# Signal Processing Tutorial

## Research School: Mathematics, Signal Processing and Learning

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

CIRM, Marseille, France
January 27th, 2021

# Contents

1. Introduction and data

1.1 Problem statement

1.2 Data

2. Basic tools and notions for signal processing

2.1 Discrete Fourier Transform

2.2 Stationarity, ergodicity and autocorrelation function

2.3 Spectrogram

3. Change-point detection

3.1 Cost functions

3.2 Search method

4. Convolutional dictionary learning

4.1 Problem

4.2 Alternated resolution

# Contents

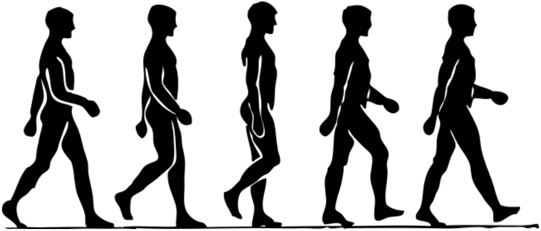
1. Introduction and data
1.1 Problem statement
1.2 Data

2. Basic tools and notions for signal processing

3. Change-point detection

4. Convolutional dictionary learning

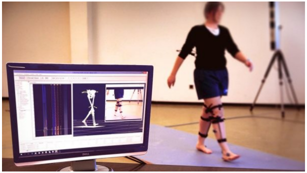
# Gait analysis



Why is is important to study locomotion?

- Most common dynamic human activity
- Can reveal a large number of neurological, orthopedic, rheumatological disorders...
- Strong influence on daily life : risk of falling, frailty, autonomy, dependency...

# Gait analysis



How can we study locomotion?

- Early tests: clinical examination by the physician, functional tests, clinical questionnaires

| | |
|---|---|
| + | Easy to perform, use of clinical expertise |
| - | Lack of precision, difficult to objectively compare two sessions |

- Dedicated platforms for the study of locomotion: instrumented mats, video/optical systems

| | |
|---|---|
| + | Great precision, extraction of a large number of useful features, objective quantification |
| - | Expensive, difficult to put in practice |

# Main principles

★ Objective quantification of human gait

→ Use of sensors and physiological measurements

★ Longitudinal follow-up and inter-individual comparison

→ Need for a fixed protocol

★ Experimentation outside the laboratory and on the field

→ User-mounted sensors and fully automatic device for consultation and routine use

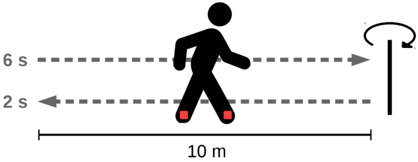
★ Clean data

→ Control of the entire measurement chain, robust and reproducible algorithms

★ Willingness to capture the expertise of the clinician

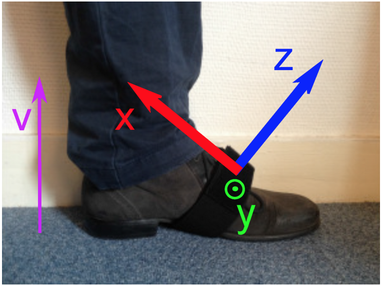
→ Clinical annotations and metadata

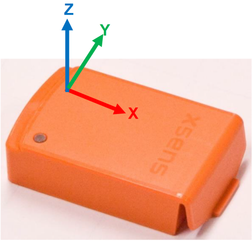
# Protocol




- Sequence of activities:
  - stand for 6 s,
  - walk 10 m at preferred walking speed on a level surface to a previously shown turn point,
  - turn around (without previous specification of a turning side),
  - walk back to the starting point,
  - stand for 2 s.
- Subjects walked at their comfortable speed with their shoes and without walking aid.

# Sensors



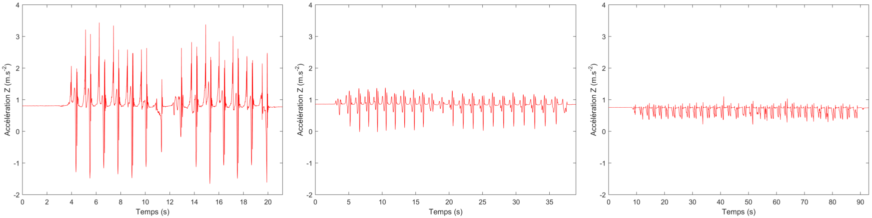

- IMU (Inertial Measurement Unit) record linear accelerations (3D), angular velocities (3D) and magnetic fields (3D) on each foot
- Sensor frame consists of 3-axis $(X, Y, Z)$
- **For this tutorial: angular velocity aroung axis $Y$**

# Database

221 recordings:

- Healthy subjects had no known medical impairment (labelled as ”T” for Témoin).
- The orthopedic group is composed of 3 cohorts of distinct pathologies: lower limb osteoarthrosis (ArtH, ArtG), cruciate ligament injury (LCA), knee injury (Genou)
- The neurological group is composed of 2 cohorts: cerebellar disorder (CER) and radiation induced leukoencephalopathy (LER)

# Main scientific questions




Sujet sain — Pathologie neurologique peu sévère — Pathologie neurologique sévère

## Non-stationary signals

→ How can be detect the different regimes (stop, walking, U-turn...)?

## Presence of repetitive patterns: the steps

→ What are they? How could we automatically extract them?

## Robust feature extraction

→ How could be extract relevant features for longitudinal follow-up and inter-individual comparison?

# Contents

1. Introduction and data

2. Basic tools and notions for signal processing

2.1 Discrete Fourier Transform

2.2 Stationarity, ergodicity and autocorrelation function

2.3 Spectrogram

3. Change-point detection

4. Convolutional dictionary learning

# What is signal?

- A time series (or signal) is a series of data points indexed in time order
- In practice, array of real numbers of size $D \times N$ where $D$ is the number of dimensions and $N$ the number of samples
  - Sample number $n$

| $n$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |

  - Time series values $x[n]$

| $x[n]$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0.7 | 0.2 | 0.8 | 0.9 | 0.3 | 0.2 | 0.7 |
| | 0.4 | 0.1 | 0.6 | 0.2 | 0.5 | 0.6 | 0.3 |

  - Time stamps $t[n]$

| $t[n]$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 16:30:01 | 16:30:23 | 16:31:43 | 16:32:38 | 16:33:06 | 16:33:16 | 16:33:56 |

# Two visions: physics vs. statistics

- The notion of time have been used and modeled in physics since 18th century and before (eg. Fourier transform).
  **First vision :** a time series $x[1:N]$ is the result of the digitization of a physical phenomenon $x(t)$. Physical properties of this phenomenon can be retrieved and analyzed through the study of $x[1:N]$ (and vice/versa).
- Randomness can also play a part to model a wider class of signals.
  **Second vision :** a time series $x[1:N]$ is a realization of a stochastic process $X[1:N]$. Statistical properties of this phenomenon can be retrieved and analyzed through the study of $x[1:N]$ (and vice/versa).

In most cases, both approaches can be combined.

# Some useful signal processing tools

In the following, we will introduce basic signal processing tools and apply them to our signals:

- Discrete Fourier Transform (DFT)
- Notion of stationarity, ergodicity and autocorrelation function
- Spectrogram

# Sampling and Fourier analysis

- Most tools for signal processing are derived from Fourier analysis
- In this context, we assume that $\mathbf{x}$ corresponds to the discrete measurement of a continuous signal $x(t)$
- **Sampling theory:** uniform sampling period $T_s$ and sampling frequency $F_s = \frac{1}{T_s}$

$$x[n] = x(nT_s)$$

# Discrete Fourier Transform (DFT)

$$\boxed{X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j2\pi \frac{kn}{N}} \text{ pour } 0 \leq k \leq N-1}$$

where $N$ is the number of samples

- The space between two observable frequencies is called **frequency resolution**

$$\boxed{\Delta f = \frac{F_s}{N}}$$

- $X[k]$ corresponds to the DFT for the physical frequency

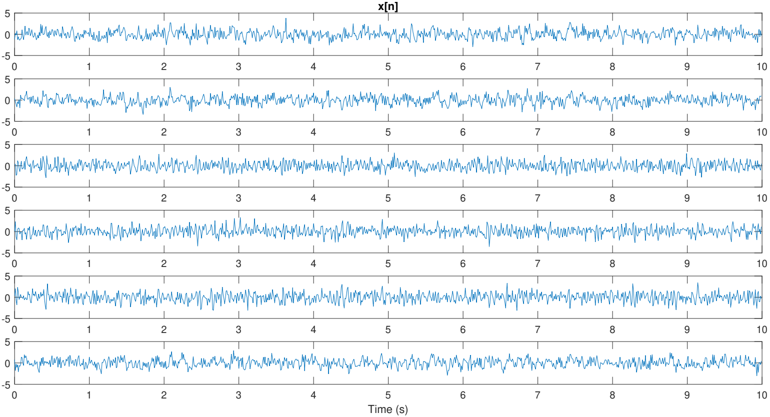
$$\boxed{f[k] = k\frac{F_s}{N} \text{ for } 0 \leq k \leq N-1}$$

- No physical frequency greater than $\frac{F_s}{2}$ can be observed (Nyquist theorem).

# Spectral analysis

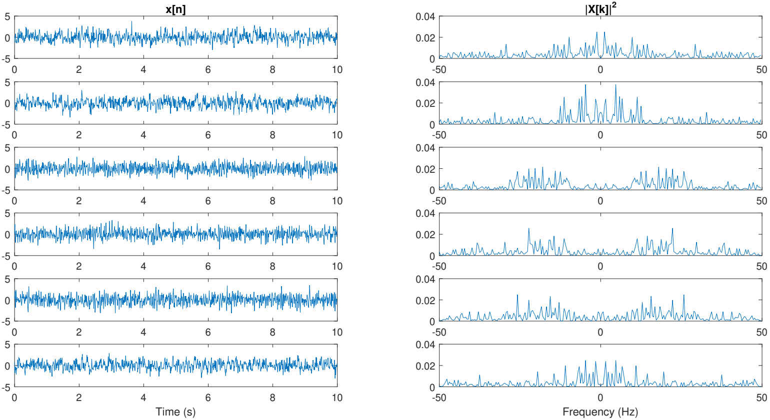
- $X[k]$ is a complex quantity: most of the time, we use the squared absolute values $|X[k]|^2$ instead
- The analysis of the quantity $|X[k]|^2$ can allow to discover interesting properties of the time series
- $|X[k]|^2$ with low frequencies $f_k$ correspond to phenomena with smooth variations
- $|X[k]|^2$ with large frequencies $f_k$ correspond to phenomena with fast variations
- One very useful plot consists in plotting $|X[k]|^2$ as a function of $f[k]$: such plot is often referred to as **spectrum** (hence spectral analysis)

# Example




Two classes of signals?

# Example




Can be distinguished based on their DFT coefficients

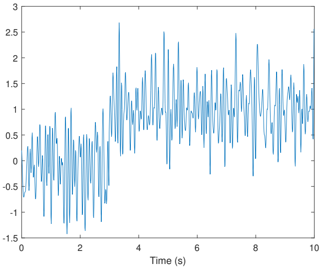
# Statistical vision

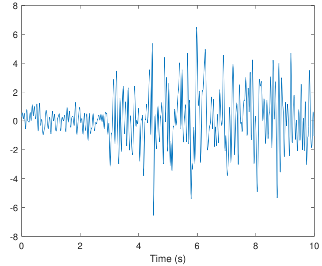
- In order to better understand the properties of a signal, deterministic analysis such as Fourier has been extended to probabilistic and statistical analysis
- In this context, we assume that $x[1:N]$ corresponds to a realization of a stochastic process $X[1:N]$
- Each $X[n]$ can be seen as a random variable
- Statistical properties of $X[1:n]$ can be retrieved from estimates based on $x[1:n]$

# Two fundamental properties

- **Stationarity :** The statistical properties of the time series do not change over time
  - Order 1
  $$\forall n, \quad \mathbb{E}\left[X[n]\right] = \mu$$
  - Order 2
  $$\forall n_1, n_2, \quad \mathbb{E}\left[X[n_1]X[n_2]\right] = \gamma_X[|n_2 - n_1|]$$
  - Order 1 + Order 2 $\rightarrow$ wide-sense stationarity (most common assumption)
- **Ergodicity :** Ensemble mean and the mean over time are equal (implies stationarity of order 1)

# Stationarity vs. non-stationarity




- None of these signals are stationary
- In order to prevent this to happen, two solutions exist
  - Divide the signals into small frames where the signal is assumed to be stationary and ergodic (see later: spectrogram)
  - Use a change-point detection algorithm to detect these changes and work separately on each segment (see later: change-point detection)

# Autocorrelation

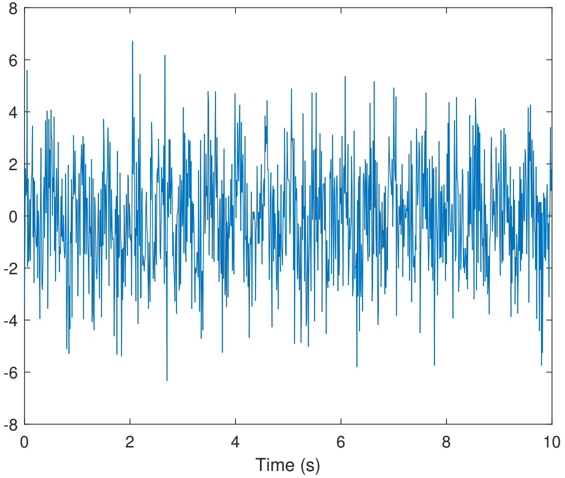
Assuming that $X[1:n]$ is ergodic and wide-sense stationary, we can estimate from $x[1:n]$ the autocorrelation function

- Autocorrelation function

$$\boxed{\hat{\gamma}_x^{biased}[m] = \frac{1}{N}\sum_{n=0}^{N-1} x[n]x[n+m]} \quad \text{where } x[n] = 0 \text{ for } n \neq 0 \ldots N-1$$

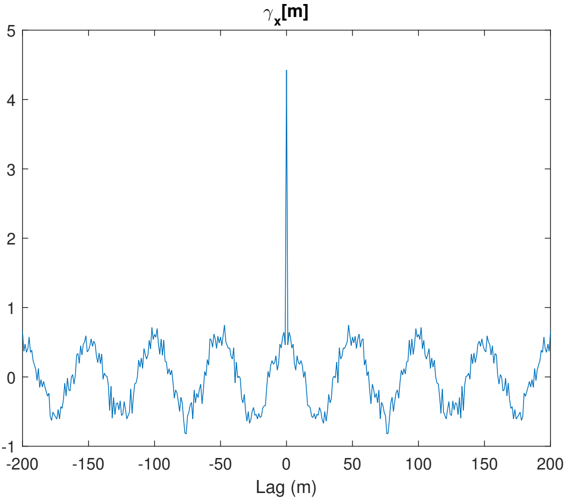
- This function helps (among other things) to discover the presence of periodic components within a signal

# How to use the autocorrelation function



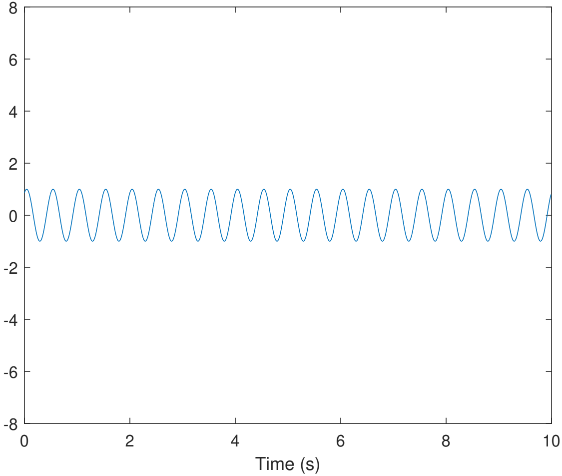

Original signal, sampling frequency 100 Hz

# How to use the autocorrelation function




Autocorrelation function, peaks are visible for lags multiple of 50

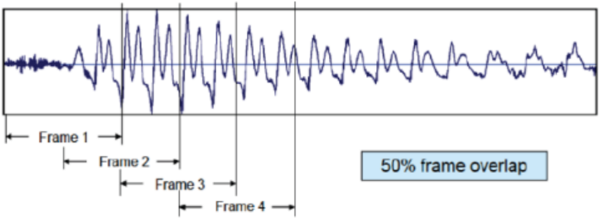
# How to use the autocorrelation function




A periodic signal with period $50 \times \frac{1}{100} = 0.5$ sec was hiding!

# Spectrogram

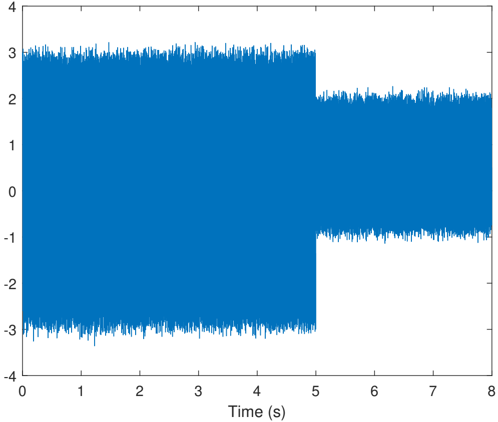
- When the properties of the time series tend to change with time (non-stationary signals), it is more careful to compute the DFT on sliding windows
- By sliding the window along the signal, we recover a time-frequency representation called **spectrogram**




- Matrix representation: each column corresponds to the DFT on the window of interest.

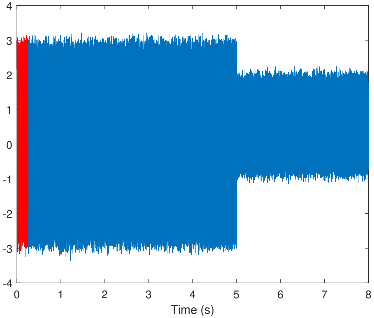
X-axis: frame number, Y-axis: frequency bin

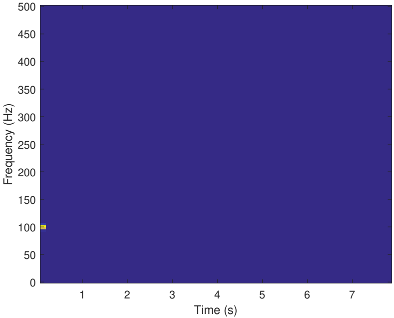
# Example




Original signal, $F_s = 1000$ Hz

# Example

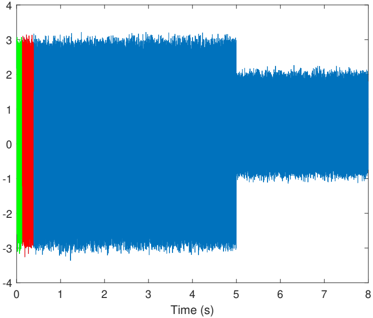


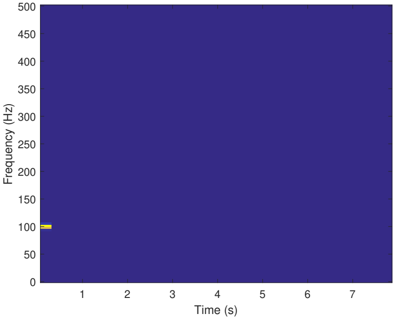




Window length $N_w = 256$

Computation of the DFT on the first frame and storage in the spectrogram matrix...

# Example

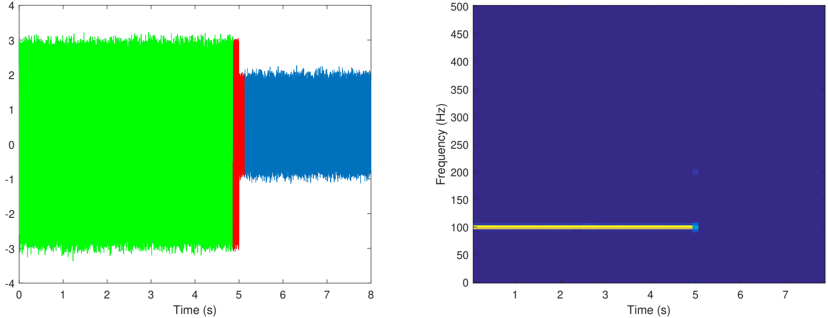



Window length $N_w = 256$

Computation of the DFT on the second frame and storage in the spectrogram matrix...

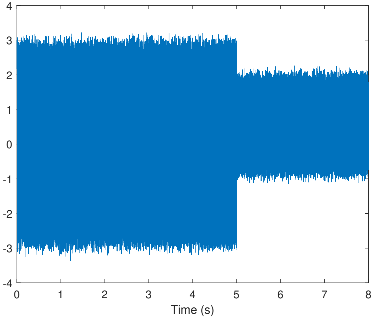
# Example

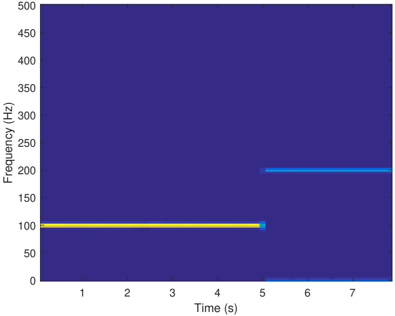



Window length $N_w = 256$

Same process...

# Example



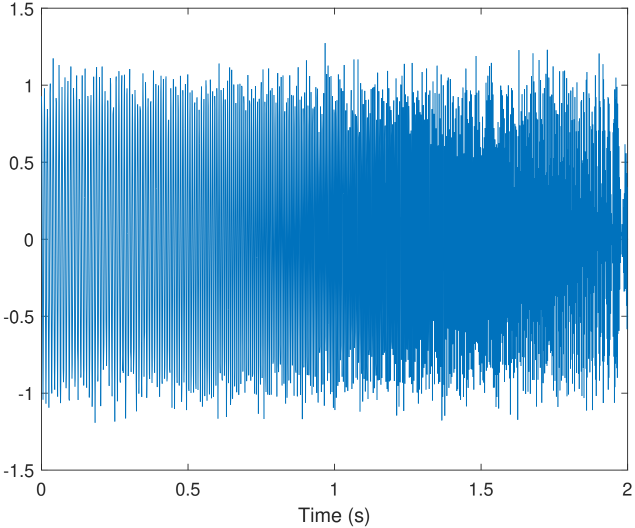




Window length $N_w = 256$
Final result

# DFT vs. Spectrogram

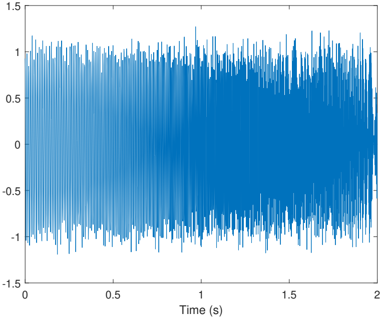
- Only use DFT when you are sure that there is no abrupt changes in the time series
- Note that using DFT will tend to average the frequency content on the whole time series, which can be tricky in some application contexts
- For safety, always first visualize the spectrogram to make sure that no significant changes occur

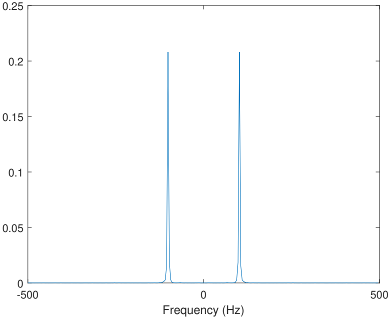
# DFT vs. Spectrogram




Periodic phenomenon ? (sinusoid ?)

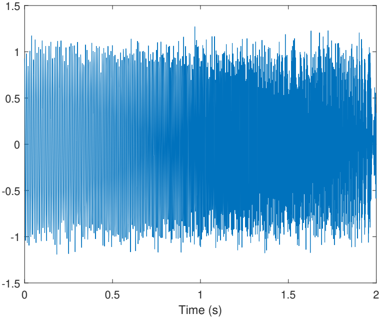
# DFT vs. Spectrogram

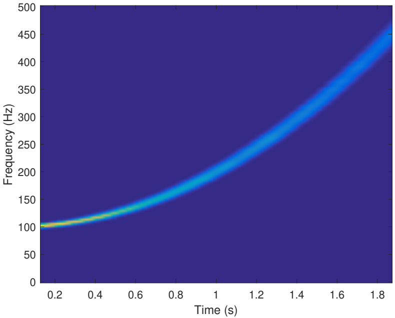



DFT suggests a sinusoidal phenomenon around frequency 100 Hz

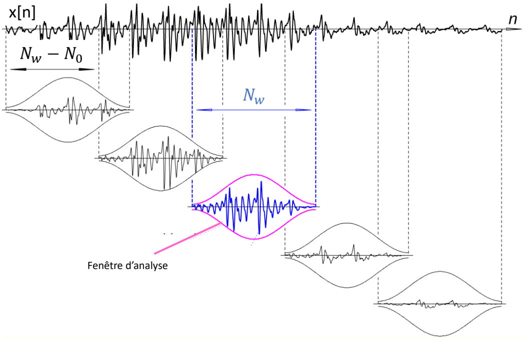
# DFT vs. Spectrogram




In fact, chirp signal between 100 and 500 Hz ‼

# Hyperparameters for spectrogram




- $N_w$ : window length (in samples)
  Often taken as a power of 2 (for FFT) and linked to the desired frequency resolution.
- $N_o$ : overlap between two successive frames (in samples)
  Often taken as 50% or 75% of the window length and characterizes the time resolution (optimal when $N_o = N_w - 1$)
- $w$ : analysis window (Hann, Hamming, Blackman...)
  Traditionally, in order to limit side effects, the signal frame is multiplied by an analysis window

# Contents

1. Introduction and data

2. Basic tools and notions for signal processing

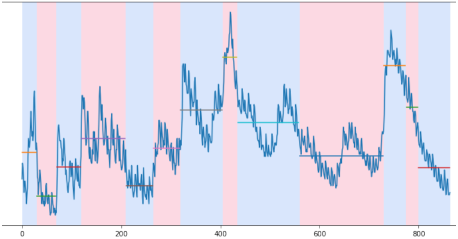
3. Change-point detection

3.1 Cost functions

3.2 Search method

4. Convolutional dictionary learning

# Problem 1: Change-Point Detection

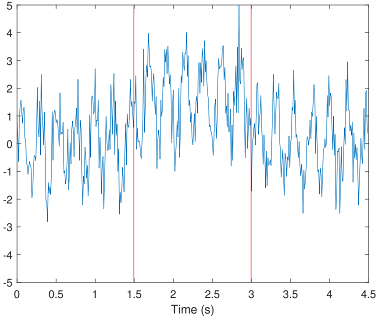



## Change-Point Detection

**Given a time series x, retrieve the times $(t_1, \ldots, t_K)$ where a significant change occurs**

- Necessitates to estimate both the change-points but also the number of changes $K$
- Highly depends on the meaning given to change

# Problem statement



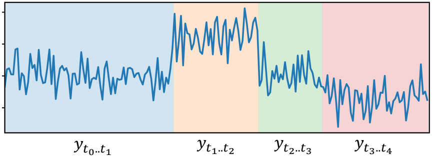

- When the changes are abrupt or when the estimation of the change-points is relevant in the context, we can use **change-point detection** methods
- Let assume that signal $x[n]$ undergoes abrupt changes at times

$$\mathcal{T}^* = (t_1^*, \dots, t_{K^*}^*)$$

- Goal: retrieve the number of change-points and $K^*$ and their times $\mathcal{T}^*$
- One assumption: offline segmentation (but can easily be adapted to online setting) [Truong et al., 2020]

# Problem statement

$$(\hat{t}_1, \ldots, \hat{t}_K) = \underset{(t_1, \ldots, t_K)}{\text{argmin}} \sum_{k=0}^{K} c(x[t_k : t_{k+1}])$$

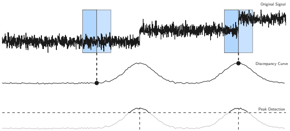



## Cost function $c(.)$

- Measures the homogeneity of the segments
- Choosing $c(.)$ conditions the type of change-points that we want to detect
- Often based on a probabilistic model for the data

## Problem solving

- Optimal resolution with dynamic programming
- Approximate resolution (sliding windows...)



# Cost function

$$(\hat{t}_1, \ldots, \hat{t}_K) = \underset{(t_1, \ldots, t_K)}{\text{argmin}} \sum_{k=0}^{K} c(x[t_k : t_{k+1}])$$

Convention : $t_0 = 0$, $t_{K+1} = N$

- Function $c(.)$ is characteristic of the notion of *homogeneity*
- The most common cost functions are linked to parametric probabilistic models: in this case change-points are defined as changes in the parameters of a probability density function [Basseville et al., 1993]
- Non parametric cost functions can also be introduced when no model is available

# Maximum likelihood estimation

Given a parametric family of distribution densities $f(\cdot|\theta)$ parametrized with $\theta \in \Theta$, a cost function can be derived:

$$c_{ML}(x[a:b]) = -\sup_{\theta} \sum_{n=a+1}^{b} \log f\left(x[n]|\theta\right)$$

- Corresponds to the assumption that on a regime, samples are i.i.d. according to a parametric distribution density
- On each regime, the parameters are estimated through maximum likelihood estimation
- This model can be adapted to several situations: change in mean, change in variance, change in both mean and variance...

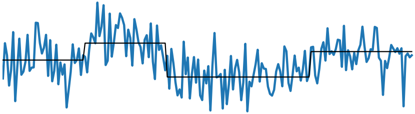
# Change in mean

The most popular is indubitably the L2 norm [Page, 1955]

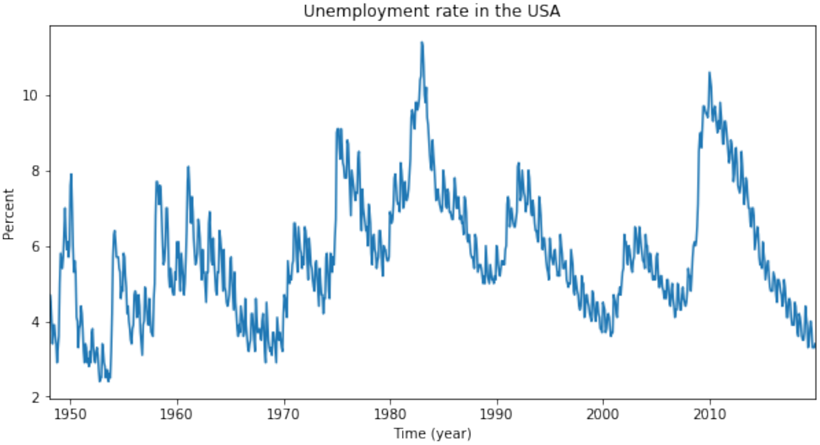
$$c_{L_2}(x[a:b]) = \sum_{n=a+1}^{b} \|x[n] - \mu_{a:b}\|_2^2$$

where $\mu_{a:b}$ is the empirical mean of the segment $x[a:b]$.

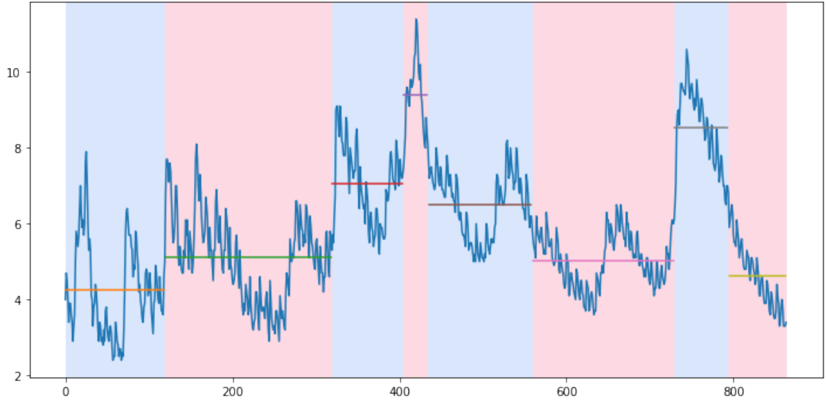
- Particular case of $c_{ML}$ with Gaussian model with fixed variance
- Allows to detect changes in mean



# Example

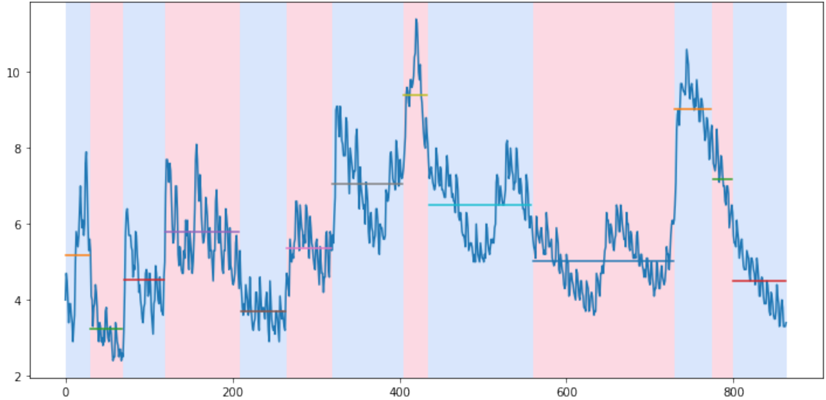

Unemployment rate in the USA
Percent
Time (year)
2
4
6
8
10
1950
1960
1970
1980
1990
2000
2010

# Example: Change-Point Detection with $c_{L_2}$




$K = 7$

# Example: Change-Point Detection with $c_{L_2}$




$K = 12$

# Search method

$$\boxed{(\hat{t}_1, \ldots, \hat{t}_K) = \underset{(t_1,\ldots,t_K)}{\text{argmin}} \sum_{k=0}^{K} c(x[t_k : t_{k+1}])}$$

Convention : $t_0 = 0$, $t_{K+1} = N$

- Several methods can be used to solve this problem with a fixed $K$
- Optimal resolution with dynamic programming: find the true solution of the problem (but costly)
- Approximated resolution with windows: test for one unique change-point on a window

# Optimal resolution

- By denoting

$$\mathcal{V}(\mathcal{T}, \mathbf{x}) = \sum_{k=0}^{K} c(x[t_k : t_{k+1}])$$

we can see that

$$\begin{aligned}
\min_{|\mathcal{T}|=K} V(\mathcal{T}, \mathbf{x}) &= \min_{0=t_0<t_1<\cdots<t_K<t_{K+1}=N} \sum_{k=0}^{K} c(x[t_k : t_{k+1}]) \\
&= \min_{t \leq T-K} \left[ c(x[0:t]) + \min_{t_0=t<t_1<\cdots<t_{K-1}<t_K=T} \sum_{k=0}^{K-1} c(x[t_k : t_{k+1}]) \right] \\
&= \min_{t \leq T-K} \left[ c(x[0:t]) + \min_{|\mathcal{T}|=K-1} V(\mathcal{T}, x[t:N]) \right]
\end{aligned}$$

- Recursive problem: resolution with dynamic programming [Bai et al., 2003]
- Two steps: computation of the cumulative costs + determination of the change-points

# Optimal resolution

**Algorithm 1** Algorithm `Opt`

**Input:** signal $\{y_t\}_{t=1}^{T}$, cost function $c(\cdot)$, number of regimes $K \geq 2$.
**for all** $(u, v)$, $1 \leq u < v \leq T$ **do**
  Initialize $C_1(u, v) \leftarrow c(\{y_t\}_{t=u}^{v})$.
**end for**
**for** $k = 2, \ldots, K-1$ **do**
  **for all** $u, v \in \{1, \ldots, T\}, v - u \geq k$ **do**
    $C_k(u, v) \leftarrow \min\limits_{u+k-1 \leq t < v} C_{k-1}(u, t) + C_1(t+1, v)$
  **end for**
**end for**
Initialize $L$, a list with $K$ elements.
Initialize the last element: $L[K] \leftarrow T$.
Initialize $k \leftarrow K$.
**while** $k > 1$ **do**
  $s \leftarrow L(k)$
  $t^* \leftarrow \operatorname{argmin}_{k-1 \leq t < s} C_{k-1}(1, t) + C_1(t+1, s)$
  $L(k-1) \leftarrow t^*$
  $k \leftarrow k - 1$
**end while**
Remove $T$ from $L$
**Output:** set $L$ of estimated breakpoint indexes.

Complexity of $\mathcal{O}(KN^2)$

# Contents

1. Introduction and data

2. Basic tools and notions for signal processing

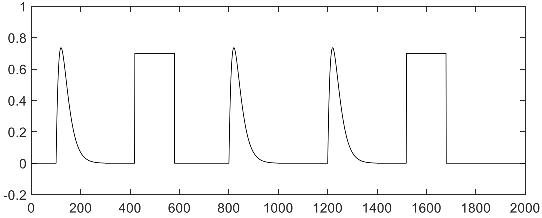
3. Change-point detection

4. Convolutional dictionary learning

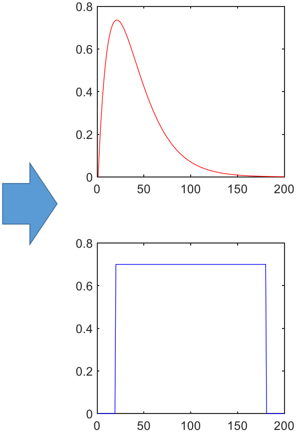
4.1 Problem

4.2 Alternated resolution

# Problem 2: Pattern Extraction




Input time series




Extracted patterns

# Problem 2: Pattern Extraction

**Pattern Extraction**

**Given an input time series x (or a set of time series), learn a dictionary of patterns $\mathcal{P}$**

- A template is a *shape* that appear repetitively in the time series (but kinda blurry notion)
- All templates are supposed to have the same length (for sake of simplicity)
- The extracted patterns can be used to characterize the time series, or studied individually

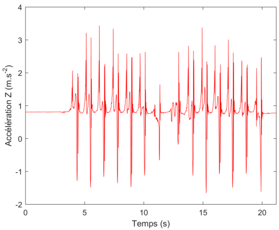
# Dictionary-based pattern extraction

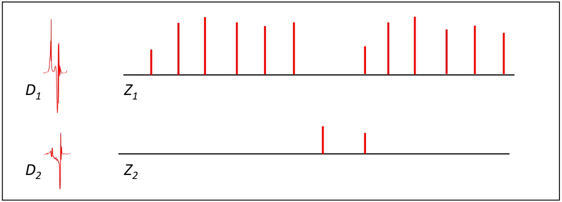
- Finding patterns in a time series can be seen as a dictionary learning optimization problem
- Given an input time series $\mathbf{x}$, learn a dictionary of $K$ patterns $\mathbf{d}_k$ of length $L$
- These patterns can be activated : activations $\mathbf{z}_k$ of length $N - L + 1$

$$z_k[n] \neq 0 \text{ if pattern } \mathbf{d}_k \text{ is activated at time } n$$

[Grosse et al., 2007 ; Wohlberg, 2014]

# Convolutional dictionary learning




## Convolutional dictionary learning

Given a time series $\mathbf{x}$, number of pattern $K$ and pattern length $L$, learn

- Patterns $\mathbf{d}_k$ of length $L$
- Activation signals $\mathbf{z}_k$ of length $N - L + 1$

$$x[n] = \sum_{k=1}^{K} (\mathbf{z}_k * \mathbf{d}_k)[n] + e[n]$$

# Optimization problem

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

- **Normalization constraint** for the dictionary atoms $\mathbf{d}_k$, that prevents numerical instabilities (otherwise setting $\alpha\mathbf{d}_k$ and $\alpha^{-1}\mathbf{z}_k$ gives the same result)
- **Sparsity constraint** for the activations $\mathbf{z}_k$, that improves the interpretability of the learned patterns (see Lecture 3)

Not convex with respect to the couple $(\mathbf{d}_k), (\mathbf{z}_k)$ but convex when the subproblems are taken individually

# Alternated resolution

Alternated resolution of two subproblems

## Dictionary learning

$$\mathbf{D}^* = \underset{\substack{\mathbf{D}=(\mathbf{d}_1,\dots,\mathbf{d}_K)\\ \forall k, \|\mathbf{d}_k\|_2^2 \leq 1}}{\operatorname{argmin}} \left\| \mathbf{x} - \sum_{k=1}^{K} \mathbf{z}_k * \mathbf{d}_k \right\|_2^2$$

## Convolutional sparse coding

$$\mathbf{Z}^* = \underset{\mathbf{Z}=(\mathbf{z}_1,\dots,\mathbf{z}_K)}{\operatorname{argmin}} \left\| \mathbf{x} - \sum_{k=1}^{K} \mathbf{z}_k * \mathbf{d}_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|\mathbf{z}_k\|_1$$

# Alternated resolution

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

- Both these problems can be solved with Proximal Gradient Descent algorithms
- Two main steps :
  1. Gradient descent step w.r.t. $\nabla_{\mathbf{D}} f(\mathbf{Z}, \mathbf{D})$ or $\nabla_{\mathbf{Z}} f(\mathbf{Z}, \mathbf{D})$
  2. Proximal step to *project* the update on the constraint set
- The main question is therefore: how can we easily compute the gradients?

# Formulation of the gradients

$$f(\mathbf{Z},\mathbf{D}) = \left\| \mathbf{x} - \sum_{k=1}^{K} \mathbf{z}_k * \mathbf{d}_k \right\|_2^2$$

- By using the convolution theorem and the Parseval theorem, we have that

$$f(\mathbf{Z},\mathbf{D}) = \left\| \hat{\mathbf{x}} - \sum_{k=1}^{K} \widehat{\mathbf{z}_k} \odot \widehat{\mathbf{d}_k} \right\|_2^2$$

  where $\hat{\cdot}$ denotes the Discrete Fourier Transform (DFT) and $\odot$ the component-wise product
- As $\mathbf{u} \odot \mathbf{d} = \mathrm{diag}(\mathbf{u})\mathbf{v}$, this expression can be rewritten in the matrix form as

$$f(\mathbf{Z},\mathbf{D}) = \left\| \hat{\mathbf{x}} - \widehat{\mathbf{D}}\hat{\mathbf{z}} \right\|_2^2$$

- The gradient of this quantity is easy to compute w.r.t. $\widehat{\mathbf{D}}$ and $\hat{\mathbf{z}}$
- Updates for $\mathbf{D}$ and $\mathbf{Z}$ can then be estimated by performing inverse DFT

# Proximal operators

- For the unit ball constraint (dictionary), the proximal operator writes as

$$\mathrm{proj}_{\|.\|_2^2 \leq 1}(\mathbf{y}) = \frac{\mathbf{y}}{\max(\|\mathbf{y}\|_2^2, 1)}$$

Projection on the unit ball

- For the L1-sparsity constraint (atoms), the proximal operator writes as

$$\mathcal{S}\gamma(\mathbf{y})[n] = \mathrm{sign}(y[n]) \times \max(|y[n]| - \gamma, 0)$$

**Soft thresholding operator**

# Dictionary learning

$$\mathbf{D}^* = \underset{\substack{\mathbf{D}=(\mathbf{d}_1,\ldots,\mathbf{d}_K) \\ \forall k, \|\mathbf{d}_k\|_2^2 \leq 1}}{\operatorname{argmin}} \underbrace{\left\| \mathbf{x} - \sum_{k=1}^{K} \mathbf{z}_k * \mathbf{d}_k \right\|_2^2}_{f(\mathbf{Z},\mathbf{D})}$$

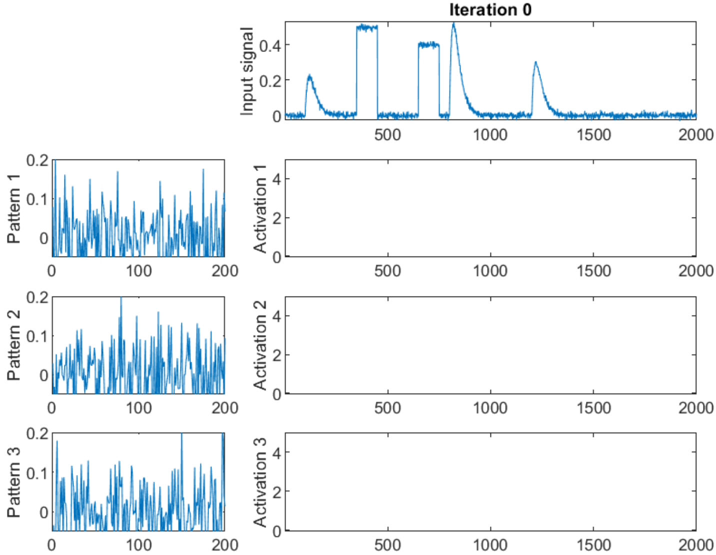
- Basic approach : **Proximal Gradient Descent** with fixed $\mathbf{Z}$
  1. Gradient step :
  $$\mathbf{D} \leftarrow \mathbf{D} - \alpha \nabla_{\mathbf{D}} f(\mathbf{Z}, \mathbf{D})$$
  2. Proximal projection step :
  $$\mathbf{d}_k \leftarrow \operatorname{proj}_{\|.\|_2^2 \leq 1}(\mathbf{d}_k) = \frac{\mathbf{d}_k}{\max(\|\mathbf{d}_k\|_2^2, 1)}$$
- Other approaches : Alternate Direction Method of Multiplier (ADMM), K-SVD... (see [Mairal et al., 2010] and Lecture 3 for more details)

# Convolutional sparse coding

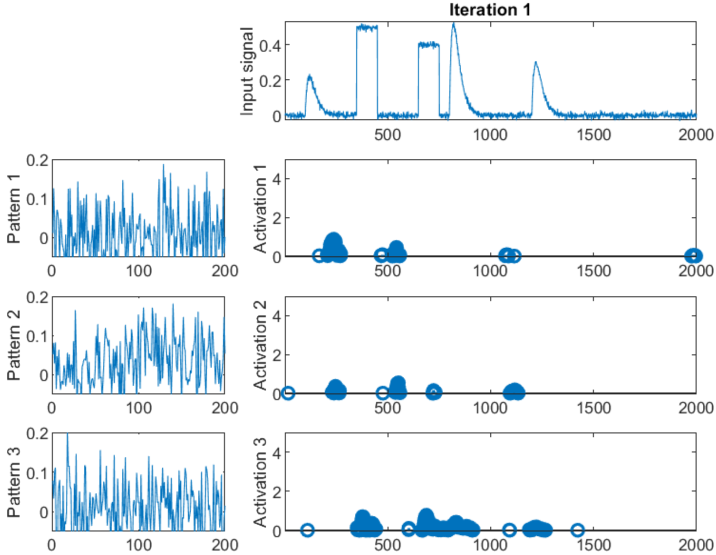
$$\mathbf{Z}^* = \underset{\mathbf{Z}=(\mathbf{z}_1,\ldots,\mathbf{z}_K)}{\operatorname{argmin}} \underbrace{\left\| \mathbf{x} - \sum_{k=1}^{K} \mathbf{z}_k * \mathbf{d}_k \right\|_2^2}_{f(\mathbf{Z},\mathbf{D})} + \underbrace{\lambda \sum_{k=1}^{K} \|\mathbf{z}_k\|_1}_{\psi(\mathbf{Z})}$$

- Basic approach : **Iterative Soft Thresholding Algorithm (ISTA)** with fixed **D** [Daubechies et al., 2004]
    1. Gradient step
    $$\mathbf{Z} \leftarrow \mathbf{Z} - \alpha \nabla_{\mathbf{Z}} f(\mathbf{Z}, \mathbf{D})$$
    2. Proximal step
    $$\mathbf{Z} = \mathcal{S}_{\lambda\alpha}(\mathbf{Z})$$
- Other approaches: Alternate Direction Method of Multiplier (ADMM), Fast Iterative Soft Thresholding Algorithm (FISTA), Coordinate Descent (CD) (see [Mairal et al., 2010] and Lecture 3 for more details)

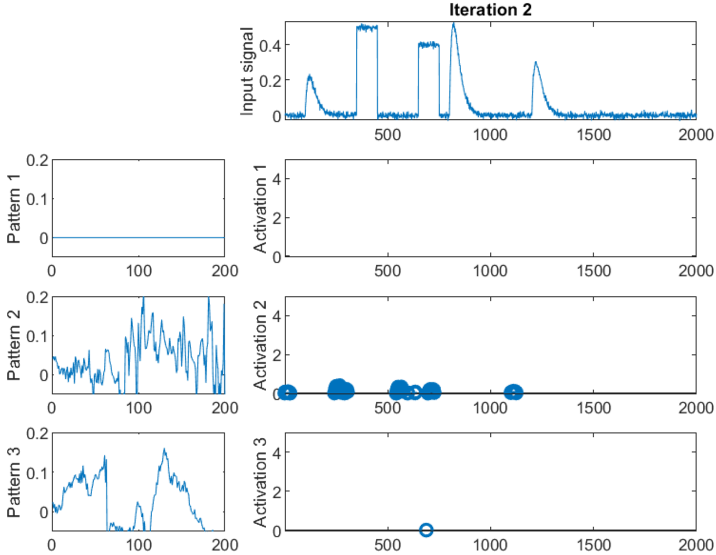
# Results


Iteration 0
Input signal
Pattern 1
Pattern 2
Pattern 3
Activation 1
Activation 2
Activation 3
0.4
0.2
0
500
1000
1500
2000
0.1
100
200
4
2

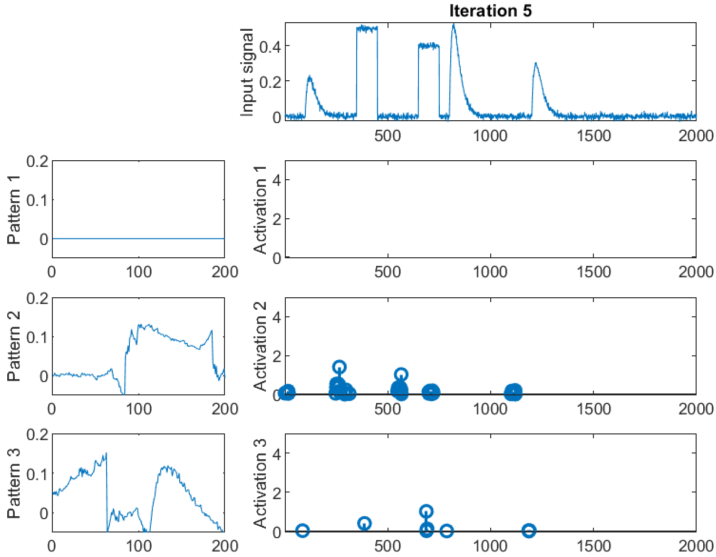
# Results


Iteration 1
Input signal
0.4
0.2
0
500
1000
1500
2000
Pattern 1
Activation 1
Pattern 2
Activation 2
Pattern 3
Activation 3
0.2
0.1
0
100
200
4
2

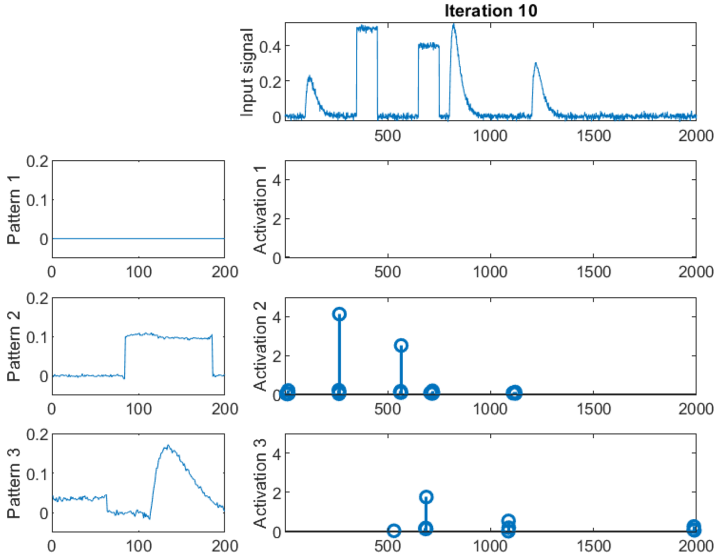
# Results


Iteration 2
Input signal
Pattern 1
Pattern 2
Pattern 3
Activation 1
Activation 2
Activation 3
0.4
0.2
0
500
1000
1500
2000
0.1
100
200
4
2

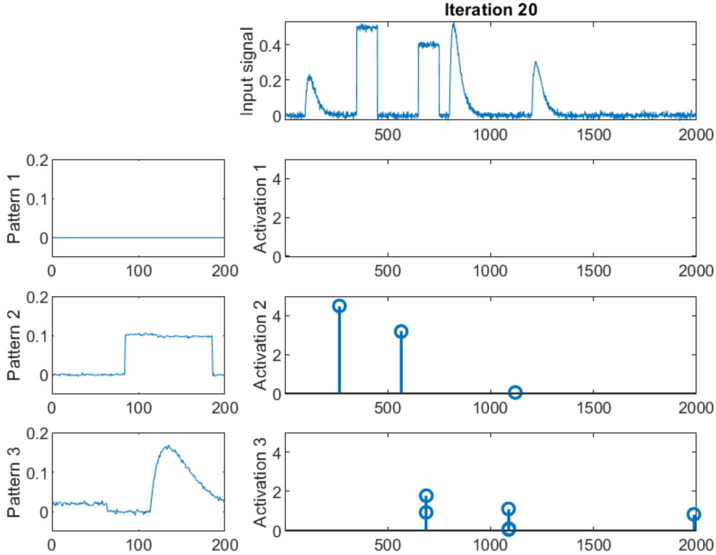
# Results


Iteration 5
Input signal
0.4
0.2
0
500
1000
1500
2000
Pattern 1
0.2
0.1
0
0
100
200
Activation 1
4
2
0
Pattern 2
Activation 2
Pattern 3
Activation 3

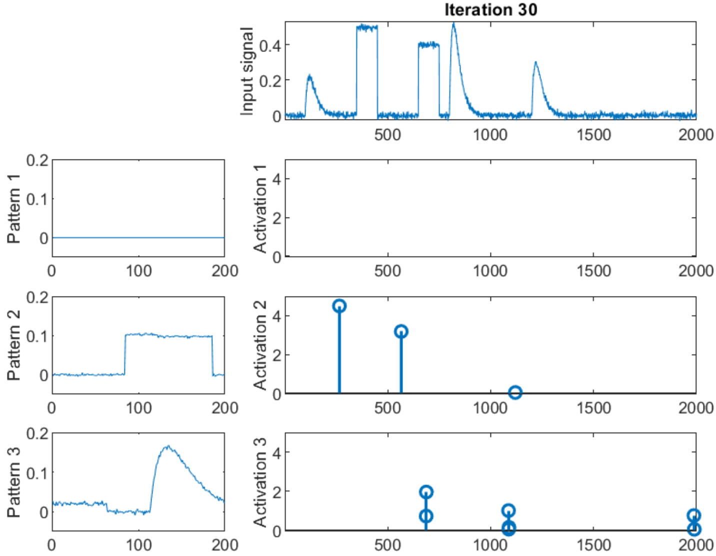
# Results


Iteration 10
Input signal
0.4
0.2
0
500
1000
1500
2000
Pattern 1
0.2
0.1
0
0
100
200
Activation 1
4
2
0
Pattern 2
Activation 2
Pattern 3
Activation 3

# Results

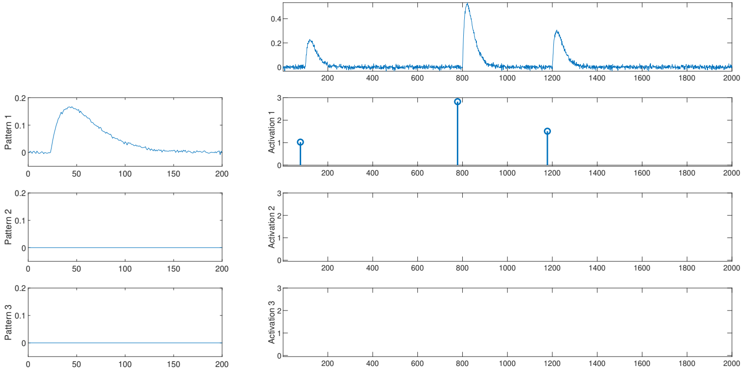

Iteration 20
Input signal
0.4
0.2
0
500
1000
1500
2000
Pattern 1
Activation 1
Pattern 2
Activation 2
Pattern 3
Activation 3
0.2
0.1
0
100
200
4
2

# Results



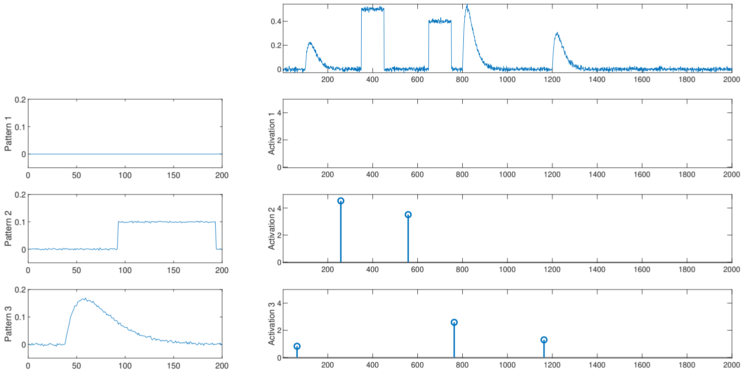
Iteration 30
Input signal
0.4
0.2
0
500
1000
1500
2000
Pattern 1
Activation 1
Pattern 2
Activation 2
Pattern 3
Activation 3
0.2
0.1
0
100
200
4
2

# Results



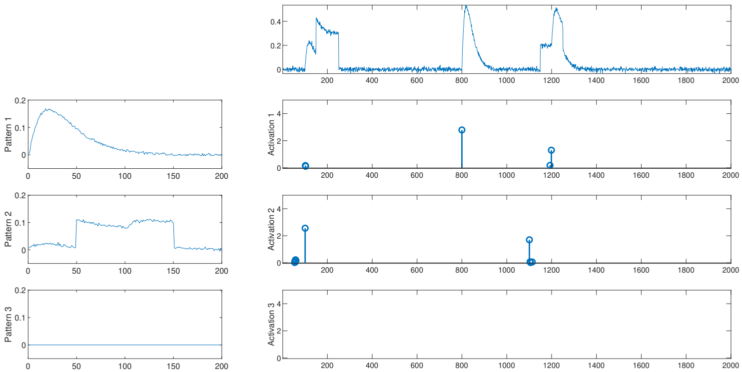

One pattern

# Results




Two patterns

# Results




Mixture of patterns

# Summary

- CDL allows to detect mixture of patterns thanks to the additive model
- CDL is traditionally quite sensitive to initialization and/or hyperparameters : random initializations (normalized gaussian noise), randomly chosen data frames, Lipschitz constant for gradient descent...
- CDL can also be used to search patterns from a fixed dictionary of templates : in this case, only perform the convolutional sparse coding step