# Machine Learning for Time Series

## Lecture 6: Multivariate Time Series

Laurent Oudre
laurent.oudre@ens-paris-saclay.fr

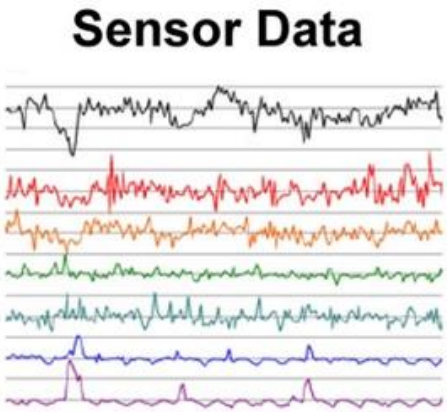
Master MVA
2023-2024

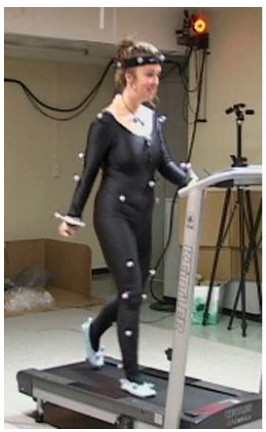
# Contents

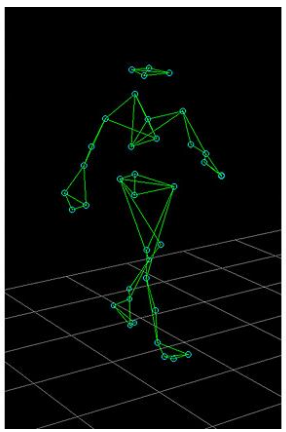
# Contents

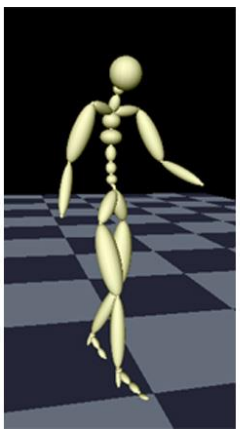
# Dealing with multivariate time series

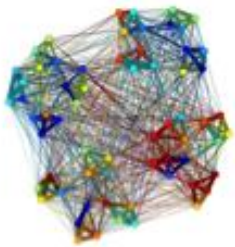

2D/3D trajectories,
Multivariate time series,
Multimodal data from sensor networks,
Graph signals

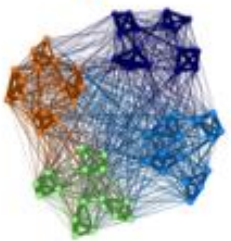


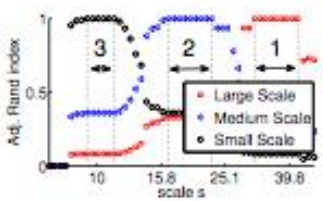













# Multivariate time series are everywhere

- Correlated quantities recorded simultaneously
  Ex: number of hospitalizations, number of patients in intensive care, number of deaths... during the COVID-19 pandemic
- 2D- and 3D- trajectories
  Ex: motion capture, mouse tracking, eye tracking
- Sensor networks
  Ex: environmental monitoring, wireless networks, social networks...

# How to process multivariate time series?

Two solutions:

- Process each dimension independently (see previous lectures)
- Use dedicated models or frameworks that take into account the links between the different dimensions

**Multivariate time series**

**Models and tools for learning from multivariate time series**

# Notations

- Time series of length $N$ with $D$ dimensions
- A multivariate sample $\mathbf{x}[n]$ is a vector of length $D$
- Multivariate time series can be stored in a matrix $\mathbf{X}$ of size $N \times D$

$$\mathbf{X} = \begin{bmatrix} x_1[1] & \cdots & x_D[1] \\ \vdots & \cdots & \vdots \\ x_1[N] & \cdots & x_D[N] \end{bmatrix}$$

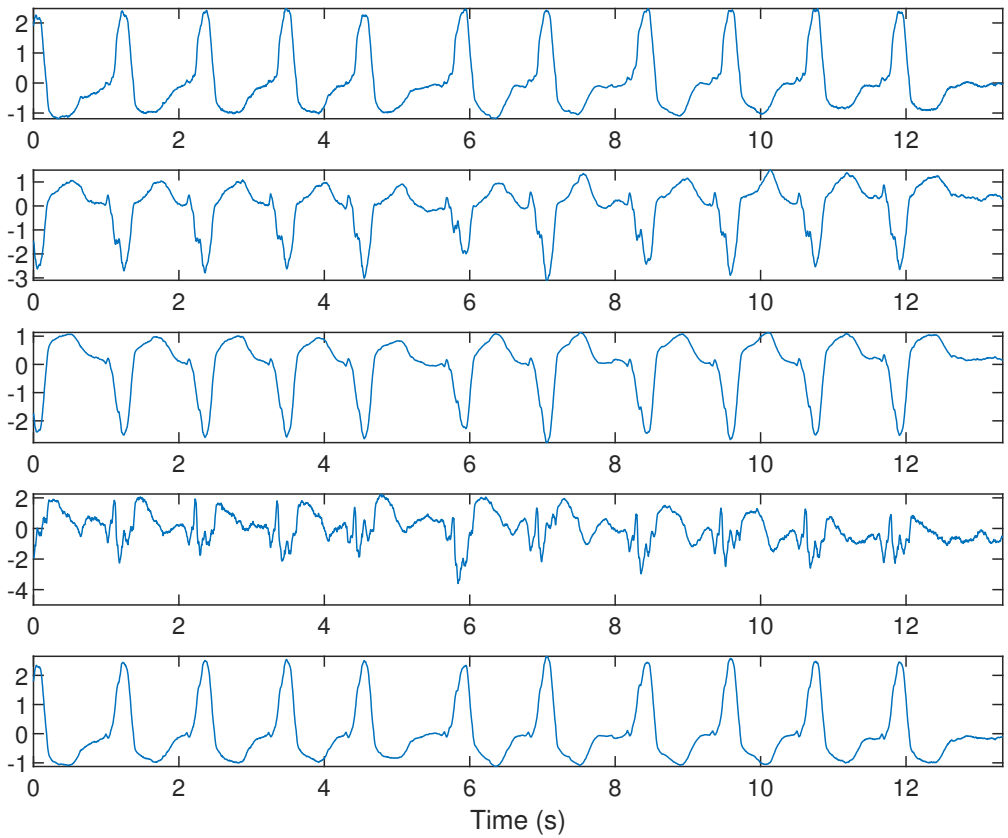
# Contents

# How to bypass the problem

In the literature, one common trick is to use PCA for dealing with multivariate time series

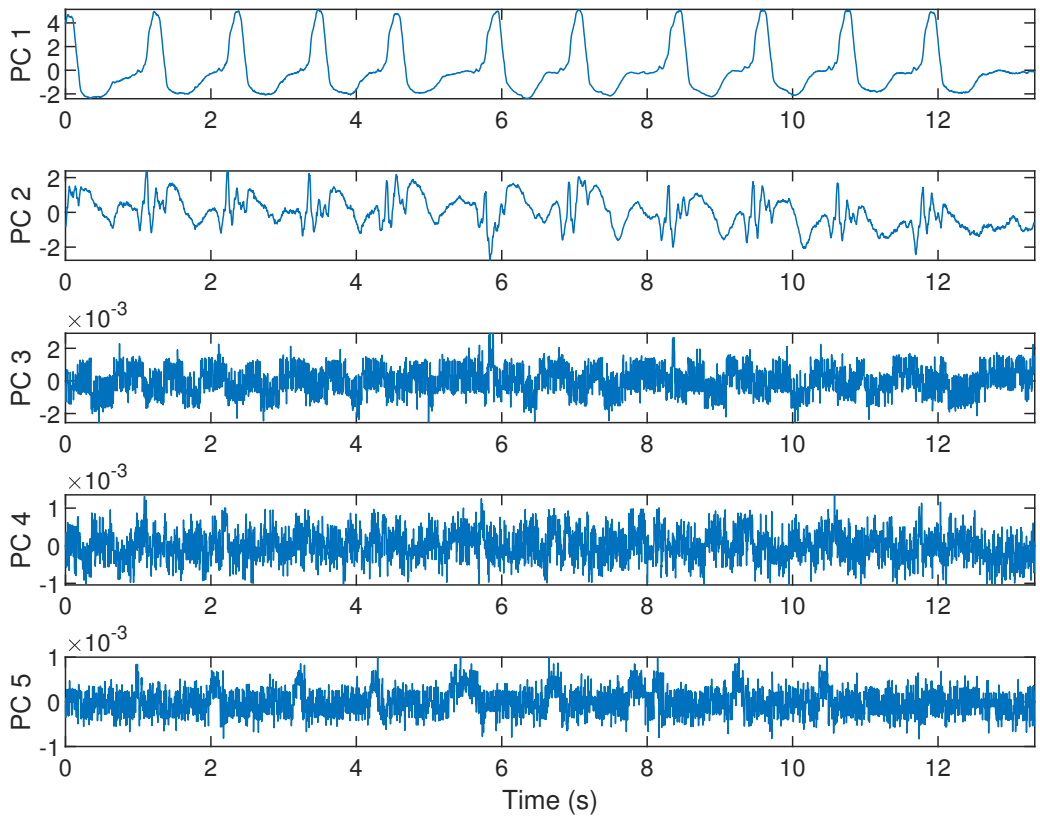
- Use PCA to compute the Principal Components and select those who contribute the most to the global variance
- Since the PCs are uncorrelated, they can be processed individually with univariate models and techniques
- Then, the results on all PCs can be merged to perform the desired task

# Example



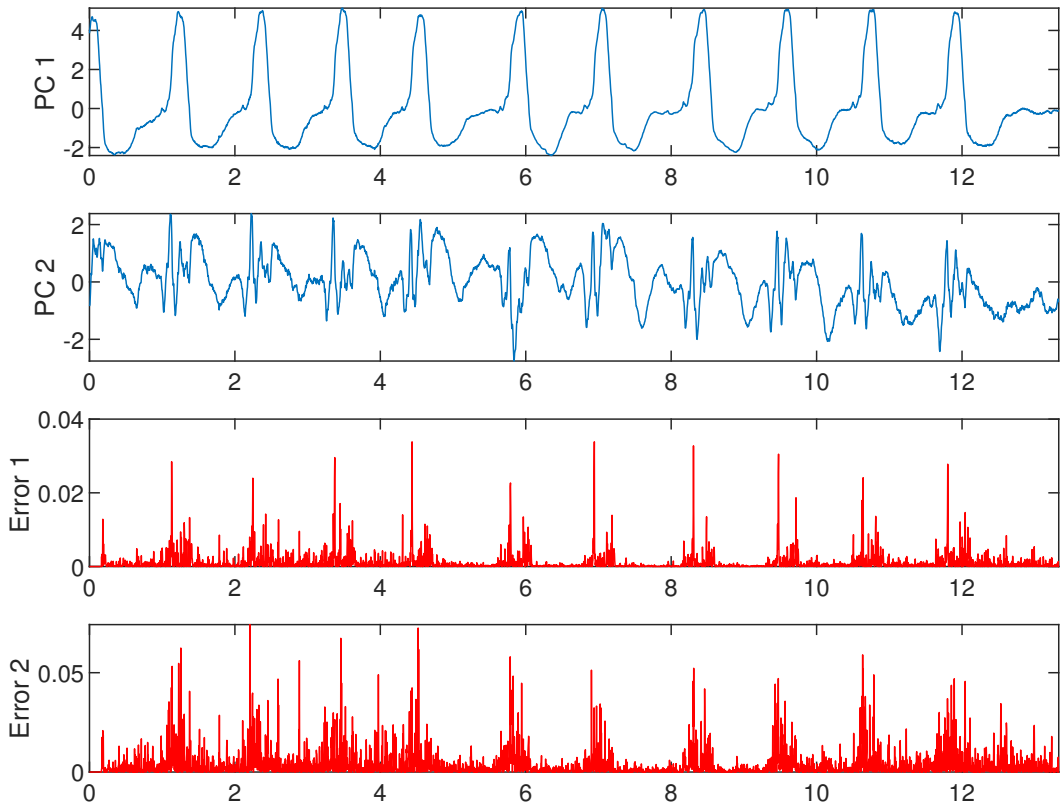

Original multivariate time series (ECG signals)

# Example



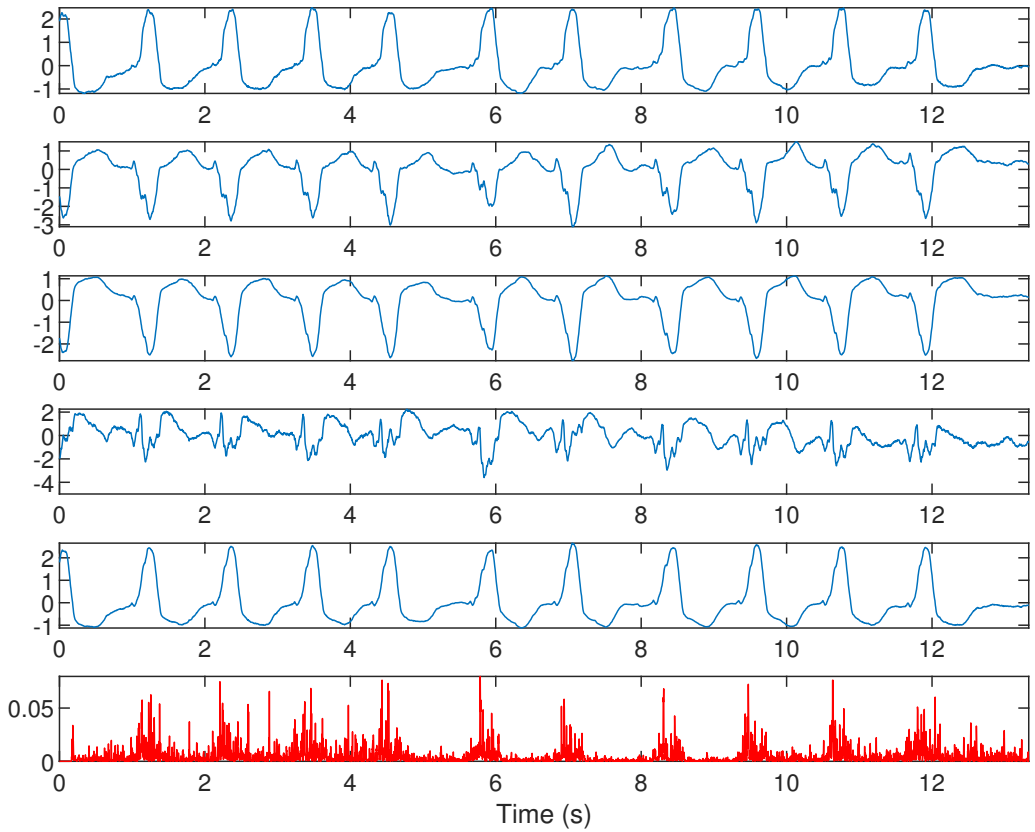

We compute the Principal Components: only the first two contribute to the variance

# Example




Each PC is modeled with an univariate AR(50) model. We compute the prediction error.

# Example




We sum up the two error terms to build an anomaly detector.

# Contents

# Models for multivariate time series

- Almost all models and approaches seen in this course have been extended to multivariate time series
- This extension can be straightforward or involve more complex data structures
- Some notions such as sparsity also have to be properly redefined so as to take into account an adequate notion of structure

1. Autoregressive models $\rightarrow$ Vector autoregressive models
2. Dictionary learning $\rightarrow$ Multivariate dictionary learning

# Vector Autoregressive model $VAR(p)$

$$\mathbf{x}[n] = \mathbf{c} + \sum_{i=1}^{p} \mathbf{A_i}\mathbf{x}[n-i] + \mathbf{b}[n]$$

- $p$ : order of the model
- $\mathbf{c}$ : intercept of size $D$
- $\mathbf{A_1}, \ldots, \mathbf{A_p}$ : VAR matrices. Each $\mathbf{A_i}$ is of size $D \times D$.
- $\mathbf{b}[n]$ : white noise (often called innovation)

[Zivot, 2006 ; Lütkepohl, 2013]

# Vector Autoregressive model $VAR(p)$

$$\mathbf{x}[n] = \mathbf{c} + \sum_{i=1}^{p} \mathbf{A_i}\mathbf{x}[n-i] + \mathbf{b}[n]$$

Noise structure

- $\forall d, \quad \mathbb{E}\left[b_d[n]\right] = 0$: zero-mean
- $\mathbb{E}\left[\mathbf{b}^T[n]\mathbf{b}[n]\right] = \mathbf{\Sigma_b}$: covariance matrix of size $D \times D$
- $\forall k \neq 0, \quad \mathbb{E}\left[\mathbf{b}^T[n]\mathbf{b}[n-k]\right] = 0$: uncorrelated samples for different times

# Example: 2-D *VAR*(1)

For $D = 2$ and $p = 1$, the model becomes:

$$\begin{cases} x_1[n] = c_1 + a_{1,1}x_1[n-1] + a_{1,2}x_2[n-1] + b_1[n] \\ x_2[n] = c_2 + a_{2,1}x_1[n-1] + a_{2,2}x_2[n-1] + b_2[n] \end{cases}$$

- Dependency with both previous samples of the current dimension and previous samples of the other dimensions
- Takes into account the possible correlations between dimensions
- Compared to $D$ univariate AR models: $D^2p$ vs. $Dp$ parameters ($+D$ for the intercept)

# Reformulation in the matrix form

By using the definition, it goes that

$$\begin{cases} \mathbf{x}[p+1] & = \mathbf{c} + \sum_{i=1}^{p} \mathbf{A_i} \mathbf{x}[p+1-i] + \mathbf{b}[p+1] \\ \vdots & \\ \mathbf{x}[N] & = \mathbf{c} + \sum_{i=1}^{p} \mathbf{A_i} \mathbf{x}[N-i] + \mathbf{b}[N] \end{cases}$$

# Reformulation in the matrix form

- By writing:

$$\mathbf{Y} = \begin{bmatrix} \mathbf{x}[p+1] \\ \vdots \\ \mathbf{x}[N] \end{bmatrix} \in \mathbb{R}^{(N-p)\times D} \quad \underline{\mathbf{A}} = \begin{bmatrix} \mathbf{c} \\ \mathbf{A_1} \\ \vdots \\ \mathbf{A_p} \end{bmatrix} \in \mathbb{R}^{(pD+1)\times D} \quad \mathbf{B} = \begin{bmatrix} \mathbf{b}[p+1] \\ \vdots \\ \mathbf{b}[N] \end{bmatrix} \in \mathbb{R}^{(N-p)\times D}$$

$$\mathbf{Z} = \begin{bmatrix} 1 & \mathbf{x}[p] & \cdots & \mathbf{x}[1] \\ 1 & \mathbf{x}[p+1] & \cdots & \mathbf{x}[2] \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \mathbf{x}[N-1] & \cdots & \mathbf{x}[N-p] \end{bmatrix} \in \mathbb{R}^{(N-p)\times(pD+1)}$$

- The model rewrites in a matrix form as

$$\mathbf{Y} = \mathbf{Z}\underline{\mathbf{A}} + \mathbf{B}$$

# Estimation of the parameters

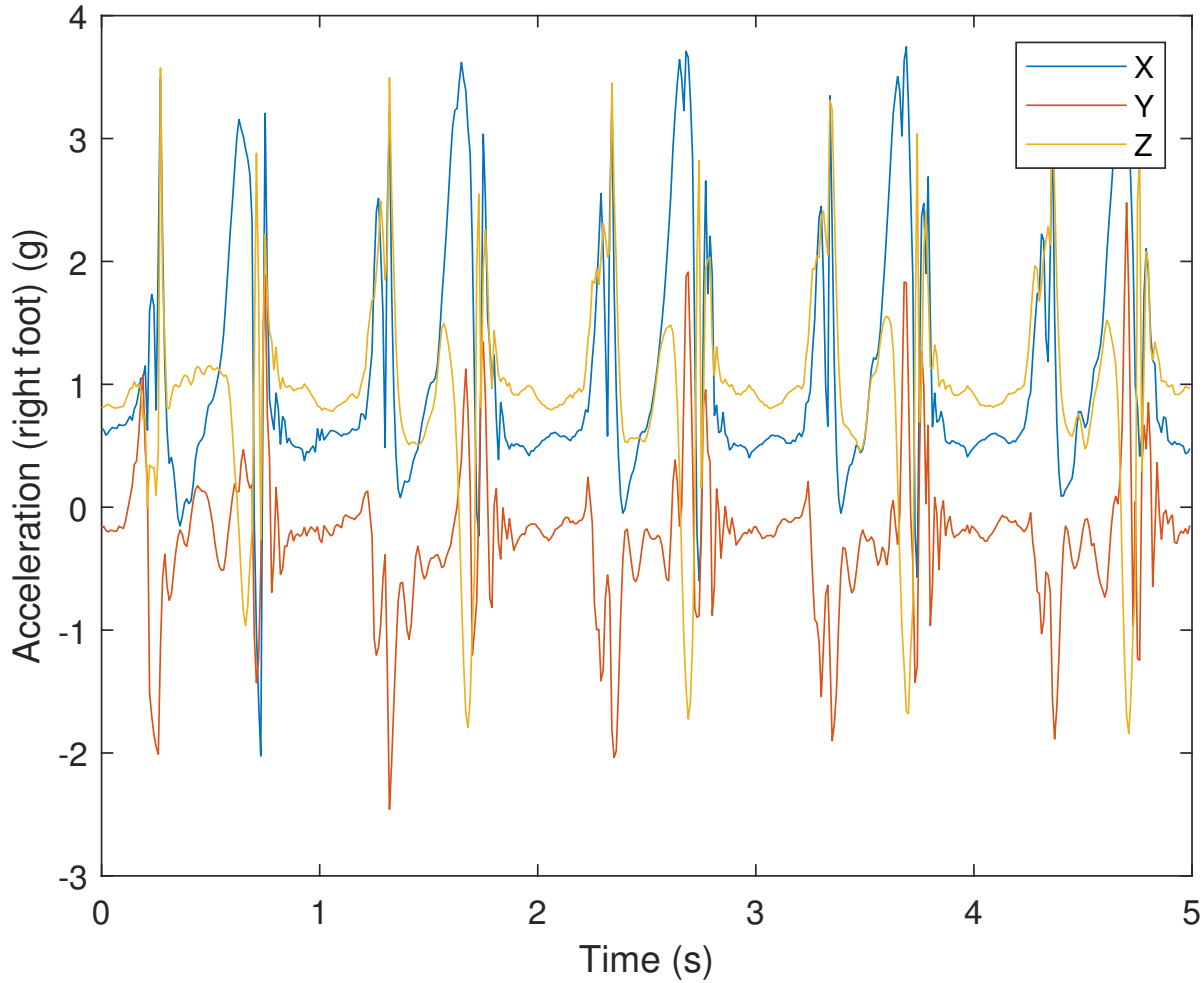
- The whole set of parameter $\underline{\mathbf{A}}$ can be estimated by minimizing the quantity $\|\mathbf{Y} - \mathbf{Z}\underline{\mathbf{A}}\|_F^2$
- Closed form solution (already described in previous lectures)

$$\hat{\underline{\mathbf{A}}} = \left(\mathbf{Z}^T\mathbf{Z}\right)^{-1}\mathbf{Z}^T\mathbf{Y}$$

- Noise covariance matrix can be estimated by computing the empirical covariance matrix of the residual $\mathbf{Y} - \mathbf{Z}\underline{\mathbf{A}}$

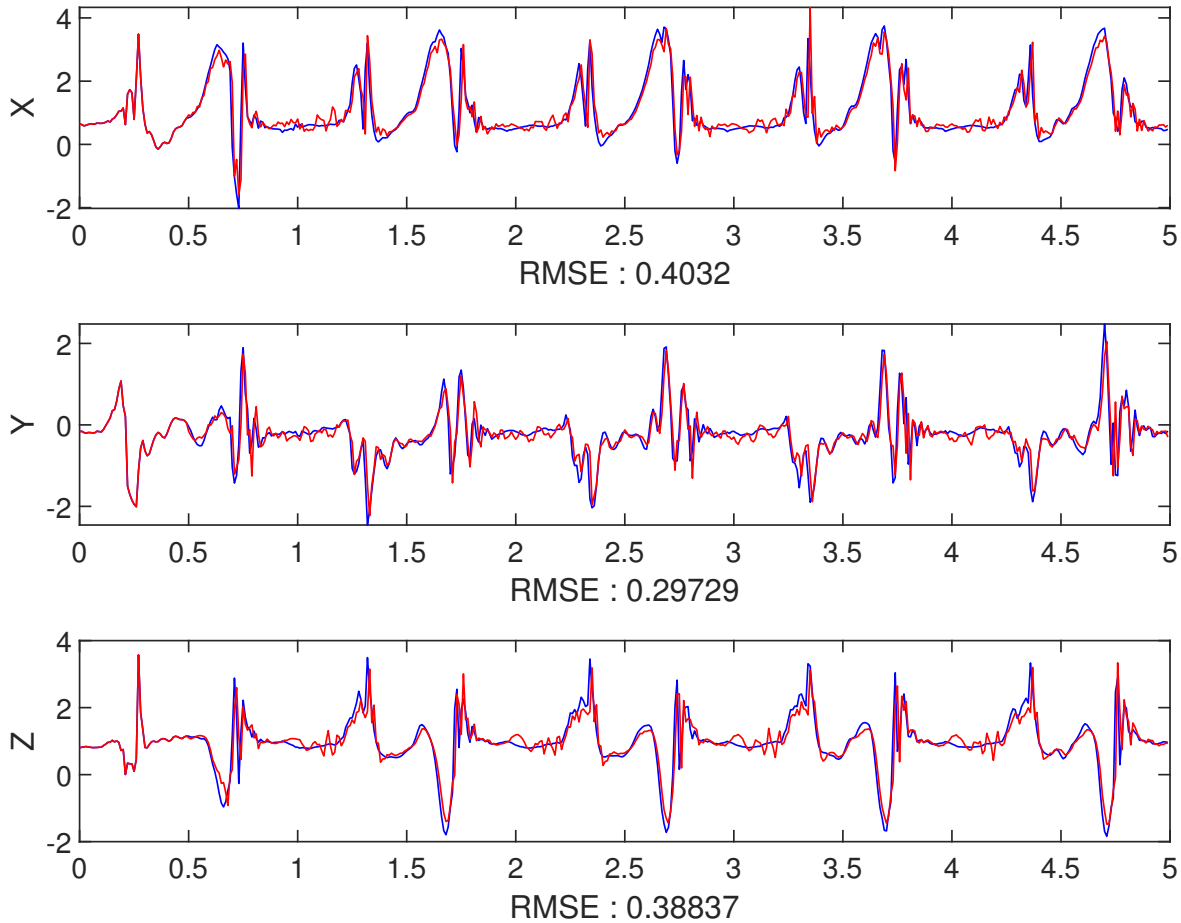
$$\hat{\boldsymbol{\Sigma}}_{\boldsymbol{b}} = \frac{1}{N-p}\left(\mathbf{Y} - \mathbf{Z}\underline{\mathbf{A}}\right)\left(\mathbf{Y} - \mathbf{Z}\underline{\mathbf{A}}\right)^T$$

# Example



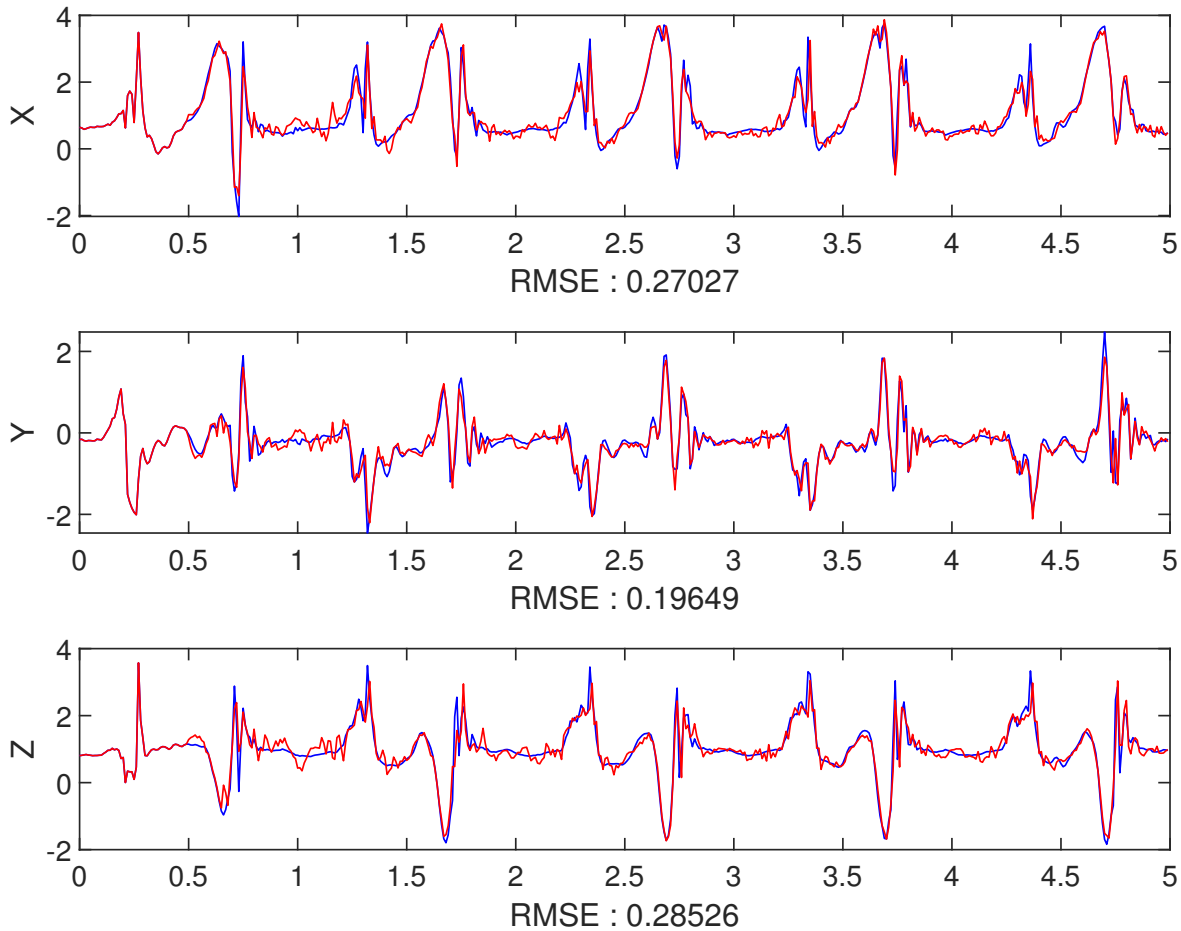

Acceleration recorded on the right foot during locomotion ($F_s = 100$ Hz)

# Example



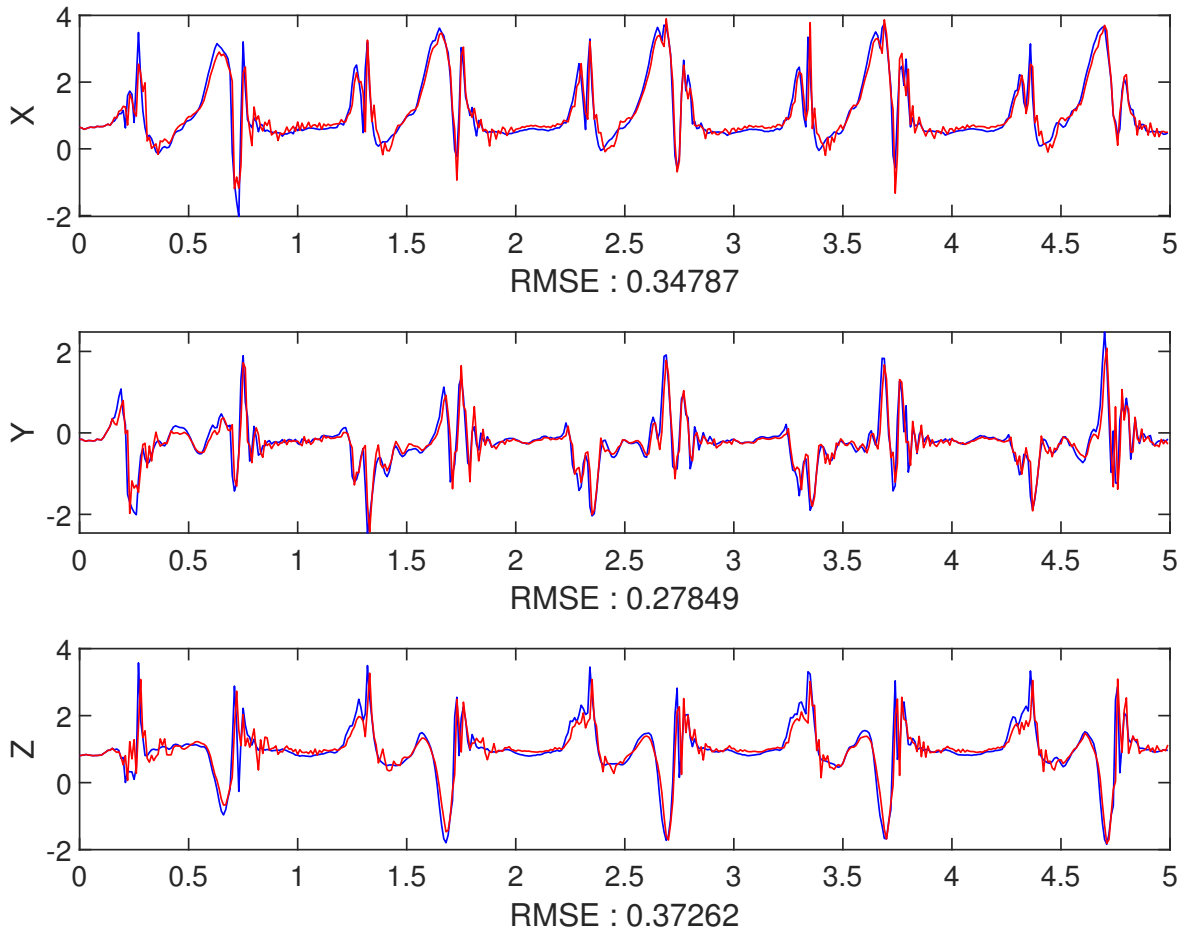

Reconstruction with univariate AR(50) model (blue: true, red: reconstruction)
150 parameters

# Example




Reconstruction with multivariate VAR(50) model (blue: true, red: reconstruction)
1350 parameters

# Example




Reconstruction with multivariate VAR(16) model (blue: true, red: reconstruction)
144 parameters

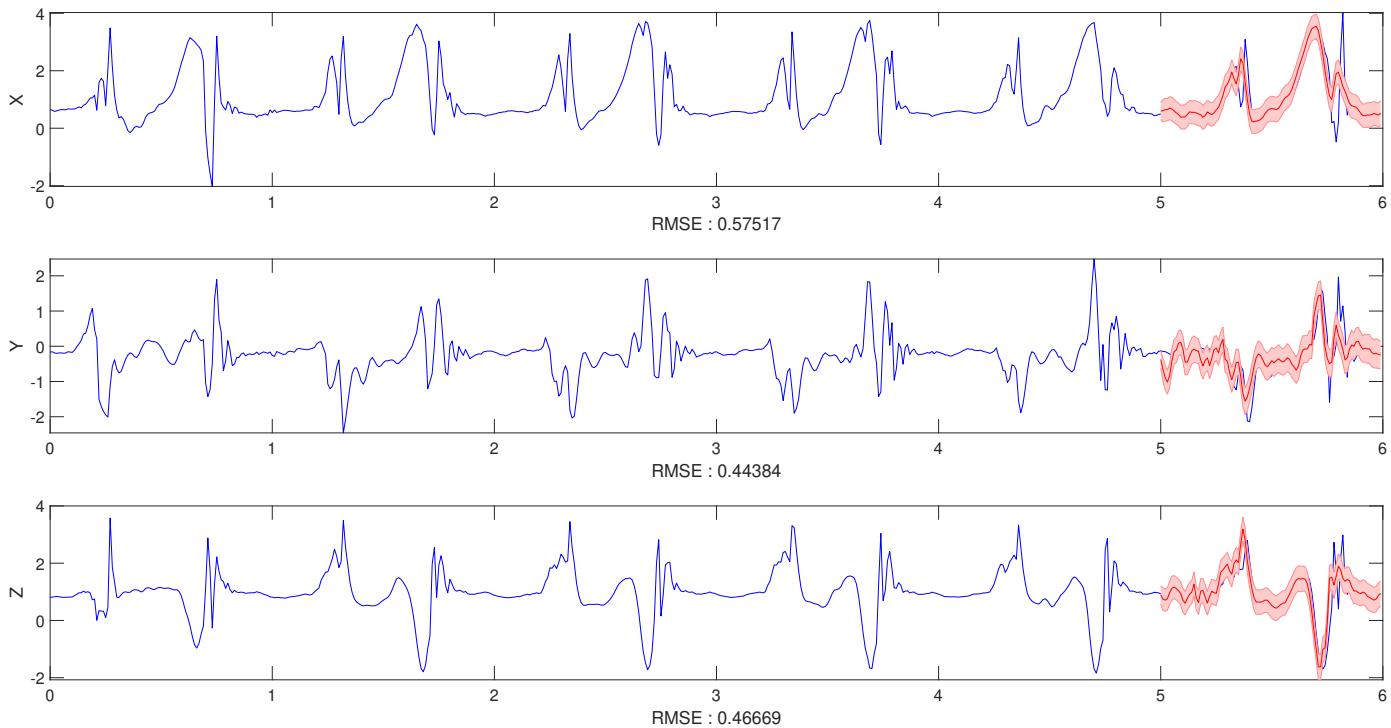
# Prediction with $VAR(p)$ model

- Just as univariate AR models, prediction can be made using parameter estimates by computing

$$\hat{\mathbf{x}}[n] = \hat{\mathbf{c}} + \sum_{i=1}^{p} \hat{\mathbf{A}}_i \mathbf{x}[n-i]$$

- Order $p$ can be selected with model selection procedures (BIC/AIC : see Lecture 5) or by plotting the reconstruction error from 1 to $p_{max} = \lfloor \frac{N-1}{D+1} \rfloor$ and search for an elbow [Hurvich et al., 1993]
- All variants of AR models have multivariate counterparts: VMA, VARMA, VARIMA... They are straightforward extensions based on the same principle

# Example




Prediction of the last 100 samples with multivariate VAR(100) model

# Multivariate dictionary learning

Two strategies for dictionary learning have been introduced in the course:

- Convolutional dictionary learning (Lecture 1)

$$\min_{\substack{(\mathbf{d}_k),(\mathbf{z}_k) \\ \forall k, \|\mathbf{d}_k\|_2^2 \leq 1}} \left\| \mathbf{x} - \sum_{k=1}^{K} \mathbf{z}_k * \mathbf{d}_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|\mathbf{z}_k\|_1$$

- Dictionary learning (Lecture 3)

$$\min_{\substack{(\mathbf{d}_k),(\mathbf{z}_k) \\ \forall k, \|\mathbf{d_k}\|_2 \leq 1}} \|\mathbf{X} - \mathbf{DZ}\|_F^2 + \lambda \sum_{k=1}^{K} \|\mathbf{z}_k\|_1$$

**How can we extend this to multivariate data?**

# Convolutional dictionary learning

$$\min_{\substack{(\mathbf{d}_k),(\mathbf{z}_k) \\ \forall k, \|\mathbf{d}_k\|_2^2 \leq 1}} \left\| \mathbf{x} - \sum_{k=1}^{K} \mathbf{z}_k * \mathbf{d}_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|\mathbf{z}_k\|_1$$

- For convolutional dictionary learning, the extension is straightforward since we were dealing with only vector data
- New shapes:
  - Input data $\mathbf{X}$ of size $N \times D$
  - Each atom $\mathbf{D}_k$ is now a matrix of size $L \times D$ ($L$ is the pattern length)
  - The associated activation signal $\mathbf{z}_k$ is still a vector of length $(N - L + 1) \times 1$
- The only difference is that we search for multivariate patterns instead of univariate ones

# Convolutional dictionary learning

- The multivariate convolutional dictionary learning rewrites as [Garcia-Cardona, 2018]

$$\min_{\substack{(\mathbf{D}_k),(\mathbf{z}_k) \\ \forall k, \|\mathbf{D}_k\|_2^2 \leq 1}} \left\| \mathbf{X} - \sum_{k=1}^{K} \mathbf{z}_k * \mathbf{D}_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|\mathbf{z}_k\|_1$$

where the multivariate convolution product $*$ is now defined as

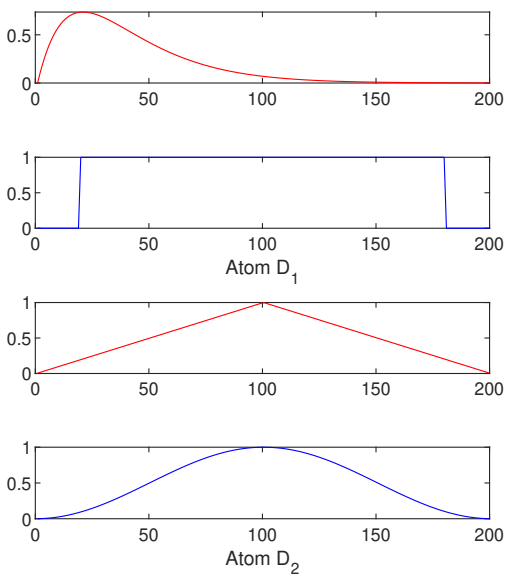
$$\mathbf{z} * \mathbf{D} = \left[ \mathbf{z} * \mathbf{D}^1, \ldots, \mathbf{z} * \mathbf{D}^D \right]$$

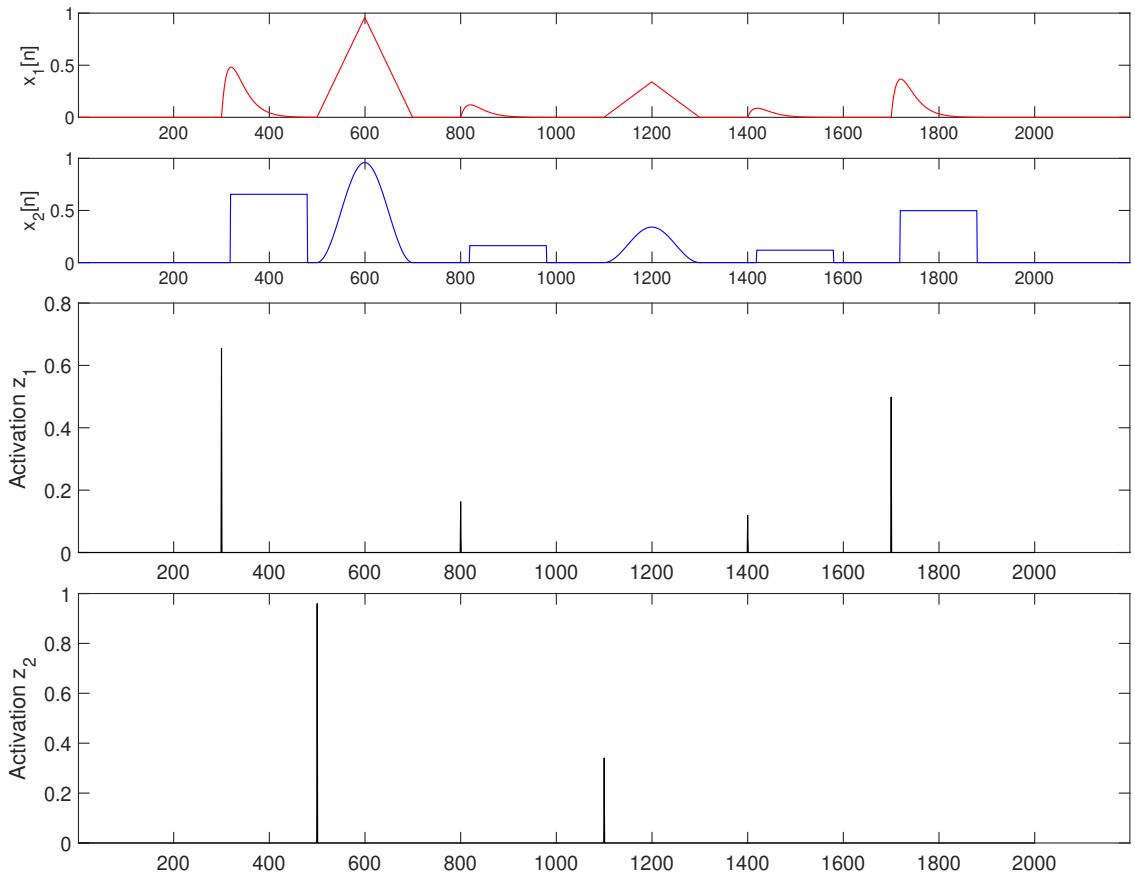
taken independently on each dimension.

- This model can even be extended to a collection of $M$ signals as:

$$\min_{\substack{(\mathbf{D}_k),(\mathbf{z}_k^{(m)}) \\ \forall k, \|\mathbf{D}_k\|_2^2 \leq 1}} \sum_{m=1}^{M} \left\| \mathbf{X}^{(m)} - \sum_{k=1}^{K} \mathbf{z}_k^{(m)} * \mathbf{D}_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|\mathbf{z}_k^{(m)}\|_1$$

# Convolutional dictionary learning



# Resolution

- This problem can be solved with exactly the same solvers than the univariate case: FISTA, ADMM... (see Lecture 1), mostly based on variants of gradient descent
- Alternated resolution between Convolutional Sparse Coding (CSC) and Convolutional Dictionary Learning (CDL)
- However, the computation of the gradients may become prohibitive due to the large dimension of the objects [Garcia-Cardona et al., 2018]
- Recent solutions propose to use parallel computing and Locally Greedy Gradient Descent that iteratively updates the $z_k^{(m)}[n]$ coefficients on several cores [Moreau et al., 2018]

# Multivariate dictionary learning

$$\min_{\substack{(\mathbf{d}_k),(\mathbf{z}_k) \\ \forall k, \|\mathbf{d_k}\|_2 \leq 1}} \|\mathbf{X} - \mathbf{DZ}\|_F^2 + \lambda \sum_{k=1}^{K} \|\mathbf{z}_k\|_1$$

- For multivariate data, dictionary learning can be reformulated in several ways depending on the assumptions
- **Solution 1 (easy)**: one dictionary per dimension
  - Solving of $D$ independent standard DL problems: as if each dimension was a different signal
  - No new method is needed (see Lecture 3)
- **Solution 2 (more complex)**: one shared dictionary for all dimensions
  - Reformulation of the sparsity constraint
  - Implies a modification of the sparse coding step: **joint sparse representation**

# Sparsity constraint for multivariate sparse coding

$$\min_{\mathbf{z}_k} \|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_F^2 + \lambda\Omega(\mathbf{Z})$$

- Input data $\mathbf{X}$ of size $N \times D$
- $K$ atoms $\mathbf{d}_k$ stored in a matrix $\mathbf{D}$ of size $N \times K$ ($K$ is the number of atoms)

$$\mathbf{D} = \begin{bmatrix} \mathbf{d}_1 & \dots & \mathbf{d}_K \end{bmatrix}$$

- The associated activation signals $\mathbf{z}_k$ are stored in a matrix of size $K \times D$

$$\mathbf{Z} = \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_K \end{bmatrix}$$

$z_{k,d}$ : activation of atom $k$ for dimension $d$

# Joint sparsity constraint

$z_{k,d}$ : activation of atom $k$ for dimension $d$

What is *good sparsity* for a **joint sparse representation**?

- $\mathbf{z}_{:,d}$ (column) should be sparse: the signal in dimension $d$ should be represented as the linear combination of a few atoms
- BUT $\mathbf{z}_k$ (row) should NOT be sparse: on the contrary, in order to learn a **joint** representation, the atoms should be used for as many dimensions as possible

$$\Omega(\mathbf{Z}) = \sum_{k=1}^{K} \left( \sum_{d=1}^{D} |z_{k,d}|^q \right)^{\frac{p}{q}}$$

$p \leq 1$ for enforcing sparsity on the columns

$q \geq 1$ for the rows

# Multivariate sparse coding algorithms

$$\min_{\mathbf{z}_k} \|\mathbf{X} - \mathbf{DZ}\|_F^2 + \lambda \sum_{k=1}^{K} \left( \sum_{d=1}^{D} |z_{k,d}|^q \right)^{\frac{p}{q}}$$

Several algorithms:

- $p = q = 1$: the problem separates into independent simple sparse approximation problems
- $p = q = 2$: weighted least squares resolution with closed form solution

$$\mathbf{Z} = (\mathbf{D}^T\mathbf{D} + \lambda \mathbf{Id}_K)^{-1}\mathbf{D}^T\mathbf{X} = \mathbf{D}^T(\mathbf{D}\mathbf{D}^T + \lambda \mathbf{Id}_N)^{-1}\mathbf{X}$$

- $p = 1$, $q = 2$, constraint equivalent to $L_{2,1}$ norm of $\mathbf{Z}^T$: iterated reweighted least squares resolution (M-FOCUSS : see next slide)

See [Tropp et al., 2006] for a review of different algorithms

# $L_{2,1}$-regularization

- With $p = 1$, $q = 2$, the sparse coding problem rewrites as

$$\min_{\mathbf{z}_k} \|\mathbf{X} - \mathbf{DZ}\|_F^2 + \lambda \sum_{k=1}^{K} \left( \sum_{d=1}^{D} |z_{k,d}|^2 \right)^{\frac{1}{2}} = \|\mathbf{X} - \mathbf{DZ}\|_F^2 + \lambda \sum_{k=1}^{K} \|\mathbf{z}_k\|_2$$

- In order to solve this problem, the idea is to use a weight matrix that will force rows with small norms to go to zero

$$\mathbf{W} = \begin{bmatrix} \sqrt{\|\mathbf{z}_1\|_2} & 0 & \cdots & 0 \\ 0 & \sqrt{\|\mathbf{z}_2\|_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sqrt{\|\mathbf{z}_K\|_2} \end{bmatrix}$$

# $L_{2,1}$-regularization

- Noticing that

$$\sum_{k=1}^{K}\sum_{d=1}^{D}\frac{|z_{k,d}|^2}{\|\mathbf{z}_k\|_2}=\sum_{k=1}^{K}\|\mathbf{z}_k\|_2$$

- The problem can be rephrased with weight matrix $\mathbf{W}$ as

$$\min_{\mathbf{z}_k}\|\mathbf{X}-\mathbf{D}\mathbf{Z}\|_F^2+\lambda\left\|\mathbf{W}^{-1}\mathbf{Z}\right\|_F^2$$

- Iterated reweighted least squares resolution [Cotter et al., 2005]: iterative computation of the weight matrix and of the weighted LS solution with

$$\mathbf{Z}=\mathbf{W}^2\mathbf{D}^T\left(\mathbf{D}\mathbf{W}^2\mathbf{D}^T+\lambda\mathbf{Id}_N\right)^{-1}\mathbf{X}$$

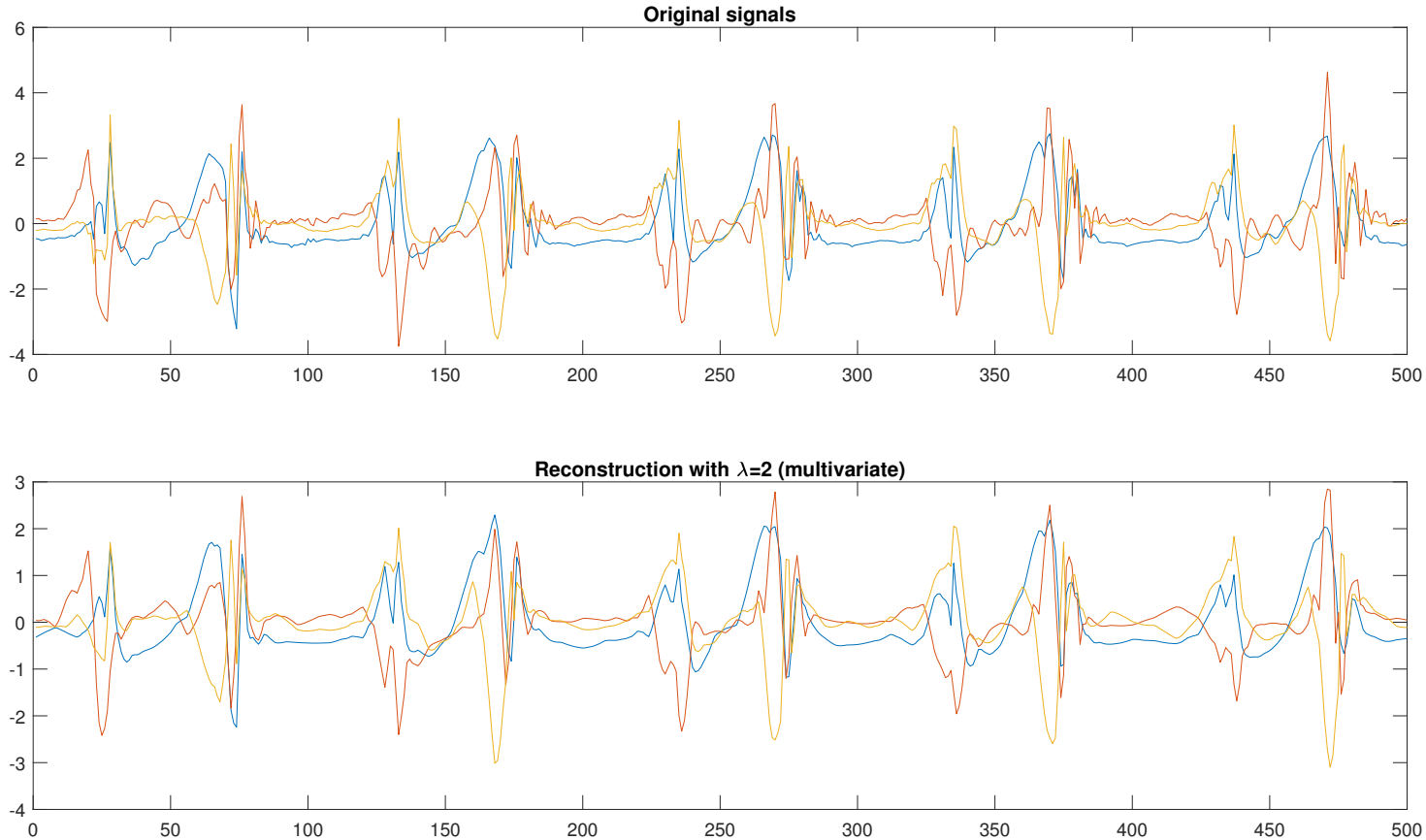
# $L_{2,1}$-regularization: M-FOCUSS

$$\min_{\mathbf{z}_k} \|\mathbf{X} - \mathbf{D}\mathbf{Z}\|_F^2 + \lambda \sum_{k=1}^{K} \left( \sum_{d=1}^{D} |z_{k,d}|^2 \right)^{\frac{1}{2}}$$

**Algorithm 1:** M-FOCUSS

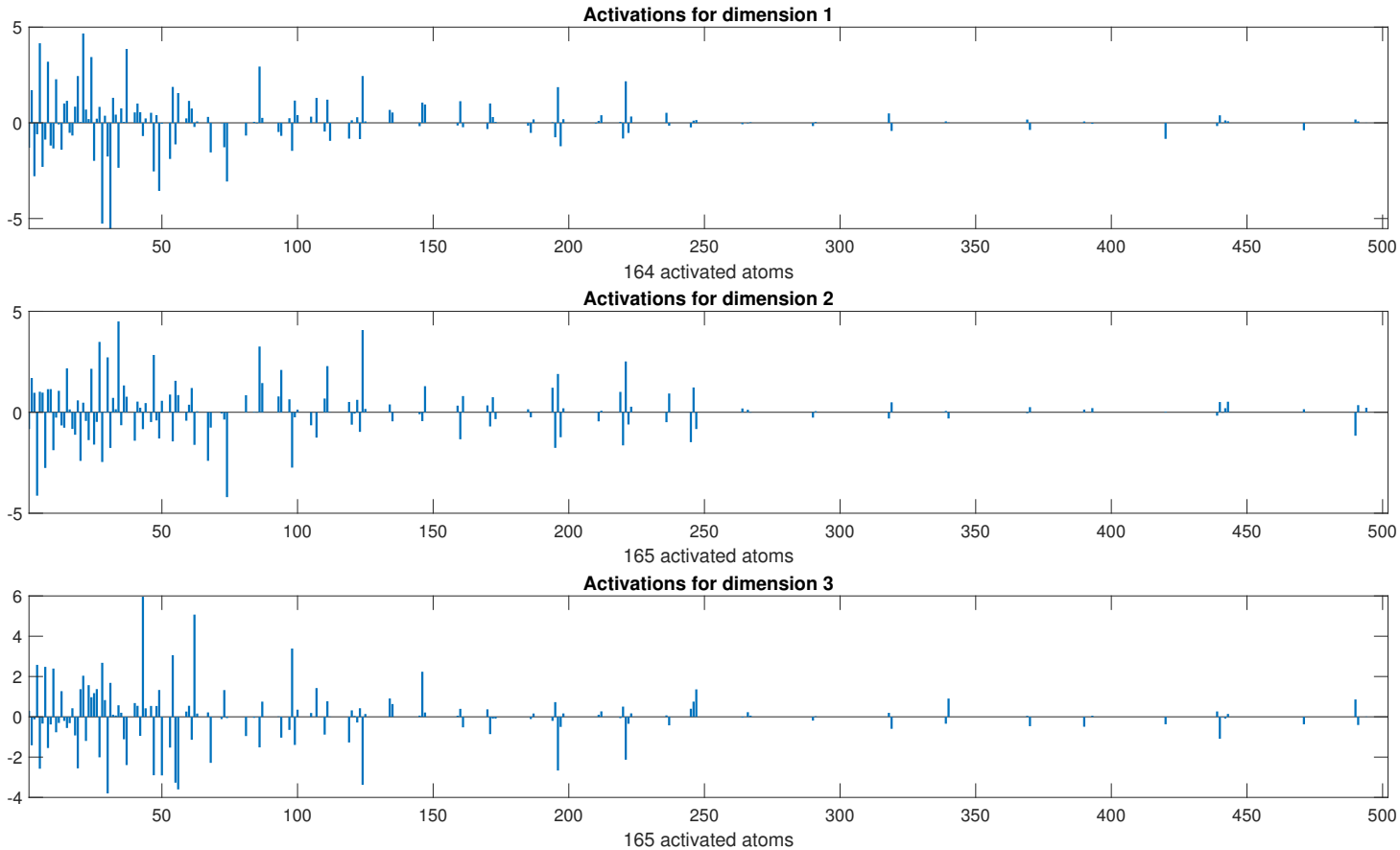
**Inputs :** Multivariate time series $\mathbf{X} \in \mathbb{R}^{N \times D}$
Dictionary $\mathbf{D} \in \mathbb{R}^{N \times K}$
Penalty $\lambda$
**Output:** Activation matrix $\mathbf{Z} \in \mathbb{R}^{K \times D}$
Initialize $\mathbf{Z}$;
**while** $n_{iter} < n_{max}$ **do**
    *Weight matrix*;
    $\mathbf{W} = \text{diag}(\sqrt{\|\mathbf{z}_1\|_2}, \sqrt{\|\mathbf{z}_2\|_2}, \cdots, \sqrt{\|\mathbf{z}_K\|_2})$;
    *Weighted LS solution*;
    $\mathbf{Z} = \mathbf{W}^2\mathbf{D}^T \left(\mathbf{D}\mathbf{W}^2\mathbf{D}^T + \lambda \mathbf{Id}_N\right)^{-1} \mathbf{X}$;
**end**

# Example



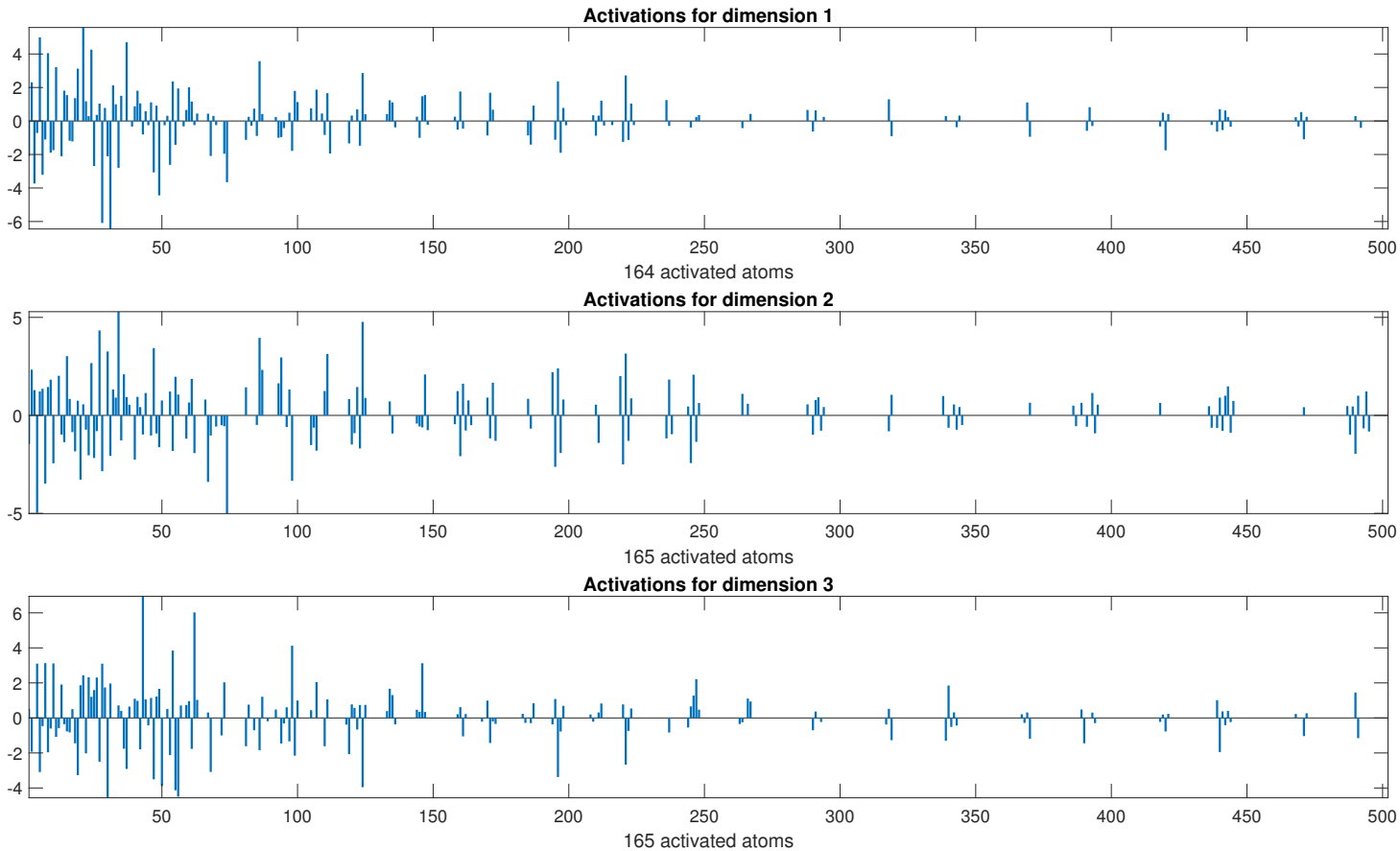

Daubechies db4 dictionary and $\lambda = 2$

# Example



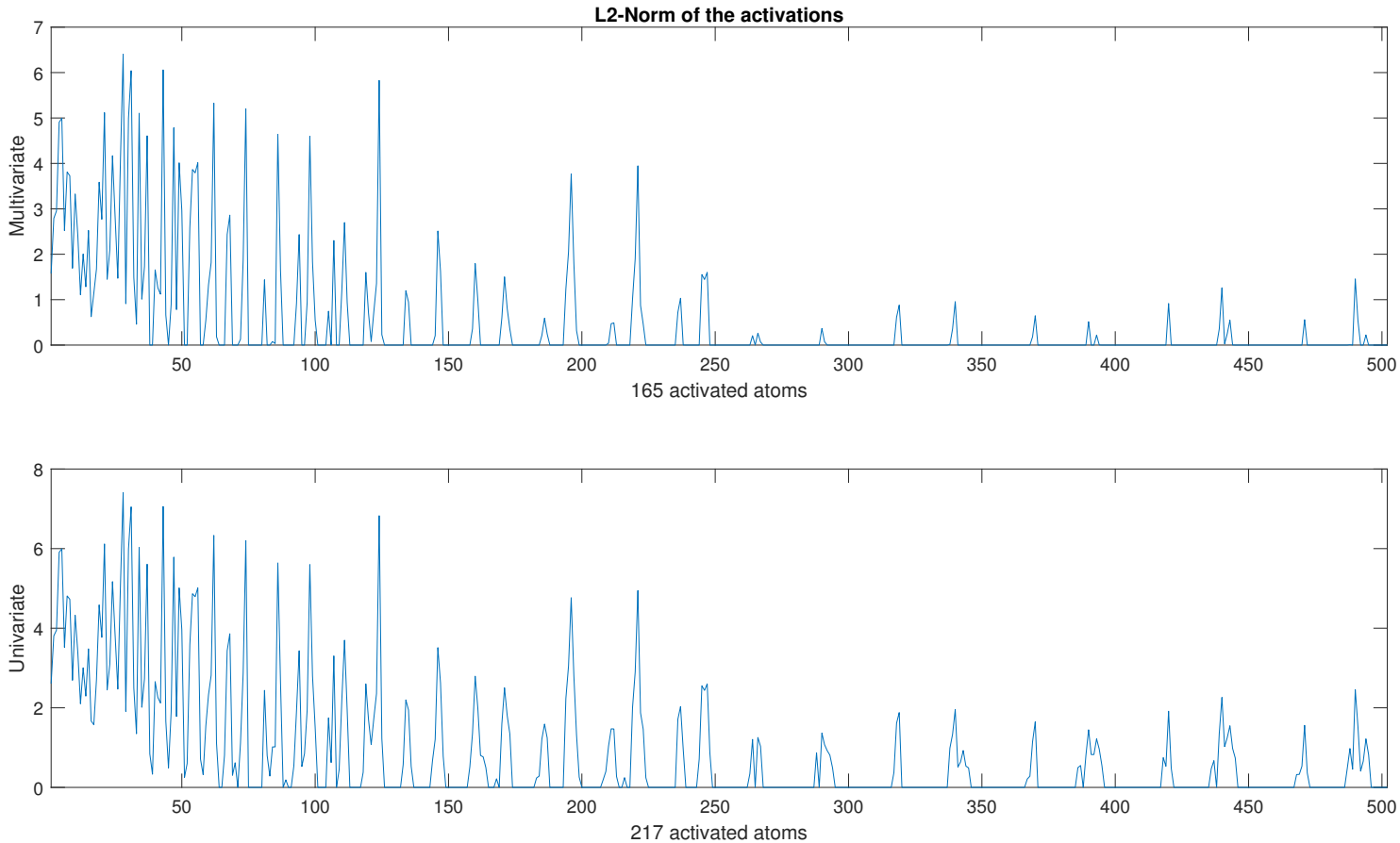

**Multivariate case:** $L_{2,1}$ regularization forces to use the same atoms for every dimension

# Example




**Univariate case** with the same number of activated atoms

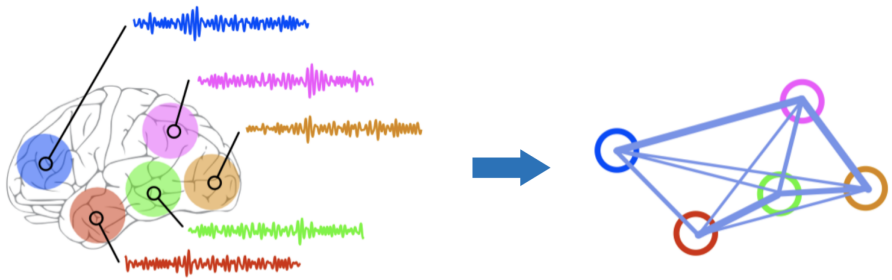
# Example




Contrary to the multivariate case, in the univariate case the activated atoms are not the same for each dimension

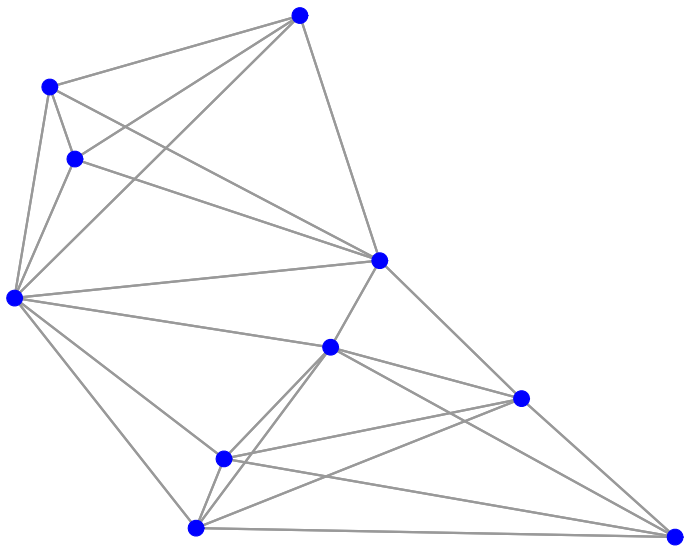
# Contents

# Notion of networks



- In most practical applications, the different dimensions of a multivariate signal $\mathbf{x}[n]$ are linked
  - Notion of correlation between recorded variables (ex: pressure/temperature/precipitation)
  - Sensor networks, body sensors, social networks...: spatial proximity, interactions...
- These links can be explicitly be modeled through a graph structure: **Graph Signal Processing** [Ortega et al., 2018]
- Each multivariate sample $\mathbf{x}[n]$ is assumed to be carried on the graph

# What is a graph ?



A graph G is a triplet $(\mathcal{V}, \mathcal{E}, \mathcal{W})$

- $\mathcal{V}$ is a finite set of $D$ nodes or vertices (usually $\{1, 2, ..., D\}$)
- $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ is a set of edges
- $\mathcal{W} : \mathcal{E} \to \mathbb{R}$ is a map from the set of edges to scalar values

Here : undirected graph, positive weights

$\mathcal{W}(i, j)$ encodes the strength of the relationship between dimensions $i$ and $j$

# Laplacian of a graph

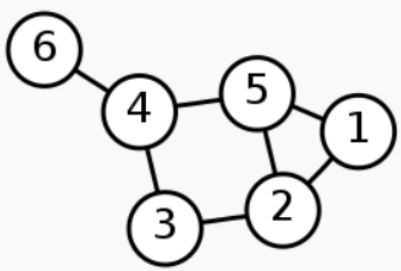
$$\mathbf{L} = \mathbf{D} - \mathbf{W}$$

- $\mathbf{W}$ (weight or adjacency matrix) :

$$W_{i,j} = \begin{cases} \mathcal{W}(i,j) & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{Otherwise} \end{cases}$$

- $\mathbf{D}$ (degree matrix) : diagonal matrix with

$$D_{i,i} = \sum_j W_{i,j}$$

| Labeled graph | Degree matrix | Adjacency matrix | Laplacian matrix |
|---|---|---|---|
|  | $\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$ | $\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ | $\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$ |



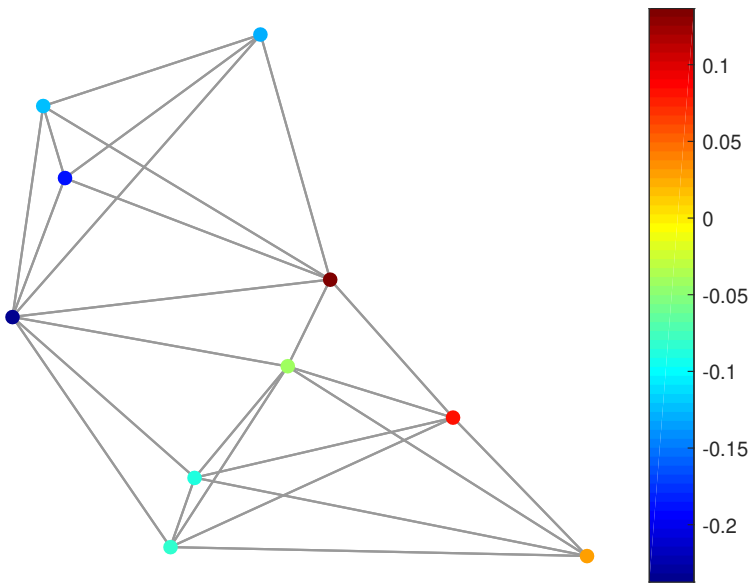
Symmetric, positive semi-definite

# Laplacian of a graph

By construction of the Laplacian matrix

- $\forall \mathbf{x} \in \mathbb{R}^D, \quad \mathbf{x}^T \mathbf{L} \mathbf{x} = \frac{1}{2} \sum_{(i,j) \in \mathcal{E}} W_{i,j} (x_i - x_j)^2 \geq 0$
- The constant vector $\mathbf{1}_D$ is an eigenvector for matrix $\mathbf{L}$ associated to eigenvalue $\lambda_1 = 0$
  Obvious as the sum of the matrix along the rows/column is equal to zero
- The number of connected components in the graph is equal to the number of eigenvalues equal to zero

# What is a graph signal ?

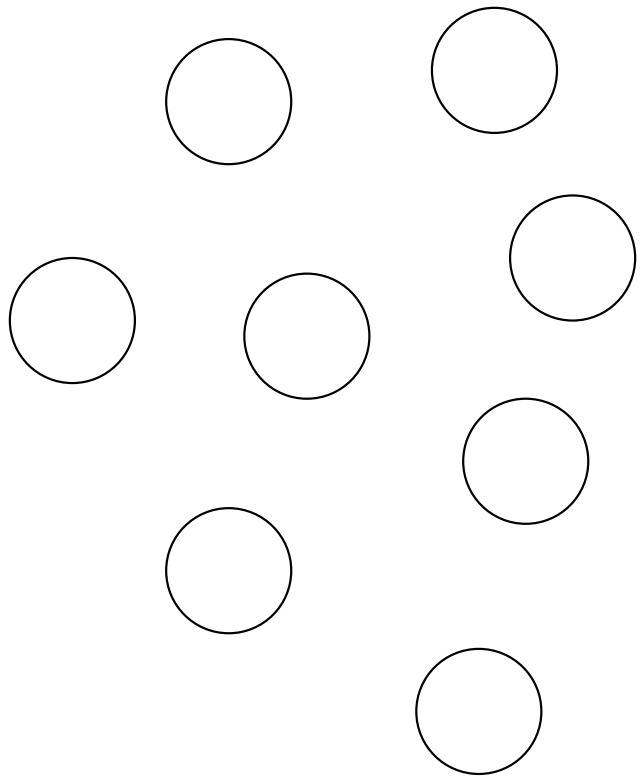



Given a graph $\mathcal{G}$ of $D$ nodes, a graph signal is an array $x \in \Omega^D$ that associates an element of $\Omega$ to each node of $\mathcal{G}$.

- $\Omega = \mathbb{R}$ : Simple graph signal.
  Each node carries one sample $x_d[n]$
- $\Omega = \mathbb{R}^N$ : Time-vertex signal
  Each node carries one time series $\mathbf{x}_d$

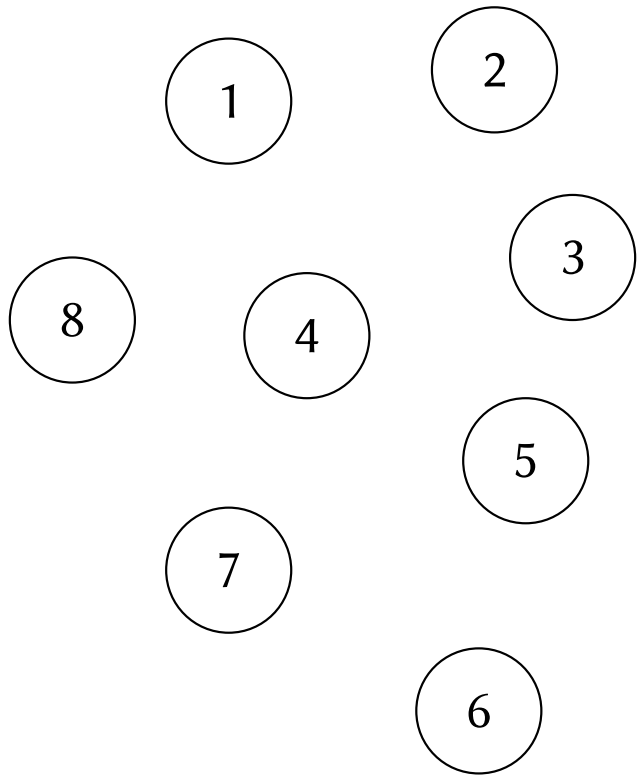
In the following we consider a simple graph signal which corresponds to only one sample $n$: $\mathbf{x}[n]$

# Example



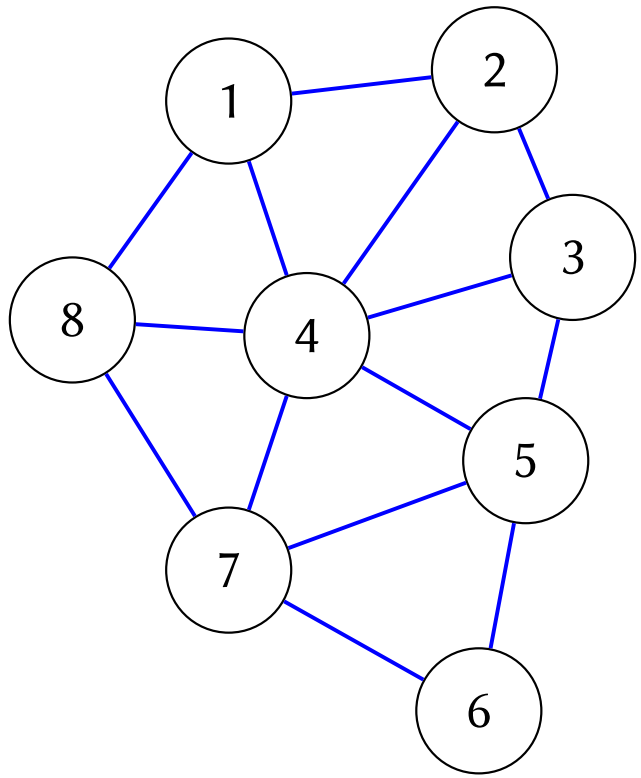
*D* nodes: one per signal dimension

# Example



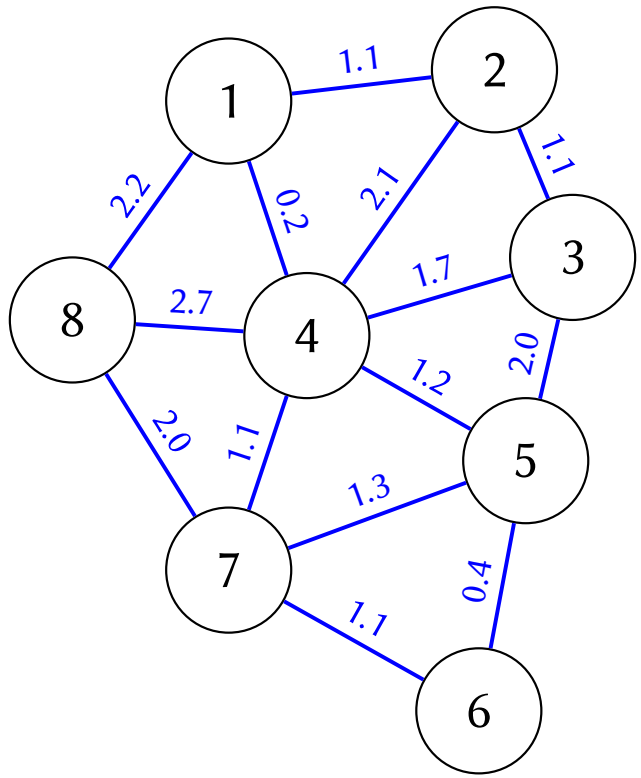

Dimension 1 lies on node 1, 2 on node 2, etc.

# Example



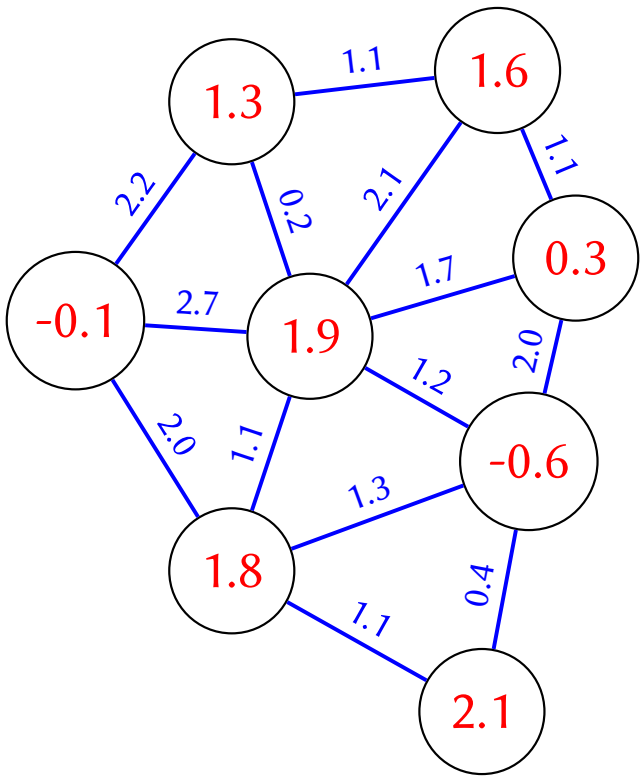

Edges model links between dimensions

# Example



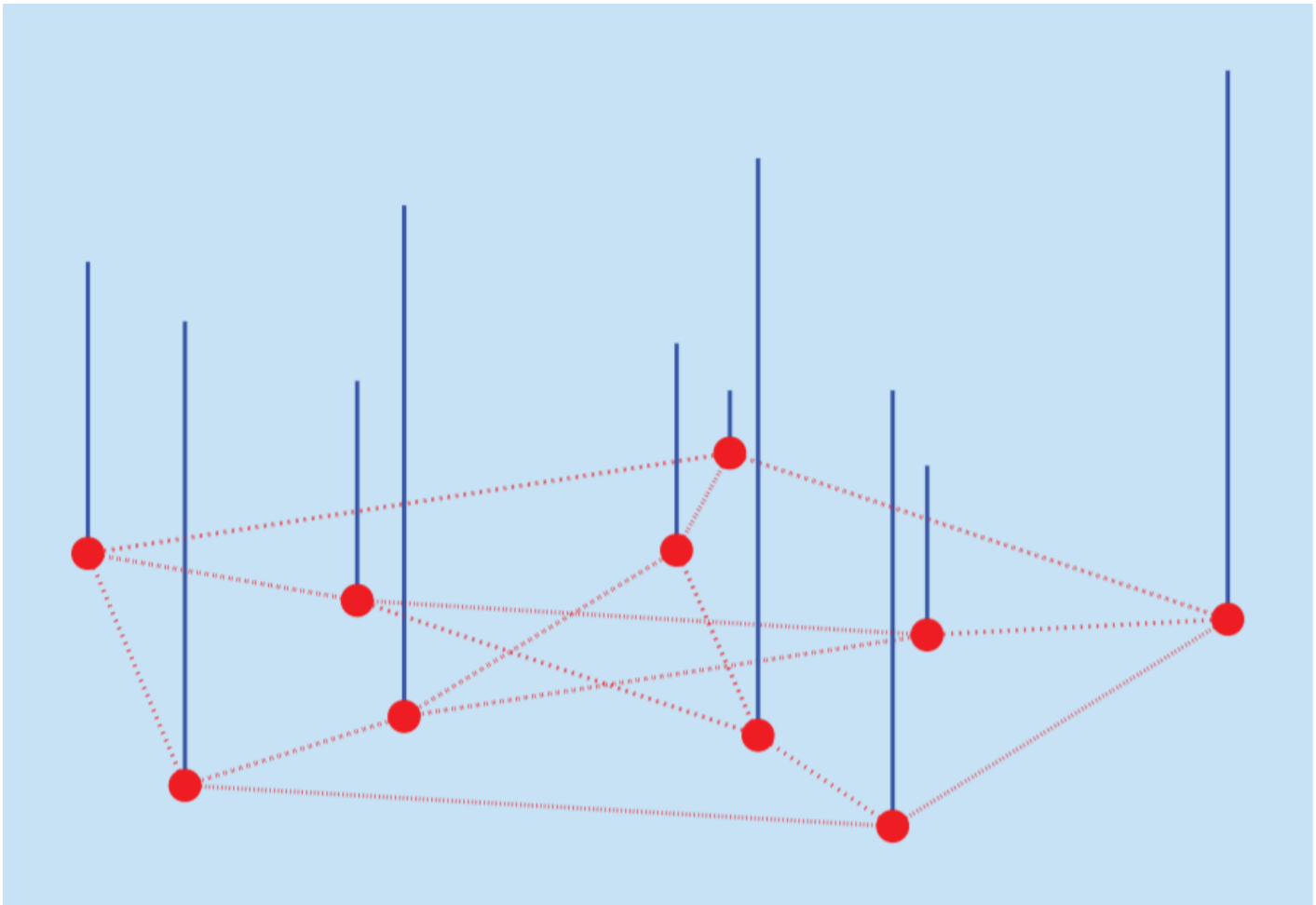

Weights model the strengths of these links

# Example

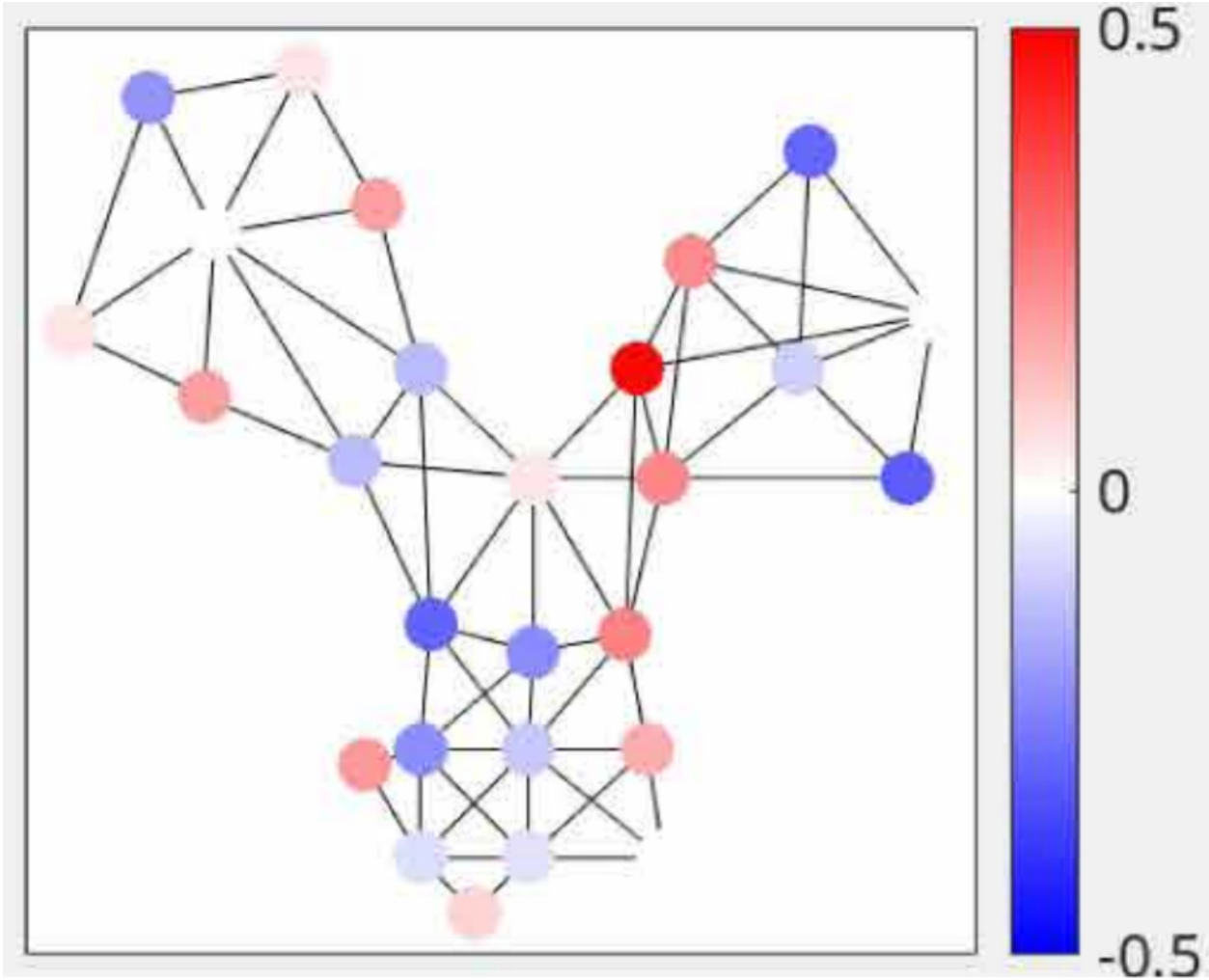



Visualization of one multivariate sample $\mathbf{x}[n]$ on the graph

# How to visualize graph signals?



# How to visualize graph signals?



# Examples

Different kinds of signals :

- Sensor networks (meteorology, population flux, energy consumption)
- Interaction networks (social networks, communication networks, ...)
- Economy based signals (market dependencies, stocks)
- Image processing (intensity, color)
- Cloud points (position, color)

# Tasks

- Several machine learning tasks can be extended to graph signals [Ortega et al., 2018]:
  - Sampling/compression: choose the most relevant nodes (i.e. dimensions) to reconstruct the whole data
  - Graph inference: learn the graph structure from data [Mateos et al., 2019]
  - Denoising/filtering: use the graph structure to remove noise, outliers... [Chen et al., 2014]
  - Interpolation: use the graph structure to reconstruct missing data [Narang et al., 2013]
  - Classification, event detection, anomaly detection, prediction...
- Use the structure to improve performances on multivariate time series

# Graph Fourier Transform

- Given a graph $\mathcal{G}$ with only one connex component, we compute the eigen-decomposition of its Laplacian $\mathbf{L}$ :

$$\mathbf{L} = \mathbf{U} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_D \end{bmatrix} \mathbf{U}^T, \quad 0 = \lambda_1 < \lambda_2 \leq \ldots \leq \lambda_D$$

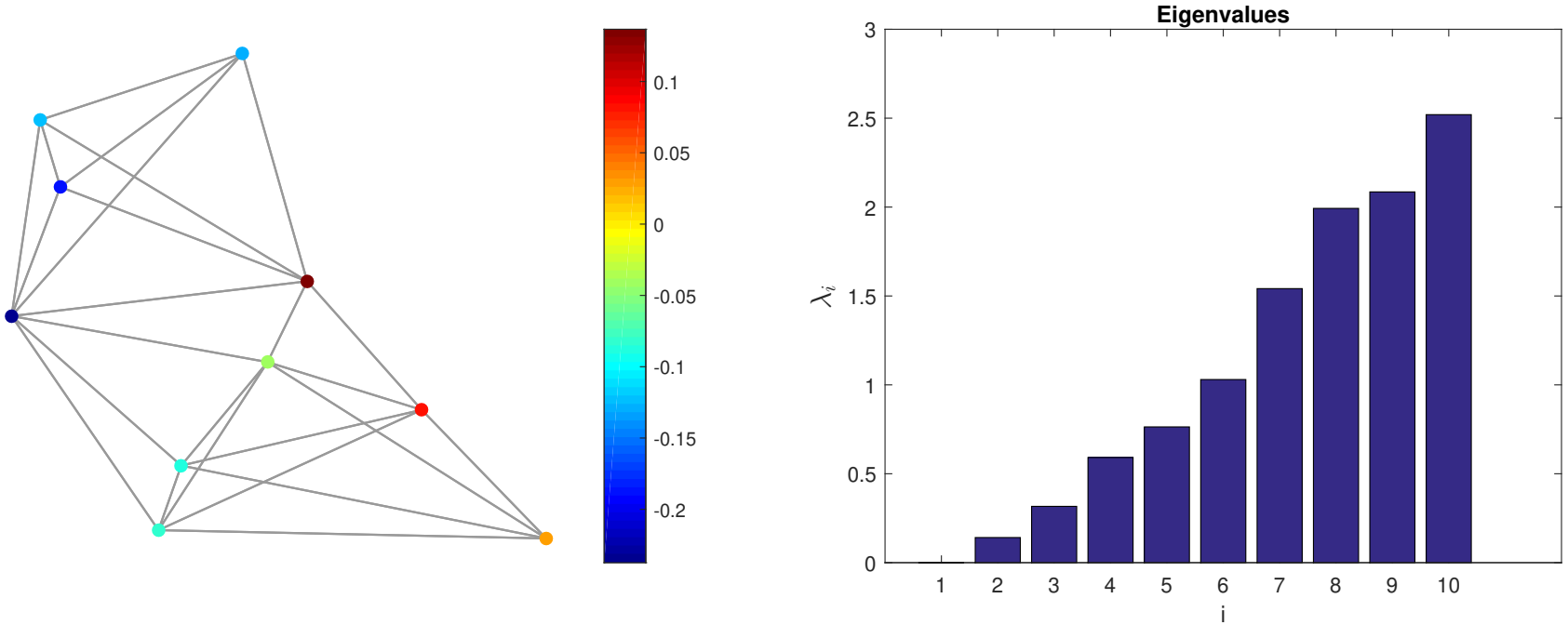
  - $\lambda_i$ are interpretable as *spatial* frequencies
    $\lambda_1 = 0$ : DC component
  - $\mathbf{u}_i$ is the eigenvector associated to frequency $\lambda_i$
    $\mathbf{u}_1$ is the constant vector

- The **Graph Fourier Transform (GFT)** $\hat{\mathbf{x}}$ of a graph signal $\mathbf{x} \in \mathbb{R}^D$ is defined as

$$\hat{\mathbf{x}} = \mathbf{U}^T \mathbf{x}$$

# Example

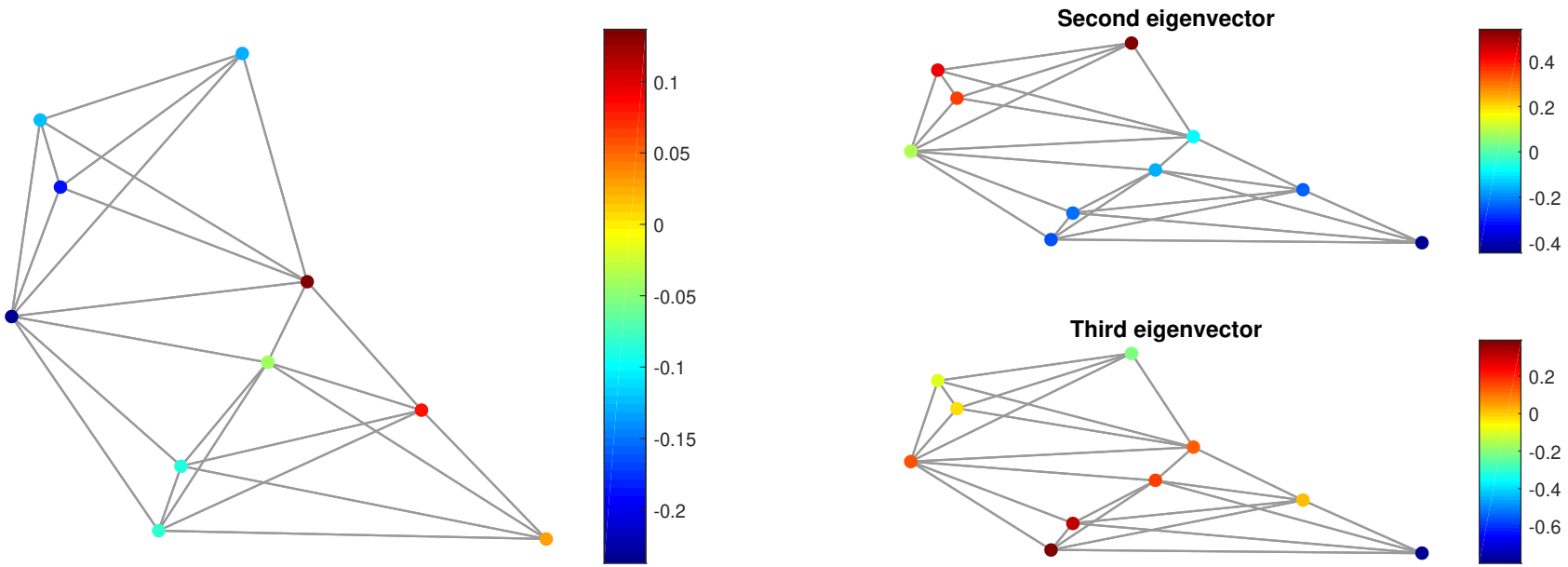
$$\hat{\mathbf{x}} = \mathbf{U}^T \mathbf{x}$$




Eigenvalues $\lambda_i$ can be interpreted as *spatial frequencies*
Low frequencies : global phenomena, high frequencies : local phenomena

# Example

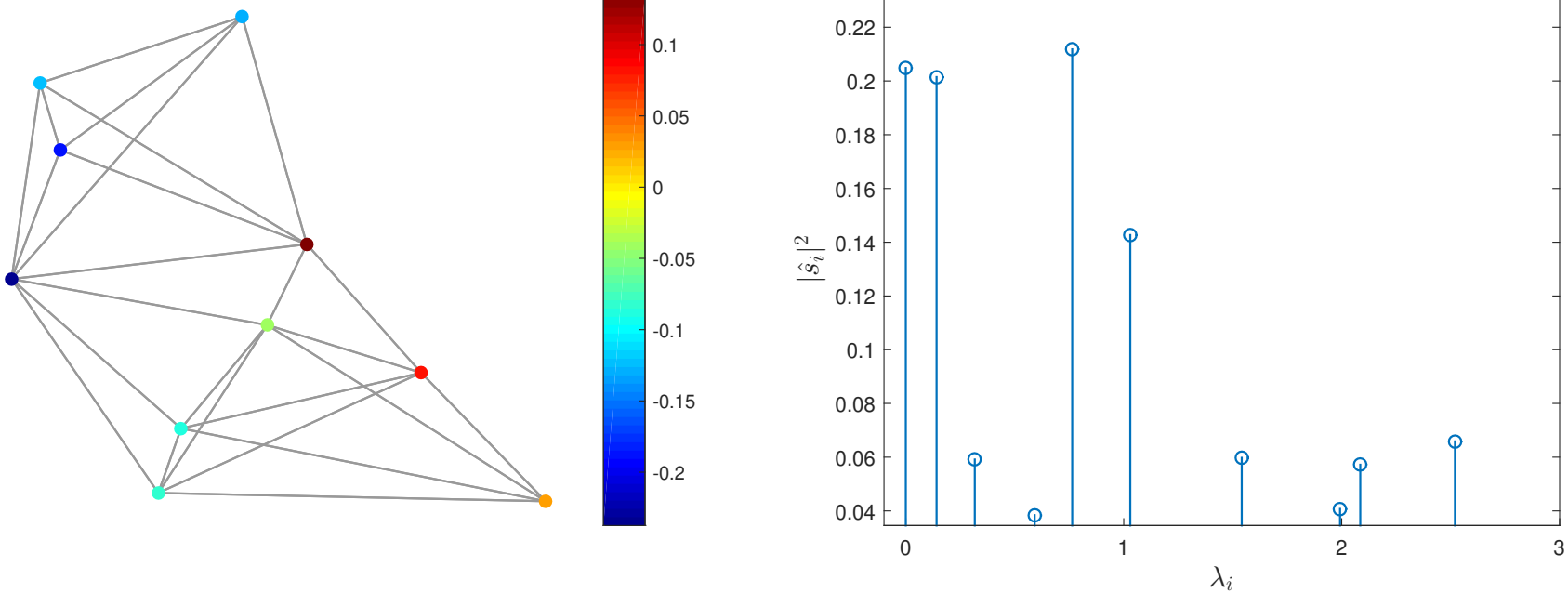
$$\hat{\mathbf{x}} = \mathbf{U}^T \mathbf{x}$$




Eigenvectors $\mathbf{u}_2$ and $\mathbf{u}_3$
Can model symmetries, anti-symmetries, spatial phenomena

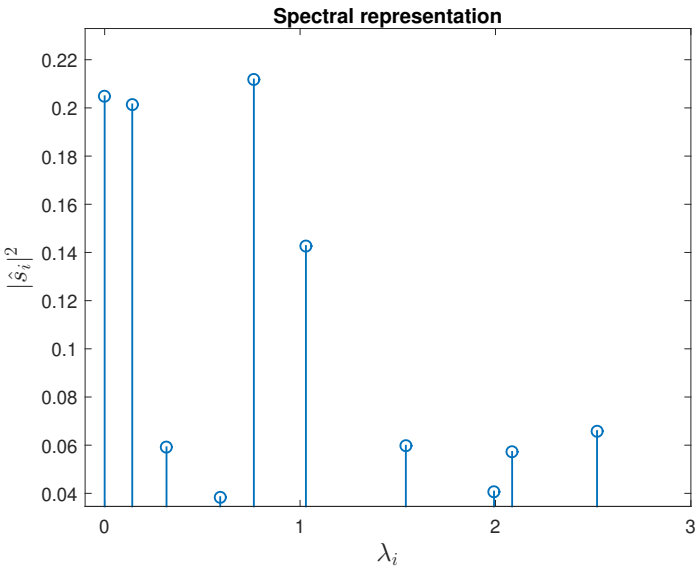
# Example

$$\hat{\mathbf{x}} = \mathbf{U}^T \mathbf{x}$$




Graph spectrum

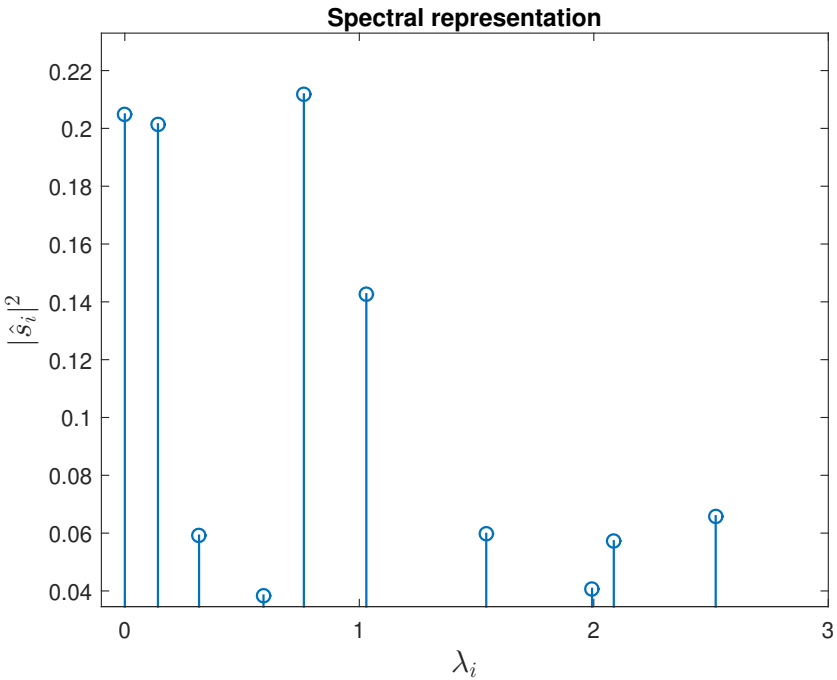
# Bandlimitedness

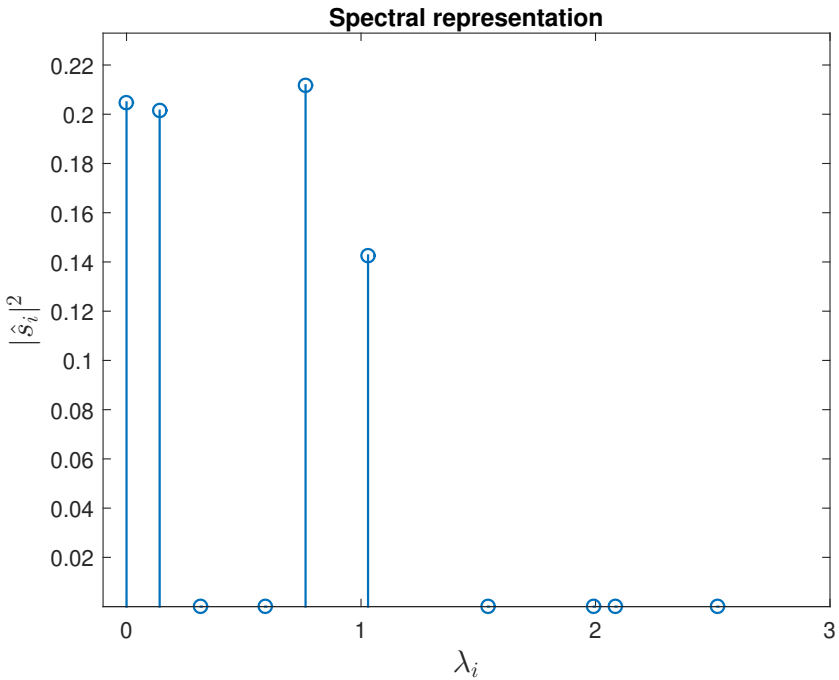



- Common assumption in signal processing : sparsity of the spectrum (see Lecture 3)
- In SP : bandlimitedness of signals (baseband, wideband...) is used for sampling, denoising
- In GSP : same notion but on the graph spectrum

$\mathbf{x}$ is $K$-bandlimited iff. $\|\hat{\mathbf{x}}\|_0 = K$

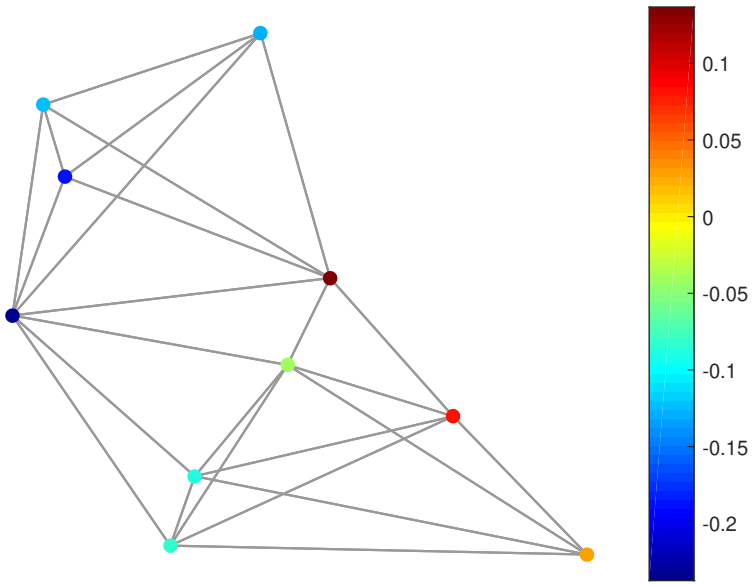
# Example

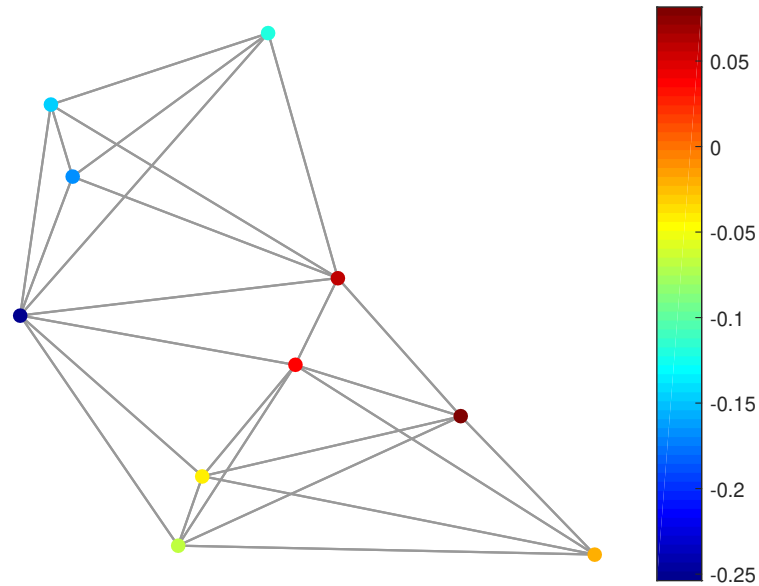



Filtering by removing all frequencies except for the 4 most dominant frequencies

# Example




4-bandlimited approximation

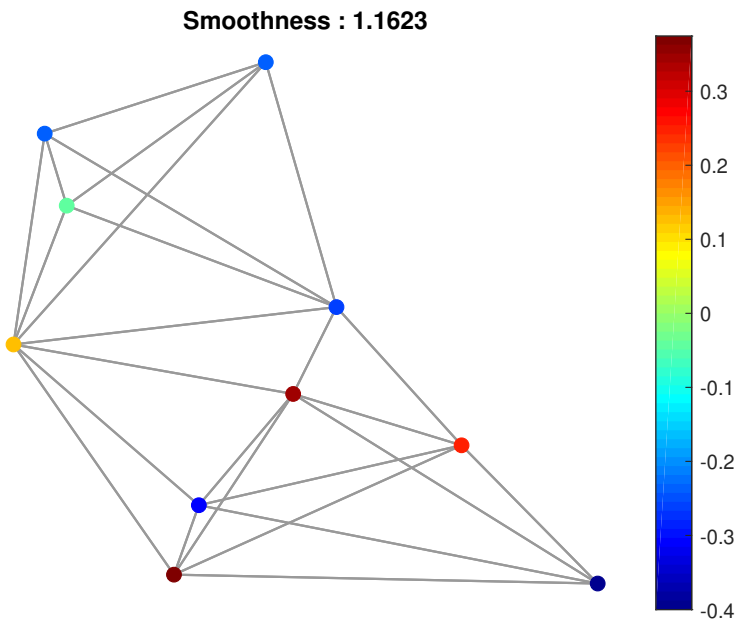
# Smoothness

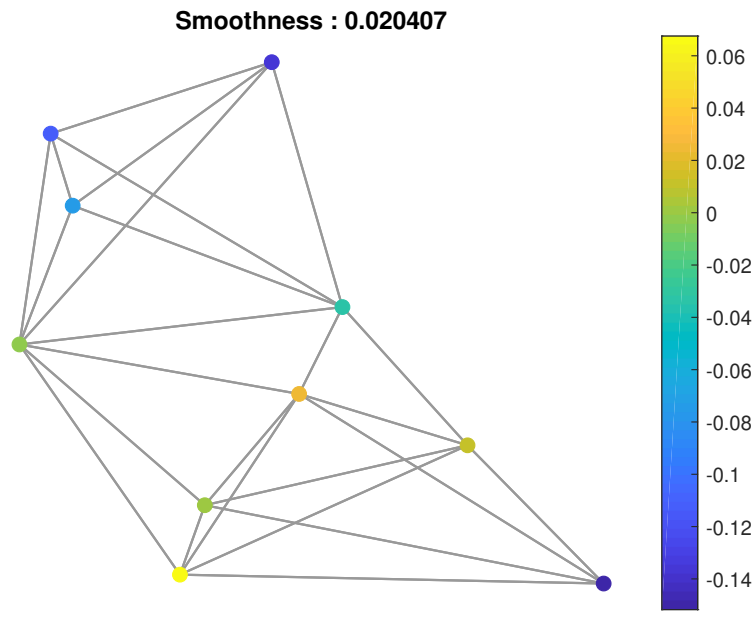
- Intuitively, signal values taken on adjacent nodes should be quite similar
- Notion of **smoothness** for a graph signal $\mathbf{x}$ :

$$S(\mathbf{x}) = \mathbf{x}^T \mathbf{L}\mathbf{x} = \frac{1}{2} \sum_{(i,j)\in\mathcal{E}} W_{i,j}(x_i - x_j)^2$$

- $S(\mathbf{x})$ is small if $(x_i - x_j)^2$ is small for large $W_{i,j}$
- Careful! This quantity in counterintuitive: large smoothness is achieved for non-smooth signals and vice-versa!

# Example




Smoothness decreases as the graph signal becomes more smooth

# Interpretation of the eigenvectors/eigenvalues

- For the eigenvectors of the Laplacian $\mathbf{u}_i$, we have

$$S(\mathbf{u}_i) = \mathbf{u}_i^T \mathbf{L} \mathbf{u}_i = \lambda_i$$

- New interpretation of the eigenvalues $\lambda_i$: **smoothness of the associated eigenvector**
- For one connex component, $\lambda_1 = 0$ and $\mathbf{u}_1 = \mathbf{1}_D$
  Constant eigenvector: perfect smoothness!

# How to interpret the graph spectrum

Parallels can be drawn between what we have seen in previous lectures and GSP:

- Notion of smoothness and low-frequency approximation
  Useful for denoising, interpolation...
- Notion of sparsity and bandlimitedness
  Useful for subsampling and reconstruction

# Graph filters

As in standard signal processing, the notion of filters can be defined in the graph or graph frequency domain. Given a graph signal $\mathbf{x}$, the filtered signal $\mathbf{y}$ writes as:

- **Graph domain:**

$$\mathbf{y} = \left( \sum_{i=0}^{p} h_i \mathbf{L}^i \right) \mathbf{x}$$

- **Graph frequency domain:**

$$\hat{\mathbf{y}} = \left( \sum_{i=0}^{p} h_i \boldsymbol{\lambda}^i \right) \hat{\mathbf{x}}$$

# Graph filters

- The link between both definitions comes from the fact that

$$\mathbf{L}^i = \mathbf{U}\boldsymbol{\lambda}^i\mathbf{U}^T$$

- Since $\boldsymbol{\lambda}$ is diagonal, the filtering in graph frequency domain goes back to computing

$$\hat{y}_j = \tilde{\lambda}_j \hat{x}_j$$

where

$$\tilde{\lambda}_j = \sum_{i=0}^{p} h_i \lambda_j^i$$

- Filtering can be achieved by building a function $h$ that transforms the eigenvalues (frequencies)

$$\tilde{\lambda}_j = h(\lambda_j)$$

# Low-pass filtering

- In Lecture 4, we have seen that low-pass filtering can be used for denoising or interpolation
- Example:
  - Search for a smooth approximation of a graph signal [Stankovic et al., 2019]

$$\min \|\mathbf{x} - \mathbf{y}\|_2^2 + \alpha \mathbf{y}^T \mathbf{L} \mathbf{y}$$

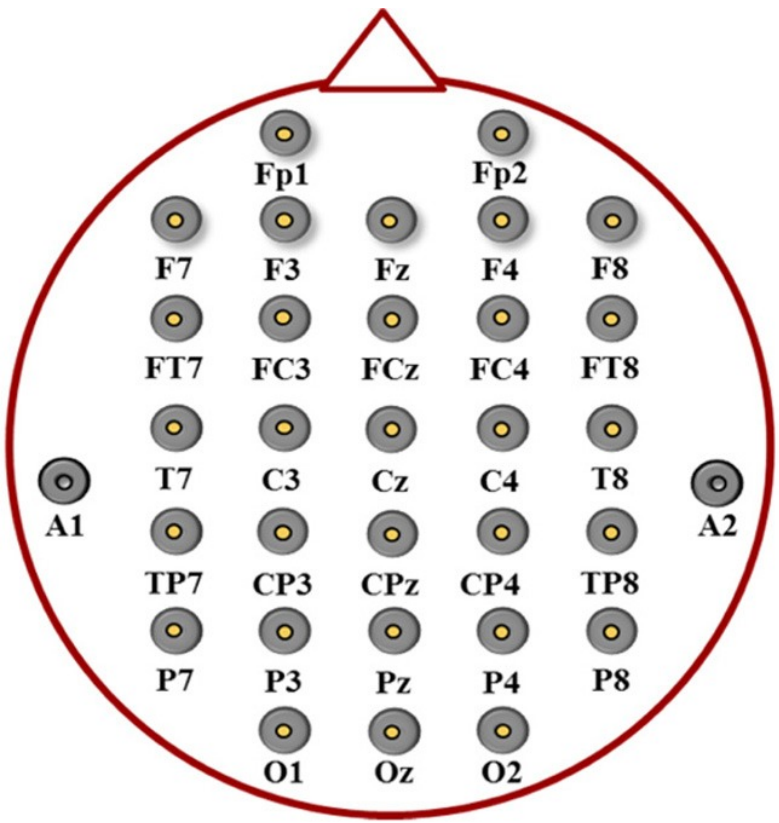
  - Closed form solution: low-pass filter

$$\mathbf{y} = (\mathbf{Id}_D + \alpha \mathbf{L})^{-1} \mathbf{x}$$

$$h(\lambda) = \frac{1}{1 + \alpha\lambda}$$

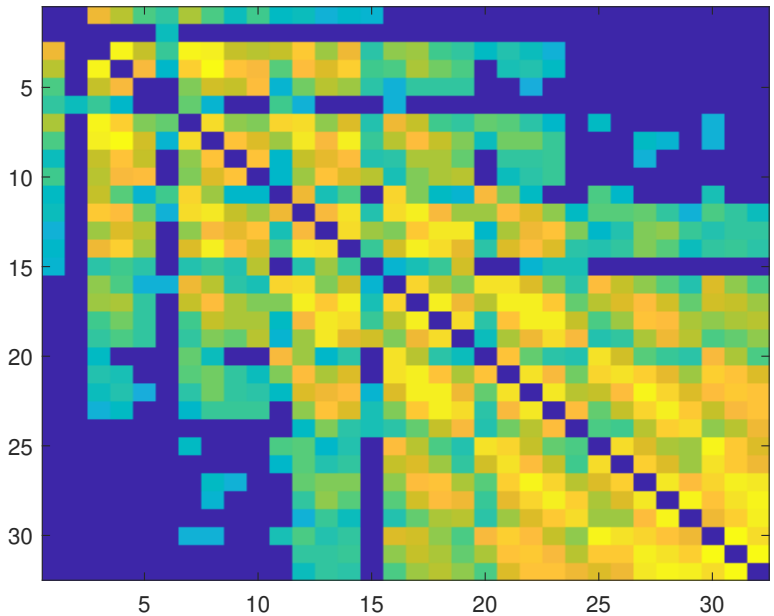
- Filtering can be applied independently to each multivariate sample $\mathbf{x}[n]$: it will use the graph information to smooth the signal, not in the time domain, but along the different signal dimensions

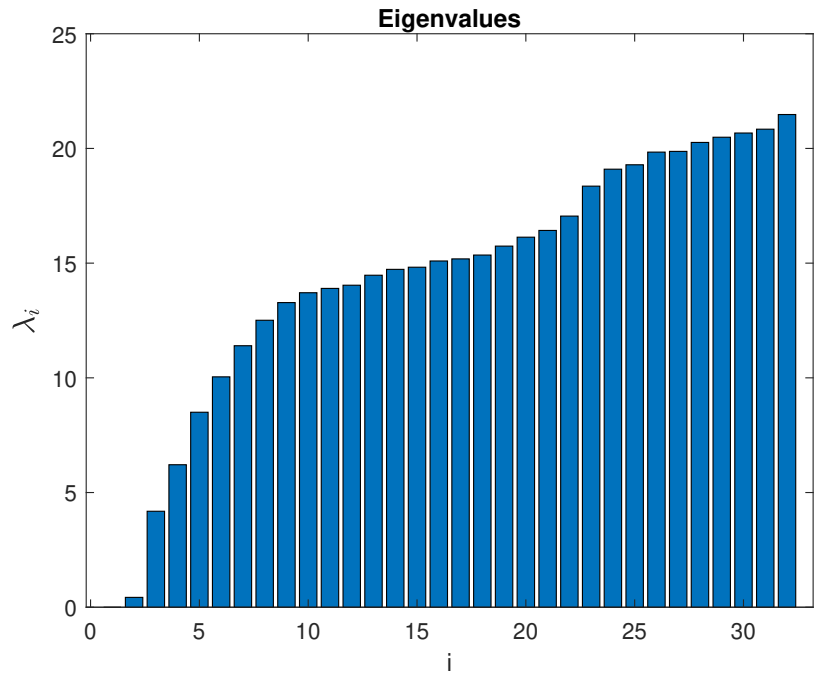
# Example: EEG




32-channel EEG

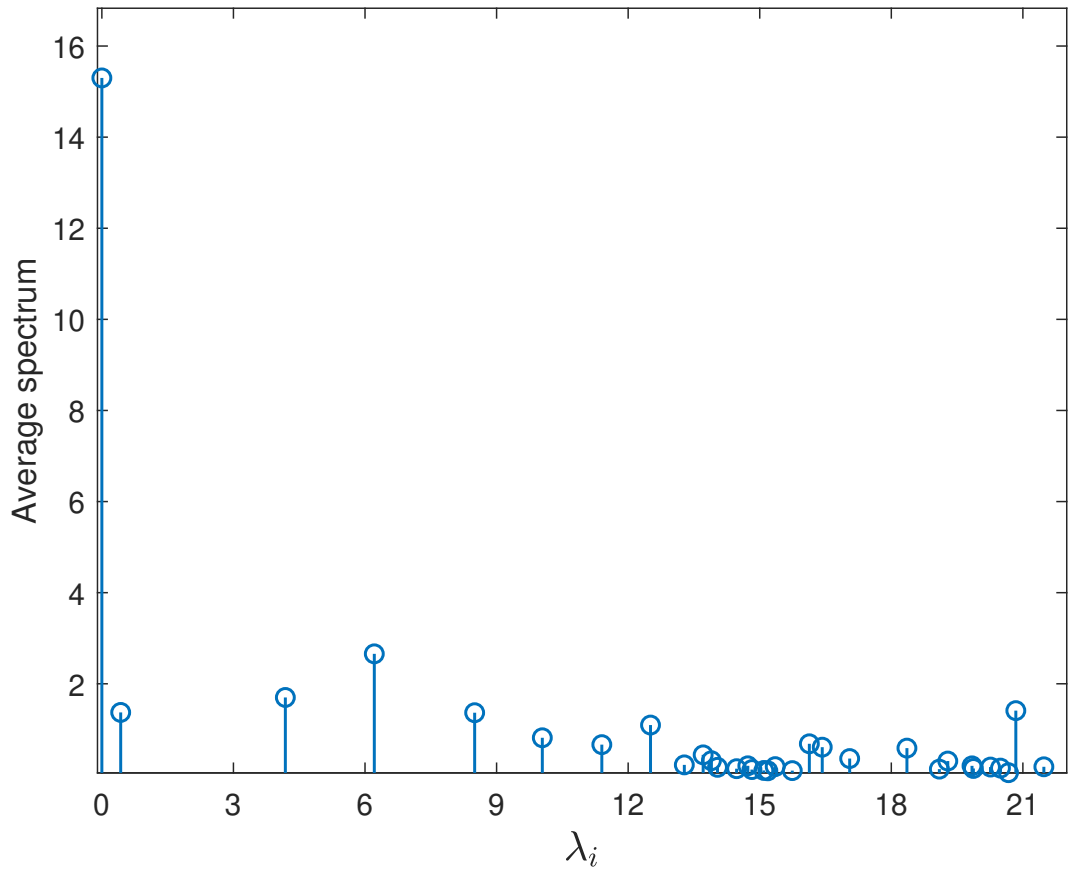
# Example: EEG



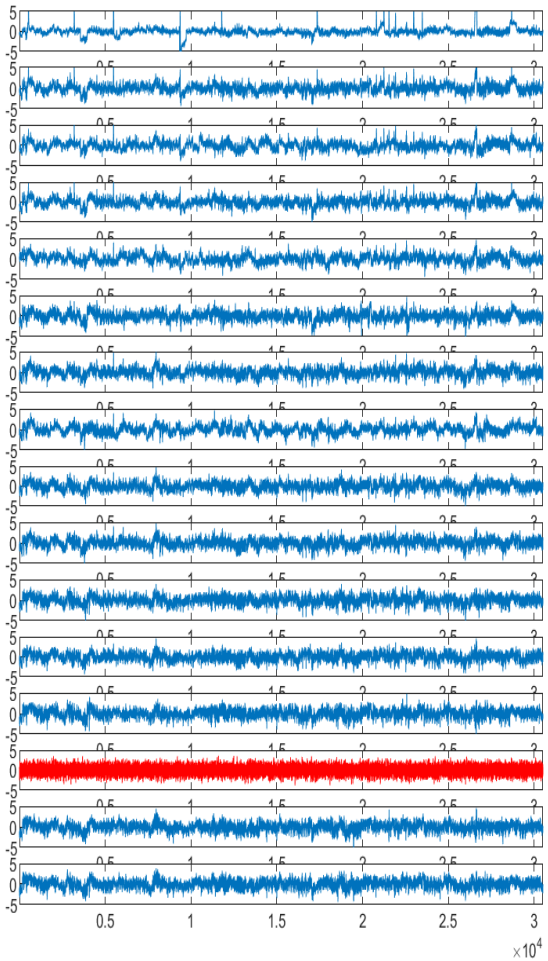

Thresholded covariance graph: one connex component
Weight matrix and eigenvalues

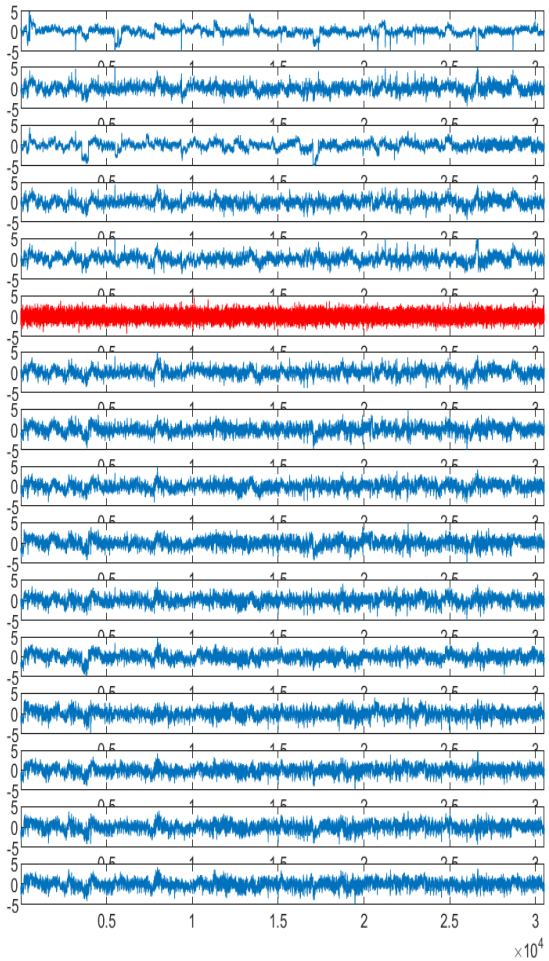
# Example: EEG




Average graph spectrum $\frac{1}{N}\sum_{n=1}^{N}|\hat{\mathbf{x}}[n]|^2$

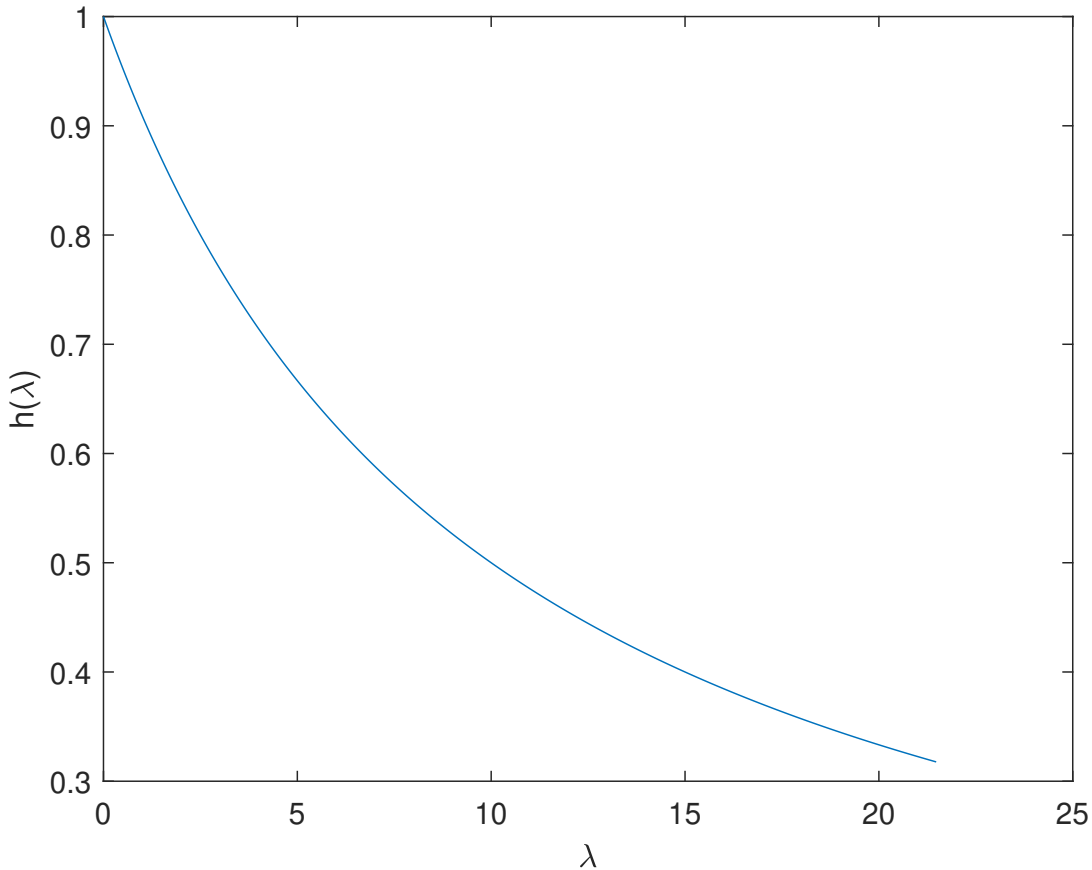
# Example: EEG




Two corrupted sensors: could we reconstruct the data by using the graph structure?

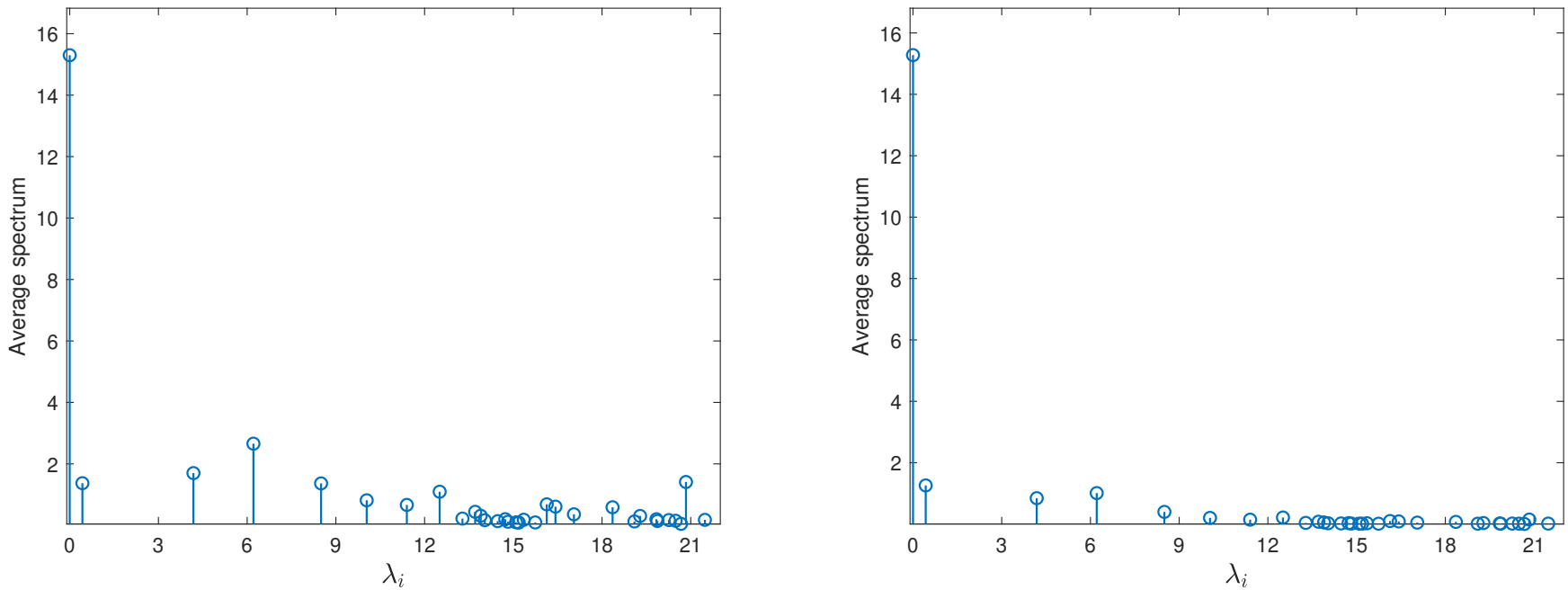
# Example: EEG




Low-pass filter design

$$h(\lambda) = \frac{1}{1 + 0.1\lambda}$$

# Example: EEG




Average graph spectrum before and after filtering

# Example: EEG

Left: original, right: corrupted

# Example: EEG

Left: original, right: reconstructed

# How to learn the graph structure?

- Given a multivariate time series $\mathbf{X}$, seen as $N$ graph signals $\mathbf{x}[1], \ldots, \mathbf{x}[N]$, how do we learn the graph structure?
- Important assumption: **graph-stationarity**
  - All multivariate samples $\mathbf{x}[n]$ are carried by the same graph
  - The graph structure stays unchanged
- Two learning strategies:
  - Ad-hoc computation
  - Learning algorithms based on constraints such as smoothness or bandlimitedness [Mateos et al., 2019 ; Dong et al., 2016]

# Ad hoc graph structure

- **Correlation**

$$W_{i,j} = \begin{cases} \text{corr}(\mathbf{x}_i, \mathbf{x}_j) & \text{if } \text{corr}(\mathbf{x}_i, \mathbf{x}_j) > \lambda \\ 0 & \text{else} \end{cases}$$

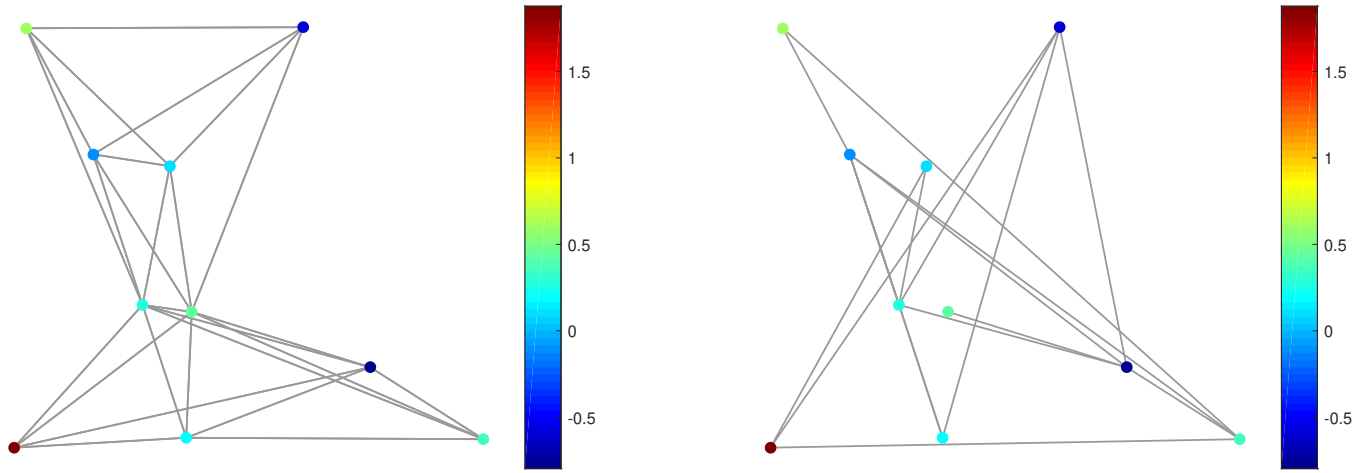
  Weights corresponds to the correlations between the signal dimensions

- **Distance** between sensor coordinates $c_i$ and $c_j$

$$W_{i,j} = \begin{cases} e^{-\gamma \|c_i - c_j\|_2^2} & \text{if } e^{-\gamma \|c_i - c_j\|_2^2} > \lambda \\ 0 & \text{else} \end{cases}$$

  Weights are small for sensors that are distant to each other

# Graph inference based on smoothness assumption




[Dong et al., 2016]

- **Aim :** Given a multivariate time series $\mathbf{X}$, seen as $N$ graphs signal of size $D$, learn the graph $\mathcal{G}$ that best explains the structure observed in the signal
- **Assumption :** The graph signals $\mathbf{x}[1], \ldots, \mathbf{x}[N]$ should be smooth for $\mathcal{G}$
- **Inputs :**

$$\mathbf{X} \in \mathbb{R}^{D \times N} : \text{input graph signals}$$

- **Outputs :**

$$\mathbf{L} = \mathbf{U}\Lambda\mathbf{U}^T : \text{Laplacian matrix of } \mathcal{G}$$

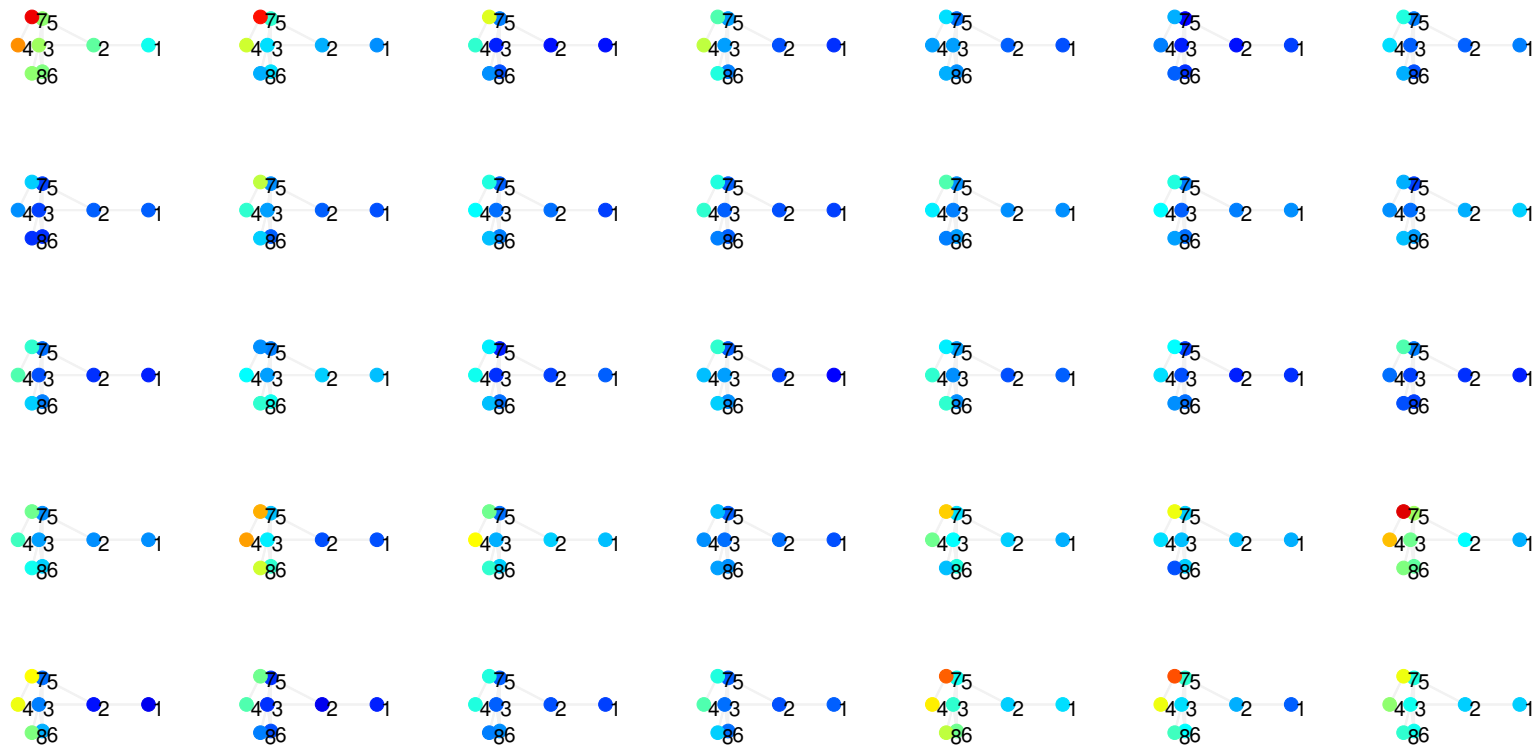
# Problem formulation

$$\min_{\mathbf{Y},\mathbf{L}} \|\mathbf{X}-\mathbf{Y}\|_F^2 + \alpha \operatorname{tr}(\mathbf{Y}^T\mathbf{L}\mathbf{Y}) + \beta \|\mathbf{L}\|_F^2$$

$$\text{s.t.} \quad \begin{cases} \operatorname{tr}(\mathbf{L}) = D \\ L_{i,j} = L_{j,i} \leq 0, i \neq j \\ \mathbf{L}\mathbf{1}_D = \mathbf{0}_D \end{cases}$$

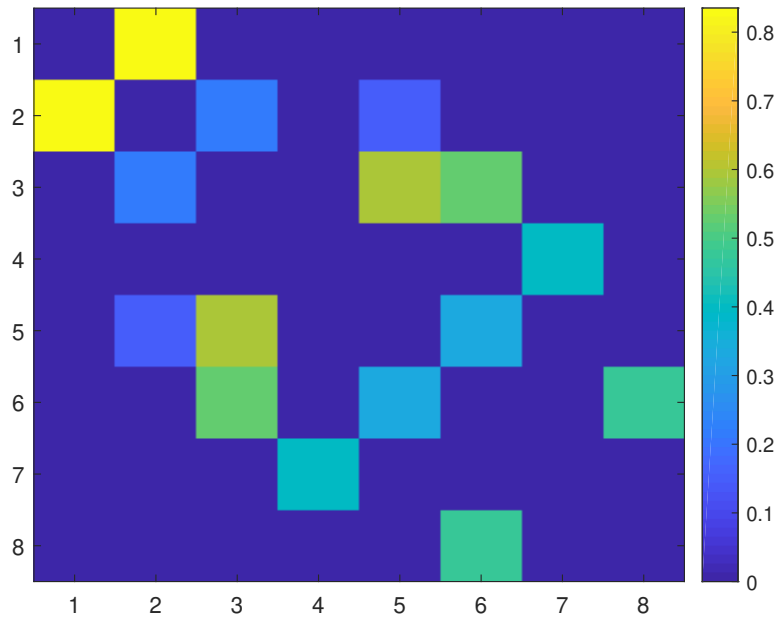
- Search for a *smooth* approximation $\mathbf{Y}$ of signals $\mathbf{X}$
- Constraints impose that $\mathbf{L}$ has a Laplacian structure
- Non-convex problem: alternate minimization w.r.t. $\mathbf{Y}$ and $\mathbf{L}$
- Minimization w.r.t. $\mathbf{L}$ can be solved with quadratic programming and w.r.t. $\mathbf{Y}$ admits a closed-form solution

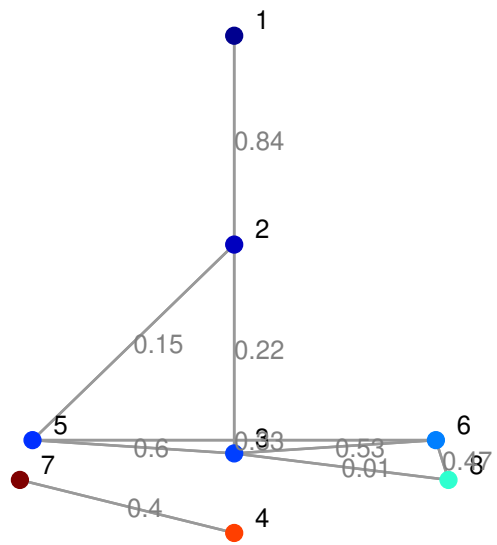
# Example




$N = 15$ graphs signals with $D = 8$ dimensions (EEG)

# Example




Learned graph (weight matrix and graph)

# References


- Zivot, E., & Wang, J. (2006). Vector autoregressive models for multivariate time series. Modeling financial time series with S-PLUS, 385-429.
- Lütkepohl, H. (2013). Vector autoregressive models. In Handbook of Research Methods and Applications in Empirical Macroeconomics. Edward Elgar Publishing.
- Hurvich, C. M., & Tsai, C. L. (1993). A corrected Akaike information criterion for vector autoregressive model selection. Journal of time series analysis, 14(3), 271-279.
- Garcia-Cardona, C., & Wohlberg, B. (2018, October). Convolutional dictionary learning for multi-channel signals. In 2018 52nd Asilomar Conference on Signals, Systems, and Computers (pp. 335-342). IEEE.
- Moreau, T., Oudre, L., & Vayatis, N. (2018, July). Dicod: Distributed convolutional coordinate descent for convolutional sparse coding. In International Conference on Machine Learning (pp. 3626-3634). PMLR.
- Tropp, J. A., Gilbert, A. C., & Strauss, M. J. (2006). Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit. Signal processing, 86(3), 572-588.
- Tropp, J. A. (2006). Algorithms for simultaneous sparse approximation. Part II: Convex relaxation. Signal Processing, 86(3), 589-602.
- Cotter, S. F., Rao, B. D., Engan, K., & Kreutz-Delgado, K. (2005). Sparse solutions to linear inverse problems with multiple measurement vectors. IEEE Transactions on Signal Processing, 53(7), 2477-2488.
- Ortega, A., Frossard, P., Kovačević, J., Moura, J. M., & Vandergheynst, P. (2018). Graph signal processing: Overview, challenges, and applications. Proceedings of the IEEE, 106(5), 808-828.
- Chen, S., Sandryhaila, A., Moura, J. M., & Kovacevic, J. (2014, December). Signal denoising on graphs via graph filtering. In 2014 IEEE Global Conference on Signal and Information Processing (GlobalSIP) (pp. 872-876). IEEE.
- Narang, S. K., Gadde, A., & Ortega, A. (2013, May). Signal processing techniques for interpolation in graph structured data. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing (pp. 5445-5449). IEEE.
- Mateos, G., Segarra, S., Marques, A. G., & Ribeiro, A. (2019). Connecting the dots: Identifying network structure via graph signal processing. IEEE Signal Processing Magazine, 36(3), 16-43.
- Stankovic, L., Mandic, D. P., Dakovic, M., Kisil, I., Sejdic, E., & Constantinides, A. G. (2019). Understanding the basis of graph signal processing via an intuitive example-driven approach [lecture notes]. IEEE Signal Processing Magazine, 36(6), 133-145.
- Dong, X., Thanou, D., Frossard, P., & Vandergheynst, P. (2016). Learning Laplacian matrix in smooth graph signal representations. IEEE Transactions on Signal Processing, 64(23), 6160-6173.

# List of possible topics/projects

- Wang, D., Zheng, Y., Lian, H., & Li, G. (2020). High-dimensional vector autoregressive time series modeling via tensor decomposition. Journal of the American Statistical Association, 1-42.
  How to apply VAR models to high dimensional data
- Athanasopoulos, G., & Vahid, F. (2008). VARMA versus VAR for macroeconomic forecasting. Journal of Business & Economic Statistics, 26(2), 237-252.
  How to use VAR and VARMA model for economic forecasting
- Li, H. (2019). Multivariate time series clustering based on common principal component analysis. Neurocomputing, 349, 239-247.
  How to use PCA to perform clustering on multivariate time series
- Chen, X., & Sun, L. (2020). Low-rank autoregressive tensor completion for multivariate time series forecasting. arXiv preprint arXiv:2006.10436.
  How to use tensor structure to forecast multivariate time series
- Barthélemy, Q., Gouy-Pailler, C., Isaac, Y., Souloumiac, A., Larue, A., & Mars, J. I. (2013). Multivariate temporal dictionary learning for EEG. Journal of neuroscience methods, 215(1), 19-28.
  How to apply multivariate dictionary learning for EEG data
- Cotter, S. F., Rao, B. D., Engan, K., & Kreutz-Delgado, K. (2005). Sparse solutions to linear inverse problems with multiple measurement vectors. IEEE Transactions on Signal Processing, 53(7), 2477-2488.
  How use multivariate sparse coding for solving inverse problems
- Dong, X., Thanou, D., Rabbat, M., & Frossard, P. (2019). Learning graphs from data: A signal representation perspective. IEEE Signal Processing Magazine, 36(3), 44-63..
  How to learn a graph from smooth graph signals
- Kumar, S., Ying, J., de Miranda Cardoso, J. V., & Palomar, D. P. (2020). A Unified Framework for Structured Graph Learning via Spectral Constraints. Journal of Machine Learning Research, 21(22), 1-60.
  How to learn a graph with spectral constraints