EMORY
UNIVERSITY
SCHOOL OF
MEDICINE

Department of
Biomedical Informatics

BMI 500: Introduction to Biomedical Informatics

Lecture 11. An Introduction to Blind Source Separation and Independent Component Analysis

Reza Sameni

Department of Biomedical Informatics, Emory University, Atlanta, GA, USA

30th October, 2020

Overview

Outline

Introduction

- **Blind source separation (BSS):** the problem of recovering random variables or random processes from an unknown linear or nonlinear mixture of such sources, using minimal assumptions on the sources and mixtures:

$$\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{s}(t), t) \tag{1}$$

- Although the problem seems unsolvable, various evidences in biology and nature prove it to be possible
- The problem was first studied in signal processing in the 1980s
- In this lecture, a short and broad introduction is presented on BSS and one of its most popular solutions, known as **independent component analysis (ICA)**

Evidence of BSS in nature: the cocktail party effect

- Cocktail party effect or selective attention: the ability to focus attention on a single speaker in a crowd
- From the signal processing perspective, people can understand a speaker even in very low quality environments (even negative SNR)
- Various properties of speech are used in selective attention

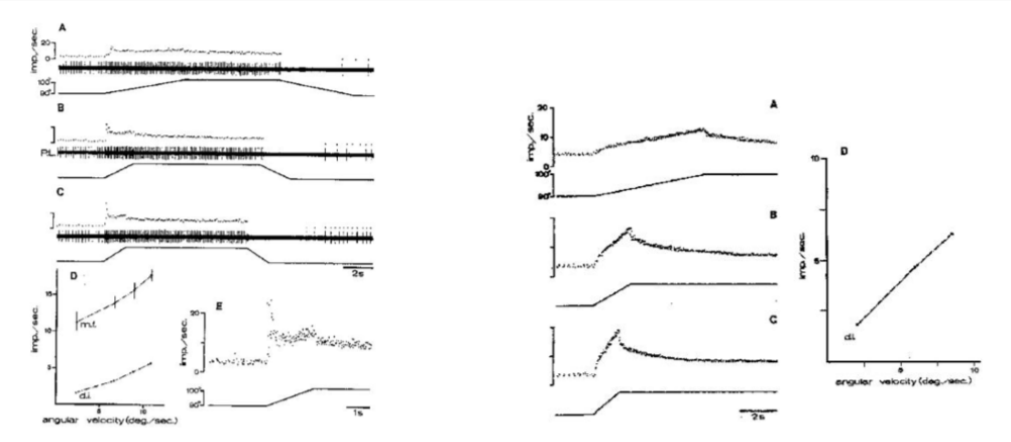
Evidence of BSS in nature: the cocktail party effect (continued)

Various factors of speech influence selective attention and intelligibility:

- Signal to noise ratio (SNR)
- Non-stationarity and sparsity of speech (time domain)
- Speaker voice tone (frequency domain)
- Facial gestures and lip reading (multimodality)
- Speech language, context, sentence structure, etc. (priors)
- Physiological and anatomic factors (two ears and their shapes improve selective listening)

Evidence of BSS in nature: motion decoding in vertebrates

Observation by Jean-Pierre Roll (Roll, 1981): The human brain is able to learn the position (joint stretch) and velocity of body extremities from mixtures of these variables received from the nervous system.



This observation inspired BSS research in signal processing:

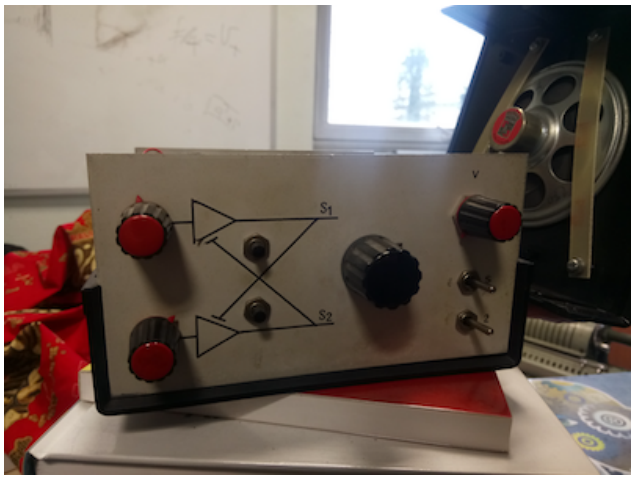
$$\begin{aligned} f_1(t) &= a_{11}p(t) + a_{12}v(t) \\ f_2(t) &= a_{21}p(t) + a_{22}v(t) \end{aligned} \qquad \begin{aligned} p(t) &: \text{joint stretch} \\ v(t) &: \text{joint velocity} \end{aligned} \tag{2}$$

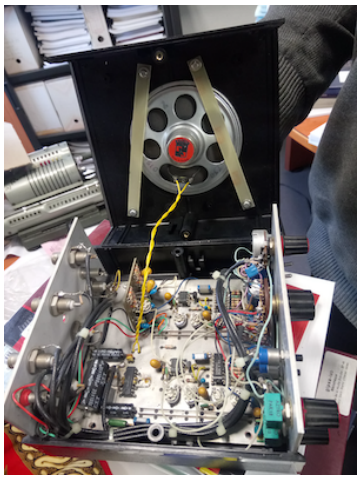
History of BSS in signal processing

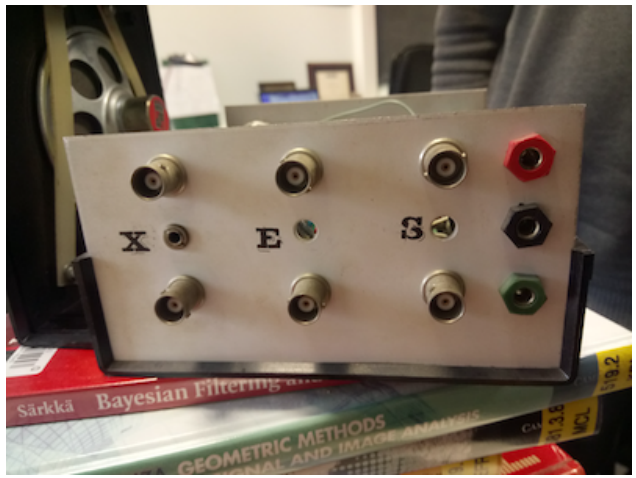
A legendary article: J. Hérault, C. Jutten, and B. Ans. Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé. In Actes du Xème colloque GRETSI, pages 1017–1022, Nice, France, 20–24 May 1985.

History of BSS in signal processing

A legendary article: J. Hérault, C. Jutten, and B. Ans. Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé. In Actes du Xème colloque GRETSI, pages 1017–1022, Nice, France, 20–24 May 1985.

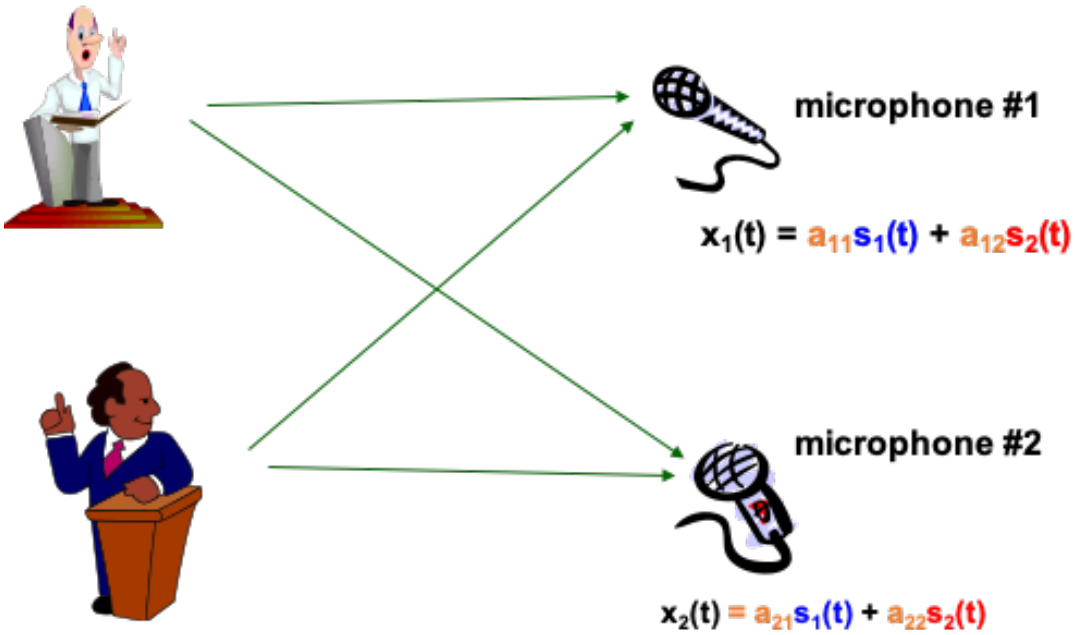


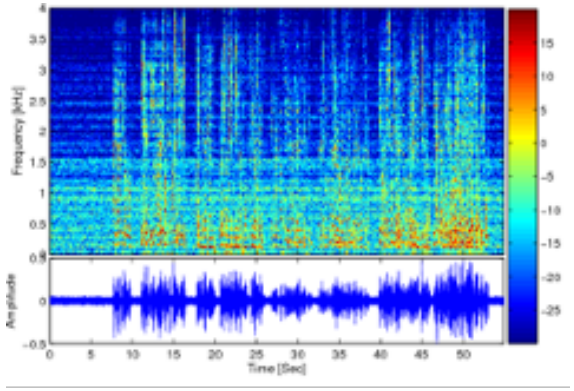


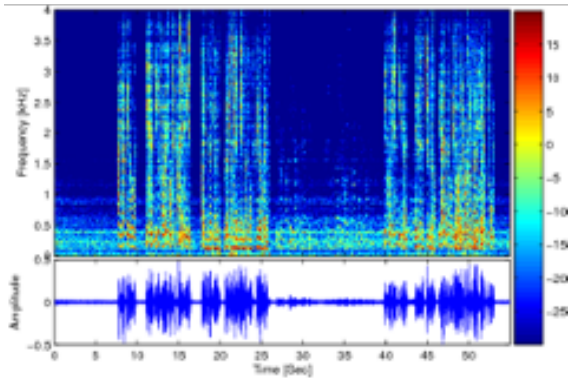


An analog source separation machine (Courtesy of Christian Jutten). See the following for notations and algorithm: C. Jutten and J. Hérault. "Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture." *Signal processing* 24, no. 1 (1991): 1–10. APA

Applications: speech separation and enhancement



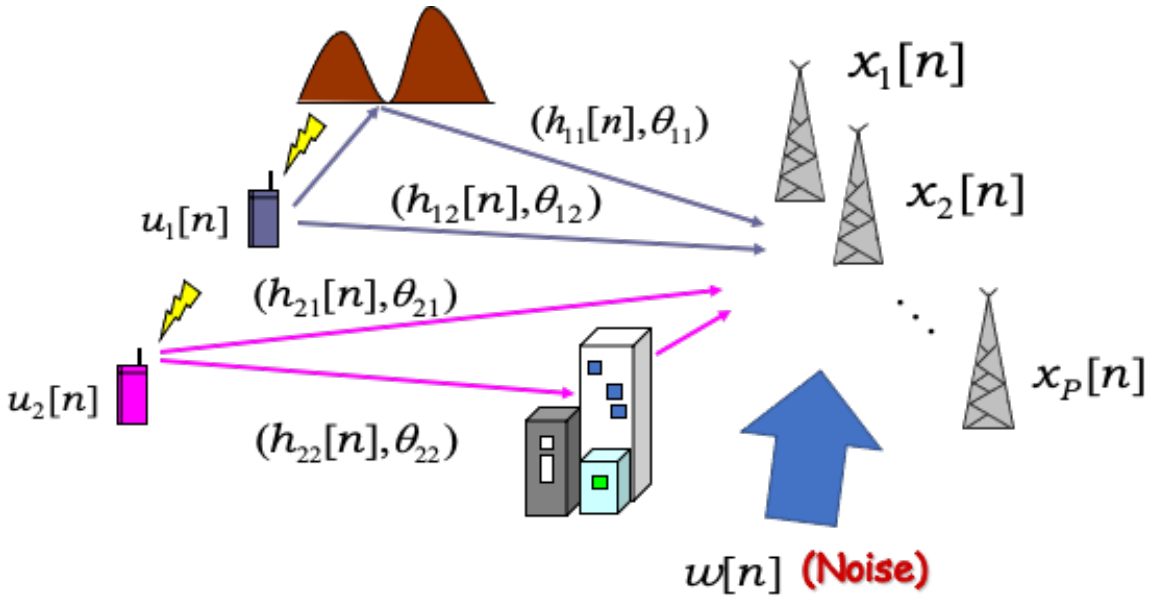


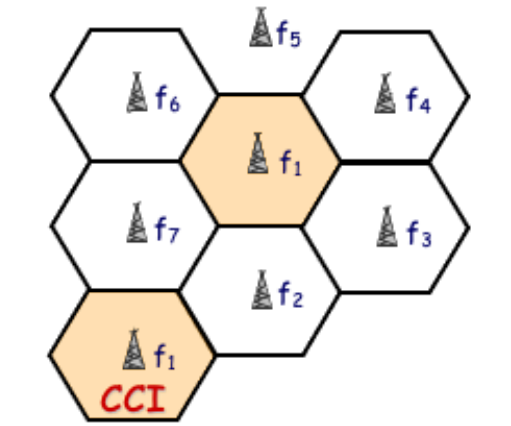


Sound propagates at 330m/s in air; so it takes 3ms for sound to propagate 1m. Therefore a linear instantaneous mixture is a good approximation of the audio scenario in some cases. For farther distances, convolutive mixtures are required.

Applications: multi-path telecommunication channels

In wireless communications, the desired signal is degraded by interference and noise received from unwanted paths.

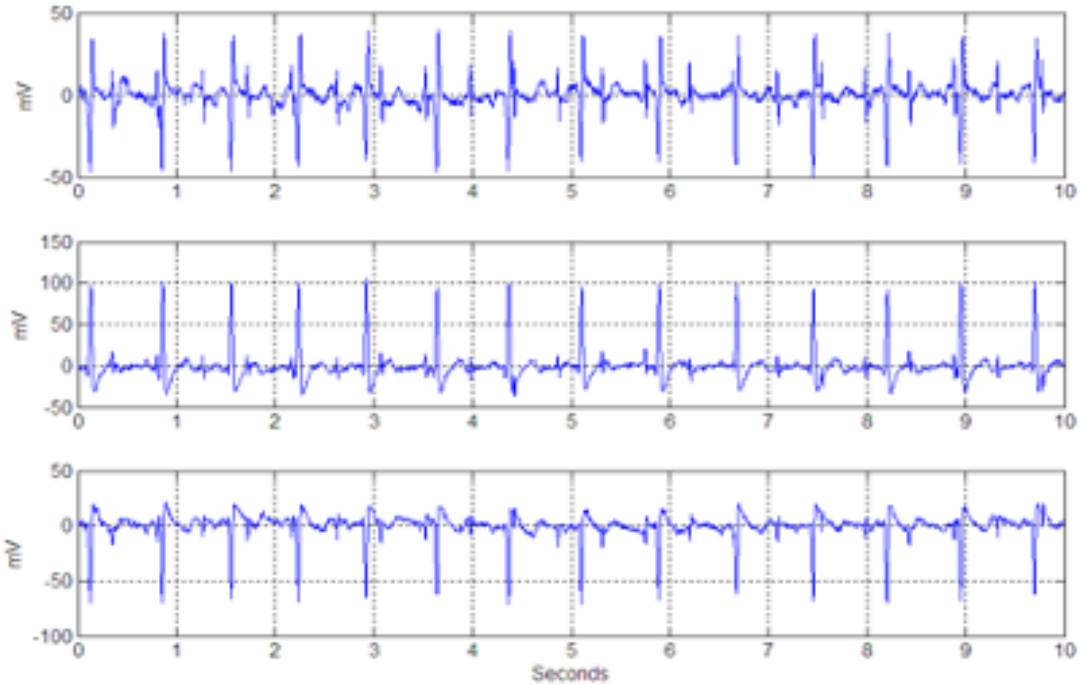


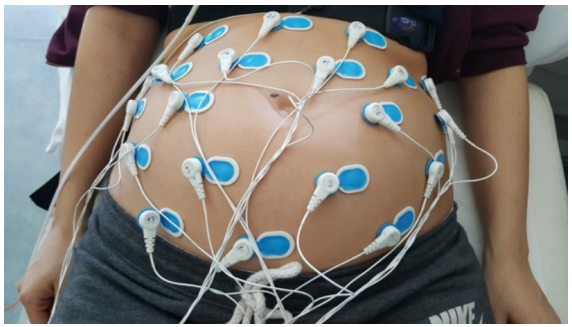


CCI: Co-channel Interference

Applications: noninvasive fetal electrocardiogram extraction

The noninvasive extraction of fetal ECG (fECG) from an array of electrodes placed on the maternal abdominal is a challenging biomedical problem

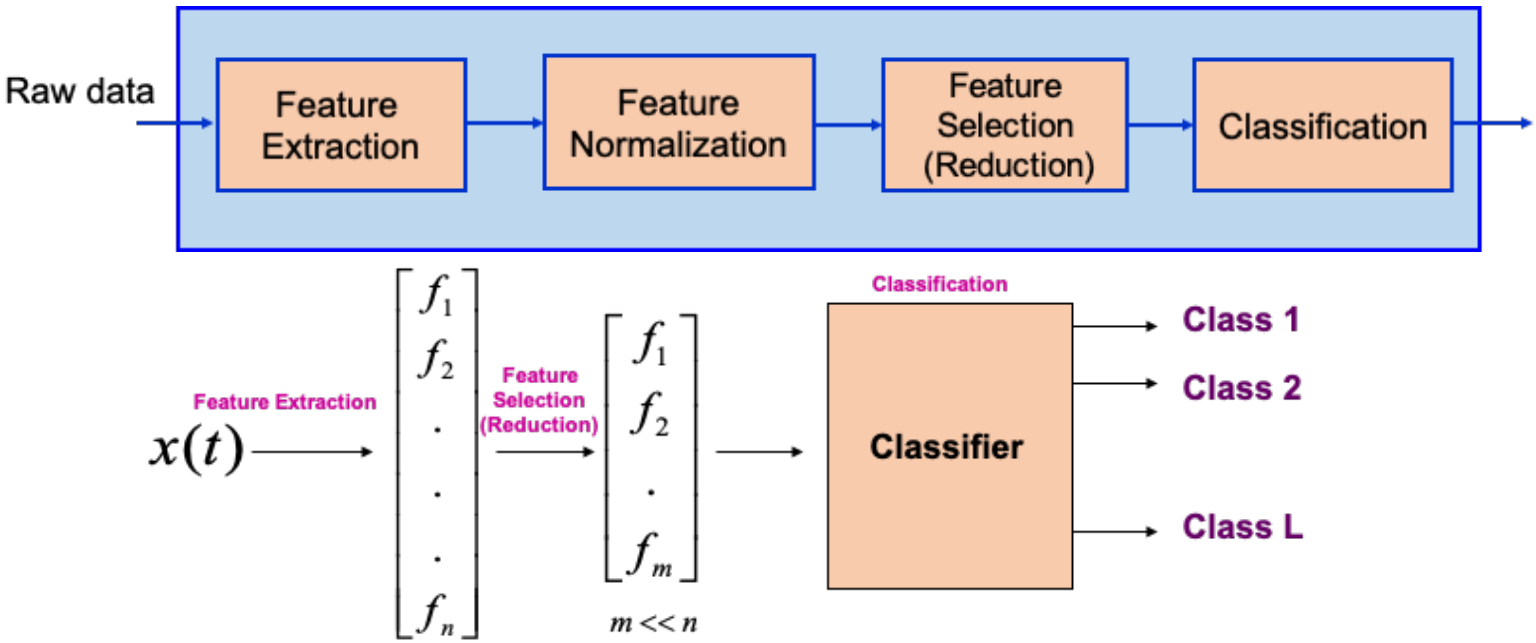




(Courtesy of Danilo Pani)

Applications: feature extraction and classification

In classification, statistically independent features are preferred over correlated or uncorrelated ones, as there is no redundancy between the features. How to make a set of features independent?



Outline

Mathematical problem definition

Data Model: In the most general case, blind source separation (BSS) can be mathematically formulated as follows:

$$\boldsymbol{x}(t) = \boldsymbol{f}(\boldsymbol{s}(t), t) \tag{3}$$

where

- t denotes time
- $\boldsymbol{x}(t)$ are N channels of observations (known)
- $\boldsymbol{s}(t)$ are M channels of sources (unknown)
- $\boldsymbol{f}(\cdot, t)$ is a (generally) nonlinear and time-variant transform (unknown)

Objective: to estimate $\boldsymbol{s}(t)$ from $\boldsymbol{x}(t)$, using minimal assumptions on the sources and mixtures (this is what the prefix "blind" stands for)

Blind source separation solutions

- The problem is apparently ill-posed and without any other prior assumptions there are no unique solutions
- Various cases of BSS with proven solutions include:
  - **Linear mixtures:**
    - Statistically independent and non-Gaussian sources
    - Signals with temporal correlation
    - Sparse sources (in various domains)
    - Temporally nonstationary sources
    - (Pseudo-)periodic sources
  - **Nonlinear mixtures:**
    - Post-nonlinear sources
    - Signals with temporal correlation
    - Sparse sources

Blind source separation solutions
(continued)

Herein, we study two approaches used for linear BSS without noise:

- **Iterative methods** (such as fastICA)
- **Algebraic methods** (such as AMUSE, JADE and ΠCA)

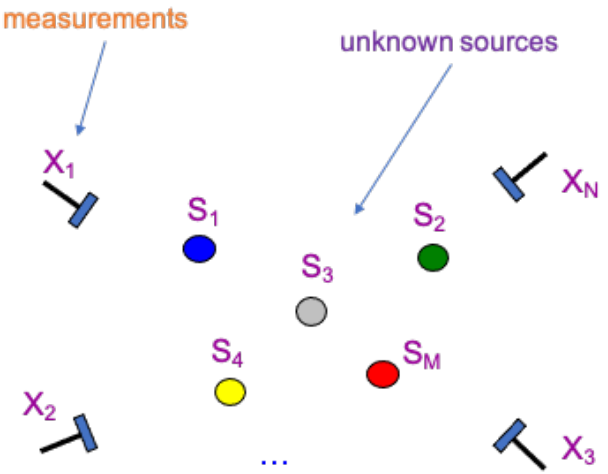
which represent two major classes of BSS methods

Outline

Linear mixtures of independent sources

Data Model: The most classical case of BSS with a proven solution is for linear mixtures of statistically independent and non-Gaussian sources (random variables of random processes): $x_i = a_{i1}s_1 + a_{i2}s_2 + \ldots a_{iM}s_M$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ \vdots & \ddots & & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix} \cdot \begin{bmatrix} s_1 \\ \vdots \\ s_M \end{bmatrix}$$



- In matrix form: $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}$
- The noisy case: $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} + \boldsymbol{n}$

Linear mixtures of independent sources
(continued)

- Different cases:
  - $N = M$: Determined
  - $N > M$: Over-determined
  - $N < M$: Under-determined (degenerate)

$$\begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1M} \\ \vdots & \ddots & & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NM} \end{bmatrix} \cdot \begin{bmatrix} s_1 \\ \vdots \\ s_M \end{bmatrix}$$

- The measurements and the sources can generally be **random processes** rather than **random variables**, i.e., $x(t)$ and $s(t)$.
- Independent component analysis (ICA) is one of the possible solutions of BSS, which assumes mixtures of statistically independent sources
- **How does it work?** we need to see how linear transforms alter the topology of random variables

Linear transforms of Gaussian random variables

Example: Suppose that s_1 and s_2 be independent Gaussian random variables with $s_1 \sim N(0,1)$ and $s_2 \sim N(0,2)$

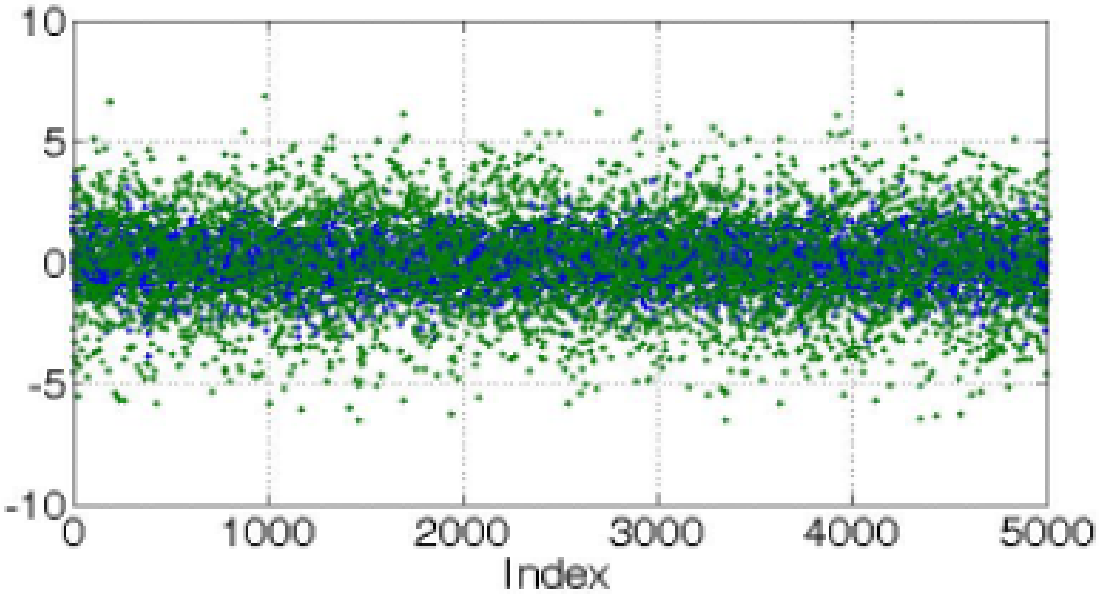


Figure: Sample plot

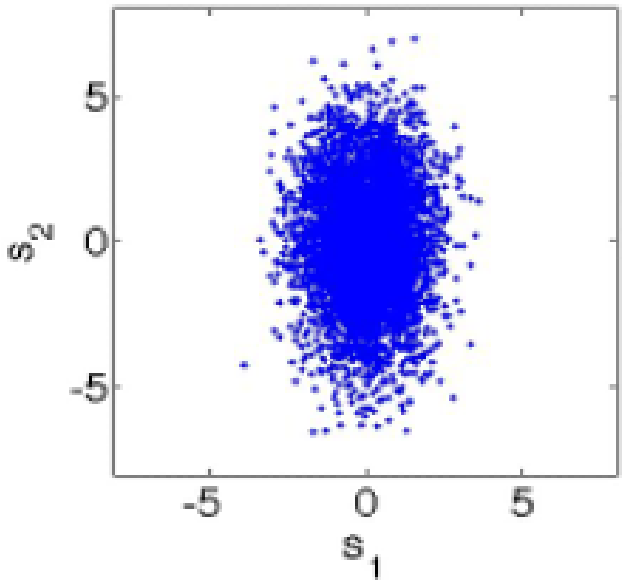
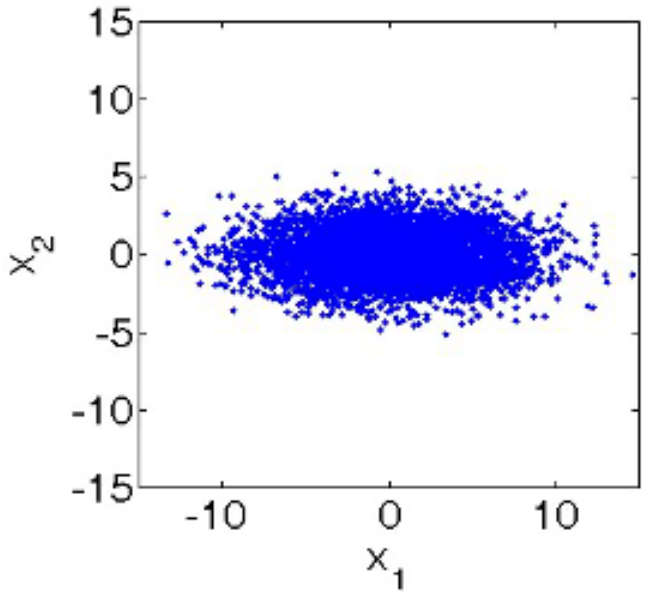


Figure: Scatter plot

Linear transforms of Gaussian random variables
(continued)

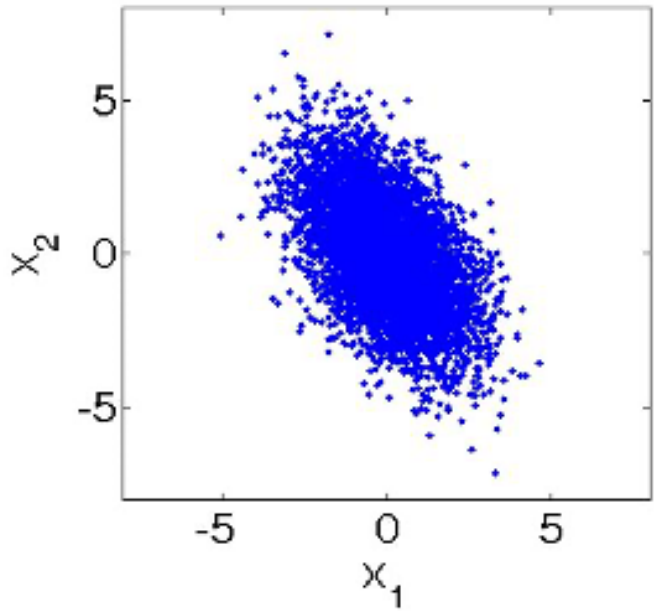
Example: Gaussian samples under transformations

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4.0 & 0 \\ 0 & 0.75 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$$



diagonal transforms stretch the distribution along the axes

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\pi}{6}) & -\sin(\frac{\pi}{6}) \\ \sin(\frac{\pi}{6}) & \cos(\frac{\pi}{6}) \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$$

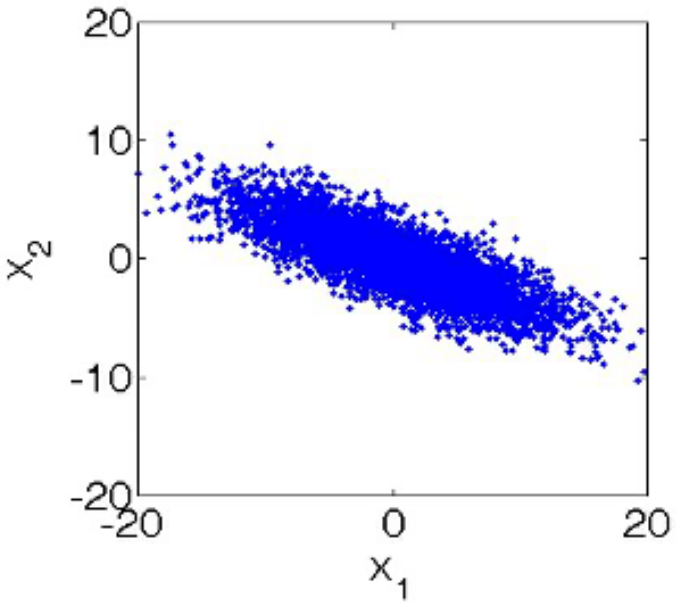


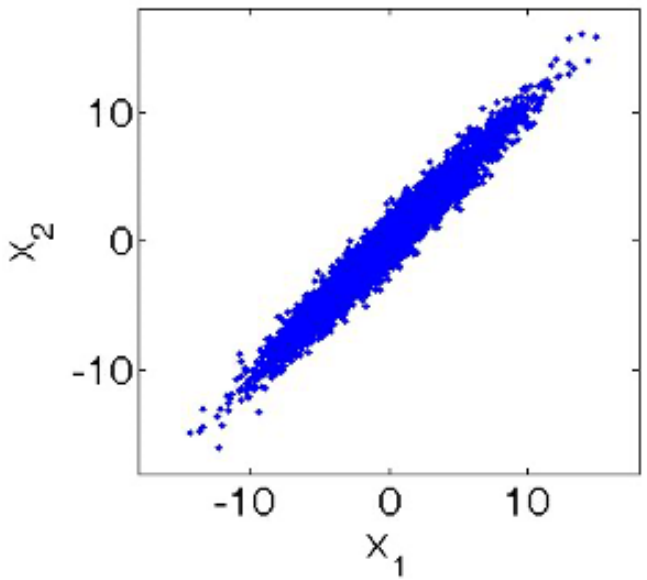
orthonormal transforms rotate the distribution around the axes

Linear transforms of Gaussian random variables
(continued)

Example: Gaussian samples under transformations

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.0 & -3.0 \\ -2.0 & 1.0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix} \qquad \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.0 & 2.0 \\ 2.0 & 2.0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$$





Arbitrary transforms (rotation + scaling), rotate and stretch the distribution

Reminder (Strang, 1988)

Singular value decomposition of an arbitrary matrix:

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{4}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are **orthonormal**, and $\boldsymbol{\Sigma}$ is a **diagonal** matrix.

- Therefore, any linear transform of the form $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s}$ can be decomposed into a series of rotations (unitary transforms) and scalings (diagonal transforms).
- In an inverse problem, recovering (estimating) $\boldsymbol{s}$ from $\boldsymbol{x}$ can be done in a sequence of rotations and scalings, with appropriate assumptions and algorithms.

Eigenvalue decomposition (EVD)

Reminders from linear algebra

A special case of SVD is when the matrix $\boldsymbol{A}$ is symmetric.

Reminder

- For a zero-mean random vector $\boldsymbol{x}$, the covariance matrix $\boldsymbol{C}_x = \mathbb{E}\{\boldsymbol{x}\boldsymbol{x}^T\}$ can be decomposed as follows:

$$\boldsymbol{C}_x = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \tag{5}$$

  where $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N]$ is an orthonormal matrix with the eigenvectors of $\boldsymbol{C}_x$ as its columns, and $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_N)$ is a diagonal matrix of the eigenvalues of $\boldsymbol{C}_x$.

- *Note for later use:* $\text{tr}(\boldsymbol{C}_x) = \sum_{k=1}^{N} \mathbb{E}(x_k^2) = \sum_{k=1}^{N} \lambda_1$

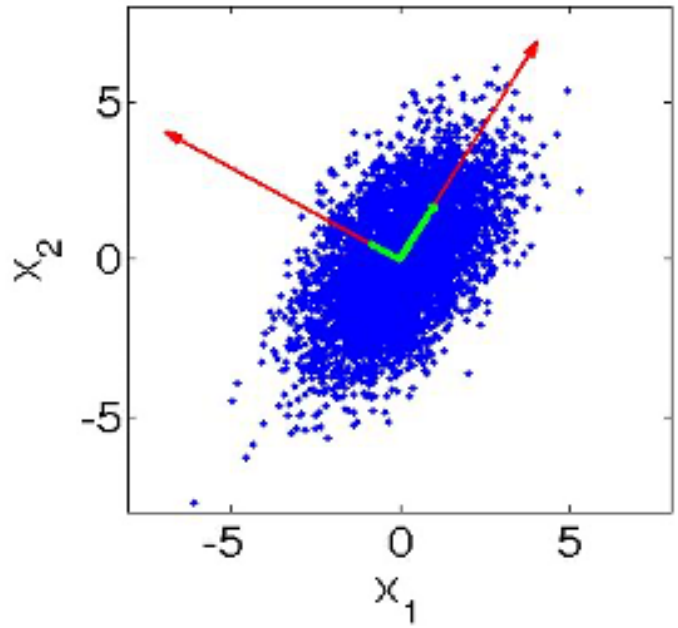
Eigenvalue decomposition (EVD)
(continued)

Example: 2-dimensional case for zero-mean random variables

$$\boldsymbol{C}_x = \begin{bmatrix} \mathbb{E}\{x_1^2\} & \mathbb{E}\{x_1 x_2\} \\ \mathbb{E}\{x_2 x_1\} & \mathbb{E}\{x_2^2\} \end{bmatrix}$$

$$\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2]$$

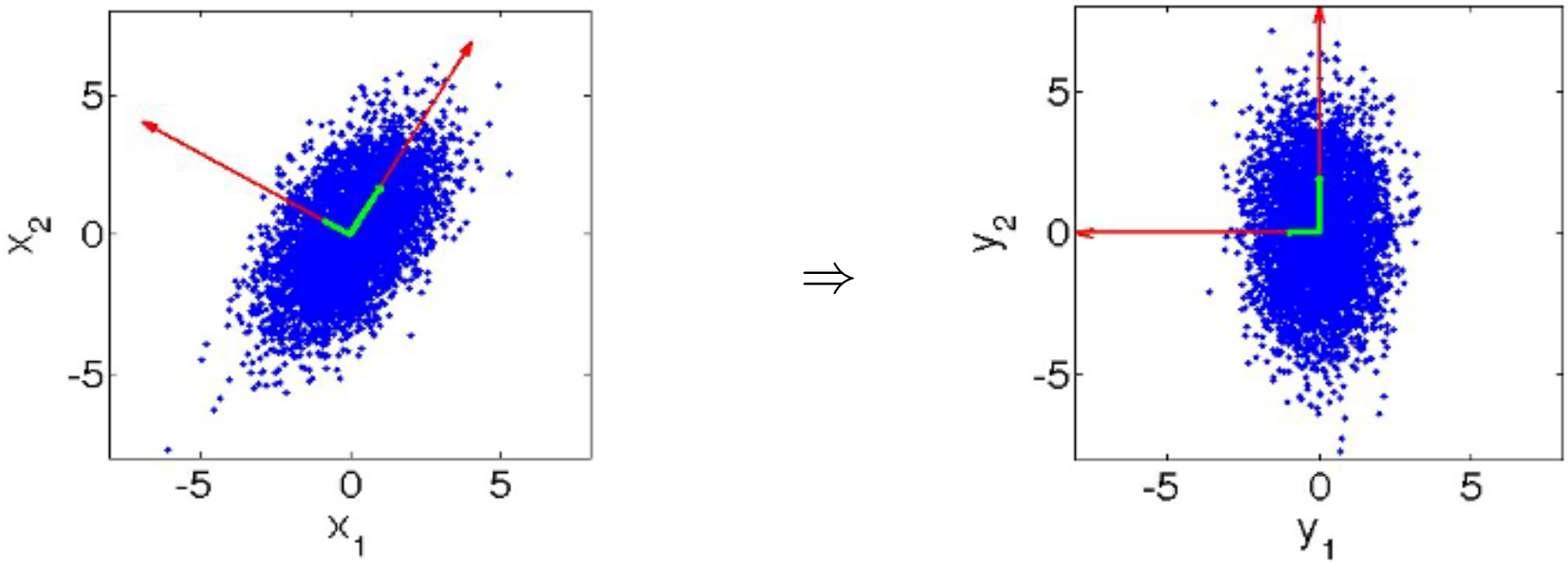
$$\boldsymbol{\Lambda}_x = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



Principal component analysis (PCA)

- **Principal component analysis (PCA)**, also known as the (discrete) **Karhunen-Loève transform** or the **Hotelling transform**, uses EVD to diagonalize the covariance matrix (decorrelates the data).
- If $\boldsymbol{U}$ is the eigen-matrix of $\boldsymbol{C}_x = \mathbb{E}\{\boldsymbol{x}\boldsymbol{x}^T\}$, the transforation $\boldsymbol{y} = \boldsymbol{U}^T\boldsymbol{x}$ results in:

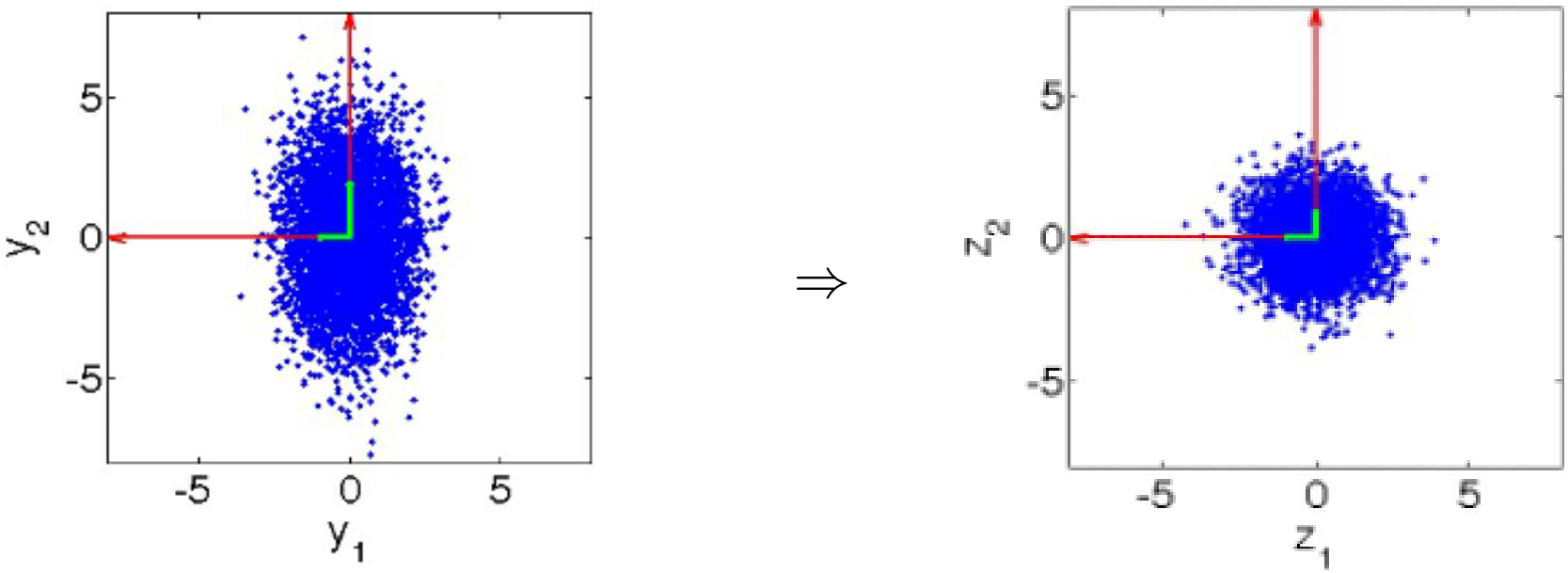
$$\boldsymbol{C}_y = \mathbb{E}\{\boldsymbol{y}\boldsymbol{y}^T\} = \boldsymbol{U}^T\boldsymbol{C}_x\boldsymbol{U} = \boldsymbol{\Lambda}$$



Sphering

- In the previous example, let's go a step further and normalize the covariance matrix via $\boldsymbol{z} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{y}$, resulting in:

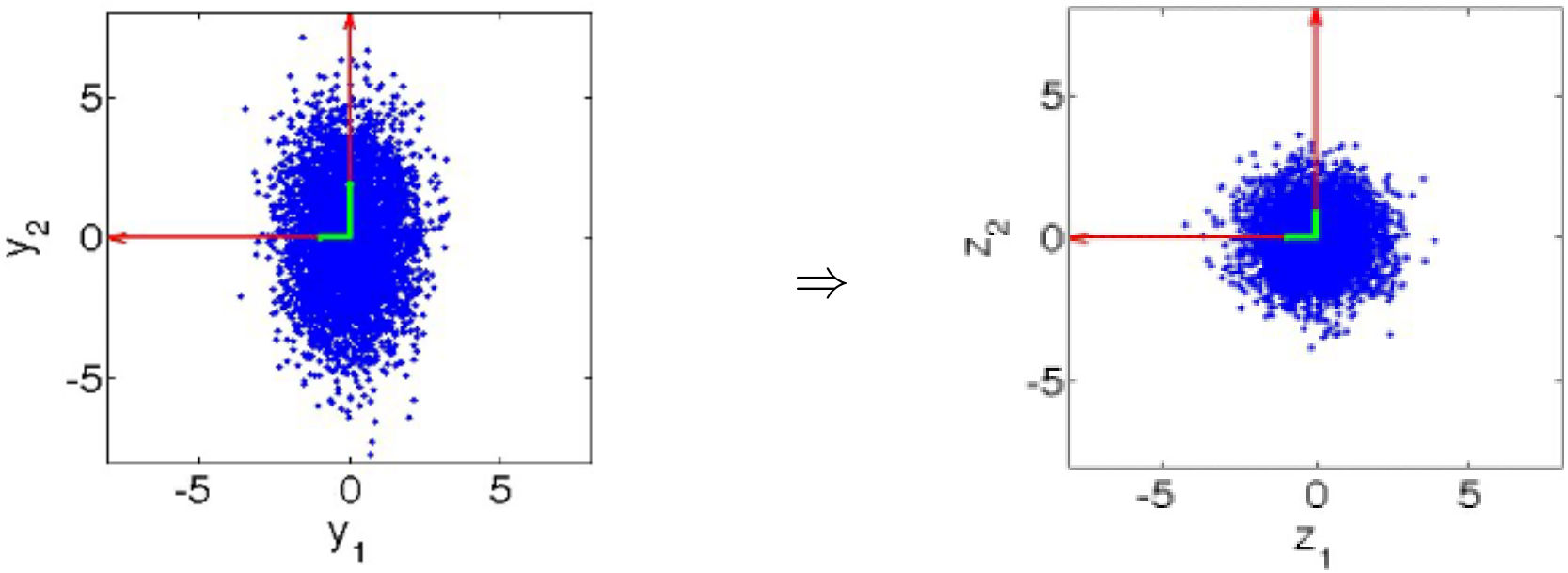
$$\boldsymbol{C}_z = \mathbb{E}\{\boldsymbol{z}\boldsymbol{z}^T\} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{C}_y\boldsymbol{\Lambda}^{-1/2} = \boldsymbol{I}$$



Sphering

- In the previous example, let's go a step further and normalize the covariance matrix via $\boldsymbol{z} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{y}$, resulting in:

$$\boldsymbol{C}_z = \mathbb{E}\{\boldsymbol{z}\boldsymbol{z}^T\} = \boldsymbol{\Lambda}^{-1/2}\boldsymbol{C}_y\boldsymbol{\Lambda}^{-1/2} = \boldsymbol{I}$$



Note: There is no further information in the second-order statistics of $\boldsymbol{z}$, i.e., multiplication of $\boldsymbol{z}$ by any orthonormal (rotation) matrix preserves its whiteness.

Uncorrelatedness vs independence of random variables

Reminders from statistics

Reminder (Hyvarinen et al., 2001)

- **Uncorrelatedness** of random variables x and y means:
  $\mathbb{E}\{xy\} = \mathbb{E}\{x\}\mathbb{E}\{y\}$

Reminder (Hyvarinen et al., 2001)

- **Uncorrelatedness** of random variables x and y means: $\mathbb{E}\{xy\} = \mathbb{E}\{x\}\mathbb{E}\{y\}$
- **Independence** of x and y means: $f_{XY}(x, y) = f_X(x)f_Y(y)$ resulting in $\mathbb{E}\{g(x)h(y)\} = \mathbb{E}\{g(x)\}\mathbb{E}\{h(y)\}$ for (almost) arbitrary $g(\cdot)$ and $h(\cdot)$

Reminder (Hyvarinen et al., 2001)

- **Uncorrelatedness** of random variables x and y means: $\mathbb{E}\{xy\} = \mathbb{E}\{x\}\mathbb{E}\{y\}$
- **Independence** of x and y means: $f_{XY}(x, y) = f_X(x)f_Y(y)$ resulting in $\mathbb{E}\{g(x)h(y)\} = \mathbb{E}\{g(x)\}\mathbb{E}\{h(y)\}$ for (almost) arbitrary $g(\cdot)$ and $h(\cdot)$
- Hence, independence is far stronger than uncorrelatedness and relies on **higher order statistics**.

Reminder (Hyvarinen et al., 2001)

- **Uncorrelatedness** of random variables x and y means: $\mathbb{E}\{xy\} = \mathbb{E}\{x\}\mathbb{E}\{y\}$
- **Independence** of x and y means: $f_{XY}(x, y) = f_X(x)f_Y(y)$ resulting in $\mathbb{E}\{g(x)h(y)\} = \mathbb{E}\{g(x)\}\mathbb{E}\{h(y)\}$ for (almost) arbitrary $g(\cdot)$ and $h(\cdot)$
- Hence, independence is far stronger than uncorrelatedness and relies on **higher order statistics**.
- For Gaussian random variables, uncorrelatedness is equivalent with independence. Therefore, their higher order statistics do not contain any additional information.

 Non-gaussianity of the sources is a key point for conventional ICA algorithms.

Linear transforms of non-Gaussian random variables

Example: Suppose that $s_1 \sim U(-0.5, 0.5)$ and $s_2 \sim U(-1.0, 1.0)$ be zero-mean independent uniformly distributed random variables.

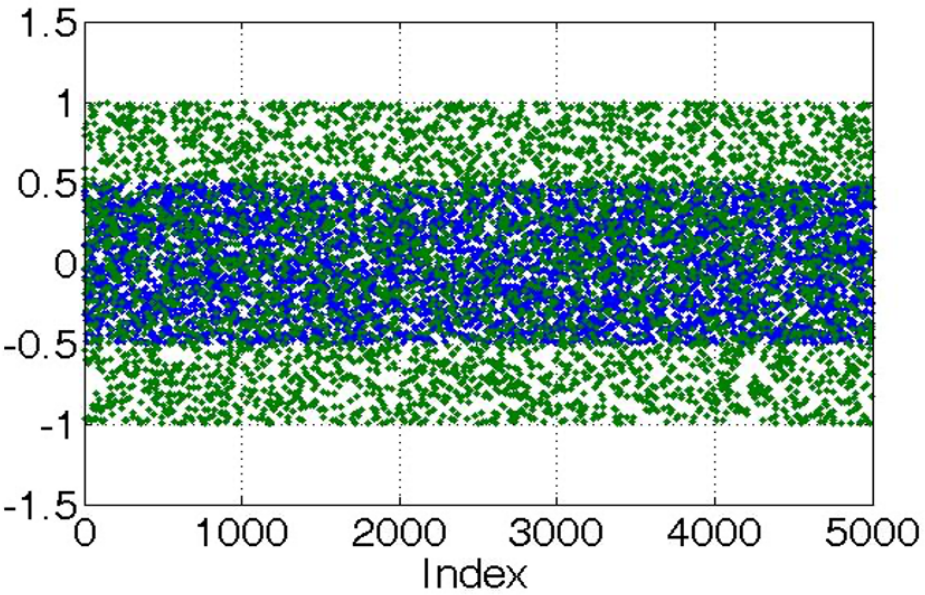


Figure: Sample plot

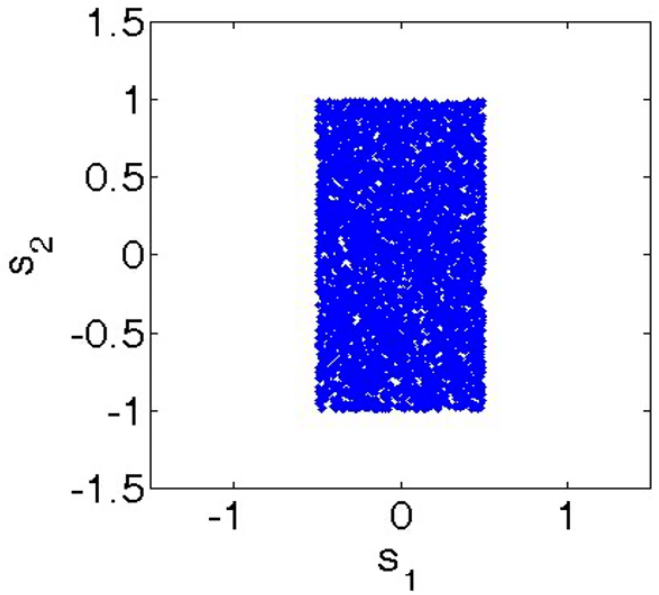
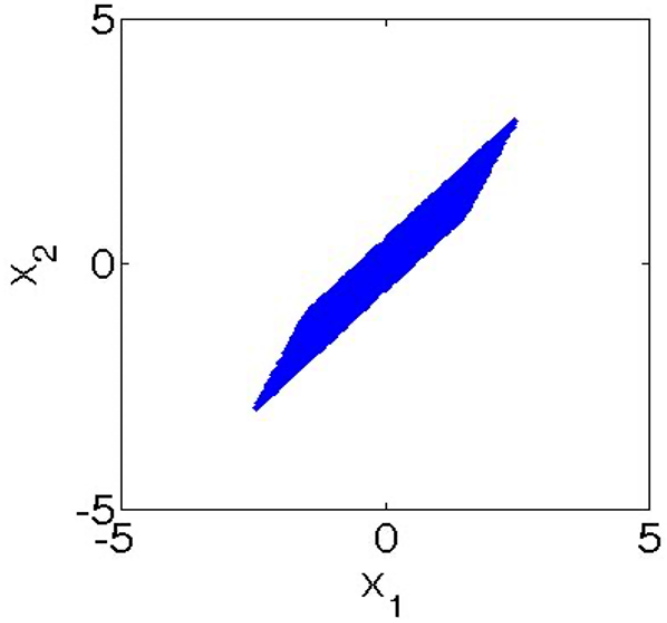


Figure: Scatter plot

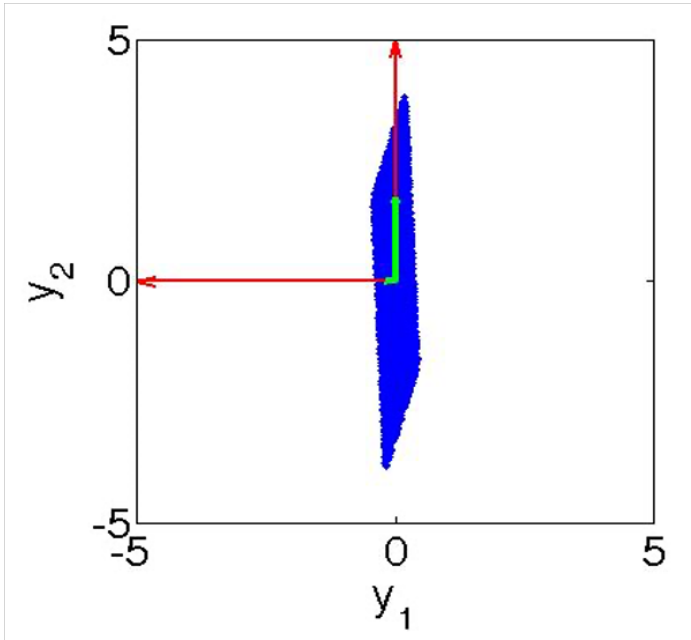
Linear transforms of non-Gaussian random variables
(continued)

Example: An arbitrary linear transform of (s_1, s_2) will stretch and rotate the mixture

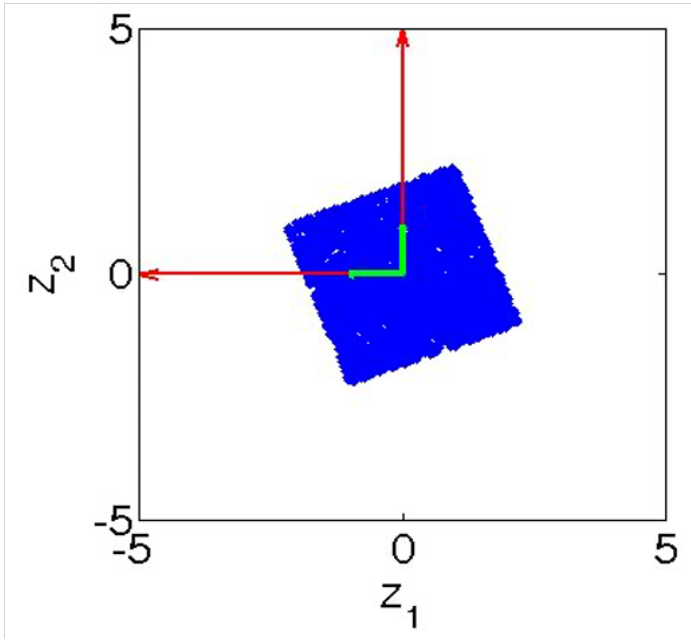
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1.0 & 2.0 \\ 2.0 & 2.0 \end{bmatrix} \begin{bmatrix} s_1 \\ s_2 \end{bmatrix}$$



Linear transforms of non-Gaussian random variables
(continued)



After PCA



After sphering

- There is still some **mutual information** in the joint distribution of z_1 and z_2, which can be used for their separation.
- Therefore, **sphering** is only half the way to **independence** for non-Gaussian variables.

From principal component analysis (PCA) to independent component analysis (ICA)

$$\boldsymbol{x} \xrightarrow[\text{(rotation)}]{\text{PCA}} \boldsymbol{y} \xrightarrow[\text{(scaling)}]{\text{Sphering}} \boldsymbol{z} \xrightarrow[\text{(rotation)}]{\text{ICA}} \boldsymbol{s}$$

- Multiplication of $\boldsymbol{z}$ by any orthonormal matrix preserves its whiteness (second order statistics) and will only change its higher order statistics.
- ICA can use the additional information in the higher-order statistics to separate the sources
- How to find the ICA rotation transform?

Reminder (Papoulis and Pillai, 2002)

- The summation of N **independent identically distributed (IID)** random variables, tends to **Gaussian** as N grows, regardless of their original distributions.
- So the summation of two random variables is "more Gaussian" than the original distribution.

In terms of ICA **non-Gaussian** is **independent**. In practice, we need measures of "independence" and "gaussianity" to quantify these terms for random variables.

How does classical ICA work?

A simplified perspective

Data model & assumptions:

- $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} = \sum_{i=1}^{N} \boldsymbol{a}_i s_i$, where $\boldsymbol{s} = [s_1, \ldots, s_N]$ is the source vector and $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N]$ is the **mixing matrix**
- The sources are statistically independent: $f_{\boldsymbol{S}}(\boldsymbol{s}) = \prod_{i=1}^{N} f_{\boldsymbol{S}_i}(s_i)$
- The sources are non-Gaussian distributed (except at most one)

How does classical ICA work?

A simplified perspective

Data model & assumptions:

- $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} = \sum_{i=1}^{N} \boldsymbol{a}_i s_i$, where $\boldsymbol{s} = [s_1, \ldots, s_N]$ is the source vector and $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N]$ is the **mixing matrix**
- The sources are statistically independent: $f_{\boldsymbol{S}}(\boldsymbol{s}) = \prod_{i=1}^{N} f_{\boldsymbol{S}_i}(s_i)$
- The sources are non-Gaussian distributed (except at most one)

According to the central limit theorem, a linear mixture of the observations

$$y = \sum_{i=1}^{N} b_i x_i = \boldsymbol{b}^T \boldsymbol{x} = \boldsymbol{b}^T \boldsymbol{A}\boldsymbol{s} = \boldsymbol{b}^T \sum_{i=1}^{N} \boldsymbol{a}_i s_i = \sum_{i=1}^{N} w_i s_i = \boldsymbol{w}^T \boldsymbol{s}$$

is "more Gaussian" than any of the components of $\boldsymbol{s}$, except when only one of the elements of $\boldsymbol{w}$ is nonzero, or $y = \hat{s}_p$ (an estimate of one of the sources).

How does classical ICA work?

A simplified perspective

Data model & assumptions:

- $\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} = \sum_{i=1}^{N} \boldsymbol{a}_i s_i$, where $\boldsymbol{s} = [s_1, \ldots, s_N]$ is the source vector and $\boldsymbol{A} = [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N]$ is the **mixing matrix**
- The sources are statistically independent: $f_{\boldsymbol{S}}(\boldsymbol{s}) = \prod_{i=1}^{N} f_{\boldsymbol{S}_i}(s_i)$
- The sources are non-Gaussian distributed (except at most one)

According to the central limit theorem, a linear mixture of the observations

$$y = \sum_{i=1}^{N} b_i x_i = \boldsymbol{b}^T \boldsymbol{x} = \boldsymbol{b}^T \boldsymbol{A}\boldsymbol{s} = \boldsymbol{b}^T \sum_{i=1}^{N} \boldsymbol{a}_i s_i = \sum_{i=1}^{N} w_i s_i = \boldsymbol{w}^T \boldsymbol{s}$$

is "more Gaussian" than any of the components of $\boldsymbol{s}$, except when only one of the elements of $\boldsymbol{w}$ is nonzero, or $y = \hat{s}_p$ (an estimate of one of the sources).

ICA Algorithm: update $\boldsymbol{b}$ in the direction of minimum gaussianity of y.

How does classical ICA work?
(continued)

ICA algorithm overview:

1. Apply PCA and sphering to remove second order correlations (optional but very helpful stage)
2. Define a measure of independence: Kurthosis (4th Order Cummulant), Negentropy, Mutual Information, etc.
3. Find the mixture $y = \boldsymbol{b}^T \boldsymbol{z}$, which maximize this measure to find independent sources

How does classical ICA work?
(continued)

ICA algorithm overview:

1. Apply PCA and sphering to remove second order correlations (optional but very helpful stage)
2. Define a measure of independence: Kurthosis (4th Order Cummulant), Negentropy, Mutual Information, etc.
3. Find the mixture $y = \boldsymbol{b}^T \boldsymbol{z}$, which maximize this measure to find independent sources

Question: Is there any guarantee that independent components obtained from this algorithm are the originally mixed components?

How does classical ICA work?
(continued)

ICA algorithm overview:

1. Apply PCA and sphering to remove second order correlations (optional but very helpful stage)
2. Define a measure of independence: Kurthosis (4th Order Cummulant), Negentropy, Mutual Information, etc.
3. Find the mixture $y = \boldsymbol{b}^T \boldsymbol{z}$, which maximize this measure to find independent sources

Question: Is there any guarantee that independent components obtained from this algorithm are the originally mixed components?
Answer: Yes, to some extent!

Limitations of classical ICA

- Requires the same number of sources as sensors

Limitations of classical ICA

- Requires the same number of sources as sensors
- Does not work for gaussian data (at most one of the sources can be gaussian)

Limitations of classical ICA

- Requires the same number of sources as sensors
- Does not work for gaussian data (at most one of the sources can be gaussian)
- Can estimate the sources, up to a "linear copy":
  1. Can't estimate the energy nor sign of the signal:

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} = (\boldsymbol{A}\alpha)(\alpha^{-1}\boldsymbol{s}) = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{s}}$$

  where α is an arbitrary diagonal matrix or scalar

Limitations of classical ICA

- Requires the same number of sources as sensors
- Does not work for gaussian data (at most one of the sources can be gaussian)
- Can estimate the sources, up to a "linear copy":
  1. Can't estimate the energy nor sign of the signal:

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} = (\boldsymbol{A}\alpha)(\alpha^{-1}\boldsymbol{s}) = \tilde{\boldsymbol{A}}\tilde{\boldsymbol{s}}$$

     where α is an arbitrary diagonal matrix or scalar
  2. Can't estimate the order of the signals:

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} = (\boldsymbol{A}\boldsymbol{P})(\boldsymbol{P}^{-1}\boldsymbol{s})$$

     where $\boldsymbol{P}$ is an arbitrary permutation matrix.

A fast converging ICA algorithm

The FastICA algorithm

1. Given $\boldsymbol{x}$, pre-whiten the data by PCA and sphering to obtain z
2. Select a non-quadratic nonlinear function $f(\cdot)$, with its first derivative $g(\cdot)$, second derivative $\dot{g}(\cdot)$, and an initial weight vector $\boldsymbol{w}$.
3. Repeat until convergence:
    1. Let $\boldsymbol{w} \leftarrow \mathbb{E}\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\} - \mathbb{E}\{\dot{g}(\boldsymbol{w}^T\boldsymbol{z})\}\boldsymbol{w}$, where $\mathbb{E}\{\cdot\}$ denotes expectation
    2. Let $\boldsymbol{w} \leftarrow \boldsymbol{w}/\|\boldsymbol{w}\|$

A fast converging ICA algorithm

The FastICA algorithm

1. Given $\boldsymbol{x}$, pre-whiten the data by PCA and sphering to obtain z
2. Select a non-quadratic nonlinear function $f(\cdot)$, with its first derivative $g(\cdot)$, second derivative $\dot{g}(\cdot)$, and an initial weight vector $\boldsymbol{w}$.
3. Repeat until convergence:
   1. Let $\boldsymbol{w} \leftarrow \mathbb{E}\{\boldsymbol{z}g(\boldsymbol{w}^T\boldsymbol{z})\} - \mathbb{E}\{\dot{g}(\boldsymbol{w}^T\boldsymbol{z})\}\boldsymbol{w}$, where $\mathbb{E}\{\cdot\}$ denotes expectation
   2. Let $\boldsymbol{w} \leftarrow \boldsymbol{w}/\|\boldsymbol{w}\|$

Examples of suitable nonlinear function

- $f(u) = \log(\cosh(u))$, $g(u) = \tanh(u)$ and $\dot{g} = 1 - \tanh^2(u)$
- $f(u) = -e^{-u^2/2}$, $g(u) = ue^{-u^2/2}$ and $\dot{g}(u) = (1 - u^2)e^{-u^2/2}$

Outline

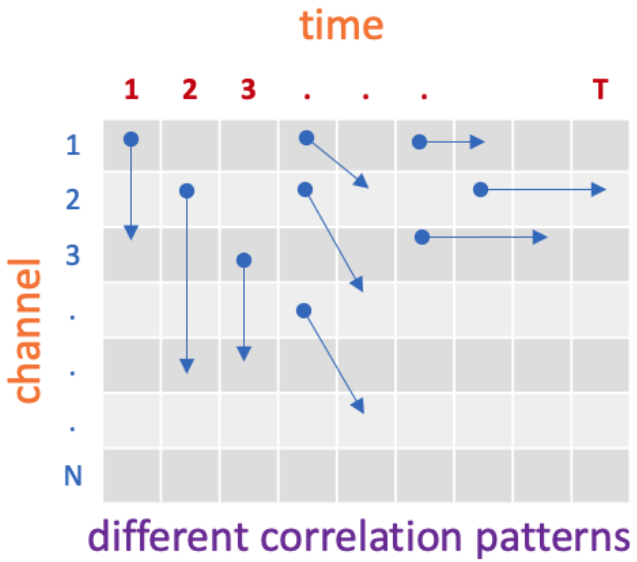
Let us restate BSS from a totally different viewpoint:

Degrees of freedom in linear transforms

- Consider the observations $\boldsymbol{x}(t) \in \mathbb{R}^{N \times T}$. We seek linear transforms of the form $\boldsymbol{y}(t) = \boldsymbol{W}\boldsymbol{x}(t)$ where $\boldsymbol{W} \in \mathbb{R}^{N \times N}$ is an unknown **separating matrix**.
- $\boldsymbol{W}$ acts as a **spatial filter** (N instantaneous linear combinations of the multichannel observations) that changes the properties of $\boldsymbol{y}(t)$
- The problem of linear source separation can be viewed as a **matrix design** problem with N^2 **degrees of freedom (DoF)**, to obtain $\boldsymbol{y}(t)$ with desired properties

Second and higher order statistics matrices

- Examples of matrices containing second and higher order statistics of a multichannel random process $\boldsymbol{x}(t)$ includes:
  - $\boldsymbol{C} = \mathbb{E}\{\boldsymbol{x}(t)\boldsymbol{x}^T(t)\}$: The covariance matrix
  - $\boldsymbol{C}(\tau) = \mathbb{E}\{\boldsymbol{x}(t)\boldsymbol{x}^T(t+\tau)\}$: The lagged covariance matrix
  - $\boldsymbol{C}_k = \mathbb{E}\{\boldsymbol{x}(t)\boldsymbol{x}^T(t)x_k(t)\}$: Slices of higher-order statistics



- This can more generally be extended to **tensors** of higher order statistics.
- These matrices carry average inter-channel similarities (correlation) over time
- Diagonalization of a number or all of these matrices results in the **decorrelation** and **independence** of the corresponding channels

Source separation by temporal decorrelation

- Consider $\boldsymbol{x}(t)$ with covariance matrix $\boldsymbol{C}_0 = \mathbb{E}\{\boldsymbol{x}(t)\boldsymbol{x}(t)^T\}$ and lagged-covariance matrices $\boldsymbol{C}_1 = \mathbb{E}\{\boldsymbol{x}(t)\boldsymbol{x}(t+\tau_1)^T\}$, $\boldsymbol{C}_2 = \mathbb{E}\{\boldsymbol{x}(t)\boldsymbol{x}(t+\tau_2)^T\}$, etc.
- For $\boldsymbol{y}(t) = \boldsymbol{W}\boldsymbol{x}(t)$:

$$\boldsymbol{A}_0 = \mathbb{E}\{\boldsymbol{y}(t)\boldsymbol{y}(t)^T\} = \mathbb{E}\{\boldsymbol{W}\boldsymbol{x}(t)\boldsymbol{x}(t)^T\boldsymbol{W}^T\} = \boldsymbol{W}\boldsymbol{C}_0\boldsymbol{W}^T$$
$$\cdots$$
$$\boldsymbol{A}_k = \mathbb{E}\{\boldsymbol{y}(t)\boldsymbol{y}(t+\tau_k)^T\} = \mathbb{E}\{\boldsymbol{W}\boldsymbol{x}(t)\boldsymbol{x}(t+\tau_k)^T\boldsymbol{W}^T\} = \boldsymbol{W}\boldsymbol{C}_k\boldsymbol{W}^T$$

- These matrices show the inter-channel correlations of the output $\boldsymbol{y}(t)$, at different time lags.
- For temporally correlated data, diagonalizing all such matrices results in channel-wise independence of $\boldsymbol{y}(t)$.

Source separation by temporal decorrelation

- Consider $\boldsymbol{x}(t)$ with covariance matrix $\boldsymbol{C}_0 = \mathbb{E}\{\boldsymbol{x}(t)\boldsymbol{x}(t)^T\}$ and lagged-covariance matrices $\boldsymbol{C}_1 = \mathbb{E}\{\boldsymbol{x}(t)\boldsymbol{x}(t+\tau_1)^T\}$, $\boldsymbol{C}_2 = \mathbb{E}\{\boldsymbol{x}(t)\boldsymbol{x}(t+\tau_2)^T\}$, etc.
- For $\boldsymbol{y}(t) = \boldsymbol{W}\boldsymbol{x}(t)$:

$$\boldsymbol{A}_0 = \mathbb{E}\{\boldsymbol{y}(t)\boldsymbol{y}(t)^T\} = \mathbb{E}\{\boldsymbol{W}\boldsymbol{x}(t)\boldsymbol{x}(t)^T\boldsymbol{W}^T\} = \boldsymbol{W}\boldsymbol{C}_0\boldsymbol{W}^T$$
$$\cdots$$
$$\boldsymbol{A}_k = \mathbb{E}\{\boldsymbol{y}(t)\boldsymbol{y}(t+\tau_k)^T\} = \mathbb{E}\{\boldsymbol{W}\boldsymbol{x}(t)\boldsymbol{x}(t+\tau_k)^T\boldsymbol{W}^T\} = \boldsymbol{W}\boldsymbol{C}_k\boldsymbol{W}^T$$

- These matrices show the inter-channel correlations of the output $\boldsymbol{y}(t)$, at different time lags.
- For temporally correlated data, diagonalizing all such matrices results in channel-wise independence of $\boldsymbol{y}(t)$.

Question: Can we find a matrix $\boldsymbol{W}$, which simultaneously diagonalizes all or a number of the above matrices?

Degrees-of-freedom (DoF) in algebraic transforms

Reminder: eigenvalue decomposition (EVD)

Using EVD, a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ can be decomposed as:

$$\boldsymbol{C}_0 = \boldsymbol{U} \boldsymbol{\Lambda} \boldsymbol{U}^T \tag{6}$$

where

- $\boldsymbol{C}_0$ is a symmetric matrix with $N(N+1)/2$ DoF
- $\boldsymbol{\Lambda}$ is a diagonal matrix with N DoF
- $\boldsymbol{U}$ is the orthonormal eigen-matrix with $N(N-1)/2$ DoF

Therefore

$$\texttt{DoF}(\boldsymbol{C}_0) = \texttt{DoF}(\Lambda) + \texttt{DoF}(\boldsymbol{U})$$

Degrees-of-freedom (DoF) in algebraic transforms
(continued)

Reminder: generalized eigenvalue decomposition (GEVD)

Two symmetric matrices $\boldsymbol{C}_0$ and $\boldsymbol{C}_1$ can be diagonalized using GEVD as follows:

$$\begin{aligned} \boldsymbol{W}\boldsymbol{C}_0\boldsymbol{W}^T &= \boldsymbol{I} \\ \boldsymbol{W}\boldsymbol{C}_1\boldsymbol{W}^T &= \boldsymbol{\Lambda} \end{aligned} \tag{7}$$

where

- $\boldsymbol{C}_0$ and $\boldsymbol{C}_1$ are symmetric matrices, each with $N(N+1)/2$ DoF
- $\boldsymbol{\Lambda}$ is a diagonal matrix with N DoF
- $\boldsymbol{W}$ is a square matrix with N^2 DoF
- $\boldsymbol{I}$ is the identity matrix with no DoF

Therefore

$$\texttt{DoF}(\boldsymbol{C}_0) + \texttt{DoF}(\boldsymbol{C}_1) = \texttt{DoF}(\Lambda) + \texttt{DoF}(\boldsymbol{W})$$

Degrees-of-freedom (DoF) in algebraic transforms
(continued)

Reminder: joint diagonalization of multiple matrices

No more than two symmetric matrices can be simultaneously diagonalized, except when they belong to the same eigen-space:

$$\begin{aligned} \boldsymbol{W}\boldsymbol{C}_0\boldsymbol{W}^T &= \boldsymbol{I} \\ \boldsymbol{W}\boldsymbol{C}_1\boldsymbol{W}^T &= \boldsymbol{\Lambda}_1 \\ &\cdots \\ \boldsymbol{W}\boldsymbol{C}_K\boldsymbol{W}^T &= \boldsymbol{\Lambda}_K \end{aligned} \tag{8}$$

For more than two matrices only **approximate joint diagonalization (AJD)** may be achieved.

Source separation by matrix diagonalization

Based on the above properties, two classes of algebraic source separation methods are common:

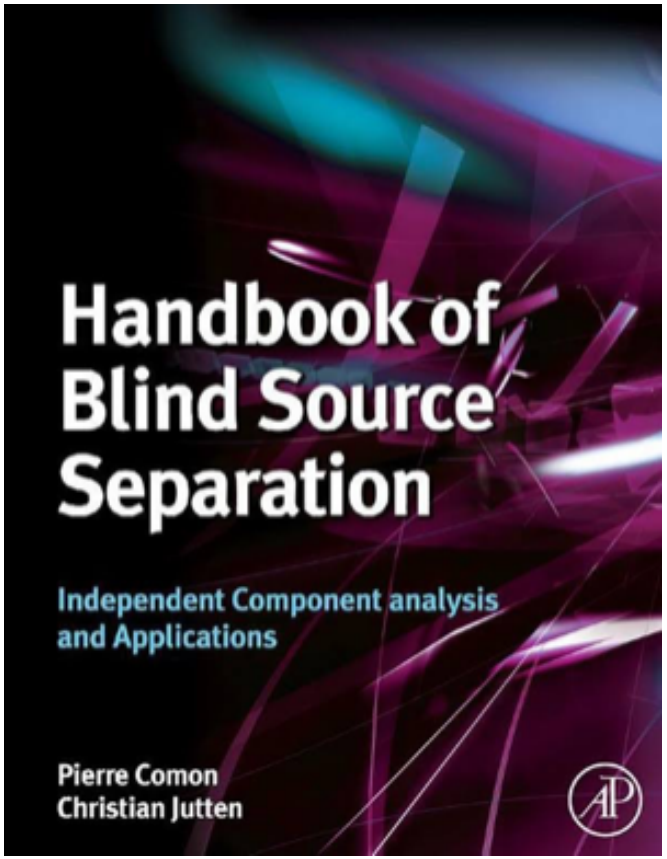
1. Exact diagonalization of two "well-chosen" matrices:
   - Advantage: The corresponding statistics are fully separated
   - Disadvantage: Requires prior information for selecting the matrices
   - Major methods of this class: AMUSE, ΠCA, etc.
2. Approximate joint diagonalization of more than two matrices:
   - Advantage: It is easier to select the matrices
   - Disadvantage: The corresponding statistics are only separated approximately; measures of approximate diagonalization are not unique
   - Major methods of this class: JADE, SOBI, etc.

Outline

Textbooks on BSS and ICA

Further reading

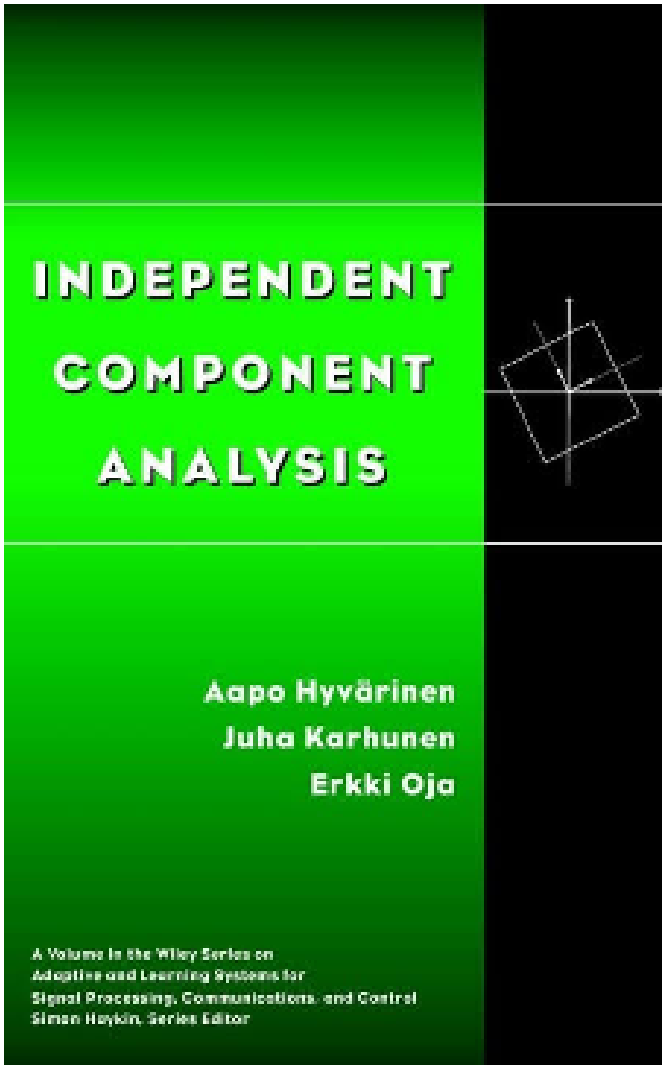
Comon, P., & Jutten, C. (Eds.). (2010). Handbook of Blind Source Separation: Independent component analysis and applications. Academic press. (Comon and Jutten, 2010)



Textbooks on BSS and ICA

Further reading

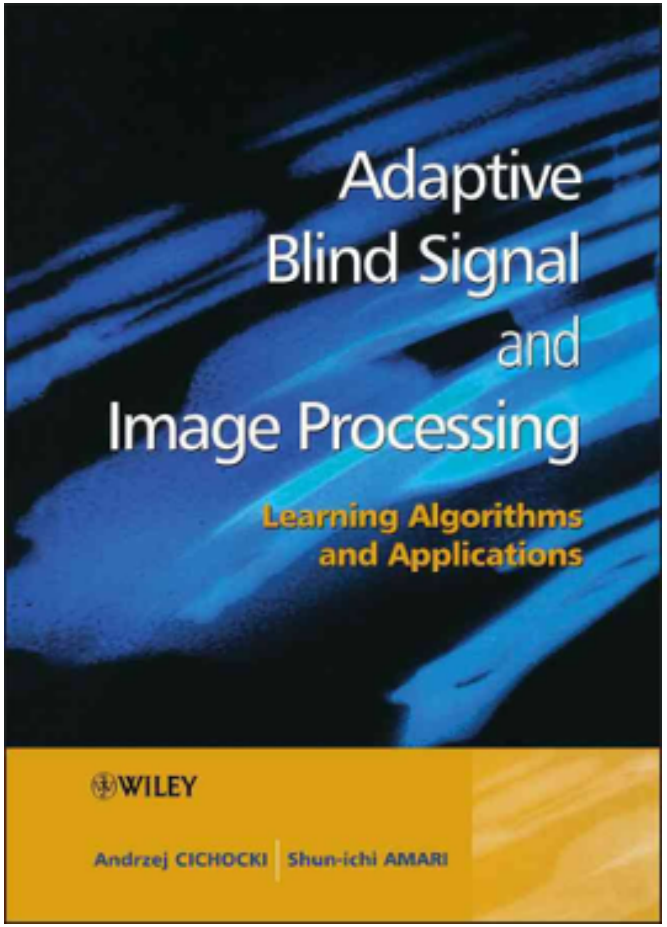
Hyvärinen, A., Karhunen, J., & Oja, E. (2004). Independent component analysis, John Wiley & Sons.(Hyvarinen et al., 2001)



Textbooks on BSS and ICA

Further reading

Cichocki, A., & Amari, S. I. (2002). Adaptive blind signal and image processing: learning algorithms and applications (Vol. 1). John Wiley & Sons. (Cichocki and Amari, 2003)



Source codes and algorithms to consider

- FastICA. Online available: `http://research.ics.aalto.fi/ica/fastica/`
- Joint Approximation Diagonalization of Eigen-matrices (JADE): `http://www2.iap.fr/users/cardoso/compsep_classic.html`
- Second order blind identification (SOBI)
- The Open-Source Electrophysiological Toolbox (OSET): `www.oset.ir`

Advanced topics

- Semi-blind source separation: temporal structure, sparsity, periodicity, etc.
- Noisy mixtures; integration of filtering and source separation
- Approximate joint diagonalization methods
- Nonlinear mixtures
- Sparse component analysis
- Distributed (non-punctual) sources
- Convolutive mixtures
- Online ICA and time-variant mixtures
- Curse of dimensionality and implementation issues
- BSS using classical estimation techniques (MAP, ML, CRLB)
- Tensor decomposition for BSS

Thank you! Questions?

Outline

BSS lab session

preface

Algorithms:

- PCA
- GEVD: AMUSE, ΠCA, NSCA
- JADE
- FastICA

Case Studies:

- Audio source separation
- Noninvasive fetal ECG extraction
- EOG artifact cancellation from EEG
- Device noise and artifact cancellation from biomedical signals

BSS lab session

requirements

Clone the OSET from either repositories:

- `https://gitlab.com/rsameni/OSET.git` (or its image repository on GitHub: `https://github.com/alphanumericslab/OSET.git`)

Clone the tutorial for efficient Matlab coding (optional):

- `https://github.com/rsameni/TechReport-EfficientMatlabCodeTutorial.git`

Clone the lab sample codes (publicly available after the lecture):

- `https://github.com/rsameni/BSSLecture.git`

Principal component analysis algorithms

Reminder

By definition, the eigenvalues and eigenvectors of a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ satisfy:

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v} \tag{9}$$

Principal component analysis algorithms

Reminder

By definition, the eigenvalues and eigenvectors of a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ satisfy:

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v} \tag{9}$$

Exercise

1. Calculate the eigenvalues and eigenvectors of $\boldsymbol{A} = \begin{bmatrix} 3 & 1 \\ 1 & 4 \end{bmatrix}$

   How?

Principal component analysis algorithms

Reminder

By definition, the eigenvalues and eigenvectors of a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ satisfy:

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v} \tag{9}$$

Exercise

1. Calculate the eigenvalues and eigenvectors of $\boldsymbol{A} = \begin{bmatrix} 3 & 1 \\ 1 & 4 \end{bmatrix}$

   How? Find the roots of the characteristic polynomial $\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = 0$; then use them to obtain the eigenvectors from (9).

   Note: Impose $\|\boldsymbol{v}\| = 1$ to obtain unique results.

Principal component analysis algorithms

Reminder

By definition, the eigenvalues and eigenvectors of a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ satisfy:

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v} \tag{9}$$

Exercise

1. Calculate the eigenvalues and eigenvectors of $\boldsymbol{A} = \begin{bmatrix} 3 & 1 \\ 1 & 4 \end{bmatrix}$

   How? Find the roots of the characteristic polynomial $\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = 0$; then use them to obtain the eigenvectors from (9).

   Note: Impose $\|\boldsymbol{v}\| = 1$ to obtain unique results.

2. Use the function `eig` in Matlab to check the result

Principal component analysis algorithms

Reminder

By definition, the eigenvalues and eigenvectors of a symmetric matrix $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ satisfy:

$$\boldsymbol{A}\boldsymbol{v} = \lambda \boldsymbol{v} \tag{9}$$

Exercise

1. Calculate the eigenvalues and eigenvectors of $\boldsymbol{A} = \begin{bmatrix} 3 & 1 \\ 1 & 4 \end{bmatrix}$

   How? Find the roots of the characteristic polynomial $\det(\boldsymbol{A} - \lambda \boldsymbol{I}) = 0$; then use them to obtain the eigenvectors from (9).

   Note: Impose $\|\boldsymbol{v}\| = 1$ to obtain unique results.
2. Use the function `eig` in Matlab to check the result
3. Generate 10000 ensembles of a Gaussian random vector with covariance matrix equal to $\boldsymbol{A}$
4. How can we calculate the eigenvalues and eigenvectors numerically?

Principal component analysis algorithms
(continued)

A simple (but inefficient) method for calculating the largest eigenvalue and its corresponding eigenvector is:

The power method for EVD

1. Initialize by a random vector $\boldsymbol{v}_0 \in \mathbb{R}^{N \times N}$
2. Repeat until convergence: $\boldsymbol{v}_{k+1} = \frac{\boldsymbol{A}\boldsymbol{v}_k}{\|\boldsymbol{A}\boldsymbol{v}_k\|}$
3. $\lambda = \|\boldsymbol{A}\boldsymbol{v}\| / \|\boldsymbol{v}\|$

Principal component analysis algorithms
(continued)

A simple (but inefficient) method for calculating the largest eigenvalue and its corresponding eigenvector is:

The power method for EVD

1. Initialize by a random vector $\boldsymbol{v}_0 \in \mathbb{R}^{N \times N}$
2. Repeat until convergence: $\boldsymbol{v}_{k+1} = \frac{\boldsymbol{A}\boldsymbol{v}_k}{\|\boldsymbol{A}\boldsymbol{v}_k\|}$
3. $\lambda = \|\boldsymbol{A}\boldsymbol{v}\| / \|\boldsymbol{v}\|$

Question: Does the algorithm necessarily converge?

Principal component analysis algorithms
(continued)

A simple (but inefficient) method for calculating the largest eigenvalue and its corresponding eigenvector is:

The power method for EVD

1. Initialize by a random vector $\boldsymbol{v}_0 \in \mathbb{R}^{N \times N}$
2. Repeat until convergence: $\boldsymbol{v}_{k+1} = \frac{\boldsymbol{A}\boldsymbol{v}_k}{\|\boldsymbol{A}\boldsymbol{v}_k\|}$
3. $\lambda = \|\boldsymbol{A}\boldsymbol{v}\| / \|\boldsymbol{v}\|$

Question: Does the algorithm necessarily converge?
Question: How to measure the convergence?

Principal component analysis algorithms
(continued)

A simple (but inefficient) method for calculating the largest eigenvalue and its corresponding eigenvector is:

The power method for EVD

1. Initialize by a random vector $\boldsymbol{v}_0 \in \mathbb{R}^{N \times N}$
2. Repeat until convergence: $\boldsymbol{v}_{k+1} = \frac{\boldsymbol{A}\boldsymbol{v}_k}{\|\boldsymbol{A}\boldsymbol{v}_k\|}$
3. $\lambda = \|\boldsymbol{A}\boldsymbol{v}\| / \|\boldsymbol{v}\|$

Question: Does the algorithm necessarily converge?
Question: How to measure the convergence?
Question: How can we find the other eigenvalues and eigenvectors?

Principal component analysis algorithms
(continued)

Reminder: rank-one expansion for EVD (Strang, 1988)

Consider the EVD of a symmetric matrix $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$. It can be shown that $\boldsymbol{A}$ can be decomposed as a summation of **rank-one matrices**:

$$\boldsymbol{A} = \lambda_1 \boldsymbol{u}_1 \boldsymbol{u}_1^T + \ldots + \lambda_N \boldsymbol{u}_N \boldsymbol{u}_N^T$$

Principal component analysis algorithms
(continued)

Reminder: rank-one expansion for EVD (Strang, 1988)

Consider the EVD of a symmetric matrix $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$. It can be shown that $\boldsymbol{A}$ can be decomposed as a summation of **rank-one matrices**:

$$\boldsymbol{A} = \lambda_1 \boldsymbol{u}_1 \boldsymbol{u}_1^T + \ldots + \lambda_N \boldsymbol{u}_N \boldsymbol{u}_N^T$$

The above property can be used to extract the other eigenvalues and eigenvectors in the power method:

The power method (continued)

1. $\tilde{\boldsymbol{A}} = \boldsymbol{A} - \lambda_1 \boldsymbol{u}_1 \boldsymbol{u}_1^T$
2. Apply the power method on $\tilde{\boldsymbol{A}}$
3. Repeat the same procedure for all eigenvalues and eigenvectors

Principal component analysis algorithms
(continued)

Reminder: rank-one expansion for EVD (Strang, 1988)

Consider the EVD of a symmetric matrix $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$. It can be shown that $\boldsymbol{A}$ can be decomposed as a summation of **rank-one matrices**:

$$\boldsymbol{A} = \lambda_1 \boldsymbol{u}_1 \boldsymbol{u}_1^T + \ldots + \lambda_N \boldsymbol{u}_N \boldsymbol{u}_N^T$$

The above property can be used to extract the other eigenvalues and eigenvectors in the power method:

The power method (continued)

1. $\tilde{\boldsymbol{A}} = \boldsymbol{A} - \lambda_1 \boldsymbol{u}_1 \boldsymbol{u}_1^T$
2. Apply the power method on $\tilde{\boldsymbol{A}}$
3. Repeat the same procedure for all eigenvalues and eigenvectors

Drawbacks: slow convergence; error accumulation during rank-one updates

Generalized eigenvalue decomposition

Exercise

- Consider zero-mean multichannel electroencephalogram signals $\boldsymbol{x}(t) \in \mathbb{R}^{N \times N}$ with $t \in \mathcal{T} = \{1, \ldots, T\}$ and covariance matrix $\boldsymbol{C}_0 = \mathbb{E}_t\{\boldsymbol{x}(t)\boldsymbol{x}(t)^T\}$
- Suppose that $\mathcal{U} = \{u\} \subset \mathcal{T}$ is a subset of time samples corresponding to event-related potentials (ERP), with $\boldsymbol{C}_1 = \mathbb{E}_u\{\boldsymbol{x}(u)\boldsymbol{x}(u)^T\}$
- We seek $y(t) = \boldsymbol{w}^T\boldsymbol{x}(t)$, such that it has the maximum energy ratio during the ERP time epochs, while its total energy remains fixed.
- How to formulate the problem?

Generalized eigenvalue decomposition

Exercise

- Consider zero-mean multichannel electroencephalogram signals $\boldsymbol{x}(t) \in \mathbb{R}^{N \times N}$ with $t \in \mathcal{T} = \{1, \ldots, T\}$ and covariance matrix $\boldsymbol{C}_0 = \mathbb{E}_t\{\boldsymbol{x}(t)\boldsymbol{x}(t)^T\}$
- Suppose that $\mathcal{U} = \{u\} \subset \mathcal{T}$ is a subset of time samples corresponding to event-related potentials (ERP), with $\boldsymbol{C}_1 = \mathbb{E}_u\{\boldsymbol{x}(u)\boldsymbol{x}(u)^T\}$
- We seek $y(t) = \boldsymbol{w}^T\boldsymbol{x}(t)$, such that it has the maximum energy ratio during the ERP time epochs, while its total energy remains fixed.
- How to formulate the problem?

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \|\boldsymbol{w}^T\boldsymbol{x}(u)\|, \text{ s.t. } \|\boldsymbol{w}^T\boldsymbol{x}(t)\| = cte.$$

- Or equivalently

$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \boldsymbol{w}^T\boldsymbol{C}_1\boldsymbol{w}, \text{ s.t. } \boldsymbol{w}^T\boldsymbol{C}_0\boldsymbol{w} = cte. \tag{10}$$

Generalized eigenvalue decomposition
(continued)

The Rayleigh quotient and generalized eigenvalue decomposition (GEVD) (Strang, 1988)

For real symmetric matrices $\boldsymbol{C}_0$ and $\boldsymbol{C}_1$, the following problems are equivalent:

- $\boldsymbol{W}\boldsymbol{C}_0\boldsymbol{W}^T = \boldsymbol{I}$ and $\boldsymbol{W}\boldsymbol{C}_1\boldsymbol{W}^T = \boldsymbol{\Lambda}$, where $\boldsymbol{W} = [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N]$, $\boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_N)$ and $\lambda_1 \geq \ldots \geq \lambda_N$
- $\boldsymbol{w}_1 = \arg\max_{\boldsymbol{w}} r = \dfrac{\boldsymbol{w}^T\boldsymbol{C}_1\boldsymbol{w}}{\boldsymbol{w}^T\boldsymbol{C}_0\boldsymbol{w}}$ (known as the Rayleigh quotient)
- $\boldsymbol{w}_1 = \arg\max_{\boldsymbol{w}} \boldsymbol{w}^T\boldsymbol{C}_1\boldsymbol{w}$, s.t. $\boldsymbol{w}^T\boldsymbol{C}_0\boldsymbol{w} = cte.$

References

Cichocki, A. and Amari, S., eds (2003). Adaptive Blind Signal and Image Processing. John Wiley & Sons Inc.

Comon, P. and Jutten, C. (2010). Handbook of Blind Source Separation: Independent Component Analysis and Applications. Independent Component Analysis and Applications Series, Elsevier Science.

Hyvarinen, A., Karhunen, J. and Oja, E. (2001). Independent Component Analysis. Wiley-Interscience.

Papoulis, A. and Pillai, S. U. (2002). Probability, random variables, and stochastic processes. Fourth edition, McGraw-Hill.

Roll, J.-P. (1981). Contribution de la proprioception musculaire la perception et au contrôle du mouvement chez I'homme. PhD thesis, Uni. D'Aix-Marseille I. Thèse d'État.

Strang, G. (1988). Linear Algebra and Its Applications. 3 edition, Brooks/Cole.