Machine Learning for Time Series

Lecture 3: Models and Representation Learning

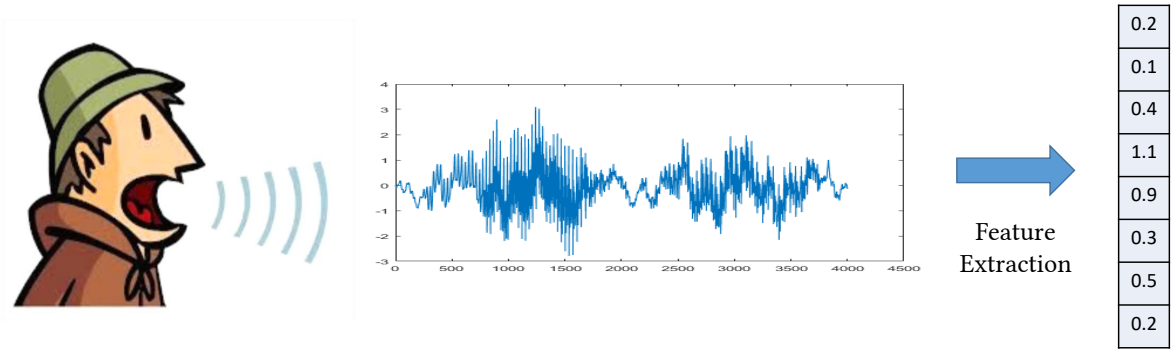
Laurent Oudre
laurent.oudre@ens-paris-saclay.fr

Master MVA
2023-2024

Contents

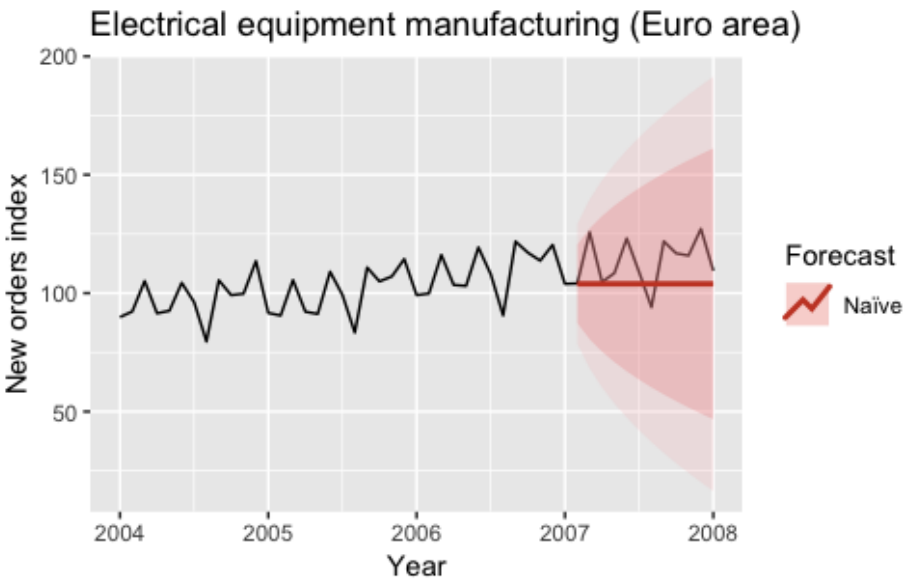
Contents

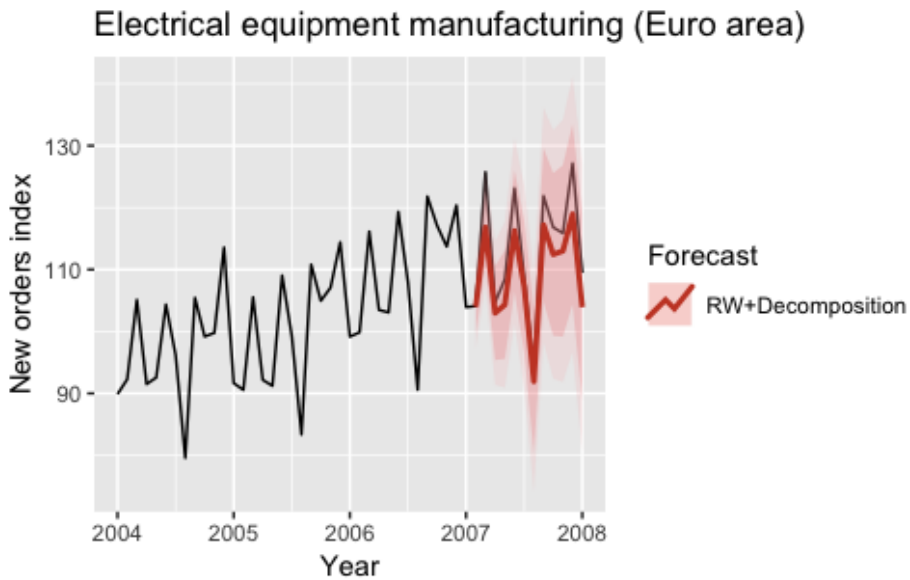
Feature Extraction



How do we represent the whole time series with a small number of features?

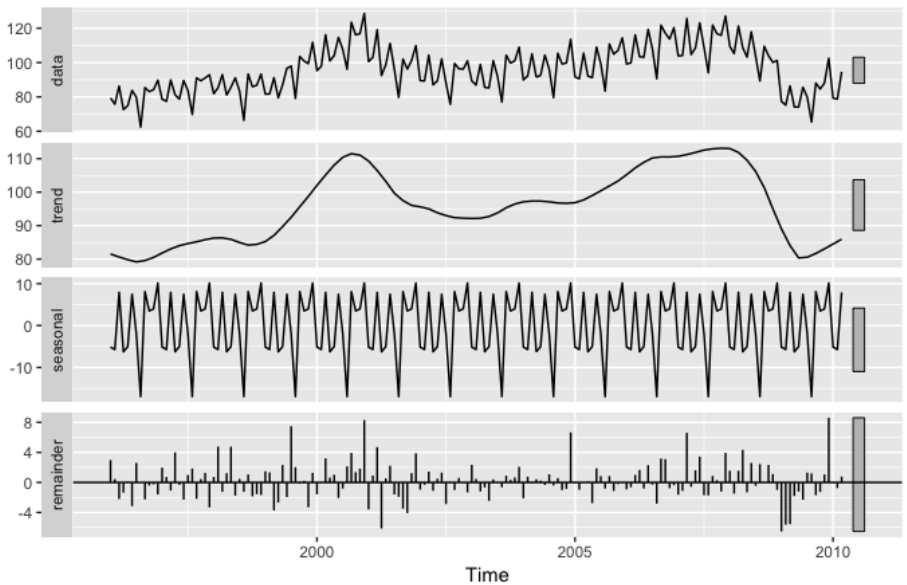
Prediction





How can we predict future values for a signal?

Problem 1: Models

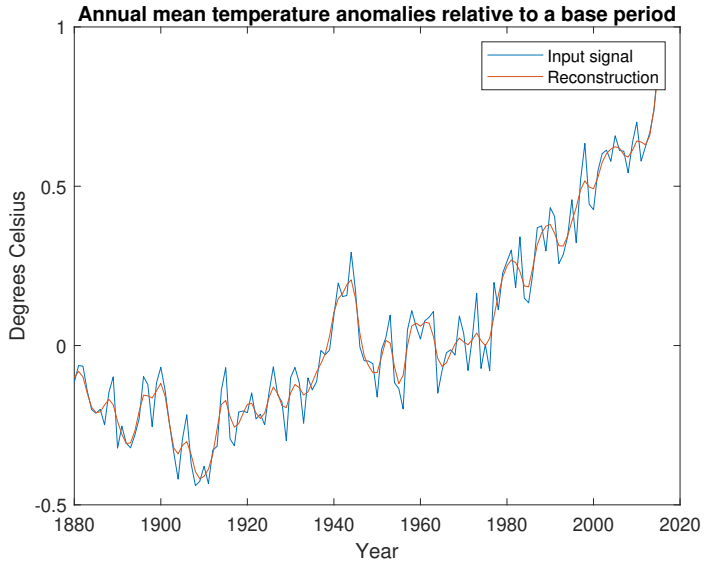


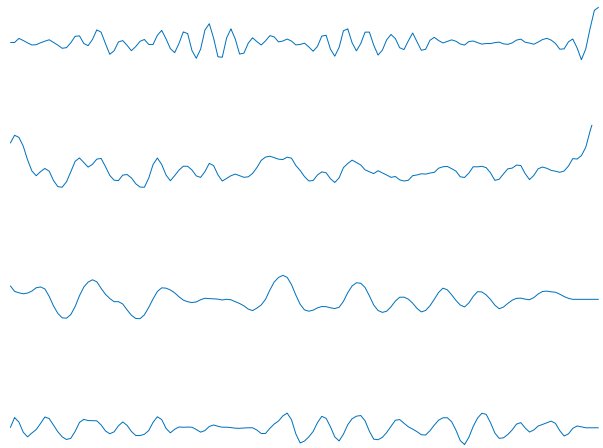
Models

Find a relevant model for a time series and infer the parameters

- Allows to represent the time series with a condensed form (set of parameters)
- These parameters can be used for clustering, classification, etc.
- The underlying model can also be used to predict future values or to reconstruct missing values

Problem 2: Representation Learning





Representation Learning

Learn an adaptive representation for a set of time series

- Instead of parametric models, learn an adequate model directly from the data
- Useful for clustering, classification, etc.
- Avoid the necessity for off-the-shelf models

Contents

Why do we need models?

- Before performing prediction, classification or feature extraction, it is important to understand the data
- To that end and since the beginning of experimental science, people have used models (mostly statistical)
- These models allow to summarize the information contained in the signal as a small number of parameters and can allow to get an idea of the behavior of the signal in the future
- These models may seem outdated but when well used, some of them often compare nicely to more recent and complex approaches (yes, including deep learning!)
- However, they are only approximations and it is important to keep in mind that in *real life*, most of the underlying hypothesis are not satisfied...

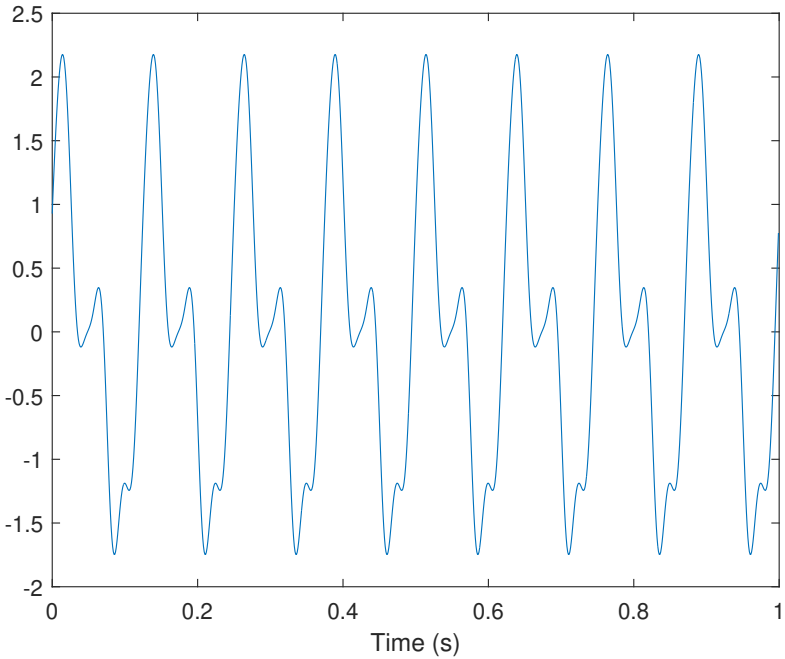
Sinusoidal model

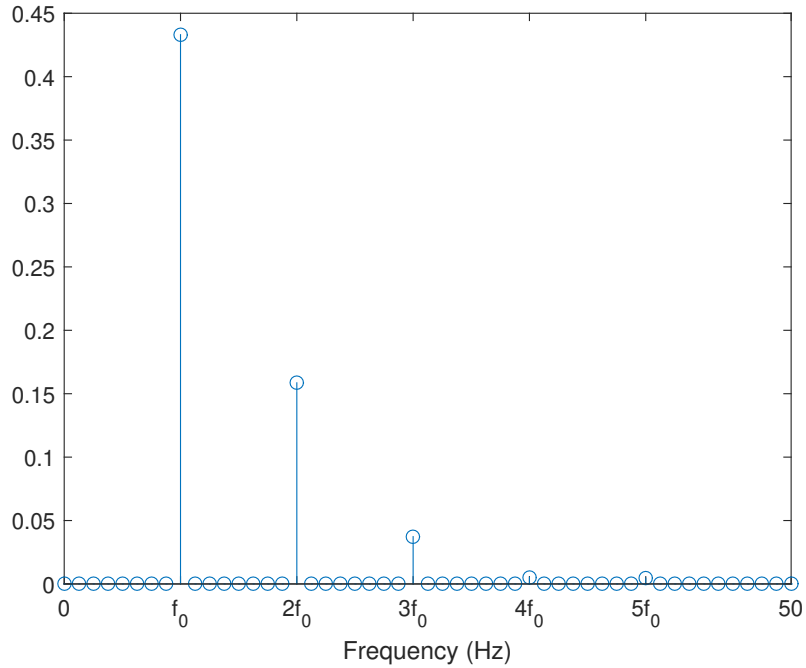
The first physical model for periodic signals is the sinusoidal model:

$$x[n] = \sum_{i=1}^{N_h} a_i \sin\left(2\pi(if_0)\frac{n}{F_s} + \phi_i\right) + b[n]$$

- F_s : sampling frequency (in Hz)
- f_0 : fundamental frequency (in Hz)
- $i \times f_0$: i^{th} harmonic
- a_i and ϕ_i : amplitudes and phases of harmonic i
- N_h : number of harmonics
- $b[n]$: white noise (uncorrelated with $x[n]$)

Example





$F_s = 1000$ Hz, $f_0 = 8$ Hz

Sinusoidal model

$$x[n] = \sum_{i=1}^{N_h} a_i \sin\left(2\pi(if_0)\frac{n}{F_s} + \phi_i\right) + b[n]$$

- Full characterization of the signal with the knowledge of f_0 and properties of the harmonics a_i
- Useful for indexation, description and prediction of purely periodic phenomenon
- An important feature is the normalized amplitude of each harmonic

$$\frac{a_2}{a_1}, \frac{a_3}{a_1}, \frac{a_4}{a_1}, \ldots$$

also referred to as *timbre*, by analogy with musical instruments

Parameter estimation

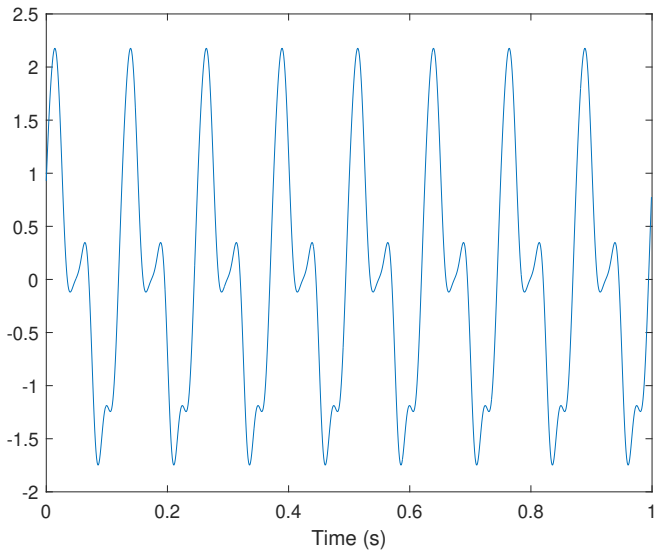
Two parameters must be estimated

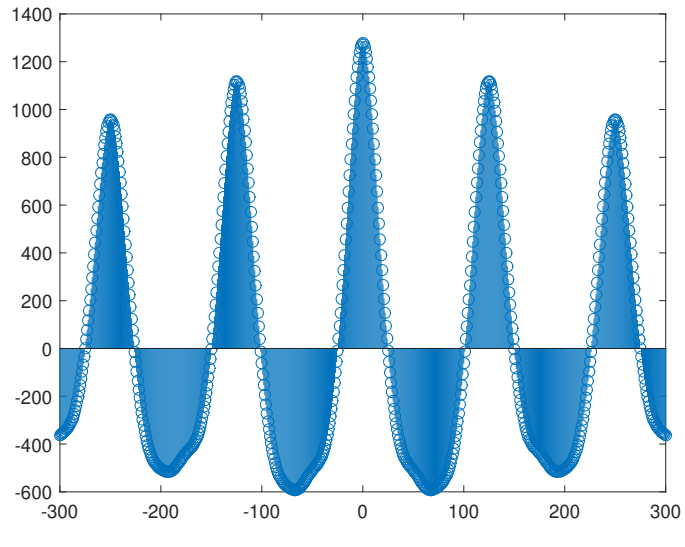
- **Fundamental frequency f_0:**
  - Search for a peak on the spectrum and determine the associated frequency
  - Often more robust: study of the autocorrelation function (cf next slide)
- **Harmonics parameters a_i and ϕ_i:**
  - Study of the DFT values for frequencies multiple of f_0
  - Regression on a basis of functions $\sin\left(2\pi(if_0)\frac{n}{F_s}\right)$ and $\cos\left(2\pi(if_0)\frac{n}{F_s}\right)$ (see trend+seasonality model)

f_0-estimation

Résiste très bien au bruit

1. Plot the autocorrelation function (cf Lecture 2)
2. First peak for positive lags corresponds to the signal period m_0
3. f_0 estimated as $\hat{f}_0 = \frac{F_s}{m_0}$





$F_s = 1000$ Hz, visible peak for $m_0 = 125$ so $\hat{f}_0 = \frac{1000}{125} = 8$ Hz

Extension to other functions

- In the sinusoidal model, we used the function $\beta(t) = \sin(2\pi f_0 t + \phi)$ to model the deterministic component of the signal
- In a more general setting, several functions can be used to model a time series
  - Some with slow variations, that will be called **trend**

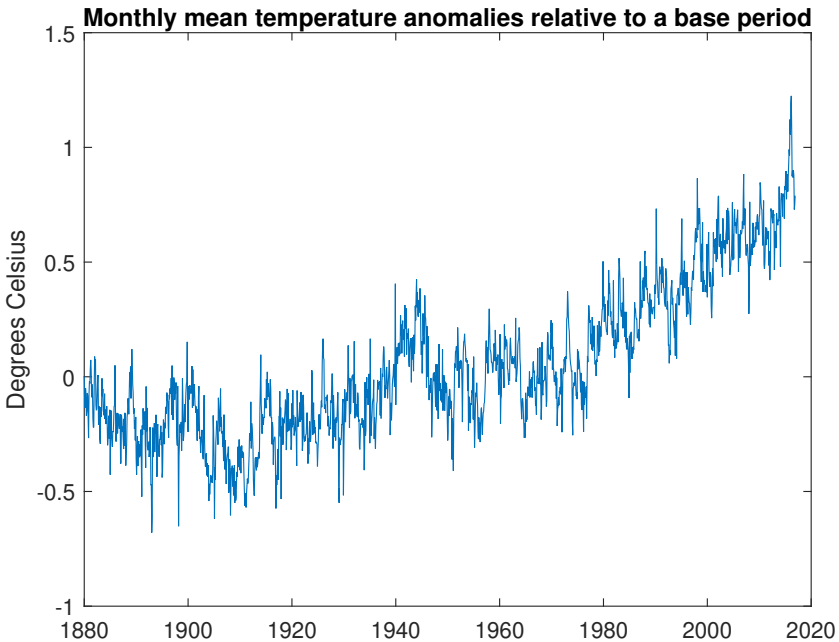
$$\beta(t) = cst \text{ (constant)}, \beta(t) = t \text{ (linear)}, \beta(t) = t^2 \text{ (quadratic)...}$$

  - Some with periodic variations, that will be called **seasonality**

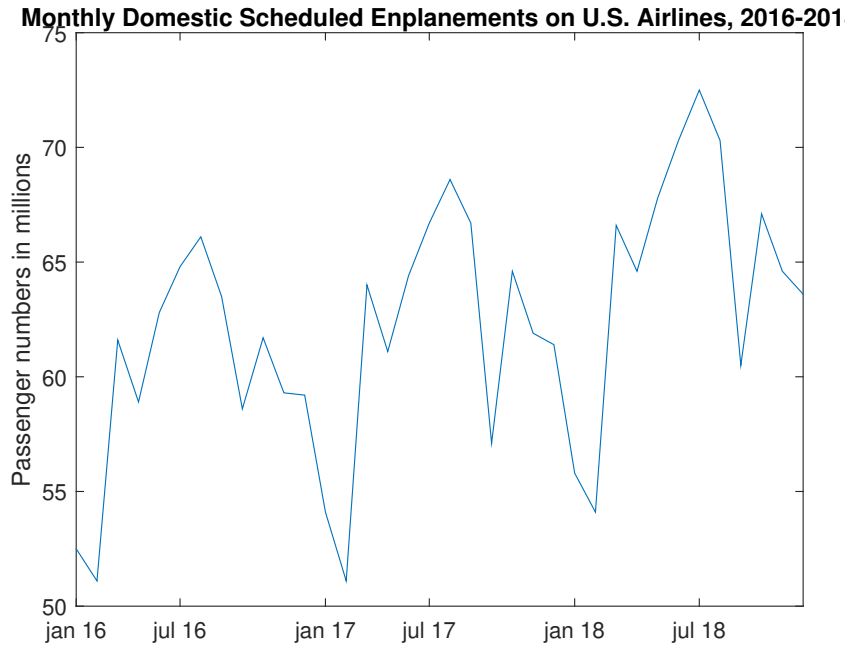
$$\beta(t) = \cos(2\pi f t), \beta(t) = \sin(2\pi f t)...$$

Examples

Tendance augmente



Tendance augmente + saisonnalité



Trend+Seasonality model

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

- $\beta_1(t), \ldots \beta_d(t)$: known functions
- $\alpha_1, \ldots \alpha_d$: real parameters to be estimated
- Model estimation is recast as a regression task, i.e. approximating the time series as a linear combination of known functions

Parameter estimation

- From $x[0:N-1]$ we need to learn $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_d)^T$
- Least-Square estimation

$$\mathbf{x} = (x[0], \ldots, x[N-1])^T$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_1(0) & \cdots & \beta_d(0) \\ \beta_1(T_s) & \cdots & \beta_d(T_s) \\ \vdots & \ddots & \vdots \\ \beta_1((N-1)T_s) & \cdots & \beta_d((N-1)T_s) \end{pmatrix}$$

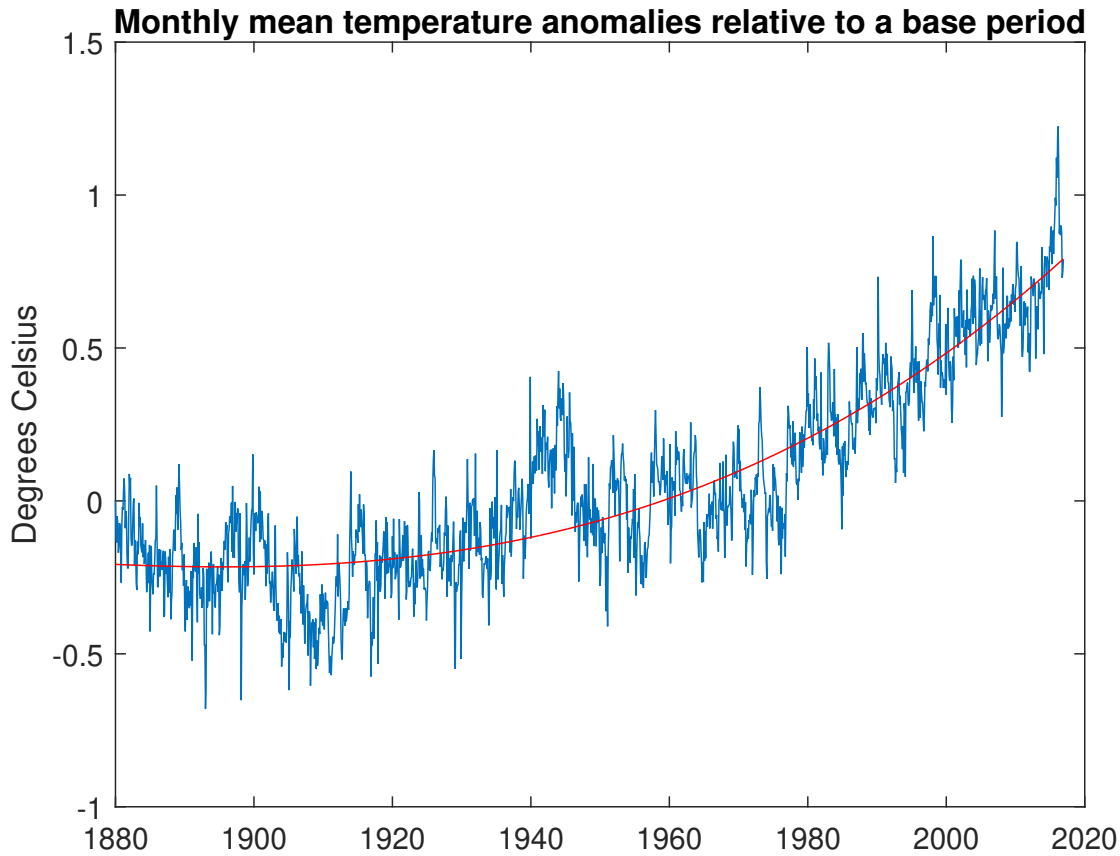
- Minimization of

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

- Closed-form solution

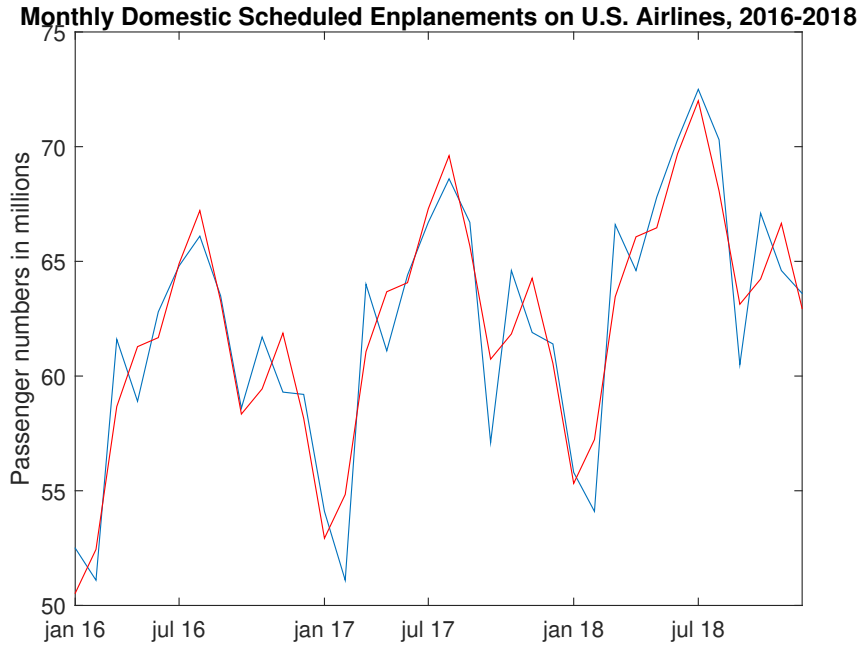
$$\hat{\boldsymbol{\alpha}} = \left(\boldsymbol{\beta}^T\boldsymbol{\beta}\right)^{-1}\boldsymbol{\beta}^T\mathbf{x}$$

Example: meteo



$$\beta_1(t) = cst, \beta_2(t) = t, \beta_3(t) = t^2, \beta_4(t) = t^3$$

Example: plane



$$\beta_1(t) = cst, \beta_2(t) = t$$

$$\beta_3(t) = \cos(2\pi f_0 t), \beta_4(t) = \sin(2\pi f_0 t) \text{ with } f_0 = \frac{1}{12}$$

$$\beta_5(t) = \cos(4\pi f_0 t), \beta_6(t) = \sin(4\pi f_0 t), \beta_7(t) = \cos(6\pi f_0 t), \beta_8(t) = \sin(6\pi f_0 t)$$

Correlations between samples

- When we consider a time series, it makes sense to think that what we observe for sample $x[n]$ is somehow linked to previous values $x[n-1], \ldots, x[n-p]$
- These correlations can indeed easily be seen by studying the autocorrelation function.
- Idea: instead of using off-the-shelf functions to perform regression, we can perform it on the signal itself: **autoregressive (AR)** model [Hamilton, 1994 ; Box et al., 2011]
- Multitude of variants: ARMA, ARIMA, etc...

Autoregressive model $AR(p)$

Utile pour se rapprocher le plus possible du signal réel

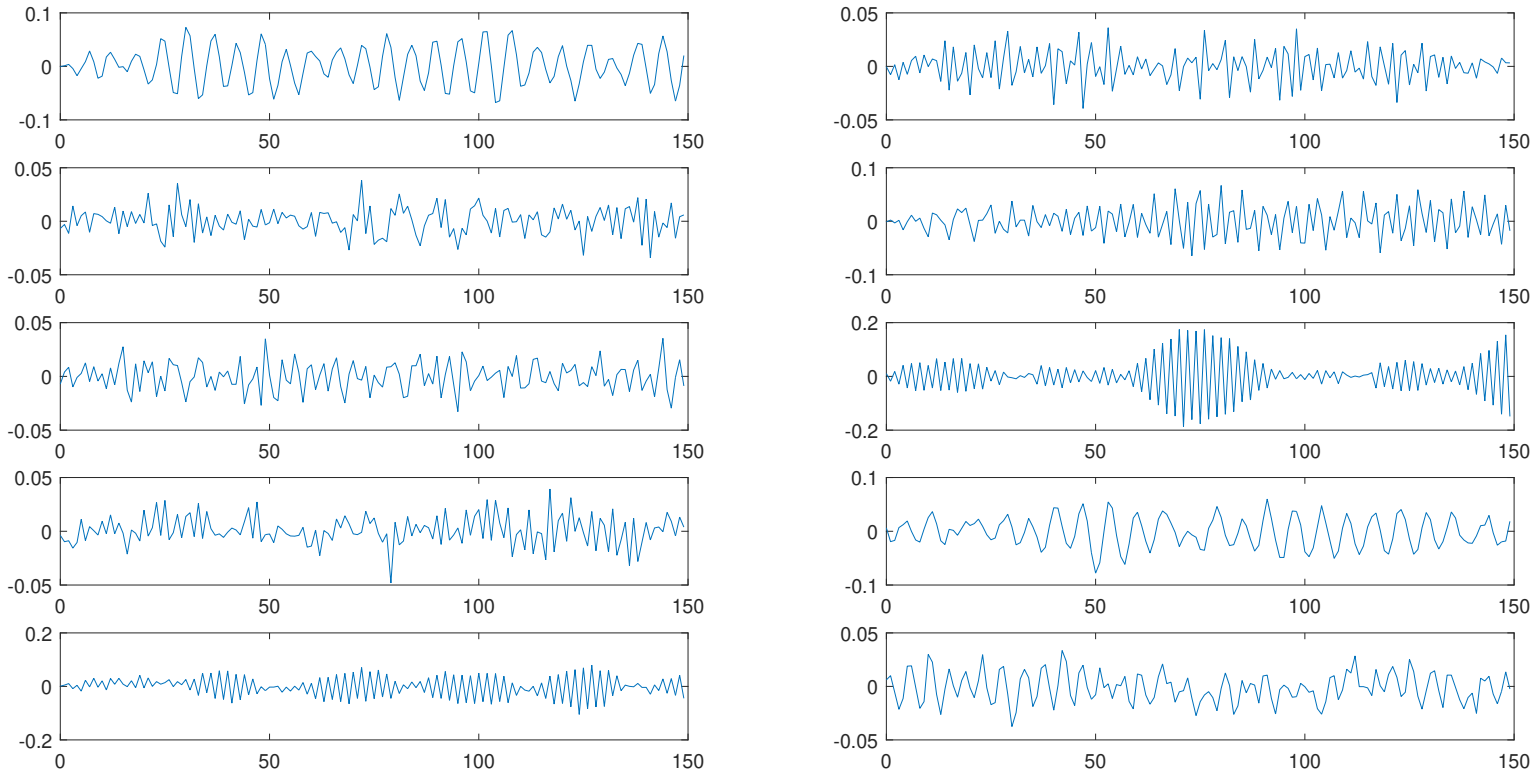
$$x[n] = -\sum_{i=1}^{p} a_i x[n-i] + b[n]$$

- p : order of the model
- $a_1, \ldots, a_p$: AR coefficients
- $b[n]$: white noise (often called innovation)

Properties

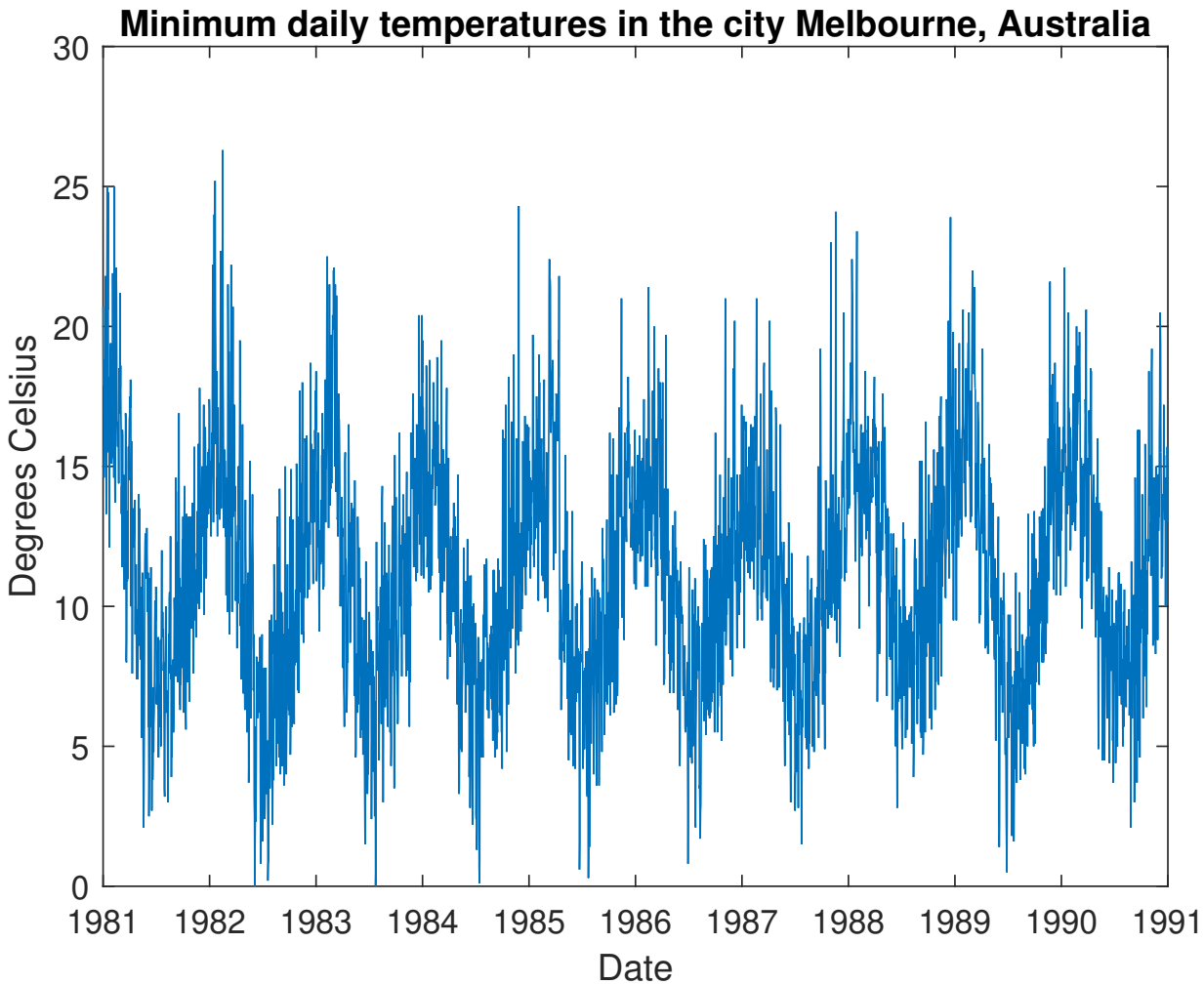
- Under some reasonable assumptions, an $AR(p)$ process can be stationary: we will consider it is true in the following
- Coefficients a_i can be used for indexing and prediction
- This model is also linked to a physical *source/filter* model when a linear filter is excited with a white noise. This model is very common for speech and sound processing.
- These coefficients also allow to approximate the power spectral density of the signal (parametric PSD estimation) [Priestley, 1981]

Examples



Several $AR(5)$ processes

Prediction



Temperature data with 1 sample per day. We use $p = 365$

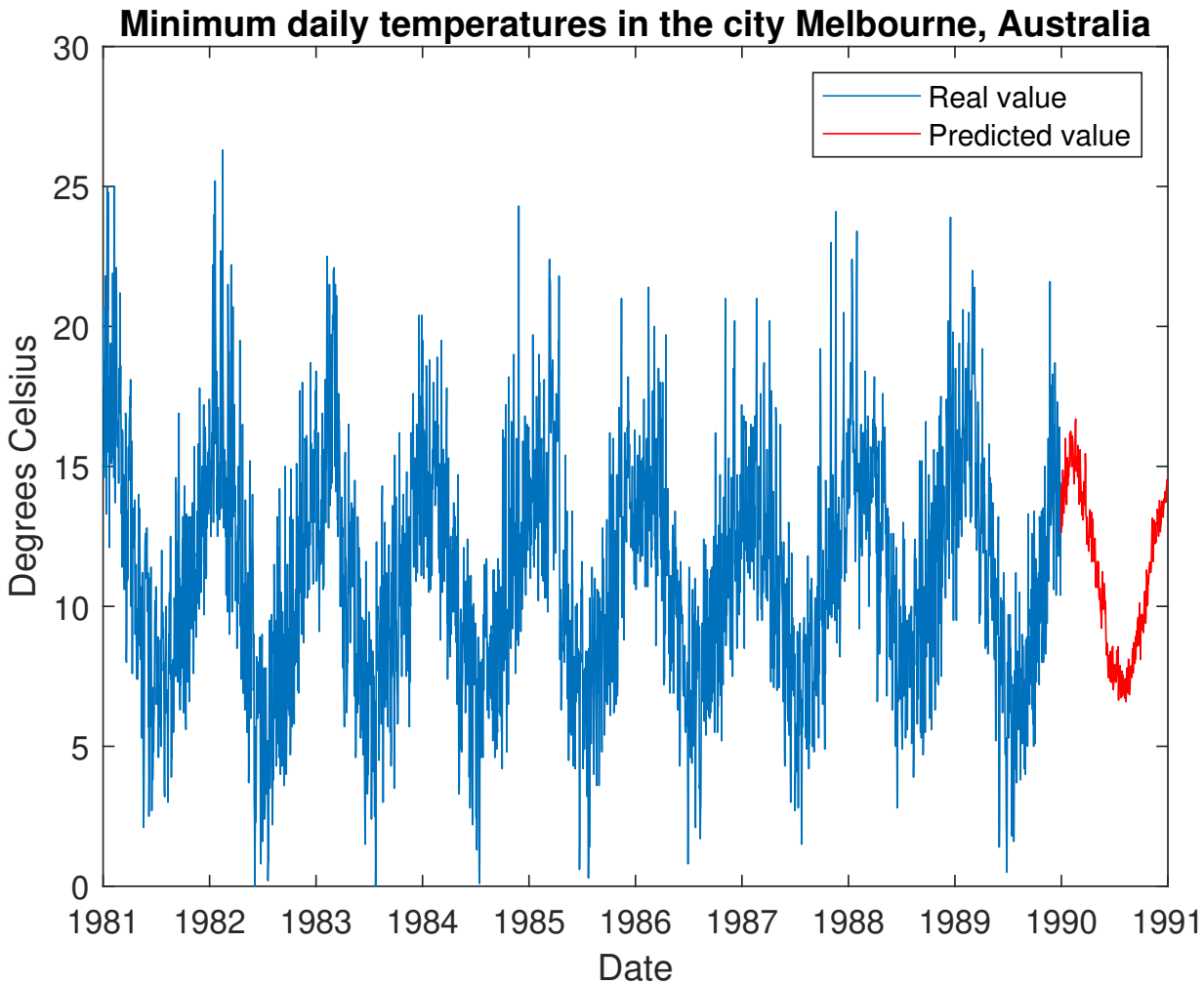
Prediction

Autoregressive models can be used for prediction at step h [Box et al., 2011 ; Pankratz, 2009]

1. Learn the AR parameters on the first $N - h + 1$ samples
2. Reconstruction of the last h samples by recursivity

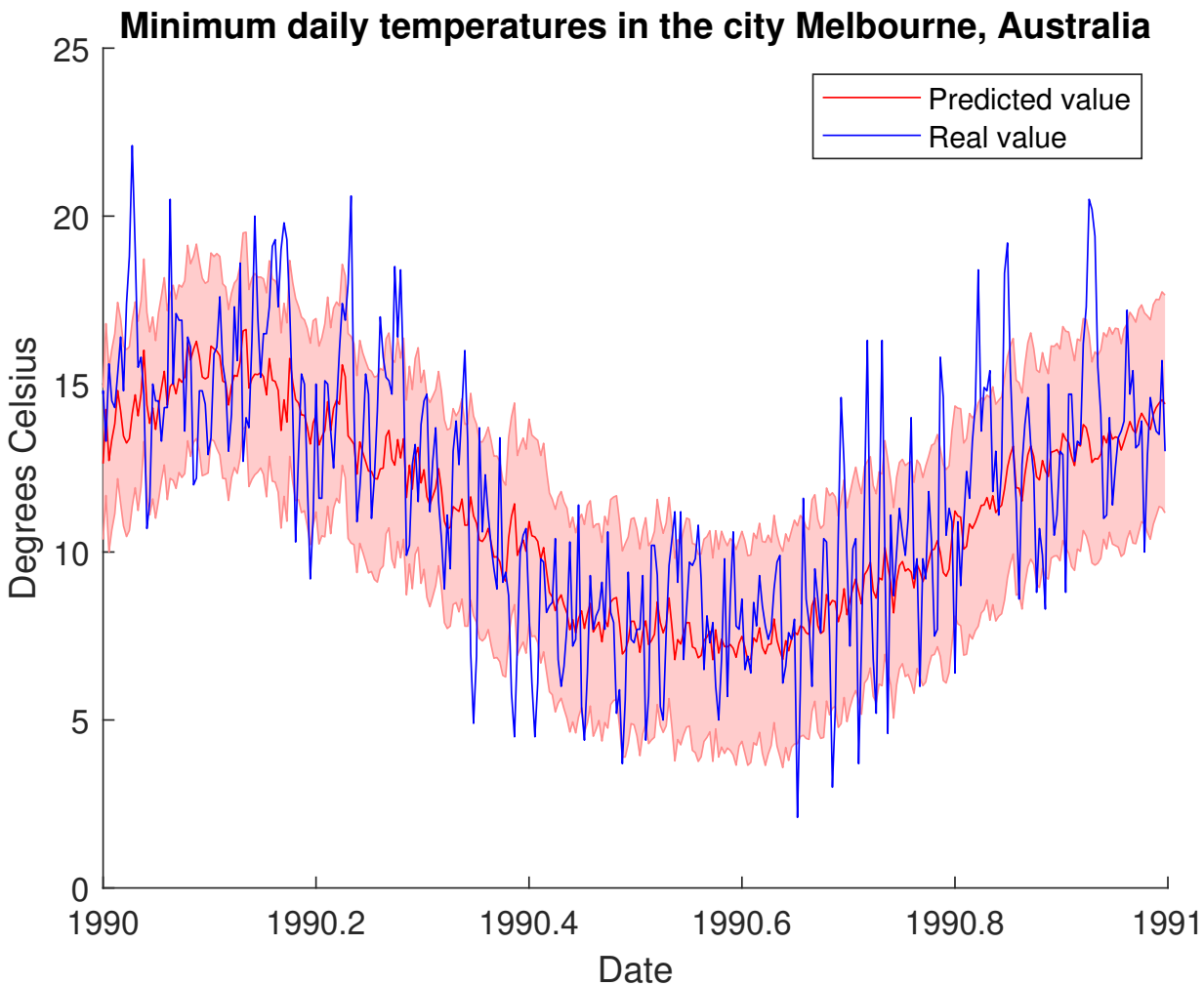
$$\hat{x}[n] = -\sum_{i=1}^{p} \hat{a}_i x[n-i]$$

Prediction



Prediction of the $h = 365$ last data points

Prediction



Prediction of the $h = 365$ last data points (in red is the standard deviation on 1000 simulations with random noise)

Parameter estimation

- In order to estimate the parameters, we can go back to the main equation

$$x[n] + a_1 x[n-1] + \ldots + a_p x[n-p] = b[n]$$

- By multiplying by $x[n-1]$ we obtain

$$x[n]x[n-1] + a_1 x[n-1]^2 + \ldots + a_p x[n-p]x[n-1] = b[n]x[n-1]$$

- By taking the expected value

$$\mathbb{E}\left[x[n]x[n-1]\right] + a_1 \mathbb{E}\left[x[n-1]^2\right] + \ldots + a_p \mathbb{E}\left[x[n-p]x[n-1]\right] = \mathbb{E}\left[b[n]x[n-1]\right]$$

- Since $b[n]$ and $x[n-1]$ are uncorrelated and $b[n]$ is a white noise

$$\mathbb{E}\left[b[n]x[n-1]\right] = \mathbb{E}\left[b[n]\right]\mathbb{E}\left[x[n-1]\right] = 0$$

- If $x[n]$ is wide-sense stationary, we have

$$\mathbb{E}\left[x[n_1]x[n_2]\right] = \gamma_x[|n_1 - n_2|]$$

Parameter estimation

- The equation becomes

$$\gamma_x[1] + a_1\gamma_x[0] + \ldots + a_p\gamma_x[p-1] = 0$$

- The same principle can be applied by multiplying the main equation by $x[n-2], x[n-3], \ldots$, leading to the following system of equations called **Yule-Walker** equations:

$$\begin{bmatrix} \gamma_x[0] & \gamma_x[1] & \cdots & \gamma_x[p-1] \\ \gamma_x[1] & \gamma_x[0] & \cdots & \gamma_x[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_x[p-1] & \gamma_x[p-2] & \cdots & \gamma_x[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} \gamma_x[1] \\ \gamma_x[2] \\ \vdots \\ \gamma_x[p] \end{bmatrix}$$

- Knowing the autocorrelation function $\gamma_x[m]$ for $m = 0 \ldots p$ is sufficient to estimate the parameters
- Most of the times, we use an empirical estimator of the autocorrelation function instead (see Lecture 2)

Estimation of the order

- Due to the Toeplitz nature of the matrix, the system

$$\begin{bmatrix} \gamma_x[0] & \gamma_x[1] & \cdots & \gamma_x[p-1] \\ \gamma_x[1] & \gamma_x[0] & \cdots & \gamma_x[p-2] \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_x[p-1] & \gamma_x[p-2] & \cdots & \gamma_x[0] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_p \end{bmatrix} = - \begin{bmatrix} \gamma_x[1] \\ \gamma_x[2] \\ \vdots \\ \gamma_x[p] \end{bmatrix}$$

  can be solved recursively
- Given the parameters $\mathbf{a}^k = [a_1^k, a_2^k, \ldots, a_k^k]$ estimated for the $k-$order model, the parameters $\mathbf{a}^{k+1}$ for the $(k+1)$-order model can easily be computed by
  1. Computing the *new parameter* a_{k+1}^{k+1}
  2. Updating the coefficients from $[a_1^k, a_2^k, \ldots, a_k^k]$ to $[a_1^{k+1}, a_2^{k+1}, \ldots, a_k^{k+1}]$

Levinson-Durbin recursion [Durbin, 1960]

Levinson-Durbin recursion

Algorithm 1: Levinson-Durbin recursion

Inputs : Order p

Autocorrelation coefficients $\gamma_x[0], \ldots, \gamma_x[p]$ (or estimates)

Output: AR coefficients $\mathbf{a} = \mathbf{a}^p$

$\mathbf{a}^1 = a_1^1 = \frac{\gamma_x[1]}{\gamma_x[0]}$;

while $k \leq p-1$ **do**

$\mathbf{c}^k = [\gamma_x[1], \ldots, \gamma_x[k]]^T$;

$\alpha^{k+1} = \gamma_x[k+1] - \left(\mathbf{a}^k\right)^T \left(\mathbf{c}^k\right)^{\leftarrow}$;

$\beta^{k+1} = \gamma_x[0] - \left(\mathbf{a}^k\right)^T \left(\mathbf{c}^k\right)$;

Computation of the new coefficient;

$a_{k+1}^{k+1} = \frac{\beta^{k+1}}{\alpha^{k+1}}$;

Update of the other coefficients;

$[a_1^{k+1}, a_2^{k+1}, \ldots, a_k^{k+1}]^T = \mathbf{a}^k - a_{k+1}^{k+1} \left(\mathbf{a}^k\right)^{\leftarrow}$;

$\mathbf{a}^{k+1} = [a_1^{k+1}, a_2^{k+1}, \ldots, a_k^{k+1}, a_{k+1}^{k+1}]^T$

end

Estimation of the order

a_k traduit de la qualité de l'information apportée par la valeur espacé de k de l'échantillon --> permet de trouver p

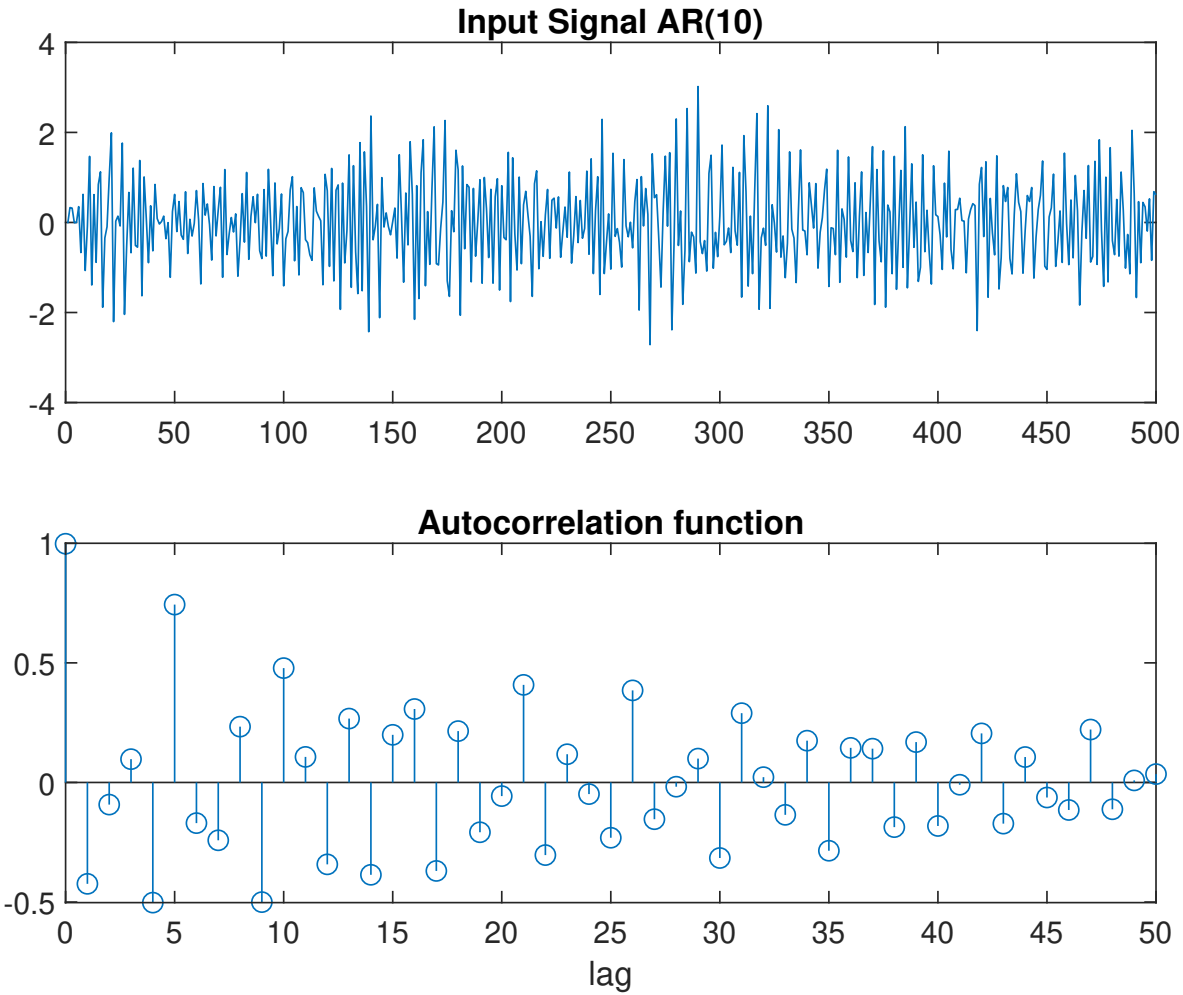
- By computing the coefficients a_k^k for several orders from 1 to p_{max}, it is possible to estimate the order
- Indeed, if $\mathbf{x}$ is $AR(p)$, then a_{p+1}^{p+1} must be equal to 0 since $x[n]$ can be fully reconstructed from $x[n-1], \ldots, x[n-p]$. The same reasoning applies to higher orders
- With the Levinson-Durbin algorithm, we can compute the so-called **partial autocorrelation function (PACF)** [Box et al., 2011]:

des que pacf s'annule --> on sait que l'on a capturé toutes les infos

$$\text{PACF}[m] = a_m^m$$

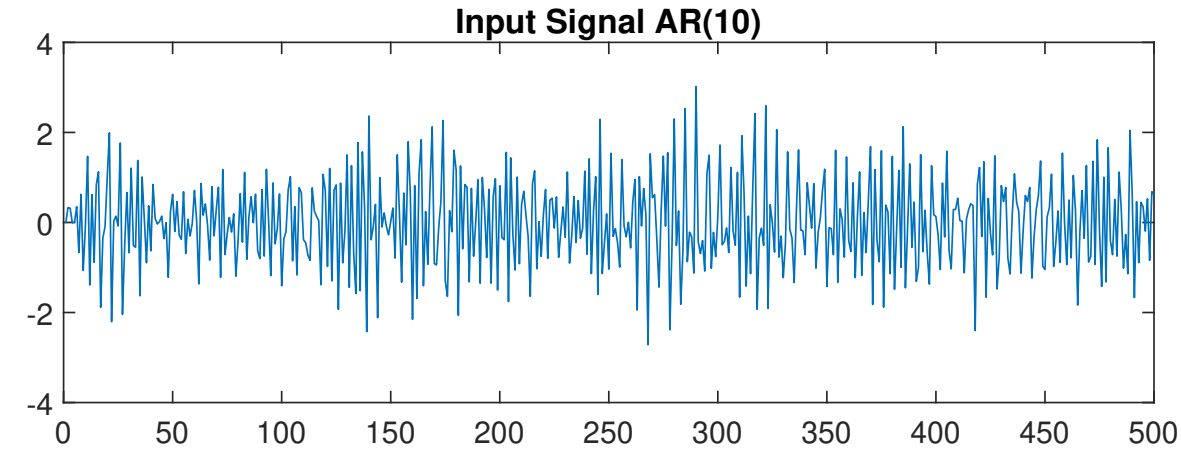
- This function quantifies the autocorrelation between $x[n]$ and $x[n-m]$, when all linear dependence of other lags $x[n-1], \ldots, x[n-m+1]$ have been removed

Example

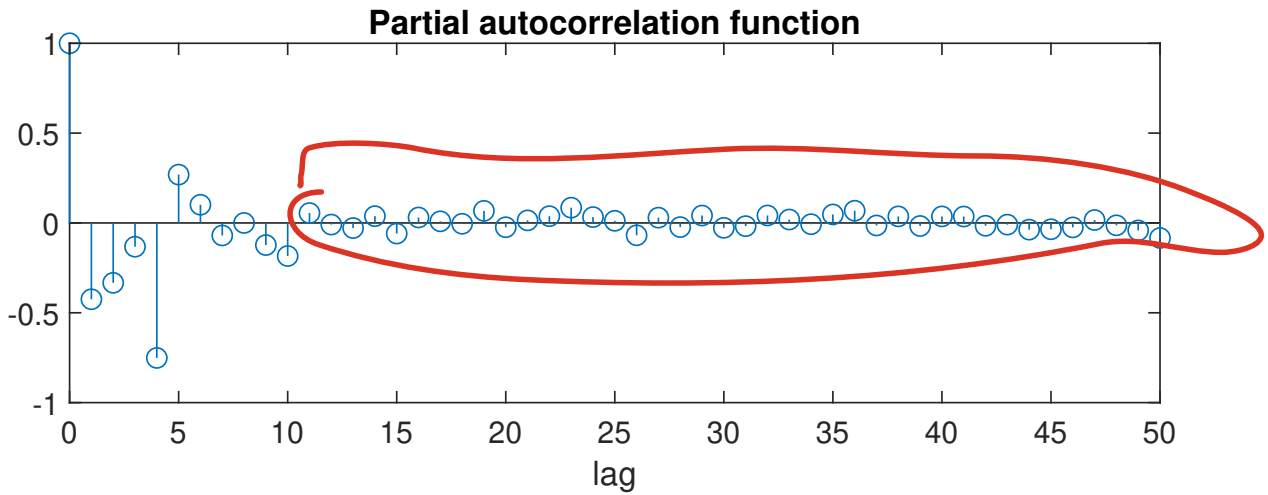


The autocorrelation function does not provide much information on the order of the process

Example



Après 10, tous égaux à 0



The partial autocorrelation function is equal to 0 for $m > p$ where p is the order of the process

Moving average model $MA(q)$

$$x[n] = b[n] + \sum_{j=1}^{q} m_j b[n-j]$$

- q : order of the model
- $m_i, \ldots, m_q$: MA coefficients
- $b[n]$: white noise

Stationary process, weighted sum of innovations

Wold's theorem (1954): every wide-sense stationary process can be modeled as the sum of a deterministic signal and a $MA(q)$ process (possibly with $q = \infty$)

Autoregressive moving average model $ARMA(p, q)$

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

- p, q : orders of the model
- $a_i, \ldots, a_p$: AR coefficients
- $m_i, \ldots, m_q$: MA coefficients
- $b[n]$: white noise

Persistence in the phenomenon and in the innovations

Autoregressive integrated moving average model $ARIMA(p, d, q)$

$$x^{(d)}[n] = -\sum_{i=1}^{p} a_i x^{(d)}[n-i] + b[n] + \sum_{j=1}^{q} m_j b[n-j]$$

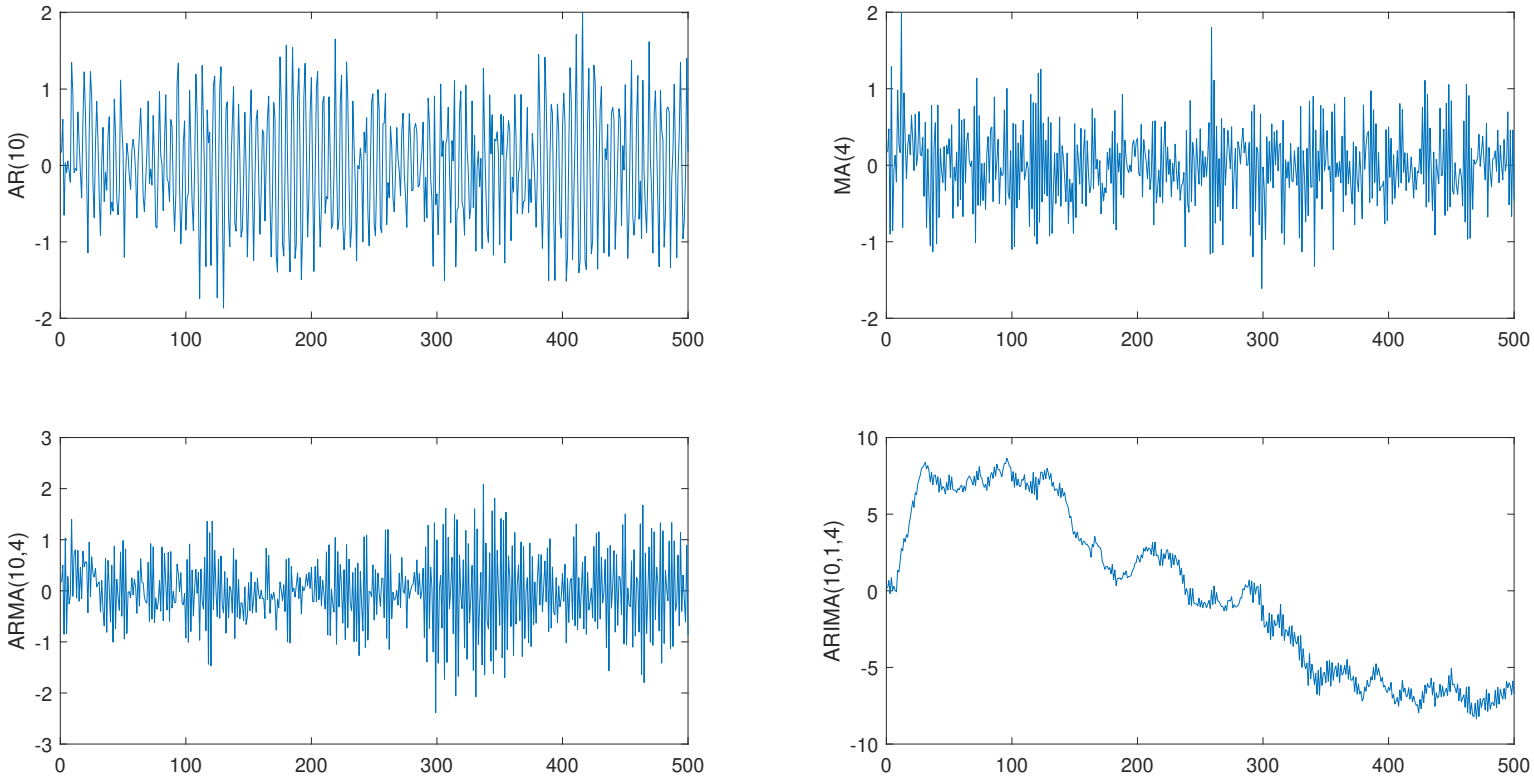
where $\mathbf{x^{(d)}}$ is the d^{th} differentiation of the time series

peut modéliser des signaux non stationnaires (en dérivant, on espère obtenir un signal sationnaire)

- p, q : orders of the model
- d : differentiation order
- $a_i, \ldots, a_p$: AR coefficients
- $m_i, \ldots, m_q$: MA coefficients
- $b[n]$: white noise

ARMA model extended to possible non-stationary processes

Examples



Same innovations, same parameters

How to infer the parameters and orders?

- Setting the orders:
  - Not as easy as for AR models: autocorrelation functions and partial autocorrelation functions only give an indicative idea. Some theoritical results can be shown but do not translate into practical do's and dont's [Box et al., 2011 ; Pankratz, 2009 ; Bisgaard et al., 2011]
  - Some model selection criteria can be used to that aim: Bayesian Information Criterion (BIC), Akaike Information Criterion (AIC)... (see Lecture 5)
- Learning the parameters:
  - Not as easy as for AR models as well!
  - Parameter estimation requires ad hoc solving of optimization problems through maximum likelihood estimation or non-linear least-squares estimation: Box-Jenkins model estimation [Box et al., 2011]

A word on Box & Jenkins methodology

1. Model identification and selection
   - Is the time series stationary? After how many numerical derivations is it stationary? Can help to find d
   - Does the time series contain a seasonal component? Can help to find orders p and q
   - Selection of the rights orders by using the autocorrelation and partial autocorrelation function, or model selection criteria
2. Parameter estimation with ad hoc optimization problems (e.g. maximium likelihood estimation)
3. Statistical model checking
   - Does the residue have the expected properties? Is is stationary? White noise?

RESIDUES IMPORTANT A REGARDER

Hidden Markov model

- In several application contexts, the values observed in the time series can be assumed to be due to an inner state
  Example: Temperature and precipitations may be influenced by the season
- This inner state is assumed to be taken in a finite alphabet $\mathcal{S} = \{1, \ldots, S\}$ of size S... and to be unknown ! (hence the hidden part)
- We only have access to some observations (possibly multivariate) $x[0], \ldots, x[N-1]$ and our aim is to infer the hidden state sequence $s[0], \ldots, s[N-1]$, where $s[n] \in \mathcal{S}$
- For sake of simplicity, we will assume that the time series is also discrete and takes its values in a finite alphabet $\mathcal{X} = \{1, \ldots, M\}$ (see Lecture 2 for quantization/symbolization for time series).

Probabilistic formulation

The Hidden Markov Model (HMM) [Rabiner, 1989] is characterized by:

- An initial state probability distribution

$$p_i^0 = \mathbb{P}(s[0] = i)$$

- The state-transition probability distribution

$$p_{i,j} = \mathbb{P}(s[n+1] = j | s[n] = i)$$

- The emission probability distribution

$$e_{i,m} = \mathbb{P}(x[n] = m | s[n] = i)$$

Markov assumptions

The Hidden Markov Model (HMM) is based on several assumptions on the state sequence $s[0], \ldots, s[N-1]$:

- It forms a Markov sequence, i.e.

$$\forall n, k, \quad \mathbb{P}(s[n+1]|s[n], \ldots, s[n-k]) = \mathbb{P}(s[n+1]|s[n])$$

  Future only depends on the present

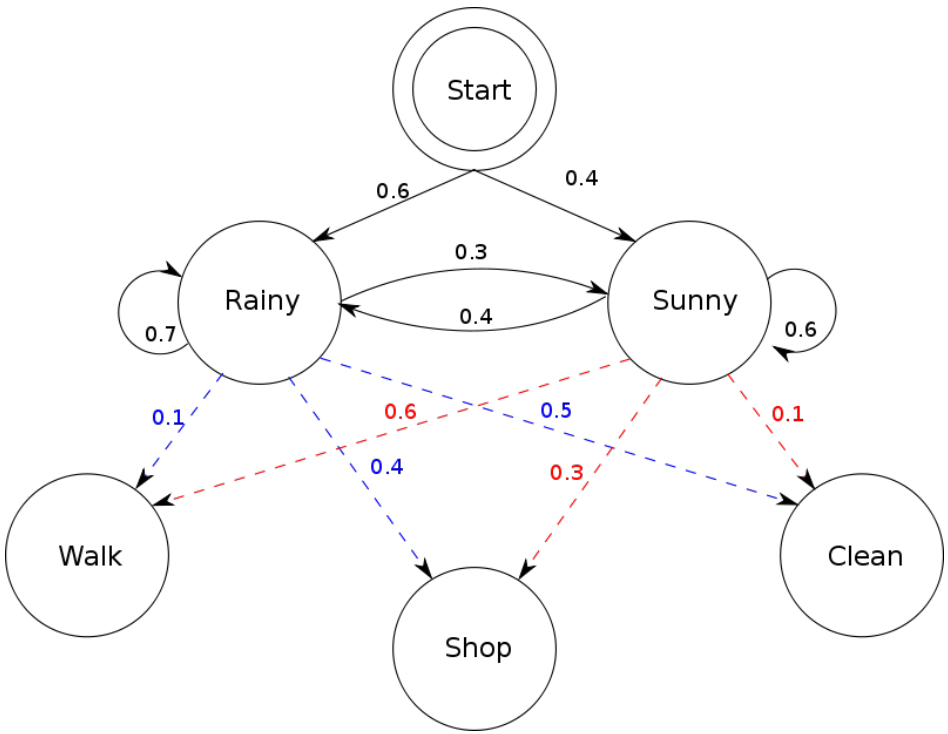
- It is is time-homogeneous

$$\forall n, n', \quad \mathbb{P}(s[n+1]|s[n]) = \mathbb{P}(s[n'+1]|s[n'])$$

- It is stationnary

$$\exists \pi, \sum_i \pi_i = 1, \quad \text{s.t. } \pi \mathbf{P} = \pi$$

Example



Here:

- Two hidden states: Rainy or Sunny
- Three possible observations: Walk, Shop, Clean

$$\mathbf{p^0} = [0.6 \;\; 0.4]$$

$$\mathbf{P} = \begin{bmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{bmatrix}$$

$$\mathbf{E} = \begin{bmatrix} 0.1 & 0.4 & 0.5 \\ 0.6 & 0.3 & 0.1 \end{bmatrix}$$

Estimation of the parameters

The learning process aims at learning from the observed sequence $\mathbf{x} = [x[0], \ldots, x[N-1]]$

- The initial state probability distribution $\mathbf{p^0}$ (vector of length S)
- The state-transition probability distribution $\mathbf{P}$ (matrix of size $S \times S$)
- The emission probability distribution $\mathbf{E}$ (matrix of size $S \times M$)

Our aim is to estimate

$$\max_{\mathbf{p^0},\mathbf{P},\mathbf{E}} \mathbb{P}\left(\mathbf{x}|\mathbf{p^0},\mathbf{P},\mathbf{E}\right)$$

Baum-Welch algorithm

Alternative du l'algo EM

Given current estimates for $\boldsymbol{\theta} = (\mathbf{p^0}, \mathbf{P}, \mathbf{E})$, alternate between three phases:

1. Forward procedure

$$\text{Compute } \alpha_i[n] = \mathbb{P}(s[n] = i, x[0:n]|\boldsymbol{\theta})$$

2. Backward procedure

$$\text{Compute } \beta_i[n] = \mathbb{P}(x[n+1:N-1]|s[n] = i, \boldsymbol{\theta})$$

3. Smoothing

$$\text{Compute } \mathbb{P}(s[n] = i|\mathbf{x}, \boldsymbol{\theta}) \text{ and } \mathbb{P}(s[n] = i, s[n+1] = j|\mathbf{x}, \boldsymbol{\theta})$$

$$\text{from } \boldsymbol{\alpha} \text{ and } \boldsymbol{\beta} \text{ using Bayes' theorem}$$

4. Update $\boldsymbol{\theta}$

See [Bilmes, 1998] for equations and more details

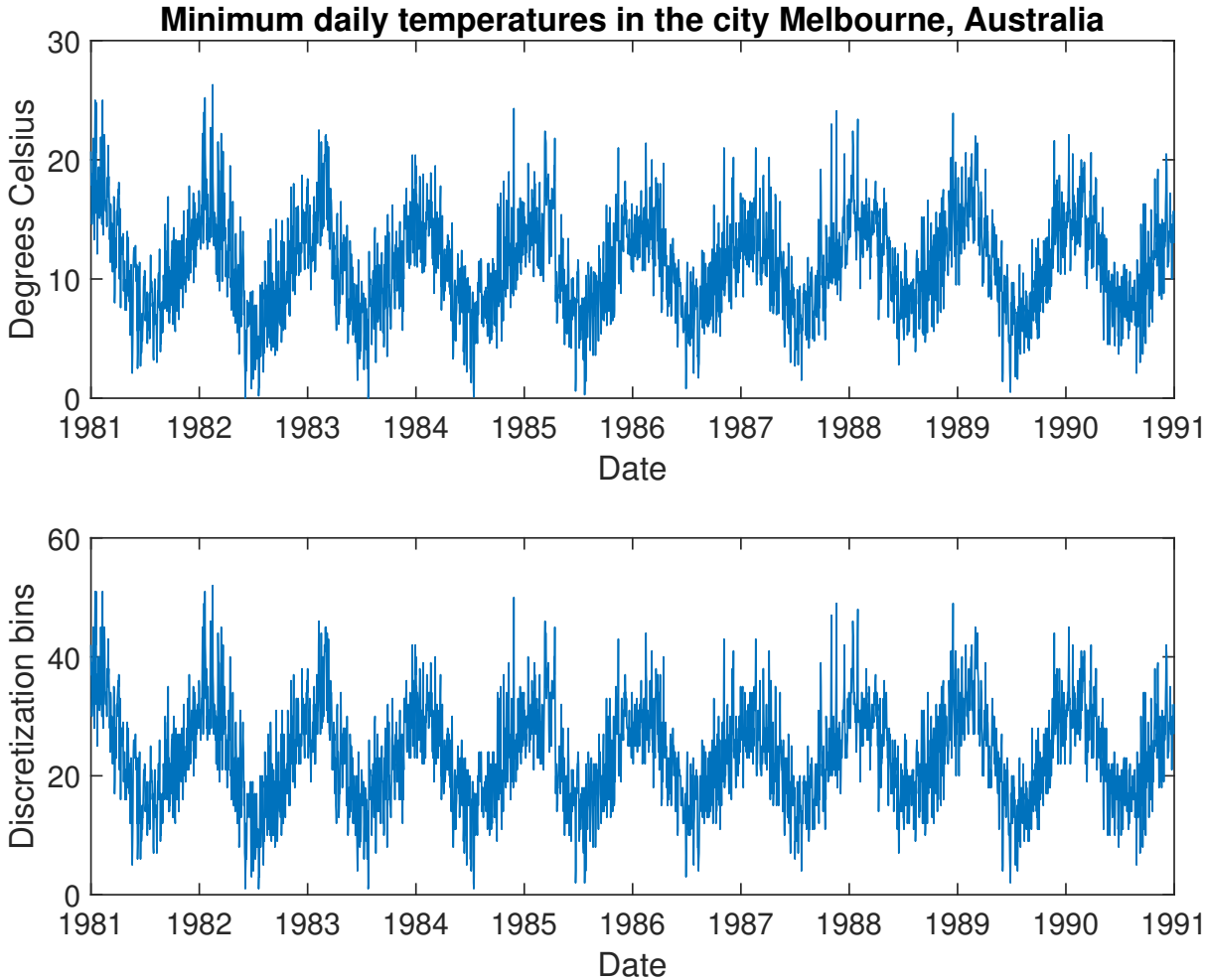
Remarks

- HMM also allow to compute the state sequence **s** from the observation sequence **x** (Viterbi algorithm). Once again, it all consists in probabilistic reformulation to find the most likely state sequence given the input sequence.
- Several emission models can be implemented for real data
  - Gaussian model. Instead of estimating the emission matrix **E**, we estimate the mean and standard deviation of a Gaussian model for each hidden state : Gaussian observation with law depending on the state

$$\mathbb{P}(x[n]|s[n] = i) \sim \mathcal{N}(\mu_i, \sigma_i^2)$$

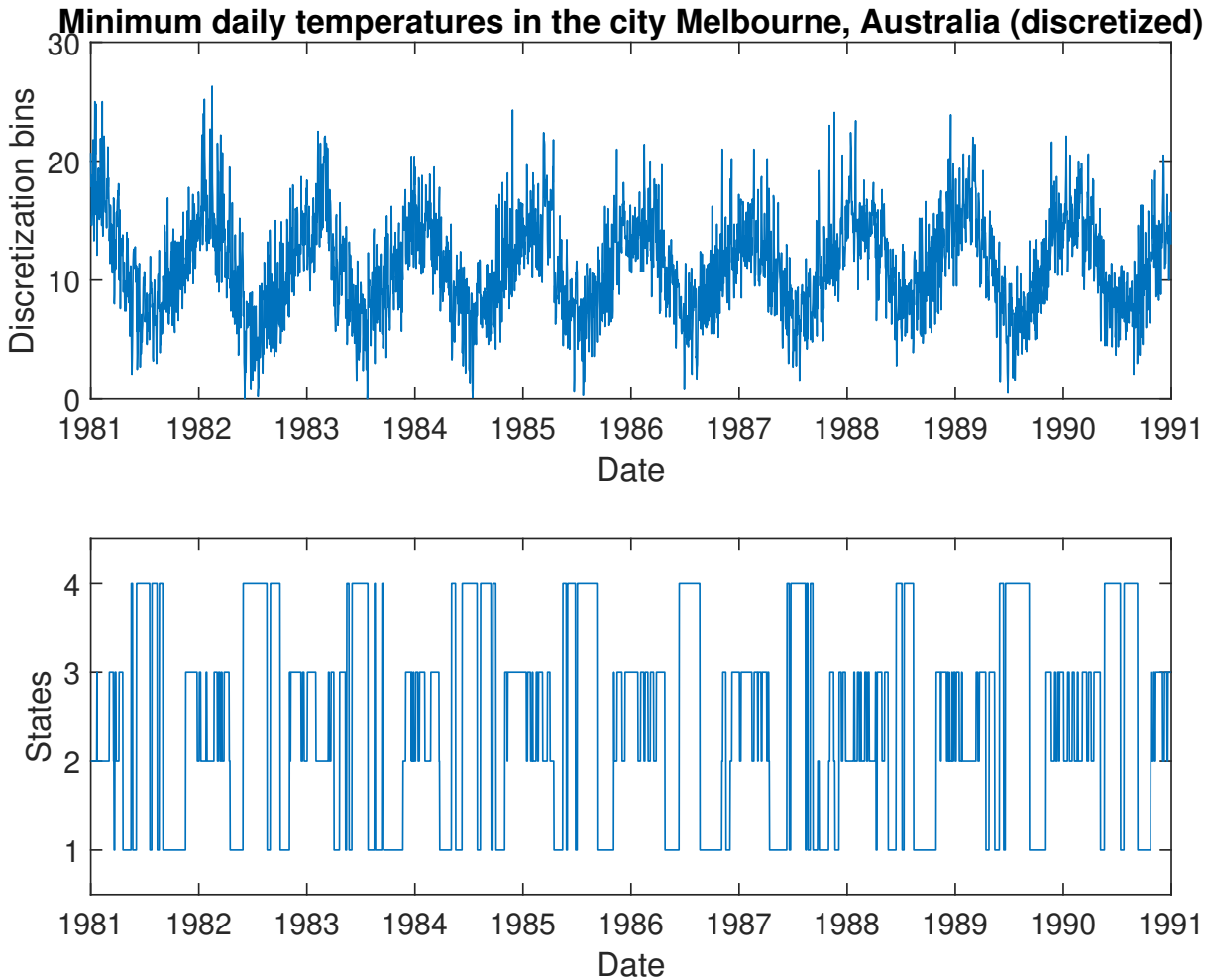
  - Gaussian Mixture model. Same emission models but each observation is assumed to be modeled with a mixture of Gaussians
- HMM can be used for modeling and prediction: huge impact in speech recognition and probabilistic modeling of genomic sequences

Example



Quantization of the time series on $M = 100$ levels

Example



After learning all parameters, recovery of the hidden state sequence with $S = 4$ states

Contents

Notion of representation

- In order to learn from time series, it is necessary to study them in the adequate representation space
- We have seen on Lecture 2 that the frequency domain was especially adapted to study time signals...
- ... but it is not the only one ! In fact, (almost) all previously seen models constitute several possible representations for time series. They all allow to represent a signal as a very condensed list of parameters
- Now, besides these off-the-shelf models, is it possible to **learn** the representation directly from the time series? Answer... YES !

Return to the DFT

- Given a time series $x[n]$ composed of N samples, the discrete Fourier transform (DFT) $X[k]$ writes

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

- Reversely, the time series can be reconstructed from the Fourier coefficients

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

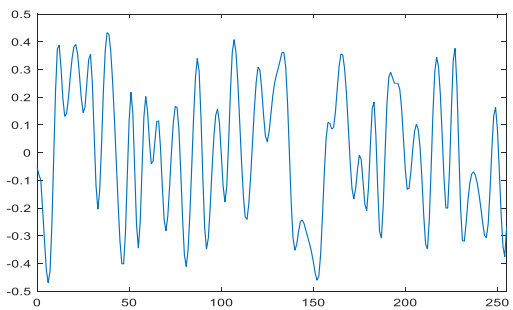
- If the signal $x[n]$ is real, by rewriting

$$a_k = \frac{1}{N} \operatorname{Re}\{X[k]\} \quad \text{and} \quad b_k = -\frac{1}{N} \operatorname{Im}\{X[k]\}$$

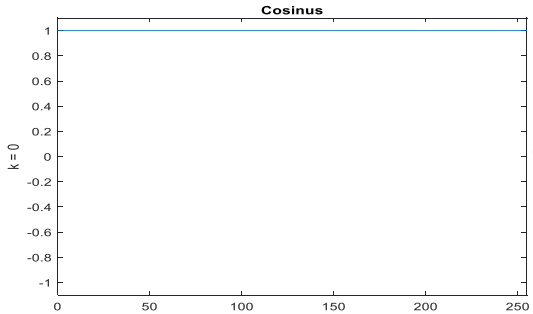
we can write

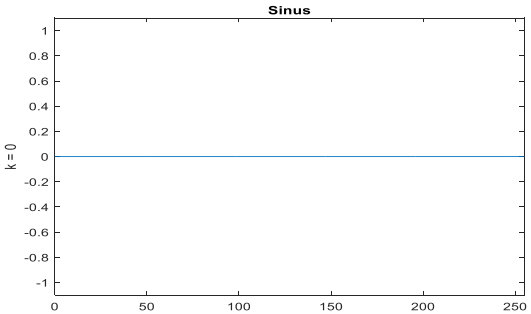
$$x[n] = \sum_{k=0}^{N-1} a_k \cos\left(2\pi \frac{kn}{N}\right) + b_k \sin\left(2\pi \frac{kn}{N}\right)$$

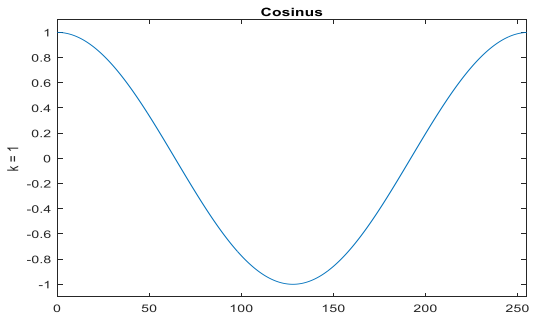
Return to the DFT

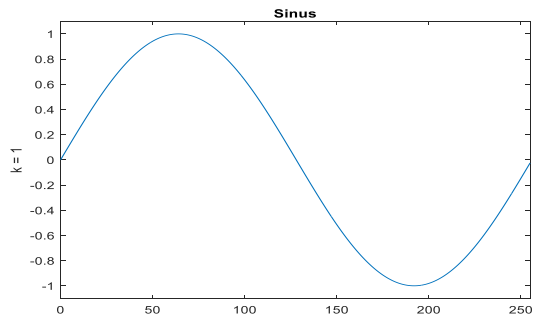


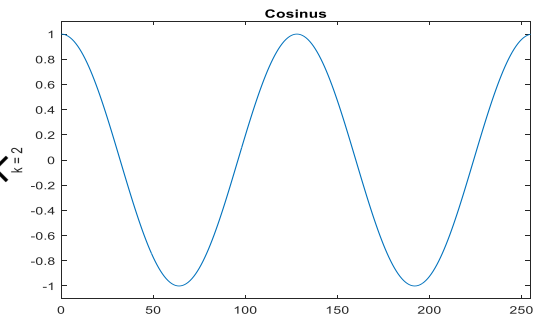
$=$

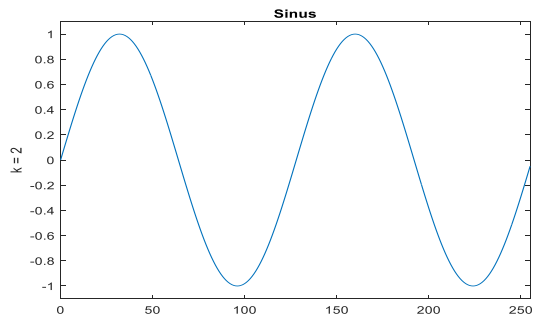
$a_0 \times$ 

$+b_0 \times$ 

$+a_1 \times$ 

$+b_1 \times$ 

$+a_2 \times$ 

$+b_2 \times$ 

$+\ldots$

Other models

- Sinusoidal model

$$x[n] = \sum_{i=1}^{N_h} a_i \sin\left(2\pi i f_0 \frac{n}{F_s} + \phi_i\right)$$

The signal is modeled as the weighted sum of functions $\sin\left(2\pi i f_0 \frac{n}{F_s}\right)$ and $\cos\left(2\pi i f_0 \frac{n}{F_s}\right)$

- Trend+seasonality model

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

Here the functions $\beta_j(nT_s)$ are used for representation

Extension

- As a matter of fact, most of the previously seen models can be cast into an unified framework
- The time series $x[n]$ is represented as a linear combination of several functions, that are stored into a **dictionary**
- K : number of functions in the dictionary
- $\{d_k[n]\}_{1 \leq k \leq K}$: functions in the dictionary (also called **atoms**)

$$x[n] = \sum_{k=1}^{K} z_k d_k[n]$$

Dictionary approach

$$\begin{pmatrix} x[0] \\ x[1] \\ \vdots \\ \vdots \\ \vdots \\ x[N-1] \end{pmatrix} = \begin{pmatrix} d_1[0] & d_2[0] & \cdots & d_K[0] \\ d_1[1] & d_2[1] & \cdots & d_K[1] \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ d_1[N-1] & d_2[N-1] & \cdots & d_K[N-1] \end{pmatrix} \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_K \end{pmatrix}$$

$$\mathbf{x} = \mathbf{D}\mathbf{z}$$

- K : number of functions in the dictionary
- $\mathbf{D}$: dictionary
- $\mathbf{z}$: activation coefficients

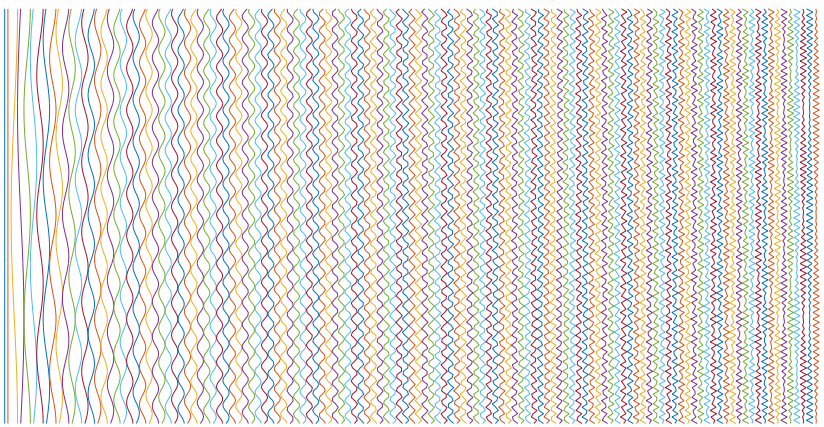
Notion of representation

- Dictionary representation consists in considering a dictionary **D** (known or unknown) and to represent the signal **x** thanks to the atoms
- This task is a simple regression task: but how do we know that we have found a good representation ?
- A good representation consists in finding a good dictionary/activation combination that allow to represent the signal by only using a small subset of all atoms in the dictionary

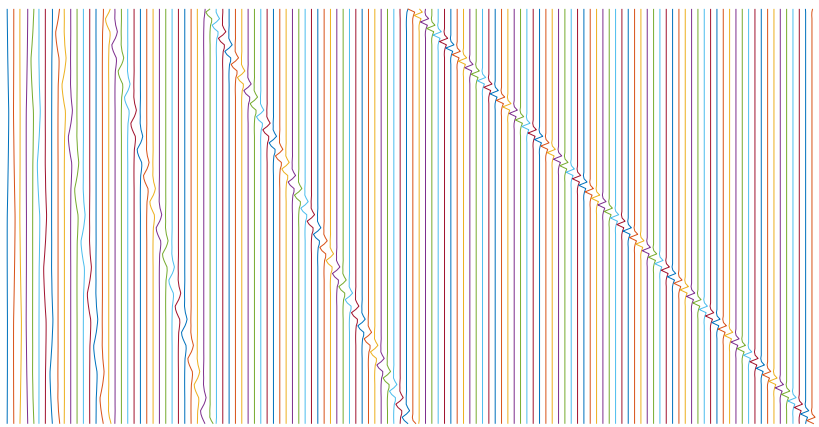
Notion of sparsity

Redundancy in the dictionary

- If we use more than N activations for the reconstruction, then it is not worth considering a dictionary to encode the signals
- Yet, most common dictionaries (Fourier basis, wavelet basis, Gabor atoms...) contains $K \approx N$ or even $K \gg N$ (redundant dictionary)

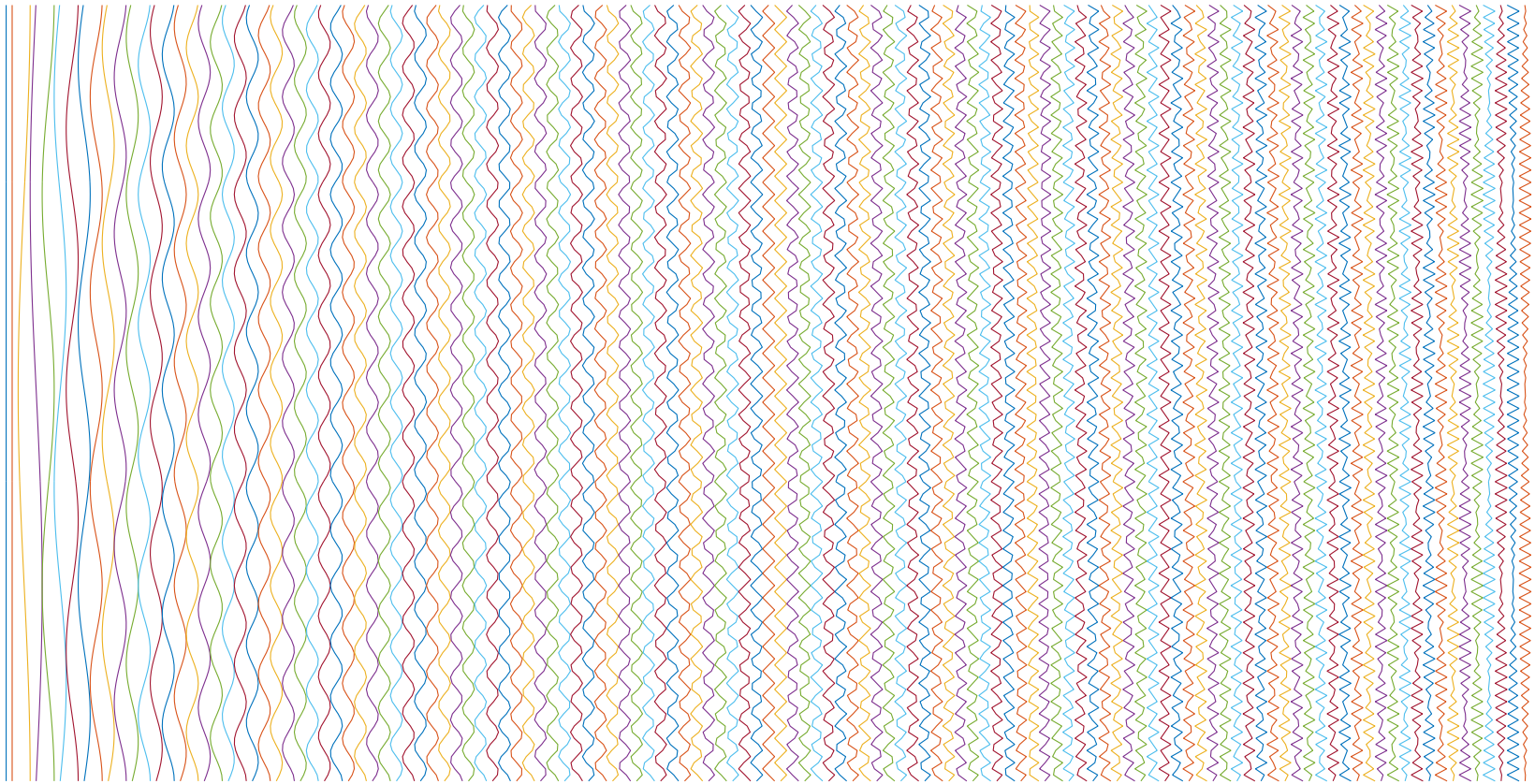


Fourier basis



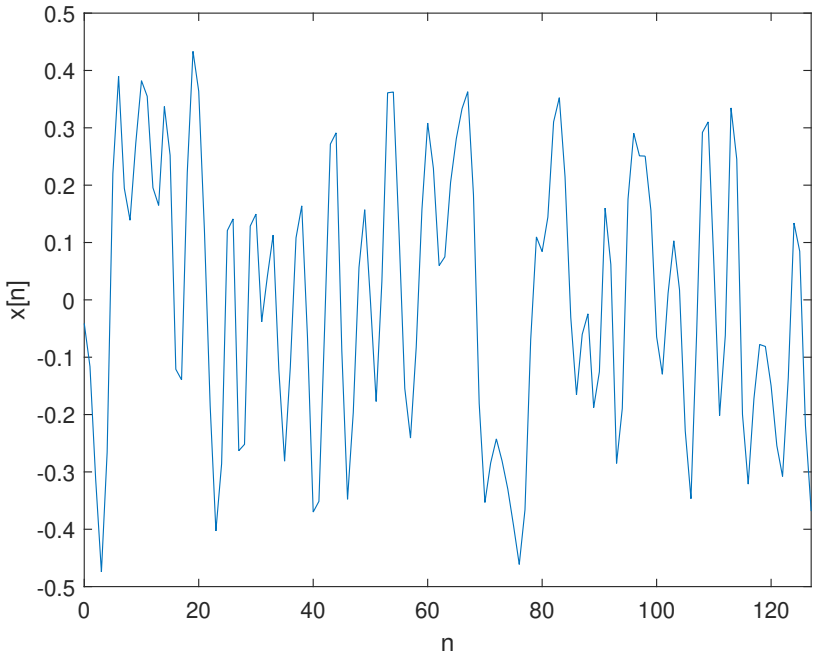
Daubechies db4

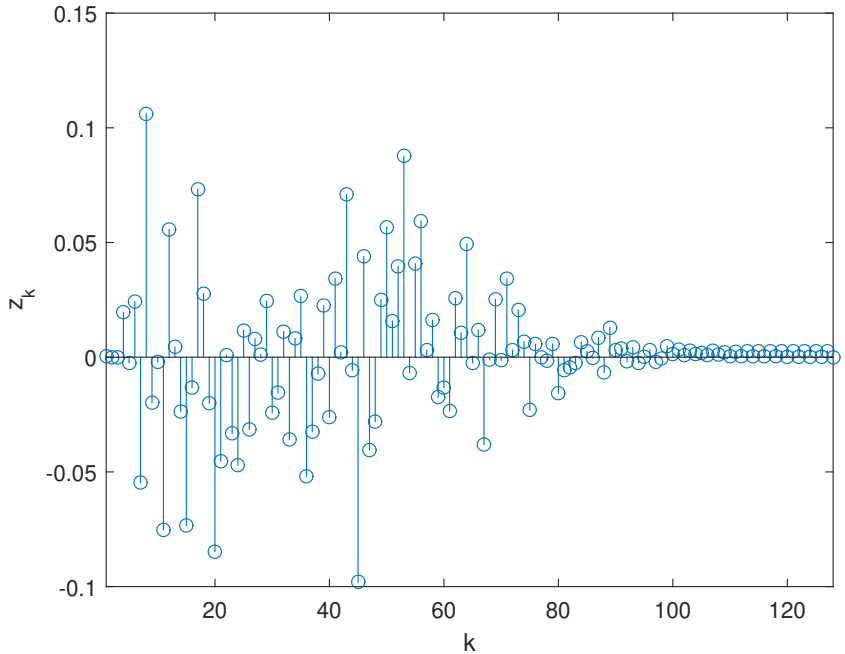
Fourier dictionary



$$N = K = 128$$

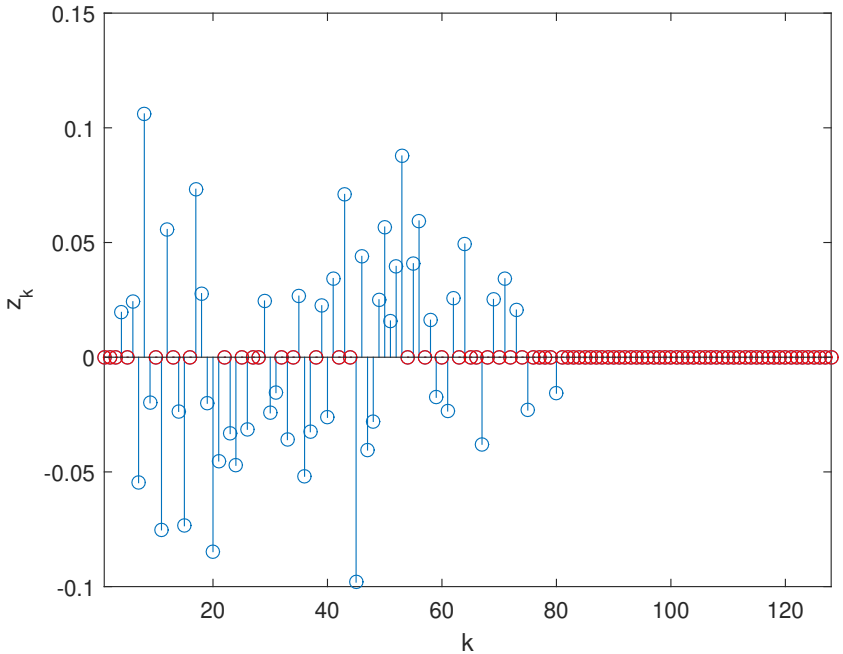
Example

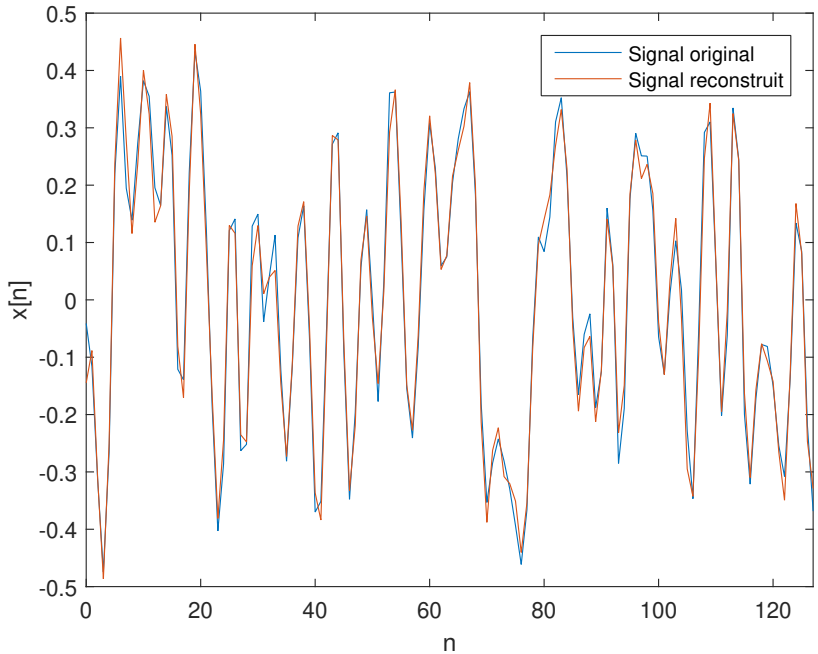




In reality, for most signals a large number of activations z_k are close to 0

Example





Keeping only the 50 largest Fourier coefficients

Notion of sparsity

- In reality, if $K \geq N$, the dictionary is likely to contain redundancy, so a large number of z_k will be close to 0
- Finding a good representation goes back to finding a sparse activation vector
- In the previous example, a signal with 128 samples could be represented by only using 50 values
- Finding a sparse activation vector $\mathbf{z}$ from an input signal $\mathbf{x}$ and a dictionary $\mathbf{D}$ is called **sparse coding**

Sparse coding

There are several formulations for the sparse coding problem:

- $\ell 0$-regularization

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

  - Iterative Hard Thresholding [Blumensath et al., 2008]
  - Matching Pursuit [Mallat et al., 1993]

- $\ell 1$-regularization (also called LASSO : Least Absolute Shrinkage and Selection Operator)

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

  - Iterative Soft Thresholding Algorithm (ISTA) [Daubechies et al., 2004]

$\ell 0$-regularization: Iterative Hard Thresholding

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

Algorithm 2: Iterative Hard Thresholding

Inputs : Time series $\mathbf{x} \in \mathbb{R}^N$
Dictionary $\mathbf{D} \in \mathbb{R}^{N \times K}$
Number of non-null coefficient K_0

Output: Activation vector $\mathbf{z} \in \mathbb{R}^K$ with K_0 non-null entries

$\mathbf{z} = \mathbf{0}_K$;

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

  Gradient descent step;

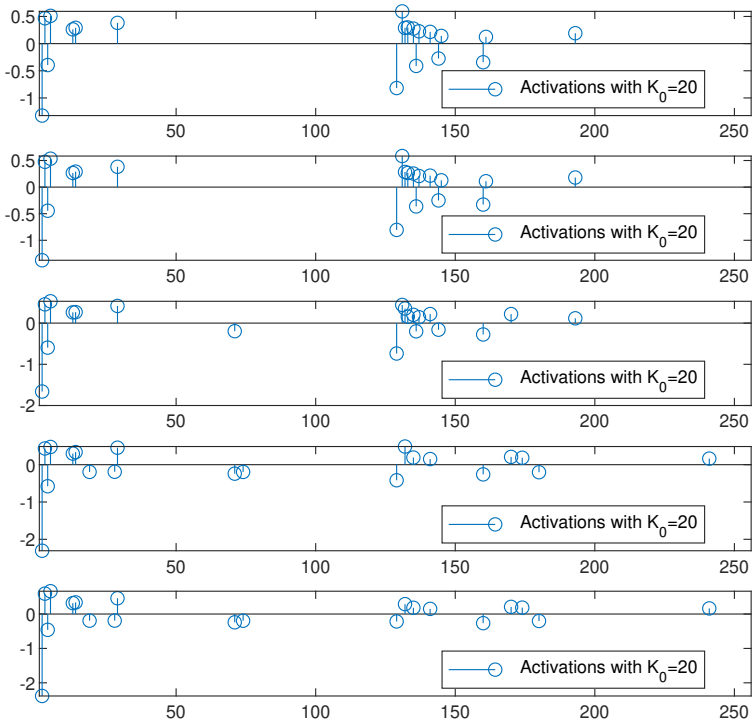
  $\mathbf{z} \leftarrow \mathbf{z} - \mu \mathbf{D}^T (\mathbf{D}\mathbf{z} - \mathbf{x})$;

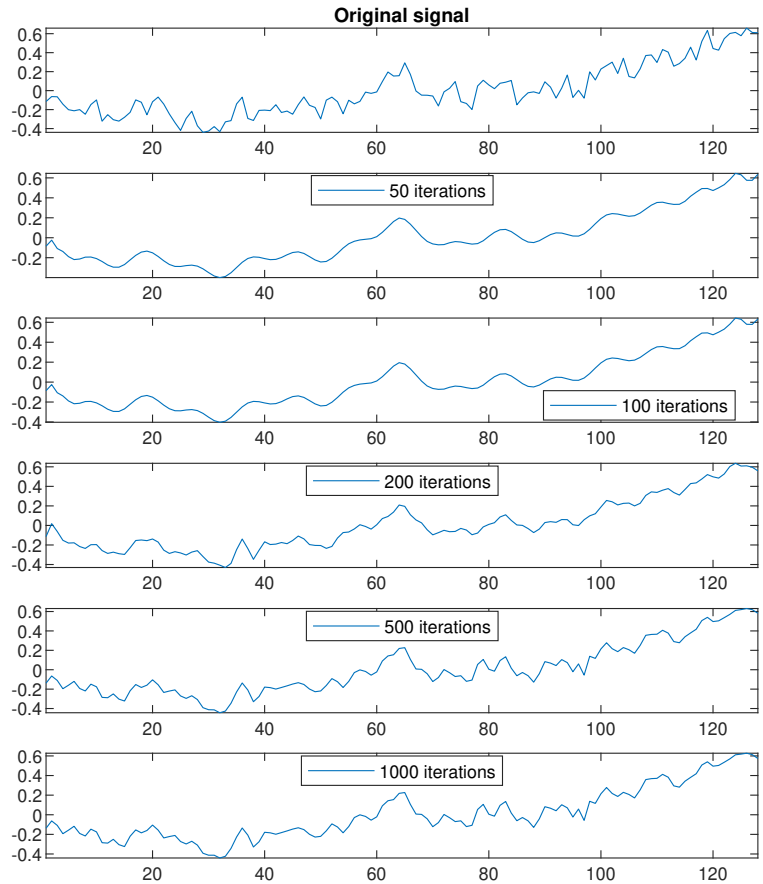
  Hard Thresholding step;

  Projection to only keep the K_0 largest values of vector $\mathbf{z}$;

end

$\ell 0$-regularization: Iterative Hard Thresholding





Dictionary : Fourier + Wavelet (db4 - level 5), $K_0 = 20$

$\ell 0$-regularization: Matching Pursuit

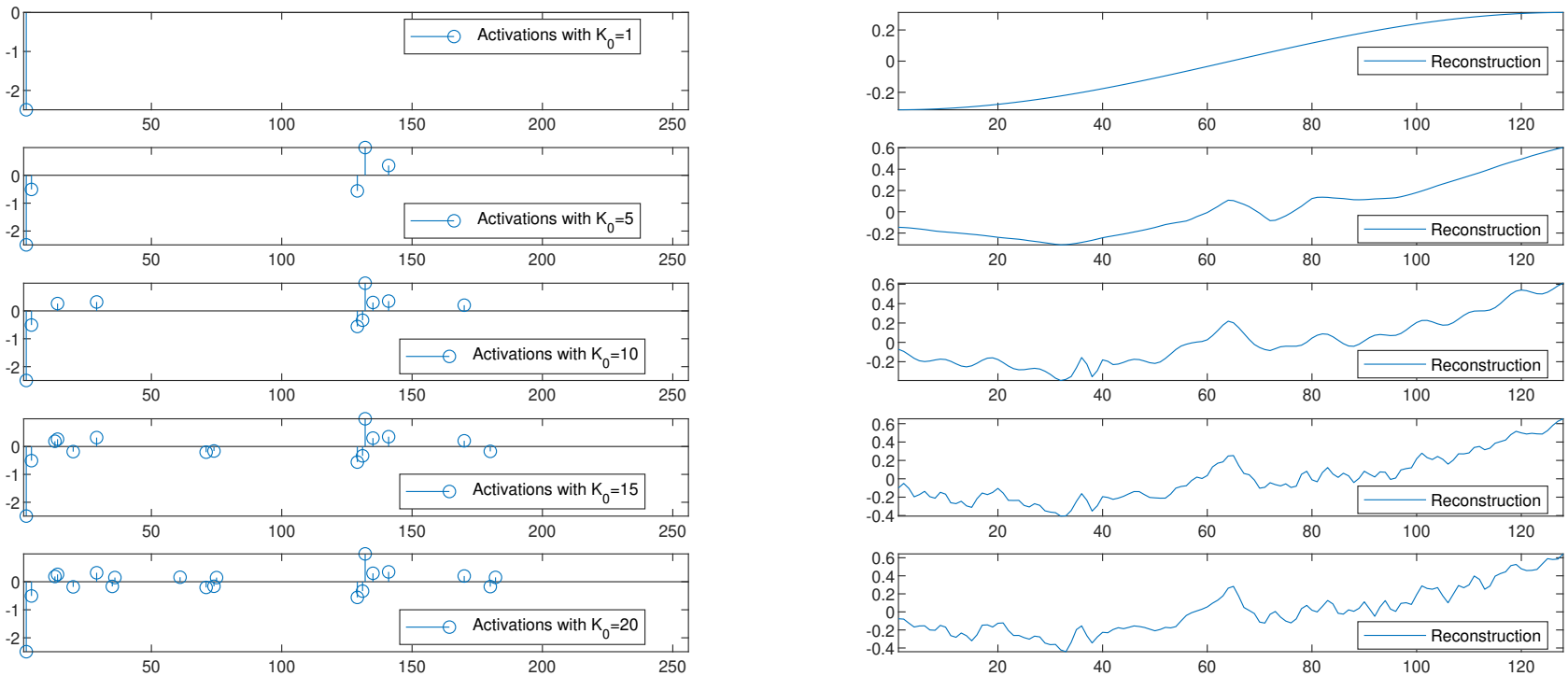
$$\mathbf{z}^* = \underset{\|\mathbf{z}\|_0 = K_0}{\operatorname{argmin}} \|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2$$

Algorithm 3: Matching Pursuit

Inputs : Time series $\mathbf{x} \in \mathbb{R}^N$
Dictionary $\mathbf{D} \in \mathbb{R}^{N \times K}$ with normalized columns
Number of non-null coefficient K_0
Output: Activation vector $\mathbf{z} \in \mathbb{R}^K$ with K_0 non-null entries

$\mathbf{z} = \mathbf{0}_K$;
$\mathbf{r} = \mathbf{x}$;
while $n_{iter} < K_0$ **do**
    Find maximum inner product;
    $k^* = \operatorname{argmax}_k |\langle \mathbf{r}, \mathbf{d}_k \rangle|$;
    Projection step;
    $z_{k^*} = \langle \mathbf{r}, \mathbf{d}_{k^*} \rangle$;
    $\mathbf{r} = \mathbf{r} - z_{k^*} \mathbf{d}_{k^*}$;
end

$\ell 0$-regularization: Matching Pursuit



Dictionary : Fourier + Wavelet (db4 - level 5), $K_0 = 20$

Theoretical guarantees

- **Iterative Hard Thresholding.** If
  - $\|\mathbf{D}\|_F < 1$
  - $\mathbf{D}$ contains a basis for the signal space
  - $\exists c > 0$ such that $\forall k$, $\|\mathbf{d_k}\|_2 > c$ (note that the dictionary is not *a priori* normalized for this algorithm)

  then the algorithm converges to a local minimum. The asymptotic convergence rate is linear [Blumensath et al., 2009]
- **Matching Pursuit.** If
  - $\mathbf{D}$ contains a basis for the signal space

  then $\lim_{\ell \to +\infty} \|\mathbf{r}^{(\ell)}\|_2 \to 0$ (exponentially) [Mallat et al., 1993], i.e. for some $0 < \beta < 1$,

$$\|\mathbf{r}^{(\ell)}\|_2 \leq \beta^{\ell} \|\mathbf{x}\|_2$$

Other approaches for $\ell 0$-regularization

OMP : Orthogonal Matching Pursuit

- At each iteration, all coefficients z_k are updated (instead of only the one corresponding to the currently selected atom)
- New coefficients: orthogonal projection of the signal onto the subspace spanned by the set of atoms selected so far
- Better performances but increased complexity
- Stability and performance guarantees

$\ell 1$-regularization: Iterative Soft Thresholding Algorithm (ISTA)

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

Algorithm 4: Iterative Hard Thresholding Algorithm (ISTA)

Inputs : Time series $\mathbf{x} \in \mathbb{R}^N$
Dictionary $\mathbf{D} \in \mathbb{R}^{N \times K}$
Penalty λ
Output: Sparse activation vector $\mathbf{z} \in \mathbb{R}^K$
$\mathbf{z} = \mathbf{0}_K$;
while $n_{iter} < n_{max}$ **do**
    Gradient descent step;
    $\mathbf{z} \leftarrow \mathbf{z} - \mu \mathbf{D}^T (\mathbf{D}\mathbf{z} - \mathbf{x})$;
    Soft Thresholding step;
    $\mathbf{z} = \mathcal{S}_{\lambda\mu}(\mathbf{z}) = \text{sign}(\mathbf{z}) \times \max(|\mathbf{z}| - \lambda\mu, 0)$;
end

Theoretical guarantees

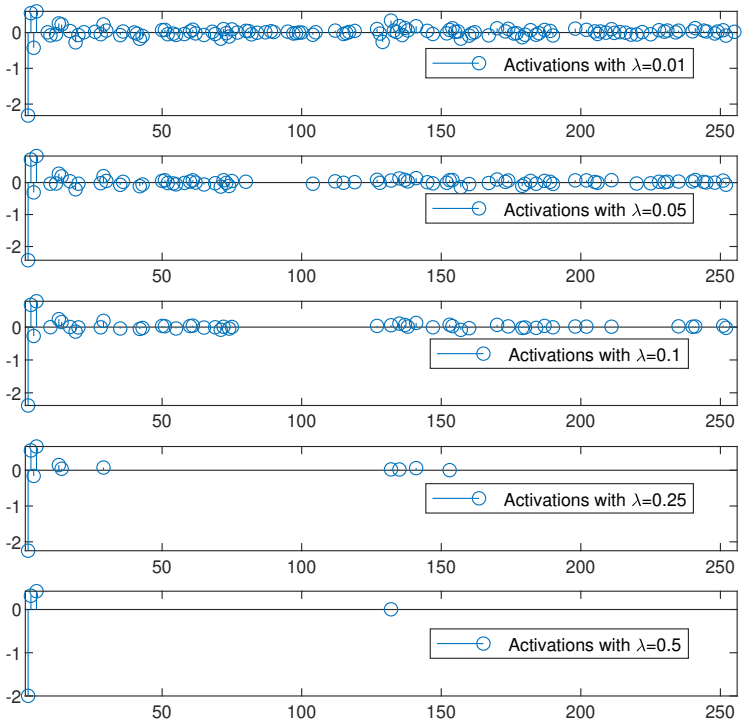
Iterative Hard Thresholding Algorithm (ISTA)

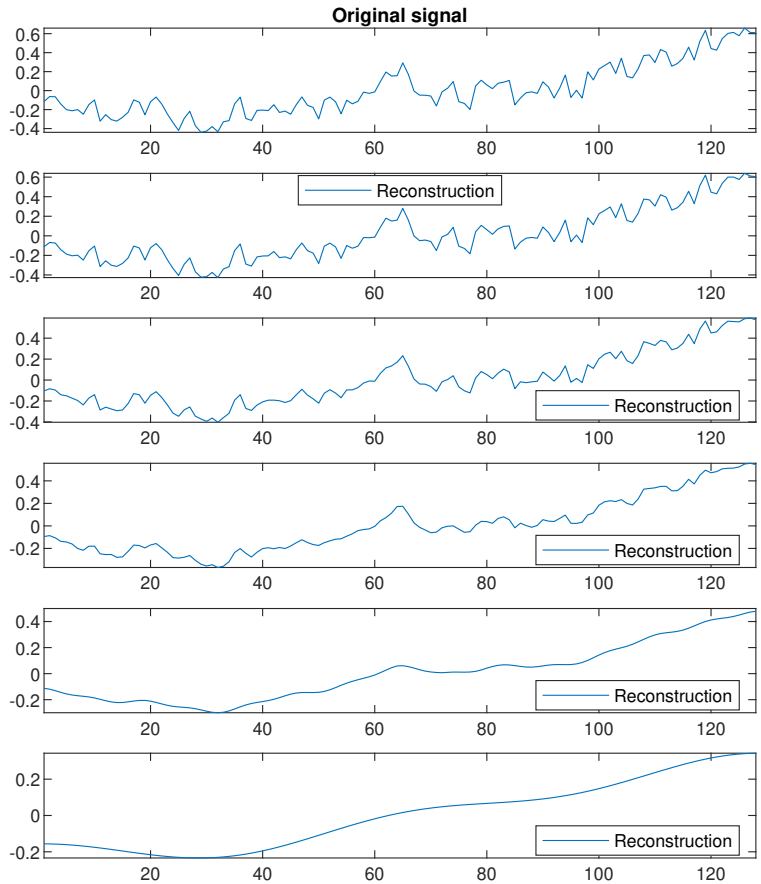
- The sequence of the objective function value converges
- The algorithm converges to a global minimum (the problem is convex)
- By denoting $F(\mathbf{z}) = \|\mathbf{x} - \mathbf{D}\mathbf{z}\|_2^2 + \lambda \|\mathbf{z}\|_1$, we can show that

$$F(\mathbf{z}^{(\ell)}) - F^* \leq \frac{L}{\ell} \|\mathbf{z}^{(0)} - \mathbf{z}^*\|_2^2$$

where L is a constant depending on $F(.)$ and the step size

$\ell 1$-regularization: Iterative Soft Thresholding Algorithm (ISTA)





Dictionary : Fourier + Wavelet (db4 - level 5)

Other approaches for $\ell 1$-regularization

- **FISTA** : Fast Iterative Soft Thresholding Algorithm [Beck et al., 2009]
  Additional step after the soft-thresholding step based on Nesterov momemtum to accelerate gradient descent
- **ADMM** : Alternate Direction Method of Multiplier [Boyd et al., 2011]
  Introduction of auxiliary variables to split the problem into one data term and one penalty term that are optimized independently
- **LARS** : Least angle regression [Efron et al., 2004]
  Iterative addition of activations linked to the atoms that are the most correlated with the residual and increasing of all chosen activations in their joint least squares direction
- **CD** : Coordinate descent
  Update each activation with closed-form solution until convergence

How to choose the dictionary?

- The choice of the dictionary depends on the phenomenon that we seek to retrieve in the signals
- Classical choices (see Lecture 4):
  - **Fourier basis** (or DCT for real signals): link to DFT + most signals are bandlimited + handles the sinusoidal and seasonality component
  - **Polynomial functions:** handle the trend component
  - **Wavelet dictionary:** Haar, Daubechies,... handle multi-scale + localized phenomenon

How to choose the dictionary?

- Would it be possible to automatically learn the dictionary that allow for the best representation of the signal?
- Task of **dictionary learning**: learn the dictionary from the input signals [Tosic et al., 2011]
- Problem: by using only one signal, trivial task by choosing the signal itself as a dictionary!
- Let us consider a dataset of M time series $\mathbf{x_1}, \ldots, \mathbf{x_M}$. These signals will be used to learn $\mathbf{D}$

Notations

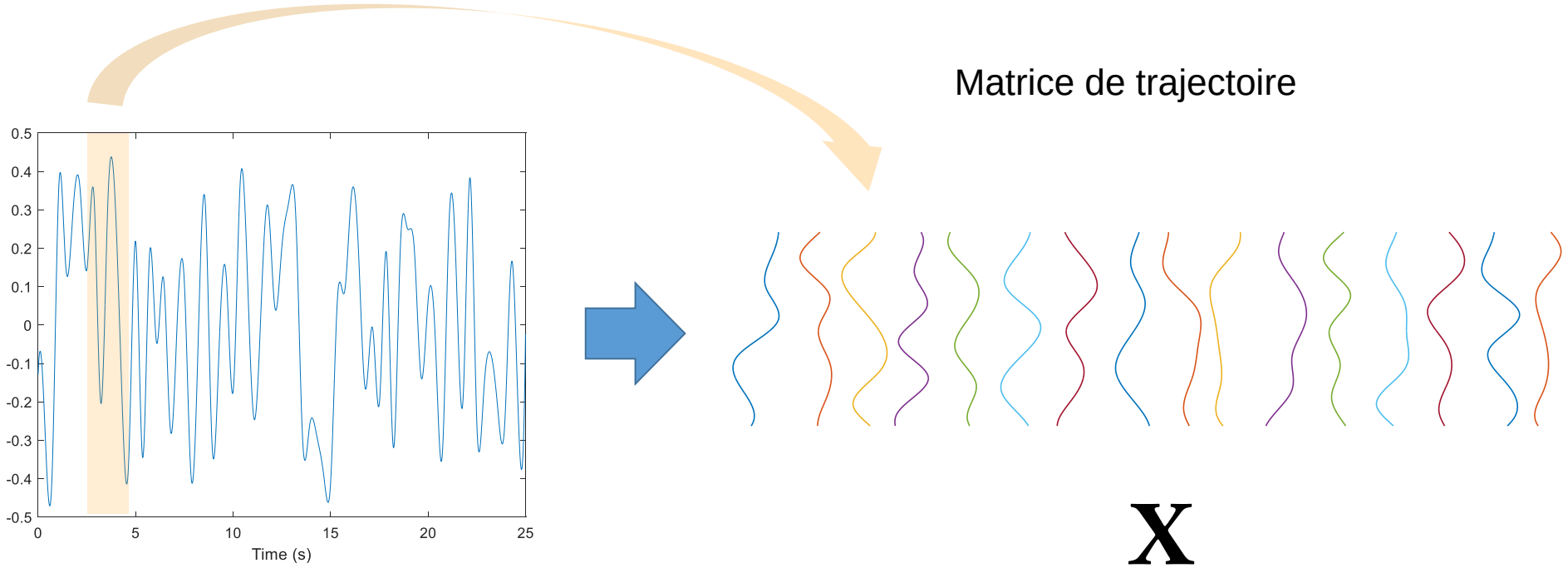
- Input signals is a matrix $\mathbf{X}$ composed of the M signals

$$\mathbf{X} = \begin{pmatrix} x_1[0] & x_2[0] & \cdots & x_M[0] \\ x_1[1] & x_2[1] & \cdots & x_M[1] \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ x_1[N-1] & x_2[N-1] & \cdots & x_M[N-1] \end{pmatrix}$$

- Dictionary $\mathbf{D}$ is still of size $N \times K$, but the activation vector becomes an activation matrix $\mathbf{Z}$ of size $K \times M$

$$\mathbf{D} = \begin{pmatrix} d_1[0] & d_2[0] & \cdots & d_K[0] \\ d_1[1] & d_2[1] & \cdots & d_K[1] \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ d_1[N-1] & d_2[N-1] & \cdots & d_K[N-1] \end{pmatrix} \quad \mathbf{Z} = \begin{pmatrix} z_{1,1} & z_{1,2} & \cdots & z_{1,M} \\ z_{2,1} & z_{2,2} & \cdots & z_{2,M} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ z_{K,1} & z_{K,2} & \cdots & z_{K,M} \end{pmatrix}$$

Example



Matrix **X** can be formed from an unique signal by extracting overlapping frames

Dictionary learning

$$\mathbf{D}^* = \underset{\forall k, \|\mathbf{d_k}\|_2 \leq 1}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{DZ}\|_F^2$$

- Normalization constraint to prevent $\mathbf{D}$ from being arbitrarily large
- Convex problem with fixed $\mathbf{X}$ and $\mathbf{Z}$
- Several solvers: Proximal Gradient Descent (see next slides), block coordinate descent, K-Singular Value Decomposition (K-SVD), ADMM... but also online dictionary learning method that process data on the fly [Mairal et al., 2009]

Dictionary learning: Proximal Gradient Descent (PGD)

$$\mathbf{D}^* = \underset{\forall k, \|\mathbf{d_k}\|_2 \leq 1}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{DZ}\|_F^2$$

Algorithm 5: Projected Gradient Descent (PGD)

Inputs : Set of time series $\mathbf{X} \in \mathbb{R}^{N \times M}$
Sparse activation matrix $\mathbf{Z} \in \mathbb{R}^{K \times M}$
Output: Dictionary $\mathbf{D} \in \mathbb{R}^{N \times K}$
Initialize $\mathbf{D}$ with normalized columns ;
while $n_{iter} < n_{max}$ **do**
    Gradient descent step;
    $\mathbf{D} \leftarrow \mathbf{D} - \mu \left(\mathbf{DZ} - \mathbf{X}\right) \mathbf{Z}^T$;
    Projection step;
    $\mathbf{d}_k \leftarrow \text{proj}_{\|.\|_2^2 \leq 1}(\mathbf{d}_k) = \frac{\mathbf{d}_k}{\max(\|\mathbf{d}_k\|_2^2, 1)}$;
end

Alternated resolution

- In the general context the activations are unknown and we need to solve the global problem

$$(\mathbf{D}^*, \mathbf{Z}^*) = \underset{\substack{\forall k, \|\mathbf{d_k}\|_2 \leq 1 \\ \forall k, \mathbf{z}_k \text{ is sparse}}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{DZ}\|_F^2$$

- This problem cannot be solved as such: in most cases we alternate between two subproblems
  - Sparse coding with known $\mathbf{D}$
  - Dictionary learning with known $\mathbf{Z}$
- Alternated resolution until the norm of the residual becomes small enough

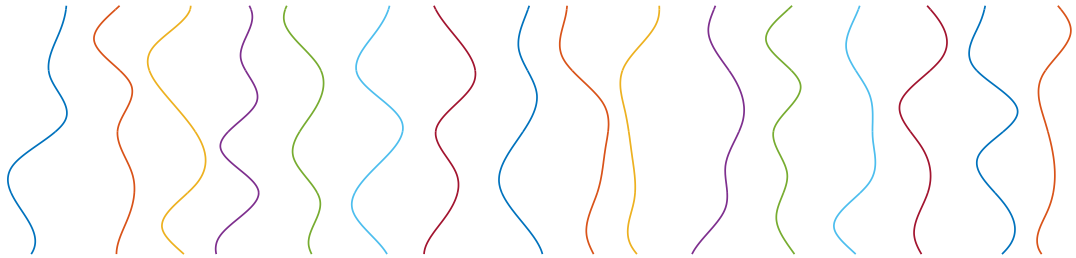
Alternated resolution for dictionary learning/sparse coding

$$(\mathbf{D}^*, \mathbf{Z}^*) = \underset{\substack{\forall k, \|\mathbf{d_k}\|_2 \leq 1 \\ \forall k, \mathbf{z}_k \text{ is sparse}}}{\operatorname{argmin}} \|\mathbf{X} - \mathbf{DZ}\|_F^2$$

Algorithm 6: Alternated resolution for dictionary learning/sparse coding

Inputs : Set of time series $\mathbf{X} \in \mathbb{R}^{N \times M}$
Number of atoms K
Sparsity constraints *sp_cst* (K_0 for $\ell 0$, λ for $\ell 1$)
Outputs: Dictionary $\mathbf{D} \in \mathbb{R}^{N \times K}$
Sparse activation matrix $\mathbf{Z} \in \mathbb{R}^{K \times M}$
Initialize $\mathbf{D}$ with normalized columns ;
while $n_{iter} < n_{max}$ **do**
    Sparse coding step;
    $\mathbf{Z} \leftarrow$ sparse_coding($\mathbf{X}, \mathbf{D}$, *sp_cst*);
    Dictionary learning step;
    $\mathbf{D} \leftarrow$ dictionary_learning($\mathbf{X}, \mathbf{D}, \mathbf{Z}$);
end

Example



d_1 

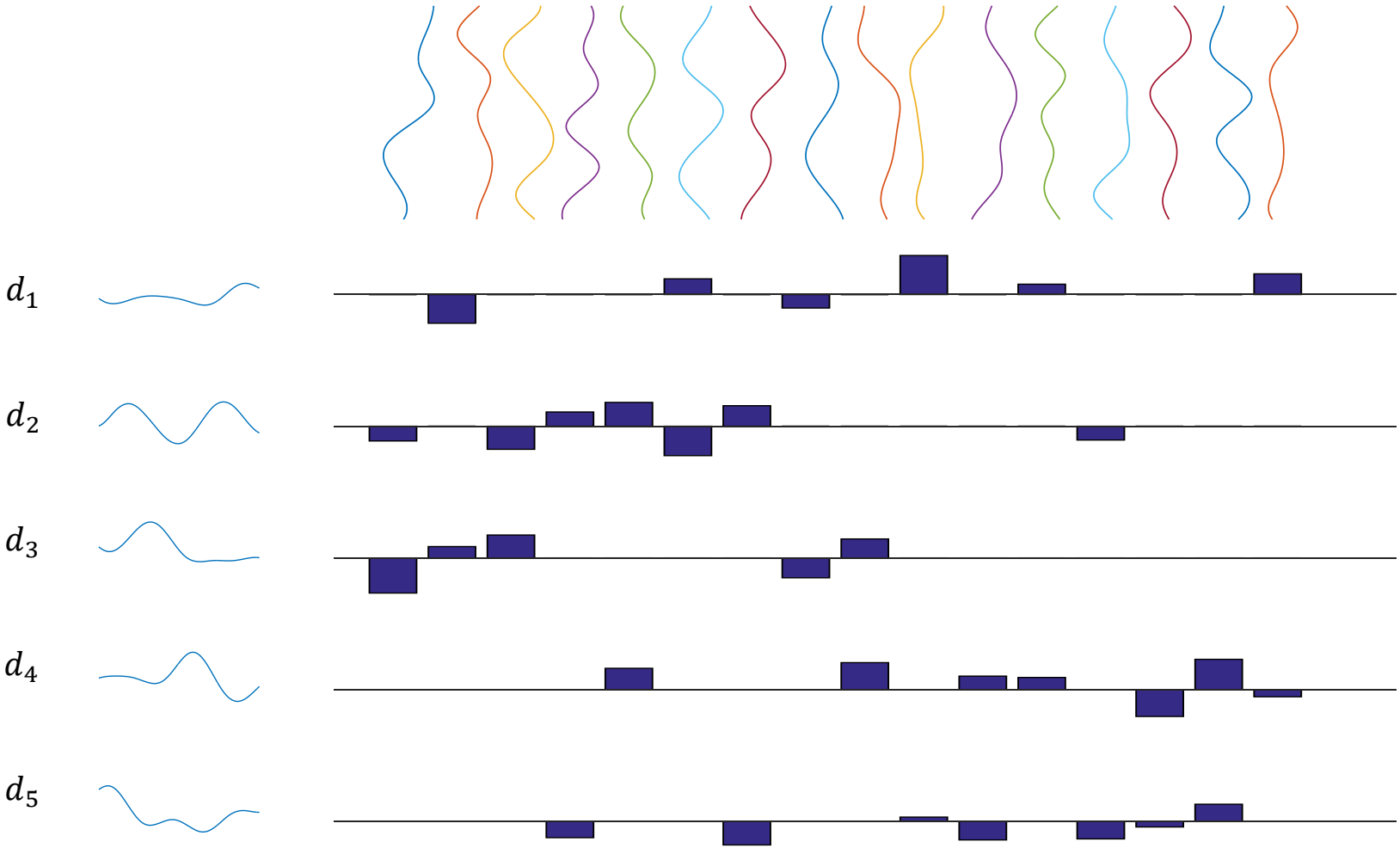
d_2 

d_3 

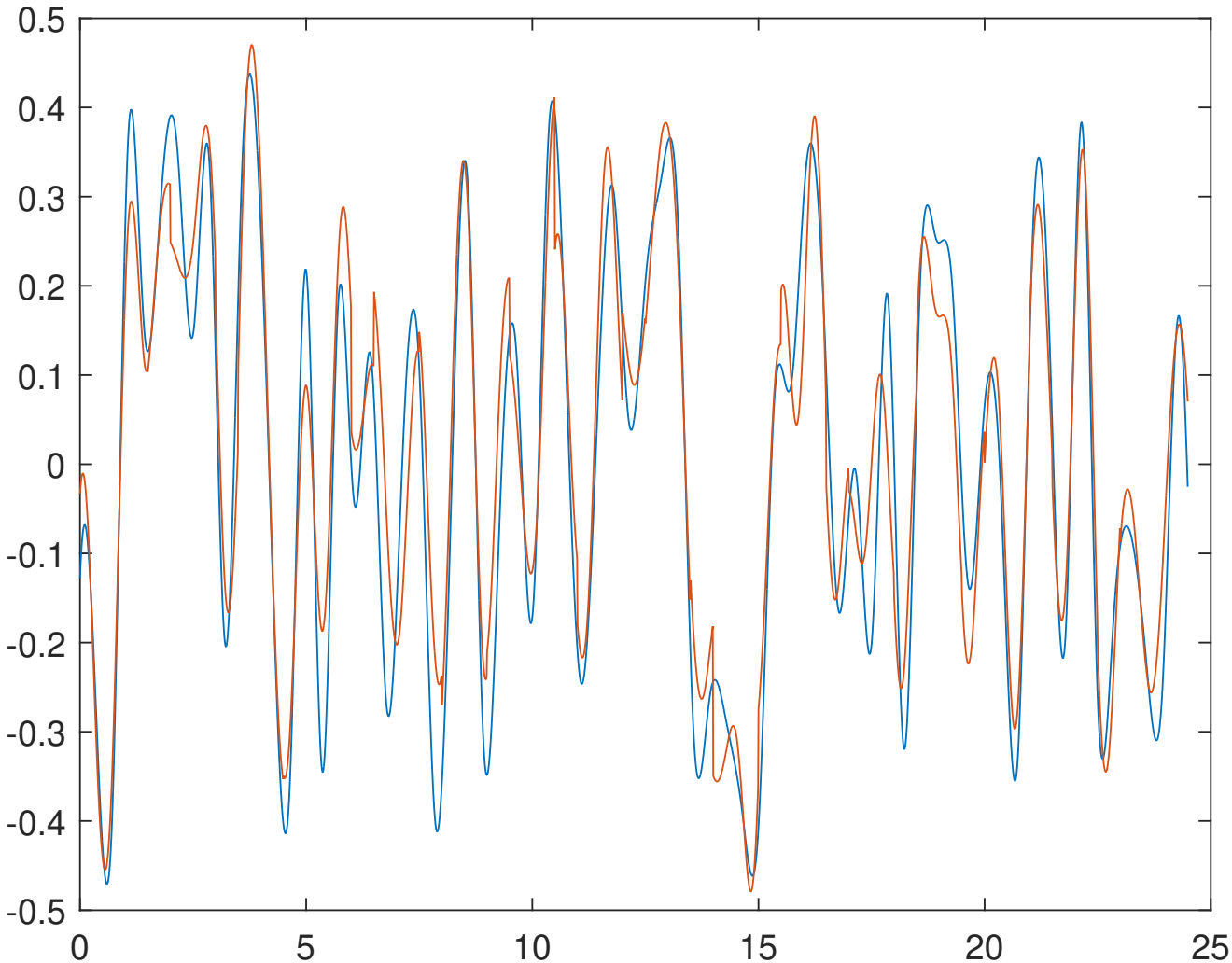
d_4 

d_5 

Example



Example

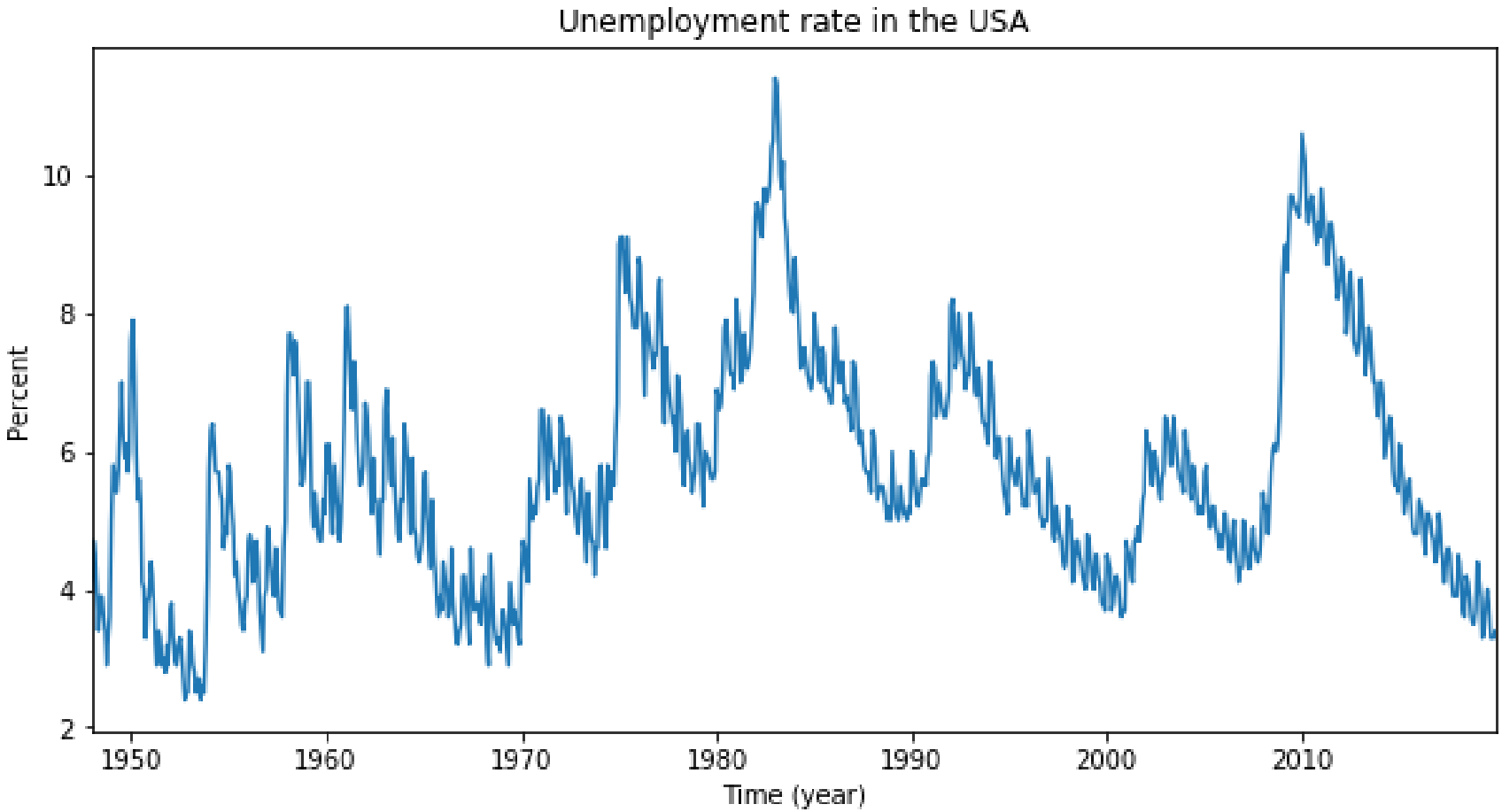


Blue : original signal, red : sparse reconstruction

How to use dictionary learning for ML?

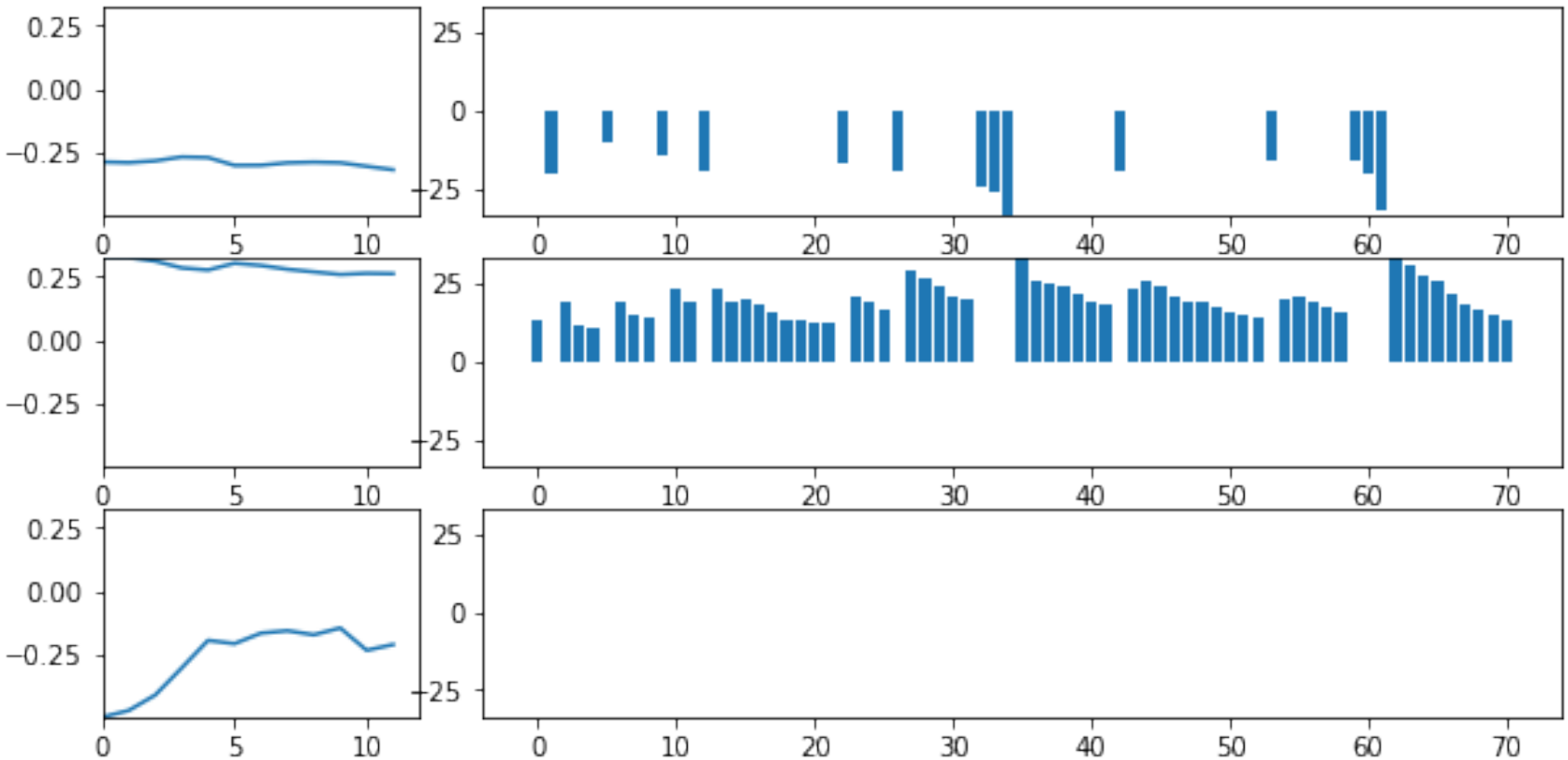
- By learning a common dictionary for a set of time series, the activations can be used for clustering/classification
- For classification, it is also possible to learn a dictionary for each class. The input signal can be classified by finding the dictionary corresponding to the sparsest representation.
- The sparse representation can also be used for denoising and data interpolation (see Lecture 4)
- By isolating certain atoms, dictionary learning can also be used for source separation

Example



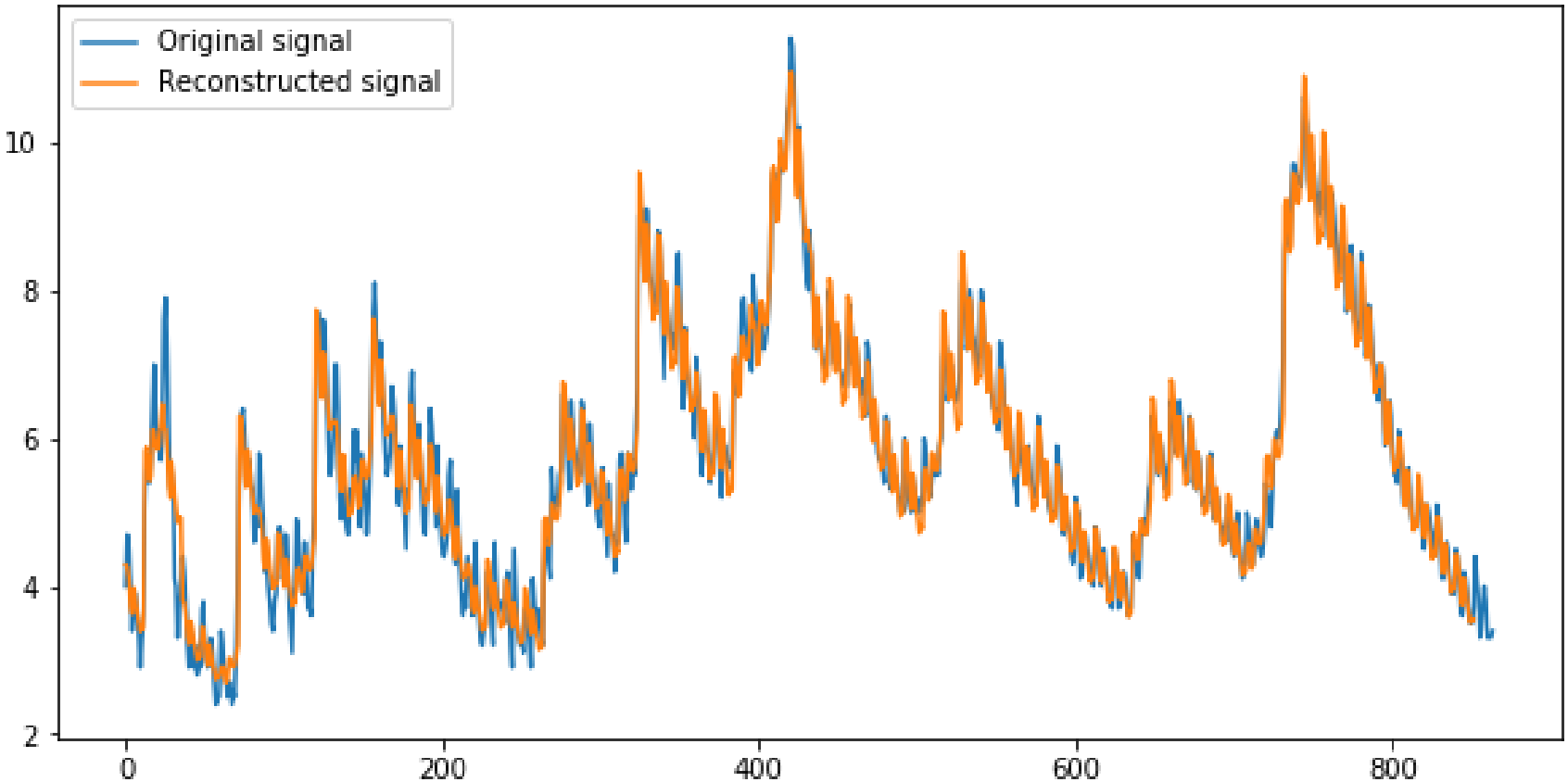
72 years, 1 sample per month: transformation into 72 non-overlapping frames of 12 frames

Example



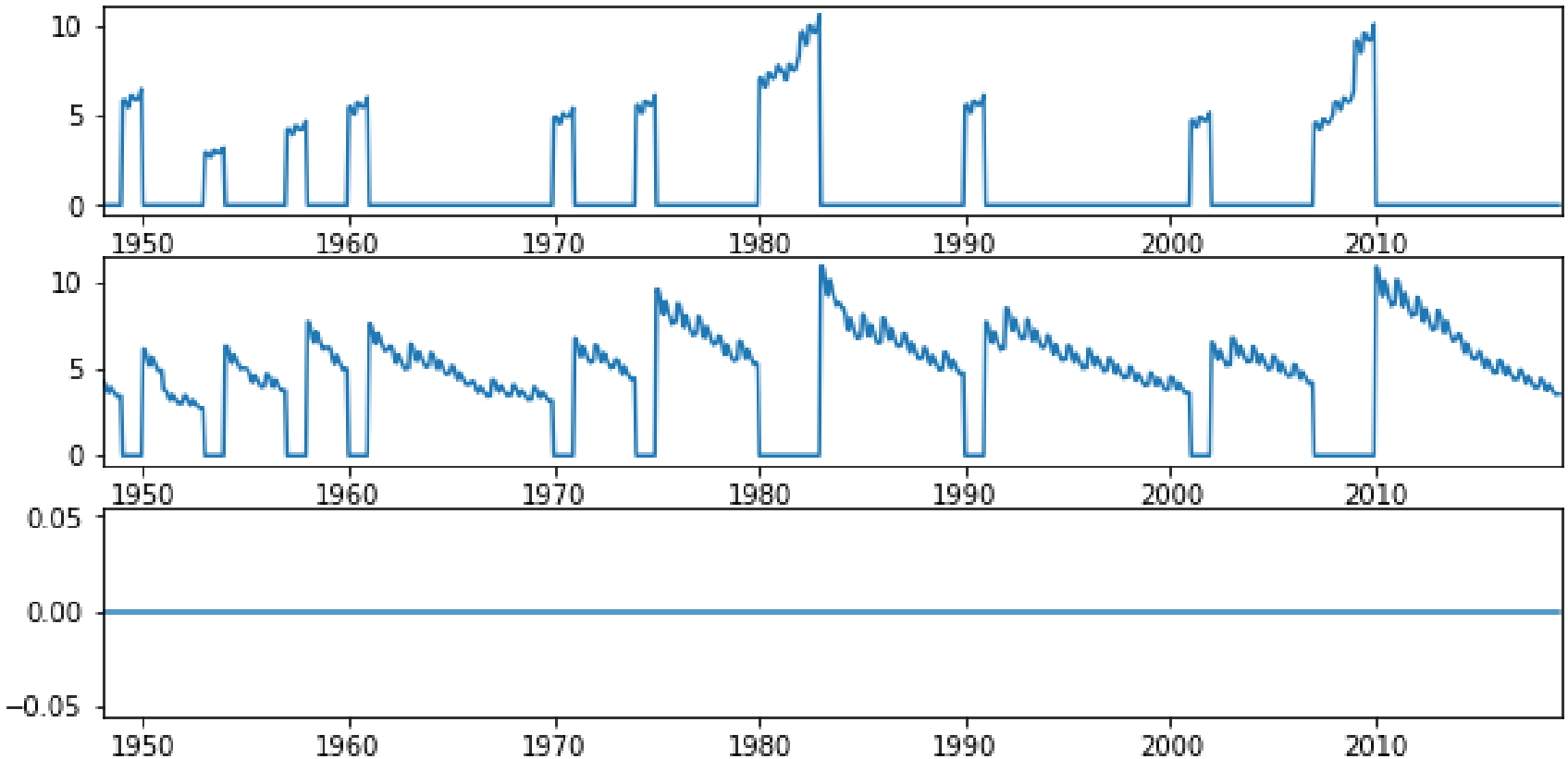
$K = 3$ atoms, $\ell 0$- sparse coding with $K_0 = 1$

Example



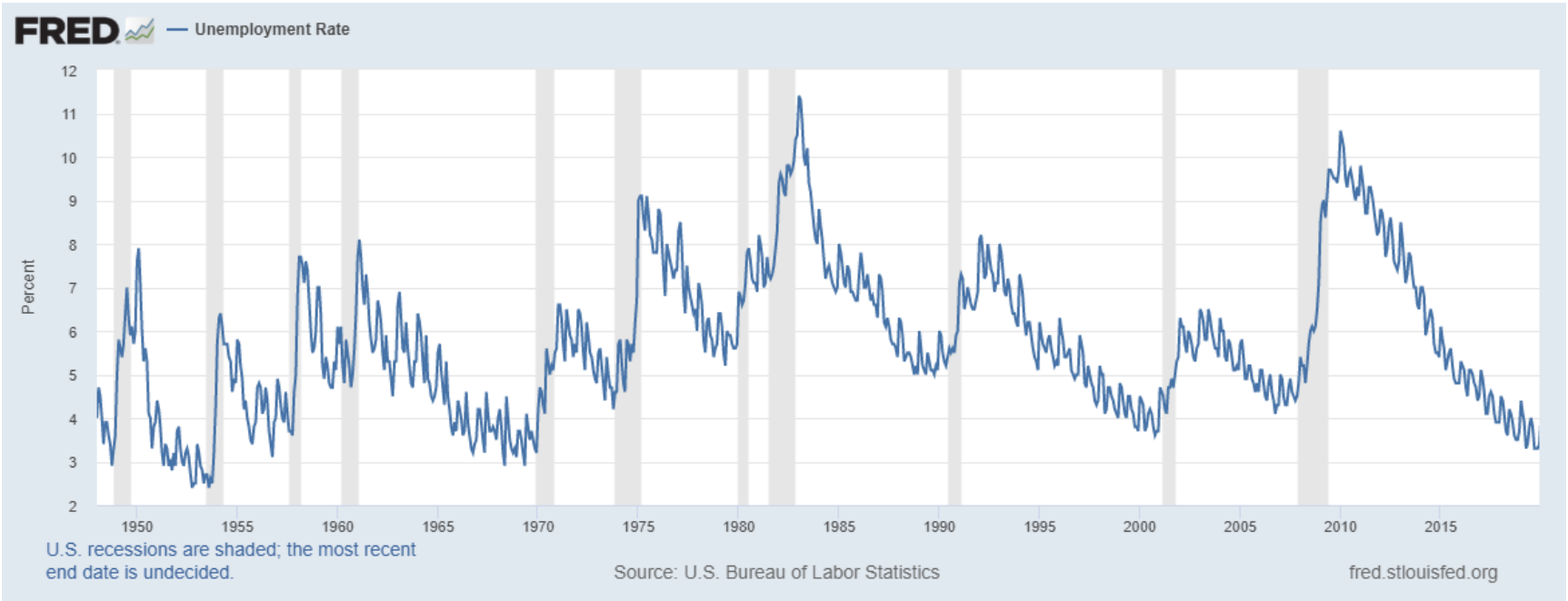
Reconstruction by using all 3 atoms

Example



Reconstruction by using each atom individually

Example



All recession periods are captured by the first atom

References

- Box, G. E., Jenkins, G. M., & Reinsel, G. C. (2011). Time series analysis: forecasting and control (Vol. 734). John Wiley & Sons.
- Hamilton, J. D. (1994). Time series analysis. Princeton University Press.
- Priestley, M. B. (1981). Spectral analysis and time series: probability and mathematical statistics (No. 04; QA280, P7.).
- Pankratz, A. (2009). Forecasting with univariate Box-Jenkins models: Concepts and cases (Vol. 224). John Wiley & Sons.
- Durbin, J. (1960). The fitting of time-series models. Revue de l'Institut International de Statistique, 233-244.
- Bisgaard, S., & Kulahci, M. (2011). Time series analysis and forecasting by example. John Wiley & Sons.
- Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2), 257-286.
- Bilmes, J. A. (1998). A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. International Computer Science Institute, 4(510), 126.
- Blumensath, T., & Davies, M. E. (2008). Iterative thresholding for sparse approximations. Journal of Fourier analysis and Applications, 14(5-6), 629-654.
- Mallat, S. G., & Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. IEEE Transactions on signal processing, 41(12), 3397-3415.
- Daubechies, I., Defrise, M., & De Mol, C. (2004). An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences, 57(11), 1413-1457.
- Beck, A., & Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM journal on imaging sciences, 2(1), 183-202.
- Boyd, S., Parikh, N., & Chu, E. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc.
- Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. The Annals of statistics, 32(2), 407-499.
- Tosic, I., & Frossard, P. (2011). Dictionary learning. IEEE Signal Processing Magazine, 28(2), 27-38.
- Mairal, J., Bach, F., Ponce, J., & Sapiro, G. (2009, June). Online dictionary learning for sparse coding. In Proceedings of the 26th annual international conference on machine learning (pp. 689-696).

List of possible topics/projects

- Mairal, J., Bach, F., Ponce, J., & Sapiro, G. (2009, June). Online dictionary learning for sparse coding. In Proceedings of the 26th annual international conference on machine learning (pp. 689-696).
  How to learn a dictionary from streaming data
- Tzagkarakis, G., Caicedo-Llano, J., & Dionysopoulos, T. (2015). Sparse modeling of volatile financial time series via low-dimensional patterns over learned dictionaries. Algorithmic Finance, 4(3-4), 139-158.
  How to model financial data with sparse dictionary representations.
- Ho, S. L., Xie, M., & Goh, T. N. (2002). A comparative study of neural network and Box-Jenkins ARIMA modeling in time series prediction. Computers & Industrial Engineering, 42(2-4), 371-375.
  How to compare deep learning and standard Box-Jenkins models for prediction
- Yazdi, S. V., & Douzal-Chouakria, A. (2018). Time warp invariant kSVD: Sparse coding and dictionary learning for time series under time warp. Pattern Recognition Letters, 112, 1-8.
  How to mix Dynamic Time Warping and dictionary learning
- Lyu, H., Strohmeier, C., Menz, G., & Needell, D. (2020). COVID-19 time-series prediction by joint dictionary learning and online NMF. arXiv preprint arXiv:2004.09112.
  How to use matrix factorization and dictionary learning to perform prediction
- Zhang, W., Wang, Z., Yuan, J., & Hao, S. (2020). Discriminative Dictionary Learning for Time Series Classification. IEEE Access, 8, 185032-185044.
  How to combine symbolic representation and dictionary learning for time series classification
- Varoquaux, G., Gramfort, A., Pedregosa, F., Michel, V., & Thirion, B. (2011, July). Multi-subject dictionary learning to segment an atlas of brain spontaneous activity. In Biennial International Conference on information processing in medical imaging (pp. 562-573). Springer, Berlin, Heidelberg.
  How to use dictionary learning for segmentation
- Sawada, H., Ono, N., Kameoka, H., Kitamura, D., & Saruwatari, H. (2019). A review of blind source separation methods: two converging routes to ILRMA originating from ICA and NMF. APSIPA Transactions on Signal and Information Processing, 8.
  How to use dictionary learning for source separation