Machine Learning
Dimensionality Reduction

Prof. Matthias Hein

Machine Learning Group
Department of Mathematics and Computer Science
Saarland University, Saarbrücken, Germany

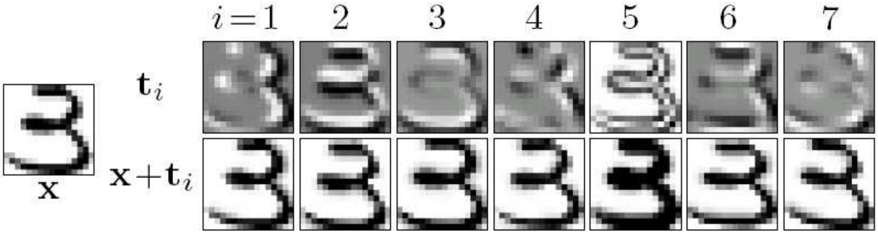
Lecture 26, 05.02.2014

Dimensionality Reduction: Construction of a mapping $\phi : \mathcal{X} \to \mathbb{R}^m$, where the dimensionality m of the target space is usually much smaller than that of the input space $\mathcal{X}$. Generally, the mapping should preserve properties of the input space $\mathcal{X}$ e.g. distances.

Why should we do dimensionality reduction ?

- **Manifold assumption:** The internal degrees of freedom are much smaller than the number of measured features $\Longrightarrow$ data lies along a low-dimensional structure in feature space $\Longrightarrow$ we want to detect these "true parameters".
- **Visualization:** interpretation of data in high dimensions is difficult - embeddings in two or three dimensions can provide insight.
- **Data compression:** compress the data but retain most of the information.

Manifold-Assumption



- digits vary smoothly (but discretization as pixels),
- internal degrees of freedom are small compared to the number of features (= number of pixels).

Dimensionality reduction

Supervised dimensionality reduction:

- Linear discriminant analysis (LDA),

Unsupervised dimensionality reduction:

- Principal Components Analysis (PCA), (also called: Karhunen-Loeve-Transformation),
- Kernel PCA,
- Laplacian Eigenmaps,
- Independent Component Analysis (ICA).

Except the last all are eigenvalue problems !

PCA - Two points of view

- the principal k-components span the k-dimensional affine subspace which yields the best approximation of the data (Euclidean norm),
- the subspace spanned by the first k principal components contains "most" of the variance in the data.

PCA - a simple coordinate transformation

- translation - mean of data points becomes new origin,
- rotation - change of the initial ONB into a new ONB which is defined by the data.

PCA - Approximation point of view

Given: $\{X_i\}_{i=1}^n$ in $\mathbb{R}^d$, Goal: find a m-dimensional affine subspace U_m, with

$$U_m = c + V_m := c + \Big\{ \sum_{j=1}^m \alpha_j u_j \mid \{u_j\}_{j=1}^m \text{ ONS }, \, c \in \mathbb{R}^d, \, \alpha_j \in \mathbb{R} \Big\},$$

which approximates the original data points optimally in the sense,

$$\mathop{\arg\min}_{Z_i \in V_m, \, c \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^n \| Z_i + c - X_i \|_2^2 \, .$$

Orthogonal projection P onto the subspace V_m: $P = \sum_{j=1}^m u_j u_j^T$.

Lemma

An orthogonal projection matrix $P : \mathbb{R}^d \to \mathbb{R}^d$ *satisfies,*

$$P = P^T, \;\; and \; P^2 = P.$$

Optimal offset c

Affine subspace: $U_m = c + V_m$, (c can be seen as origin of U_m).

$$\nabla_c \Big(\sum_{i=1}^n \|Z_i + c - X_i\|_2^2 \Big) = 2\sum_{i=1}^n (Z_i - X_i) + 2nc \implies c = \frac{1}{n}\sum_{i=1}^n (X_i - Z_i).$$

- c depends on Z_i - the origin of the subspace U_m can be changed without changing the approximation.
- fix degree of freedom by requiring that

$$\sum_{i=1}^n Z_i = 0 \quad \text{and thus} \quad c = \frac{1}{n}\sum_{i=1}^n X_i.$$

We center the original data points X_i: $\tilde{X}_i = X_i - \frac{1}{n}\sum_{j=1}^n X_j$.

New Objective: $$\sum_{i=1}^n \|Z_i + c - X_i\|_2^2 = \sum_{i=1}^n \left\|Z_i - \tilde{X}_i\right\|_2^2.$$

$$\left\|Z_i - \tilde{X}_i\right\|_2^2 = \left\|Z_i - P\tilde{X}_i\right\|_2^2 + \left\|P\tilde{X}_i - \tilde{X}_i\right\|_2^2,$$

for the orthogonal projection P onto $U_m \Longrightarrow$ choose $Z_i = P\tilde{X}_i$.

New **transformed objective:**

$$\begin{aligned}
\sum_{i=1}^n \left\|Z_i - \tilde{X}_i\right\|_2^2 &= \sum_{i=1}^n \left\|(P - \mathbb{1})\tilde{X}_i\right\|_2^2 \\
&= \sum_{i=1}^n \tilde{X}_i^T(\mathbb{1} - P)\tilde{X}_i \\
&= \sum_{i=1}^n \tilde{X}_i^T\tilde{X}_i - \sum_{i=1}^n \tilde{X}_i^T P\tilde{X}_i \\
&= \sum_{i=1}^n \tilde{X}_i^T\tilde{X}_i - \sum_{j=1}^n u_j^T\Big(\sum_{i=1}^n \tilde{X}_i\tilde{X}_i^T\Big)u_j
\end{aligned}$$

PCA - Approximation IV

Final objective:

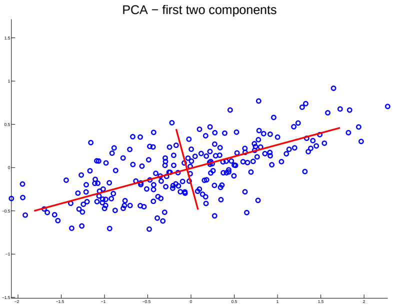
$$\sum_{i=1}^{n} \left\| Z_i - \tilde{X}_i \right\|^2 = \sum_{i=1}^{n} \tilde{X}_i^T \tilde{X}_i - \sum_{j=1}^{m} u_j^T \Big(\sum_{i=1}^{n} \tilde{X}_i \tilde{X}_i^T \Big) u_j.$$

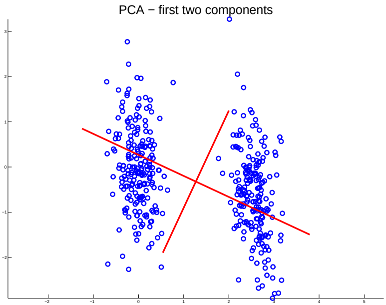
Define the symmetric, positive semi-definite matrix $C \in \mathbb{R}^{d \times d}$ as,

$$C = \sum_{i=1}^{n} \tilde{X}_i \tilde{X}_i^T,$$

- objective is minimized by using the projection P onto the the m largest eigenvectors of C
- These eigenvectors are called the **principal components** of the data.

PCA - Illustration





- red directions: principal directions in the data
- length of red line: $4\sqrt{\lambda}$, where λ is the eigenvalue of C.

Subspace containing most of the variance of a probability measure

One-dimensional subspace U_1 spanned by $u \Rightarrow$ variance of the data projected onto u is given as

$$\mathrm{var}(u) = \mathbb{E}_X[\langle u, X - \mathbb{E}X\rangle^2] = \mathbb{E}_X\Big[\big(\langle u, X\rangle - \langle u, \mathbb{E}_X\rangle\big)^2\Big].$$

Rewrite $\mathrm{var}(u)$ as

$$\mathrm{var}(u) = \mathbb{E}_X[u^T(X - \mathbb{E}X)(X - \mathbb{E}X)^T u] = \langle u, Cu\rangle\,,$$

where

$$C = \mathbb{E}_X(X - \mathbb{E}X)(X - \mathbb{E}X)^T,$$

is the **covariance of** P_X.

Subject to $\|u\|^2 = 1 \Rightarrow$ using Rayleigh-Ritz principle, $\mathrm{var}(u)$ is maximized by the eigenvector of C corresponding to the largest eigenvalue.

PCA - Variance II

Best *m*-dimensional subspace: m "largest" eigenvectors.

- the ev, $\{u_i\}_{i=1}^d$, of C determine an **uncorrelated** ONB,

$$\langle u_i, C u_j \rangle = \lambda_i \delta_{ij}, \quad i,j = 1, \ldots, d.$$

- For **Gaussian** data: $p(x) = \frac{1}{(2\pi)^{\frac{d}{2}} |\det C|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)^T C^{-1}(x-\mu)}$,
  we get in new coordinates z defined as,

$$z \;=\; C^{-\frac{1}{2}}(x-\mu) \;=\; \sum_{i=1}^d \frac{1}{\sqrt{\lambda_i}} u_i \, u_i^T (x-\mu),$$

  components z_j which are **independent** and equally distributed,

$$p(z) = \frac{1}{(2\pi)^{\frac{d}{2}}} e^{-\frac{\|z\|^2}{2}} = \prod_{j=1}^d \frac{1}{\sqrt{2\pi}} e^{-\frac{z_j^2}{2}}.$$

This process is called **whitening**.

PCA - Whitening

Whitening: PCA + rescaling.

$$z = C^{-\frac{1}{2}}(x - \mu).$$

Whitening are three concatenated operations:

- **centering** - equivalent to a translation in $\mathbb{R}^d$,
- **projection onto (all) principal components** - equivalent to a change from the initial basis to the basis spanned by the eigenvectors of C
  $\Longrightarrow$ rotation,
- **rescaling** - one rescales each axis by the square-root of the corresponding eigenvalue - thus one has unit variance in each direction.

In practice:

- pre-processing of data $\Rightarrow$ resulting features are uncorrelated,
- Whitening "spheres" the data - eliminates differences in scaling.

PCA - In practice

Probability measure unknown only given i.i.d. sample $\{X_i\}_{i=1}^n$
$\Longrightarrow$ use **empirical covariance matrix**,

$$C = \frac{1}{n}\sum_{i=1}^n (X_i - \overline{X})(X_i - \overline{X})^T, \quad \text{with} \quad \overline{X} = \frac{1}{n}\sum_{i=1}^n X_i$$

and use its eigenvalues and eigenvectors as principal components.

Further practical issues:

- never cut the spectrum where two eigenvalues are close,
- several people use the first k-principal components to define new coordinates for supervised problems e.g. classification. This is problematic since the class structure need not have anything to do with the principal components.
  Supervised case: use LDA or other supervised extensions of PCA.

Kernel PCA

Non-linear extension of PCA:

- given: positive definite kernel $k : \mathcal{X} \to \mathcal{X} \to \mathbb{R}$,
- map data into the corresponding feature space (RKHS) $\mathcal{H}_k$,

$$\phi : \mathcal{X} \to \mathcal{H}_k, \qquad x \to \phi(x).$$

- do PCA in $\mathcal{H}_k$ (resp. subspace spanned by the data).
- principal components correspond to functions $\mathcal{X}$.

Questions:

- how to define eigenvectors in $\mathcal{H}_k$?
- how many principal components are there ?
- what is a principal component in $\mathcal{H}_k$?

Standard-PCA:

$$Cv = \lambda v, \quad \Longrightarrow \quad \frac{1}{n}\sum_{i=1}^{n} \langle X_i, v\rangle \; X_i = \lambda\, v.$$

$\Longrightarrow$ all eigenvectors lie in the span of the data points.

Kernel-PCA: map $\phi : \mathcal{X} \to \mathcal{H}_k$

$$C = \frac{1}{n}\sum_{j=1}^{n} \phi(X_j)\phi(X_j)^T.$$

If $\dim \mathcal{H}_k = \infty$ then C is a linear operator in $\mathcal{H}_k$.
As in PCA we want to find the eigenvectors of C,

$$Cv = \lambda v \quad \Longrightarrow \quad \frac{1}{n}\sum_{i=1}^{n} \langle \phi(X_i), v\rangle_{\mathcal{H}_k} \; \phi(X_i) = \lambda\, v.$$

$\Longrightarrow$ all eigenvectors lie in the span of the **mapped** data points.

Kernel PCA - the essential

Kernel-PCA: $Cv = \lambda v \quad \Longrightarrow \quad \frac{1}{n}\sum_{i=1}^{n} \langle \phi(X_i), v\rangle_{\mathcal{H}_k} \, \phi(X_i) = \lambda \, v.$

Equivalently, solve for all $j = 1, \ldots, n$,

$$\frac{1}{n}\sum_{i=1}^{n} \langle \phi(X_i), v\rangle_{\mathcal{H}_k} \, \langle \phi(X_i), \phi(X_j)\rangle_{\mathcal{H}_k} = \lambda \, \langle v, \phi(X_j)\rangle_{\mathcal{H}_k}.$$

Moreover, from the above derivation we know: $v = \sum_{r=1}^{n} \alpha_r \, \phi(X_r)$,

$$\frac{1}{n}\sum_{i,r=1}^{n} \alpha_r \, \langle \phi(X_i), \phi(X_r)\rangle_{\mathcal{H}_k} \, \langle \phi(X_i), \phi(X_j)\rangle_{\mathcal{H}_k} = \lambda \sum_{r=1}^{n} \alpha_r \, \langle \phi(X_r), \phi(X_j)\rangle_{\mathcal{H}_k}.$$

This can be summarized using $k(X_i, X_j) = \langle \phi(X_i)\phi(X_j)\rangle_{\mathcal{H}_k}$ as,

$$K^T K \alpha = n \, \lambda \, K^T \alpha.$$

This is (almost) equivalent to: $K\alpha = n \, \lambda \alpha$.

What is the difference of the two equations ?

Kernel-PCA: solve eigen-problem: $K\alpha = n\,\lambda\alpha$.

- normalize eigenvectors $v^{(s)}$, $s = 1, \ldots, n$,

$$\left\langle v^{(s)}, v^{(s)} \right\rangle_{\mathcal{H}_k} = \sum_{i,j=1}^{n} \alpha_i^{(s)} \alpha_j^{(s)} K_{ij} = \lambda^{(s)} \sum_{i=1}^{n} \alpha_i^{(s)} \alpha_i^{(s)}.$$

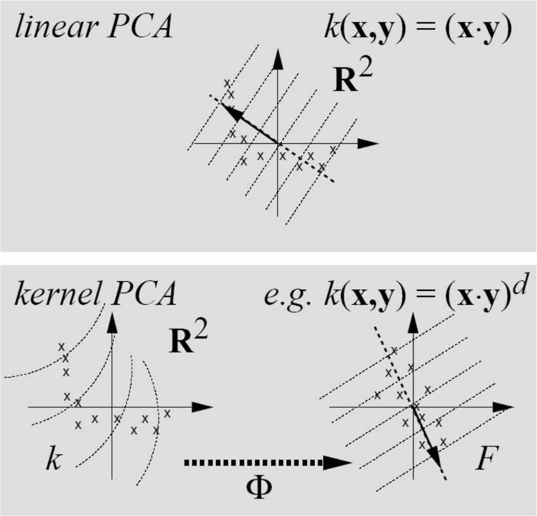
- What are the principal components (functions) ? Compute projection of mapped test point x on $v^{(s)}$,

$$\left\langle v^{(s)}, \phi(x) \right\rangle_{\mathcal{H}_k} = \sum_{i=1}^{n} \alpha_i^{(s)} \left\langle \phi(X_i), \phi(x) \right\rangle_{\mathcal{H}_k} = \sum_{i=1}^{n} \alpha^{(s)} k(X_i, x).$$

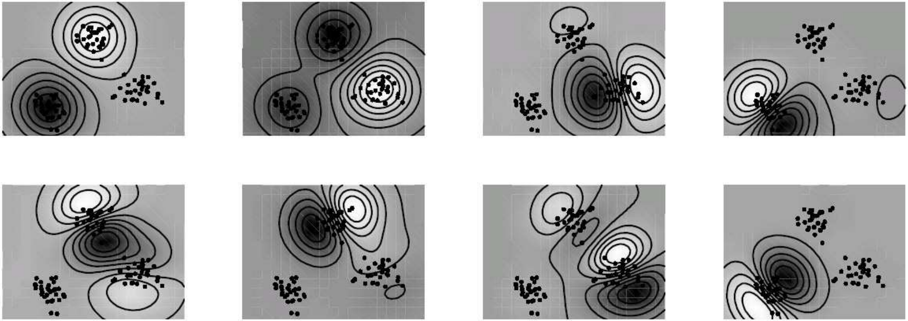
Standard PCA components are linear functions ! Variation into the direction of the principal component.

- What requirement of PCA did we not integrate into the derivation of Kernel PCA ?

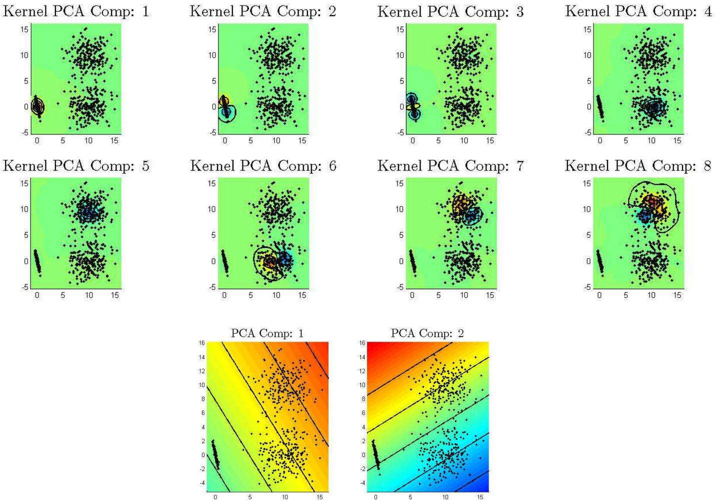
Illustration: PCA versus Kernel-PCA



Balanced clusters: Higher principal components of Kernel-PCA

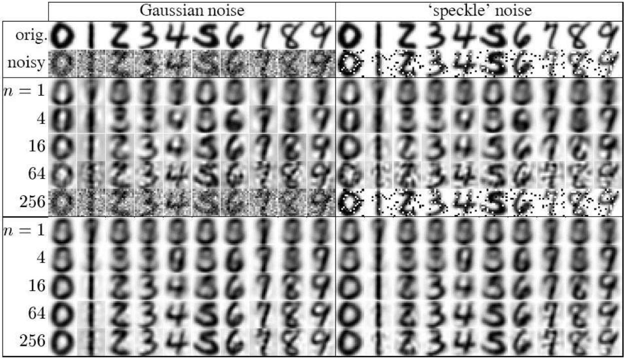


Disbalanced clusters: Higher principal components of Kernel-PCA



Kernel PCA - Denoising

Kernel-PCA for denoising of data



- PCA allows for reconstruction of the original image (just a basis transformation),
- for Kernel PCA this is not directly possible - need to find a pre-image for $\sum_{i=1}^{n} \alpha_i \phi(x_i) \in \mathcal{H}_k$ in the original space $\mathcal{X}$.

Laplacian eigenmaps

The continuous Laplacian

$$\mathbb{R}^d, \qquad \Delta = \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}.$$

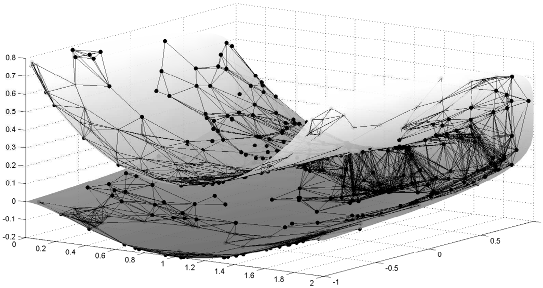
Why is it interesting ?

- Laplacian is symmetric (self-adjoint),
- eigenfunctions, $\Delta f = \lambda f$, define an ONB of $L_2(\mathbb{R}^d)$.
- these eigenfunctions have nice properties
  - $\mathbb{R}$: Fourierbasis $\phi_{2k}(x) = \cos(x)$, $\phi_{2k+1}(x) = \sin(x)$,
  - sphere S^2: spherical harmonics.

  $\Longrightarrow$ multi-scale decomposition of the data,
- Fourier-transform is the corresponding basis transformation.

Can we do the same for discrete data ?

The data manifold



- we would like to find the parameters underlying the data-generating process $\Rightarrow$ parameterization of the data-manifold.
- **Idea:** build graph - use graph Laplacian as surrogate of the continuous Laplacian.
  $\Longrightarrow$ eigenvectors generate multi-scale decomposition of the data.

Use the graph Laplacian

Three types of graph Laplacians:

unnormalized:
$$(\Delta^{(u)} f)(i) = d(i) f(i) - \sum_{j=1}^{n} w_{ij} f(j),$$
$$(\Delta^{(u)} f) = (D - W) f,$$

random walk:
$$(\Delta^{(rw)} f)(i) = f(i) - \frac{1}{d(i)} \sum_{j=1}^{n} w_{ij} f(j),$$
$$(\Delta^{(rw)} f) = (\mathbb{1} - D^{-1} W) f,$$

normalized:
$$(\Delta^{(n)} f)(i) = f(i) - \sum_{j=1}^{n} \frac{w_{ij}}{\sqrt{d_i\, d_j}} f(j),$$
$$(\Delta^{(n)} f) = (\mathbb{1} - D^{-1/2} W D^{-1/2}) f.$$

Laplacian eigenmaps

Laplacian Eigenmaps

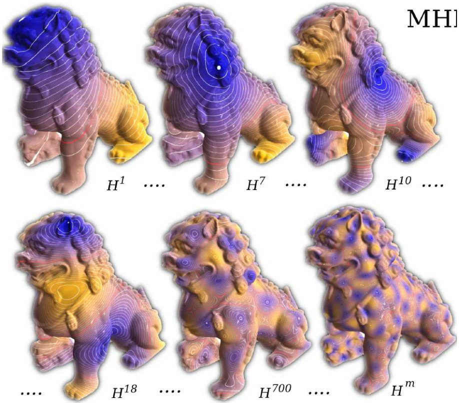
Chooose the graph Laplacian: unnormalized, random walk and normalized.

- compute the graph Laplacian $n \times n$-matrix for n points,
- compute the first k eigenvectors $\{u_i\}_{i=1}^k$ (each eigenvector is normalized, $\|u_i\| = 1$, $i = 1, \ldots, k$),
- Embedding $\phi : V \to \mathbb{R}^k$, of the n vertices into $\mathbb{R}^k$ by $i \to z_i = (u_1(i), \ldots, u_k(i))$,

The embedding: $\phi : V \to \mathbb{R}^k$, $i \to \phi(i) = (u_1(i), \ldots, u_k(i))$ is the **Laplacian eigenmap**.

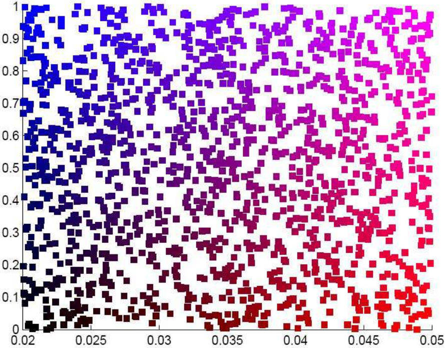
Relation to Kernel-PCA:

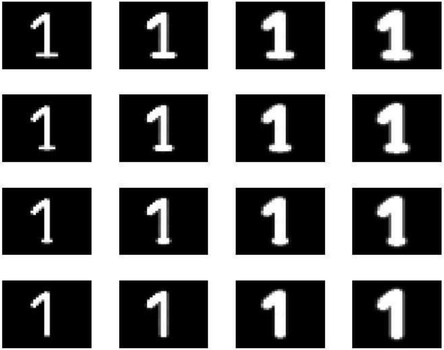
One can see Laplacian eigenmaps as Kernel PCA with a special data-dependent kernel (pseudo-inverse of the graph Laplacian).



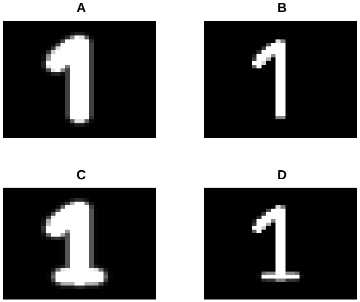
- compute eigenvectors of the Laplacian on the mesh,
- can be used for denoising of meshes, varying of meshes etc.

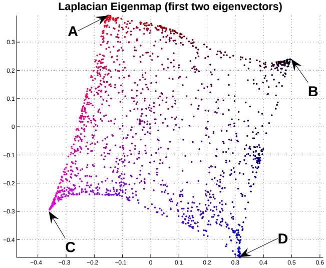
Laplacian Eigenmaps - Illustration





- **Right:** artificial datasets of ones - two variations: line thickness and style variation (bottom line) - digits are of size 28×28 - 784 pixels,
- **Left:** sampling is done uniformly in the parameterization.





- the original parameter set is equivalent to $[0,1]^2$ and the examples A, B, C, D are the corners of $[0,1]^2 \implies$ Laplacian eigenmap finds the parameterization.

Independent Component Analysis (ICA)

Motivation: cocktail party problem - blind source separation

- k different speakers (sources),

$$s_1(t), \dots, s_k(t).$$

- d microphones (sensors),

$$x_1(t), \dots, x_d(t).$$

Assumption: measured signal is linear superposition of sources.

Goal: having only the signal of the microphones, find the sources - determine A, where

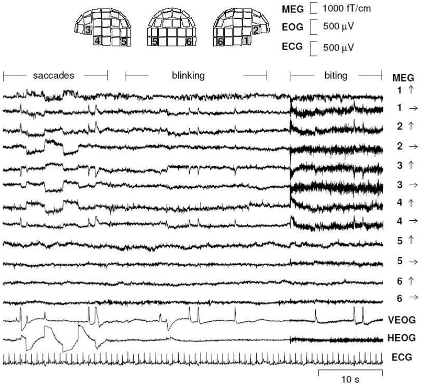
$$x(t) = A\, s(t).$$

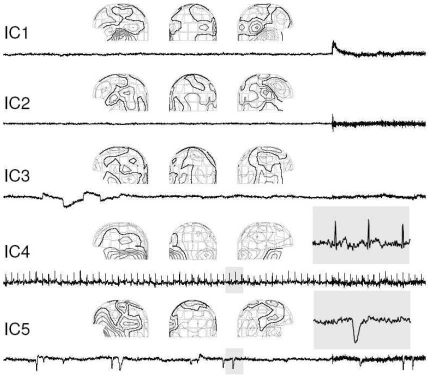
- A is called the **mixing matrix**.

Application scenarios

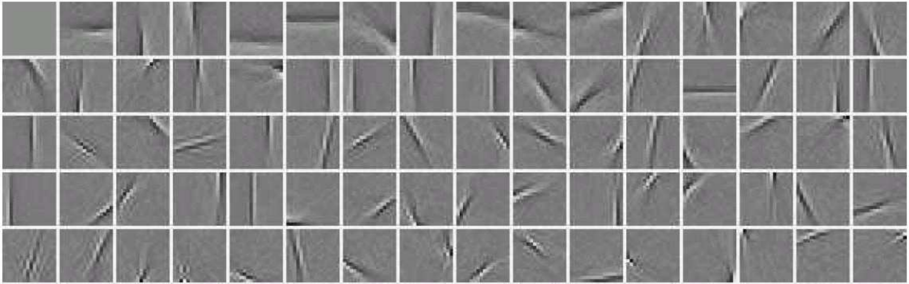
- sound (speech, music,...) signals,
- EEG signals,
- natural images (patches),
- financial data,
- ...

ICA for EEG analysis





ICA for natural images - 16×16 - patches



- ICA components for 16×16-patches of natural images,
- $\Longrightarrow$ one observes that independent components look like edge detectors.

Motivation for ICA

- speakers (sources) are independent of each other.

$$s_1(t), \ldots, s_k(t),$$

in the stochastic sense (source signals are independent random variables),

$$p_s\Big(s_1(t), \ldots, s_k(t)\Big) = \prod_{i=1}^{k} p_{s_i}\big(s_i(t)\big).$$

Find new representation such that components are maximally independent !

$\Longrightarrow$ how can one optimize for independent components ?

$\Longrightarrow$ **for simplicity we assume** $d = k$ (nr. sensors = nr. sources).

ICA III

What kind of independent components can we hope for ?

- **non-Gaussian sources:** suppose that $s(t) \in \mathbb{R}^k$ is Gaussian distributed $\Longrightarrow x = As$ is again Gaussian distributed,

$$\mathbb{E}[xx^T] = \mathbb{E}[A\, ss^T A^T] = A\mathbb{E}[s\, s^T]A^T = A\mathbb{1}_k A^T = AA^T.$$

  Whitening yields independent components - but not necessarily $s(t)$.

- **Sources can be identified only up to rescaling:**

$$x(t) = A\, s(t) = (A\, D^{-1})\,(D\, s(t)),$$

  where D is a diagonal matrix - $D\, s(t)$ is also independent. W.l.o.g.,

$$\mathbb{E}[s(t)\, s(t)^T] = \mathbb{1}_k.$$

- **Sources cannot be ordered:** Let P be a permutation matrix, then $Ps(t)$ is independent, $x(t) = A\, s(t) = (A\, P^{-1})\,(Ps(t))$.

Whitening as a pre-processing step for ICA

Whitening transforms the signal $x(t)$,

$$y(t) = W\,x(t) = W\,A\,s(t),$$

such it becomes **uncorrelated**,

$$\mathbb{1}_k = \mathbb{E}[y(t)\,y(t)^T] = \mathbb{E}[W\,x(t)x(t)^T W^T] = W\,A\,\mathbb{E}[s\,s^t]A^T W^T = W\,A\,A^T W$$

$\Longrightarrow$ whitening simplifies the problem since the mixing matrix $W\,A$ for $y(t)$ is orthogonal.

New problem: find the **orthogonal mixing matrix** $B = W\,A$

$$y(t) = B\,s(t),$$

resp. B^T such that $B^T y(t) = B^T B s(t) = s(t)$ is maximally independent.

ICA - Steps

Steps for ICA:

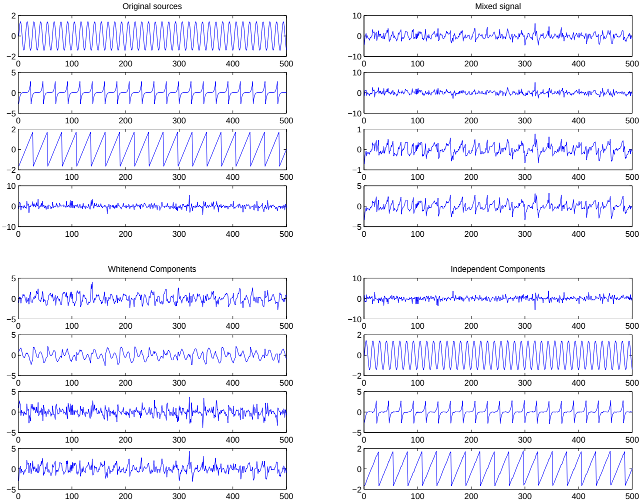
- apply whitening to the data: $y(t) = W\,x(t)$.
- find orthogonal de-mixing matrix B s.th. $B\,y(t)$ is maximally independent.
  Different criteria:
  - maximize non-gaussianity of $By(t)$,
  - minimize mutual information $I\big(\{By(t)\}_{i=1}^{k}\ By(t)\big)$ - mutual information is zero if and only if joint density of $By(t)$ factorizes into the product of the marginal densities $\Longrightarrow By(t)$ is independent.

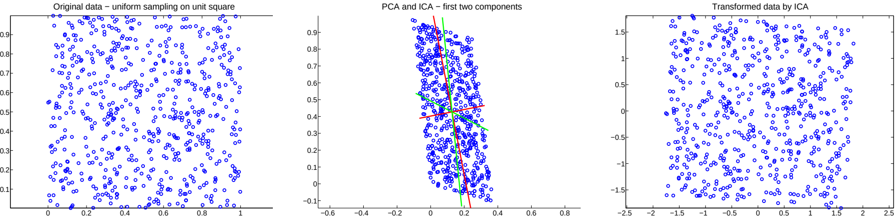
Problems:

- joint density of $B\,y(t)$ hard to estimate $\rightarrow$ problems with mutual inf.
- instead: minimize higher order correlations e.g. **kurtosis**

$$\mathrm{kurt}(y) = \mathbb{E}[y^4] - 3\Big(\mathbb{E}[y^2]\Big)^2.$$

Illustration of ICA for signal data





- **Left:** Original sources - individual features are independent $p(x_1, x_2) = p(x_1)p(x_2)$.
- **Middle:** Measured signal - directions of PCA (eigenvectors of covariance matrix) and directions of ICA (columns of estimated mixing matrix) are shown - note that the directions of ICA are **not** orthogonal,
- **Right:** Source signal estimated by ICA - coincides up to rescaling with the original signal.

Cocktail party demo.