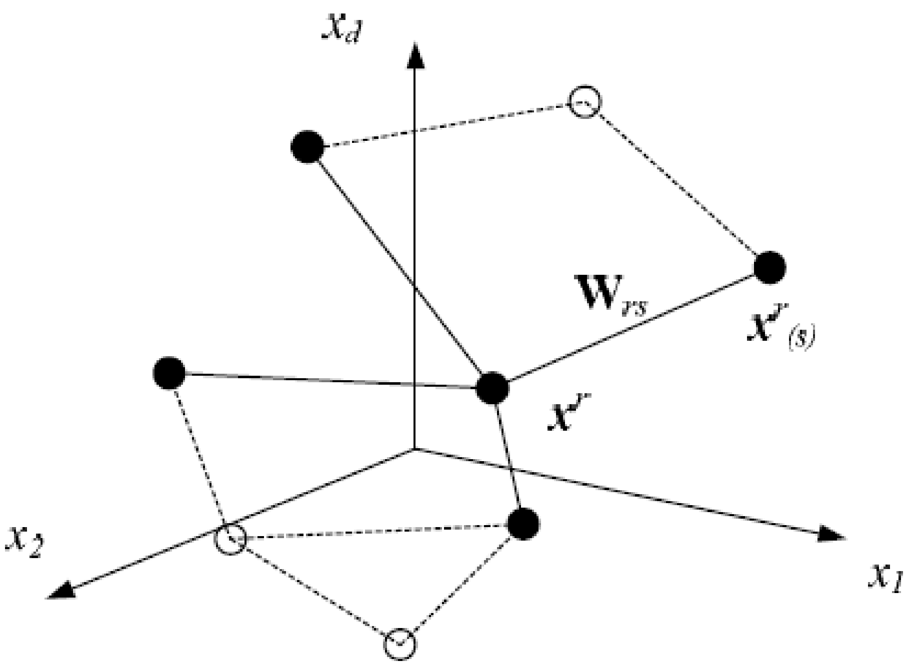
$$f(\boldsymbol{W}) = \sum_{i=1}^{N} \left\| \boldsymbol{x}_i - \sum_{j=1}^{N} w_{ij} \boldsymbol{x}_j \right\|_2^2 \qquad \sum_{k=1}^{K} w_{ik} = 1 \quad w_{ii} = 0$$

   $w_{ij} = 0$ if $\boldsymbol{x}_j$ is not a neighbor of $\boldsymbol{x}_i$

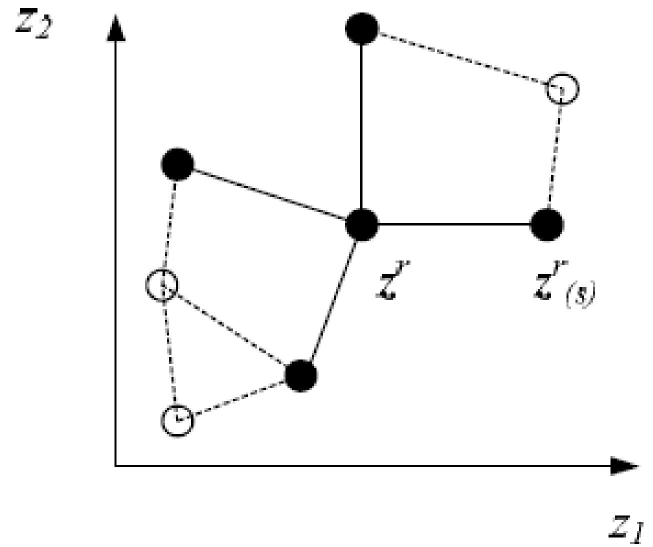
3) Given the weights **estimate new coordinates** $\boldsymbol{z}_i \in \mathbb{R}^E$
   called: **embedded coordinates** $E[\boldsymbol{z}_i] = 0,\ \mathrm{Cov}(\boldsymbol{z}) = \boldsymbol{I}$

$$g(\boldsymbol{Z}|\boldsymbol{W}) = \sum_{i=1}^{N} \left\| \boldsymbol{z}_i - \sum_{j=1}^{N} w_{ij} \boldsymbol{z}_j \right\|_2^2$$

Locally Linear Embedding (LLE)

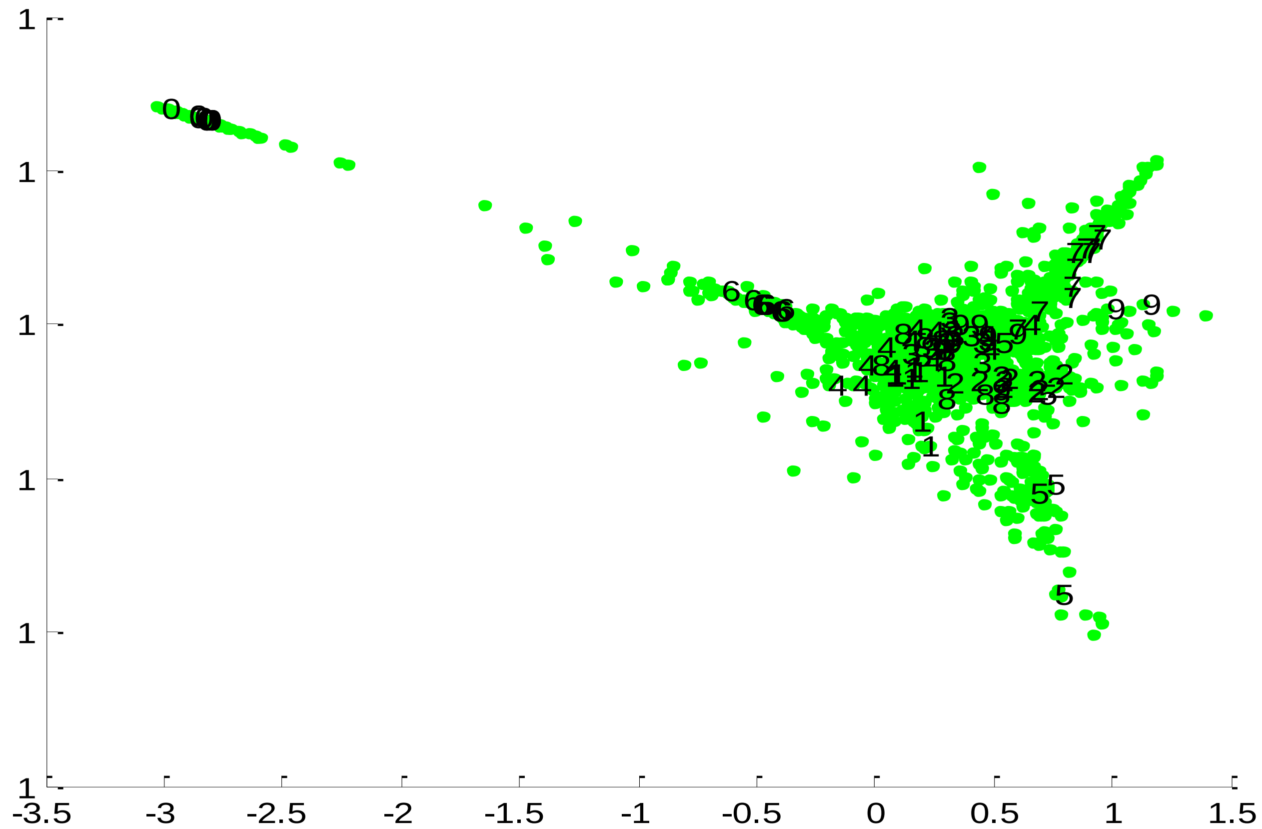


x space



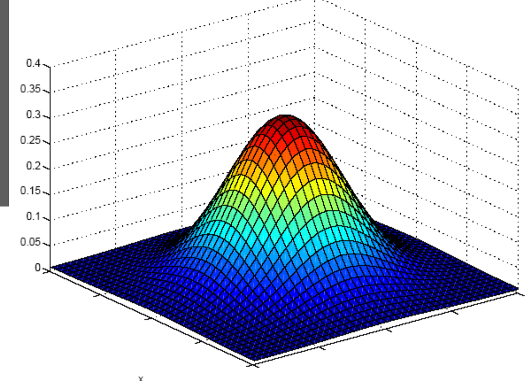
z space

LLE on Optdigits



Matlab source from http://www.cs.toronto.edu/~roweis/lle/code.html

Laplacian Eigenmaps



$$\boldsymbol{x}_i \in \mathbb{R}^D \rightsquigarrow \boldsymbol{z}_i \in \mathbb{R}^E, \; E < D$$

1) Create neighbor graph:

For each point get nearest neighbors and connect them

2) Define weights of connections as similarity values

$$w_{ij} = w_{ji} = \begin{cases} \exp\left[-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2}{2s}\right] & \text{, if points are connected} \\ 0 & \text{, else} \end{cases}$$

3) **Graph Laplacian:**

with diagonal matrix **D** elements: $d_{ii} = \sum_j w_{ij}$

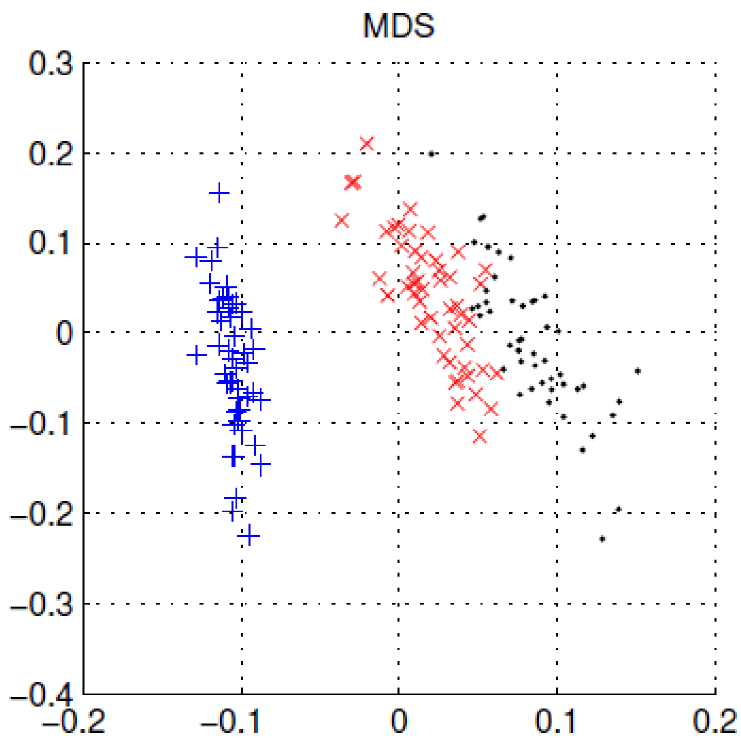
$$\min \boldsymbol{z}^{\mathrm{T}} \boldsymbol{L} \boldsymbol{z}, \quad \|\boldsymbol{z}\| = 1$$

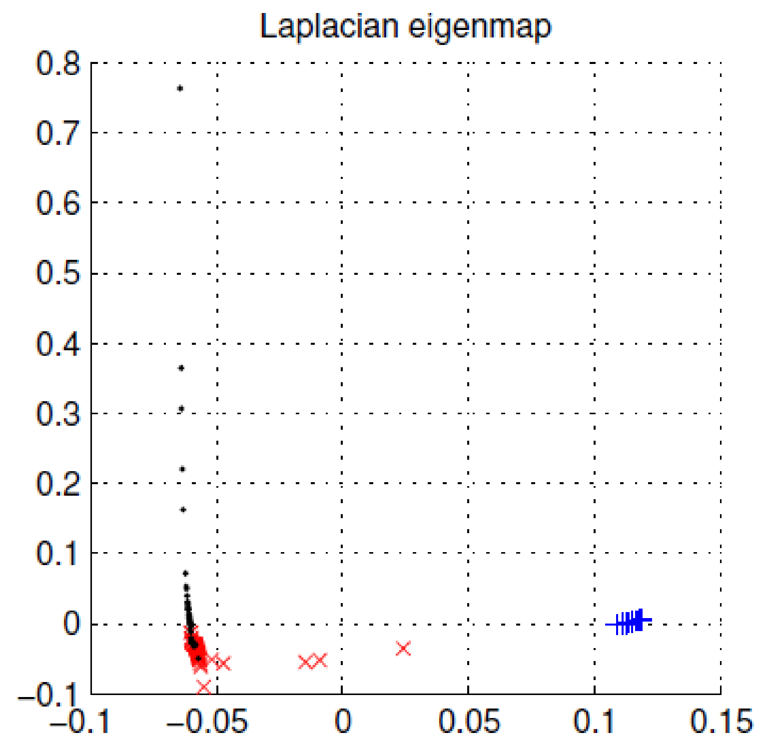
$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$$

$$\boldsymbol{L}\boldsymbol{z}_k = \lambda_k \boldsymbol{z}_k$$

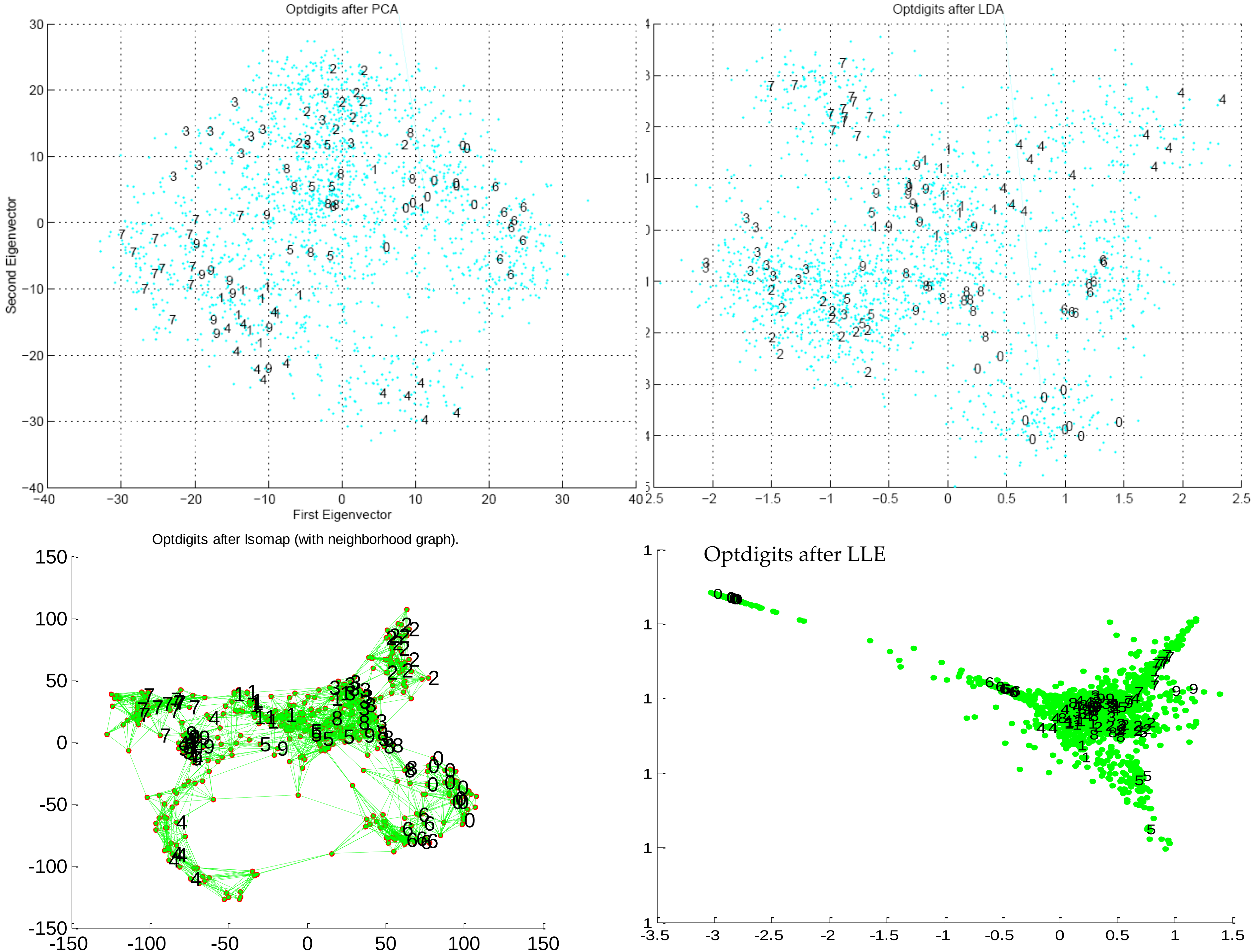
- solve eigenvalue problem
- keep smallest EV

Laplacian Eigenmaps on Iris



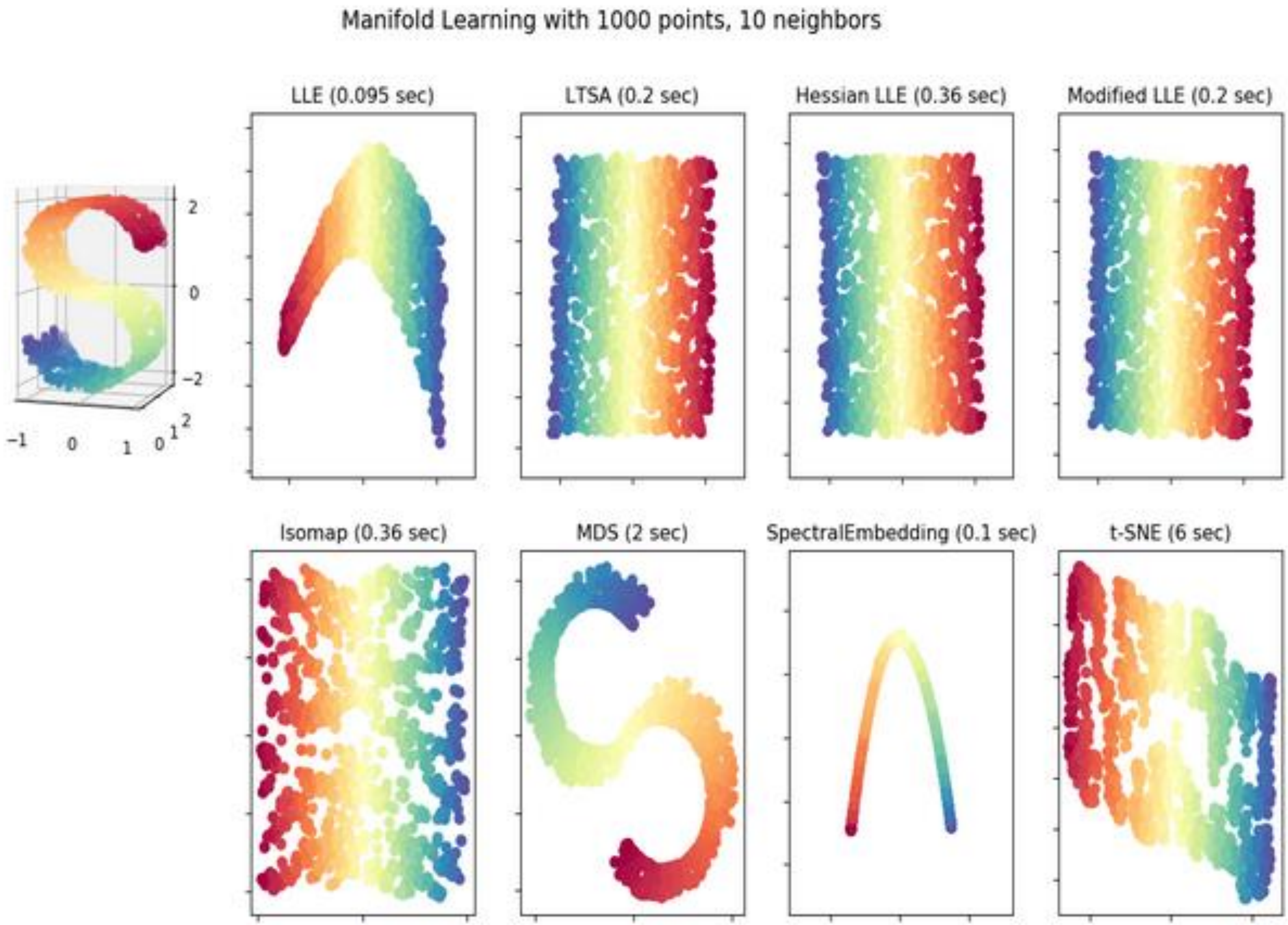


Spectral clustering (chapter 7)


Optdigits after PCA
First Eigenvector
Second Eigenvector
Optdigits after LDA
Optdigits after Isomap (with neighborhood graph).
Optdigits after LLE

Additionally…

https://scikit-learn.org/stable/modules/manifold.html



Overview

Method	supervised	local	nonlinear	Diff. dim.
PCA	✗	✗	✗	✗
LDA	✓	✗	✗	✗
MDS	✗	✗	(✓)	✗
CCA	✗	✗	✗	✓
Isomap	✗	✓	✓	✗
LLE	✗	✓	✓	✗
Laplacian Eigenmaps	✗	✓	✓	✗

- Supervised: uses class labels
- Local: uses local information
- Nonlinear: can handle nonlinear data
- Diff. dim. = different dimensions for each input

Summary

- Feature Selection: selects dimensions from data
- Feature Extraction: creates new features
- Factorization Methods for dimensionality reduction:
  - with(out) class information (supervised)
  - (non)linear
  - local / global
- NOT discussed:
  - Number of parameters
  - runtime

Additional Sources

- LLE
  https://cs.nyu.edu/~roweis/lle/algorithm.html
- "13 ways to look at the correlation coefficient",
  https://www.stat.berkeley.edu/~rabbee/correlation.pdf