Machine Learning for Time Series

Lecture 4: Data Enhancement and Preprocessings

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

Master MVA
2023-2024

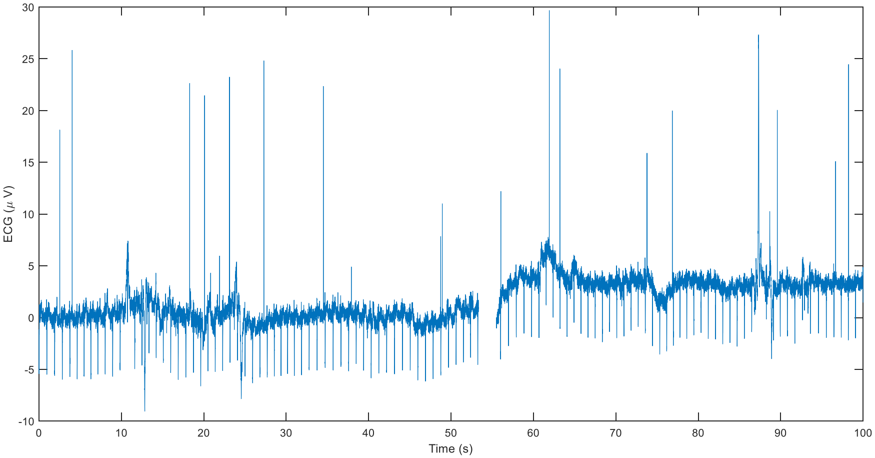
Contents

Contents

The need for preprocessing

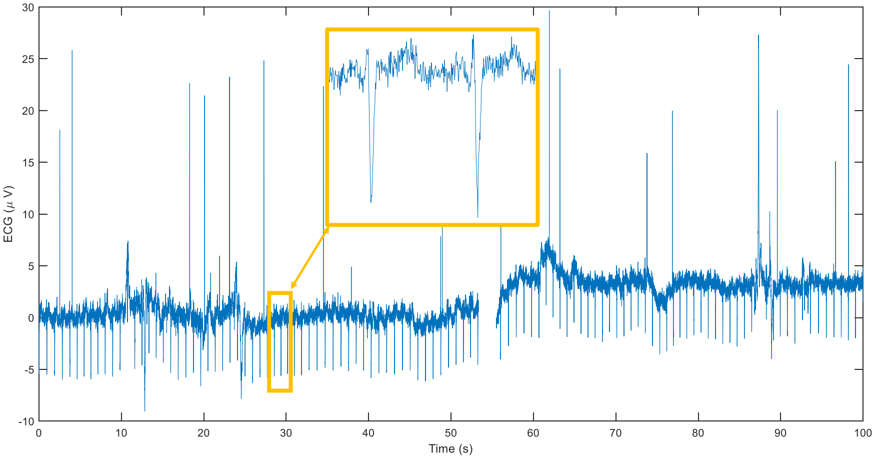
- Typical usecase: noisy time series with outliers and missing values
- In order to apply ML algorithms, the data scientist needs to *clean* and *consolidate* the data
- Time-consuming and tedious task: fortunately, ML also provides tools to that aim!
- Careful! All these preprocessing have a strong **impact** on the expected results and on the future learned rules!

Introductory example



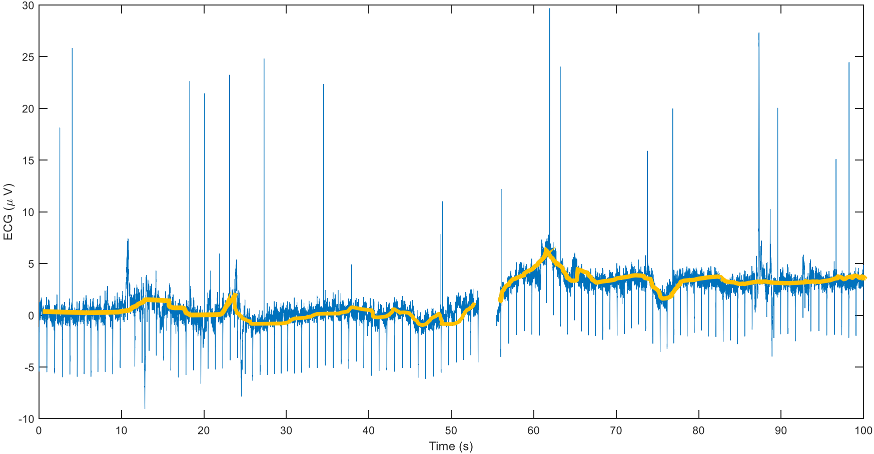
ECG signal during general anesthesia

Introductory example



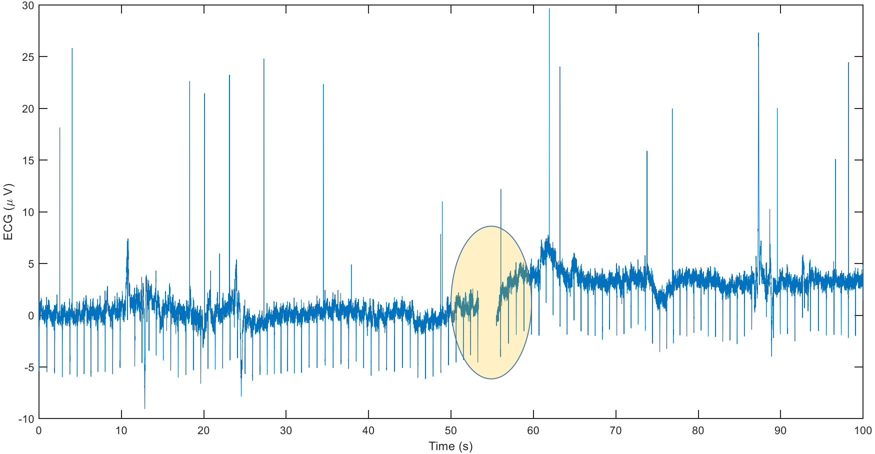
Presence of measurement noise → **Denoising**

Introductory example



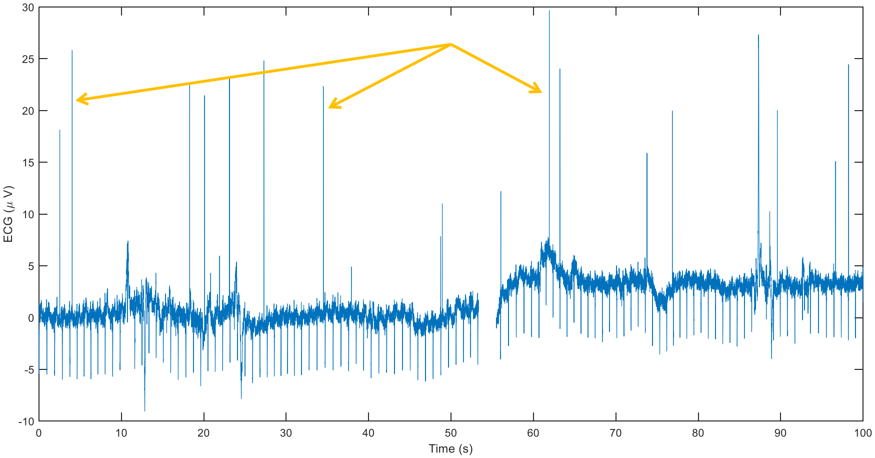
Presence of a trend → **Detrending**

Introductory example



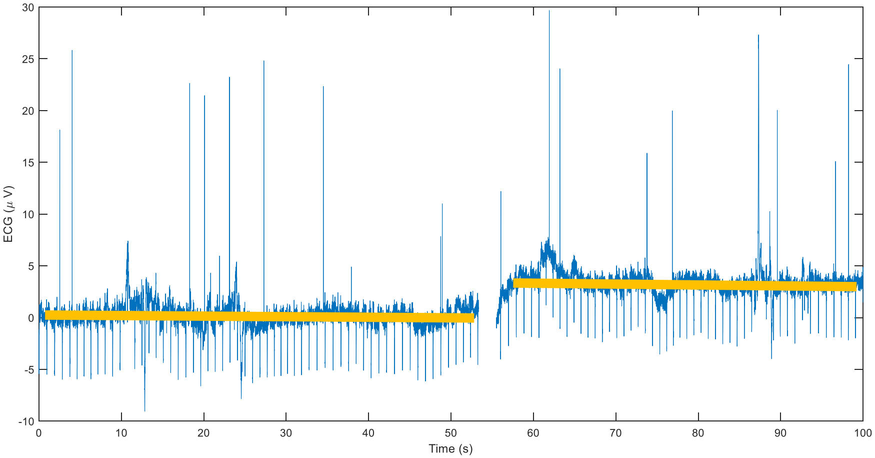
Data loss causing missing samples → **Interpolation**

Introductory example



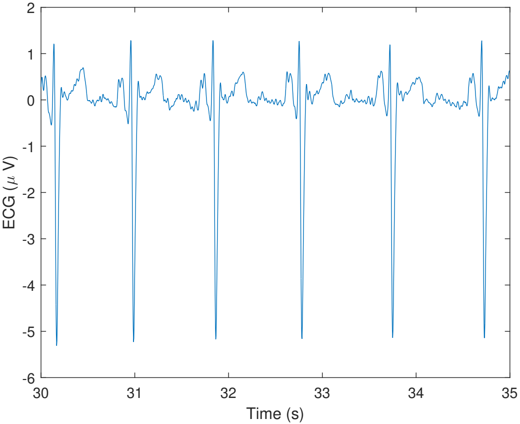
Presence of outliers → **Outlier removal and suppression of impulsive noise**

Introductory example



Break in stationarity → **Change-point detection** (see Lecture 5)

Introductory example



When all preprocessings have been performed, it becomes possible to retrieve the heartbeats and thus to perform ML

Contents

Additive white Gaussian noise (AWGN) model

The most common model for noisy signals is

$$\boxed{y[n] = x[n] + b[n]}$$

- $x[n]$ is the clean (unknown) signal
- $b[n]$ is the measurement noise, assumed to be additive, white and Gaussian (AWGN)
- $y[n]$ is the measured signal
- $x[n]$ and $b[n]$ are uncorrelated

Denoising

Given a noisy signal $y[n]$ corrupted by AWGN, retrieve the clean signal $x[n]$

Notion of AWGN

An AWGN $b[n]$ is:

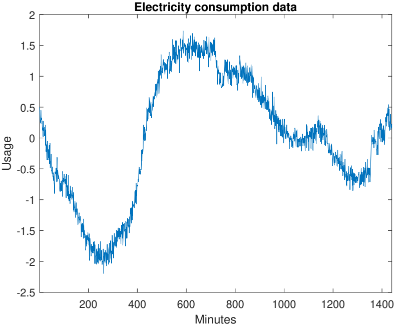
- **Additive**: the noise therefore corrupts all the samples
- **White**: stationary process with zero-mean and all samples are pairwise uncorrelated

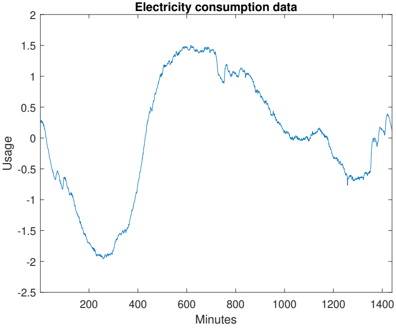
$$\gamma_b[m] = \begin{cases} \sigma^2 & m = 0 \\ 0 & \text{otherwise} \end{cases}$$

- **Gaussian**: all samples are i.i.d. according to

$$b[n] \sim \mathcal{N}(0, \sigma^2)$$

Example





How can we remove the noise component?

Filtering

- The first solution consists in using results from signal processing and statistics
- Knowing that $\gamma_x[m] = \mathbb{E}\left[x[n]x[n+m]\right]$ and using the fact that $x[n]$ and $b[n]$ are uncorrelated, we get that

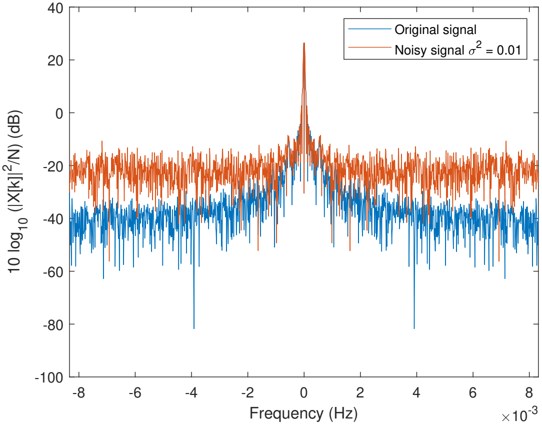
$$\gamma_y[m] = \gamma_x[m] + \gamma_b[m]$$

- By computing the DFT of this equation, we have

$$\boxed{|Y[k]|^2 = |X[k]|^2 + N\sigma^2}$$

- Adding AGWN is equivalent to adding a constant on the DFT of the signal (in linear scale)

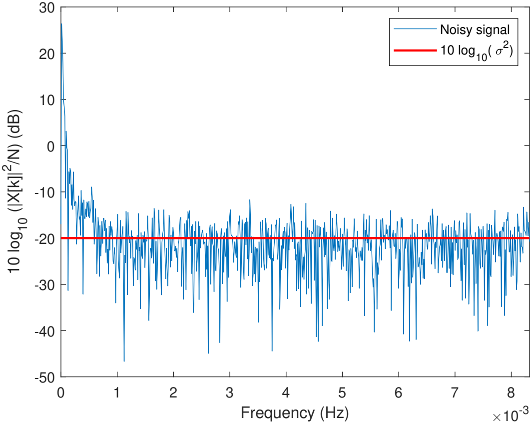
Example



In the frequency band where only AGWN is present (here with $\sigma^2 = 0.01$), the log-spectrum is equal to

$$10\log_{10}\left(\frac{|Y[k]|^2}{N}\right) = 10\log_{10}\left(\frac{|X[k]|^2}{N} + \sigma^2\right) = 10\log_{10}(0.01) = -20dB$$

Example



By plotting the log-spectrum of the noisy signal and knowing the noise variance σ^2, one can guess that all frequencies greater that e.g. 0.001 Hz are likely to only contain noise.

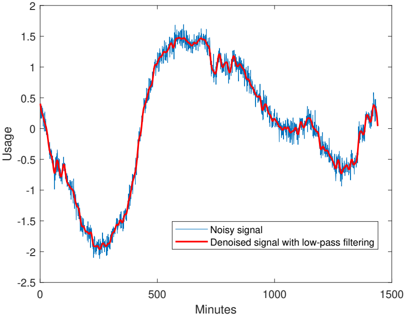
Filter design

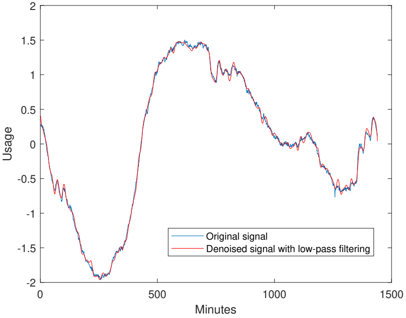
- By observing the log-spectrum of the noisy signal and using either prior knowledge on the original signal bandwidth or on the noise level, we can determine the type of filter and associated cut-off frequencies that can be used for denoising
- From that, it is only digital filter design (out of scope for this course !). Two popular solutions
  - **Moving average filter of length L:**

$$\hat{x}[n] = \frac{1}{L}\sum_{k=1}^{L-1} y[n-k]$$

    Low-pass filter with cut-off frequency $f_c \approx \frac{0.442947 \times F_s}{\sqrt{L^2-1}}$
  - **Butterworth filters:** can be low-pass, bandpass, etc...

Example





Low-pass filtering (Butterworth filter of order 4) with $f_c = 0.001$ Hz

Filtering vs. sparsity

- As such, filtering a signal consists in picking the frequencies that we want to keep
- Instead of designing a filter, we can attempt to retrieve a sparse frequency representation for the signal, which is equivalent to remove the small values on the spectrum
- Assumption:

$$\text{Large Fourier coefficients} \rightarrow \text{Signal}$$

$$\text{Small Fourier coefficients} \rightarrow \text{Noise}$$

- Principle of data compression: thresholding of small values in an appropriate representation space

$$\hat{\mathbf{x}} = \sum_{k \in \mathcal{K}} z_k \mathbf{d}_k, \quad \text{with } |\mathcal{K}| < N$$

Dictionaries

Several dictionaries can be used for denoising [Rubinstein et al., 2010]

- **Fourier dictionary** (also called Discrete Cosine Transform for real signals):

$$d_k[n] = \begin{cases} \frac{1}{\sqrt{N}} & k = 0 \\ \frac{2}{N} \cos\left(\frac{\pi}{N}\left(n + \frac{1}{2}\right)k\right) & 1 \leq k \leq N-1 \end{cases}$$

- **Wavelet dictionary** with wavelet function $\psi(t)$ and scaling function $\phi(t)$ [Percival et al., 2000 ; Mallat, 1999]

$$\phi_{m,l}[n] = 2^{-m/2}\phi(2^{-m}n - l) \qquad \psi_{m,k}[n] = 2^{-m/2}\psi(2^{-m}n - l)$$

  The dictionary is often computed up to level j_{max} :
  - $N \times 2^{-j}$ wavelets functions $\phi_{j,l}$ at level $1 \leq j \leq j_{max}$ with l multiple of 2^j: details
  - $N \times 2^{-j_{max}}$ scaling function $\phi_{j_{max},l}$ with l multiple of $2^{j_{max}}$: approximation
- Gabor dictionary, Modified Discrete Cosine Transform (MDCT)...

Sparse coding

Given an input dictionary $\mathbf{D}$, the denoising task is equivalent to a sparse coding task, and all previously seen algorithms can be used to that aim (see Lecture 3)

- $\ell 0$-based algorithms with hard thresholding

$$\boxed{\mathbf{z}^* = \underset{\substack{\mathbf{z} \\ \|\mathbf{z}\|_0 = K_0}}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2}$$

  Only keep the K_0 largest coefficients in the decomposition

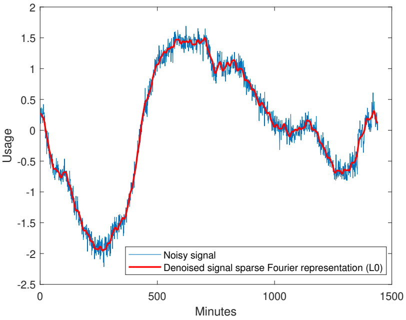
- $\ell 1$-based algorithms with soft thresholding

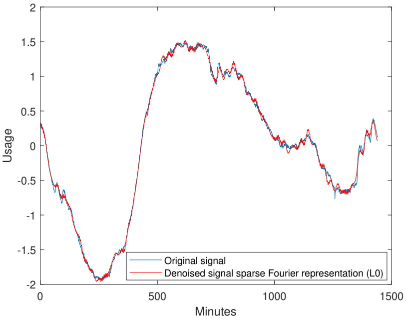
$$\boxed{\mathbf{z}^* = \underset{\mathbf{z}}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda \|\mathbf{z}\|_1}$$

  Set to zero the coefficients that are lower than a given threshold (and shrink the other ones)

$$\mathcal{S}_\lambda(\mathbf{z}) = \operatorname{sign}(\mathbf{z}) \times \max\left(|\mathbf{z}| - \lambda, 0\right)$$

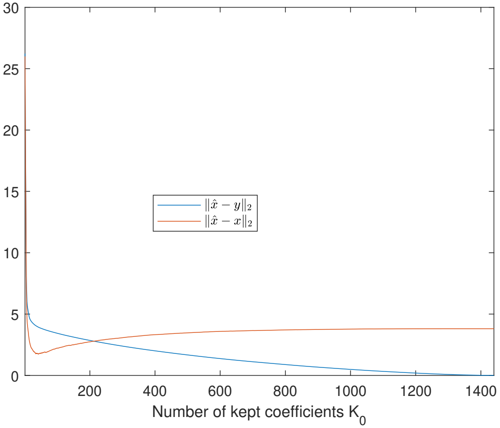
Example





Matching pursuit with Fourier dictionary and $K_0 = 40$

Example



Influence of the K_0 parameter on the denoising performances
Blue: distance to the noisy signal, red: distance to the clean signal

How to set K_0 or λ

- The parameters depend on the used dictionary (in particular on orthogonality properties of the atoms) and on the used algorithm
- Heuristics :
  - Use a training set
  - Use for λ a certain percentage of $\lambda_{max} = \|\mathbf{D}^t\mathbf{X}\|_\infty$
  - Empirical observation of the distribution of activations can also be used to choose the parameters (find an elbow on the curve)
  - Stochastic strategies: divide the over redundant dictionary into several small dictionary and average the decomposition made on these dictionaries
- Statistics (often require a probabilistic model for the data and/or noise) :
  - Some statistical results on estimators can be used to have an idea of the range of relevant parameters (Stein's Unbiased Estimate Risk Estimator (SURE), Minimax criteria) : see next slides for an example
  - Model selection strategies can also be used (see Lecture 5 and Tutorial session 3)

How to determine the stopping criterion?

- **Example for the Matching Pursuit algorithm.** For greedy denoising approaches (such as Matching Pursuit), a *good* denoising strategy would be to stop once the atoms in the dictionary have captured all relevant information on the signal, i.e. when the residual is composed of pure noise
- An interesting measure is the **normalized coherence** between the signal $\mathbf{x}$ and the dictionary $\mathbf{D}$

$$\boxed{\lambda_{\mathbf{D}}(\mathbf{x}) = \max_{\mathbf{d}\in\mathbf{D}} \frac{|\langle \mathbf{x}, \mathbf{d} \rangle|}{\|\mathbf{x}\|_2}}$$

Stopping criteria for matching pursuit

- By denoting $\mathbf{r}^{(\ell)}$ the residual at iteration ℓ,

$$\mathbf{r}^{(\ell)} = \mathbf{r}^{(\ell-1)} - \langle \mathbf{r}^{(\ell-1)}, \mathbf{d}^* \rangle \mathbf{d}^*$$

  where $\mathbf{d}^*$ is the atom most correlated to $\mathbf{r}^{(\ell-1)}$
- Basic calculations give that

$$\frac{\|\mathbf{r}^{(\ell)}\|_2^2}{\|\mathbf{r}^{(\ell-1)}\|_2^2} = 1 - \lambda_{\mathbf{D}}^2(\mathbf{r}^{(\ell-1)})$$

- The decreasing of the L2 norm of the residual is therefore linked to the normalized coherence of the residual with the dictionary
  - If $\lambda_{\mathbf{D}}(\mathbf{r}^{(\ell-1)})$ is large, it is worth continuing
  - If $\lambda_{\mathbf{D}}(\mathbf{r}^{(\ell-1)})$ becomes too small, the algorithm can stop
- When can we say that the coherence becomes *too low*?

Stopping criteria for matching pursuit

- One interesting question is therefore: what is the value of $\lambda_{\mathbf{D}}(\mathbf{r})$ when the residual $\mathbf{r}$ is pure noise ?
- If $\mathbf{r}$ is pure random with a known distribution $p(\mathbf{r})$ (e.g. AGWN), we can be interested in the quantity

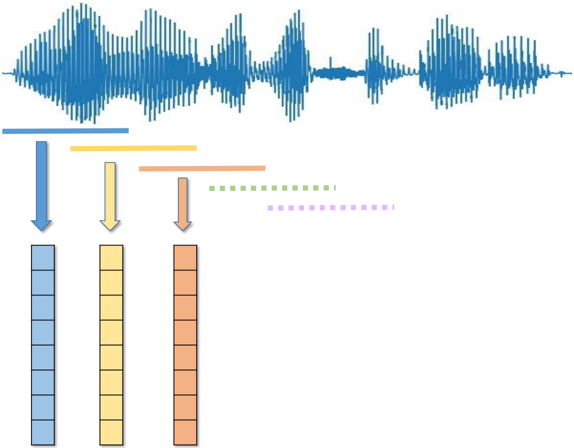
$$\lambda_{p(\mathbf{r})}(\mathbf{D}) = \mathbb{E}_{p(\mathbf{r})}\left[\lambda_{\mathbf{D}}(\mathbf{r})\right]$$

- Intuitively, denoising for $\mathbf{x}$ can then be achieved by stopping when the norrmalized coherence of the residual has the same order of magnitude as this value
- How to compute $\lambda_{p(\mathbf{r})}(\mathbf{D})$?
  - Use a training set of noise signals
  - Use statistical considerations with parametrized distribution (see mini-project)

Use of adaptive dictionaries

- Instead of using off-the-shelf dictionaries, we can learn the representation directly from the signal: **dictionary learning** (see Lecture 3)
- Use of the **trajectory matrix $\mathbf{X}$**: matrix representation of the input signal frames
- Noise is random: when sparsity is enforced, the approximation tends to only model signal

Trajectory matrix



N_w: window length, N_o: overlap length

N_w rows, $N_f = \lfloor \frac{N-N_w}{N_w-N_o} \rfloor + 1$ columns

Dictionary learning

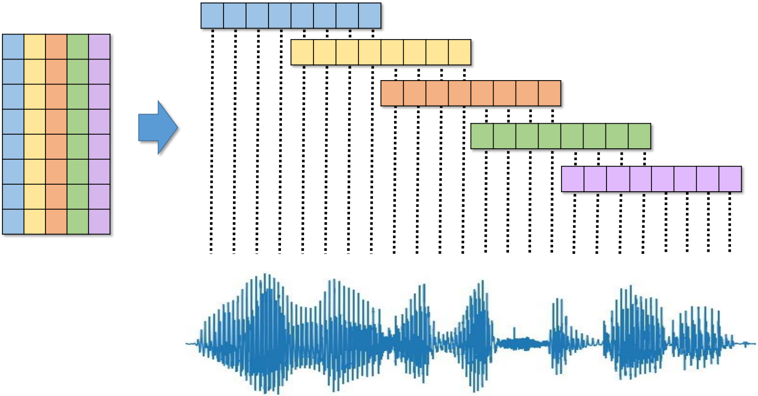
With algorithms already described in Lecture 3, compute an approximation of the trajectory matrix

$$\boxed{\hat{\mathbf{X}} = \mathbf{DZ} \approx \mathbf{X}}$$

- $\mathbf{D} \in \mathbb{R}^{N_w \times K}$: dictionary composed of K atoms
- $\mathbf{Z} \in \mathbb{R}^{K \times N_f}$: sparse activations (sparsity level specified with K_0 or λ)

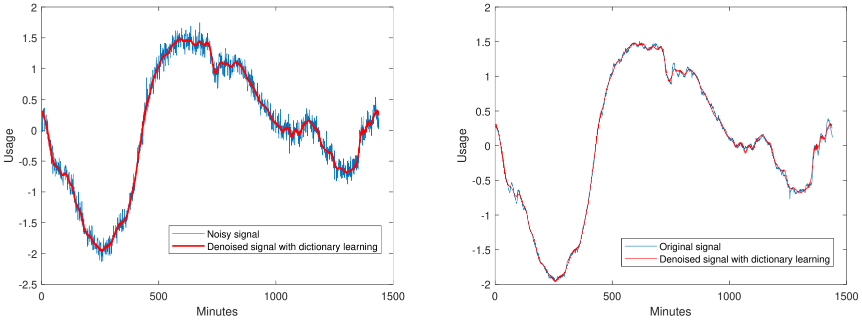
Each frame is approximated as a sparse linear combination of the learned atoms

Reconstruction from the approximated trajectory matrix



Unfolding of the matrix and averaging along overlapping frames

Example



Dictionary learning with $K = 5$, $K_0 = 2$, $N_w = 32$, $N_o = 28$

Trajectory matrix

- When $N_o = N_w - 1$, we have $N_f = N - N_w + 1$ and the trajectory matrix $\mathbf{X} \in \mathbb{R}^{N_w \times N_f}$ has a particular form

$$\mathbf{X} = \begin{pmatrix} x[0] & \cdots & x[N - N_w - 1] \\ x[1] & \cdots & x[N - N_w] \\ \vdots & \ddots & \vdots \\ x[N_w - 1] & \cdots & x[N - 1] \end{pmatrix}$$

- It contains all N_f sequences of length N_w in the time series
- Low-rank approximations attempt to reconstruct matrix $\mathbf{X}$ as the sum of $K < \min(N_w, N_f)$ rank-one matrices

Singular Value Decomposition

Assuming that $N_w < N_f$, the Singular Value Decomposition (SVD) of matrix $\mathbf{X}$ writes:

$$\mathbf{X} = \underbrace{\mathbf{U}}_{N_w \times N_w} \underbrace{\mathbf{\Lambda}}_{N_w \times N_f} \underbrace{\mathbf{V}^t}_{N_f \times N_f}$$

where

- $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices
- $\mathbf{\Lambda}$ is a diagonal matrix containing on its first diagonal at most N_w singular values $\lambda_1 \geq \ldots \geq \lambda_{N_w}$

$$\boxed{\mathbf{X} = \sum_{k=1}^{N_w} \lambda_k \mathbf{u}_k \mathbf{v}_k^t}$$

Interpretation of the singular values

- For a zero-mean stationary signal, the lag-covariance matrix for lag N_w can be estimated as:

$$\mathbf{C}_{\mathbf{X}} = \frac{1}{N_f}\mathbf{X}\mathbf{X}^t$$

- Definite positive matrix with eigen decomposition

$$\mathbf{C}_{\mathbf{X}} = \tilde{\mathbf{V}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{V}}^t$$

  $\tilde{\lambda}_k$ corresponds to the contribution of the direction given by k^{th} eigenvector to the global variance (see Lecture 2 on Principal Component Analysis (PCA))

- Basic computations give that

$$\lambda_k \propto \sqrt{\tilde{\lambda}_k}$$

  which provides a natural interpretation of the singular values of the trajectory matrix

Singular Spectrum Analysis (SSA)

This principle is the core of the **Singular Spectrum Analysis (SSA)** algorithm [Vautard et al., 1992]:

1. Compute the SVD of the trajectory matrix

$$\mathbf{X} = \sum_{k=1}^{N_w} \lambda_k \mathbf{u}_k \mathbf{v}_k^t$$

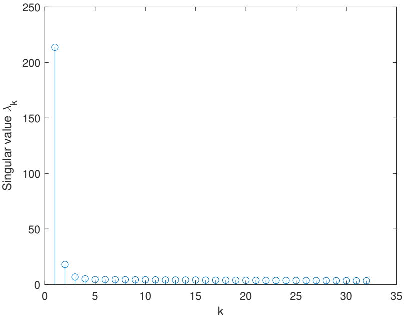
2. By analyzing the singular value distribution, form groups $\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_M$ of singular values corresponding to similar phenomenon

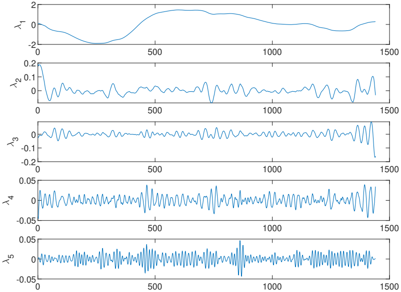
$$\mathbf{X} \approx \sum_{k \in \mathcal{K}_1} \lambda_k \mathbf{u}_k \mathbf{v}_k^t + \ldots + \sum_{k \in \mathcal{K}_M} \lambda_k \mathbf{u}_k \mathbf{v}_k^t$$

Using SSA for denoising

- Intuitively, for reasonable signal-to-noise ratio, signal should be dominant and thus corresponds to the largest singular values
- By plotting the singular values λ_k as a function of k, it is possible to detect and group the different phenomenon within the time series
- For denoising, it is common to remove all components corresponding to small singular values
- Choice of N_w (only parameter): longest periodicity captured by SSA

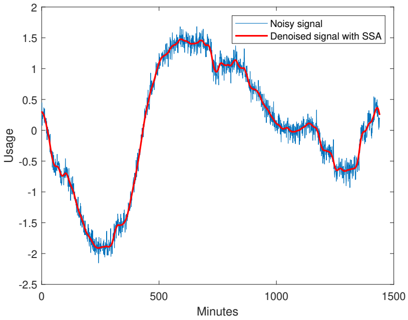
Example

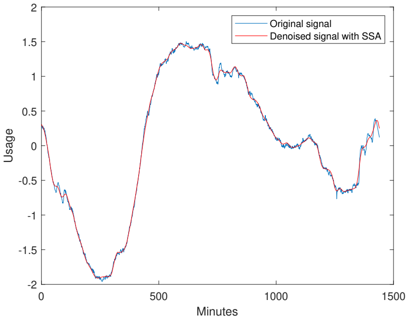




Singular values and reconstructed components with $N_w = 32$. From the graphs it appears that the two first singular values are likely to be signal

Example





Denoising with SSA with $N_w = 32$ and using only the two first components.

Other techniques

Several other decomposition techniques can be used:

- **Independent Component Analysis (ICA)**: decompose the signal into the sum of statistically independent components [Comon, 1994]
  - Useful for blind source separation and unmixing (e.g. in EEG data or audio)
  - Algorithms based on the optimization of several measures of independence (mutual information, gaussianity etc.)
- **Empirical Mode Decomposition (EMD)**: decompose the signal into the sum of oscillary modes with various amplitude and frequency [Flandrin et al., 2004 ; Boudraa et al., 2006]
  - Useful for denoising but also detrending
  - Algorithms based on the iterative modeling of the signal as splines

Contents

Trend+Seasonality model

The trend+seasonality model writes as

$$x[n] = \underbrace{\alpha_1\beta_1(nT_s) + \ldots + \alpha_j\beta_j(nT_s)}_{x^{trend}[n]} + \underbrace{\alpha_{j+1}\beta_{j+1}(nT_s) + \ldots + \alpha_d\beta_d(nT_s)}_{x^{seasonality}[n]} + b[n]$$

- Seasonality: pseudo-periodic component
- Trend: smooth variations, systematic increase or decrease in the data

Detrending

Given a signal $x[n]$, estimate and remove the trend component $x^{trend}[n]$

Standard models

The most common trend models are:

- Constant trend

$$x^{trend}[n] = \alpha_0$$

- Linear trend

$$x^{trend}[n] = \alpha_1 (nT_s) + \alpha_0$$

- Polynomial trend

$$x^{trend}[n] = \sum_{k=0}^{K} \alpha_k (nT_s)^k$$

Least-square regression

- Least-square estimator: minimization of

$$\|\mathbf{x} - \boldsymbol{\beta}\boldsymbol{\alpha}\|_2$$

where

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0(0) & \cdots & \beta_K(0) \\ \beta_0(T_s) & \cdots & \beta_K(T_s) \\ \vdots & \ddots & \vdots \\ \beta_0((N-1)T_s) & \cdots & \beta_K((N-1)T_s) \end{pmatrix}$$

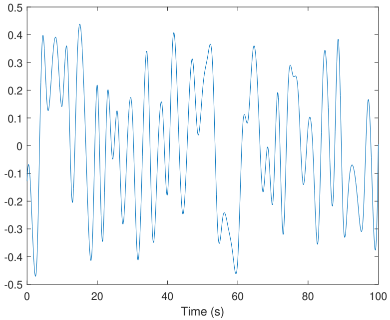
- Closed form solution

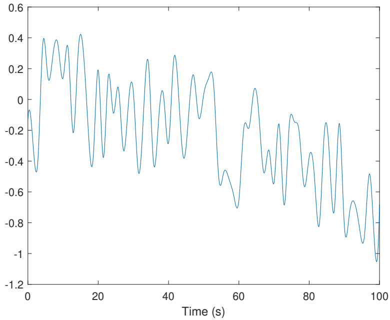
$$\boxed{\hat{\boldsymbol{\alpha}} = \left(\boldsymbol{\beta}^T\boldsymbol{\beta}\right)^{-1}\boldsymbol{\beta}^T\mathbf{x}}$$

- Estimation of the trend

$$\boxed{\mathbf{x}^{\textbf{trend}} = \boldsymbol{\beta}\hat{\boldsymbol{\alpha}}}$$

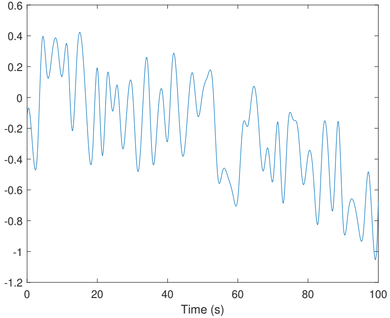
Example

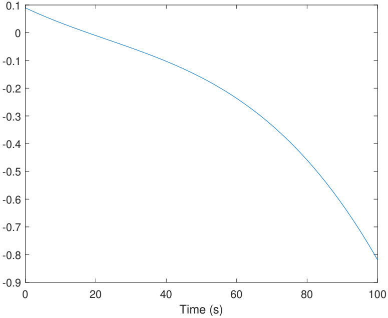




Signal with/without trend

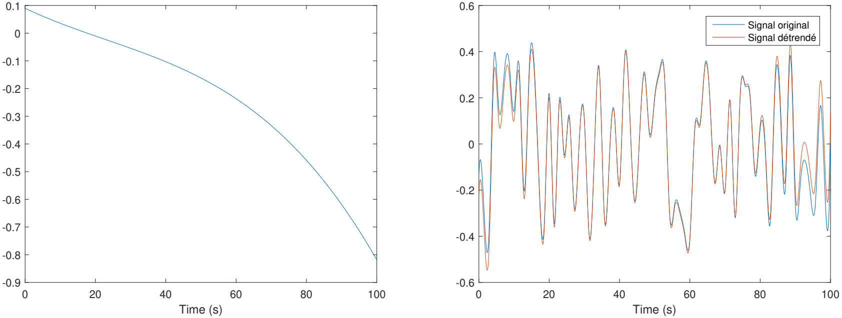
Example





Regression on polynomials of order 3

Example



Regression on polynomials of order 3

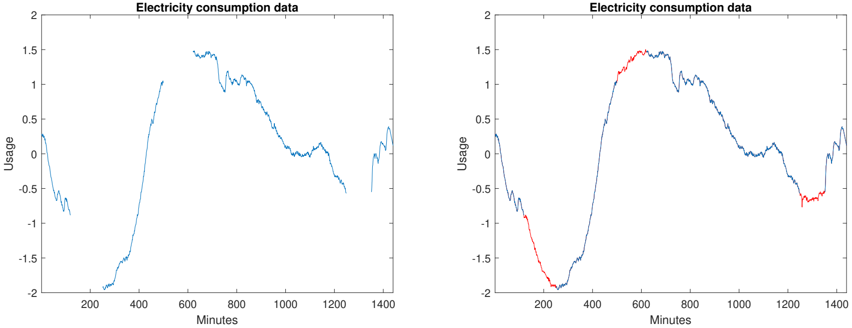
Other approaches

Other approaches for detrending include

- **Filtering techniques**, as trends often correspond to low frequencies or smooth components (low-pass/bandpass filters, Fourier or wavelets thresholding...)
- **Decomposition techniques**, as trends may be considered independent of the seasonality and/or the noise component (EMD, SSA, ICA...)

Contents

Interpolation of missing samples



Interpolation of missing samples

Given a signal x and a set of missing samples $\mathcal{T}$, estimate the missing samples $\hat{\mathbf{x}}_{\mathcal{T}}$

Interpolation of missing samples

- Missing data are very frequent :
  - Sensor malfunctions
  - Clipping effect
  - Corrupted samples
- Missing data can take several forms
  - Isolated samples: easy to handle
  - Contiguous samples (up to 100): necessitates a full reconstruction
- Interpolation includes prediction and inpainting [Lepot et al., 2017]

Polynomial interpolation

Given a time series **x** that we want to interpolate on the integer set $\mathcal{T} = [\![n_{start}, n_{end}]\!]$, the easiest interpolation strategy consists in using polynomial models for the reconstruction

- **Constant value**

$$\forall n \in \mathcal{T}, \quad \boxed{\hat{x}[n] = \frac{x[n_{start} - 1] + x[n_{end} + 1]}{2}}$$

- **Linear interpolation**

$$\forall n \in \mathcal{T}, \quad \boxed{\hat{x}[n] = \beta_1 n + \beta_0}$$

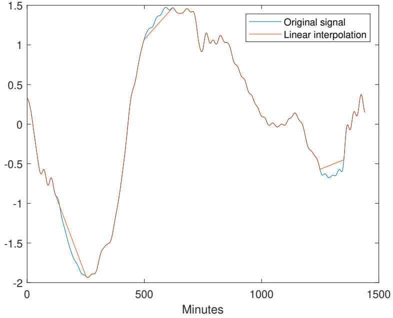
  where β_0, β_1 are determined with the values $x[n_{start} - 1]$ and $x[n_{end} + 1]$

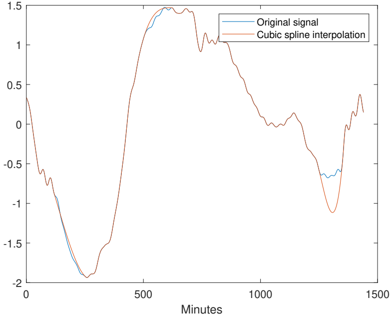
- **Cubic spline interpolation** [McKinley et al., 1998]

$$\forall n \in \mathcal{T}, \quad \boxed{\hat{x}[n] = \beta_3 n^3 + \beta_2 n^2 + \beta_1 n + \beta_0}$$

  where β_k are determined by solving a system of equations based on $x[n_{start} - 2]$, $x[n_{start} - 1]$, $x[n_{end} + 1]$ and $x[n_{end} + 2]$

Example


Original signal
Linear interpolation
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
500
1000
1500
Minutes


Original signal
Cubic spline interpolation
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
500
1000
1500
Minutes

Pros and cons

- Easy to implement and good results for small segments
- In particular, when only a few missing samples: constant values is often the best
- When the degree of the polynomial increases, instabilities may occur (strong dependency with the neighborhood samples)
- When used extensively, may lead to a smoothing of the signal hence a change in the spectrum (boosting of the low frequencies)

Low-rank interpolation

- The low-rank assumption on the trajectory matrix can also be used for reconstructing missing samples

$$\mathbf{X} = \begin{pmatrix} x[0] & \cdots & x[N-N_w-1] \\ x[1] & \cdots & x[N-N_w] \\ \vdots & \ddots & \vdots \\ x[N_w-1] & \cdots & x[N-1] \end{pmatrix}$$

- In this case, we will use the Singular Value Decomposition adapted to data with missing values [Srebro et al., 2003]
- These techniques are efficient for medium-size missing patches, as the low-rank assumption is usually only valid for relatively small windows

Principle

- The main idea is to compute a low-rank approximation of the trajectory matrix $\mathbf{X} \in \mathbb{R}^{N_w \times N_f}$, where only the **largest** singular values are kept

$$\boxed{\hat{\mathbf{X}} = \sum_{k=1}^{K} \lambda_k \mathbf{u}_k \mathbf{v}_k^t}$$

  where $k < \min(N_w, N_f)$

- But how do we compute the SVD for a matrix that contains missing values?
- **Mask matrix:**

$$W_{i,j} = \begin{cases} 0 & \text{if } X_{i,j} \text{ is missing} \\ 1 & \text{else} \end{cases}$$

- Low-rank approximation will only be used to update the missing samples

Low-rank interpolation

Algorithm 1: Low-rank interpolation

Input : Trajectory matrix $\mathbf{X}$ with missing values
Mask matrix $\mathbf{W}$
Expected rank K

Output: Interpolated trajectory matrix $\hat{\mathbf{X}}$

Initialize $\hat{\mathbf{X}}$;

while $n_{iter} < n_{max}$ **do**

SVD computation;

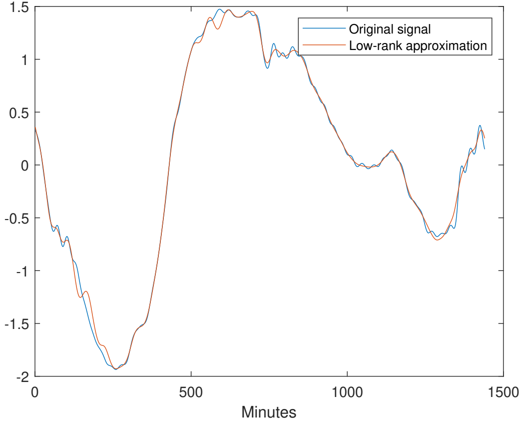
$[\mathbf{U}, \mathbf{\Lambda}, \mathbf{V}] = \mathrm{SVD}\left(\mathbf{X} \odot \mathbf{W} + \hat{\mathbf{X}} \odot (1 - \mathbf{W})\right)$;

Low-rank approximation;

$\hat{\mathbf{X}} = \sum_{k=1}^{K} \lambda_k \mathbf{u}_k \mathbf{v}_k^t$

end

Example



$$N_w = 300,\ K = 10$$

Pros and cons

- Good results for medium size segments
- Always set N_w greater than the largest missing patch: when N_w becomes too large, the low-rank approximation becomes less valid
- Variant with adaptive rank can provide better results [Srebro et al., 2003]

$$K_{n_{iter}} = \max(\min(N_w, N_f) - n_{iter}, K)$$

- Rank K can be estimated by computing the SVD of the initialized trajectory matrix (with polynomial reconstruction for instance)

Model-based interpolation

- For long segments of missing samples, interpolation becomes a full reconstruction task
- In this case a model is necessary to obtain a satisfactory interpolation
  1. Choice of an adequate model
  2. Parameter inference from the known samples
  3. Replacement of the missing samples by values in adequacy with the learned model

Model-based interpolation

- Problem: how do we estimate the parameters from a time series with missing data?
- Iterative solution
    1. Initialization of the missing samples with simple rough estimates (set to zero, constant or linear interpolation...)
    2. Parameter inference from all samples
    3. Reconstruction of the missing samples from the learned model
    4. Repeat steps 2 and 3 until convergence

AR-based interpolation

- For an $AR(p)$ model, given estimates of parameters $\hat{\mathbf{a}}$, the signal can be reconstructed by assuming that

$$x[n] \approx -\sum_{i=1}^{p} \hat{a}_i x[n-i]$$

- The prediction error on the whole time series writes

$$E(\mathbf{x}) = \sum_{n=p}^{N-1} \left| x[n] + \sum_{i=1}^{p} \hat{a}_i x[n-i] \right|^2$$

- The main idea is to minimize this quantity in order to retrieve appropriate values for the missing samples [Janssen et al., 1986]

AR-based interpolation

$$\mathbf{x}^* = \underset{\forall n \notin \mathcal{T}, \tilde{x}[n]=x[n]}{\text{argmin}} \quad E(\tilde{\mathbf{x}})$$

- This optimization problem has a closed form solution (least-square estimates) that is obtained by rewritting $E(\mathbf{x})$ as the sum of terms depending on the missing samples $n \in \mathcal{T}$ and other depending only on the known samples.
- By denoting $\mathbf{x}_\mathcal{T}$ the set of missing samples, the equation rewrites

$$E(\mathbf{x}) = \mathbf{x}_\mathcal{T}{}^T \mathbf{B} \mathbf{x}_\mathcal{T} + 2\mathbf{x}_\mathcal{T}\mathbf{d} + C$$

  where

  - $\forall (t, t') \in \mathcal{T}, \;\; b_{t,t'} = \begin{cases} \sum_{l=0}^{p-|t-t'|} \hat{a}_l \hat{a}_{l+|t-t'|} & \text{if } 0 \leq |t-t'| \leq p \\ 0 & \text{else} \end{cases}$
  - $\forall (t, t') \in \mathcal{T}, \;\; d_t = \sum_{\substack{-p \leq k \leq p \\ t-k \notin \mathcal{T}}} b_{|k|} x[t-k]$
  - C is a constant only depending on the known samples
- The final problem is simply a linear system and thus easy to solve

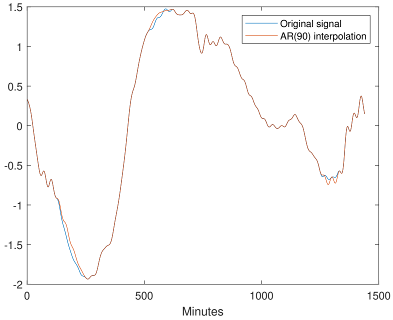
$$\mathbf{B}\mathbf{x}_\mathcal{T} = -\mathbf{d}$$

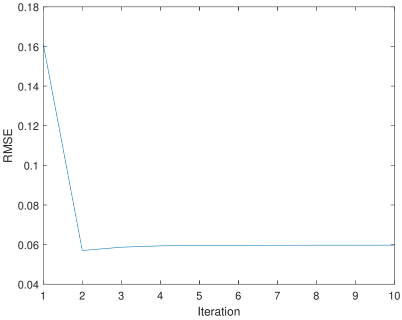
AR-based interpolation

Algorithm 2: AR-based interpolation

Inputs : Time series $\mathbf{x} \in \mathbb{R}^N$ with missing values
Set of missing samples $\mathcal{T}$
AR model order p
Output: Interpolated samples $\hat{\mathbf{x}}_{\mathcal{T}}$
$\hat{\mathbf{x}}_{\mathcal{T}} = \mathbf{0}_{|\mathcal{T}|}$;
while $n_{iter} < n_{max}$ **do**
    AR estimation step;
    Estimate $\hat{\mathbf{a}}$ with the Levinson Durbin algorithm;
    AR interpolation step;
    Compute $\mathbf{B}$ and $\mathbf{d}$ and solve for $\hat{\mathbf{x}}_{\mathcal{T}}$;
    Set $\mathbf{x}_{\mathcal{T}} = \hat{\mathbf{x}}_{\mathcal{T}}$;
end

Example

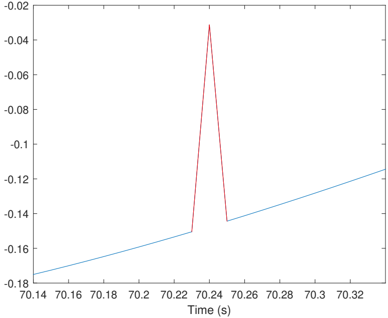


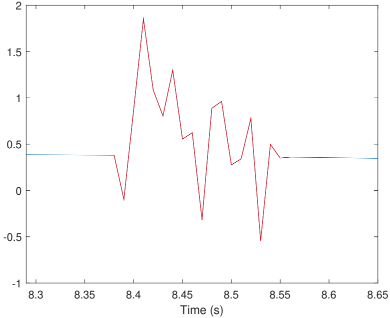


Interpolation with $AR(90)$ model

Contents

Outlier removal





Outliers, also called impulsive noise (as opposed to AWGN) correspond to spurious samples (isolated or continuous) that take unlikely values

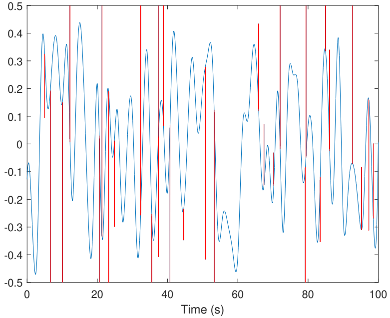
Outlier removal

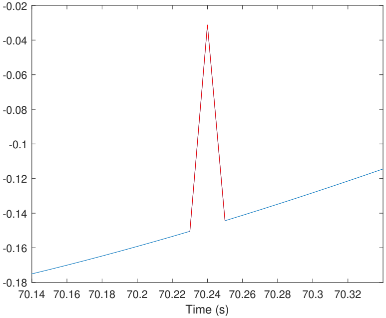
Outlier removal

Given a signal $x[n]$, outlier removal consists in detecting the locations $\mathcal{T}$ of the outliers (detection phase) and to replace these values with more adequate values (interpolation phase)

- Interpolation phase can be done by using the previously described algorithms. We will therefore focus on the detection phase.
- Two settings: isolated samples or contiguous group of samples
- Outliers are not only characterized by their values but also on their positions in the time series: context is fundamental

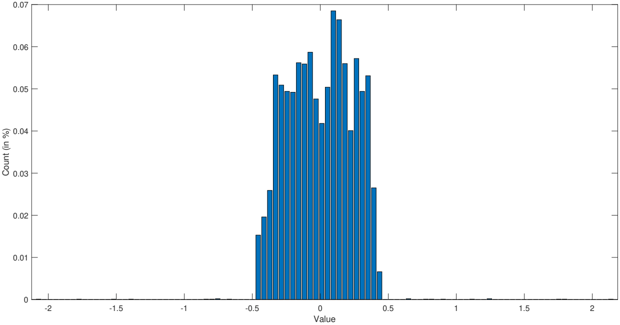
Isolated samples





Impulsive noise that only corrupts isolated samples

Histogram



If the values taken by the impulsive noise are particularly large with respect to the signal, they can be detected by looking at the histogram of the values taken by the samples: similar to outlier detection in statistical data

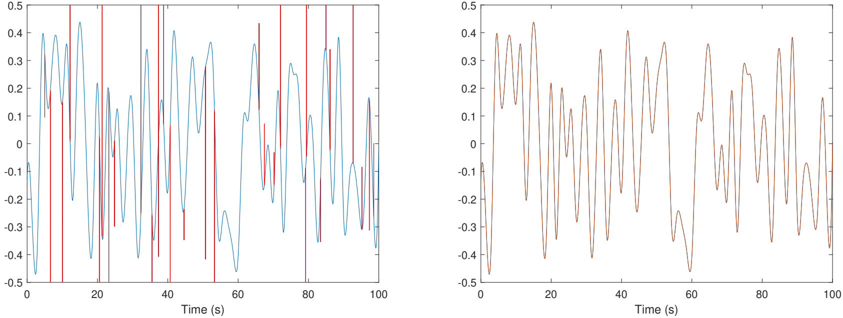
Median filtering

- Outliers can be detected AND removed by using a sliding median filtering that replaces each value by the median of the samples in a window of length $2w+1$:

$$\boxed{\hat{x}[n] = \text{median}_{-w \leq i \leq +w} \left\{ x[n-i] \right\}}$$

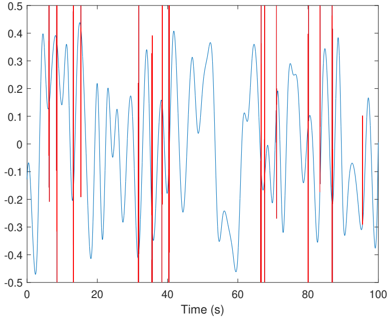
- Median filtering allows to *smooth* the time series while preserving the discontinuities
- Example : original signal $[0.3 \;\; 0.4 \;\; 0.45]$ and noisy signal $[0.3 \;\; 0.9 \;\; 0.45]$
  - Moving average filter: $0.9 \rightarrow 0.55$
  - Median filter: $0.9 \rightarrow 0.375$

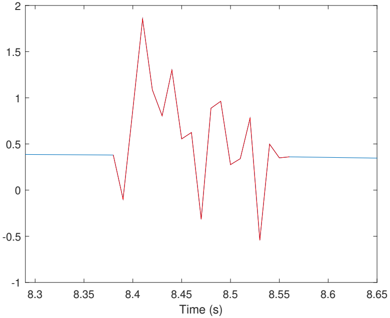
Median filtering



Perfect reconstruction with median filtering ($2w + 1 = 3$ samples)

Contiguous samples





Impulsive noise that corrupts groups of contiguous samples

Contiguous samples

- When the impulsive noise corrupts groups of contiguous samples, studying the values is not sufficient
- In order to retrieve the set of outliers $\mathcal{T}$, using a model may be necessary
- Outliers: samples that are *far* from their predicted values according to a model
- Same principle that model-based interpolation: parameter estimation, detection, interpolation and reiterate
- Note: this task is close to the Anomaly Detection task (see Lecture 5)

AR-based outlier detection

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

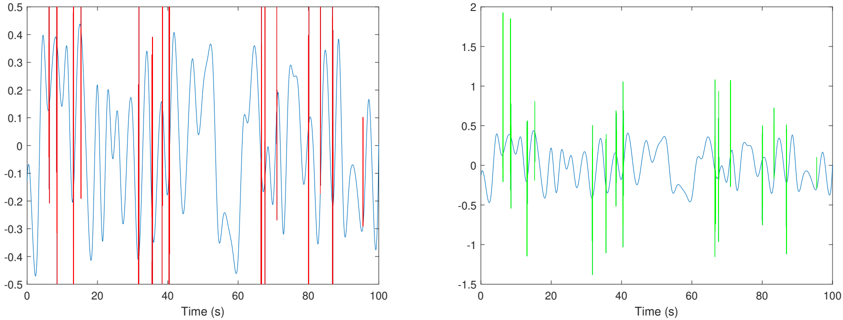
- Given estimates of the AR parameters $\hat{\mathbf{a}}$, the prediction error writes:

$$e[n] = x[n] + \sum_{i=1}^{p} \hat{a}_i x[n-i]$$

- If adapted model, good parameter estimation and low noise variance, this quantity must be rather small for samples that are not outliers [Oudre, 2015]
- Detection method with threshold λ :

$$\mathcal{T} = \{n \text{ s.t. } |e[n]| > \lambda\}$$

AR-based outlier detection



Detection with $AR(10)$ model

AR-based outlier detection and removal

In order to perform both detection and removal of impulsive noise, alternance between

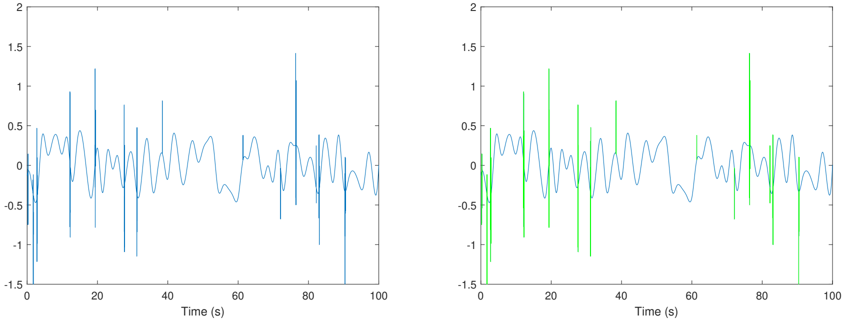
1. Estimation step: learn the AR parameters from the current time series
2. Detection step: detect the set of outliers
3. Interpolation step: replace these outliers by appropriate values
4. Reiterate steps 1, 2, 3

AR-based outlier detection and removal

Algorithm 3: AR-based outlier detection and removal

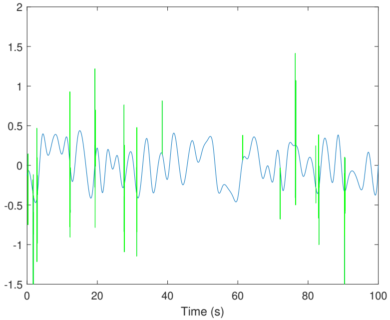
Inputs : Time series $\mathbf{x} \in \mathbb{R}^N$ with outliers
AR model order p, Threshold λ
Output: Denoised time series $\hat{\mathbf{x}} \in \mathbb{R}^N$
$\hat{\mathbf{x}} = \mathbf{x}$;
while $n_{iter} < n_{max}$ **do**
    AR estimation step;
    Estimate $\hat{\mathbf{a}}$ from $\hat{\mathbf{x}}$ with the Levinson Durbin algorithm;
    Detection step;
    Compute $\mathbf{e}$ and set $\mathcal{T} = \{n \text{ s.t. } |e[n]| > \lambda\}$;
    AR interpolation step;
    Compute $\mathbf{B}$ and $\mathbf{d}$ and solve for $\hat{\mathbf{x}}_{\mathcal{T}}$;
end

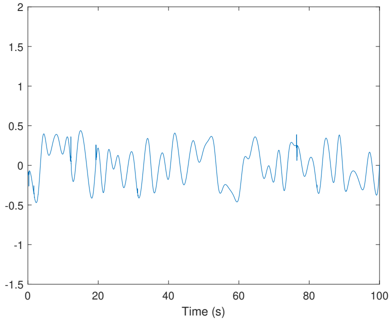
Example



Iteration 1: Detection with $AR(10)$ model

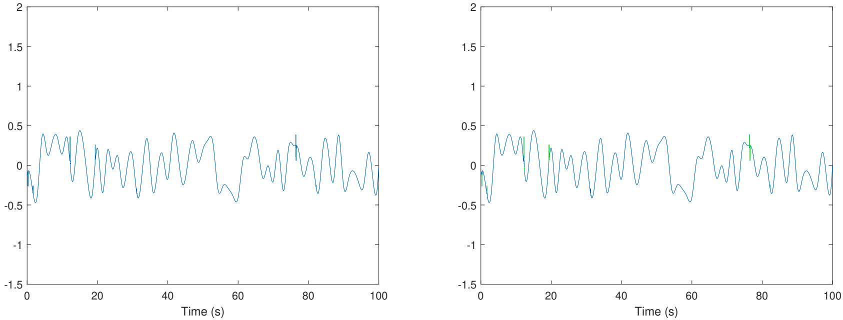
Example





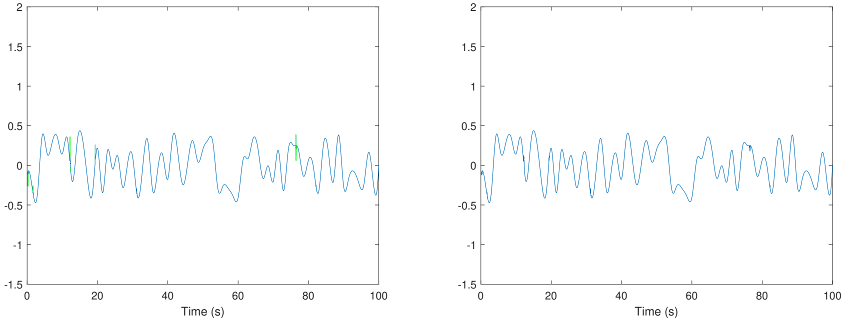
Iteration 1: Interpolation with $AR(10)$ model

Example



Iteration 2: Detection with $AR(10)$ model

Example



Iteration 2: Interpolation with $AR(10)$ model

References

- Rubinstein, R., Bruckstein, A. M., & Elad, M. (2010). Dictionaries for sparse representation modeling. Proceedings of the IEEE, 98(6), 1045-1057.
- Percival, D. B., & Walden, A. T. (2000). Wavelet methods for time series analysis (Vol. 4). Cambridge university press.
- Mallat, S. (1999). A wavelet tour of signal processing. Elsevier.
- Vautard, R., Yiou, P., & Ghil, M. (1992). Singular-spectrum analysis: A toolkit for short, noisy chaotic signals. Physica D: Nonlinear Phenomena, 58(1-4), 95-126.
- Flandrin, P., Goncalves, P., & Rilling, G. (2004, September). Detrending and denoising with empirical mode decompositions. In 2004 12th European Signal Processing Conference (pp. 1581-1584). IEEE.
- Boudraa, A. O., & Cexus, J. C. (2006). Denoising via empirical mode decomposition. Proc. IEEE ISCCSP, 4(2006).
- Comon, P. (1994). Independent component analysis, a new concept?. Signal processing, 36(3), 287-314.
- Lepot, M., Aubin, J. B., & Clemens, F. H. (2017). Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment. Water, 9(10), 796.
- McKinley, S., & Levine, M. (1998). Cubic spline interpolation. College of the Redwoods, 45(1), 1049-1060.
- Srebro, N., & Jaakkola, T. (2003). Weighted low-rank approximations. In Proceedings of the 20th International Conference on Machine Learning (ICML-03) (pp. 720-727).
- Janssen, A. J. E. M., Veldhuis, R., & Vries, L. (1986). Adaptive interpolation of discrete-time signals that can be modeled as autoregressive processes. IEEE Transactions on Acoustics, Speech, and Signal Processing, 34(2), 317-330.
- Oudre, L. (2015). Automatic detection and removal of impulsive noise in audio signals. Image Processing On Line, 5, 267-281.

List of possible topics/projects

- Flandrin, P., Goncalves, P., & Rilling, G. (2004, September). Detrending and denoising with empirical mode decompositions. In 2004 12th European Signal Processing Conference (pp. 1581-1584). IEEE.
  How to use EMD for denoising and detrending.
- Rhif, M., Ben Abbes, A., Farah, I. R., Martínez, B., & Sang, Y. (2019). Wavelet transform application for/in non-stationary time-series analysis: a review. Applied Sciences, 9(7), 1345.
  How to use wavelets to work on non-stationary time series.
- Bayer, F. M., Kozakevicius, A. J., & Cintra, R. J. (2019). An iterative wavelet threshold for signal denoising. Signal Processing, 162, 10-20.
  How to use adaptive wavelet thresholding for denoising
- Moussallam, M., Gramfort, A., Daudet, L., & Richard, G. (2014). Blind denoising with random greedy pursuits. IEEE Signal Processing Letters, 21(11), 1341-1345.
  How to use statistical considerations to set the parameters in greedy denoising approaches
- Aharon, M., Elad, M., & Bruckstein, A. (2006). K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on signal processing, 54(11), 4311-4322.
  How to learn an overcomplete dictionary with K-SVD
- de Cheveigné, A., & Arzounian, D. (2018). Robust detrending, rereferencing, outlier detection, and inpainting for multichannel data. Neuroimage, 172, 903-912.
  How to combine detrending, outlier detection and removal for multichannel data
- Hassani, H., & Mahmoudvand, R. (2013). Multivariate singular spectrum analysis: A general view and new vector forecasting approach. International Journal of Energy and Statistics, 1(01), 55-83.
  How to use SSA for forecasting time series
- Adler, A., Emiya, V., Jafari, M. G., Elad, M., Gribonval, R., & Plumbley, M. D. (2011). Audio inpainting. IEEE Transactions on Audio, Speech, and Language Processing, 20(3), 922-932..
  How to use sparse representation to perform audio inpainting